Réseaux de neurones informés par la physique pour la résolution d'équations de mécanique des fluides

par

Mathieu MULLINS

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE, ÉNERGIES RENOUVELABLES ET
EFFICACITÉ ÉNERGÉTIQUE
M. Sc. A.

MONTRÉAL, LE 22 AVRIL 2025

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Azzeddine Soulaïmani, directeur de mémoire
Département de génie mécanique à l'École de technologie supérieure

M. Marco Pedersoli, président du jury
Département de génie des systèmes à l'École de technologie supérieure

M. Emmanuel Lorin, examinateur externe
School of Mathematics and Statistics à l'Université Carleton

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 1er AVRIL 2025

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens tout d'abord à exprimer ma profonde gratitude envers mon directeur de recherche, Azzeddine Soulaïmani, dont le soutien et la confiance ont rendu ce travail possible. Azzeddine m'a offert l'opportunité de contribuer au développement d'un récent nouvel axe de recherche au sein de son groupe GRANIT, à l'ÉTS : l'application de l'intelligence artificielle à la mécanique des milieux continus. Bien que je disposais de peu d'expérience en programmation au début de ce projet, il m'a donné les moyens d'acquérir des compétences précieuses dans les deux domaines du génie qui me passionnent le plus, soit l'intelligence artificielle et la mécanique des fluides.

J'aimerais remercier Hamza Kamil, doctorant au sein du groupe de recherche GRANIT à l'ÉTS, pour sa patience et son soutien inestimable. Hamza a joué un rôle clé dans l'implémentation de mon code, m'aidant à surmonter rapidement les défis de programmation auxquels j'ai été confronté.

Je remercie Adil Fahsi, professeur à l'Université Sultan Moulay Slimane au Maroc pour m'avoir fourni le code servant à simuler les exemples de références pour nos modèles level set.

Je remercie également Compute Canada et Calcul Québec pour l'accès aux ressources de calcul, sans lesquelles ce travail n'aurait pas pu voir le jour. Leurs services de support ont été essentiels, et leurs réponses rapides et précises ont grandement facilité le déroulement de ce projet.

J'exprime ma gratitude aux professeurs Marco Pedersoli et Emmanuel Lorin pour leur relecture attentive et leur évaluation de ce travail.

Enfin, je tiens tout particulièrement à remercier ma famille, dont le soutien constant et inestimable m'a permis de poursuivre mes études supérieures à l'ÉTS.

Réseaux de neurones informés par la physique pour la résolution d'équations de mécanique des fluides

Mathieu MULLINS

RÉSUMÉ

Ce mémoire explore l'application des réseaux de neurones informés par la physique (PINNs) aux problèmes d'interface mobile via la méthode level set ainsi qu'aux écoulements à surface libre régis par les équations de Saint-Venant. Plus spécifiquement, nous mettons en avant les performances de l'architecture PirateNet, qui intègre des améliorations telles que l'apprentissage séquence-par-séquence, la factorisation aléatoire des poids, l'équilibrage des termes de la fonction de perte par la norme des gradients, l'entraînement causal et l'incorporation de caractéristiques de Fourier aléatoires.

Pour les problèmes d'interface mobile, le PINN amélioré a démontré une capacité supérieure à résoudre le disque de Zalesak et un cas complexe impliquant de fortes déformations, l'écoulement tourbillonnaire inversé, atteignant une erreur $L^2 = 0.81\%$ pour ce dernier test. Contrairement aux méthodes numériques classiques, les PINNs peuvent capturer l'évolution de l'interface sans stabilisation numérique en amont ni réinitialisation géométrique. De plus, bien que l'ajout d'un terme de régularisation basé sur l'équation d'Eikonal puisse améliorer les résultats sous certaines conditions, il doit être soigneusement pondéré pour éviter des effets négatifs.

Concernant les équations de Saint-Venant, le PINN amélioré a surpassé le PINN classique en prédisant l'évolution de la surface libre, y compris en présence de bathymétrie variable et d'interférences entre vagues, atteignant une erreur $L_h^2 = 0.30\%$ pour le cas de propagation d'une vague sur une digue en 1D tout en maintenant une perte de masse très faible et une excellente capture de la vitesse de propagation de l'onde. Cependant, bien que la hauteur d'eau puisse être bien prédite, les PINNs éprouvent des difficultés à modéliser précisément les champs de vitesses dans les régions présentant de forts gradients. L'utilisation de viscosité numérique permet de réduire l'erreur, mais entraîne un coût informatique supérieur, surtout en 2D.

Un constat important de cette étude est que, malgré leurs performances prometteuses, les PINNs demeurent significativement plus coûteux en calcul que les méthodes numériques classiques. Si la phase d'inférence est rapide, l'entraînement du modèle nécessite un temps considérable, rendant leur application limitée pour des problèmes directs d'équations différentielles partielles.

En résumé, ce mémoire démontre que le PINN amélioré avec les méthodes recensés constitue une avancée significative pour l'application des PINNs aux problèmes d'interface mobile et aux écoulements à surface libre. Bien que ces approches nécessitent encore des améliorations pour rivaliser pleinement avec les méthodes numériques traditionnelles, elles offrent une alternative intéressante pour des contextes où l'intégration de données réelles est possible.

Mots-clés: PINN, SWE, Saint-Venant, CFD, Level set

Physics-Informed Neural Networks for Solving Continuum Mechanics Equations

Mathieu MULLINS

ABSTRACT

This thesis explores the application of Physics-Informed Neural Networks (PINNs) to moving interface problems using the level set method, as well as to free surface flows governed by the Shallow Water Equations (SWE). Specifically, we highlight the performance of the PirateNet architecture, which integrates improvements such as sequence-to-sequence training, random weight factorization, gradient-norm-based loss term balancing, causal training, and the incorporation of random Fourier features.

For moving interface problems, the enhanced PINN demonstrated superior capabilities in solving Zalesak's disk and a complex case involving significant interface deformation, the time-reversed vortex flow, achieving an error of $L^2 = 0.81\%$ in the latter. Unlike classical numerical methods, PINNs can capture interface evolution without the need for upwind numerical stabilization or geometric reinitialization. Furthermore, although the addition of an Eikonal-based regularization term may improve results under certain conditions, it must be carefully weighted to avoid adverse effects.

Regarding the Shallow Water Equations, the enhanced PINN outperformed the standard PINN by accurately predicting free surface evolution, including in the presence of variable bathymetry and interfering waves. An error of $L_h^2 = 0.30\%$ was achieved for the 1D wave propagation over a hump test case, while maintaining very low mass loss and accurately capturing the wave speed. However, despite reliable height prediction, PINNs struggle to model velocity fields accurately in regions with strong gradients. Numerical viscosity can help reduce this error, but it significantly increases computational cost, especially in 2D.

A key finding of this study is that, despite their promising performance, PINNs remain substantially more computationally expensive than traditional numerical methods. While inference is fast, the initial model training requires significant time, limiting their practicality for direct PDE problems.

In summary, this thesis demonstrates that the enhanced PINN equipped with the discussed improvements represents a meaningful advancement in applying PINNs to moving interface problems and free surface flows. While further development is needed to fully match the precision of classical numerical methods, PINNs offer an appealing alternative in contexts where real data can be integrated to refine predictions and improve physical modeling.

Keywords: PINN, SWE, Saint-Venant, CFD, Level set

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

CFD	Dynamique des fluides numérique
CPU	Central processing pnit
EDP	Équation différentielle partielle
ÉTS	École de technologie supérieure
FEM	Méthode des éléments finis
GPU	Graphical processing unit
GRANIT	Groupe de recherche sur les applications numériques en ingénierie et en technologie
IA	Intelligence artificielle
MAPE	Erreur absolue moyenne en pourcentage
MLP	Perceptron multicouche
MMLP	Perceptron multicouche modifié
PINN	Réseaux de neurones informés par la physique
RAD	Distribution adaptative basée sur les résiduels
RBA	Masque d'attention résiduel
RWF	Factorisation aléatoire des poids
SDF	Signed distance function
SWE	Équations en eaux peu profondes, Équations de Saint-Venant
S2S	Apprentissage séquence par séquence

LISTE DES SYMBOLES ET UNITÉS DE MESURE

Ω	Domaine de calcul
t	Temps
x	Position sur l'axe x
y	Position sur l'axe y
u	Vitesse selon l'axe x
v	Vitesse selon l'axe y
b	Bathymétrie
h	Hauteur d'eau
s	Hauteur de la surface libre
g	Accélération gravitationnelle
ρ	Masse volumique
ν	Viscosité cinématique
n	Nombre de manning
S_f	Terme source de friction
S_O	Terme source de bathymétrie
	Indices :
t	Dérivée par rapport à t
x	Dérivée par rapport à x
y	Dérivée par rapport à y

INTRODUCTION

Selon le rapport 1 "The Human cost of weather related disasters 1995-2015 " produit en commun par l'UNISDR (United Nations Office for Disaster Risk Reduction) et le CRED (Centre for Research on the Epidemology of Disasters), entre 1995 et 2015, les inondations ont représenté à elles seules 47 % des catastrophes liées aux conditions météorologiques, affectant 2.3 milliards de personnes CRED & UNDRR (2015). De plus, ces événements sont de plus en plus fréquents avec 335 désastres climatiques reportés annuellement entre 2005 et 2014, ce qui est presque deux fois plus élevé que la fréquence reportée de 1985 à 1994. Ces chiffres illustrent l'augmentation continue des risques pour les populations humaine, mais aussi pour nos infrastructures et l'économie avec des dommages économiques annuels estimés de 250 à 300 milliards US.

Les impacts des catastrophes naturelles liées à l'eau, telles que les inondations, les ruptures de barrages et les crues soudaines, représentent donc un défi majeur de notre société actuelle. L'archipel de Montréal, en particulier, fait face à des inondations récurrentes qui soulignent l'urgence de développer des outils prédictifs précis pour anticiper les impacts de ces phénomènes et élaborer des stratégies de gestion des risques. Suite aux crues printanières records de 2017 et 2019, le gouvernement québécois a d'ailleurs mandaté la Communauté Métropolitaine de Montréal (CMM) pour développer une cartographie des zones inondables CMM (2019) et un réseau de stations de mesure couplée à une plateforme de prévision capable d'anticiper les niveaux d'eau jusqu'à trois jours à l'avance CMM (2025).

De tels outils existent grâce aux simulations numériques qui jouent un rôle crucial en permettant de modéliser les phénomènes physiques liés aux écoulements d'eau à surface libre. Elles offrent une alternative à la réalisation d'expériences sur des maquettes physiques, souvent coûteuses et limitées en échelle. Parmi les méthodes disponibles, les équations de Saint-Venant (1871), aussi appelées Shallow-Water Equations (SWE), constituent le modèle mathématique de référence pour modéliser des écoulements à surface libre. Malgré que les équations datent

du 19^{e} siècle, le développement de méthodes numériques pouvant résoudre ces équations a été un domaine de recherche très actif durant les dernières décennies avec plusieurs codes développés dans Audusse, Bouchut, Bristeau, Klein & Perthame (2004); Bradford & Sanders (2002); Brufau, García-Navarro & Vázquez-Cendón (2004); Loukili & Soulaimani (2007); Zokagoa & Soulaïmani (2010); Delmas & Soulaïmani (2022).

Les SWE utilisent l'approximation que l'accélération verticale est nulle. Cela fonctionne bien pour les écoulements à surface libres standards, mais moins bien pour les écoulements non linéaires complexes avec de forts gradients verticaux ou des chocs. C'est là que d'autres méthodes deviennent plus utiles comme les méthodes level set. Celles-ci permettent de suivre avec précision l'évolution d'interfaces mouvantes, un élément clé dans des problèmes où des interfaces complexes et dynamiques doivent être modélisées, par exemple lors de ruptures de barrages ou de phénomènes tourbillonnaires.

Toutefois, ces simulations reposent traditionnellement sur des méthodes numériques classiques qui, bien que robustes, sont gourmandes en ressources de calcul, notamment lorsqu'il s'agit de modéliser des phénomènes à grande échelle ou à long terme. Cela limite leur applicabilité à des scénarios complexes et les rend inadéquats pour de la prévision en temps réel. Face à ces contraintes, de nouvelles approches émergent, exploitant les avancées en intelligence artificielle et, plus spécifiquement, les réseaux de neurones informés par la physique (PINN). Ces méthodes combinent la puissance des techniques d'apprentissage profond avec les lois fondamentales de la physique pour offrir des solutions précises et efficaces, réduisant considérablement les coûts d'inférences une fois que les modèles sont entraînés.

Ainsi, l'étude des PINN couplés avec les SWE et des méthodes Level Set offre un potentiel immense pour répondre aux défis de simulations posés par les changements climatiques et les catastrophes naturelles. Elle permet non seulement de mieux comprendre et prévoir ces

phénomènes, mais aussi de développer des outils stratégiques pour la gestion des risques et la protection des populations.

On propose dans la suite de ce chapitre d'exposer les besoins de recherche sur les PINN.

0.1 Limites des simulations numériques classiques

Les simulations numériques dans leur état actuelles présentent des limites, soit la dépendance sur un maillage et les ressources de calculs intensives. Ces limites sont détaillées dans cette section.

Traditionnellement, la dynamique des fluides numérique (CFD) propose deux approches principales pour résoudre numériquement les problèmes d'écoulement multiphasique : les méthodes lagrangiennes et eulériennes. Les méthodes lagrangiennes comme la méthode des éléments finis suivent l'interface à l'aide d'un maillage qui se déplace et s'adapte à l'écoulement, ce qui permet des simulations très précises de l'évolution de l'interface. Cependant, cette approche devient difficile à mettre en œuvre dans les problèmes tridimensionnels (3D), en particulier lorsqu'il s'agit d'interfaces complexes ou très déformables. En revanche, les méthodes eulériennes, y compris les techniques telles que le suivi du front et la capture du front, utilisent un maillage de calcul fixe qui reste stationnaire tout au long de la simulation. Les approches eulériennes s'adaptent mieux aux géométries complexes et aux grandes déformations. Cependant, ces approches nécessitent une résolution de maillage fin et des changements significatifs dans les formulations de discrétisation pour garantir la précision près de l'interface.

Il reste néanmoins que ces deux approches ont une chose en commun, elles nécessitent toutes deux une discrétisation du domaine en un maillage. Cela veut dire que la solution est calculée à des points discrets dans le domaine. Une interpolation est donc nécessaire si nous souhaitons avoir les résultats entre ces points. Cependant, cette interpolation n'obéit pas intrinsèquement aux lois physiques de l'équation différentielle partielle (EDP) sous-jacente. Cela dit, une discrétisation

adéquate du domaine est primordiale dans l'atteinte de résultats de simulations satisfaisants. Le processus de discrétisation implique donc souvent plusieurs itérations et inspections manuelles qui dépendent de l'expérience de l'utilisateur.

Par ailleurs, les méthodes numériques classiques présentent une forte demande en ressources de calcul, en grande partie due à la nécessité d'utiliser des maillages très fins pour obtenir des résultats précis. Pour les problèmes à grande échelle ou ceux comportant des phénomènes complexes, le nombre de points de calcul augmente exponentiellement, ce qui entraîne une croissance significative des coûts de calcul et des temps de simulation. Dans certains cas, les simulations doivent être effectuées sur des superordinateurs ou des grappes de calcul, mobilisant des ressources informatiques coûteuses et énergivores à chaque fois qu'un calcul doit être effectué.

Cette exigence rend difficile l'application des méthodes classiques à des scénarios où des résultats rapides sont requis, comme les simulations en temps réel. Par exemple, dans le cadre de la prévention des inondations ou des ruptures de barrages, il est souvent nécessaire de prévoir rapidement l'évolution des niveaux d'eau ou des interfaces fluides pour guider des interventions d'urgence. Les méthodes classiques peinent à répondre à ces exigences en raison de leurs longues durées de calcul, ce qui limite leur applicabilité pratique dans de tels contextes.

0.2 Pourquoi utiliser les PINN

Les réseaux de neurones informés par la physique (PINNs) s'imposent comme une alternative prometteuse aux méthodes numériques classiques et aux approches d'intelligence artificielle traditionnelles pour la résolution de problèmes en mécanique des fluides, comme les méthodes data-driven. Des méthodes telles que les LSTM, les TCN, ou encore l'architecture plus récente du Convolutional Autoencoder Bhatt, Kumar & Soulaimani (2023), ont été explorées pour modéliser des phénomènes physiques. Toutefois, ces approches rencontrent des limites, notamment lorsqu'il

s'agit de prédire des comportements hors du domaine d'apprentissage. La conception unique des PINNs offre plusieurs avantages clés qui répondent directement aux limites des techniques existantes.

0.2.1 Intégration des lois physiques

Contrairement aux modèles d'apprentissage profond standards, les PINNs intègrent explicitement les lois physiques sous-jacentes au problème, comme les équations de Navier-Stokes ou les équations de Saint-Venant. Cela est rendu possible en ajoutant les EDP sous forme de termes de pénalité dans la fonction de perte du réseau.

Cette intégration garantit que les prédictions du modèle respectent les lois de conservation de la masse, de la quantité de mouvement ou de toute autre loi pertinente au problème étudié. Cela offre ainsi une précision accrue et une fiabilité physique, même dans des situations où les données empiriques sont rares ou absentes.

0.2.2 Apprentissage semi supervisé

Les méthodes classiques d'intelligence artificielle dépendent souvent de vastes ensembles de données étiquetées, dont la collecte et le traitement peuvent s'avérer extrêmement coûteux, voire impraticables pour certains phénomènes physiques. Les PINNs, en revanche, nécessitent uniquement des informations essentielles telles que les conditions aux limites, les conditions initiales et les équations gouvernantes du problème étudié. Cette approche d'apprentissage non supervisé permet de se passer de jeux de données exhaustifs tout en respectant les contraintes physiques inhérentes au système modélisé.

0.2.3 Capacité d'extrapoler hors du domaine d'apprentissage

L'un des principaux avantages des PINNs est leur capacité à généraliser au-delà du domaine d'apprentissage. Cette propriété repose sur le fait que le réseau est formé non seulement sur des données mais aussi sur des lois physiques fondamentales, qui restent valides hors des échelles explorées par les données.

Cela permet aux PINNs de prédire des comportements physiques dans des régions non échantillonnées ou sous des conditions différentes de celles rencontrées lors de l'entraînement. En revanche, les approches traditionnelles d'apprentissage profond sont habituellement peu performantes lorsque confrontées à des scénarios à l'extérieur du domaine d'apprentissage.

0.2.4 Approche sans maillage

Les PINNs se distinguent par leur approche sans maillage, offrant une solution aux limitations des méthodes numériques classiques qui reposent sur une discrétisation fine du domaine.

Contrairement aux méthodes traditionnelles qui résolvent les EDP à l'aide d'un maillage fixe ou adaptatif, les PINNs modélisent les solutions dans un espace continu, en évaluant directement les points requis à travers le domaine. Cette approche permet d'éviter les obstacles liés à la création, à l'ajustement ou à la gestion du maillage, souvent coûteux en temps et en ressources. Elle offre aussi la flexibilité d'évaluer la solution à différents points que ceux choisis lors de l'entraînement.

En conséquence, les PINNs s'adaptent naturellement à des géométries complexes et à des interfaces dynamiques, réduisant les contraintes informatiques tout en préservant la précision des simulations.

En réunissant ces atouts, les PINN constituent une révolution dans la modélisation des systèmes physiques complexes. Leur capacité à allier les lois physiques à un apprentissage non supervisé, à prédire des comportements hors du domaine d'apprentissage et à fonctionner sans maillage en fait un outil puissant pour relever les défis de modélisation des fluides, notamment face aux catastrophes naturelles et aux impacts du changement climatique.

0.3 Objectifs du mémoire

L'objectif général du mémoire est de développer de nouvelles approches de résolution de problèmes de mécanique des fluides en utilisant les dernières avancées dans le domaine de l'apprentissage profond, notamment les réseaux de neurones informés par la physique (PINN).

Les objectifs spécifiques du mémoire sont les suivants :

- Développer un code permettant l'entraînement et l'évaluation d'un PINN standard et de plusieurs variantes de PINN ;
- Entraîner et évaluer les PINN sur des problèmes de références pour les SWE et pour les méthodes level set ;
- Comparer les performances de différentes variantes de PINN pour les problèmes testés.

0.4 Plan du mémoire

Ce mémoire est structuré en plusieurs parties qui permettent d'introduire, de développer et d'analyser les approches proposées pour la modélisation des phénomènes physiques à l'aide des réseaux de neurones informés par la physique (PINN).

Le premier chapitre présente une analyse des travaux précédents dans les domaines des méthodes level set, des SWE et des PINNs. Il explore d'abord les méthodes numériques classiques pour les méthodes level set et des SWE. Ensuite, l'état actuel de la recherche sur les PINNs est exploré, mettant en évidence les forces et les faiblesses des approches standards. Plusieurs améliorations

issues de la littérature sont ensuite présentées, suivies d'une revue des librairies disponibles pour réaliser les expériences.

Le deuxième chapitre décrit en détail les méthodes et outils utilisés pour mener cette étude. Il comprend une description des méthodes level set et leur application pour le suivi d'interfaces mouvantes, les SWE comme modèle mathématique pour les écoulements à surface libre, ainsi qu'une présentation des architectures PINNs testées. Ces architectures incluent le PINN standard de Raissi, Perdikaris & Karniadakis (2019), le PINN amélioré de Wang, Wang & Perdikaris (2021c) et le PirateNet de Wang *et al.* (2025).

Le troisième chapitre présente et analyse les résultats obtenus à partir des différentes simulations. Il inclut des tests de référence pour valider les modèles sur des problèmes classiques, soit les équations de Burgers 1D et 2D de Burgers (1948), des simulations de méthodes level set soit le disque de Zalesak de Zalesak (1979) et les écoulements tourbillonnaires inversés dans le temps tels que dans Touré & Soulaïmani (2016), ainsi que des analyses des SWE appliqués aux problèmes de propagation d'une vague sur une digue et à la rupture de barrage radiale en 2D.

Enfin, la conclusion synthétise les contributions principales du mémoire, en mettant en évidence les forces des approches PINNs appliquées à des problèmes complexes de mécanique des fluides.

À noter que le premier volet de ce mémoire, consacré à l'application des PINNs aux méthodes level set, repose en grande partie sur un travail publié dans Mullins, Kamil, Fahsi & Soulaimani (2025). De même, les résultats liés aux équations de Saint-Venant ont fait l'objet d'une présentation à la conférence DTE & AICOMAS en février 2025 Mullins & Soulaimani (2025).

CHAPITRE 1

REVUE DE LITTÉRATURE

1.1 Méthodes numériques classiques

Les méthodes numériques classiques ont longtemps constitué la base de la dynamique des fluides computationnelle (CFD) pour résoudre des problèmes d'écoulements multiphasiques. Deux approches principales se distinguent : les méthodes lagrangiennes et les méthodes eulériennes. Ces approches présentent des caractéristiques distinctes, chacune offrant des avantages et des inconvénients en fonction des problèmes à traiter.

Les méthodes lagrangiennes suivent l'interface à l'aide d'un maillage qui se déplace et s'adapte avec l'écoulement. Cette approche permet des simulations très précises de l'évolution de l'interface, ce qui est idéal pour des cas où une représentation fidèle de l'interface est cruciale. Par ailleurs, les méthodes lagrangiennes se sont révélées particulièrement utiles dans les problèmes d'interaction fluide-structure, comme l'illustre l'approche développée par Donea, Giuliani & Halleux (1982) dans leur travail sur une méthode des éléments finis arbitraire lagrangienne-eulérienne pour des interactions dynamiques transitoires. Cependant, son application devient complexe dans les problèmes tridimensionnels (3D), en particulier lorsqu'il s'agit d'interfaces complexes ou très déformables.

En revanche, les méthodes eulériennes utilisent un maillage fixe qui reste stationnaire tout au long de la simulation. Elles comprennent notamment des techniques telles que le suivi du front ("front-tracking") et la capture du front ("front-capturing"). Ces approches sont mieux adaptées aux géométries complexes et aux grandes déformations. Toutefois, elles exigent une résolution de maillage fin et des modifications substantielles dans les formulations de discrétisation pour garantir la précision près de l'interface.

Les méthodes de suivi du front utilisent des marqueurs pour suivre l'évolution de l'écoulement. Ces techniques se divisent en deux catégories : le suivi par volume ("volume-tracking") et le suivi par surface ("surface-tracking").

Dans la famille du suivi par volume, la méthode Marker and Cell a été utilisée pour traiter des problèmes comme la rupture de barrage. Cependant, Harlow & Welch (1965) ont noté que cette méthode exigeait une redistribution fréquente des marqueurs pour maintenir la précision. Le suivi par surface, quant à lui, offre une précision supérieure à celle des méthodes basées sur le volume. Ces techniques ont été appliquées à des problèmes tels que la montée et la fusion de bulles par Unverdi & Tryggvason (1992).

Les méthodes de capture du front suivent l'évolution d'une fonction scalaire liée à la position de l'interface. Leur dépendance à une fonction scalaire simple rend des techniques comme le Volume of Fluid (VOF) et les méthodes level set relativement facile à implémenter. Dans la méthode VOF, la fonction utilisée est la fraction volumique d'une phase, comme illustré dans la résolution des problèmes d'interaction fluide-surface à l'aide du simple line interface calculation dans Noh & Woodward (1976).

La suite de cette section portera sur deux méthodes eulériennes populaires en mécanique des fluides, soit la méthode level set et la méthode des volumes finis.

1.1.1 Méthode level set

La méthode level set définit l'interface mouvante comme le contour zéro d'une fonction de distance signée (SDF), ce qui permet à l'interface de changer de forme et de topologie de manière fluide. Initialement développée par Osher & Sethian (1988), cette méthode a montré qu'elle pouvait capturer avec précision la formation de gradients abrupts et de cuspides dans des fronts mouvants. Elle est particulièrement adaptée aux problèmes impliquant des ruptures et des topologies complexes.

Malgré ses avantages, la méthode level set présente certains défis, notamment l'accumulation d'erreurs au fil du temps. La méthode repose sur une équation de transport pour déplacer l'interface en fonction d'un champ de vitesse donné. Cependant, ces erreurs peuvent entraîner une perte de lissage à l'interface et une perte de masse, selon les paramètres de simulation. Pour pallier ces problèmes, de nombreux travaux se sont concentrés sur le développement

d'algorithmes de réinitialisation. Ces techniques visent à rétablir la propriété de distance signée de la fonction level set, bien qu'elles ajoutent une complexité supplémentaire, notamment en ce qui concerne le choix de la fréquence de réinitialisation et de l'algorithme utilisé Gomes & Faugeras (2000). Certaines méthodes évitent complètement la réinitialisation, mais au prix d'un coût de calcul accru Touré & Soulaïmani (2016).

1.1.2 Méthode des volumes finis

La méthode des volumes finis est une approche très populaire en CFD pour résoudre les équations de conservation, notamment les équations de Navier-Stokes. Elle repose sur la discrétisation du domaine en cellules de contrôle, où les flux à travers les faces des cellules sont calculés pour garantir la conservation des quantités physiques comme la masse, la quantité de mouvement et l'énergie. Cette méthode est particulièrement adaptée aux écoulements compressibles et incompressibles, ainsi qu'aux géométries complexes.

Un avantage clé des volumes finis est leur capacité à maintenir la conservation stricte des quantités physiques au niveau local, ce qui est crucial pour des simulations précises. De plus, la méthode des volumes finis est compatible avec des maillages non structurés, ce qui la rend flexible pour des applications industrielles et des configurations géométriques complexes. Par exemple, cette méthode a été utilisée avec succès pour modéliser des écoulements océaniques dans Marshall, Adcroft, Hill, Perelman & Heisey (1997). Plusieurs travaux recensent également l'utilisation de volumes finis pour résoudre les SWE tels que Bradford & Sanders (2002) et Loukili & Soulaimani (2007).

Cependant, cette approche présente également des défis, tels que la complexité des calculs qui augmente rapidement avec la taille du domaine et le raffinement du maillage, ce qui peut entraîner des coûts de calcul importants. Afin de diminuer les impacts de ce problème, plusieurs codes ont d'ailleurs été modifiés pour être exécutés sur plusieurs GPU simultanément comme dans Delmas & Soulaïmani (2022).

1.2 Apprentissage profond

L'intelligence artificielle (IA) englobe un large éventail de techniques qui visent à doter les machines de la capacité d'apprendre et de réaliser des tâches complexes. Au sein de ce domaine, l'apprentissage automatique (machine learning) se distingue comme une approche centrale, permettant aux modèles de s'améliorer automatiquement en analysant des données. Une sous-branche influente de l'apprentissage automatique est l'apprentissage profond (deep learning), qui exploite des réseaux de neurones artificiels à plusieurs niveaux pour apprendre des représentations hiérarchiques, rendant possible le traitement efficace de données complexes et massives. Une représentation graphique de ces sous-sections est présente à la Figure 1.1.

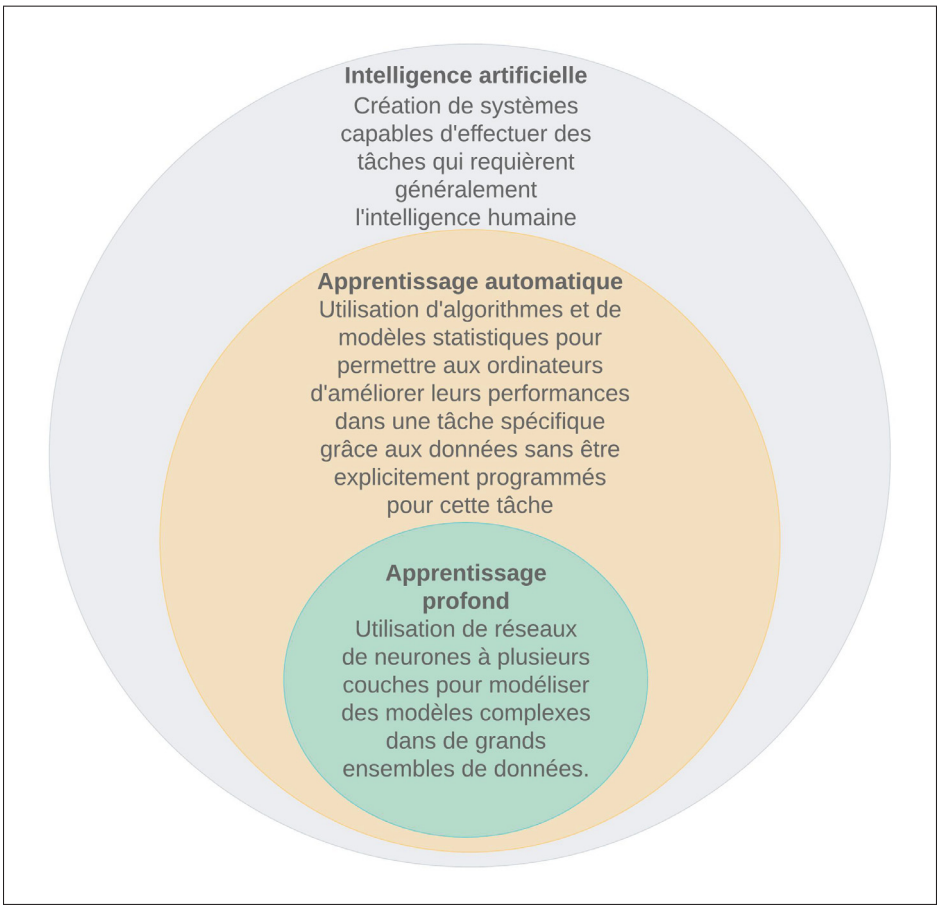


Figure 1.1 Représentation graphique de l'apprentissage profond à titre de sous-ensemble de l'IA

Grâce à son utilisation de réseaux de neurones à plusieurs couches, l'apprentissage profond s'est imposé comme une approche prometteuse dans divers domaines scientifiques, y compris la dynamique des fluides computationnelle (CFD). En effet, il a montré son potentiel pour prédire des écoulements complexes tout en réduisant les coûts de calcul lors de l'inférence. En capturant des représentations pertinentes des phénomènes fluides, l'apprentissage profond offre une perspective innovante pour améliorer nos méthodes de modélisation des écoulements.

Toutefois, les applications de l'IA à la mécanique des fluides ne se limitent pas à l'utilisation exclusive de grands ensembles de données. En effet, on distingue trois catégories de méthodes : les méthodes fondées sur les données, les méthodes fondées sur la physique et les méthodes hybrides.

Les méthodes fondées sur les données exploitent principalement de grands ensembles de données pour extraire des motifs et construire des modèles capables de prédire des phénomènes physiques. En revanche, les méthodes fondées sur la physique intègrent directement les équations physiques fondamentales dans le processus d'apprentissage, garantissant ainsi que les solutions respectent les lois physiques sous-jacentes. Enfin, les méthodes hybrides combinent ces deux approches, exploitant à la fois la richesse des données et la rigueur des lois physiques, afin d'améliorer la précision et la généralisation des modèles.

Les sections qui suivent examineront chacune de ces catégories en détail, en mettant en évidence leurs forces, leurs limites et des exemples concrets de leur utilisation à la mécanique des fluides.

1.2.1 Méthodes fondées sur les données

Les méthodes d'apprentissage profond fondées sur les données exploitent de grands ensembles de données pour découvrir des motifs, construire des modèles prédictifs et reproduire des comportements complexes observés dans les systèmes physiques. Ces approches reposent sur la capacité des modèles d'intelligence artificielle à apprendre directement des données sans recourir explicitement aux équations gouvernantes de la physique. C'est d'ailleurs pour cette raison que ces modèles sont qualifiés de boîtes noires, car ils produisent des prédictions sans

fournir d'explication explicite sur les relations physiques sous-jacentes qu'ils exploitent. Elles ont été largement adoptées dans divers domaines, y compris la mécanique des fluides.

1.2.1.1 Applications en mécanique des fluides

Sachant que les méthodes fondées sur les données ne respectent pas absolument les lois physiques, mais qu'elles peuvent fournir rapidement de bonnes approximations, elles ont été appliquées avec succès dans divers contextes tels que les suivants.

Guo, Leitão, Simões & Moosavi (2021) ont proposé une méthode pour l'émulation rapide des inondations urbaines à l'aide d'une approche fondée sur les données. Celle-ci est basée sur des réseaux de neurones convolutifs et a permis de prédire la profondeur maximale d'eau en transformant le problème en une traduction image à image, éliminant ainsi le besoin de simulations physiques complètes. Cette méthode a réduit les temps de calcul à seulement 0.5 % de celui des modèles traditionnels, tout en maintenant une précision prometteuse.

Récemment, Bhatt *et al.* (2023) ont proposés un Convolutional Autoencoder pour la prédiction extrapolative des problèmes de mécaniques des fluides comme les équations de Burgers et Stoker. Ils ont également appliqué l'architecture à un problème de rupture de barrage sur une bathymétrie réelle en 2D.

De même, Aulich, Goinis & Voß (2024) ont développé une méthode basée sur les transformeurs pour produire des approximations rapides d'écoulements 3D de turbomachines. Ces approximations rapides sont très utiles pour le développement de nouveaux tracés de turbomachines qui requièrent de nombreuses itérations initiales avant de pouvoir peaufiner le modèle sur des simulations numériques plus précises.

1.2.1.2 Avantages et limites

Les méthodes fondées sur les données présentent plusieurs avantages. Elles peuvent exploiter de grands ensembles de données expérimentales ou simulées pour construire des modèles

prédictifs puissants, sans avoir besoin de connaissances préalables sur les équations physiques. De plus, leur facilité d'implémentation et leur rapidité d'inférence les rendent très utiles pour le prototypage rapide. Cependant, ces approches sont fortement dépendantes de la qualité et de la quantité des données disponibles. Lorsqu'elles sont appliquées à des scénarios hors du domaine des données d'entraînement, elles peuvent produire des résultats physiquement incohérents.

En dépit de ces limites, les méthodes fondées sur les données restent un outil puissant pour explorer et modéliser des écoulements fluides complexes.

1.2.2 Méthodes fondées sur la physique : les PINN

Introduits par Raissi *et al.* (2019), les réseaux de neurones informés par la physique (PINN) offrent une approche novatrice pour résoudre des problèmes directs, inverses et mixtes gouvernés par des équations différentielles partielles (EDP). Ces réseaux intègrent directement les lois physiques dans le processus d'apprentissage à travers un terme de régularisation ajouté à la fonction de perte. Ce terme est obtenu en appliquant un opérateur différentiel directement sur les sorties du réseau neuronal. L'expression résultante est ensuite évaluée et minimisée sur des points de collocation sélectionnés dans le domaine de calcul. Le problème est ainsi transformé en une optimisation de la fonction de perte, remplaçant la résolution explicite des équations gouvernantes, comme cela est traditionnellement effectué par les méthodes numériques classiques.

Cette approche est rendue possible grâce à la différenciation automatique (AD) de Baydin, Pearlmutter, Radul & Siskind (2017), qui utilise la règle de chaîne pour calculer les dérivées des expressions mathématiques dans les réseaux de neurones. De plus, les PINN fonctionnent sans maillage et peuvent apprendre de manière complètement non supervisée, éliminant ainsi le besoin de données étiquetées coûteuses. Cela en fait une option attrayante pour résoudre des problèmes où les données sont rares ou pour des domaines à haute dimensionnalité.

1.2.2.1 Applications des PINN

Depuis leur introduction, les PINN ont montré des résultats prometteurs dans divers domaines scientifiques et techniques, notamment :

- **Science des matériaux :** Simulation des défauts cristallins dans Salvati, Tognan, Laurenti, Pelegatti & De Bona (2022) et modélisation des propriétés mécaniques dans Fang & Zhan (2020).
- **Énergie :** Optimisation des échanges thermiques dans Cho, Zhu, Campbell & Wang (2022) et augmentation de l'efficacité des systèmes dans Pombo, Bindner, Spataru, Sørensen & Bacher (2022).
- **Thermodynamique :** Modélisation de systèmes complexes dans Liu *et al.* (2023) et dans Masclans, Vázquez-Novoa, Bernades, Badia & Jofre (2023).
- **Dynamique des fluides :** Résolution des équations de Navier-Stokes dans Dazzi (2024) et des écoulements à surface libre dans Donnelly, Daneshkhah & Abolfathi (2024).

1.2.2.2 Améliorations des PINNs

Malgré leur succès, les PINNs présentent certaines limitations dans leur forme originale, telles que le biais spectral tel que vu dans Rahaman *et al.* (2019); Wang, Wang & Perdikaris (2021b), des flux de gradients déséquilibrés relevés par Wang, Teng & Perdikaris (2021a); Wang, Yu & Perdikaris (2022c), et des problèmes de violation de la causalité temporelle dans Wang, Sankaran & Perdikaris (2022a). Plusieurs améliorations ont été proposées pour surmonter ces défis :

- **Modifications des architectures neuronales :** Utilisation de perceptrons multicouches modifiés (modified MLP) Wang *et al.* (2021a), de l'architecture PirateNet Wang *et al.* (2025) et des fonctions de Fourier aléatoires (random Fourier features) Rahaman *et al.* (2019); Tancik *et al.* (2020).

- **Améliorations des algorithmes d'entraînement :** Factorisation aléatoire des poids (random weight factorization) Wang *et al.* (2022b), rééchantillonnage adaptatif des points de collocation (adaptive resampling of collocation points) Daw, Bu, Wang, Perdikaris & Karpatne (2023); Wu, Zhu, Tan, Kartha & Lu (2023), respect de la causalité temporelle (causal training) Wang *et al.* (2022a), rééquilibrage adaptatif du poids des termes de la fonction de perte par la méthode de norme des gradients (gradient normalization-based re-weighting of loss terms) Chen, Badrinarayanan, Lee & Rabinovich (2018), l'apprentissage par curriculum Bengio, Louradour, Collobert & Weston (2009); Krishnapriyan, Gholami, Zhe, Kirby & Mahoney (2021) et l'apprentissage séquence par séquence (S2S) Krishnapriyan *et al.* (2021).

Malgré leurs défis, les PINNs représentent une solution prometteuse pour les problèmes complexes où les méthodes traditionnelles rencontrent des limites. C'est d'ailleurs pourquoi la recherche a beaucoup avancé dans les dernières années avec la proposition de multiples améliorations aux PINNs standards de Raissi *et al.* (2019). Les améliorations qui seront utilisées dans le cadre de cette recherche seront détaillées mathématiquement dans la section 2.3.2.

1.2.3 Méthodes hybrides

Les méthodes hybrides combinent des approches basées sur les données avec des modèles physiques pour tirer parti des points forts des deux approches. Contrairement aux méthodes purement basées sur les données, les méthodes hybrides intègrent explicitement les lois physiques, ce qui permet d'assurer une meilleure cohérence des résultats avec la physique et l'application réelle. De plus, ces méthodes permettent souvent de réduire la dépendance à des ensembles de données volumineux tout en préservant la capacité d'apprendre des relations complexes à partir des données disponibles.

1.2.3.1 Assimilation de données

L'assimilation de données est une technique clé des méthodes hybrides. Elle consiste à intégrer des observations réelles dans un modèle physique pour améliorer ses prédictions. Cette approche

est particulièrement utile dans des domaines où les systèmes à modéliser sont complexes et où les observations seules ne suffisent pas pour capturer toutes les dynamiques sous-jacentes. Une méthode populaire en assimilation est le filtre de Kalman introduit par Kalman (1960).

1.2.3.2 Applications des méthodes hybrides

Un exemple classique est l'utilisation de l'assimilation de données dans les modèles météorologiques. Par exemple, l'approche d'assimilation de données utilisant une technique de filtre de Kalman ensembliste, telle que présentée par Houtekamer & Mitchell (1998), a été appliquée avec succès dans les modèles de prévision météorologique.

En mécanique des fluides, l'assimilation de données a été appliquée à des problèmes complexes tels que la prévision des inondations. Par exemple, Ziliani, Ghostine, Ait-El-Fquih, McCabe & Hoteit (2019) ont proposé une approche combinant l'assimilation de données avec des filtres de Kalman ensembliste aux équations de Saint-Venant pour améliorer la précision des prévisions d'inondations. Cette méthode a montré son efficacité dans la réduction des incertitudes liées aux paramètres hydrologiques tout en intégrant des observations réelles.

Les méthodes hybrides sont également utiles pour d'autres branches de la science. En effet, en épidémiologie, Silva, Heaney, Li & Pain (2022) ont combiné l'assimilation de données à un réseau adversaire génératif (GAN) pour la prévision de la propagation de la Covid-19.

Malgré l'immense potentiel des méthodes hybrides, notamment pour les problèmes de prévision en temps réel, notre recherche se concentrera sur les méthodes informées par la physique. Tout cela dans le but d'améliorer nos connaissances des PINNs avant de pouvoir les appliquer à des cas réels nécessitant de l'assimilation de données.

1.2.4 PINN pour les level set

La littérature suggère que les réseaux de neurones informés par la physique (PINN) peuvent résoudre avec succès une variété de problèmes pilotés par les EDP ; cependant, peu de recherches

ont été consacrées à l'application des PINN pour résoudre l'équation de level set, en particulier dans les problèmes de mécanique des fluides complexes. Dans M. Silva, Grave & Coutinho (2024) et Zhou, Miwa & Okamoto (2024), une méthode level set basée sur les PINNs a été utilisée pour résoudre un problème de bulle montante dans un champ d'écoulement constant. Plus récemment, Tang, Xin & Wang (2024) a utilisé un PINN couplé aux équations level set pour résoudre un problème de condensation de vapeur. À la connaissance de l'auteur, il s'agit de la première étude à résoudre un problème level set en utilisant un modèle basé sur les PINNs dans un champ de vitesse variable complexe qui démontre une forte vorticité et un étirement de l'interface.

1.2.5 PINN pour les équations de Saint-Venant

Depuis leur introduction par Raissi *et al.* (2019), les PINNs ont rapidement évolué pour être appliqués à des problèmes complexes de mécanique des fluides. De nombreux problèmes dans ce domaine présentent des caractéristiques difficiles à apprendre pour des réseaux de neurones conventionnels, nécessitant ainsi des modifications pour garantir des résultats physiquement cohérents.

Les équations de Saint-Venant, qui modélisent l'écoulement de l'eau à surface libre, comportent des termes d'advection pouvant engendrer de forts gradients et des discontinuités. Ces variations brusques sont particulièrement difficiles à capturer pour les réseaux de neurones et les PINNs Tian, Ding, Su, Huang & Chen (2025); Zhan, Zhang, Zhang & Yang (2025). C'est pourquoi les cas de rupture de barrage sont fréquemment utilisés dans la littérature pour évaluer la performance des modèles.

Récemment, plusieurs études ont exploré différentes architectures de PINNs pour modéliser ces scénarios. Leiteritz, Hurler & Pflüger (2021) ont testé divers algorithmes d'optimisation du taux d'apprentissage sur un PINN afin de résoudre la rupture d'un barrage en 1D avec une bathymétrie variable ainsi que la rupture de barrage radiale en 2D. Une approche hybride combinant données et PINNs a été proposée par Li *et al.* (2023) pour modéliser différentes ruptures de barrages en 2D.

De leur côté, Zhan *et al.* (2025) ont adopté une stratégie de décomposition des caractéristiques spatiales des ondes de crue selon différentes directions de propagation pour traiter des ruptures de barrages en 1D et 2D. Par ailleurs, Yin *et al.* (2025) ont montré que les PINNs peuvent extrapoler dans le temps pour prédire raisonnablement les solutions aux SWE en 1D.

Bien que des avancées récentes aient été proposées pour modéliser les SWE à l'aide de variantes de PINNs, ces derniers restent moins précis que les méthodes numériques classiques. Il est donc essentiel de développer de nouvelles approches pour combler cette lacune. Des architectures de pointe comme les PirateNets Wang *et al.* (2025), combinées à des stratégies d'entraînement avancées mentionnées en 1.2.2.2, n'ont pas encore été testées sur les SWE au meilleur des connaissances de l'auteur. Cela constitue une contribution scientifique inédite et représente une opportunité d'améliorer la précision des PINNs face aux problèmes fortement non linéaires des SWE.

1.3 Optimisation des hyperparamètres

Les hyperparamètres sont des paramètres fixes définis avant l'entraînement d'un modèle d'apprentissage automatique. À l'opposé, les paramètres internes comme les poids et les biais sont ajustés au cours de l'entraînement. Les hyperparamètres contrôlent différents aspects du processus d'apprentissage, tels que le taux d'apprentissage, le nombre de couches ou le nombre de neurones dans un réseau. Le choix des hyperparamètres est très important puisqu'il influence directement la performance et la généralisation du modèle. Les hyperparamètres peuvent être fixés par les utilisateurs à des valeurs qu'ils jugent adéquates. Cependant, cela fait en sorte que la performance du modèle dépend grandement de l'expérience de cet utilisateur. Sachant que les valeurs optimales des hyperparamètres varient grandement de problème en problème, il est presque impossible que l'utilisateur trouve du premier coup l'ensemble optimal.

Cela dit, l'optimisation automatique des hyperparamètres joue un rôle essentiel dans la performance des modèles d'apprentissage automatique, car elle permet de déterminer les hyperparamètres optimaux. Cela affecte directement la capacité du modèle à généraliser

efficacement. En testant et ajustant différentes valeurs, l'optimisation des hyperparamètres permet de trouver l'ensemble d'hyperparamètres générant la plus petite erreur de validation. Cela fonctionne habituellement ainsi. Un ensemble d'hyperparamètres est prédéterminé parmi l'espace d'hyperparamètres. Le modèle est entraîné avec cet ensemble et est ensuite évalué sur un ensemble de données de validation. L'évaluation du modèle peut par exemple être basée sur l'erreur L^2 relative. Plusieurs ensembles sont testés et après n tentatives, l'ensemble ayant donné l'erreur de validation la plus petite est considéré comme l'ensemble optimal. Ce processus est résumé dans l'algorithme 1.1.

Algorithme 1.1 Optimisation des hyperparamètres pour l'apprentissage automatique

Input : Ensemble de données d'entraînement $\mathcal{D}_{\text{train}}$, ensemble de données de validation $\mathcal{D}_{\text{val}}$, espace des hyperparamètres $\mathcal{H}$, nombre de tentatives n
Output : Ensemble optimal d'hyperparamètres $\mathbf{h}^*$

Initialiser $\mathbf{h}^* \leftarrow \emptyset$ et l'erreur minimale $E_{\min} \leftarrow +\infty$;
for $i = 1$ ***to*** n **do**
    Sélectionner un ensemble d'hyperparamètres $\mathbf{h}_i \in \mathcal{H}$;
    Entraîner le modèle $M(\mathbf{h}_i)$ sur $\mathcal{D}_{\text{train}}$;
    Évaluer $M(\mathbf{h}_i)$ sur $\mathcal{D}_{\text{val}}$ et calculer l'erreur relative L_i^2 ;
    if $L_i^2 < E_{\min}$ **then**
        Mettre à jour $E_{\min} \leftarrow L_i^2$
        Mettre à jour $\mathbf{h}^* \leftarrow \mathbf{h}_i$
    end if
end for
return $\mathbf{h}^*$

Plusieurs méthodes s'offrent pour effectuer cette optimisation, chacune ayant ses avantages et inconvénients. Les trois principales sont la recherche en grille, la recherche aléatoire et l'optimisation bayésienne.

La recherche en grille (Grid Search) explore systématiquement toutes les combinaisons possibles d'hyperparamètres prédéfinis. Cette méthode a pour avantage d'être simple à implémenter et d'assurer une couverture complète de l'espace des hyperparamètres. Toutefois, elle est extrêmement coûteuse en temps de calcul, notamment lorsque l'espace des hyperparamètres est

étendu, ce qui la rend peu adaptée aux problèmes complexes. Par exemple, avec seulement trois hyperparamètres à ajuster et 3 options possibles par hyperparamètres, $3^3 = 27$ combinaisons doivent être testées.

En revanche, la recherche aléatoire (Random Search) propose une exploration plus rapide en sélectionnant des points de manière aléatoire dans l'espace des hyperparamètres. Bien que cette approche soit plus efficace pour de grands espaces, elle peut néanmoins manquer des régions prometteuses, car les points sont choisis au hasard.

L'optimisation bayésienne, quant à elle, utilise un modèle probabiliste pour approximer la fonction objectif reliant les hyperparamètres à la performance du modèle. Elle réduit significativement le nombre d'évaluations nécessaires en équilibrant exploration et exploitation de l'espace. Cependant, cette méthode peut être complexe à implémenter et exige souvent l'utilisation de modèles comme les processus gaussiens.

Dans le cadre de cette recherche, l'optimisation bayésienne sera utilisée, car elle permet de trouver rapidement les configurations les plus performantes. De plus, des librairies telles que WandB de Biewald & Van Pelt (2020) rendent l'implémentation moins complexe.

1.4 Choix de la librairie d'apprentissage profond

Plusieurs cadres Python sont disponibles pour mener à terme des projets d'apprentissage profond. Parmi les plus populaires se trouvent TensorFlow Martín Abadi *et al.* (2015), PyTorch Paszke *et al.* (2019) et Jax Bradbury *et al.* (2018). Voici les caractéristiques de chacune.

1.4.1 TensorFlow

Développé par Google, TensorFlow a rapidement été adopté par la communauté d'apprentissage profond comme un cadre de choix. C'est d'ailleurs pourquoi ce cadre contient autant de documentation et de support mis à jour par sa vaste communauté. Tensorflow est d'autant plus reconnu pour son extensibilité lors d'un entraînement sur des systèmes distribués comme des

GPU, mais peut poser problème aux débutants dû à sa complexité. De plus, son compilateur XLA (Accelerated Linear Algebra) permet d'optimiser le code pour de meilleures performances.

1.4.2 PyTorch

Développé par le laboratoire de recherche en IA de Facebook, PyTorch a gagné une immense popularité grâce à son graphique de calcul dynamique et à son interface intuitive. Le graphique de calcul dynamique de PyTorch permet une plus grande flexibilité et facilite le débogage contrairement au graphique statique de TensorFlow. Cela rend toutefois l'entraînement moins efficace, c'est pourquoi PyTorch supporte la compilation juste-à-temps (JIT) pour améliorer les performances. L'interface et les ressources éducatives abondantes rendent le cadre très facile à utiliser, notamment pour les débutants.

1.4.3 JAX

Google a développé JAX spécialement pour le calcul numérique haute performance. En transformant des fonctions Python avec une syntaxe similaire à Numpy et grâce à la compilation juste-à-temps (JIT), JAX est capable d'exécuter des programmes efficacement sur GPU et sur TPU. Ces capacités de différentiation automatique le rendent aussi excellent pour l'apprentissage profond. Cependant, étant plus récent et requérant de la programmation fonctionnelle, JAX est moins populaire parmi les débutants.

1.4.4 Librairie choisie

Notre recherche vise à développer une preuve de concept des PINN dans le but de résoudre les équations de Saint-Venant sur des bathymétries simples. Cependant, notre recherche sera éventuellement appliquée sur des bathymétries réelles dont les fichiers sont souvent volumineux. Cela dit, il est important que le code développé soit extensible à ces problèmes de plus grande ampleur et donc qu'il soit très performant. JAX est donc l'option la plus intéressante pour nous.

Notre choix est d'autant plus motivé par la publication d'une librairie JAX hautement optimisée et adaptée au développement d'architectures PINN destinées à résoudre des problèmes basés sur les EDP par Wang, Sankaran, Wang & Perdikaris (2023). Cette librairie facilitera grandement le développement de notre code.

D'autres librairies spécifiques à la résolution d'EDP à l'aide de méthodes d'apprentissage profond basée sur la physique existent aussi comme DeepXDE. Celle-ci permet rapidement et facilement de résoudre des problèmes classiques à l'aide de PINNs. Cependant, il est plus difficile d'apporter des modifications aux architectures déjà mises en place, ce qui freine le développement de nouvelles architectures et l'application à des problèmes plus complexes.

CHAPITRE 2

MÉTHODOLOGIE

Ce chapitre élabore les fondements mathématiques permettant de définir les modèles qui seront utilisés pour les expérimentations. Le chapitre présentera d'abord le cadre théorique des méthodes level set et des équations de Saint-Venant. Ensuite, les réseaux de neurones informés par la physique, leurs améliorations et comment les utiliser pour résoudre des méthodes level set et les équations de Saint-Venant seront abordés.

2.1 Méthodes level set

Dans ce qui suit, nous présentons l'approche level set comme un cadre pour le suivi des interfaces dans les problèmes d'écoulements multiphasiques.

2.1.1 Formulation générale de la méthode level set

La méthode level set, introduite par Osher et Sethian Osher & Sethian (1988), est un outil efficace pour suivre le mouvement des interfaces dans les problèmes d'écoulements multiphasiques. Cette méthode utilise une fonction scalaire, appelée fonction level set $\phi(\boldsymbol{x}, t)$, où le niveau zéro $\phi(\cdot, t) = 0$ définit l'interface qui sépare deux ou plusieurs phases. Mathématiquement, l'interface au temps t est représentée comme suit :

$$\Gamma(t) = \left\{ \boldsymbol{x} \in \mathbb{R}^d \;\middle|\; \phi(\boldsymbol{x}, t) = 0 \right\}, \tag{2.1}$$

avec $d = 2, 3$ correspondant aux dimensions spatiales.

La figure 2.1 illustre un exemple de problème d'écoulement diphasique. Dans cet exemple, $\Omega_i(t)$ représente la région occupée par la phase i au temps t, avec le domaine entier donné par $\Omega = \Omega_1 \cup \Omega_2$. L'interface qui sépare les sous-domaines Ω_1 et Ω_2 est $\Gamma(t) = \Omega_1 \cap \Omega_2$. La fonction

level set ϕ est définie selon une convention de signe comme suit :

$$\begin{cases} \phi(\boldsymbol{x},t) < 0 & \text{if } \boldsymbol{x} \in \Omega_1, \\ \phi(\boldsymbol{x},t) = 0 & \text{if } \boldsymbol{x} \in \Gamma, \\ \phi(\boldsymbol{x},t) > 0 & \text{if } \boldsymbol{x} \in \Omega_2. \end{cases} \tag{2.2}$$

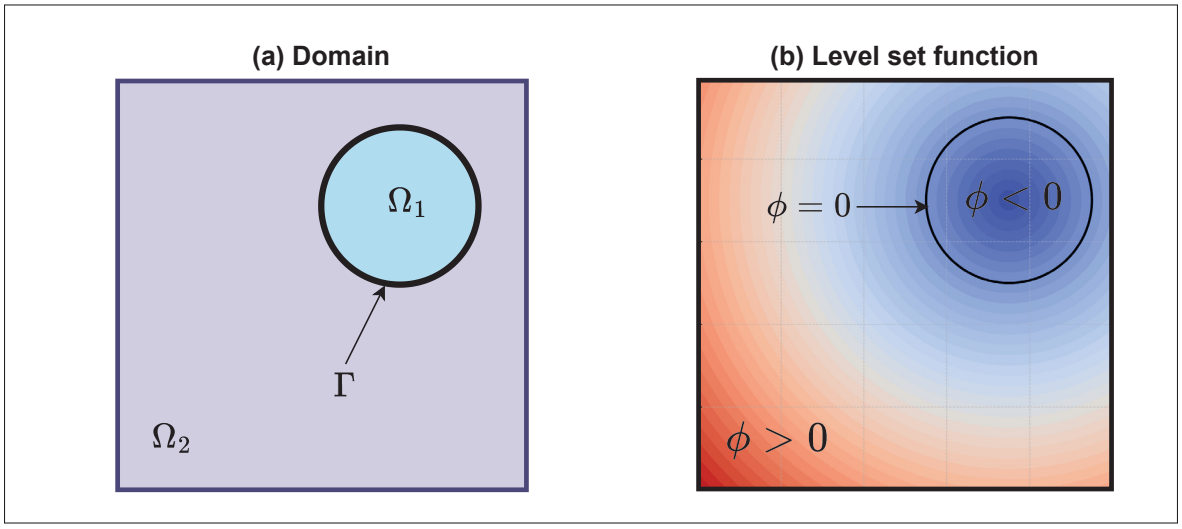


Figure 2.1 Illustration d'un domaine diphasique (a), avec Ω_i représentant la phase i (b) Montre un exemple de fonction level set correspondant au domaine en (a)

La fonction level set ϕ est souvent définie comme une fonction de distance signée (SDF) , exprimée par :

$$d_{\text{signed}}(\boldsymbol{x},t) = \begin{cases} -d(\boldsymbol{x},t) & \text{if } \boldsymbol{x} \in \Omega_1, \\ 0 & \text{if } \boldsymbol{x} \in \Gamma, \\ +d(\boldsymbol{x},t) & \text{if } \boldsymbol{x} \in \Omega_2, \end{cases} \tag{2.3}$$

avec $d(\boldsymbol{x},t)$ étant une fonction distance par rapport à l'interface Γ :

$$d(\boldsymbol{x},t) = \min_{\boldsymbol{x}_\Gamma \in \Gamma} \|\boldsymbol{x} - \boldsymbol{x}_\Gamma\|. \tag{2.4}$$

La SDF d_{signed} satisfait l'équation Eikonal suivante :

$$||\nabla d_{\text{signed}}|| = 1. \tag{2.5}$$

L'approche level set suit l'évolution de l'interface $\Gamma(t)$ au fil du temps en identifiant le niveau zéro de la solution de l'équation de transport :

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u} \cdot \nabla \phi = 0, \tag{2.6}$$

avec $\boldsymbol{u}$ représentant le champ de vitesse. Initialement, la fonction level set ϕ est définie comme la SDF $\phi(\boldsymbol{x}, 0) = d_{\text{signed}}(\boldsymbol{x}, 0)$.

2.1.2 Réinitialisation

Les méthodes numériques traditionnelles pour résoudre l'équation level set souffrent souvent d'une accumulation d'erreurs, ce qui peut conduire la fonction level set à perdre sa propriété de distance signée. Préserver cette propriété est essentiel pour calculer avec précision des quantités géométriques telles que la courbure et les vecteurs normaux à l'interface, qui sont cruciaux dans de nombreuses applications.

Pour résoudre ce problème, des méthodes de réinitialisation ont été introduites afin de restaurer la propriété de distance signée de la fonction level set. L'idée principale est de garantir que le gradient de la fonction reste proche de 1, $||\nabla \phi|| \simeq 1$.

Une méthode proposée par Sussman, Smereka & Osher (1994) ajuste la fonction level set de manière itérative en résolvant une EDP sur un pseudo-temps.

Certains articles ont exploré différentes méthodes de réinitialisation, telles que la réinitialisation géométrique Fahsi & Soulaïmani (2017); Ausas, Dari & Buscaglia (2011) qui réinitialise la fonction level set en mesurant la distance entre les nœuds et l'interface du level set et la méthode Fast Marching Sethian (1996a,b). D'autres ont également trouvé des moyens de contourner

l'étape de réinitialisation en utilisant des méthodes d'énergie variationnelle Li, Xu, Gui & Fox (2005); Touré & Soulaïmani (2016).

2.1.3 Conservation de la masse

La conservation de la masse dans les méthodes level set garantit que la surface totale de chaque phase ou matériau représenté par la fonction level set reste constante dans le temps. Cette propriété est importante, car les erreurs numériques peuvent entraîner une perte ou un gain de masse artificiels, en particulier dans les maillages plus grossiers ou lors de simulations à long terme. Le maintien de la conservation de la masse est essentiel lors de l'utilisation d'approches level set, car il n'est pas garanti de manière inhérente au niveau discret.

Malgré divers efforts pour améliorer la conservation de la masse van der Pijl, Segal, Vuik & Wesseling (2005); Olsson & Kreiss (2005); Olsson, Kreiss & Zahedi (2007), l'obtention d'une conservation de la masse très précise reste un défi important.

Pour un champ d'écoulement conservatif, la surface totale de chaque phase représentée par la fonction de niveau doit rester constante dans le temps, comme le décrivent les équations globales de conservation de la surface :

$$\frac{dA_1}{dt} = 0 \quad \text{for } \phi(\boldsymbol{x}, t) < 0, \quad \frac{dA_2}{dt} = 0 \quad \text{for } \phi(\boldsymbol{x}, t) > 0, \tag{2.7}$$

où A_1 et A_2 représentent les surfaces des deux phases séparées par le zéro level set. Ces équations expriment l'exigence selon laquelle les zones enfermées restent invariantes dans le temps, assurant la conservation de la masse dans la formulation continue.

2.2 Équations de Saint-Venant

Les équations de Saint-Venant ou les équations d'écoulement à surface libre à faible profondeur (SWE) sont une simplification des équations de Navier-Stokes. Elles peuvent être utilisées pour modéliser des écoulements à surface libre où la profondeur du fluide est petite par rapport à

l'échelle horizontale comme pour les écoulements dans les rivières, les lacs, ou les océans. Tous deux se basent sur les mêmes principes de conservation de la masse (continuité) et de conservation de la quantité de mouvement (momentum).

Sous leur forme générale, les équations de Navier-Stokes sont définies ainsi :

$$\rho_t + \nabla \cdot (\rho \mathbf{U}) = 0 \tag{2.8}$$

$$(\rho \mathbf{U})_t + \nabla \cdot [(\rho \mathbf{U} \otimes \mathbf{U} + pI - \sigma] = \rho g \tag{2.9}$$

En émettant les hypothèses suivantes, nous pouvons dériver les équations de Saint-Venant.

- Écoulement à surface libre : Le fluide a une surface libre (interface entre l'eau et l'air).
- Profondeur faible : La profondeur h du fluide est petite par rapport aux échelles horizontales L, $h \ll L$, les vitesses et accélérations verticales peuvent être négligées.
- Pression hydrostatique : La pression est supposée hydrostatique dans la direction verticale, ce qui signifie que la pression p est uniquement due au poids de la colonne d'eau, $p = \rho g(h - z)$.
- Fluide incompressible : La densité ρ est constante et l'équation de continuité peut être simplifiée à $\nabla \cdot u = 0$.
- Vitesses uniformes en profondeur : Les vitesses (u, v) sont uniformes sur la profondeur et notés par $\bar{u}(t, x, y)$ et $\bar{v}(t, x, y)$

Une dérivation en détail est disponible dans de nombreux ouvrages dont Delmas (2020).

2.2.1 Équations de Saint-Venant conservatrices

Les équations de Saint-Venant existent sous plusieurs formes. Leur forme la plus commune est la forme conservatrice qui prend en entrée les coordonnées spatio-temporelles $(\boldsymbol{x}, t) \in \mathbb{R} \times [0, T]$ et retourne en sortie les prédictions de hauteur d'eau et de momentum $(h, h\bar{\boldsymbol{u}})$.

Les équations de Saint-Venant sous leur forme conservatrice sont définies ainsi :

Équation de continuité :

$$\frac{\partial h}{\partial t} + \frac{\partial (h\bar{u})}{\partial x} + \frac{\partial (h\bar{v})}{\partial y} = 0 \tag{2.10}$$

Équation de momentum en x :

$$\frac{\partial (h\bar{u})}{\partial t} + \frac{\partial}{\partial x}\left(h\bar{u}^2 + \frac{1}{2}gh^2\right) + \frac{\partial}{\partial y}\left(h\bar{u}\bar{v}\right) = -ghb_x + S_{f,x} \tag{2.11}$$

Équation de momentum en y :

$$\frac{\partial (h\bar{v})}{\partial t} + \frac{\partial}{\partial x}\left(h\bar{u}\bar{v}\right) + \frac{\partial}{\partial y}\left(h\bar{v}^2 + \frac{1}{2}gh^2\right) = -ghb_y + S_{f,y} \tag{2.12}$$

où :

- h est la hauteur d'eau.
- $\bar{u}$ et $\bar{v}$ sont les vitesses moyennes en profondeur dans les directions x et y, respectivement.
- g est l'accélération due à la gravité.
- b_x et b_y sont les gradients de bathymétrie dans les directions x et y, respectivement.
- $S_{f,x}$ et $S_{f,y}$ sont les termes sources de frottement dans les directions x et y, respectivement.
- s est la hauteur de la surface libre, $s = b + h$

Une illustration de la notation choisie est présentée à la Figure 2.2.

La forme conservatrice convient pour la résolution à l'aide de méthodes numériques classiques comme la méthode des volumes finis. Par contre, tel que mentionné dans Leiteritz *et al.* (2021), elle pose problème lorsque nous souhaitons l'appliquer aux PINNs. Pour calculer le résiduel des SWE dans le PINN, nous devons calculer les termes $\frac{\partial h\bar{u}^2}{\partial x}$ de l'équation 2.11 et $\frac{\partial h\bar{v}^2}{\partial y}$ de l'équation 2.12 en multipliant les vitesses $\bar{u}$ et $\bar{v}$ par les momentum hu et hv. Cependant, puisque les vitesses $\bar{u}$ et $\bar{v}$ ne sont pas explicitement obtenues en sortie du PINN, nous devons les obtenir en divisant le momentum par la hauteur, $\bar{u} = \frac{h\bar{u}}{h}$. Cela pose problème lors de l'entraînement du PINN puisque la prédiction n'obéit pas toujours aux EDP en début d'entraînement, la hauteur

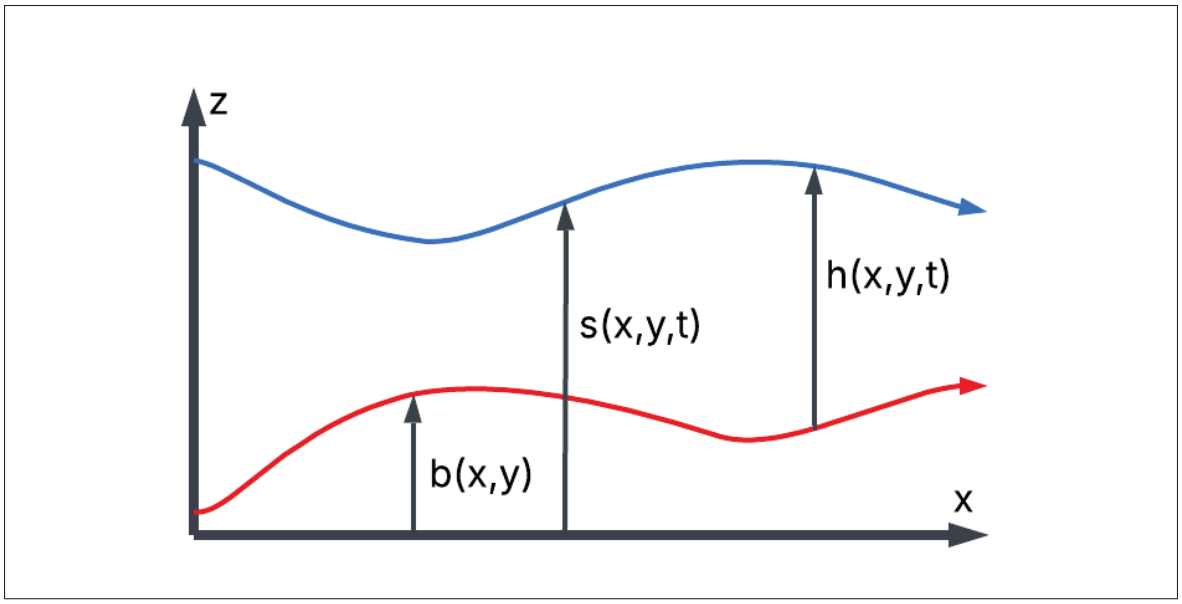


Figure 2.2 Illustration de la notation pour les équations de Saint-Venant

h peut donc prendre des valeurs de zéro ou proches de zéro. Cela produit donc des divisions par zéro qui font planter l'entraînement ou bien des valeurs de perte élevées qui ralentissent ou même inhibent l'apprentissage. Cela peut être évité en ajoutant une petite quantité d'eau de manière constante pour que h demeure plus grand que zéro, soit $h = h + \epsilon$. Cependant, cette option n'est pas optimale puisqu'elle augmente artificiellement notre quantité d'eau et affecte la conservation de la masse.

2.2.2 Équations de Saint-Venant non-conservatrices

Cela dit, la forme non-conservatrice des SWE est préférable puisqu'elle prend en entrée les coordonnés spatio-temporelles $(\boldsymbol{x}, t) \in \mathbb{R} \times [0, T]$ et retourne directement en sortie les prédictions d'hauteur d'eau et de vitesse $(h, \bar{\boldsymbol{u}})$. Les équations de Saint-Venant sous leur forme non-conservatrice sont définies ainsi :

Équation de continuité :

$$\frac{\partial h}{\partial t} + \bar{u}\frac{\partial h}{\partial x} + \bar{v}\frac{\partial h}{\partial y} + h\left(\frac{\partial \bar{u}}{\partial x} + \frac{\partial \bar{v}}{\partial y}\right) = 0 \tag{2.13}$$

Équation de momentum en x :

$$\frac{\partial \bar{u}}{\partial t} + \bar{u}\frac{\partial \bar{u}}{\partial x} + \bar{v}\frac{\partial \bar{u}}{\partial y} + g\frac{\partial h}{\partial x} = -g\frac{\partial b}{\partial x} + S_{f,x} \tag{2.14}$$

Équation de momentum en y :

$$\frac{\partial \bar{v}}{\partial t} + \bar{u}\frac{\partial \bar{v}}{\partial x} + \bar{v}\frac{\partial \bar{v}}{\partial y} + g\frac{\partial h}{\partial y} = -g\frac{\partial b}{\partial y} + S_{f,y} \tag{2.15}$$

Les termes de friction $S_{f,x}$ et $S_{f,x}$ sont définis par :

$$S_{f,x} = -\frac{g(n^2\bar{u}\sqrt{\bar{u}^2 + \bar{v}^2})}{h^{4/3}}, \tag{2.16}$$

$$S_{f,y} = -\frac{g(n^2\bar{v}\sqrt{\bar{u}^2 + \bar{v}^2})}{h^{4/3}}. \tag{2.17}$$

où n est le nombre de Manning.

2.2.3 Équations de Saint-Venant adimensionnelles

Dans le domaine de la CFD, les équations adimensionnelles sont privilégiées puisqu'elles simplifient les équations en supprimant la dépendance à des unités spécifiques et en les rendant universellement applicables à différentes échelles physiques. Cette approche améliore également la stabilité numérique de l'entraînement du PINN en mettant à l'échelle les variables à des ordres de grandeur similaires. Utiliser les équations adimensionnelles offre les mêmes bénéfices que la mise à l'échelle des données, une étape cruciale pour l'entraînement de réseaux de neurones.

Pour adimensionnaliser les SWE non-conservatrices, nous introduisons les échelles caractéristiques suivantes :

- U^* : vitesse caractéristique,
- L^* : longueur caractéristique,

- H^* : hauteur caractéristique,
- $T^* = \frac{L^*}{U^*}$: temps caractéristique.

Les variables adimensionnelles sont définies comme suit :

$$x^* = \frac{x}{L^*}, \quad y^* = \frac{y}{L^*}, \quad t^* = \frac{t}{T^*},$$

$$\bar{u}^* = \frac{\bar{u}}{U^*}, \quad \bar{v}^* = \frac{\bar{v}}{U^*}, \quad h^* = \frac{h}{H^*}, \quad b^* = \frac{b}{H^*}.$$

Le nombre de Froude est défini par :

$$\mathrm{Fr} = \frac{U^*}{\sqrt{gH^*}}.$$

2.2.3.1 Adimensionnalisation de l'équation de continuité

En substituant les variables adimensionnelles dans l'équation de continuité (2.13), nous obtenons :

$$\frac{\partial(H^*h^*)}{\partial(t^*T^*)} + (U^*\bar{u}^*)\frac{\partial(H^*h^*)}{\partial(x^*L^*)} + (U^*\bar{v}^*)\frac{\partial(H^*h^*)}{\partial(y^*L^*)} + (H^*h^*)\left(\frac{\partial(U^*\bar{u}^*)}{\partial(x^*L^*)} + \frac{\partial(U^*\bar{v}^*)}{\partial(y^*L^*)}\right) = 0.$$

En simplifiant et en divisant par $\frac{U^*H^*}{L^*}$, nous obtenons l'équation de continuité adimensionnelle :

$$\frac{\partial h^*}{\partial t^*} + \bar{u}^*\frac{\partial h^*}{\partial x^*} + \bar{v}^*\frac{\partial h^*}{\partial y^*} + h^*\left(\frac{\partial \bar{u}^*}{\partial x^*} + \frac{\partial \bar{v}^*}{\partial y^*}\right) = 0.$$

2.2.3.2 Adimensionnalisation de l'équation de quantité de mouvement en x

En substituant les variables adimensionnelles dans l'équation de quantité de mouvement en x (2.14), nous obtenons :

$$\frac{\partial(U^*\bar{u}^*)}{\partial(t^*T^*)} + (U^*\bar{u}^*)\frac{\partial(U^*\bar{u}^*)}{\partial(x^*L^*)} + (U^*\bar{v}^*)\frac{\partial(U^*\bar{u}^*)}{\partial(y^*L^*)} + g\frac{\partial(H^*h^*)}{\partial(x^*L^*)} = -g\frac{\partial(H^*b^*)}{\partial(x^*L^*)} - \frac{gn^2(U^*\bar{u}^*)\sqrt{(U^*\bar{u}^*)^2 + (U^*\bar{v}^*)^2}}{(H^*h^*)^{4/3}}.$$

En simplifiant et en divisant par $\frac{U^{*2}}{L^*}$, nous obtenons :

$$\frac{\partial\bar{u}^*}{\partial t^*} + \bar{u}^*\frac{\partial\bar{u}^*}{\partial x^*} + \bar{v}^*\frac{\partial\bar{u}^*}{\partial y^*} + \frac{gH^*}{U^{*2}}\frac{\partial h^*}{\partial x^*} = -\frac{gH^*}{U^{*2}}\frac{\partial b^*}{\partial x^*} + \lambda\frac{\bar{u}^*\sqrt{\bar{u}^{*2} + \bar{v}^{*2}}}{h^{*\frac{4}{3}}},$$

avec

$$\lambda = \frac{-gn^2U^{*2}}{\frac{U^{*2}}{L^*}H^{*\frac{4}{3}}} = \frac{-gn^2L^*}{H^{*\frac{4}{3}}}.$$

En utilisant le nombre de Froude $\mathrm{Fr} = \frac{U^*}{\sqrt{gH^*}}$, cette équation devient :

$$\frac{\partial\bar{u}^*}{\partial t^*} + \bar{u}^*\frac{\partial\bar{u}^*}{\partial x^*} + \bar{v}^*\frac{\partial\bar{u}^*}{\partial y^*} + \frac{1}{\mathrm{Fr}^2}\frac{\partial h^*}{\partial x^*} = -\frac{1}{\mathrm{Fr}^2}\frac{\partial b^*}{\partial x^*} + \lambda\frac{\bar{u}^*\sqrt{\bar{u}^{*2} + \bar{v}^{*2}}}{h^{*\frac{4}{3}}}$$

2.2.3.3 Équations adimensionnelles finales

Les équations de Saint-Venant adimensionnelles sous forme non conservatrices sont :

1. **Équation de continuité :**

$$\frac{\partial h^*}{\partial t^*} + \bar{u}^*\frac{\partial h^*}{\partial x^*} + \bar{v}^*\frac{\partial h^*}{\partial y^*} + h^*\left(\frac{\partial\bar{u}^*}{\partial x^*} + \frac{\partial\bar{v}^*}{\partial y^*}\right) = 0. \tag{2.18}$$

2. **Équation de quantité de mouvement en x :**

$$\frac{\partial \bar{u}^*}{\partial t^*} + \bar{u}^* \frac{\partial \bar{u}^*}{\partial x^*} + \bar{v}^* \frac{\partial \bar{u}^*}{\partial y^*} + \frac{1}{\mathrm{Fr}^2} \frac{\partial h^*}{\partial x^*} = -\frac{1}{\mathrm{Fr}^2} \frac{\partial b^*}{\partial x^*} + \lambda \frac{\bar{u}^* \sqrt{\bar{u}^{*2} + \bar{v}^{*2}}}{h^{*\frac{4}{3}}}, \tag{2.19}$$

où :

$$\lambda = \frac{-gn^2 L^*}{H^{*\frac{4}{3}}}.$$

3. **Équation de quantité de mouvement en y :**

$$\frac{\partial \bar{v}^*}{\partial t^*} + \bar{u}^* \frac{\partial \bar{v}^*}{\partial x^*} + \bar{v}^* \frac{\partial \bar{v}^*}{\partial y^*} + \frac{1}{\mathrm{Fr}^2} \frac{\partial h^*}{\partial y^*} = -\frac{1}{\mathrm{Fr}^2} \frac{\partial b^*}{\partial x^*} + \lambda \frac{\bar{v}^* \sqrt{\bar{u}^{*2} + \bar{v}^{*2}}}{h^{*\frac{4}{3}}}, \tag{2.20}$$

où :

$$\lambda = \frac{-gn^2 L^*}{H^{*\frac{4}{3}}}.$$

2.3 Réseaux de neurones informés par la physique (PINNs)

Dans la section suivante, nous nous appuyons sur la discussion précédente pour présenter la stratégie adoptée pour résoudre des problèmes complexes de mécanique des fluides à l'aide de PINNs.

2.3.1 PINNs standards

Considérons une EDP générale de la forme :

$$\boldsymbol{u}_t(\boldsymbol{x}, t) + \mathcal{N}[\boldsymbol{u}](\boldsymbol{x}, t) = 0, \quad t \in (0, T],\ \boldsymbol{x} \in \Omega, \tag{2.21}$$

où $\boldsymbol{u}$ est la solution inconnue de l'EDP, $\mathcal{N}[\cdot]$ représente un opérateur différentiel linéaire ou non linéaire, t indique la coordonnée temporelle, $\boldsymbol{x}$ représente les coordonnées spatiales, $\boldsymbol{u}_t$ représente la dérivée partielle de $\boldsymbol{u}$ par rapport au temps, Ω est un sous-ensemble de $\mathbb{R}^D$ définissant le domaine spatial, et T est le temps de simulation final. L'EDP (2.21) est soumise aux conditions

initiales et aux conditions limites suivantes :

$$\boldsymbol{u}(\boldsymbol{x}, 0) = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{2.22}$$

$$\mathcal{B}[\boldsymbol{u}] = 0, \quad t \in (0, T], \quad \boldsymbol{x} \in \partial\Omega, \tag{2.23}$$

où g est la condition initiale, et $\mathcal{B}[\cdot]$ est un opérateur de frontière (Dirichlet, Neumann, Robin, périodique). Nous approximons la solution de l'EDP (2.21) à l'aide d'un réseau de neurones profond $\boldsymbol{u}_\theta(t, \boldsymbol{x})$, où θ représente l'ensemble des paramètres pouvant être entraînés. Le cadre PINN trouve la solution en minimisant une fonction de perte composite :

$$\mathcal{L}(\theta) = \lambda_{ic}\mathcal{L}_{ic}(\theta) + \lambda_{bc}\mathcal{L}_{bc}(\theta) + \lambda_r\mathcal{L}_r(\theta), \tag{2.24}$$

où chaque composante applique une contrainte spécifique du problème. Le terme $\mathcal{L}_{ic}(\theta)$ correspond à la condition initiale, $\mathcal{L}_{bc}(\theta)$ représente la condition limite, et $\mathcal{L}_r(\theta)$ garantit que les lois physiques régissant le système sont satisfaites. Les poids λ_{ic}, λ_{bc} et λ_r équilibrent l'influence de ces contraintes dans la fonction de perte globale, orientant le réseau vers l'apprentissage d'une solution qui respecte toutes les exigences du problème.

La perte de la condition initiale est donnée par :

$$\mathcal{L}_{ic}(\theta) = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} |\boldsymbol{u}_\theta(x_{ic}^i, 0) - g(x_{ic}^i)|^2, \tag{2.25}$$

La perte de la condition frontière est donnée par :

$$\mathcal{L}_{bc}(\theta) = \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} |\mathcal{B}[\boldsymbol{u}_\theta](x_{bc}^i, t_{bc}^i)|^2, \tag{2.26}$$

La perte résiduelle garantit le respect de l'EDP et est définie comme suit :

$$\mathcal{L}_r(\theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} |\mathcal{R}_\theta(x_r^i, t_r^i)|^2, \tag{2.27}$$

où le résidu $\mathcal{R}_\theta(x,t)$ est donné par :

$$\mathcal{R}_\theta(x,t) = \frac{\partial \boldsymbol{u}_\theta}{\partial t}(x_r,t_r) + \mathcal{N}[\boldsymbol{u}_\theta](x_r,t_r). \tag{2.28}$$

Les termes $\{x_{ic}^i\}_{i=1}^{N_{ic}}$, $\{t_{bc}^i, x_{bc}^i\}_{i=1}^{N_{bc}}$, et $\{t_r^i, x_r^i\}_{i=1}^{N_r}$ représentent des ensembles de points de collocation échantillonnés dans le domaine de calcul pour évaluer la condition initiale, la condition limite et les pertes résiduelles, respectivement. La différenciation automatique est utilisée pour calculer les dérivées partielles requises, ce qui permet une évaluation efficace de $\mathcal{R}_\theta(t,x)$. Cette approche garantit que le réseau de neurones prédit des solutions qui satisfont aux conditions initiales et aux conditions frontières, ainsi qu'à l'EDP gouvernante.

En général, le solveur PINN standard rencontre plusieurs difficultés lorsqu'il est appliqué à certains types de problèmes. Ces défis comprennent la violation de la causalité temporelle Wang *et al.* (2022a), les difficultés à gérer les simulations à long terme Penwarden, Jagtap, Zhe, Karniadakis & Kirby (2023), et divers autres modes d'échec. Comme nous l'avons mentionné dans l'introduction, ces limitations nécessitent des améliorations de l'architecture PINN standard. Plusieurs améliorations ont été proposées pour résoudre ces problèmes et améliorer la robustesse et l'efficacité des PINNs. Dans ce qui suit, nous présentons le solveur PINN modifié adopté dans notre étude, qui intègre certaines de ces avancées.

2.3.2 PINNs améliorés

Cette section présente les principales améliorations par rapport au PINN standard.

2.3.2.1 PirateNets

Dans la plupart des études PINN, le perceptron multicouche (MLP) est couramment utilisé comme architecture d'apprentissage profond. Les MLP sont généralement efficaces pour les problèmes simples, mais des améliorations ont été proposées pour renforcer leurs capacités. Par exemple, Wang et al. ont proposé un MLP modifié Wang *et al.* (2021a) qui introduit deux encodeurs dans les coordonnées d'entrée, ce qui améliore généralement les résultats avec PINN.

Cependant, à mesure que les MLP s'approfondissent pour traiter des solutions complexes, ils sont souvent confrontés à des inefficacités d'apprentissage significatives. Pour relever ce défi, Wang et al. Wang *et al.* (2025) ont introduit les Physics-Informed Residual Adaptive Netowrks, ou *PirateNets*, une architecture qui exploite les connexions résiduelles adaptatives pour améliorer les performances d'apprentissage. Cette approche atténue les inefficacités de l'apprentissage en commençant par un réseau artificiellement peu profond et en augmentant progressivement sa profondeur au cours de l'apprentissage. Les connexions de saut adaptatives (adaptive skip connections) au sein du bloc résiduel permettent une optimisation efficace et améliorent la stabilité de l'apprentissage. Un schéma de la passe avant *PirateNet* est présenté dans la Figure 2.3 et est décrit en détail ci-dessous.

Les coordonnées d'entrée $\boldsymbol{x} = (x, t)$ sont d'abord transformées en un espace de caractéristiques à plus haute dimension à l'aide d'une technique d'intégration des coordonnées, telle que les caractéristiques de Fourier aléatoires (Random Fourier Features) Tancik *et al.* (2020) :

$$\Phi(\boldsymbol{x}) = [\cos(\mathbf{F}\boldsymbol{x}), \sin(\mathbf{F}\boldsymbol{x})]^T, \tag{2.29}$$

où chaque élément de la matrice $\mathbf{F}$ est échantillonné à partir d'une distribution gaussienne $\mathcal{N}(0, \sigma^2)$, et σ est un hyperparamètre. Les caractéristiques de Fourier aléatoires sont particulièrement efficaces pour réduire le biais spectral, ce qui permet d'améliorer les prédictions pour les composantes à haute fréquence dans la solution Wang *et al.* (2021b).

Les coordonnées intégrées $\Phi(\boldsymbol{x})$ sont ensuite envoyées à deux couches denses agissant comme des portes dans chaque bloc résiduel du modèle :

$$U = \sigma(\boldsymbol{W}_1\Phi(\boldsymbol{x}) + \boldsymbol{b}_1), \qquad V = \sigma(\boldsymbol{W}_2\Phi(\boldsymbol{x}) + \boldsymbol{b}_2), \tag{2.30}$$

où σ est une fonction d'activation ponctuelle. Soit $\boldsymbol{x}^{(l)}$ l'entrée du l-ième bloc résiduel où $1 \leq l \leq L - 1$. Les portes sont motivés par l'étude des pathologies du flux de gradient dans les PINNs Wang *et al.* (2021a). Le passage en avant d'un bloc résiduel *PirateNet* peut maintenant

être défini comme suit :

$$
\begin{aligned}
f^{(l)} &= \sigma(\boldsymbol{W}_1^{(l)}\boldsymbol{x}^{(l)} + \boldsymbol{b}_1^{(l)}), && \text{(Dense layer)} \\
z_1^{(l)} &= f^{(l)} \odot U + (1 - f^{(l)}) \odot V, && \text{(Gating operation)} \\
g^{(l)} &= \sigma(\boldsymbol{W}_2^{(l)} z_1^{(l)} + \boldsymbol{b}_2^{(l)}), && \text{(Dense layer)} \\
z_2^{(l)} &= g^{(l)} \odot U + (1 - g^{(l)}) \odot V, && \text{(Gating operation)} \\
h^{(l)} &= \sigma(\boldsymbol{W}_3^{(l)} z_2^{(l)} + \boldsymbol{b}_3^{(l)}), && \text{(Dense layer)} \\
\boldsymbol{x}^{(l+1)} &= \alpha^{(l)} h^{(l)} + (1 - \alpha^{(l)})\boldsymbol{x}^{(l)}, && \text{(Adaptive skip connection)}
\end{aligned}
\tag{2.31}
$$

où $\odot$ est un produit de Hadamard et $\alpha^{(l)}$ sont des paramètres entraînables pour chaque bloc résiduel. Tous les poids sont initialisés avec le schéma de Glorot Glorot & Bengio (2010) tandis que les biais, b, sont initialisés à zéro. La sortie d'un PirateNet avec L blocs est finalement donnée par $\boldsymbol{W}^{(L+1)}\boldsymbol{x}^{(L)}$.

Un élément clé de *PirateNet* est la connexion de saut adaptative, caractérisée par son paramètre d'entraînement $\alpha^{(l)}$, qui contrôle le degré de non-linéarité introduit par le l-ième bloc. Au départ, $\alpha^{(l)}$ est fixé à zéro, ce qui réduit effectivement le bloc à une cartographie d'identité. Au cours de l'entraînement, $\alpha^{(l)}$ augmente progressivement, ce qui permet au réseau d'introduire dynamiquement des transformations non linéaires uniquement lorsqu'elles commencent à contribuer de manière significative à la solution. Cette stratégie d'activation progressive atténue les difficultés associées à l'initialisation de réseaux plus profonds, en améliorant la stabilité et l'efficacité de l'apprentissage Wang *et al.* (2025).

Un autre élément crucial de l'architecture *PirateNet* est l'approche d'initialisation informée par la physique (physics-informed initialization), conçue pour fournir un point de départ optimal pour l'entraînement en incorporant les données disponibles Wang *et al.* (2025). Comme indiqué précédemment, le modèle *PirateNet* est représenté comme une combinaison linéaire de fonctions de base de Fourier lors de l'initialisation, exprimée comme $\boldsymbol{W}^{(L+1)}\Phi(\boldsymbol{x})$. Cela implique que la matrice de poids $\boldsymbol{W}^{(L+1)}$ de la couche finale peut être initialisée en résolvant un problème de

moindres carrés :

$$\boldsymbol{W}^{(L+1)} = \arg\min_{\boldsymbol{W}} \|\boldsymbol{W}\Phi(\boldsymbol{x}) - Y\|_2^2, \tag{2.32}$$

où Y représente certaines données disponibles, telles que les conditions initiales, les données aux frontières ou les résultats des modèles de substitution. Cette stratégie d'initialisation élimine les problèmes courants associés à l'initialisation aléatoire, tels que l'instabilité ou la lenteur de la convergence. En alignant la sortie initiale du réseau sur une approximation physiquement significative de la solution, elle accélère l'entraînement et améliore les performances globales.

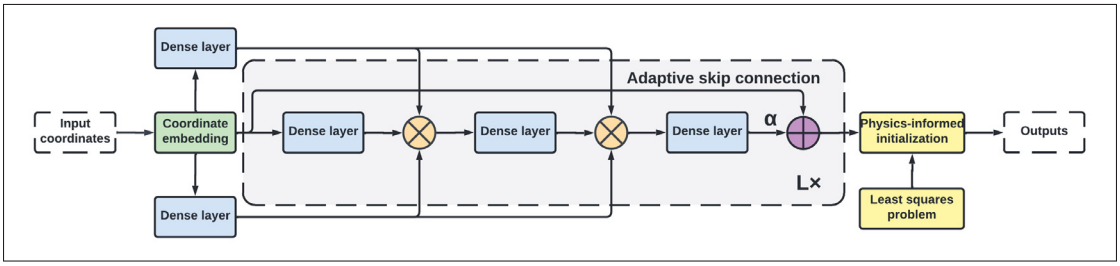


Figure 2.3 Architecture *PirateNet* Wang *et al.* (2025). Les coordonnées d'entrée sont transmises à une intégration de coordonnées comme les caractéristiques de Fourier aléatoires avant de passer par deux couches denses et le bloc résiduel. Le bloc résiduel est contenu dans la zone grise de la figure et est répété L fois. Les couches denses initiales sont transmises à deux opérations d'adaptation en orange dans le bloc résiduel. La connexion de saut adaptative est affectée par le paramètre entraînable α et contrôle la quantité d'informations transmises directement à partir de l'intégration des coordonnées. Enfin, un schéma d'initialisation basé sur la physique est appliqué avant la sortie

2.3.2.2 Factorisation aléatoire des poids (RWF)

La factorisation des poids aléatoires (RWF) est une technique simple, mais efficace proposée par Wang *et al.* (2022b), dont il a été démontré qu'elle permettait d'accélérer et d'améliorer l'entraînement des réseaux de neurones. La méthode RWF factorise les poids associés à chaque neurone du réseau comme suit :

$$\boldsymbol{w}^{(k,l)} = \boldsymbol{s}^{(k,l)}\boldsymbol{v}^{(k,l)}, \quad k = 1, 2, ..., d_l, \quad l = 1, 2, ..., L+1, \tag{2.33}$$

où $\boldsymbol{w}^{(k,l)} \in \mathbb{R}^{d_l-1}$ représente la k-ième ligne de la matrice de poids $\boldsymbol{W}^{(l)}$, $\boldsymbol{s}^{(k,l)} \in \mathbb{R}$ est un facteur d'échelle entraînable spécifique à chaque neurone, et $\boldsymbol{v}^{(k,l)} \in \mathbb{R}^{d_{l-1}}$. Par conséquent, le RWF peut être exprimé sous forme matricielle comme suit :

$$\boldsymbol{W}^{(l)} = \text{diag}(\boldsymbol{s}^{(l)})\boldsymbol{V}^{(l)}, \quad l = 1, 2, ..., L+1, \tag{2.34}$$

où $\boldsymbol{s}^{(l)} \in \mathbb{R}^{d_l}$ représente le vecteur des facteurs d'échelle pour la couche l.

Les détails de l'implémentation du RWF sont les suivants Wang *et al.* (2023). Tout d'abord, les paramètres du réseau neuronal sont initialisés à l'aide d'un schéma standard, tel que l'initialisation de Glorot Glorot & Bengio (2010). Ensuite, pour chaque matrice de poids $\boldsymbol{W}$, un vecteur d'échelle $\boldsymbol{s}$ est échantillonné à partir d'une distribution normale multivariée $\mathcal{N}(\mu, \sigma I)$, avec μ et σ comme hyperparamètres, et I comme matrice d'identité. Lors de l'initialisation, la matrice de poids est factorisée comme suit : $\boldsymbol{W} = \text{diag}(\exp(\boldsymbol{s})) \cdot \boldsymbol{V}$, où $\boldsymbol{V}$ représente le paramètre factorisé restant. Enfin, l'optimisation par descente de gradient est appliquée directement aux nouveaux paramètres, $\boldsymbol{s}$ et $\boldsymbol{V}$. Selon Wang *et al.* (2022b), une sélection appropriée d'hyperparamètres RWF peut réduire la distance entre l'initialisation et le minimum global dans l'espace des paramètres factorisés par rapport à l'espace des paramètres d'origine dans leurs espaces de pertes respectifs. En outre, Wang *et al.* (2023) a démontré par le biais d'expériences numériques que, dans le cas des PINN, la technique RWF améliore à la fois la précision et les performances d'entraînement sur une gamme de problèmes de références basée sur des EDP. Une représentation graphique de la RWF est présente à la Figure 2.4 pour un réseau avec un seul poids.

2.3.2.3 Respect de la causalité temporelle

Récemment, Wang *et al.* (2022a) ont démontré que les PINN peuvent violer la causalité temporelle lors de la résolution d'EDP dépendantes du temps, ce qui peut conduire à des prédictions incorrectes pour les problèmes qui dépendent fortement de la satisfaction de la condition initiale (par exemple, les modèles de champ de phase). Les auteurs ont montré que, pour certaines références, les PINN ont tendance à donner la priorité à la minimisation du résidu

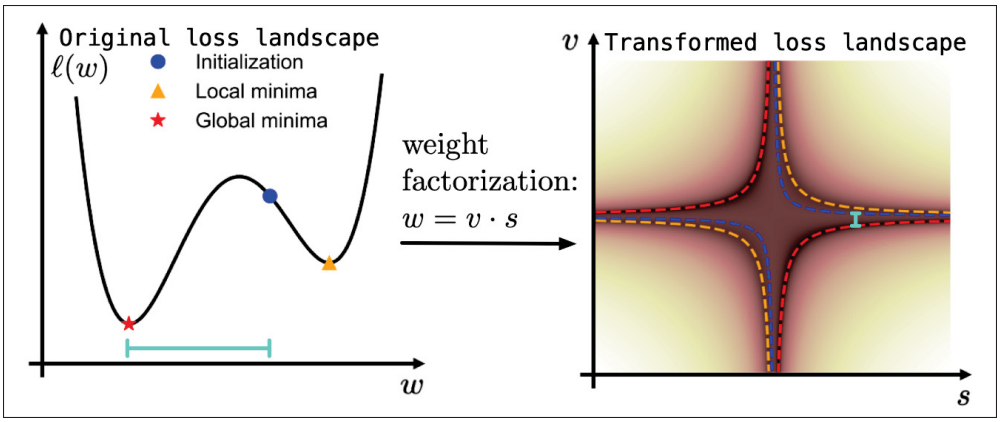


Figure 2.4 Factorisation aléatoire des poids,
tiré de (Wang *et al.*, 2022b, p. 3)

de l'EDP à des moments ultérieurs avant d'apprendre avec précision les solutions à des moments antérieurs. Pour résoudre ce problème, une approche d'entraînement causale temporelle a été proposée dans Wang *et al.* (2022a).

La stratégie d'entraînement causale temporelle est décrite comme suit. Tout d'abord, le domaine temporel $[0, T]$ divisé en M segments séquentiels, dénotés comme $0 = t_1 < \cdots < t_M = T$. Ensuite, la fonction de perte $\mathcal{L}_r(t_i, \theta)$ est définie comme la perte résiduelle de l'EDP dans le i-ème segment temporel :

$$\mathcal{L}_r(t_i, \theta) = \frac{1}{N_x} \sum_{j=1}^{N_x} \left| \mathcal{R}_\theta(x_j, t_i) \right|^2, \tag{2.35}$$

où $(x_j)_{j=1}^{N_x} \in \Omega$ représentent des points échantillonnés dans le domaine de calcul.

La perte résiduelle de l'EDP d'origine peut alors être réécrite comme suit :

$$\mathcal{L}_r(\theta) = \frac{1}{M} \sum_{i=1}^{M} w_i \mathcal{L}_r(t_i, \theta), \tag{2.36}$$

où w_i sont des poids conçus pour être importants uniquement lorsque tous les résidus antérieurs $\{\mathcal{L}_r(t_k, \theta)\}_{k=1}^{i-1}$ sont suffisamment minimisés afin de garantir la causalité temporelle séquentielle.

Pour ce faire, les poids sont définis comme suit :

$$w_i = \exp\left(-\varepsilon \sum_{k=1}^{i-1} \mathcal{L}_r(t_k, \theta)\right), \quad \text{for } i = 2, 3, ..., M, \tag{2.37}$$

où ε est un paramètre de causalité qui contrôle la pente des poids w_i. Ici, les poids w_i sont inversement exponentiellement proportionnels à la perte résiduelle cumulée des pas de temps précédents.

La perte résiduelle pondérée peut maintenant s'écrire comme suit :

$$\mathcal{L}_r(\theta) = \mathcal{L}_r(t_0, \theta) + \frac{1}{M} \sum_{i=2}^{M} \exp\left(-\varepsilon \sum_{k=1}^{i-1} \mathcal{L}_r(t_k, \theta)\right) \mathcal{L}_r^i(\theta), \tag{2.38}$$

où $\mathcal{L}_r(t_0, \theta)$ est la perte de condition initiale.

L'équation (2.38) garantit que la perte résiduelle de l'EDP dans un segment temporel donné n'est pas minimisée tant que les résidus de tous les segments temporels précédents n'ont pas été suffisamment réduits. Cette approche permet à la solution de l'EDP d'évoluer d'une manière temporellement causale, s'alignant sur la propagation physique de l'information dans les systèmes dépendant du temps. Le schéma de causalité temporelle s'est avéré efficace pour l'apprentissage des EDP rigides, en particulier celles caractérisées par une forte non-linéarité Wang *et al.* (2023, 2022a). Nous démontrerons que l'intégration de cette approche permet de résoudre des problèmes level set complexes impliquant une grande déformation de l'interface en plus des problèmes de SWE.

2.3.2.4 Équilibrage des poids de la fonction de perte

L'une des difficultés rencontrées lors de l'apprentissage des PINN avec une fonction de perte composite est la différence d'échelle de chaque terme de la fonction de perte lors de la minimisation du résidu de l'EDP. Ce déséquilibre peut entraîner une instabilité lors de la descente stochastique du gradient. Étant donné que les pertes ne peuvent pas être normalisées

lors d'une étape de prétraitement, la normalisation doit être effectuée pendant l'entraînement. Une approche rudimentaire de ce problème consisterait à attribuer manuellement des poids à chaque terme de perte avant l'apprentissage. Toutefois, cette approche n'est pas pratique, car les poids dépendent fortement du problème à résoudre. Les poids pourraient être optimisés à l'aide d'un schéma d'optimisation hyperparamétrique, mais cette approche est coûteuse en termes de calcul et parfois impossible si aucun ensemble de données de validation n'est disponible. À cette fin, quelques approches ont été développées pour permettre un rééquilibrage automatique des termes de perte pendant l'entraînement, comme la normalisation du gradient et les schémas de neural tangent kernel (NTK).

La méthode de normalisation du gradient Wang *et al.* (2023) consiste à ajuster le poids de chaque terme de perte de sorte que les amplitudes de leurs gradients contribuent de manière égale à la perte totale pendant l'optimisation. Nous commençons par calculer les poids globaux $\hat{\lambda}$ comme suit :

$$\begin{aligned}
\hat{\lambda}_{ic} &= \frac{||\nabla_\theta \mathcal{L}_{ic}(\theta)|| + ||\nabla_\theta \mathcal{L}_{bc}(\theta)|| + ||\nabla_\theta \mathcal{L}_r(\theta)||}{||\nabla_\theta \mathcal{L}_{ic}(\theta)||}, \\
\hat{\lambda}_{bc} &= \frac{||\nabla_\theta \mathcal{L}_{ic}(\theta)|| + ||\nabla_\theta \mathcal{L}_{bc}(\theta)|| + ||\nabla_\theta \mathcal{L}_r(\theta)||}{||\nabla_\theta \mathcal{L}_{bc}(\theta)||}, \\
\hat{\lambda}_r &= \frac{||\nabla_\theta \mathcal{L}_{ic}(\theta)|| + ||\nabla_\theta \mathcal{L}_{bc}(\theta)|| + ||\nabla_\theta \mathcal{L}_r(\theta)||}{||\nabla_\theta \mathcal{L}_r(\theta)||},
\end{aligned} \tag{2.39}$$

où $||\cdot||$ représente la norme L^2. Si d'autres termes de perte sont présents, ils doivent être ajoutés aux calculs des poids globaux. L'équation (2.39) établit la relation suivante :

$$||\hat{\lambda}_{ic} \nabla_\theta \mathcal{L}_{ic}(\theta)|| = ||\hat{\lambda}_{bc} \nabla_\theta \mathcal{L}_{bc}(\theta)|| = ||\hat{\lambda}_r \nabla_\theta \mathcal{L}_r(\theta)||. \tag{2.40}$$

Cela garantit que les gradients de tous les termes de perte pondérés ont des amplitudes égales, empêchant le modèle de donner la priorité à la minimisation de termes spécifiques par rapport à d'autres au cours de l'entraînement. Les poids peuvent ensuite être mis à jour à l'aide d'une moyenne mobile, comme suit :

$$\lambda_{new} = \alpha \lambda_{old} + (1 - \alpha) \hat{\lambda}_{new}. \tag{2.41}$$

où $\alpha = 0,9$ est un hyperparamètre quantifiant l'influence des anciens poids sur les nouveaux. La mise à jour de la pondération est effectuée à une fréquence spécifiée par l'utilisateur, ce qui rend la charge de calcul négligeable. Sauf indication contraire, cette fréquence a été fixée à toutes les 500 itérations pour nos expériences.

La méthode NTK s'appuie sur les matrices NTK des PINN associées à chaque terme de perte pour déterminer les poids Wang *et al.* (2022c, 2023). La matrice NTK saisit la relation entre les paramètres du réseau et ses résultats, sa trace (somme des valeurs propres) indiquant le taux de convergence d'un terme de perte. En calculant la trace pour chaque matrice NTK, les poids sont définis de manière à garantir des taux de convergence comparables pour tous les termes de perte. Cette méthode permet des mises à jour de poids plus stables qu'avec la normalisation du gradient, mais elle est plus coûteuse en termes de calcul tel que démontré par Wang *et al.* (2023). C'est pourquoi nos expériences ont été calculées à l'aide de la méthode de normalisation du gradient.

2.3.2.5 Apprentissage séquence par séquence (S2S)

Les PINN traditionnels abordent l'apprentissage des EDP pour l'ensemble de l'espace-temps, ce qui signifie qu'ils tentent de prédire la solution pour tous les emplacements à tous les points dans le temps. Cependant, de nombreux phénomènes physiques présentent de fortes dépendances séquentielles où la sortie à un pas de temps donné dépend fortement des pas de temps précédents. Dans ces cas, l'apprentissage de séquence à séquence (S2S) de Krishnapriyan *et al.* (2021) pourrait fournir de meilleurs résultats.

Dans l'approche S2S, le domaine temporel $[0, T]$ est divisé en plusieurs fenêtres temporelles, $I_i = [t_i, t_{i+1}]$, où $t_i = i\Delta T$. Une stratégie d'entraînement séquentielle est employée. À chaque étape, un modèle PINN est formé sur le domaine $\Omega \times I_i$. La condition initiale de ce domaine au moment t_i est obtenue à partir des prédictions du modèle PINN formé précédemment. Pour la première fenêtre temporelle, la condition initiale exacte est utilisée. Cependant, pour les fenêtres

temporelles suivantes, les conditions initiales sont dérivées des prédictions du modèle PINN précédent.

L'approche S2S s'est avérée être une stratégie efficace pour traiter les EDP rigides, telles que les modèles du quatrième ordre Mattey & Ghosh (2022) et les phénomènes de transport avec des valeurs à grande vitesse Penwarden *et al.* (2023). Inspirés par ces études, nous adoptons l'approche S2S pour entraîner notre modèle PINN pour les problèmes level set et SWE, dans le but d'améliorer la précision.

2.3.2.6 Distribution adaptative basée sur les résiduels (RAD)

La distribution adaptative des points de collocations basée sur les résiduels (Residual-based Adaptive distribution, RAD) Wu *et al.* (2023) est la première de deux méthodes testées dans ce mémoire qui affecte directement les points de collocations. La méthode échantillonne dynamiquement les points pendant l'entraînement en mettant l'accent sur les régions où le résidu de l'EDP est important, orientant ainsi le réseau vers les zones plus difficiles à approximer.

L'idée principale de RAD est d'échantillonner les points de collocation selon une fonction de densité de probabilité (PDF) proportionnelle aux résidus normalisés de l'EDP. Cette approche contraste avec les stratégies d'échantillonnage uniforme, qui traitent toutes les régions du domaine de manière égale, indépendamment de l'erreur locale.

Soit $\varepsilon(\boldsymbol{x}) = |f(\boldsymbol{x}; \hat{u}(\boldsymbol{x}))|$ la magnitude du résidu de l'EDP évalué en un point $\boldsymbol{x}$ du domaine Ω. La méthode RAD définit une PDF comme suit :

$$p(\boldsymbol{x}) \propto \frac{\varepsilon^k(\boldsymbol{x})}{\mathbb{E}[\varepsilon^k(\boldsymbol{x})]} + c, \tag{2.42}$$

où $k \geq 0$ et $c \geq 0$ sont des hyperparamètres, et $\mathbb{E}[\varepsilon^k(\boldsymbol{x})]$ désigne l'espérance de $\varepsilon^k(\boldsymbol{x})$, qui peut être approximée par intégration de Monte Carlo.

Les hyperparamètres k et c sont essentiels pour façonner la distribution d'échantillonnage. Augmenter k concentre davantage l'échantillonnage sur les régions à fort résidu, tandis qu'une augmentation de c favorise une distribution plus uniforme. Leurs valeurs optimales dépendent du problème, $k = 1$ et $c = 1$ constituant un choix par défaut robuste dans de nombreux cas.

L'algorithme RAD utilisé dans notre étude est résumé dans l'algorithme 2.1. Notons que cet algorithme est conçu pour les basses dimensions ; d'autres méthodes sont proposées dans Wu *et al.* (2023) pour les dimensions plus élevées.

Algorithme 2.1 Échantillonnage RAD des points de collocation

for *toutes les itérations d'entraînement* **do**
  À partir d'un échantillonnage aléatoire, générer un ensemble dense de points candidats $\mathcal{S}_0$;
  Calculer $p(\boldsymbol{x})$ pour tous les points de $\mathcal{S}_0$;
  Définir une fonction de masse de probabilité $\tilde{p}(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{A}$ avec la constante de normalisation $A = \sum_{\boldsymbol{x} \in \mathcal{S}_0} p(\boldsymbol{x})$;
  Échantillonner un sous-ensemble de points de collocation à partir de $\mathcal{S}_0$ selon $\tilde{p}(\boldsymbol{x})$;
  for κ *itérations avant le rééchantillonnage* **do**
    Entraîner le PINN.
  end for
end for

Pour que cette méthode fonctionne, il est nécessaire d'échantillonner un ensemble de points candidats significativement plus grand que la taille de lot désirée. Dans le cadre de cette étude, nous avons trouvé qu'un ensemble de taille quatre fois supérieure à la taille du lot était suffisant.

Étant donné qu'il est nécessaire de calculer les résidus pour ce nouvel ensemble important de candidats, nous précisons ici que l'effort de calcul associé à cette méthode n'est pas négligeable. Le choix d'une fréquence de rééchantillonnage appropriée (κ) peut aider à réduire cette charge de calcul.

Néanmoins, la méthode RAD permet au PINN de concentrer son effort de calcul sur les régions à forte erreur d'approximation, ce qui aide à réduire le sous-apprentissage dans les régions complexes et peut conduire à une meilleure convergence et généralisation.

2.3.2.7 Masque d'attention résiduel (RBA)

Le masque d'attention résiduelle (Residual-Based Attention, RBA) Anagnostopoulos, Toscano, Stergiopulos & Karniadakis (2024) est une stratégie de pondération dynamique conçue pour améliorer l'efficacité de l'entraînement des PINNs. Cette deuxième méthode affectant les points de collocation alloue également de manière adaptative les ressources computationnelles aux régions où les contraintes physiques sont les moins bien satisfaites. Cependant, elle fonctionne très différemment que RAD.

Le mécanisme RBA attribue des poids d'attention λ_i^k évolutifs à chaque point de collocation i à chaque itération d'entraînement k. Tous les poids sont d'abord initialisés uniformément à $\lambda_i^0 = 1$, garantissant une attention initiale équilibrée sur tout le domaine. À chaque passage avant, le résidu physique e_i est calculé pour chaque point, quantifiant localement le résidu des équations gouvernantes. Les poids sont alors mis à jour selon :

$$\lambda_i^{k+1} = \gamma \lambda_i^k + \eta \frac{|e_i|}{\|e\|_\infty} \tag{2.43}$$

où $\gamma \in (0, 1)$ contrôle la rétention mémoire, η détermine l'intensité de mise à jour, et $\|e\|_\infty$ normalise par le résidu maximal. Pour nos expériences, nous avons choisi $\gamma = 0.999$ et $\eta = 0.01$.

Les poids RBA présentent plusieurs avantages. Ils détiennent notamment des bornes garanties :

$$\lambda_i^k \in \left(0, \frac{\eta}{1-\gamma}\right] \quad \forall i, k \tag{2.44}$$

Cette contrainte assure la stabilité numérique tout en permettant une adaptation dynamique. De plus, cette méthode utilise les résiduels déjà calculés dans la passe avant de la descente de

gradient. L'effort numérique requis pour cette méthode est donc négligeable. Le coût en mémoire est d'autant plus tolérable alors que les requis de mémoire évoluent linéairement avec le nombre de points de collocations.

Une attention particulière doit être portée à l'échantillonnage des points de collocation dans l'espace-temps. Un échantillonnage purement aléatoire s'avère inadéquat, car il ne permet pas d'assurer une correspondance cohérente entre les poids résiduels et les régions physiques. Deux approches principales peuvent être envisagées.

La première consiste à utiliser un ensemble fixe de points de collocation. Bien que fonctionnelle, cette méthode nécessite une densité de points élevée pour capturer fidèlement toutes les caractéristiques de la solution, ce qui se traduit par une augmentation significative des besoins en mémoire et en temps de calcul.

Une alternative plus efficace repose sur une partition structurée de l'espace-temps en sous-domaines distincts. Chaque sous-domaine fournit un point de collocation échantillonné aléatoirement, préservant ainsi la couverture uniforme de l'espace tout en maintenant une structure exploitable par le mécanisme d'attention. Cette approche permet d'associer systématiquement à chaque nouveau point un poids résiduel approprié selon sa position dans l'espace-temps. La Figure 2.5 montre un exemple en 1D de cet échantillonneur structuré-aléatoire à deux itérations de descente de gradient différentes.

La méthode de masque d'attention résiduelle permet donc aux PINNs de concentrer automatiquement leurs efforts sur les régions physiquement critiques, améliorant la convergence avec un effort minime.

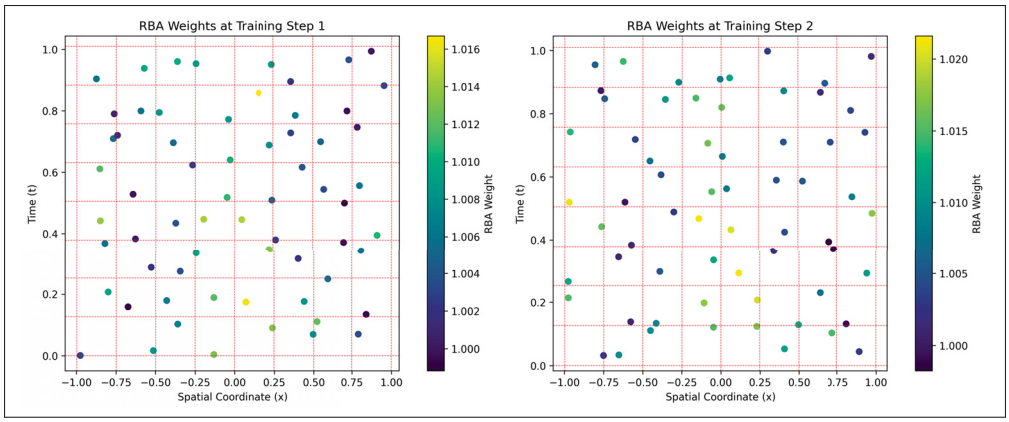


Figure 2.5 Échantillonneur structuré-aléatoire de points de collocations pour le masque d'attention résiduel. Points échantillonnés à deux itérations distinctes

2.3.3 PINNs pour les méthodes level set

Dans le solveur PINN que nous proposons pour les problèmes basés sur des level set, la fonction de perte contient les pertes initiales et résiduelles :

$$\mathcal{L}_{level\ set} = \lambda_{ic}\mathcal{L}_{ic} + \lambda_r\mathcal{L}_r. \tag{2.45}$$

Le résiduel de level set est défini par :

$$\mathcal{L}_r(\boldsymbol{x}, t) = \frac{\partial \phi}{\partial t}(\boldsymbol{x}, t) + \boldsymbol{v}(\boldsymbol{x}, t) \cdot \nabla\phi(\boldsymbol{x}, t), \tag{2.46}$$

où $\boldsymbol{v}(\boldsymbol{x}, t)$ est le champ de vitesse imposé. Nous notons que les conditions aux frontières ne sont pas appliquées dans les problèmes de référence de level set.

Dans ce qui suit, nous proposons deux termes de pertes supplémentaires qui peuvent être incorporés dans la fonction de perte PINN de level set. Il s'agit d'un *perte d'Eikonal*, conçu pour appliquer la propriété d'Eikonal (2.5), et d'un *perte de masse*, visant à améliorer la conservation de la masse du modèle PINN.

2.3.3.1 Perte d'Eikonal

Pour une fonction level set donnée ϕ, le résidu d'Eikonal est défini par :

$$\epsilon_{eik}(\boldsymbol{x}, t) = \|\nabla\phi(\boldsymbol{x}, t)\|^2 - 1, \tag{2.47}$$

où le résidu représente l'écart par rapport à la propriété d'Eikonal. La perte d'Eikonal est ensuite calculée aux points résiduels sélectionnés $(\boldsymbol{x}^i_{eik}, t^i_{eik})$ comme suit :

$$\mathcal{L}_{eik}(\theta) = \frac{1}{N_{eik}} \sum_{i=1}^{N_{eik}} |\epsilon_{eik}(\boldsymbol{x}^i_{eik}, t^i_{eik})|^2, \tag{2.48}$$

où N_{eik} est le nombre de points résiduels utilisés pour évaluer la perte d'Eikonal. La fonction de perte mise à jour peut être réécrite comme suit :

$$\mathcal{L}(\theta) = \lambda_{ic}\mathcal{L}_{ic}(\theta) + \lambda_r\mathcal{L}_r(\theta) + \lambda_{eik}\mathcal{L}_{eik}(\theta). \tag{2.49}$$

En incorporant ce terme directement dans le processus d'entraînement du PINN, notre approche élimine la nécessité d'une étape de réinitialisation séparée pour appliquer la propriété de distance signée. Cette intégration simplifie non seulement le flux de travail, mais offre également un avantage significatif par rapport aux techniques numériques traditionnelles, pour lesquelles une telle initialisation est généralement nécessaire.

2.3.3.2 Perte de masse

Le second terme de perte qui peut être ajouté est la perte de masse, qui implique le calcul de la masse de la phase i à chaque pas de temps et contraint le modèle PINN à la maintenir égale à la masse initiale. Dans notre étude des problèmes à deux phases, nous mettons en œuvre la perte de masse en calculant la surface de Ω_1 et en la contraignant à rester constante au fil du temps. L'aire est calculée à l'aide de l'approche d'échantillonnage de Monte Carlo (MC). Dans ce contexte, la surface $\mathcal{A}_\theta$ est estimée en échantillonnant N points aléatoires $\boldsymbol{x}_i$ uniformément

dans l'ensemble du domaine Ω et en vérifiant s'ils appartiennent à la région Ω_1, définie comme le sous-ensemble de Ω où la fonction level set $\phi(\boldsymbol{x}, t) < 0$:

$$\Omega_1(t) = \{\boldsymbol{x} \in \Omega \mid \phi(\boldsymbol{x}, t) < 0\}. \tag{2.50}$$

L'aire $\mathcal{A}_\theta$ est calculée ainsi :

$$\mathcal{A}_\theta(t) \approx \frac{\text{Number of points in } \Omega_1(t)}{N} \cdot \mathcal{V}(\Omega), \tag{2.51}$$

où $\mathcal{V}(\Omega)$ est le volume du domaine d'échantillonnage Ω.

Étant donné que la surface exacte à $t = 0$, notée $\mathcal{A}_{exact}$, doit être conservée au fil du temps, la perte de masse est calculée comme suit :

$$\mathcal{L}_{mass}(t_i, \theta) = (\mathcal{A}_\theta(t_i) - \mathcal{A}_{exact})^2, \tag{2.52}$$

où $\mathcal{A}_\theta(t_i)$ est la surface prédite à des points temporels spécifiques $t_i \; dans [0, T]$. Pour calculer la perte de masse globale, on utilise la moyenne de $\mathcal{L}_{mass}(t_i, \theta)$ sur l'ensemble des points temporels échantillonnés :

$$\mathcal{L}_{mass}(\theta) = \frac{1}{N_t} \sum_{i=1}^{N_t} \mathcal{L}_{mass}(t_i, \theta), \tag{2.53}$$

où N_t est le nombre total de points temporels échantillonnés. Cette approche garantit que le modèle conserve la zone de manière cohérente pour tous les intervalles de temps sélectionnés.

Deux méthodes ont été testées pour mettre en œuvre la perte de masse dans la fonction de perte. *Méthode 1* : cette méthode est simple et consiste à ajouter directement la perte de masse à la fonction de perte composite :

$$\mathcal{L}(\theta) = \lambda_{ic}\mathcal{L}_{ic}(\theta) + \lambda_r\mathcal{L}_r(\theta) + \lambda_{mass}\mathcal{L}_{mass}(\theta). \tag{2.54}$$

Bien que simple, cette méthode peut s'avérer coûteuse en termes de calcul, car la perte de masse doit être calculée à chaque étape de l'apprentissage. Cela implique le calcul de la solution du

level set pour l'ensemble du domaine spatio-temporel, ce qui devient coûteux lorsque ce calcul est effectué plusieurs milliers de fois. Une approche plus efficace est donc souhaitable pour limiter la fréquence de calcul de la perte de masse.

Méthode 2 : répond à cette limitation en utilisant un cadre à deux réseaux, qui implique le calcul de la perte de masse dans un deuxième réseau distinct initialisé avec l'apprentissage par transfert.

Le transfer learning, ou apprentissage par transfert, est une approche de l'apprentissage automatique qui consiste à réutiliser un modèle préalablement entraîné sur une tâche pour en résoudre une nouvelle, souvent différente, mais reliée. Plutôt que de former un modèle à partir de zéro, le transfer learning exploite les connaissances acquises dans une source d'apprentissage pour améliorer les performances dans un domaine cible. Cette approche repose sur le transfert des poids et des biais du modèle source vers le modèle cible, permettant d'adapter les paramètres déjà optimisés pour accélérer et stabiliser l'apprentissage dans le nouveau contexte.

Le premier réseau de la *Méthode 2*, le réseau source, est donc formé sans le terme de perte de masse. Les paramètres formés du premier réseau sont ensuite utilisés pour initialiser un second réseau, le réseau cible. Ce second réseau est entraîné exclusivement sur le terme de perte de masse. Le modèle est ensuite évalué avec les paramètres sauvegardés du second réseau. Les deux fonctions de perte sont définies comme suit :

$$\begin{aligned} \mathcal{L}_{net\,1} &= \lambda_{ic}\mathcal{L}_{ic} + \lambda_r\mathcal{L}_r, \\ \mathcal{L}_{net\,2} &= \mathcal{L}_{mass}. \end{aligned} \tag{2.55}$$

En découplant l'entraînement, *Méthode 2* réduit la charge de calcul et permet un nombre adapté d'étapes d'entraînement pour la perte de masse. Cette méthode présente également l'avantage de nous permettre de choisir un nombre personnalisé de points de collocation.

2.3.4 PINNs pour les équations de Saint-Venant

Dans le solveur PINN que nous proposons pour les problèmes basés sur les SWE, la fonction de perte contient les pertes initiales et résiduelles :

$$\mathcal{L}_{swe} = \lambda_{ic}\mathcal{L}_{ic} + \lambda_{bc}\mathcal{L}_{bc} + \lambda_r\mathcal{L}_r. \tag{2.56}$$

Le résiduel des SWE est défini par les résiduels des équations de continuité et de conservation de la quantité de mouvement en x et y :

$$\mathcal{L}_{r\,cont}(\boldsymbol{x}, t) = \left(\frac{\partial h^*}{\partial t^*} + \bar{u}^*\frac{\partial h^*}{\partial x^*} + \bar{v}^*\frac{\partial h^*}{\partial y^*} + h^*\left(\frac{\partial \bar{u}^*}{\partial x^*} + \frac{\partial \bar{v}^*}{\partial y^*}\right)\right)^2, \tag{2.57}$$

$$\mathcal{L}_{r\,x-mom}(\boldsymbol{x}, t) = \left(\frac{\partial \bar{u}^*}{\partial t^*} + \bar{u}^*\frac{\partial \bar{u}^*}{\partial x^*} + \bar{v}^*\frac{\partial \bar{u}^*}{\partial y^*} + \frac{1}{\mathrm{Fr}^2}\frac{\partial h^*}{\partial x^*} + \frac{1}{\mathrm{Fr}^2}\frac{\partial b^*}{\partial x^*}\right)^2, \tag{2.58}$$

$$\mathcal{L}_{r\,y-mom}(\boldsymbol{x}, t) = \left(\frac{\partial \bar{v}^*}{\partial t^*} + \bar{u}^*\frac{\partial \bar{v}^*}{\partial x^*} + \bar{v}^*\frac{\partial \bar{v}^*}{\partial y^*} + \frac{1}{\mathrm{Fr}^2}\frac{\partial h^*}{\partial y^*} + \frac{1}{\mathrm{Fr}^2}\frac{\partial b^*}{\partial y^*}\right)^2, \tag{2.59}$$

avec

$$\lambda_r\mathcal{L}_r = \lambda_{r\,cont}\mathcal{L}_{r\,cont} + \lambda_{r\,x-mom}\mathcal{L}_{r\,x-mom} + \lambda_{r\,y-mom}\mathcal{L}_{r\,y-mom}. \tag{2.60}$$

À noter que nous avons négligé les termes de frictions pour les problèmes testés dans ce mémoire.

2.3.4.1 Diffusion numérique

Les équations de Saint-Venant contiennent des termes non linéaires responsables de la formation de discontinuités, telles que les chocs. Ces fortes variations spatiales peuvent poser des défis importants lors de l'entraînement des PINNs, entraînant une instabilité numérique ou une mauvaise convergence de la solution. Une approche bien établie en analyse des EDP hyperboliques consiste à utiliser la méthode de viscosité artificielle (vanishing viscosity method) Crandall & Lions (1983); Lax (2006) pour obtenir des solutions physiques admissibles. Cette méthode repose sur le fait que les solutions des équations inviscides, y compris celles avec des

chocs, sont les limites des solutions des équations visqueuses lorsque le coefficient de viscosité tend vers zéro. Inspiré de cette idée, Fuks et Tchelepi Fuks & Tchelepi (2020) ont proposé d'ajouter un terme de diffusion du second ordre à un problème de transport diphasique non linéaire en milieu poreux. Cet ajout rend l'EDP parabolique, c'est-à-dire que sa solution est lisse, sans choc et plus facile à approximer pour un PINN. Cela dit, Fuks et Tchelepi ont réussi à capturer avec précision la position et l'amplitude d'un choc dans leur problème de transport avec cette simple modification à leur PINN.

Similairement, un terme de diffusion numérique du second ordre, $\nu(u_{xx} + u_{yy})$, peut être ajouté aux équations de Saint-Venant afin de rendre l'EDP parabolique. Ainsi, l'équation 2.19 de quantité de mouvement en x devient alors :

$$\frac{\partial \bar{u}^*}{\partial t^*} + \bar{u}^* \frac{\partial \bar{u}^*}{\partial x^*} + \bar{v}^* \frac{\partial \bar{u}^*}{\partial y^*} + \frac{1}{\mathrm{Fr}^2} \frac{\partial h^*}{\partial x^*} = -\frac{1}{\mathrm{Fr}^2} \frac{\partial b^*}{\partial x^*} + \nu \left(\frac{\partial^2 \bar{u}^*}{\partial x^{*2}} + \frac{\partial^2 \bar{u}^*}{\partial y^{*2}} \right), \tag{2.61}$$

où le coefficient ν représente la viscosité numérique. Son choix est critique : une valeur trop faible rend le terme inefficace, tandis qu'une valeur trop élevée lisse excessivement la solution et peut compromettre la précision du modèle. Ce coefficient dépend du problème, il n'y a donc pas de loi qui régit la valeur optimale de ν. Une série de tests empiriques doivent donc être effectués afin de déterminer une valeur optimale pour ν. C'est cette stratégie qui a été utilisée dans cette étude. Un terme de diffusion analogue est également ajouté dans l'équation de la quantité de mouvement en y.

2.4 Implémentation algorithmique

Le code a été construit sur JAX Bradbury *et al.* (2018) avec la bibliothèque JAXPI Wang *et al.* (2023), qui permet une mise en œuvre facile des PINNs, *PirateNets*, et les améliorations mentionnées dans la section 2.3.2. Les entraînements ont été suivis avec Weights & Biases (WandB) Biewald & Van Pelt (2020) et les visualisations ont été effectuées avec Matplotlib Hunter (2007) et WandB. Les résultats ont été calculés à l'aide d'un seul GPU NVIDIA V100l

sur l'une des grappes de calcul de l'Alliance de la recherche numérique au Canada (DRAC). Les calculs ont été effectués en simple précision. Le code est disponible sur https ://github.com/m-mullins/LS-PINN/ et https ://github.com/m-mullins/SWE-PINN/.

CHAPITRE 3

RÉSULTATS

Dans les sections suivantes, nous présentons les résultats numériques obtenus par le solveur PINN proposé pour des problèmes de références bien connus dans le domaine des level set et des SWE. Avant de plonger dans ces problèmes, nous validons d'abord notre cadre *PirateNet* à l'aide de deux problèmes de référence difficiles en dynamique des fluides, les équations de Burgers en 1D et 2D. Ensuite, nous nous concentrons sur deux problèmes de références en level : le disque de Zalesak et l'écoulement tourbillonnaire inversé dans le temps. Finalement, nous présentons les résultats des problèmes de références en SWE : la propagation d'une vague sur une digue et la rupture de barrage radiale.

Pour évaluer la précision du solveur PINN, nous comparons ses performances à une solution de référence en utilisant la métrique d'erreur relative L^2 :

$$\text{Relative } L^2 \text{ Error} = \frac{\|u_{\text{PINN}} - u_{\text{reference}}\|_2}{\|u_{\text{reference}}\|_2}, \tag{3.1}$$

où u_{PINN} est la solution obtenue par le solveur PINN, et $u_{\text{référence}}$ est la solution de référence haute-fidélité.

Dans ce chapitre, nous définissons les configurations suivantes :

- *Plain* : la formulation PINN originale de Raissi *et al.* (2019),
- *Default* : l'approche PINN améliorée de Wang *et al.* (2023),
- *PirateNet* : l'architecture PirateNet de Wang *et al.* (2025) incluant les améliorations de *Default*,
- *Plain**, *Default**, *PirateNet** : Configuration avec les hyperparamètres optimaux, déterminés par un balayage détaillé des hyperparamètres à l'aide de l'approche de balayage bayésien de WandB Biewald & Van Pelt (2020).

3.1 Tests de références

Cette section présente les résultats des problèmes de références en mécanique des fluides, les équations de Burgers' en 1D et 2D.

3.1.1 Burgers 1D

L'équation de Burgers 1D est une EDP fondamentale qui présente des similitudes avec les problèmes d'écoulement des fluides. Grâce à ses termes d'advection et de diffusion, l'EDP peut modéliser des comportements de dynamique des fluides tels que la formation de chocs et la propagation d'ondes non linéaires. Ce benchmark nous donne un aperçu de la capacité du modèle PINN à capturer la dynamique non linéaire et à gérer les problèmes de stabilité associés aux gradients élevés et aux discontinuités. L'équation de Burgers 1D, ainsi que la condition initiale et la condition frontière de Dirichlet, s'écrivent comme suit :

$$\begin{aligned} &\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \quad t \in [0,T],\, x \in [-L,L], \\ &u(x=-L,t) = u(x=L,t) = 0, \\ &u(x,t=0) = 0 \end{aligned} \tag{3.2}$$

où $u = u(x,t)$ représente le champ de vitesse en fonction de l'espace et du temps et ν est la viscosité cinématique. Dans la partie gauche de l'équation, le premier terme représente la dérivée temporelle. Le second terme est le terme d'advection, qui décrit comment le champ de vitesse se transporte à travers le domaine et qui est responsable de la non-linéarité de l'équation. Du côté droit, le terme diffusif, contrôlé par la viscosité ν lisse les gradients aigus. Dans notre cas, nous avons fixé $L = 1$, $T = 1$, $\nu = 0.002$, et la vitesse caractéristique $U = 1$, ce qui nous donne un nombre de Reynolds élevé $Re = 1000$, rendant le problème difficile. Nous comparons notre modèle PINN avec une solution de référence générée à l'aide de la librairie Chebfun dans MATLAB Driscoll, Hale & Trefethen (2014). La solution de référence est basée sur une résolution spatiale de 201 points dans le temps et 512 points dans l'espace, respectivement.

Pour évaluer l'efficacité des caractéristiques décrites dans la section méthodologique 2.3.2, nous avons mené une étude d'ablation. Cette étude a systématiquement désactivé des paramètres spécifiques pendant la phase de formation afin d'analyser leur impact sur la précision des prédictions du modèle PINN. Dans un souci de cohérence, nous avons d'abord utilisé une architecture MLP pour toutes les configurations. Le modèle *PirateNet*, cependant, utilise l'ensemble complet des améliorations et emploie l'architecture *PirateNet* au lieu du MLP. Les paramètres hyperparamétriques des modèles *Plain*, *Default* et *PirateNet* sont détaillés dans le tableau I-1. À l'exception de l'architecture et des hyperparamètres d'ablation, tous les autres hyperparamètres ont été maintenus constants.

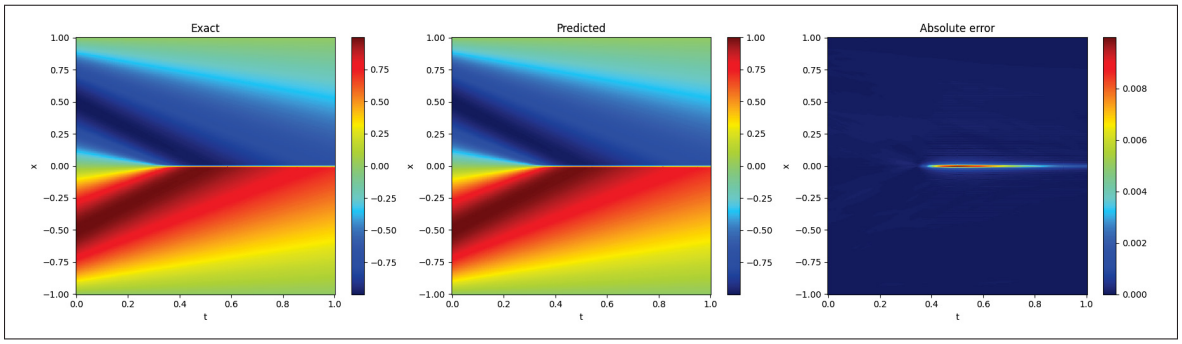
D'après le tableau 3.1, nous pouvons voir que *PirateNet* est plus performant que les autres modèles, avec une erreur relative de L^2 de 0.070%, alors que le pire est de loin le modèle *Plain* PINN, avec une erreur relative de L^2 de $0,496\%$. Nous remarquons que les résultats des meilleurs modèles sont très proches, ce qui montre que certaines améliorations ont moins d'influence que d'autres pour le problème 1D de Burgers. Cependant, il est suggéré de garder toutes ces caractéristiques activées, car elles améliorent généralement les performances lors de la résolution d'EDP avec des PINN tel que mentionnée dans Wang *et al.* (2023). Cela est confirmé par le fait que la configuration *Défaut* est très proche des autres configurations les plus performantes avec $L^2_{rel} = 0.073\%$. La solution graphique pour les PINN avec l'architecture *PirateNet* est présentée dans la figure 3.1. Nous pouvons voir à partir du graphique d'erreur absolue que la plupart des erreurs sont concentrées au point de formation du choc. Cela peut expliquer pourquoi la plupart des configurations ont des erreurs L^2_{rel} très similaires. La plupart du problème est facile à résoudre, ce qui rend la zone la plus difficile du problème très petite, ce qui entraîne un impact plus faible sur le L^2_{rel}. La différence de performances du modèle sera plus évidente pour d'autres problèmes.

3.1.2 Burgers 2D

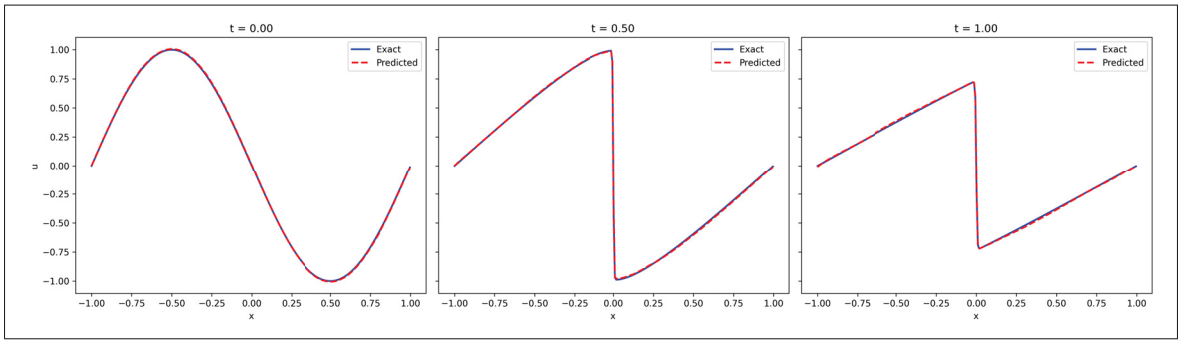
Le prochain problème de référence est le problème des Burgers 2D. Il offre la même perspective que son homologue 1D mais avec une dimension spatiale supplémentaire en entrée et un champ

Tableau 3.1 Résultats de l'étude d'ablation pour l'équation de Burgers 1D : comparaison de l'erreur relative L^2 entre les configurations

Configuration	Erreur L^2 relative (%)
PirateNet	**0.070**
No Causal Training	0.072
Default	0.073
No RWF	0.073
No Fourier Feature	0.093
No Grad Norm	0.151
Plain	0.496



a) Burgers 1D : Comparaison de la prédiction *PirateNet* avec la solution de référence pour la grille spatio-temporelle complète



b) Burgers 1D : Prédiction et solution de référence aux pas de temps $t = [0, 0.5, 1]$s

Figure 3.1 Burgers 1D : Comparaison de la prédiction *PirateNet* avec la solution de référence.

de vitesse supplémentaire en sortie. Les équations s'écrivent comme suit :

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \quad t \in [0, T], \quad x, y \in [0, L], \tag{3.3}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \quad t \in [0, T], \quad x, y \in [0, L], \tag{3.4}$$

La condition initiale et les conditions frontières de Dirichlet sont données par :

$$u(x, y, t = 0) = \sin(2\pi x)\sin(2\pi y), \tag{3.5}$$

$$v(x, y, t = 0) = \sin(\pi x)\sin(\pi y), \tag{3.6}$$

$$u(x, y, t) = 0 \quad \text{for} \quad (x, y) \in \partial\Omega, \tag{3.7}$$

$$v(x, y, t) = 0 \quad \text{for} \quad (x, y) \in \partial\Omega, \tag{3.8}$$

Pour ce problème, $T = 0.5$, $L = 1$, $\nu = \frac{0.015}{\pi}$, ce qui donne un nombre de Reynolds Re = 237. La résolution dans t, x et y est fixée à 101. L'ensemble de données utilisé comme solution de référence a été généré à l'aide d'une méthode des différences finies Mathias *et al.* (2022).

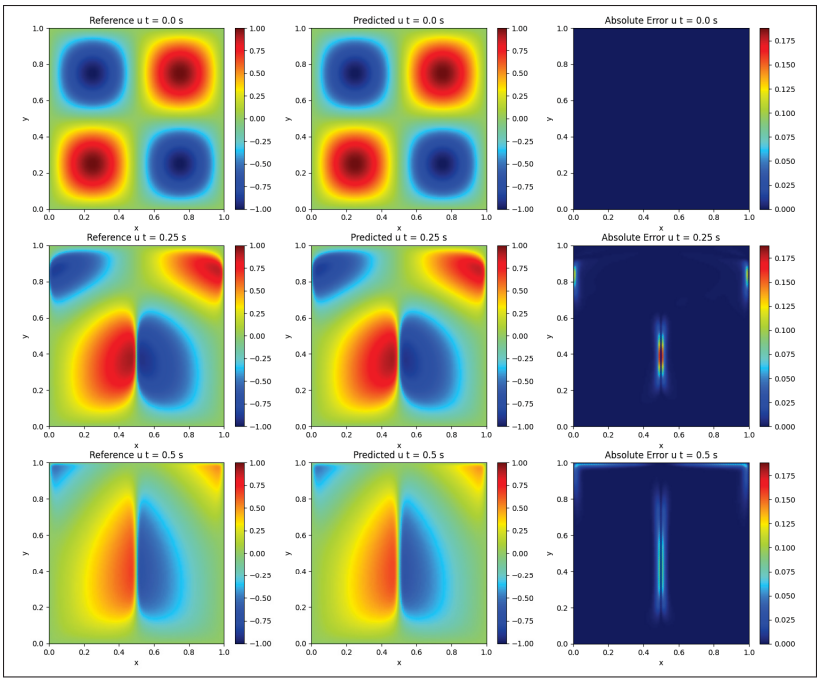
Comme dans le cas 1D de Burgers, une étude d'ablation a été effectuée pour comparer l'architecture *PirateNet* avec d'autres configurations sous les mêmes hyperparamètres de base qui peuvent être trouvés dans le tableau I-2 pour *Plain*, *Default* et *PirateNet*.

Le tableau 3.2 montre l'erreur L^2 pour les deux champs de vitesse de sortie u et v, ainsi qu'une norme des erreurs $L^2_{norm} = \sqrt{(L^2_u)^2 + (L^2_v)^2}$. Nous pouvons à nouveau voir que *PirateNet* fonctionne mieux que les autres configurations, avec une $L^2_{norm} = 2.52\%$. Une représentation graphique des champs de vitesse de référence et prédits à différents moments est présentée dans la figure 3.2 pour *PirateNet*. Similairement à la configuration 1D de Burgers, nous remarquons que les erreurs sont très proches l'une de l'autre, ce qui est attendu, car les configurations 1D et 2D de Burgers sont régies par le même type d'équations. Là encore, la majeure partie de l'erreur est également proche du front d'onde.

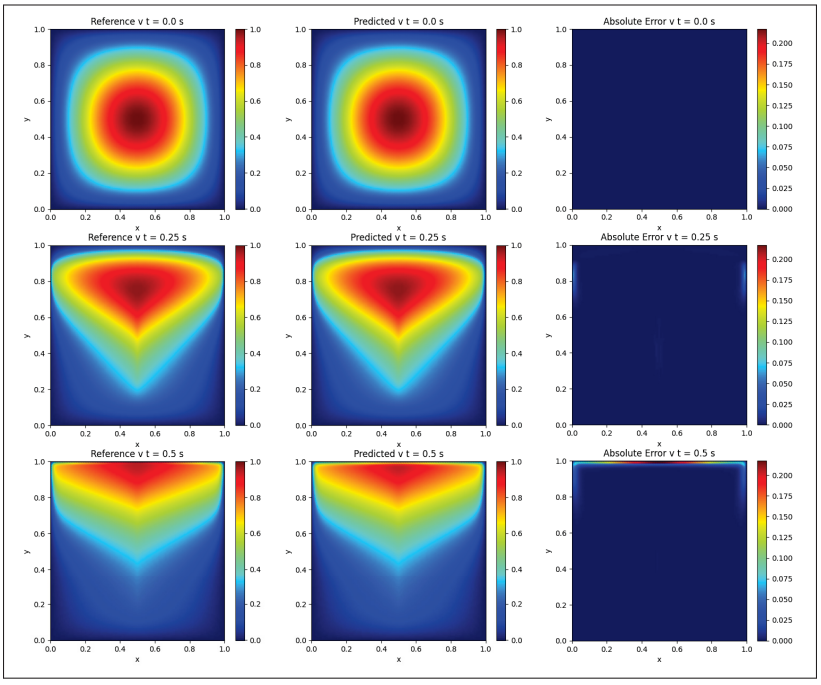
Tableau 3.2 Burgers 2D : Étude d'ablation

Configuration	Erreur L_u^2 (%)	Erreur L_v^2 (%)	Norme L^2 (%)	Temps d'entraînement (min)
Plain	2.20	1.46	2.64	**57**
Default	2.11	1.40	2.53	128
No Fourier feature	2.12	1.40	2.54	100
No RWF	**2.10**	1.40	2.53	108
No grad norm	2.11	1.40	2.53	127
No causal training	**2.10**	**1.39**	**2.52**	148
No modified MLP	2.11	1.40	**2.52**	61
PirateNet	**2.10**	**1.39**	**2.52**	168

Il convient également de noter l'impact de certaines fonctionnalités sur la charge de calcul. Bien que les temps d'entraînements puissent varier en fonction de l'état du matériel et des nœuds de calcul du cluster de calcul, on constate une augmentation significative du temps d'entraînement lorsque l'on compare la configuration *Plain* avec la plupart des autres configurations avec des fonctionnalités supplémentaires. Le *PirateNet* prend près de trois fois plus de temps pour s'entraîner, soit 168 minutes, tandis que le *Default* prend plus de deux fois plus de temps, soit 128 minutes, contre 57 minutes pour la configuration *Plain*. Cependant, il convient de noter que ce temps d'entraînement concerne les 70k pas d'entraînement. Le graphique 3.3 démontre que *PirateNet* et *Default* convergent beaucoup plus rapidement à environ 25k pas par rapport à *Plain* à environ 40k.



a) Champs de vitesse u à $t = [0, 0.25, 0.5]$s



b) Champs de vitesse v à $t = [0, 0.25, 0.5]$s

Figure 3.2 Burgers 2d : Référence (gauche), Prédiction (milieu) et erreur absolue (droite) des champs de vitesse u, v à différents pas de temps avec la configuration *PirateNet*.

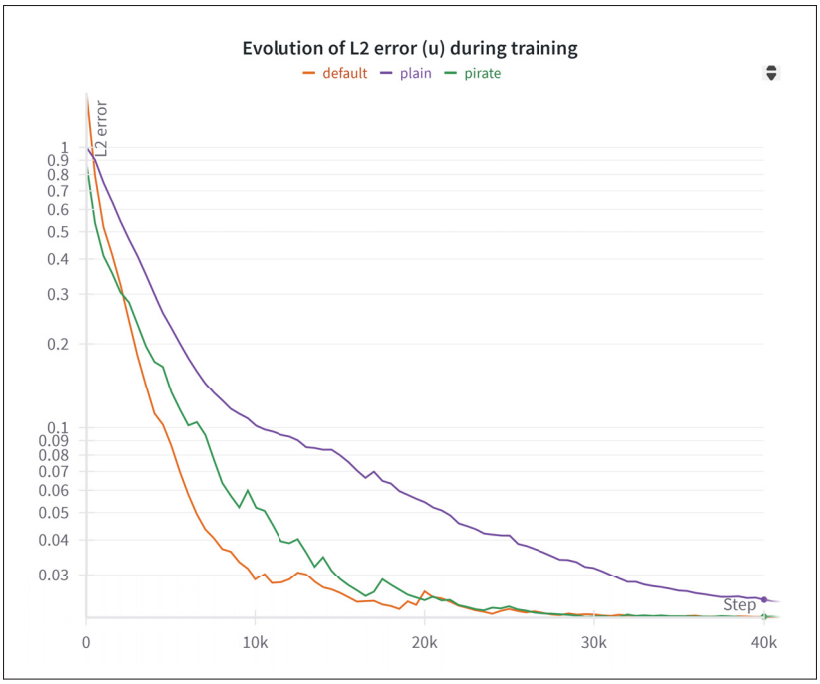


Figure 3.3 Burgers 2D : Évolution de l'erreur L^2 lors de l'apprentissage pour le champs de vitesse u

Un autre aspect que nous avons testé était l'impact que pouvait avoir un schéma de marche temporelle sur les résultats. Par conséquent, un schéma d'entraînement S2S a été implémenté et testé pour la configuration *PirateNet*. Comme le montre le tableau 3.3, nous pouvons voir que le schéma S2S avec l'architecture *PirateNet* est le plus performant avec une $L^2_{norm} = 2.49\%$, ce qui met en évidence la nature séquentielle du problème.

Tableau 3.3 Burgers 2D : Impact du programme d'entraînement séquence par séquence

Configuration	Erreur L^2_u (%)	Erreur L^2_v (%)	Norme L^2 (%)
PirateNet	2.10	1.39	2.52
PirateNet S2S	**2.08**	**1.37**	**2.49**

3.2 Tests des équations level set

Dans cette section, nous présentons une série de tests de référence level set pour valider rigoureusement l'efficacité et la fiabilité de l'approche PINN proposée. Nous commençons

avec la rotation du disque de Zalesak et finissons avec le problème plus difficile d'écoulement tourbillonnaire inversé dans le temps.

3.2.1 Disque de Zalesak

Tout d'abord, nous explorons un problème de référence populaire pour tester les level set, la rotation du corps rigide du disque de Zalesak Zalesak (1979); Rider & Kothe (1998). Ce test consiste à soumettre une interface non déformante au champ d'écoulement rotatif suivant :

$$u(x, y) = \frac{2\pi}{T}(0.5 - y) \quad \text{and} \quad v(x, y) = \frac{2\pi}{T}(x - 0.5), \tag{3.9}$$

avec $T = 2\pi$ s. Le domaine de calcul est un cercle de rayon $R = 0, 5$, centré en (0,5, 0,5) dans un carré unitaire. L'interface est constituée d'un disque fendu centré en $(0, 5, 0, 75)$ avec un rayon $r = 0, 15$. La fente est alignée le long de la ligne centrale du disque et s'étend verticalement avec une hauteur $h = 0, 25$ et une largeur $w = 0, 05$.

Le domaine circulaire est choisi pour éviter les problèmes numériques liés à la condition limite d'entrée lors de la résolution de la solution de référence avec une méthode des éléments finis (FEM). La solution de référence a été obtenue en utilisant la méthode des éléments finis stabilisés par SUPG, avec une approximation quadratique pour la fonction level set sur les éléments triangulaires non structurés (T6) et un schéma d'intégration temporelle explicite de Runge–Kutta (SSP–RK) préservant la stabilité forte du troisième . Le solveur a été implémenté avec MATLAB par Fahsi & Soulaïmani (2017). De plus, le maillage utilisé pour le solveur FEM avait 10470 éléments, ce qui nous a permis d'interpoler les résultats sur une grille compatible avec Python et uniformément espacée 101×101. La simulation FEM a duré $t_{end} = T$, et le pas de temps était $\Delta t_{FEM} = 1, 57 \times 10^{-3}$ s. La résolution temporelle a été sous-échantillonnée à $\Delta t_{PINN} = 1, 57 \times 10^{-1}$ s pour la comparaison des modèles FEM et PINN.

Nous avons entraîné un ensemble de configurations PINN (*Plain*, *Default*, *PirateNet*) pour apprendre l'évolution de l'interface. Les configurations suivent la même norme que dans les sections précédentes. Leurs hyperparamètres détaillés peuvent être trouvés dans le tableau I-3.

À partir du tableau 3.4, nous pouvons à nouveau voir que *PirateNet* performe mieux que *Default* et *Plain* avec une erreur relative $L^2_{PirateNet} = 0.35\%$ par rapport à $L^2_{Default} = 2.96\%$ et $L^2_{Plain} = 4.18\%$. Avec l'architecture PirateNet, nous avons ensuite effectué un balayage d'hyperparamètres bayésien à l'aide du module de balayage Biewald & Van Pelt (2020) de Wandb. Les configurations suivies d'un * du tableau I-3 résument les hyperparamètres optimaux, ce qui nous permet de réduire les erreurs à $L^2_{PirateNet^*} = 0.12\%$ et $L^2_{Default^*} = 0.15\%$. La figure 3.4 pour la configuration *PirateNet** superpose les résultats graphiques à chaque quart de simulation, où nous pouvons confirmer que l'interface est bien capturée. En utilisant les poids

Tableau 3.4 Level set disque de Zalesak : Erreur L^2 erreur du champ de sortie ϕ

Configuration	Erreur L^2 (%)
Plain	4.18
Default	2.96
PirateNet	0.35
Plain*	3.48
Default*	0.15
PirateNet*	**0.12**

entraînés de *PirateNet**, nous avons quantifié la perte de masse tout au long de la simulation. La masse de référence est calculée de manière triviale en utilisant la surface du disque moins la surface de la fente à $t = 0$. Celle-ci devrait rester constante tout au long de la simulation. Nous avons comparé l'évolution de la perte de masse de *PirateNet** et du solveur *FEM* dans la figure 3.5. Nous remarquons que le solveur PINN a une variance plus élevée, tandis que *FEM* suit un modèle d'accumulation d'erreur plus constant. Cela est attendu puisque les PINN résolvent toute la grille spatio-temporelle en une seule fois, tandis que *FEM* utilise un schéma de marche temporelle. Néanmoins, *PirateNet** démontre ici qu'il peut capturer efficacement les modèles de masse avec des résultats similaires à un solveur level set FEM de haute précision. Leur erreur de masse absolue moyenne (MAPE) respective est de $MAPE_{PirateNet^*} = 0.31\%$ et $MAPE_{FEM} = 0.21\%$.

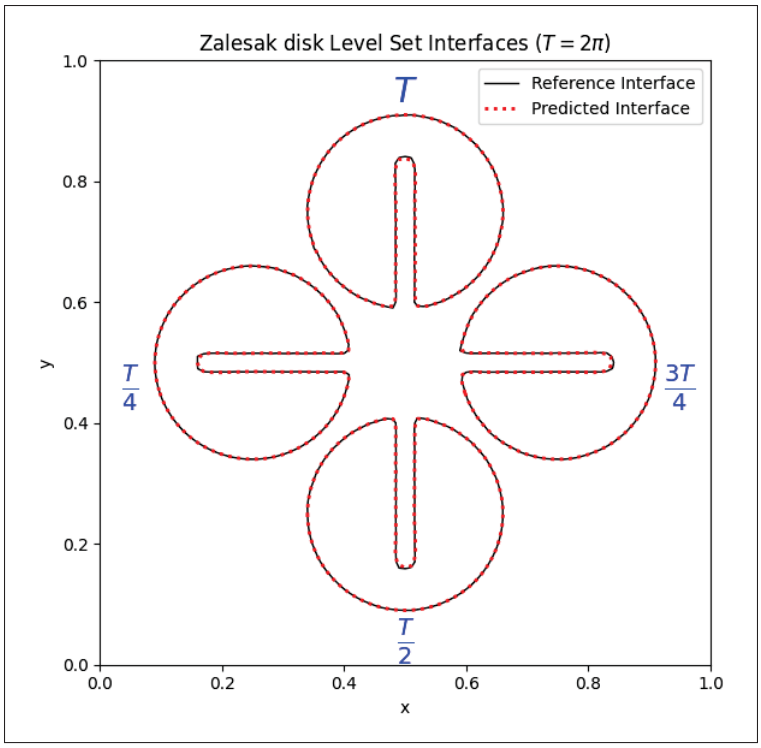


Figure 3.4 Level set disque de Zalesak : Évolution de la solution *reference* et *PirateNet** ($T = 2\pi$ s)

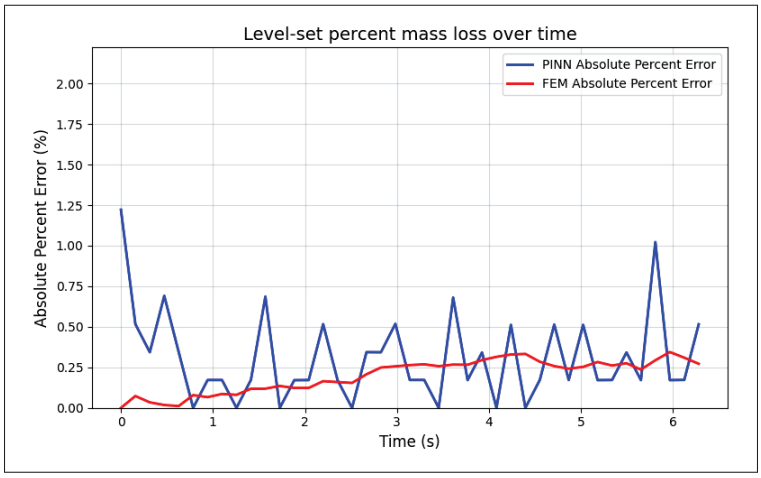
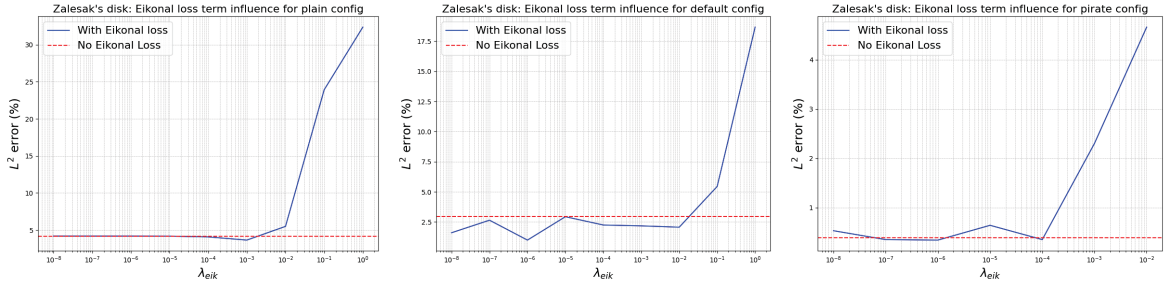


Figure 3.5 Level set disque de Zalesak : Évolution de l'erreur de masse absolue moyenne ($T = 2\pi$ s)

Après avoir évalué les performances d'un *PirateNet* de base sur le disque de Zalesak, nous avons décidé de tester comment les améliorations proposées dans la section 2.3.3 affecteraient les résultats.

Pour le terme de perte Eikonal, plusieurs méthodes de pondération ont été testées. Le poids λ_{eik} a d'abord été inclus dans le schéma de normalisation du gradient en utilisant le même poids initial que les autres termes : $\lambda_{eik} = \lambda_{ic} = \lambda_r = 1$. Même avec une mise à jour fréquente du poids toutes les 500 itérations d'entraînement, cela a aggravé la solution de près de deux ordres de grandeur par rapport au *PirateNet* de référence. Après avoir essayé un poids initial inférieur de $\lambda_{eik} = 0.01$, les résultats sont meilleurs, mais toujours d'un ordre de grandeur inférieur à celui de référence. Nous avons également remarqué que la normalisation du gradient avait tendance à augmenter le poids de perte Eikonal tout au long de l'entraînement à des valeurs supérieures aux autres poids, aggravant encore la solution.

Compte tenu de cela, nous considérons la perte Eikonale comme un terme de régularisation qui serait maintenu constant tout au long de l'entraînement et qui serait exclu des mises à jour de poids basées sur la normalisation du gradient. Nous avons testé une gamme de poids, λ_{eik}, et mesuré leur influence sur l'erreur. D'après la figure 3.6, nous pouvons voir que l'ajout de la perte Eikonale n'apporte aucune amélioration significative à la solution du problème de rotation du corps rigide du disque de Zalesak. Nous remarquons également que des poids plus élevés au-dessus de 10^{-3} ont un impact négatif sur la solution. Cependant, comme on le voit avec la configuration *Default*, nous constatons que dans certaines circonstances, la perte eikonale peut légèrement améliorer les résultats avec un poids λ_{eik} soigneusement choisi.



a) *Plain* b) *Default* c) *PirateNet*

Figure 3.6 Level set disque de Zalesak : Influence du terme de perte Eikonal λ_{eik}.

Ensuite, nous avons testé l'influence de l'ajout d'une perte de masse en utilisant *Méthode 2* de la section 2.3.3.2. *Méthode 1* a été ignorée dans ce cas en raison de sa surcharge de calcul.

L'approche de *Méthode 2* consiste à entraîner un réseau exclusivement avec une perte de masse, en utilisant les poids entièrement entraînés d'un réseau initialement entraîné avec la condition initiale standard et les pertes résiduelles. En utilisant les poids de *PirateNet**, nous avons entraîné le réseau pour 1000 itérations supplémentaires en utilisant une taille de lot de 64×64 points spatiaux sur tous les pas de temps pour calculer la prédiction du champ de sortie. Les autres hyperparamètres étaient les mêmes que ceux de *PirateNet**. L'ajout de cette perte de masse n'a pas amélioré l'erreur pour le problème du disque de Zalesak, car l'erreur est passée de $L^2_{PirateNet^*} = 0.12\%$ à $L^2_{PirateNet^*+masse} = 0.29\%$. De plus, l'erreur de masse absolue moyenne en pourcentage est passée de $MAPE_{PirateNet^*} = 0.31\%$ à $MAPE_{PirateNet^*+masse} = 1.09\%$. En ce qui concerne les méthodes de régularisation, l'ajout des termes Eikonal et de perte de masse avec des poids bien choisis peut légèrement affecter la précision, mais apporter une amélioration de la stabilisation.

3.2.2 Écoulement tourbillonnaire inversé dans le temps

Cette section montrera les capacités des PINNs à résoudre un problème level set de référence tel que défini dans Rider & Kothe (1998), l'écoulement tourbillonnaire inversé dans le temps. Ce test consiste à étirer et à faire tourner un cercle au cours de la première demi-période. Au cours de la seconde moitié, l'écoulement est inversé et le cercle revient à son état initial. Cela testera la capacité des PINNs à apprendre la fonction level set ϕ sous un champ de vitesse variable avec une forte déformation. La définition de la fonction de flux d'un seul écoulement tourbillonnaire est

$$\Psi_{single\ vortex}(x, y) = \frac{1}{\pi} \sin^2(\pi x) \sin^2(\pi y) \tag{3.10}$$

À partir de l'équation 3.10, nous pouvons obtenir la fonction de flux tourbillonnaire inversé dans le temps en multipliant par $\cos\left(\frac{\pi t}{T}\right)$, où T est la période.

$$\Psi_{time-reversed\ vortex}(x, y, t) = \frac{1}{\pi} \sin^2(\pi x) \sin^2(\pi y) \cos\left(\frac{\pi t}{T}\right) \tag{3.11}$$

À partir de l'équation 3.11, nous pouvons dériver les champs d'écoulement avec :

$$u = -\frac{\partial \Psi}{\partial y} \quad \text{and} \quad v = \frac{\partial \Psi}{\partial x} \tag{3.12}$$

où

$$u_{time-reversed\,vortex}(x, y, t) = -2\sin^2(\pi x)\cos(\pi y)\sin(\pi y)\cos\left(\frac{\pi t}{T}\right), \tag{3.13}$$

et

$$v_{time-reversed\,vortex}(x, y, t) = 2\cos(\pi x)\sin(\pi x)\sin^2(\pi y)\cos\left(\frac{\pi t}{T}\right). \tag{3.14}$$

Nous avons utilisé la même période $T = 8$ s que dans Touré & Soulaïmani (2016) pour nos tests. Le domaine de calcul a été défini comme $x, y \in [0, 1]$, et le domaine de calcul a été divisé en deux sous-domaines avec un cercle ou un rayon de $r = 0,15$, centré à $(0,5, 0,75)$. Pour ce problème, aucune condition frontière n'est appliquée. Cela implique que le terme de perte de condition frontière $\mathcal{L}_{bc}$ est supprimé de l'équation 2.24.

L'objectif est d'apprendre le champ de sortie de la fonction level set ϕ. La solution de référence a été générée à l'aide du même solveur FEM que dans la section 3.2.1.

Nous avons d'abord testé nos trois principales configurations de modèles, le *Plain*, le *Default* et le *PirateNet* sur le problème de l'écoulement tourbillonnaire inversé dans le temps. Ces configurations représentent l'utilisation des mêmes fonctionnalités que dans les sections 3.1.1 à 3.2.1. Elles ont été comparées en utilisant les mêmes hyperparamètres de base ainsi qu'avec des hyperparamètres optimisés. Le tableau I-4 recense les hyperparamètres où un * représente la version avec hyperparamètres optimisés. Une particularité de l'écoulement tourbillonnaire inversé dans le temps est que l'écoulement s'inverse à la demi-période. Par conséquent, nous devons utiliser un schéma S2S avec deux fenêtres temporelles pour garantir que les gradients sont réinitialisés avec les signes appropriés lorsque l'écoulement s'inverse.

À partir du tableau 3.5, nous pouvons voir que le *PirateNet* est celui qui a obtenu les meilleurs résultats avec $L^2_{PirateNet} = 5.24\%$. En revanche, avec des hyperparamètres optimisés, *Default* performent aussi bien que *PirateNet* avec des erreurs respectives de $L^2_{Default^*} = 0.81\%$ et

$L^2_{PirateNet^*} = 0.85\%$. De son côté, *Plain* manque de capacité d'expression avec une erreur $L^2_{Plain} = 51.20\%$. Les hyperparamètres sont disponibles au Tableau I-4 alors qu'une visualisation de l'optimisation des hyperparamètres avec WandB est disponible à l'annexe 2.2. Les résultats

Tableau 3.5 Level set tourbillon : Erreur L^2 du champ de sortie ϕ

Configuration	Erreur L^2 (%)
Plain	51.2
Default	20.86
PirateNet	5.24
Plain*	45.9
Default*	**0.81**
PirateNet*	0.85

graphiques pour *PirateNet** à cinq pas de temps différents sont présentés à la figure 3.7. Nous avons ensuite quantifié la perte de masse au cours de la simulation en utilisant l'aire du cercle de rayon $r = 0.15$ comme masse de référence. L'évolution du pourcentage absolu de perte de masse pour *PirateNet** et la référence *FEM* est présentée dans la figure 3.8. La perte de masse est plus élevée pour *PirateNet** que pour *FEM*, mais elle reste raisonnable. Leurs erreurs moyennes respectives en pourcentage absolu de masse sont $MAPE_{PirateNet^*} = 1.18\%$ et $MAPE_{FEM} = 0.07\%$. Comme nous l'avons fait avec le disque de Zalesak dans la section 3.2.1, nous avons testé l'effet des améliorations proposées dans la section 2.3.3, mais cette fois sur le problème plus difficile de l'écoulement tourbillonnaire inversé dans le temps.

Nous avons commencé par l'ajout du terme Eikonal à la fonction de perte. Comme nous l'avons constaté précédemment, l'ajout de la perte Eikonal à la mise à jour du poids de normalisation du gradient a aggravé les résultats. Par conséquent, nous avons implémenté la perte Eikonal comme terme de régularisation qui a été maintenu constant tout au long de l'entraînement. En utilisant les trois configurations de référence, nous avons testé une gamme de poids, λ_{eik}, et mesuré leur influence sur l'erreur. Conformément à notre expérience précédente, nous avons constaté que l'ajout d'un terme de perte eikonale à la fonction de perte n'apporte aucune amélioration significative à la solution. Cependant, dans certains cas, comme avec *Default* et une pondération

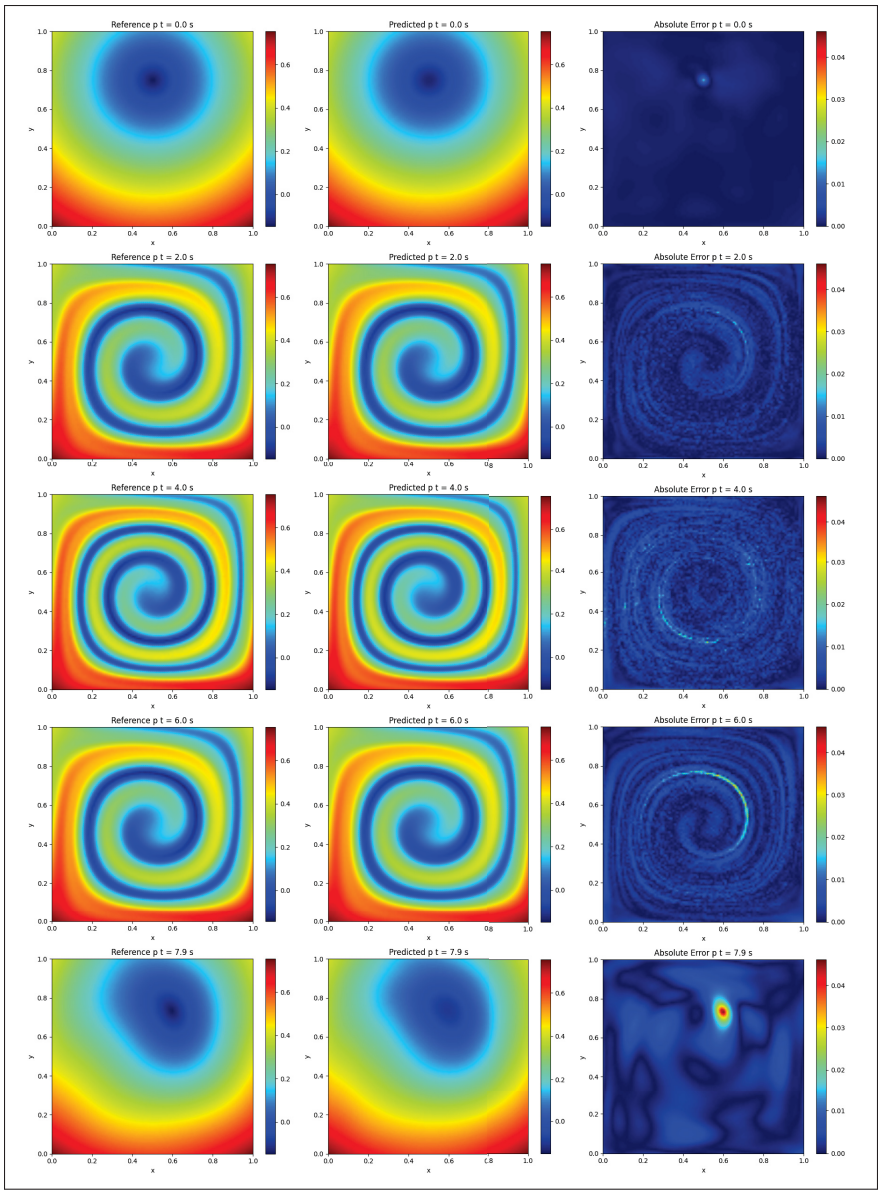


Figure 3.7 Level set tourbillon : Référence (gauche), Prédiction (milieu) et erreur absolue (droite) des champs ϕ à différents pas de temps

soigneusement choisie, la perte eikonale traitée comme terme de régularisation peut légèrement améliorer les résultats.

En traçant les champs level set prédits dans la figure 3.10 avec $\lambda_{eik} = 0.01$, nous remarquons que le terme Eikonal tend à maintenir l'interface dans une forme circulaire. Cela empêche l'interface de s'étirer et de se déformer comme cela est nécessaire avec ce problème. Nous

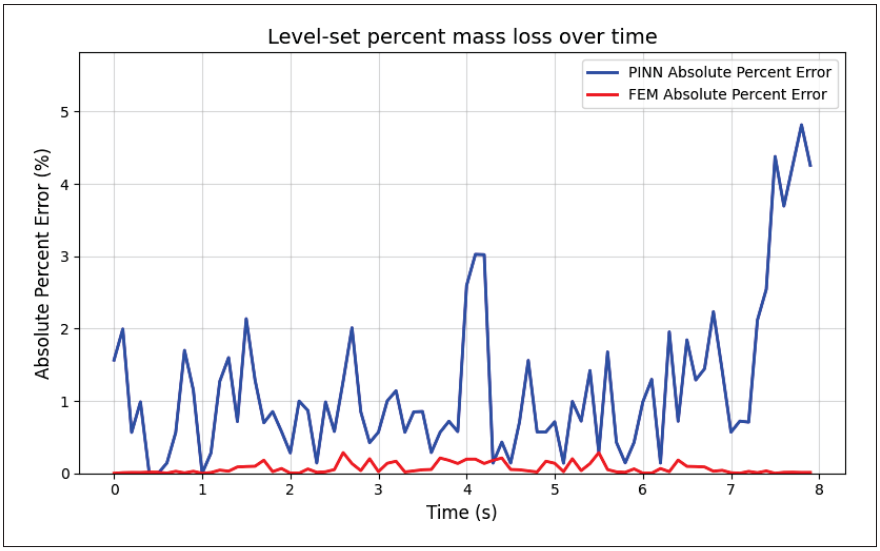
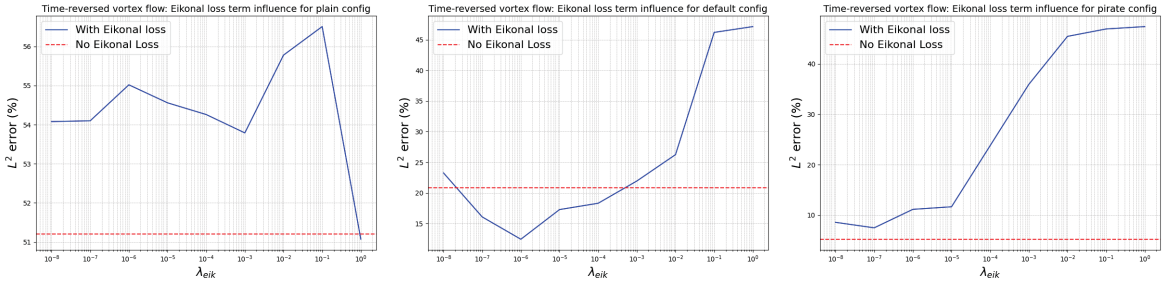


Figure 3.8 Level set tourbillon : évolution de la perte de masse avec *PirateNet** entièrement entraîné



a) *Plain* b) *Default* c) *PirateNet*

Figure 3.9 Level set tourbillon : Influence du terme de perte Eikonal λ_{eik}.

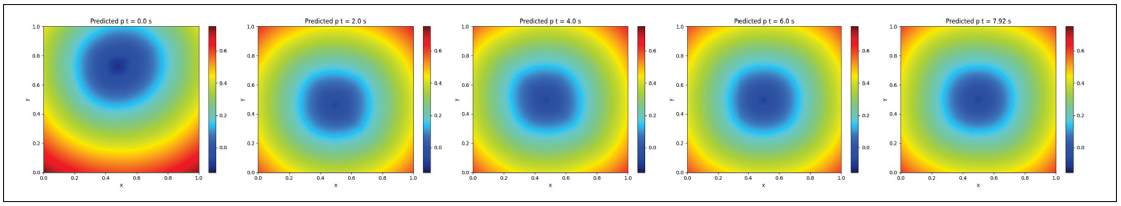


Figure 3.10 Level set tourbillon : Test du terme de perte Eikonal

avons ensuite testé l'influence de l'ajout d'un terme de perte de masse. En commençant par *Méthode 1* de la section 2.3.3.2, nous avons ajouté la perte de masse directement à la fonction de perte en utilisant l'équation 2.54. En utilisant les mêmes hyperparamètres que *PirateNet**, nous avons formé le réseau en utilisant une taille de lot personnalisée de $128 \times 128 \times 128$ points spatio-temporels pour le terme de perte de masse. Nous avons testé deux approches, λ_{mass} dans la normalisation du gradient ou λ_{mass} maintenue constante avec des valeurs variables de λ_{mass}. En utilisant *Méthode 2*, nous avons implémenté la perte de masse dans un deuxième réseau initialisé avec les poids formés de *PirateNet**. Par souci de cohérence, nous avons conservé la même taille de lot de $128 \times 128 \times 128$ points spatio-temporels. Les résultats sont présentés dans le tableau 3.6. La configuration de référence *PirateNet** sans aucun terme de perte de masse

Tableau 3.6 Level set tourbillon : influence du terme de perte de masse

Méthode	L^2 error (%)	MAPE (%)
PirateNet* [no mass loss]	**0.85**	1.18
Méthode 1 [Grad norm]	1.00	1.06
Méthode 1 [$\lambda_{mass} = 0.1$]	1.08	1.06
Méthode 1 [$\lambda_{mass} = 1$]	1.02	1.66
Méthode 1 [$\lambda_{mass} = 10$]	1.02	1.66
Méthode 1 [$\lambda_{mass} = 100$]	1.00	1.06
Méthode 2	6.53	16.8

reste la meilleure, car aucune des méthodes n'améliore l'erreur L^2 ou l'erreur de masse MAPE. En faisant varier λ_{mass}, nous voyons que le terme de perte de masse a peu d'influence sur la précision du modèle. De plus, d'après la figure 3.11, nous voyons avec *Méthode 2* que la perte de masse ne converge pas après 2000 itérations au deuxième niveau. À partir de cela, nous pouvons conclure qu'une attention particulière doit être portée avant d'inclure le terme de perte de masse dans la fonction de perte des problèmes basés sur des level set. Cela montre à nouveau que l'architecture *PirateNet* est capable d'apprendre la solution à des problèmes complexes basés sur des level set sans ajouter de termes Eikonal ou de perte de masse.

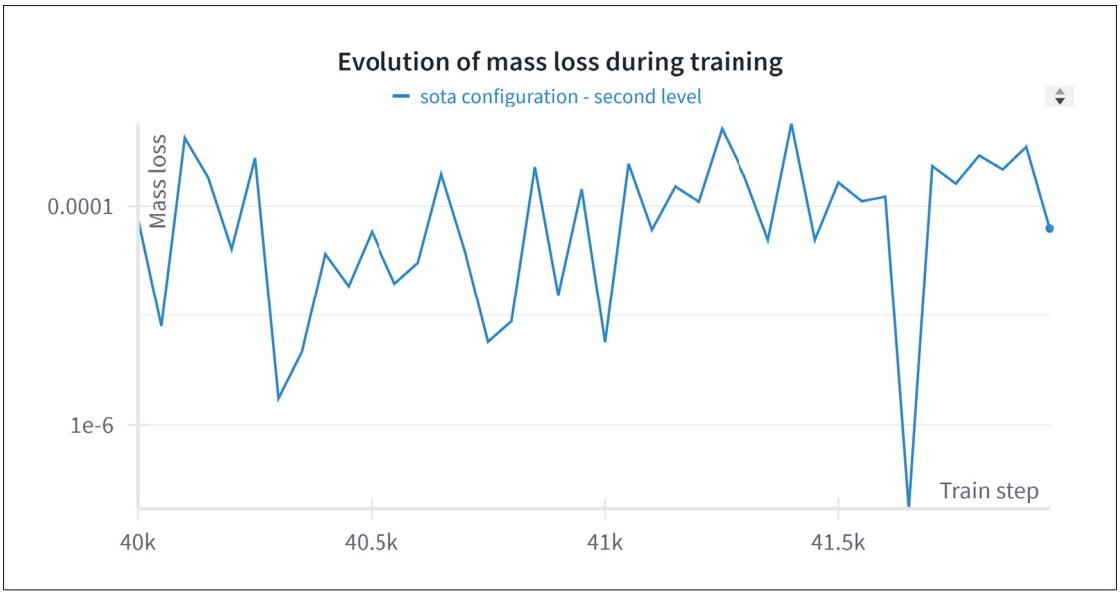


Figure 3.11 Level set tourbillon : Évolution de la perte de masse au cours de l'entraînement (2e niveau)

3.3 Tests des équations de Saint-Venant

Cette section présente les résultats des problèmes de références pour les équations de Saint-Venant, la propagation d'une vague sur une digue et la rupture de barrage radiale. Avant de montrer les résultats, nous présentons une nouvelle métrique d'évaluation de nos PINNs soit, l'erreur de vitesse de propagation de la vague.

3.3.1 Calcul de la vitesse des ondes

Au-delà de l'évaluation de la capacité de notre solveur PINN à reproduire fidèlement les champs de sortie des équations de Saint-Venant (SWE), nous nous intéressons particulièrement à sa capacité à capturer la dynamique sous-jacente du système. En particulier, nous évaluons si le PINN peut modéliser correctement la vitesse de propagation des ondes et le comportement des chocs.

En raison de leur nature intrinsèquement lisse, les PINNs peuvent avoir du mal à représenter des transitions abruptes et des discontinuités, telles que celles rencontrées dans les ondes de choc

ou les fronts d'onde à propagation rapide. Pour mieux comprendre la fidélité dynamique de la solution PINN, nous calculons et analysons les vitesses caractéristiques des ondes dérivées à la fois des solutions prédites et de référence.

Le système d'équations 2.18-2.20 est hyperbolique, et ses vitesses caractéristiques, correspondant aux valeurs propres de la matrice Jacobienne de la fonction de flux, sont données par :

$$\lambda_1 = u - \sqrt{gh}, \quad \lambda_2 = u + \sqrt{gh}, \tag{3.15}$$

représentant respectivement les vitesses de propagation des ondes vers la gauche et vers la droite.

Étant donné les sorties prédites par le PINN $u_{\text{pred}}(x,t)$ et $h_{\text{pred}}(x,t)$, ainsi qu'une solution de référence $u_{\text{ref}}(x,t)$ et $h_{\text{ref}}(x,t)$, nous calculons les vitesses d'onde comme suit :

$$\begin{aligned} \lambda_1(x,t) &= u(x,t) - \sqrt{gh(x,t)}, \\ \lambda_2(x,t) &= u(x,t) + \sqrt{gh(x,t)}. \end{aligned} \tag{3.16}$$

Pour quantifier l'erreur de prédiction des vitesses d'onde, nous calculons l'erreur absolue :

$$\varepsilon_{\pm}(x,t) = \left| \lambda_{\pm}^{\text{pred}}(x,t) - \lambda_{\pm}^{\text{ref}}(x,t) \right|, \tag{3.17}$$

où l'indice ± désigne les ondes se propageant vers la gauche (−) ou vers la droite (+).

L'erreur moyenne absolue est ensuite obtenue en moyennant sur le domaine temporel et spatial :

$$\bar{\varepsilon}_{\pm} = \frac{1}{N_t N_x} \sum_{j=1}^{N_t} \sum_{i=1}^{N_x} \varepsilon_{\pm}(x_i, t_j), \tag{3.18}$$

où N_t et N_x sont les nombres de points de discrétisation temporelle et spatiale.

Capturer avec précision les vitesses des ondes garantit que le PINN ne se contente pas uniquement d'approximer correctement les valeurs de la solution, mais qu'il reflète également la bonne propagation de l'information physique à travers le domaine.

3.3.2 Propagation d'une vague sur une digue

Dans cette section, nous présentons les résultats de simulation pour le cas de propagation d'une vague sur une digue à l'aide des SWE en 1D. Ce scénario modélise l'écoulement d'une masse de fluide initialement sous la forme d'une bosse gaussienne qu'on laisse s'écrouler simplement à l'aide de la gravité. Cette masse forme des vagues qui se propagent ensuite sur une digue aussi sous la forme d'une bosse gaussienne. L'objectif est d'analyser la propagation des ondes en présence d'une bathymétrie variable et d'étudier la capacité des PINNs à capturer ces dynamiques complexes. Un problème similaire a été étudié dans Leiteritz *et al.* (2021), cependant nous utilisons une version avec des conditions frontières différentes.

Nous considérons un domaine unidimensionnel où $x \in [-1, 1]$ m et $t \in [0, 1]$ secondes. De plus, la constante gravitationnelle est fixée à $g = 3.5\frac{m}{s^2}$.

Les conditions initiales sont définies comme suit :

$$\begin{aligned} b(x) &= 0.8 \cdot \exp\left(\frac{-x^2}{0.2^2}\right) - 1, \\ h(x,0) &= 0.2 \cdot \exp\left(\frac{-(x+0.4)^2}{0.2^2}\right) - b(x), \\ u(x,0) &= 0, \end{aligned}$$

où $b(x)$ est la bathymétrie initiale définie par une bosse gaussienne et $h(x, 0)$ est la masse d'eau initiale que nous laissons s'écrouler sous l'effet de la gravité. Les vitesses sont initialisées à zéro.

Les conditions aux limites imposées sont :

$$u(x_0, t) = u(x_1, t) = 0$$

Pour valider nos résultats, nous comparons les prédictions obtenues avec les PINNs à une solution de référence calculée à l'aide du package *Pyclaw* Ketcheson *et al.* (2012), qui implémente des méthodes de volumes finis pour la résolution des équations hyperboliques. La configuration du

problème suit une modification légère d'un exemple standard fourni par *Pyclaw* pour les SWE en 1D.

La solution de référence est obtenue à l'aide d'un solveur de Riemann intégré à *Pyclaw*, assurant une capture précise des discontinuités de l'écoulement. Une interpolation est ensuite appliquée afin d'obtenir une représentation compatible avec l'implémentation en *JAX*, facilitant ainsi une comparaison directe avec les PINNs.

Les résultats obtenus sont évalués en comparant les prédictions des hauteurs d'eau h et des vitesses u avec la solution de référence. Une animation vidéo est disponible sur le GitHub : github.com/m-mullins/SWE-PINN. Nous analysons notamment la capacité des différentes configurations de PINN (*Plain*, *Default* et *PirateNet*) à capturer les variations spatiales de l'écoulement, en mettant en avant les performances des architectures testées. Les hyperparamètres de chacune de ces configurations sont présentés dans le tableau I-5. Ici encore, nous comparons les versions avec hyperparamètres de bases ainsi que celles avec les hyperparamètres provenant d'une optimisation bayésienne, notés avec un *. Le nombre d'itérations d'entraînement diffère pour chaque configuration afin que chacune puisse converger.

Lors de l'optimisation des hyperparamètres, plusieurs configurations du modèle *Default* ont rencontré des échecs durant l'entraînement. Ces plantages sont dus à la complexité du paysage de la fonction de perte, qui peut entraîner une explosion des gradients. L'ajout de diffusion numérique permet d'atténuer ce problème, bien qu'il subsiste encore quelques cas instables. Ce phénomène peut être entièrement contrôlé en appliquant un *gradient clipping* avec une borne de $\pm 10^3$, ce qui s'est avéré suffisant pour tous les cas étudiés dans ce mémoire.

Ainsi, toutes les configurations incluent à la fois de la diffusion numérique et une limitation des gradients. Comme le souligne Fuks & Tchelepi (2020), la valeur de la viscosité artificielle utilisée dans la diffusion influence fortement son efficacité. Après avoir testé différentes valeurs, il a été déterminé que $\nu = 10^{-4}$ offre le meilleur compromis pour nos simulations. Sauf mention contraire, cette valeur est utilisée dans l'ensemble des expériences portant sur la propagation d'une vague sur une digue.

Le tableau 3.7 démontre que *Default** et *PirateNet** performent similairement avec des erreurs $L_h^2 = 0.30\%$ $L_u^2 = 6.36\%$et $L_u^2 = 6.53\%$ respectivement. Cependant, le PINN standard de *Plain** obtient des erreur près de 2 fois plus élevées. Bien que les erreurs ne soit pas mauvaises avec des hyperparamètres choisis selon notre expertise, l'intérêt d'une optimisation des hyperparamètres se révèlent par les erreurs qui sont presque diminuées de moitié. Les paramètres et résultats de cette optimisation sont disponibles à l'annexe 2.3.

Tableau 3.7 SWE 1D propagation d'une vague sur une digue : Erreurs L^2 des prédictions de chacun des champs de sortie

Configuration name	$L_h^2(\%)$	$L_u^2(\%)$	$MAPE_{mass}$ (%)
Plain	0.65	13.33	0.027
Default	0.60	11.73	**0.010**
PirateNet	0.52	10.19	**0.013**
Plain*	0.52	10.85	0.024
Default*	**0.30**	**6.36**	0.025
PirateNet*	**0.30**	**6.53**	0.021

L'erreur de prédiction des champs de vitesses s'avère nettement plus élevée que celle associée à la hauteur d'eau, ce qui s'explique par la nature intrinsèque de ces variables. Les figures 3.12 et 3.13 illustrent l'évolution temporelle des champs prédits et montrent que la majorité de l'erreur se concentre sur la vague en propagation. À noter ici que la bathymétrie n'est pas prédite, c'est une constante dans le modèle. Cela explique pourquoi son erreur est nulle. Ces figures révèlent également que les champs de vitesse et de quantité de mouvement présentent des discontinuités et des gradients plus marqués que les champs de hauteur ou de surface libre. Ce contraste justifie la plus grande difficulté à modéliser précisément les vitesses par rapport à la hauteur d'eau.

D'autre part, la vitesse de l'onde a été calculé à l'aide de l'équation 3.18. Les erreurs moyennes absolues en pourcentage de vitesse de propagation de l'onde sont de $MAPE_{c+} = 0.33\%$ et $MAPE_{c+} = 0.32\%$ respectivement pour *Default** et *PirateNet**. Cela démontre que bien l'erreur de vitesse soit élevée, la vitesse de propagation de l'onde peut être bien captée par les PINNs.

Une animation vidéo est disponible sur le Github.

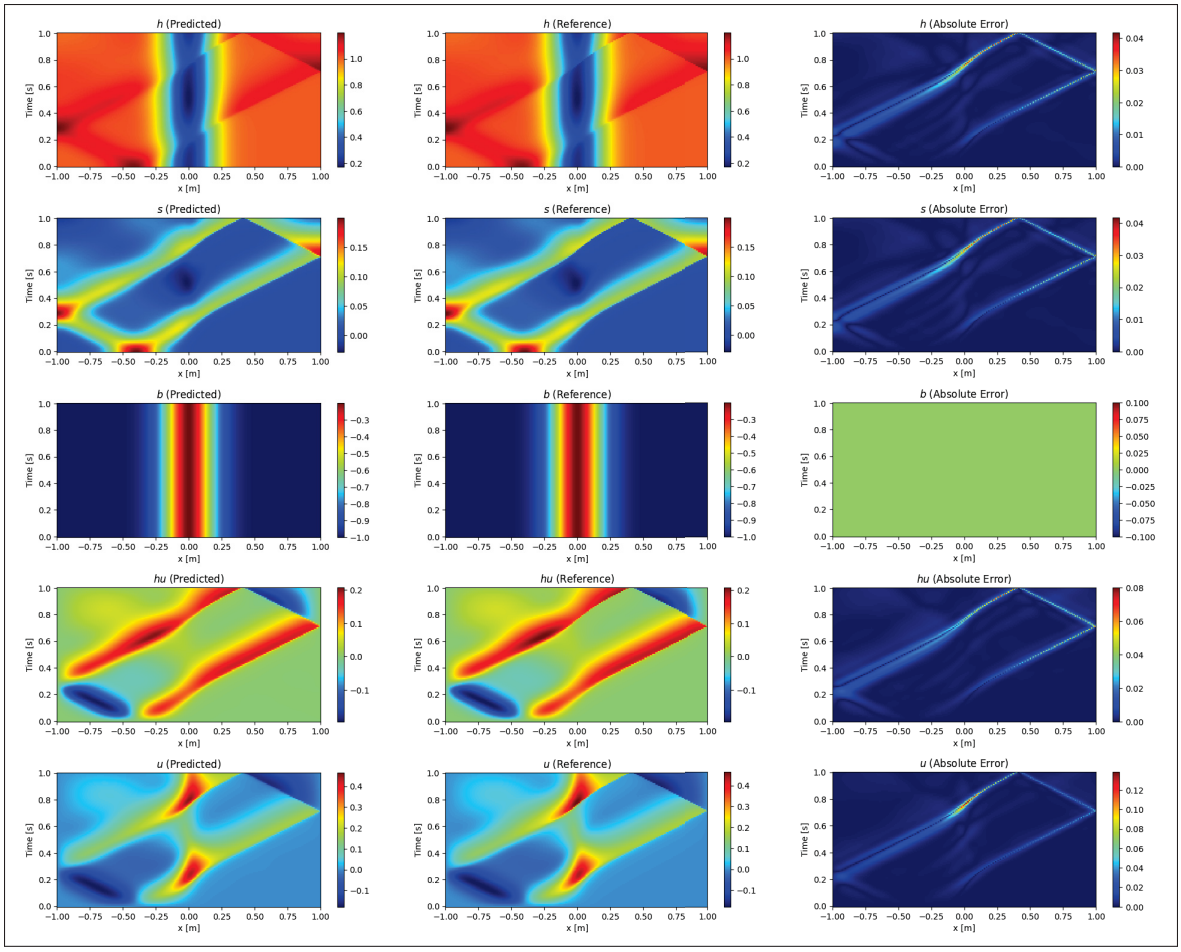


Figure 3.12 SWE 1D propagation d'une vague sur une digue : Prédiction, référence et erreur absolue *PirateNet*

3.3.2.1 Forme des équations

Le choix d'utiliser la forme non conservative des équations de Saint-Venant est ici validé. Cette décision repose sur le besoin d'éviter les divisions par zéro au début de l'entraînement, lorsque la hauteur d'eau h peut atteindre des valeurs nulles. Une solution à ce problème consiste à régulariser le terme en ajoutant une constante ϵ à la hauteur, soit $h = h + \epsilon$. La valeur $\epsilon = 10^{-4}$ a été retenue, car elle représente le plus petit terme testé qui n'entraîne ni explosion des gradients ni erreur lors de l'entraînement. Le tableau 3.8 indique que les erreurs sur les vitesses et la hauteur sont comparables entre les deux formulations. Toutefois, la conservation de masse se révèle moins précise avec la forme conservative, ce qui s'explique par l'ajout artificiel de masse

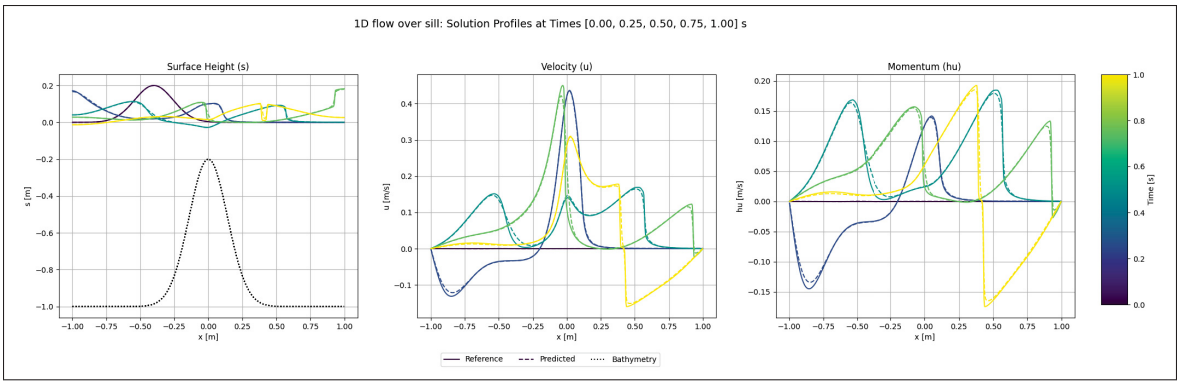


Figure 3.13 SWE 1D propagation d'une vague sur une digue : Surface libre, vitesse et momentum pour *PirateNet* à $t = [0, 0.25, 0.5, 0.75, 1.0]$s

Tableau 3.8 SWE 1D propagation d'une vague sur une digue : Comparaison des performances des équations conservatrices et non conservatrices avec *PirateNet*

Configuration name	L_h^2(%)	L_u^2(%)	$MAPE_{mass}$ (%)
Conservatrices	0.51	11.4	0.08
Non conservatrices	0.52	10.2	0.03

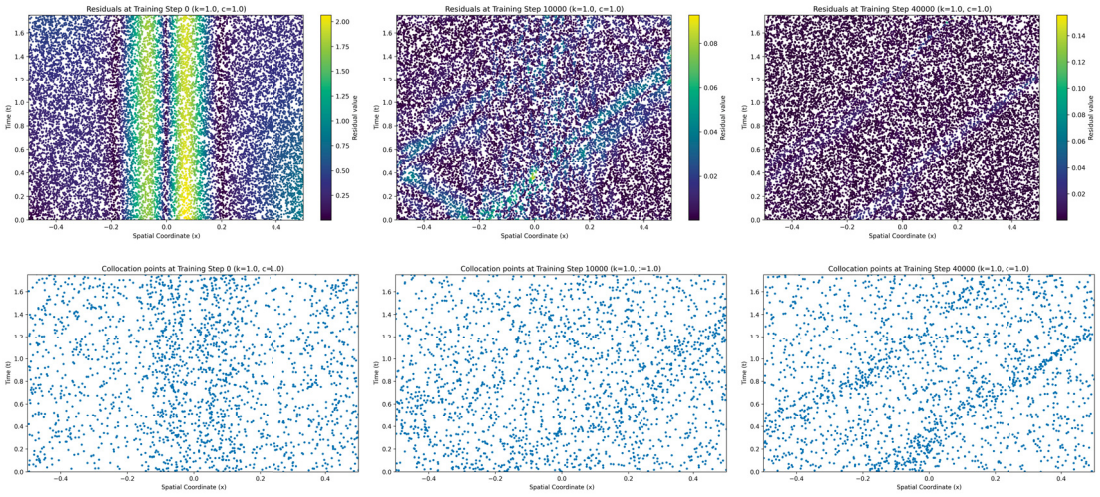
dû à la régularisation $h = h + \epsilon$. Ces résultats confirment la pertinence du choix de la formulation non conservative pour ce problème.

3.3.2.2 Influence de la méthode d'échantillonnage

Cette section porte sur l'expérimentation avec différentes méthodes visant à influencer la méthode d'échantillonnage. Ces méthodes sont la distribution adaptative des points de collocations basée sur les résiduels (RAD), vu à la section 2.3.2.6 ainsi que le masque d'attention résiduel appliqué aux points de collocations de la section 2.3.2.7. Les comparaisons de cette section sont entre le statut quo, un échantillonnage aléatoire à chaque itération, et la méthode étudiée (RAD ou RBA). À noter que RAD et RBA ne peuvent pas être utilisés ensemble.

Utilisant la méthodologie décrite à la section 2.3.2.6, la méthode d'échantillonnage est appliqué au problème de propagation d'une vague. La figure 3.14 présente les résiduels de tous les points

candidats à trois moments dans l'entraînement de *PirateNet*. Elle montre aussi les points de collocations choisis parmi les candidats. On constate qu'avec les hyperparamètres recommandés par Wu *et al.* (2023), soit $k = 1$ et $c = 1$, les points choisis sont généralement bien distribués, mais avec une concentration notable aux endroits à résiduels élevés. Dans ce cas, le résiduel élevé se trouve au front de propagation de la vague.



a) Itération 0 b) Itération 10k c) Itération 40k

Figure 3.14 RAD : résiduels (haut) et points de collocations échantillonnés (bas) avec $k = 1$ et $c = 1$

L'échantillonnage par RAD nécessite de recalculer les résiduels à un nombre de points plus élevé que la taille de lot. Chaque nouvel échantillonnage est donc bien plus couteux qu'un échantillonnage aléatoire standard. Cela dit, il est intéressant d'évaluer quelle fréquence de rééchantillonnage permet d'obtenir le meilleur compromis entre la qualité des résultats et le temps de calcul. Pour ce faire, plusieurs fréquences de rééchantillonnage ont été testées avec *PirateNet*. La figure 3.15 démontre qu'en haut de 5, le rééchantillonnage n'est pas assez fréquent et la qualité des résultats est impactée. Cependant, le temps de calcul est régi par une fonction exponentielle, une fréquence inférieure à 5 est donc très couteuse. Une fréquence de rééchantillonnage de 5 est donc utilisée pour la suite.

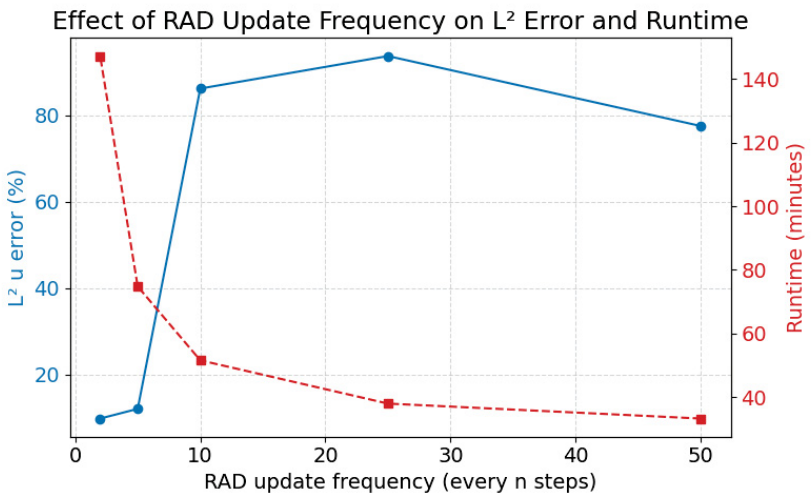


Figure 3.15 RAD : Impact de la fréquence de rééchantillonnage sur les résultats

Bien que $k = 1$ et $c = 1$ soient des hyperparamètres généralement acceptables pour l'implémentation de RAD selon Wu *et al.* (2023), il est possible qu'un autre ensemble soit plus performant pour notre problème. En utilisant une fréquence de rééchantillonnage de 5 ainsi que les hyperparamètres optimisés de *Plain**, un ensemble de valeurs de k et c a été testé. La figure 3.16 démontre que $k = 1$ donne généralement les meilleurs résultats pour ce problème avec une erreur minimale de $L_u^2 = 11.7\%$ avec $c = 4$. Cependant, cette valeur demeure plus élevée que *Plain** avec échantillonnage aléatoire qui obtient une erreur $L_u^2 = 10.9\%$. De plus, en enlevant toute optimisation des hyperparamètres, c'est-à-dire en utilisant les hyperparamètres recommandés de *Plain* et ceux de RAD soit $k = 1$ et $c = 1$, RAD n'arrives pas à battre le statut quo. *Plain* sans RAD obtient une erreur $L_u^2 = 13.3\%$ alors que *Plain* avec RAD obtient $L_u^2 = 13.6\%$.

L'influence de la taille de lot a également été testée. En utilisant les hyperparamètres optimisés de *Plain** et avec $k = 1$ et $c = 1$, l'évolution de l'erreur à travers l'entraînement est présentée à la figure 3.17 pour des PINNs avec et sans RAD. La ligne pleine représente l'erreur moyenne des 4 tailles de lots testées, soit $1024, 2048, 4096, 8192$. La partie ombragée représente les erreurs minimales et maximales. Cette figure illustre le fait qu'encore une fois RAD n'arrive pas à battre un PINN avec échantillonnage aléatoire.

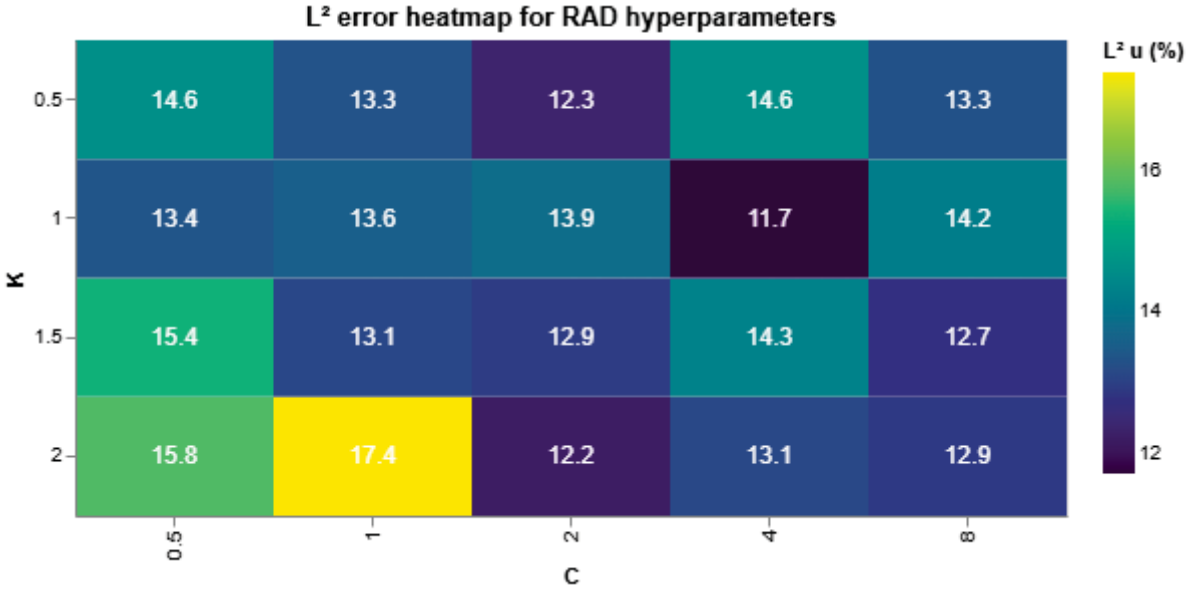


Figure 3.16 RAD : Influence de k et c sur l'erreur L_u^2 avec *Plain**

La méthode RBA a également été testée sur ce problème de SWE en 1D en utilisant la méthodologie de la section 2.3.2.7. La figure 3.18 présente l'évolution des poids RBA lors de l'entraînement pour le problème de SWE en 1D. L'influence du type d'échantillonnage et de la taille de lot a été testée avec les configurations *Plain** et *Default** ainsi qu'avec les hyperparamètres recommandés de RBA, $\gamma = 0.999$ et $\eta = 0.01$. Les types d'échantillonnages testés avec RBA sont l'échantillonnage fixe et espacé linéairement ainsi que l'échantillonnage structuré-aléatoire. Sans RBA, l'échantillonnage est aléatoire. Noter que contrairement à RAD, RBA est très peu coûteux, le rééchantillonnage peut donc se faire à chaque itération d'entraînement. La figure 3.19 démontre que cette étude n'a pas permis de trouver de combinaison avec RBA permettant de surpasser les performances sans RBA. Au contraire, pour le problème étudié, l'ajout de RBA semble rentre l'entraînement encore plus instable.

Bien que cette étude n'ait pas démontré que RAD et RBA puisse améliorer les performances d'un PINN standard avec échantillonnage aléatoire, il n'est présentement pas possible d'écarter ces méthodes complètement. RAD ayant été testé sur plusieurs autres problèmes de référence

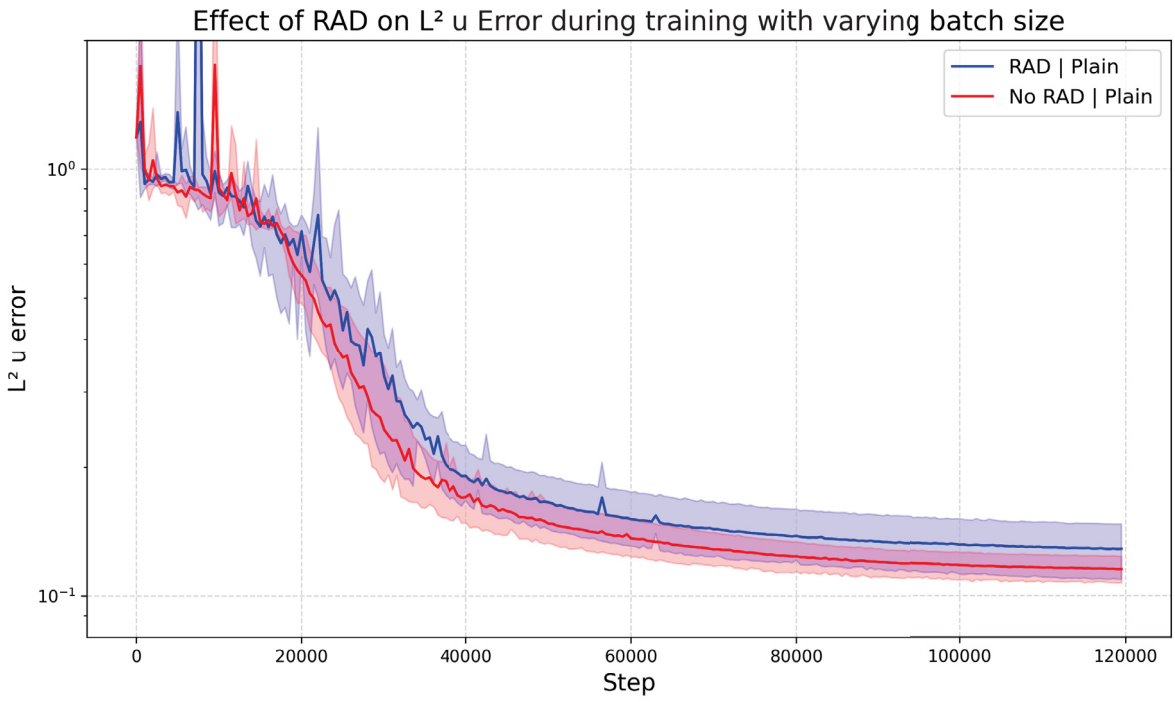
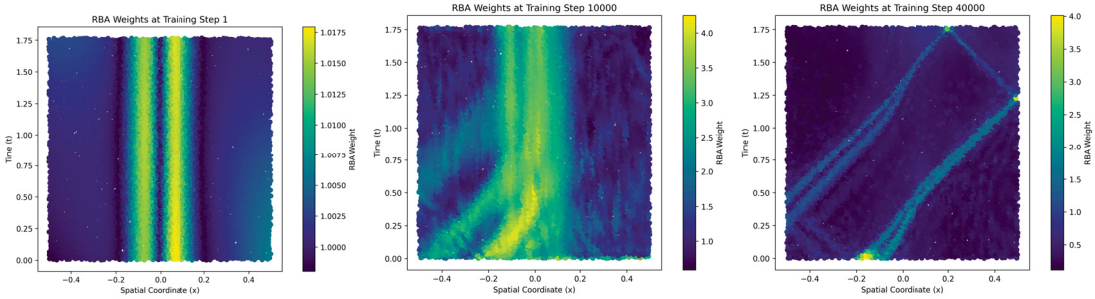


Figure 3.17 RAD : Évolution de l'erreur L_u^2 avec *Plain** et différentes tailles de lots (1024, 2048, 4096, 8192)



a) Itération 1 b) Itération 10k c) Itération 40k

Figure 3.18 RBA : poids associés aux points de collocations échantillonné

avec d'excellents résultats, une étude plus approfondie de cette méthode appliquée aux SWE devrait être faite afin de mener à une conclusion plus claire. Il en va de même pour RBA.

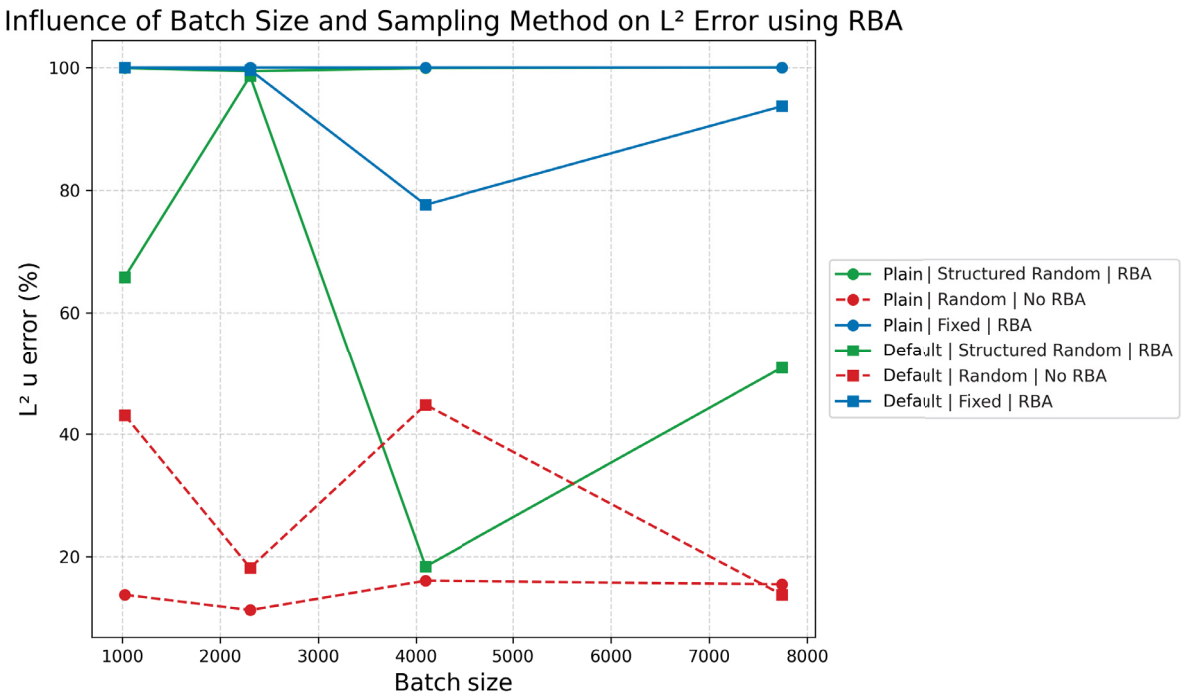


Figure 3.19 RBA : Influence de l'échantillonneur et de la taille de lot

3.3.2.3 Impact individuel des améliorations

Afin d'évaluer l'impact individuel des améliorations aux PINNs recensées dans la section 2.3.2, une étude d'impact et une étude d'ablation ont été effectuées. Tout d'abord, l'étude d'impact consiste à prendre un PINN standard avec les hyperparamètres optimisés de *Plain** et y ajouter individuellement chaque amélioration en mesurant son impact sur l'erreur L^2. Le tableau 3.9 présente les résultats de cette étude.

Pour l'étude d'ablation, la configuration *Default** contenant toutes les améliorations sert de point de départ. Les performances du réseau sont alors évaluées individuellement lorsqu'une amélioration est enlevée. Le tableau 3.10 présente les résultats de cette étude. Ces deux études permettent de dégager plusieurs conclusions importantes. Tout d'abord, la configuration sans caractéristiques de Fourier aléatoires présente les erreurs les plus élevées. En leur absence, le PINN ne parvient pas à apprendre efficacement, ce qui souligne le rôle crucial de ces caractéristiques dans la modélisation de notre problème, qui contient des composantes oscillatoires issues des

Tableau 3.9 SWE 1D propagation d'une vague sur une digue : étude d'impact des améliorations recensées

Architecture name	Causal training	Fourier feature	Grad norm	RWF	Numerical diffusion	RAD	RBA	L_h^2 (%)	L_u^2 (%)	$MAPE_{mass}$ (%)	Runtime (min)
MLP								0.71	14.2	0.031	111
MLP	x							0.66	13.4	0.031	110
MLP		x						**0.58**	**11.8**	**0.018**	115
MLP			x					0.75	14.9	**0.015**	113
MLP				x				0.68	13.8	0.033	127
MLP					x			**0.64**	**12.8**	0.026	122
MLP						x		0.72	14.3	0.026	109
MLP							x	1.21	23.5	0.069	116

Tableau 3.10 SWE 1D propagation d'une vague sur une digue : étude d'ablation des améliorations recensées

Architecture name	Causal training	Fourier feature	Grad norm	RWF	Numerical diffusion	L_h^2 (%)	L_u^2 (%)	$MAPE_{mass}$ (%)
MMLP	x	x	x	x	x	**0.29**	**6.23**	0.025
MMLP		x	x	x	x	0.76	14.7	0.057
MMLP	x		x	x	x	4.40	99.8	**0.008**
MMLP	x	x		x	x	1.28	24.0	0.277
MMLP	x	x	x		x	**0.35**	**6.8**	0.029
MMLP	x	x	x	x		0.87	16.9	0.032
PirateNet	x	x	x	x	x	**0.35**	**7.4**	0.021

équations de Saint-Venant. Ce constat est en accord avec les résultats de la littérature sur les PINNs appliqués à des phénomènes oscillatoires Tancik *et al.* (2020). À noter que la petite perte de masse pour la configuration sans Fourier ne permet pas à elle seule de dire que le modèle performe bien. Au contraire, en regardant l'erreur L^2, il est possible de voir que le PINN n'apprends tout simplement pas l'évolution temporelle, il reste pris avec la condition initiale. Puisque l'erreur de masse se fie à la condition initiale, l'erreur de masse est par conséquent très faible.

Par ailleurs, l'ablation de la factorisation aléatoire des poids semble avoir peu d'impact sur la performance du modèle. Cela suggère que cette composante, bien que potentiellement utile,

n'est pas essentielle dans le contexte de notre problème et pourrait être omise sans dégradation significative des résultats.

3.3.2.4 Résilience aux hyperparamètres

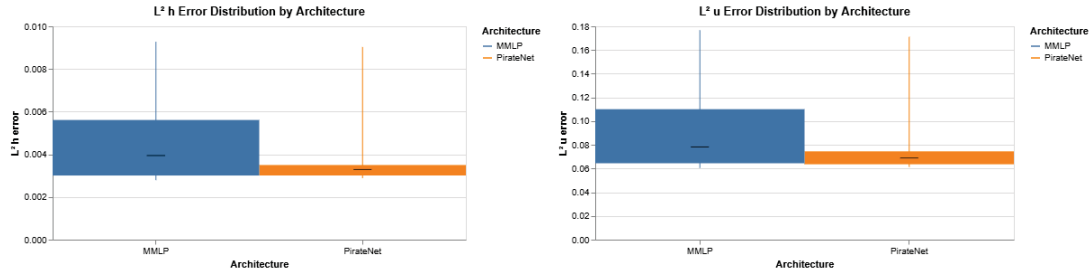
Dans le but d'investiguer l'influence qu'a le choix des hyperparamètres sur le choix de l'architecture entre le MLP modifié (MMLP) et PirateNet, nous avons testé toutes les combinaisons d'hyperparamètres pour les configurations *Default* et *PirateNet*. L'espace des hyperparamètres contient donc toutes les combinaisons possibles d'hyperparamètres. Cet espace a été défini en fonction des recommandations d'hyperparamètres contenus dans Wang *et al.* (2023) et notre expérience. Les combinaisons possibles se trouvent au Tableau 3.11. Un total de 54 combinaisons ont donc été testées pour chaque architecture.

Tableau 3.11 SWE 1D écoulement sur une bosse : Espace d'hyperparamètres pour l'étude de résilience

Hyperparameter	**Possible values**
Architecture	MMLP, PirateNet
Activation function	Tanh, Gelu, Swish
Fourier feature scale	1, 1.5, 2
RWF mean	0.5, 1
Causal tolerance	1, 1.5, 2

La figure 3.20 présente les résultats de l'analyse sous forme de boîtes à moustaches. On observe d'abord que l'erreur minimale est comparable pour les deux architectures. Toutefois, la dispersion des résultats est visiblement plus grande pour le MMLP, ce qui suggère une plus forte sensibilité aux choix d'hyperparamètres. Cette observation est appuyée par les valeurs de moyenne et d'écart type pour l'erreur L_u^2 de $\mu = 7.4\%$, $\sigma = 1.9\%$ pour PirateNet (*PirateNet*) et $\mu = 9.1\%$, $\sigma = 3.1\%$ pour le MMLP (*Default*). Le phénomène est semblable pour l'erreur L_h^2 avec des erreurs respectives de $\mu = 0.35\%$, $\sigma = 0.10\%$ pour PirateNet (*PirateNet*) et

$\mu = 0.45\%$, $\sigma = 0.17\%$ pour le MMLP (*Default*). En somme, bien que les deux architectures puissent atteindre des performances comparables après optimisation, PirateNet se distingue par une meilleure robustesse aux choix d'hyperparamètres. Elle représente donc un choix plus fiable lorsque l'optimisation fine des hyperparamètres n'est pas envisageable.



a) L_h^2 b) L_u^2

Figure 3.20 Résilience de l'architecture au choix des hyperparamètres

3.3.2.5 Vagues en interférence

Un dernier test que nous avons effectué est de tester la capacité du PINN à prédire de l'interférence constructive lorsque deux fronts de vagues se rencontrent. Pour ce faire nous gardons la configuration *PirateNet* avec le même problème de base, mais cette fois avec une constante gravitationnelle, $g = 9.8\frac{m}{s^2}$. Nous constatons visuellement à la Figure 3.21 que les vagues sont généralement bien captées. Cependant, sans surprise les erreurs $L_h^2 = 0.71\%$, $L_u^2 = 15.4\%$ et $MAPE_{mass} = 0.07\%$ sont légèrement plus élevées. Cela est normal puisqu'avec un g plus élevé, les gradients sont plus élevés aussi et le problème est maintenant plus difficile.

3.3.2.6 Coût informatique

Il est aussi intéressant de noter le coût informatique d'entraînement du PINN. La solution de référence, générée avec un solveur de Riemann Ketcheson *et al.* (2012) prend environ 10 secondes à résoudre sur un CPU. Cependant, l'entraînement du Plain PINN et du PirateNet

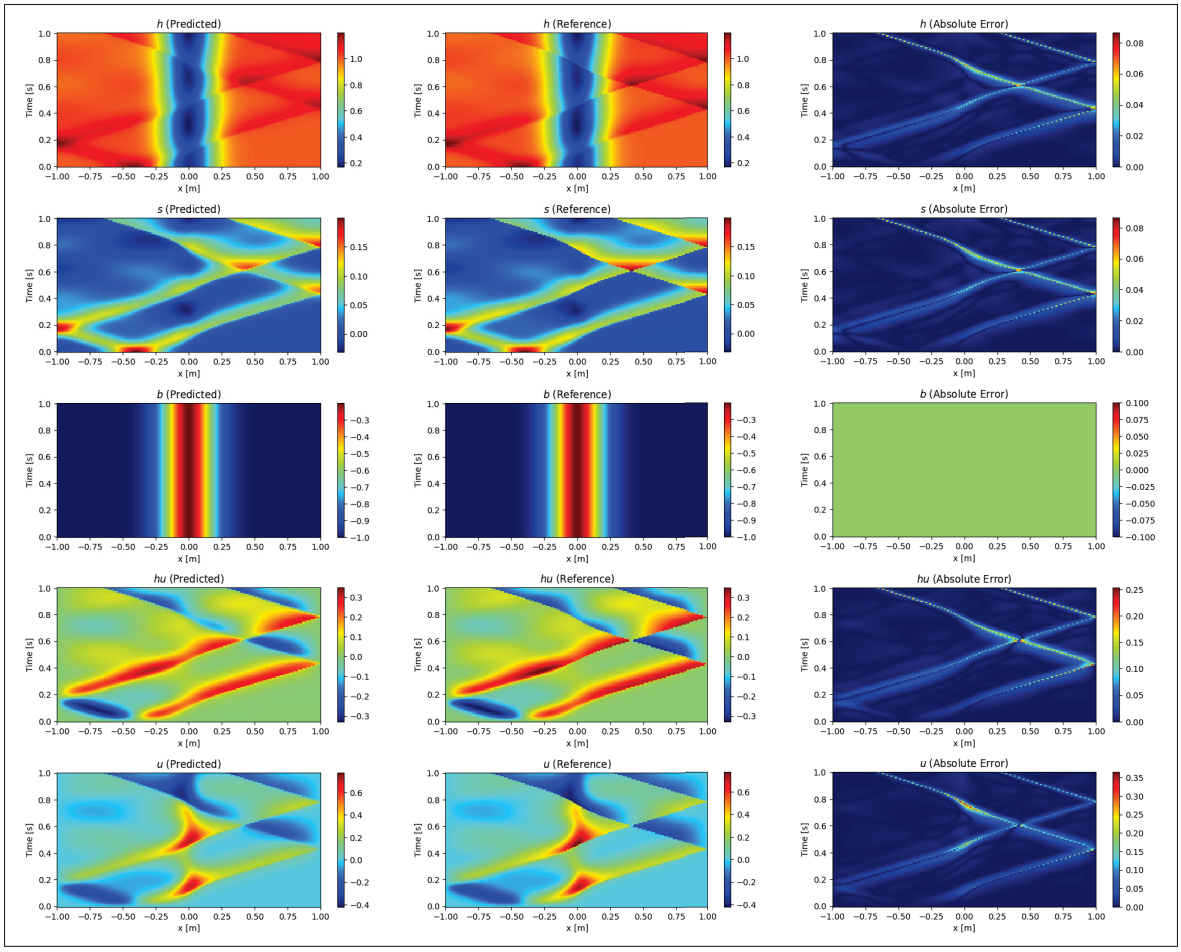


Figure 3.21 SWE 1D propagation d'une vague sur une digue : Prédiction, référence et erreur absolue pour *PirateNet* avec $g = 9.8$

prend respectivement 23 min et 80 min sur un GPU. Le PINN demande donc beaucoup plus de ressources que les méthodes numériques classiques pour ce genre de problème.

3.3.3 Rupture de barrage radial

Dans cette section, nous présentons les résultats de simulation pour le cas de la rupture d'un barrage radial en deux dimensions à l'aide des SWE en 2D. Ce scénario modélise l'écoulement d'un fluide après la rupture soudaine d'un barrage cylindrique initialement rempli d'eau. D'abord étudié par Alcrudo & Garcia-Navarro (1993), ce problème est très populaire dans l'étude des SWE.

Nous considérons un barrage cylindrique de rayon 2.5m, situé au centre d'un domaine carré de 40×40m. L'eau est initialement contenue à l'intérieur du barrage avec une hauteur de 2.5m, tandis que la hauteur de l'eau à l'extérieur est de 1m. À l'instant initial, les vitesses sont nulles dans tout le domaine. L'évolution de l'écoulement est simulée sur une durée de 4 secondes. Une animation vidéo est disponible sur le GitHub : github.com/m-mullins/SWE-PINN.

Les conditions initiales sont donc définies ainsi :

À $t = 0$, la hauteur d'eau $h(x, y, 0)$ est donnée par :

$$h(x, y, 0) = \begin{cases} h_{\text{intérieur}}, & \sqrt{(x-20)^2 + (y-20)^2} \leq r_0, \\ h_{\text{extérieur}}, & \text{autrement.} \end{cases}$$

où $r_0 = 5\,m$ est le rayon initial du barrage et :

$$h_{\text{intérieur}} = 2.0\,m, \quad h_{\text{extérieur}} = 0.5\,m.$$

Pour valider nos résultats, nous comparons les prédictions obtenues avec les PINNs à une solution de référence calculée à l'aide d'un solveur par volumes finis à haute résolution Loukili & Soulaimani (2007); Zokagoa & Soulaïmani (2010); Delmas (2020). Le solveur Cuteflow est disponible au lien github.com/ETS-GRANIT/CuteFlow. Ce solveur utilise un solveur de Riemann pour résoudre les équations de Saint-Venant et garantir une capture précise des discontinuités de l'écoulement. La solution de référence est obtenue sur un maillage de 93 k éléments, puis interpolée sur une grille uniforme de 101×101 points pour être compatible avec l'implémentation en JAX et permettre une comparaison directe avec les PINNs. Une interpolation linéaire est utilisée afin de maintenir une solution appropriée sans nécessiter trop de ressources.

Les résultats obtenus avec les PINNs sont évalués en comparant la hauteur d'eau h et les champs de vitesse u et v avec ceux de la solution de référence.

Nous avons entraîné un ensemble de configurations PINN (*Plain*, *Default*, *PirateNet*) pour apprendre l'évolution de l'écoulement. Les configurations suivent la même norme que dans les sections précédentes. Les configurations suivies d'un * représentent celles avec les hyperparamètres optimisés. Les hyperparamètres détaillés peuvent être trouvés dans le tableau I-6.

Comme avec le problème de SWE en 1D, de la diffusion numérique et du gradient clipping a été ajoutée à chaque configuration afin d'assurer une réussite de l'entraînement. Sauf indication contraire, le coefficient de viscosité numérique utilisé dans cette section est $\nu = 10^{-4}$.

Tel que présenté au tableau 3.12, *Default** est la configuration qui performe le mieux avec des erreurs $L_h^2 = 2.0\%$, $L_u^2 = 23.5\%$ et $L_h^2 = 25.2\%$. Cependant, *PirateNet** obtient des performances similaires. L'intérêt d'une optimisation des hyperparamètres est d'une fois de plus présenté alors que les erreurs sont diminués pour *Plain* et *Default*. *Default* n'a d'ailleurs pas appris au meilleur de ces capacités avec l'ensemble d'hyperparamètres de base. Cependant, c'est *Default** qui obtient le meilleur résultat une fois optimisé. En effet, avec l'optimisation bayésienne des hyperparamètres, un phénomène similaire au test de résilience aux hyperparamètres de la section 3.3.2.4 est remarqué, l'architecture PirateNet est plus robuste aux choix d'est hyperparamètres que le MMLP de *Default*.

Une représentation graphique du dernier pas de temps simulé est disponible pour la prédiction, la solution de référence ainsi que l'erreur absolue à la Figure 3.22.

Tableau 3.12 SWE 2D rupture de barrage radiale : Erreurs L^2 des prédictions de chacun des champs de sortie

Configuration name	$L_h^2(\%)$	$L_u^2(\%)$	$L_v^2(\%)$	$MAPE_{mass}(\%)$
Plain	3.1	33.0	34.5	0.12
Default	8.1	86.9	87.7	0.98
PirateNet	2.3	26.1	27.6	0.09
Plain*	2.9	31.7	33.5	0.20
Default*	**2.0**	**23.5**	**25.2**	0.06
PirateNet*	2.2	25.4	25.9	**0.03**

Tel que constaté avec les résultats de SWE en 1D, les vitesses sont beaucoup plus difficiles à approximer que celles de hauteur. Les figures 3.22 et 3.23 démontrent que la majorité de l'erreur se trouve aux endroits de fortes discontinuités. Cependant, les PINNs n'ont pas de difficulté à prédire la vitesse de propagation des ondes alors que les erreur moyennes absolues en pourcentage sont de $MAPE_{c+} = 2.8\%$, $MAPE_{c+} = 2.5\%$ et $MAPE_{c+} = 2.7\%$ respectivement pour *Plain**, *Default** et *PirateNet**.

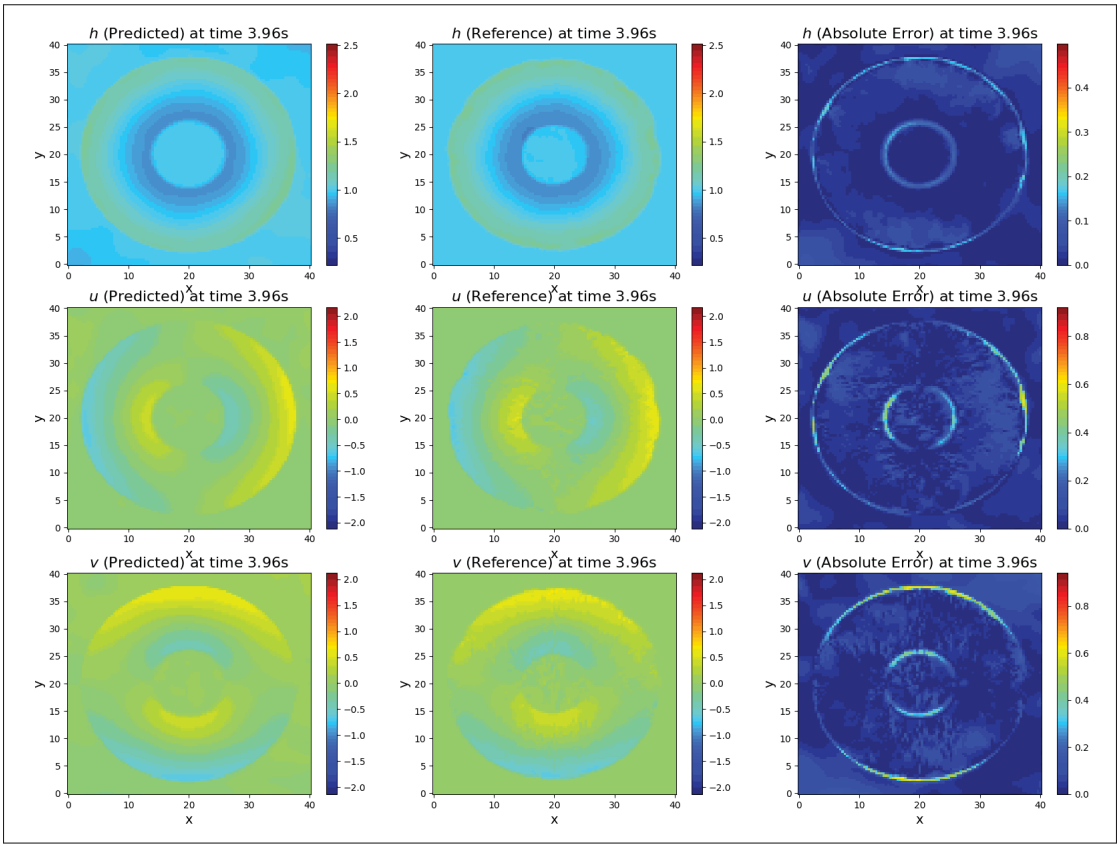


Figure 3.22 SWE 2d rupture de barrage radiale : Référence (gauche), Prédiction (milieu) et erreur absolue (droite) des champs prédits h, u, et v

Le problème de rupture de barrage radial est marqué par une forte discontinuité entre les deux premiers pas de temps. À l'instant initial les champs de vitesses sont tous à zéro alors que dès l'instant un, il y a un fort pic dans ce champ de vitesse. Cela est démontré à la Figure 3.23. Le terme $\mathcal{L}_{ic}$ de la fonction de perte du PINN est donc facilement diminué et peu utile pour ce problème. En effet, la rupture de barrage constitue un cas d'étude populaire dans la

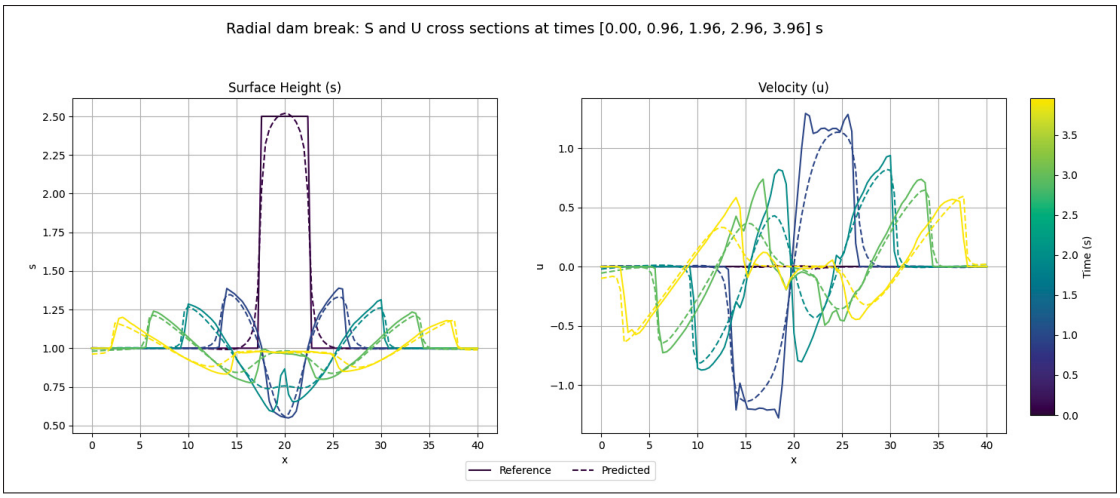


Figure 3.23 SWE 2d rupture de barrage radiale : Évolution temporelle des champs prédits h, u, et v à $y = 0$

littérature en raison de ces discontinuités marquées, qui restent un défi majeur pour les approches basées sur les PINNs. Comme l'indiquent Tian *et al.* (2025), "Given the continuous yet limited approximation capabilities of the PINN function, deviations from strong discontinuous initial conditions are inevitable. These deviations propagate within the characteristic field, progressively increasing the discrepancy between the approximate and exact solutions." Cette observation met en évidence les difficultés inhérentes à l'approximation des solutions avec des discontinuités marquées et souligne la nécessité d'améliorations dans la modélisation des PINNs pour ces types de problèmes.

3.3.3.1 Impact individuel des améliorations

Afin d'évaluer l'impact individuel des améliorations aux PINNs recensées dans la section 2.3.2, une étude d'impact et une étude d'ablation ont été effectuées. La procédure est la même qu'avec le problème de propagation d'une vague en 1D de la section 3.3.2. Le tableau 3.13 présente les résultats de l'étude d'impact.

Comme effectué avec l'étude d'ablation de la section précédente, la configuration *Default** contenant toutes les améliorations sert de point de départ. Les performances du réseau sont alors

Tableau 3.13 SWE 2D rupture de barrage radial : étude d'impact des améliorations recensées

Architecture name	Causal training	Fourier feature	Grad norm	RWF	Numerical diffusion	L_h^2 (%)	L_u^2 (%)	L_v^2(%)	$MAPE_{mass}$ (%)	Runtime (min)
MLP						3.2	33.8	35.2	0.08	**95**
MLP	x					3.2	34.3	35.6	0.09	100
MLP		x				**2.9**	**32.0**	**33.4**	0.11	104
MLP			x			9.4	100	100	**0.00**	97
MLP				x		3.3	35.9	36.9	0.17	**82**
MLP					x	**2.1**	**25.0**	**27.2**	**0.05**	120

évaluées individuellement lorsqu'une amélioration est enlevée. Le tableau 3.14 présente les résultats de cette étude. L'étude d'impact démontre encore une fois l'utilité des caractéristiques

Tableau 3.14 SWE 2D rupture de barrage radial : étude d'ablation des améliorations recensées

Architecture name	Causal training	Fourier feature	Grad norm	RWF	Numerical diffusion	L_h^2 (%)	L_u^2 (%)	L_v^2(%)	$MAPE_{mass}$ (%)
MMLP	x	x	x	x	x	2.3	26.3	27.2	**0.040**
MMLP		x	x	x	x	**2.1**	**24.9**	**25.6**	0.120
MMLP	x		x	x	x	2.3	**24.0**	**26.6**	0.123
MMLP	x	x		x	x	3.0	32.0	35.6	0.439
MMLP	x	x	x		x	2.6	28.4	31.0	0.122
MMLP	x	x	x	x		**2.2**	**24.9**	27.3	**0.028**

de Fourier aléatoires et de la viscosité numérique alors que ce sont les configurations avec les erreurs les plus basses. Bien que la viscosité numérique soit l'ajout le plus efficace pour ce problème, c'est également l'ajout le plus coûteux avec un temps d'entraînement de 113 minutes alors que les autres tournent de 80 à 100 minutes sur le même modèle de GPU de la grappe de calcul. Cela s'explique par le fait que les calculs de quatre dérivées secondes ($u_{xx}, u_{yy}, v_{xx}, v_{yy}$) doivent maintenant être fait à chaque itération. D'autre part, l'ajout de la mise à jour automatique des poids basée sur la norme du gradient présente l'erreur la plus élevée de l'étude d'impact avec un PINN qui n'a pas réussi à s'entraîner convenablement. Cependant, il n'est pas possible de dire que cet ajout est mauvais pour l'entraînement des PINN, car il s'agit d'une seule valeur aberrante. On remarque d'ailleurs dans l'étude d'ablation que c'est sans la mise à jour automatique des

poids qu'on obtient la pire erreur, les résultats semblent contradictoires. Toutefois, cela met en évidence les difficultés inhérentes à l'entraînement des PINN. Parfois, l'entraînement des PINNs échoue et c'est la combinaison de plusieurs techniques qui augmente les chances de réussites.

3.3.3.2 Coût informatique

Notons également le coût informatique de la résolution par PINNs par rapport aux méthodes numériques classiques. La solution de référence générée par volumes finis à pris 3 minutes à résoudre sur un GPU NVDIA V100l de la grappe de calcul Cedar du DRAC. Cependant, sur le même type de GPU, le *Plain* PINN le plus rapide à pris 82 minutes. Les temps de calcul augmentent considérablement lorsqu'on ajoute des améliorations avec *Default* et *PirateNet* qui ont respectivement pris 115 et 174 minutes. Cela démontre que les PINNs sont conceptuellement attirants, mais qu'à leur état actuel, ils sont loin d'être capables de remplacer les méthodes numériques pour la résolution de problèmes directs purs dues à leur coût en calcul bien plus élevé.

CONCLUSION ET RECOMMANDATIONS

4.1 Conclusion

Cette étude a mis en évidence l'efficacité des architectures avancées de PINNs, notamment PirateNet, pour résoudre des problèmes complexes impliquant les méthodes level set et les équations de Saint-Venant. Grâce à une série de tests numériques, les PINNs augmentées par les améliorations recensées de factorisation aléatoire des poids, respect de la causalité temporelle, équilibrage des poids de la fonction de perte par norme de gradient, apprentissage séquence-par-séquence, diffusion numérique, les caractéristiques de Fourier aléatoires et l'architecture PirateNet ont surpassés les architectures classiques de PINNs en termes de précision et de stabilité computationnelle.

Dans le cadre des méthodes level set, le PINN amélioré a démontré des performances supérieures dans la modélisation des interfaces dynamiques, notamment dans des cas présentant des mouvements rigides avec le disque de Zalesak et des déformations significatives avec l'écoulement tourbillonnaire inversé. En particulier, pour l'écoulement tourbillonnaire inversé, le PINN standard s'est montré incapable de suivre l'interface alors que le PINN amélioré et optimisé de *Default** a réussi à atteindre une excellente précision avec une erreur $L^2 = 0.81\%$.

Toutefois, l'intégration de termes de régularisation supplémentaires, tels que la conservation de masse et l'équation d'Eikonal, n'a pas systématiquement amélioré les résultats. L'équation d'Eikonal, bien qu'utile comme terme de régularisation dans certains contextes, a eu un impact négatif sur les solutions présentant des interfaces fortement déformées si son poids n'était pas soigneusement ajusté. De même, les termes de conservation de masse, malgré leur pertinence théorique, ont introduit une surcharge computationnelle sans bénéfices notables en termes de précision. Ces résultats ont d'ailleurs été publiés dans Mullins *et al.* (2025).

En ce qui concerne l'application des PINNs aux équations de Saint-Venant, les résultats montrent que les PINNs améliorés offrent également une amélioration notable par rapport aux PINNs classiques introduits par Raissi et al. Les améliorations telles que la factorisation aléatoire des poids, l'équilibrage des poids de la fonction de perte via la norme des gradients, le respect de la causalité temporelle, les caractéristiques de Fourier aléatoires, la viscosité numérique et l'utilisation de l'architecture PirateNet sont des éléments essentiels pour maximiser la précision du modèle. Les expériences ont démontré que les améliorations recensées permettent de prédire l'évolution de la surface libre sous l'effet de diverses conditions, y compris la présence de bathymétrie variable et d'interférences entre vagues. Ces améliorations ont également permis aux PINNs de prédire avec succès la propagation d'une vague pour le cas de rupture de barrage radiale en 2D. Des erreurs $L_h^2 = 0.30\%$ et $L_u^2 = 6.53\%$ ont d'ailleurs été obtenues pour le cas de propagation d'une vague sur une digue en 1D avec l'architecture PirateNet et des hyperparamètres optimisés. Avec cette même configuration et pour le même problème, la perte de masse est également très faible avec un $MAPE_{mass} = 0.02\%$.

D'autre part, il est important de souligner que la forme conservatrice des équations de Saint-Venant, couramment utilisée dans les méthodes numériques classiques en raison de ses propriétés de conservation de la masse, peut s'avérer problématique lorsqu'appliquée à des PINNs. En effet, lors des premières étapes de l'entraînement, les valeurs faibles ou nulles de la hauteur peuvent entraîner des instabilités numériques ou des divisions par zéro. À cet effet, l'utilisation de la forme non-conservatrice permet de contourner ces difficultés tout en maintenant une bonne conservation de la masse dans les solutions apprises par les PINNs.

Bien que les architectures PirateNet et MMLP obtiennent des erreurs similaires avec des hyperparamètres optimisés, l'architecture PirateNet présente un avantage significatif en termes de robustesse. En effet, elle se montre plus résiliente aux choix des hyperparamètres que

l'architecture MLP modifiée. Cette propriété en fait une option préférable, notamment dans les cas où une optimisation des hyperparamètres n'est pas réalisable ou souhaitable.

Par ailleurs, bien qu'une étude détaillée de chaque amélioration individuelle puisse permettre de mieux comprendre son impact sur la performance, il est recommandé de combiner l'ensemble de ces améliorations pour maximiser les chances d'obtenir de bons résultats dès les premiers essais. Cette approche intégrée s'avère particulièrement efficace pour garantir la stabilité et la précision des prédictions des PINNs sur des problèmes complexes d'écoulements à surface libre.

Cependant, une difficulté persistante réside dans la prédiction des champs de vitesses lorsque de forts gradients sont présents. Si l'évolution générale de la vague est bien capturée, avec une erreur moyenne absolue de la vitesse de l'onde pour le cas en 1D, $MAPE_{c+} = 0.33\%$, les pics et vallées restent difficiles à modéliser avec précision. L'utilisation de viscosité numérique est très importante pour diminuer l'impact de ce problème, mais vient à un coût informatique supplémentaire, surtout pour les problèmes à plus d'une dimension où plus qu'un gradient de second ordre doit être calculé à chaque itération. Un approfondissement des recherches est nécessaire pour mieux appréhender ces discontinuités.

Ces résultats des SWE appliqués aux PINNs ont d'ailleurs été présentés dans le cadre de la conférence DTE & AICOMAS à Paris en février 2025 Mullins & Soulaimani (2025).

Toutefois, un constat majeur de cette étude est que les PINNs restent beaucoup plus coûteux en calcul que les méthodes numériques classiques. Si l'apprentissage profond présente l'avantage d'une phase d'inférence rapide, l'entraînement initial du modèle est extrêmement long comparé aux approches numériques traditionnelles. Comme l'a souligné Georges E. Karniadakis lors de la conférence DTE & AICOMAS 2025, les PINNs ne devraient pas être utilisés pour résoudre des EDP directes, car ils ne rivaliseront jamais avec les méthodes numériques classiques en termes

d'efficacité et de précision. Toutefois, leur véritable potentiel réside dans les applications où des données réelles peuvent être intégrées pour adapter le modèle et améliorer ses performances.

En résumé, cette étude constitue une avancée significative dans l'application des PINNs aux problèmes de capture de l'interface et d'écoulements à surface libre. Bien que des améliorations soient encore nécessaires pour égaler la précision des méthodes numériques classiques, ces résultats montrent que les PINNs offrent une alternative prometteuse, notamment pour des scénarios intégrant des données réelles afin d'affiner les prédictions et d'optimiser les modèles physiques.

ANNEXE I

HYPERPARAMÈTRES

L'annexe suivante contient les hyperparamètres utilisés pour entraîner les modèles présentés dans la section résultats. Elle présente aussi le résultat des optimisations bayésiennes des hyperparamètres.

1. Hyperparamètres selon la configuration

Tableau-A I-1 Burgers 1D : Hyperparamètres pour les configurations étudiées

Category	Parameter	*Plain*	*Default*	*PirateNet*
Architecture	Architecture	MLP	MLP	PirateNet
	Number of layers	4	4	4
	Layer size	256	256	256
	Activation	Tanh	Tanh	Tanh
	Fourier feature scale	-	1.0	1.0
	RWF (μ, σ)	-	$(0.5, 0.1)$	$(0.5, 0.1)$
Training	Optimizer	Adam	Adam	Adam
	Training steps	80,000	80,000	80,000
	Batch size	4,096	4,096	4,096
	Learning rate	0.001	0.001	0.001
	Decay steps	2,000	2,000	2,000
	S2S windows	1	1	1
Weighting	Scheme	Grad norm	Grad norm	Grad norm
	Initial weights $(\lambda_{ic}, \lambda_{bc}, \lambda_r)$	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)
	Causal tolerance	-	1.0	1.0
	Number of chunks	-	32	32

Tableau-A I-2 Burgers 2D : Hyperparamètres pour les configurations étudiées

Category	Parameter	*Plain*	*Default*	*PirateNet*
Architecture	Architecture	MLP	Modified MLP	PirateNet
	Number of layers	4	4	4
	Layer size	256	256	256
	Activation	Gelu	Gelu	Gelu
	Fourier feature scale	1.0	1.0	1.0
	RWF (μ, σ)	(0.5, 0.1)	(0.5, 0.1)	(0.5, 0.1)
Training	Optimizer	Adam	Adam	Adam
	Training steps	70,000	70,000	70,000
	Batch size	4,096	4,096	4,096
	Learning rate	0.001	0.001	0.001
	Decay steps	2,000	2,000	2,000
	S2S windows	1	1	1
Weighting	Scheme	Grad norm	Grad norm	Grad norm
	Initial weights $(\lambda_{ic}, \lambda_{bc}, \lambda_r)$	(10, 1, 1)	(10, 1, 1)	(10, 1, 1)
	Causal tolerance	1.0	1.0	1.0
	Number of chunks	32	32	32

Tableau-A I-3 Level set disque de Zalesak : Hyperparamètres pour les configurations *Plain*, *Default*, *PirateNet* avec et sans optimisation

	Parameter	***Plain***	***Default***	***PirateNet***	***Plain****	***Default****	***PirateNet****
Architecture	Architecture	MLP	Modified MLP	PirateNet	MLP	Modified MLP	PirateNet
	Number of layers	3	3	3	8	8	8
	Layer size	256	256	256	256	256	256
	Activation	Gelu	Gelu	Gelu	Gelu	Relu	Tanh
	Fourier feature scale	-	1.0	1.0	-	2.0	2.0
	RWF (μ, σ)	-	(0.5, 0.1)	(0.5, 0.1)	-	(1.0, 0.1)	(1.0, 0.1)
Training	Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
	Training steps	20,000	20,000	20,000	80,000	80,000	80,000
	Batch size	2048	2048	2048	2048	2048	2048
	Learning rate	0.001	0.001	0.001	0.001	0.001	0.001
	Decay steps	2,000	2,000	2,000	2,000	2,000	2,000
	S2S windows	1	1	1	1	1	1
	Physics-Informed initialization	False	False	True	False	False	True
Weighting	Scheme	-	Grad norm	Grad norm	-	Grad norm	Grad norm
	Initial weights (λ_{ic}, λ_r)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)
	Causal tolerance	-	1.0	1.0	-	2.0	2.0
	Number of chunks	-	32	32	-	32	32

Tableau-A I-4 Level set écoulement tourbillonnaire inversé dans le temps : Hyperparamètres pour les configurations *Plain*, *Default*, *PirateNet* avec et sans optimisation

	Parameter	***Plain***	***Default***	***PirateNet***	***Plain****	***Default****	***PirateNet****
Architecture	Architecture	MLP	Modified MLP	PirateNet	MLP	Modified MLP	PirateNet
	Number of layers	3	3	3	8	8	8
	Layer size	256	256	256	256	256	256
	Activation	Relu	Relu	Relu	Swish	Swish	Swish
	Fourier feature scale	-	1.0	1.0	-	2.0	2.0
	RWF (μ, σ)	-	(0.5, 0.1)	(0.5, 0.1)	-	(1.0, 0.1)	(0.5, 0.1)
Training	Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
	Training steps	20,000	20,000	20,000	20,000	20,000	20,000
	Batch size	2048	2048	2048	2048	2048	2048
	Learning rate	0.001	0.001	0.001	0.001	0.001	0.001
	Decay steps	2,000	2,000	2,000	2,000	2,000	2,000
	S2S windows	2	2	2	2	2	2
	Physics-Informed initialization	False	False	True	False	False	True
Weighting	Scheme	-	Grad norm	Grad norm	-	Grad norm	Grad norm
	Initial weights (λ_{ic}, λ_r)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)
	Causal tolerance	-	1.0	1.0	-	2.0	1.5
	Number of chunks	-	32	32	-	32	32

Tableau-A I-5 SWE 1D écoulement sur une bosse : Hyperparamètres pour les configurations *Plain*, *Default*, *PirateNet* avec et sans optimisation

Category	Parameter	***Plain***	***Default***	***PirateNet***	***Plain****	***Default****	***PirateNet****
Architecture	Architecture	MLP	Modified MLP	PirateNet	MLP	Modified MLP	PirateNet
	Number of layers	8	8	8	8	8	8
	Layer size	256	256	256	256	256	256
	Activation	Gelu	Gelu	Gelu	Tanh	Gelu	Gelu
	Viscosity (ν)	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Fourier feature scale	-	1.0	1.0	-	2.0	2.0
	RWF (μ, σ)	-	$(1.0, 0.1)$	$(1.0, 0.1)$	-	$(0.5, 0.1)$	$(0.5, 0.1)$
Training	Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
	Training steps	120,000	80,000	50,000	120,000	80,000	50,000
	Batch size	2048	2048	2048	2048	2048	2048
	Learning rate	0.001	0.001	0.001	0.001	0.001	0.001
	Decay steps	2,000	2,000	2,000	2,000	2,000	2,000
	S2S windows	1	1	1	1	1	1
Weighting	Scheme	-	Grad norm	Grad norm	-	Grad norm	Grad norm
	Initial weights ($\lambda_{ic}, \lambda_{bc}, \lambda_r$)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)
	Causal tolerance	-	1.0	1.0	-	1.0	2.0
	Number of chunks	-	32	32	-	32	32
Non dim	$U*$	3.5	3.5	3.5	3.5	3.5	3.5
	$L*$	2	2	2		2	2
	$H*$	1	1	1	2	1	1

Tableau-A I-6 SWE 2D rupture de barrage radial : Hyperparamètres pour les configurations *Plain*, *Default*, *PirateNet* avec et sans optimisation

Category	Parameter	*Plain*	*Default*	*PirateNet*	*Plain**	*Default**	*PirateNet**
Architecture	Architecture	MLP	Modified MLP	PirateNet	MLP	Modified MLP	PirateNet
	Number of layers	8	8	8	8	8	8
	Layer size	256	256	256	256	256	256
	Activation	Gelu	Gelu	Gelu	Tanh	Tanh	Tanh
	Viscosity (ν)	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Fourier feature scale	-	1.0	1.0	-	2.0	2.0
	RWF (μ, σ)	-	(1.0, 0.1)	(1.0, 0.1)	-	(0.5, 0.1)	(1.0, 0.1)
Training	Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
	Training steps	120,000	60,000	50,000	120,000	60,000	50,000
	Batch size	2048	2048	2048	2048	2048	4096
	Learning rate	0.001	0.001	0.001	0.001	0.001	0.001
	Decay steps	2,000	2,000	2,000	2,000	2,000	2,000
	S2S windows	1	1	1	1	1	1
Weighting	Scheme	-	Grad norm	Grad norm	-	Grad norm	Grad norm
	Initial weights ($\lambda_{ic}, \lambda_{bc}, \lambda_r$)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)
	Causal tolerance	-	1.0	1.0	-	2.0	2.0
	Number of chunks	-	32	32	-	32	32
Non dim	$U*$	50	50	50	50	50	50
	$L*$	40	40	40	40	40	40
	$H*$	2.5	2.5	2.5	2.5	2.5	2.5

2. Optimisation bayésienne des hyperparamètres

Cette section de l'annexe présente le résultat des optimisation bayésiennes réalisée avec WandB Biewald & Van Pelt (2020). L'objectif de l'optimisation est de diminuer l'erreur de validation L^2. La configuration présentant la plus petite erreur devient donc la configuration *Sota* utilisée dans les tableaux de résultats des différents tests expérimentaux.

Un tableau et une figure sont utilisés pour représenter une optimisation des hyperparamètres. Le tableau présente l'espace des hyperparamètres. Il contient les différentes valeurs discrètes ou continues que peut prendre les hyperparamètres. La figure contient toutes les combinaisons

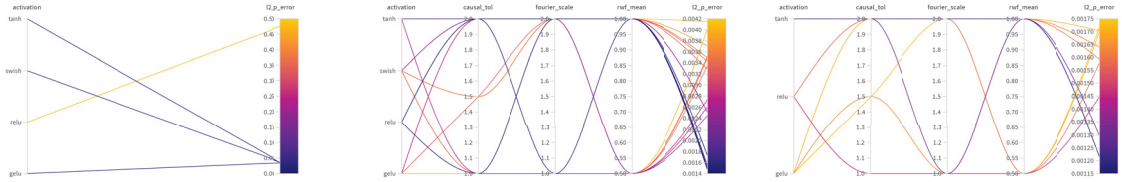
testées ainsi que leur erreur respective. La configuration optimale est représentée par les valeurs en gras du tableau.

2.1 Disque de Zalesak : optimisation des hyperparamètres

Cette section présente la mise en place et le résultat de l'optimisation bayésienne des hyperparamètres pour le disque de Zalesak. À noter que nous avons fait une optimisation des hyperparamètres de type *grid* pour la configuration *Plain* puisque le balayage ne contient que la fonction d'activation.

Tableau-A I-7 Level set Zalesak : Espace d'hyperparamètres

Hyperparameter	**Possible values**
Architecture	MLP, Modified MLP, **PirateNet**
Number of layers	**8**
Activation function	Relu, **Tanh**, Gelu, Swish
Fourier feature scale	1, **2**
RWF mean	0.5, **1**
Causal tolerance	1, 1.5, **2**



a) *Plain* b) *Default* c) *PirateNet*

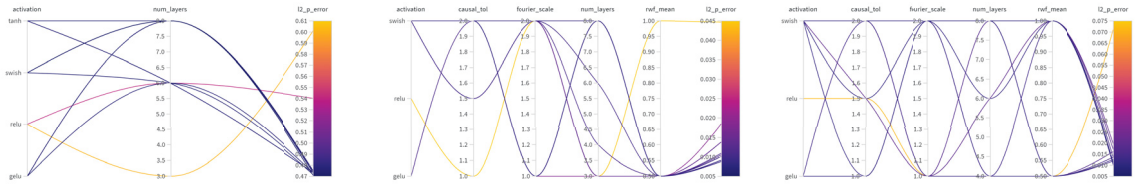
Figure-A I-1 Level set Zalesak : Résultat de l'optimisation bayésienne des hyperparamètres avec WandB

2.2 Écoulement tourbillonnaire inversé dans le temps : optimisation des hyperparamètres

Cette section présente la mise en place et le résultat de l'optimisation bayésienne des hyperparamètres pour l'écoulement tourbillonnaire inversé dans le temps. À noter que nous avons fait une optimisation des hyperparamètres de type *grid* pour la configuration *Plain* puisque le balayage ne contient que la fonction d'activation et le nombre de couches.

Tableau-A I-8 Level set tourbillon : Espace d'hyperparamètres

Hyperparameter	**Possible values**
Architecture	MLP, **MMLP**, PirateNet
Number of layers	4, 6, **8**
Activation function	Relu, Tanh, Gelu, **Swish**
Fourier feature scale	1, **2**
RWF mean	**0.5**, 1
Amount of S2S windows	**2**
Causal tolerance	1, 1.5, **2**



a) *Plain* b) *Default* c) *PirateNet*

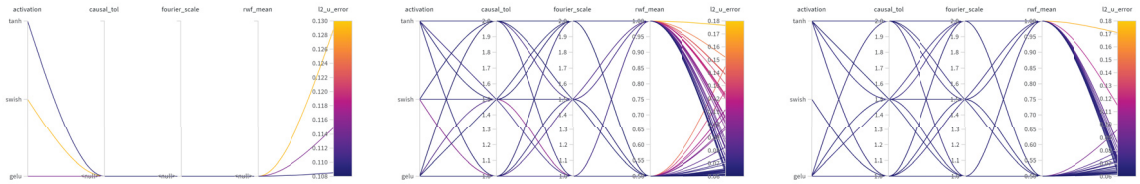
Figure-A I-2 Level set tourbillon : Résultat de l'optimisation bayésienne des hyperparamètres avec WandB

2.3 SWE 1D écoulement sur une bosse : optimisation des hyperparamètres

Cette section présente la mise en place et le résultat de l'optimisation des hyperparamètres pour l'écoulement sur une bosse avec les SWE 1D. À noter que nous avons fait une optimisation des hyperparamètres de type *grid* pour les trois configurations pour mener l'étude de résilience aux hyperparamètres.

Tableau-A I-9 SWE 1D écoulement sur une bosse : Espace d'hyperparamètres

Hyperparameter	**Possible values**
Architecture	MLP, **MMLP**, PirateNet
Number of layers	**8**
Activation function	Tanh, **Gelu**, Swish
Fourier feature scale	1, 1.5, **2**
RWF mean	**0.5**, 1
Causal tolerance	1, 1.5, **2**



a) *Plain* b) *Default* c) *PirateNet*

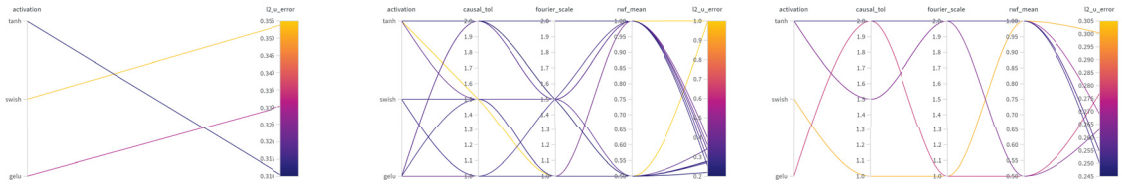
Figure-A I-3 SWE 1D écoulement sur une bosse : Résultat de l'optimisation des hyperparamètres avec WandB

2.4 SWE 2D rupture de barrage radiale : optimisation des hyperparamètres

Cette section présente la mise en place et le résultat de l'optimisation des hyperparamètres pour la rupture de barrage radiale avec les SWE 2D. À noter que nous avons fait une optimisation des hyperparamètres de type *grid* pour *Plain* et une optimisation bayésienne pour *Default* et *PirateNet*.

Tableau-A I-10 SWE 2D rupture de barrage radiale : Espace d'hyperparamètres

Hyperparameter	**Possible values**
Architecture	MLP, **MMLP**, PirateNet
Number of layers	**8**
Activation function	**Tanh**, Gelu, Swish
Fourier feature scale	1, 1.5, **2**
RWF mean	**0.5**, 1
Causal tolerance	1, 1.5, **2**



a) *Plain* b) *Default* c) *PirateNet*

Figure-A I-4 SWE 2D rupture de barrage radiale : Résultat de l'optimisation des hyperparamètres avec WandB

BIBLIOGRAPHIE

Alcrudo, F. & Garcia-Navarro, P. (1993). A high-resolution Godunov-type scheme in finite volumes for the 2D shallow-water equations. *International Journal for Numerical Methods in Fluids*, 16(6), 489–505. doi : 10.1002/fld.1650160604. Publisher : John Wiley & Sons, Ltd.

Anagnostopoulos, S. J., Toscano, J. D., Stergiopulos, N. & Karniadakis, G. E. (2024). Residual-based attention in physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421, 116805. doi : 10.1016/j.cma.2024.116805.

Audusse, E., Bouchut, F., Bristeau, M.-O., Klein, R. & Perthame, B. (2004). A Fast and Stable Well-Balanced Scheme with Hydrostatic Reconstruction for Shallow Water Flows. *SIAM Journal on Scientific Computing*, 25(6), 2050–2065. doi : 10.1137/S1064827503431090. Publisher : Society for Industrial and Applied Mathematics.

Aulich, M., Goinis, G. & Voß, C. (2024). Data-Driven AI Model for Turbomachinery Compressor Aerodynamics Enabling Rapid Approximation of 3D Flow Solutions. *Aerospace*, 11(9), 723. doi : 10.3390/aerospace11090723. Number : 9 Publisher : Multidisciplinary Digital Publishing Institute.

Ausas, R. F., Dari, E. A. & Buscaglia, G. C. (2011). A geometric mass-preserving redistancing scheme for the level set function | Request PDF. *International Journal for Numerical Methods in Fluids*, 65(8), 989–1010. doi : 10.1002/fld.2227.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A. & Siskind, J. M. (2017). Automatic differentiation in machine learning : a survey. *J. Mach. Learn. Res.*, 18(1), 5595–5637.

Bengio, Y., Louradour, J., Collobert, R. & Weston, J. (2009). Curriculum learning. *Proceedings of the 26th Annual International Conference on Machine Learning*, (ICML '09), 41–48. doi : 10.1145/1553374.1553380.

Bhatt, P., Kumar, Y. & Soulaimani, A. (2023). Deep convolutional architectures for extrapolative forecasts in time-dependent flow problems. *Advanced Modeling and Simulation in Engineering Sciences*, 10(1), 17. doi : 10.1186/s40323-023-00254-y.

Biewald, L. & Van Pelt, C. (2020). Weights & Biases : The AI Developer Platform. Repéré le 2024-11-19 à https://wandb.ai/site/.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., MacLaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S. & Zhang, Q. (2018). JAX : composable transformations of Python+NumPy programs. jax-ml. Repéré le 2024-12-12 à http://github.com/jax-ml/jax.

Bradford, S. F. & Sanders, B. F. (2002). Finite-Volume Model for Shallow-Water Flooding of Arbitrary Topography. *Journal of Hydraulic Engineering*, 128(3), 289–298. doi : 10.1061/(ASCE)0733-9429(2002)128:3(289). Publisher : American Society of Civil Engineers.

Brufau, P., García-Navarro, P. & Vázquez-Cendón, M. E. (2004). Zero mass error using unsteady wetting–drying conditions in shallow flows over dry irregular topography. *International Journal for Numerical Methods in Fluids*, 45(10), 1047–1082. doi : 10.1002/fld.729. _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.729.

Burgers, J. M. (1948). A Mathematical Model Illustrating the Theory of Turbulence. Dans Von Mises, R. & Von Kármán, T. (Éds.), *Advances in Applied Mechanics* (vol. 1, pp. 171–199). Elsevier. doi : 10.1016/S0065-2156(08)70100-5.

Chen, Z., Badrinarayanan, V., Lee, C.-Y. & Rabinovich, A. [arXiv :1711.02257]. (2018). GradNorm : Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. arXiv. Repéré le 2024-11-07 à http://arxiv.org/abs/1711.02257.

Cho, G., Zhu, D., Campbell, J. J. & Wang, M. (2022). An LSTM-PINN Hybrid Method to Estimate Lithium-Ion Battery Pack Temperature. *IEEE Access*, 10, 100594–100604. doi : 10.1109/ACCESS.2022.3208103. Conference Name : IEEE Access.

CMM. (2019). Cartes des zones inondables. Repéré le 2025-01-14 à https://observatoire.cmm.qc.ca/produits/cartes-interactives/.

CMM. (2025). Crues Grand Montréal | Prévisions et risques d'inondation dans le Grand Montréal. Repéré le 2025-01-14 à https://www.cruesgrandmontreal.ca/.

Crandall, M. G. & Lions, P.-L. (1983). Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 277(1), 1–42. doi : 10.1090/S0002-9947-1983-0690039-8.

CRED & UNDRR. (2015). *The human cost of weather-related disasters 1995-2015*. Report, Paris. Repéré le 2025-01-14 à https://www.undrr.org/publication/human-cost-weather-related-disasters-1995-2015.

Daw, A., Bu, J., Wang, S., Perdikaris, P. & Karpatne, A. [arXiv :2207.02338]. (2023). Mitigating Propagation Failures in Physics-informed Neural Networks using Retain-Resample-Release (R3) Sampling. arXiv. Repéré le 2024-11-07 à http://arxiv.org/abs/2207.02338.

Dazzi, S. (2024). Physics-Informed Neural Networks for the Augmented System of Shallow Water Equations With Topography. *Water Resources Research*, 60(10), e2023WR036589. doi : 10.1029/2023WR036589. _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1029/2023WR036589.

Delmas, V. (2020). *Implémentation multi-GPU d'un code en volumes finis pour le calcul de haute performance des écoulements à surface libre*. (Mémoire de maîtrise électronique, École de technologie supérieure, Montréal). Repéré à https://espace.etsmtl.ca/id/eprint/2581/.

Delmas, V. & Soulaïmani, A. (2022). Multi-GPU implementation of a time-explicit finite volume solver using CUDA and a CUDA-Aware version of OpenMPI with application to shallow water flows. *Computer Physics Communications*, 271, 108190. doi : 10.1016/j.cpc.2021.108190.

Donea, J., Giuliani, S. & Halleux, J. P. (1982). An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33(1), 689–723. doi : 10.1016/0045-7825(82)90128-1.

Donnelly, J., Daneshkhah, A. & Abolfathi, S. (2024). Physics-informed neural networks as surrogate models of hydrodynamic simulators. *Science of The Total Environment*, 912, 168814. doi : 10.1016/j.scitotenv.2023.168814.

Driscoll, T. A., Hale, N. & Trefethen, L. N. (2014). *Chebfun guide* (éd. Pafnuty Publications). Oxford.

Fahsi, A. & Soulaïmani, A. (2017). Numerical investigations of the XFEM for solving two-phase incompressible flows. *International Journal of Computational Fluid Dynamics*, 31(3), 135–155. doi : 10.1080/10618562.2017.1322200. Publisher : IAHR Website _eprint : https ://doi.org/10.1080/10618562.2017.1322200.

Fang, Z. & Zhan, J. (2020). Deep Physical Informed Neural Networks for Metamaterial Design. *IEEE Access*, 8, 24506–24513. doi : 10.1109/ACCESS.2019.2963375. Conference Name : IEEE Access.

Fuks, O. & Tchelepi, H. A. (2020). Limitations of Physics Informed Machine Learning for Nonlinear Two-Phase Transport in Porous Media. *Journal of Machine Learning for Modeling and Computing*, 1(1), 19–37. doi : 10.1615/JMachLearnModelComput.2020033905. Publisher : Begel House Inc.

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256. Repéré à https://proceedings.mlr.press/v9/glorot10a.html.

Gomes, J. & Faugeras, O. (2000). Reconciling Distance Functions and Level Sets. *Journal of Visual Communication and Image Representation*, 11(2), 209–223. doi : 10.1006/jvci.1999.0439.

Guo, Z., Leitão, J. P., Simões, N. E. & Moosavi, V. (2021). Data-driven flood emulation : Speeding up urban flood predictions by deep convolutional neural networks - Astrophysics Data System. *Journal of Flood Risk Management*, 14(1), –. doi : 10.1111/jfr3.12684.

Harlow, F. H. & Welch, J. E. (1965). Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *The Physics of Fluids*, 8(12), 2182–2189. doi : 10.1063/1.1761178.

Houtekamer, P. L. & Mitchell, H. L. (1998). Data Assimilation Using an Ensemble Kalman Filter Technique. 796–811. Repéré à https://journals.ametsoc.org/view/journals/mwre/126/3/1520-0493_1998_126_0796_dauaek_2.0.co_2.xml. Section : Monthly Weather Review.

Hunter, J. D. (2007). Matplotlib : A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3), 90–95. doi : 10.1109/MCSE.2007.55. Conference Name : Computing in Science & Engineering.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1), 35–45. doi : 10.1115/1.3662552.

Ketcheson, D. I., Mandli, K. T., Ahmadia, A. J., Alghamdi, A., Quezada de Luna, M., Parsani, M., Knepley, M. G. & Emmett, M. (2012). PyClaw : Accessible, Extensible, Scalable Tools for Wave Propagation Problems. *SIAM Journal on Scientific Computing*, 34(4), C210–C231.

Krishnapriyan, A. S., Gholami, A., Zhe, S., Kirby, R. M. & Mahoney, M. W. [arXiv :2109.01050 [physics]]. (2021). Characterizing possible failure modes in physics-informed neural networks. arXiv. Repéré le 2024-08-08 à http://arxiv.org/abs/2109.01050.

Lax, P. (2006). *Hyperbolic Partial Differential Equations*. Providence, RI : American Mathematical Soc. doi : 10.1090/cln/014.

Leiteritz, R., Hurler, M. & Pflüger, D. [arXiv :2111.09705]. (2021). Learning Free-Surface Flow with Physics-Informed Neural Networks. arXiv. Repéré le 2024-11-11 à http://arxiv.org/abs/2111.09705.

Li, C., Han, Z., Li, Y., Li, M., Wang, W., Dou, J., Xu, L. & Chen, G. (2023). Physical information-fused deep learning model ensembled with a subregion-specific sampling method for predicting flood dynamics. *Journal of Hydrology*, 620, 129465. doi : 10.1016/j.jhydrol.2023.129465.

Li, C., Xu, C., Gui, C. & Fox, M. (2005). Level set evolution without re-initialization : a new variational formulation. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1, 430–436 vol. 1. doi : 10.1109/CVPR.2005.213.

Liu, K., Luo, K., Cheng, Y., Liu, A., Li, H., Fan, J. & Balachandar, S. (2023). Surrogate modeling of parameterized multi-dimensional premixed combustion with physics-informed neural networks for rapid exploration of design space. *Combustion and Flame*, 258, 113094. doi : 10.1016/j.combustflame.2023.113094.

Loukili, Y. & Soulaimani, A. (2007). Numerical tracking of shallow water waves by the unstructured finite volume WAF approximation. *International Journal of Computational Methods in Engineering Science and Mechanics*, 8(2), 75–88. doi : 10.1080/15502280601149577. Number : 2.

M. Silva, R., Grave, M. & Coutinho, A. L. G. A. (2024). A PINN-based level-set formulation for reconstruction of bubble dynamics. *Archive of Applied Mechanics*, 94(9), 2667–2682. doi : 10.1007/s00419-024-02622-5.

Marshall, J., Adcroft, A., Hill, C., Perelman, L. & Heisey, C. (1997). A finite-volume, incompressible Navier Stokes model for studies of the ocean on parallel computers. *Journal of Geophysical Research : Oceans*, 102(C3), 5753–5766. doi : 10.1029/96JC02775. _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1029/96JC02775.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu & Xiaoqiang Zheng. (2015). TensorFlow : Large-Scale Machine Learning on Heterogeneous Systems. Repéré à https://www.tensorflow.org/.

Masclans, N., Vázquez-Novoa, F., Bernades, M., Badia, R. M. & Jofre, L. (2023). Thermodynamics-informed neural network for recovering supercritical fluid thermophysical information from turbulent velocity data. *International Journal of Thermofluids*, 20, 100448. doi : 10.1016/j.ijft.2023.100448.

Mathias, M. S., de Almeida, W. P., de Barros, M. R., Coelho, J. F., de Freitas, L. P., Moreno, F. M., Netto, C. F. D., Cozman, F. G., Costa, A. H. R., Tannuri, E. A., Gomi, E. S. & Dottori, M. [arXiv :2301.07824 [physics]]. (2022). Augmenting a Physics-Informed Neural Network for the 2D Burgers Equation by Addition of Solution Data Points. Repéré le 2024-09-24 à http://arxiv.org/abs/2301.07824.

Mattey, R. & Ghosh, S. (2022). A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390, 114474. doi : 10.1016/j.cma.2021.114474.

Mullins, M. & Soulaimani, A. (2025). Physics-Informed Neural Networks for Solving the Shallow-Water Equations. Paris, France. Repéré à https://dte_aicomas_2025.iacm.info/event/contribution/77aaf391-8735-11ef-b344-000c29ddfc0c.

Mullins, M., Kamil, H., Fahsi, A. & Soulaimani, A. [arXiv :2502.02440 [physics]]. (2025). Physics-informed neural networks for solving moving interface flow problems using the level set approach. arXiv. Repéré le 2025-03-04 à http://arxiv.org/abs/2502.02440.

Noh, W. F. & Woodward, P. (1976). SLIC (Simple Line Interface Calculation). *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28 – July 2, 1976 Twente University, Enschede*, pp. 330–340. doi : 10.1007/3-540-08004-X_336.

Olsson, E. & Kreiss, G. (2005). A conservative level set method for two phase flow. *Journal of Computational Physics*, 210(1), 225–246. doi : 10.1016/j.jcp.2005.04.007.

Olsson, E., Kreiss, G. & Zahedi, S. (2007). A conservative level set method for two phase flow II. *Journal of Computational Physics*, 225(1), 785–807. doi : 10.1016/j.jcp.2006.12.027.

Osher, S. & Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed : Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1), 12–49. doi : 10.1016/0021-9991(88)90002-2.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. [arXiv :1912.01703 [cs]]. (2019). PyTorch : An Imperative Style, High-Performance Deep Learning Library. arXiv. Repéré le 2025-01-17 à http://arxiv.org/abs/1912.01703.

Penwarden, M., Jagtap, A. D., Zhe, S., Karniadakis, G. E. & Kirby, R. M. (2023). A unified scalable framework for causal sweeping strategies for Physics-Informed Neural Networks (PINNs) and their temporal decompositions. *Journal of Computational Physics*, 493, 112464. doi : 10.1016/j.jcp.2023.112464. arXiv :2302.14227 [physics].

Pombo, D. V., Bindner, H. W., Spataru, S. V., Sørensen, P. E. & Bacher, P. (2022). Increasing the Accuracy of Hourly Multi-Output Solar Power Forecast with Physics-Informed Machine Learning. *Sensors*, 22(3), 749. doi : 10.3390/s22030749. Number : 3 Publisher : Multidisciplinary Digital Publishing Institute.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y. & Courville, A. (2019). On the Spectral Bias of Neural Networks. *Proceedings of the 36th International Conference on Machine Learning*, pp. 5301–5310. Repéré à https://proceedings.mlr.press/v97/rahaman19a.html.

Raissi, M., Perdikaris, P. & Karniadakis, G. E. (2019). Physics-informed neural networks : A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 686–707. doi : 10.1016/j.jcp.2018.10.045.

Rider, W. J. & Kothe, D. B. (1998). Reconstructing Volume Tracking. *Journal of Computational Physics*, 141(2), 112–152. doi : 10.1006/jcph.1998.5906.

Saint-Venant, A. J. C. (1871). Théorie du mouvement non permanent des eaux, avec application aux crues des rivières et a l'introduction de marées dans leurs lits. *Comptes Rendus des Séances de Académie des Science*, 73, 147–237.

Salvati, E., Tognan, A., Laurenti, L., Pelegatti, M. & De Bona, F. (2022). A defect-based physics-informed machine learning framework for fatigue finite life prediction in additive manufacturing. *Materials & Design*, 222, 111089. doi : 10.1016/j.matdes.2022.111089.

Sethian, J. A. (1996a). A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4), 1591–1595. doi : 10.1073/pnas.93.4.1591. Publisher : Proceedings of the National Academy of Sciences.

Sethian, J. (1996b). Theory, algorithms, and applications of level set methods for propagating interfaces. *Acta Numerica*, 5, 309–395. doi : 10.1017/S0962492900002671.

Silva, V. L. S., Heaney, C. E., Li, Y. & Pain, C. C. (2022). Data Assimilation Predictive GAN (DA-PredGAN) Applied to a Spatio-Temporal Compartmental Model in Epidemiology. *Journal of Scientific Computing*, 94(1), 25. doi : 10.1007/s10915-022-02078-1.

Sussman, M., Smereka, P. & Osher, S. (1994). A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *Journal of Computational Physics*, 114(1), 146–159. doi : 10.1006/jcph.1994.1155.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. & Ng, R. (2020). Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *Advances in Neural Information Processing Systems*, 33, 7537–7547. Repéré à https://proceedings.neurips.cc/paper/2020/hash/55053683268957697aa39fba6f231c68-Abstract.html.

Tang, M., Xin, Z. & Wang, L. (2024). Physics-Informed neural network for level set method in vapor condensation. *International Journal of Heat and Fluid Flow*, 110, 109651. doi : 10.1016/j.ijheatfluidflow.2024.109651.

Tian, Y., Ding, S., Su, G., Huang, L. & Chen, J. [arXiv :2501.11372 [physics]]. (2025). Physics-Informed Neural Networks for Solving the Two-Dimensional Shallow Water Equations with Terrain Topography and Rainfall Source Terms. arXiv. Repéré le 2025-02-12 à http://arxiv.org/abs/2501.11372.

Touré, M. K. & Soulaïmani, A. (2016). Stabilized finite element methods for solving the level set equation without reinitialization. *Computers & Mathematics with Applications*, 71(8), 1602–1623. doi : 10.1016/j.camwa.2016.02.028.

Unverdi, S. O. & Tryggvason, G. (1992). A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 100(1), 25–37. doi : 10.1016/0021-9991(92)90307-K.

van der Pijl, S. P., Segal, A., Vuik, C. & Wesseling, P. (2005). A mass-conserving Level-Set method for modelling of multi-phase flows. *International Journal for Numerical Methods in Fluids*, 47(4), 339–361. doi : 10.1002/fld.817. _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.817.

Wang, S., Teng, Y. & Perdikaris, P. (2021a). Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. *SIAM J. Sci. Comput.*, 43(5), A3055–A3081. doi : 10.1137/20M1318043.

Wang, S., Wang, H. & Perdikaris, P. (2021b). On the eigenvector bias of Fourier feature networks : From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384, 113938. doi : 10.1016/j.cma.2021.113938.

Wang, S., Wang, H. & Perdikaris, P. [arXiv :2110.01654 [physics, stat]]. (2021c). Improved architectures and training algorithms for deep operator networks. arXiv. Repéré le 2024-05-29 à http://arxiv.org/abs/2110.01654.

Wang, S., Sankaran, S. & Perdikaris, P. (2022a). Respecting causality is all you need for training physics-informed neural networks. Repéré le 2024-09-11 à https://arxiv.org/abs/2203.07404v1.

Wang, S., Wang, H., Seidman, J. H. & Perdikaris, P. [arXiv :2210.01274 [cs]]. (2022b). Random Weight Factorization Improves the Training of Continuous Neural Representations. arXiv. Repéré le 2024-08-08 à http://arxiv.org/abs/2210.01274.

Wang, S., Yu, X. & Perdikaris, P. (2022c). When and why PINNs fail to train : A neural tangent kernel perspective. *Journal of Computational Physics*, 449, 110768. doi : 10.1016/j.jcp.2021.110768.

Wang, S., Sankaran, S., Wang, H. & Perdikaris, P. [arXiv :2308.08468 [physics]]. (2023). An Expert's Guide to Training Physics-informed Neural Networks. arXiv. Repéré le 2024-02-07 à http://arxiv.org/abs/2308.08468.

Wang, S., Li, B., Chen, Y. & Perdikaris, P. (2025). PirateNets : Physics-informed Deep Learning with Residual Adaptive Networks. *Journal of Machine Learning Research*, 25, 1–51. Repéré à http://arxiv.org/abs/2402.00326.

Wu, C., Zhu, M., Tan, Q., Kartha, Y. & Lu, L. (2023). A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403, 115671. doi : 10.1016/j.cma.2022.115671.

Yin, Z., Shi, J., Bian, L., Campbell, W., Zanje, S., Hu, B. & Leon, A. (2025). Physics-Informed Neural Network Approach for Solving the One-Dimensional Unsteady Shallow-Water Equations in Riverine Systems. *Journal of Hydraulic Engineering*, 151(1). doi : 10.1061/JHEND8.HYENG-13572.

Zalesak, S. T. (1979). Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, 31(3), 335–362. doi : 10.1016/0021-9991(79)90051-2.

Zhan, C., Zhang, T., Zhang, S. & Yang, D. (2025). Solving complex flood wave propagation using split Coefficient-based Physical Informed Neural Network. *Journal of Hydrology*, 132835. doi : 10.1016/j.jhydrol.2025.132835.

Zhou, W., Miwa, S. & Okamoto, K. (2024). Self-adaptive and time divide-and-conquer physics-informed neural networks for two-phase flow simulations using interface tracking methods. *Physics of Fluids*, 36(7), 073305. doi : 10.1063/5.0214646.

Ziliani, M. G., Ghostine, R., Ait-El-Fquih, B., McCabe, M. F. & Hoteit, I. (2019). Enhanced flood forecasting through ensemble data assimilation and joint state-parameter estimation. *Journal of Hydrology*, 577, 123924. doi : 10.1016/j.jhydrol.2019.123924.

Zokagoa, J.-M. & Soulaïmani, A. (2010). Modeling of wetting–drying transitions in free surface flows over complex topographies. *Computer Methods in Applied Mechanics and Engineering*, 199(33), 2281–2304. doi : 10.1016/j.cma.2010.03.023.