# UNIVERSITE DE STRASBOURG

## ECOLE DOCTORALE MATHEMATIQUES SCIENCES DE L'INFORMATION ET DE L'INGENIEUR

# RESUME DE LA THESE DE DOCTORAT

Discipline : Mathématiques

Spécialité (facultative) : /

Présentée par : Frédérique Lecourtier

**Titre : Méthodes hybrides éléments finis et réseaux de neurones pour les EDP paramétriques**

Unité de Recherche : Laboratoire des sciences de l'Ingénieur, de l'Informatique et de l'Imagerie, Icube, UMR 7357

Directeur de Thèse : Emmanuel Franck - Directeur de recherche

Co-Directeur de Thèse (s'il y a lieu) : /

Localisation : Strasbourg, France

Thèse confidentielle : ■ NON ☐ OUI

# 1 Contexte et motivations

Les équations aux dérivées partielles (EDP) modélisent de nombreux problèmes de la science et de l'ingénierie, comme l'écoulement d'un fluide, la déformation d'une structure élastique ou la diffusion de la chaleur. Dans la grande majorité des cas d'intérêt pratique, elles n'admettent pas de solution analytique, et il faut recourir à des méthodes numériques pour en construire une approximation. Résoudre ces problèmes avec précision et en un temps de calcul raisonnable est un enjeu central, en particulier lorsque les solutions doivent être calculées rapidement et de façon répétée. C'est le cas de la simulation en temps réel et des jumeaux numériques, des applications interactives telles que la simulation chirurgicale, de l'optimisation de forme, des problèmes inverses ou de la quantification d'incertitudes, qui nécessitent tous un grand nombre de résolutions.

**Méthodes haute fidélité et leurs limites.** Plusieurs grandes familles de méthodes dites de haute fidélité ont été développées pour discrétiser ces EDP. Les différences finies [34] approchent directement les opérateurs différentiels sur une grille structurée, au prix d'une rigidité géométrique mal adaptée aux domaines complexes. La méthode des éléments finis (EF) [14, 22] repose sur une formulation variationnelle et un maillage non structuré. Elle offre une grande flexibilité géométrique et des fondements mathématiques solides, avec des analyses d'erreur a priori, ce qui en fait aujourd'hui la méthode dominante en ingénierie. Les volumes finis [23] garantissent par construction la conservation locale des grandeurs physiques, et les méthodes spectrales [17] sont remarquablement efficaces pour les solutions régulières sur des géométries simples.

Ces méthodes présentent néanmoins trois limites qui motivent des développements plus récents.

— *La géométrie.* Construire un maillage conforme au bord d'un domaine complexe peut être coûteux, voire difficile. Cela a conduit aux méthodes dites non ajustées, qui plongent le domaine dans un maillage de fond simple, comme les méthodes de frontières immergées [42], CutFEM [16] ou la méthode $\varphi$-FEM [20], qui représente le domaine par une fonction level-set. L'analyse isogéométrique [28] répond autrement à la difficulté en utilisant la même représentation spline pour la géométrie et pour l'approximation.

— *Le coût paramétrique.* Lorsque le problème dépend de paramètres physiques et doit être résolu pour de nombreuses valeurs de ceux-ci, répéter une résolution complète devient rapidement prohibitif. Les modèles d'ordre réduit construisent hors ligne un espace d'approximation de faible dimension à partir de simulations haute fidélité, puis résolvent en ligne, à moindre coût, le problème projeté. On y trouve la décomposition orthogonale aux valeurs propres [11], les bases réduites [41] et la décomposition propre généralisée [18]. Cette approche est efficace lorsque la variété des solutions, c'est-à-dire l'ensemble des solutions obtenues quand le paramètre varie, est bien approchée par un sous-espace linéaire de petite dimension, ce qui n'est pas le cas par exemple pour les problèmes dominés par le transport.

— *L'expressivité.* Ces méthodes reposent sur des espaces de dimension finie fixés a priori, formés de polynômes par morceaux ou de modes globaux. Capter des

structures complexes de la solution, comme des couches limites ou des singularités, peut exiger un maillage très fin. Une première réponse, issue des méthodes de partition de l'unité comme la méthode des éléments finis généralisés [5, 47], enrichit localement l'espace avec des fonctions spéciales, mais leur choix dépend fortement du problème et se prête mal au cadre paramétrique. Cet enrichissement se généralise toutefois en remplaçant ces fonctions par un a priori obtenu autrement. La méthode $\varphi$-FEM en est un exemple, en enrichissant multiplicativement l'espace avec un a priori géométrique, la fonction level-set, une structure que nous reprenons plus loin avec un a priori neuronal.

**Méthodes neuronales.** Face à cette limite d'expressivité, une seconde réponse, plus radicale, abandonne l'espace vectoriel de dimension finie au profit d'un espace d'approximation d'une autre nature. C'est ce qu'a proposé le machine learning scientifique, qui remplace ou complète ces espaces classiques par des réseaux de neurones, pour deux raisons. D'une part, les réseaux de neurones sont des approximateurs universels [19] capables de représenter des fonctions complexes sans discrétisation par un maillage. La solution n'est plus projetée sur un sous-espace vectoriel mais sur une sous-variété de dimension finie de l'espace de fonctions, paramétrée par les poids du réseau. D'autre part, dans le contexte paramétrique, les paramètres peuvent simplement être ajoutés en entrée du réseau, et un unique entraînement hors ligne réduit alors le coût sur toute la famille de paramètres, dans le même esprit hors ligne/en ligne que les modèles d'ordre réduit, mais avec une approximation non linéaire.

Deux paradigmes se distinguent selon la manière d'entraîner le réseau. Le premier, dans lequel le réseau constitue lui-même l'espace d'approximation, minimise une fonction de coût construite à partir de l'EDP, sans données : résidu de la forme forte pour les réseaux informés par la physique (PINN) [43], énergie pour la méthode Deep Ritz [21], ou résidu de la forme faible pour les PINN variationnels [31]. Le second, celui des opérateurs neuronaux, repose sur un apprentissage supervisé et consiste à apprendre un opérateur entre espaces de fonctions, comme DeepONet [37] ou les FNO [35], dont les variantes informées par la physique ajoutent un terme de résidu à la fonction de coût [36].

Ces méthodes ont cependant des limites bien identifiées, comme l'absence de garanties théoriques et d'estimateurs d'erreur fiables, des difficultés d'entraînement et une forte sensibilité aux hyperparamètres, des performances dégradées pour les solutions non régulières, ou encore la difficulté à imposer exactement les conditions aux limites. Ces limites font que les méthodes neuronales ne sont pas encore compétitives avec les méthodes classiques pour une résolution isolée [26], mais elles peuvent devenir avantageuses dès qu'une multitude de résolutions est requise, typiquement dans le cadre paramétrique.

**Hybridation.** Pour combiner les atouts de ces deux grandes familles, plusieurs stratégies d'hybridation ont été proposées. Une première enrichit l'espace d'approximation d'une méthode classique par un a priori neuronal, comme pour une base d'éléments finis généralisés [50] ou une base de Galerkin discontinu [25]. Une deuxième structure

la représentation du réseau avec des éléments finis, en interpolant ses prédictions sur un maillage [6, 51] ou en intégrant directement les fonctions de forme dans son architecture [45]. D'autres formes existent, comme un couplage multi-échelle dans lequel un réseau corrige localement une résolution grossière classique [38], une décomposition du problème selon ses variables [24], ou l'usage d'une prédiction pour accélérer un solveur itératif tel que la méthode de Newton [40].

Enfin, la programmation différentiable a été appliquée aux schémas numériques eux-mêmes, écrits dans un cadre de différentiation automatique pour propager les gradients à travers la résolution [52]. Un réseau peut alors être entraîné à travers le solveur pour corriger le schéma, soit par un terme source neuronal dans un solveur Galerkin discontinu différentiable [30], soit par des corrections de la forme faible dans un schéma éléments finis [29].

C'est dans cette direction hybride que s'inscrit la thèse : plutôt que de remplacer les méthodes classiques sur maillage, nous nous appuyons sur leur structure, soit en enrichissant un espace éléments finis fixé par un a priori neuronal, soit en rendant le schéma numérique lui-même différentiable pour pouvoir l'entraîner directement.

# 2 Problématique et contributions

La thèse porte sur la résolution d'EDP paramétriques, en particulier elliptiques, qui doivent être résolues un grand nombre de fois pour différentes valeurs de paramètres physiques. L'objectif est de concevoir des méthodes hybrides, capables de résoudre efficacement ce problème en combinant la structure et les garanties des méthodes de Galerkin classiques avec l'expressivité et la généralisation paramétrique des réseaux de neurones. Cette hybridation est développée autour de deux contributions, qui reposent sur deux principes de couplage différents.

**Première contribution.** Elle consiste à enrichir l'espace d'approximation éléments finis de Lagrange continus avec un a priori neuronal fixé. Un réseau informé par la physique (PINN) est entraîné une seule fois, hors ligne, pour toute la famille de paramètres. Pour chaque nouvelle valeur du paramètre, sa prédiction est ensuite injectée en ligne dans l'espace éléments finis, qui est résolu de manière classique, soit additivement (la solution s'écrit comme la somme de l'a priori et d'une correction éléments finis), soit multiplicativement (comme leur produit). Nous démontrons, dans les deux cas, des estimations d'erreur qui montrent que les ordres de convergence classiques sont conservés, avec un gain qui dépend explicitement de la qualité de l'a priori et en particulier de celle de ses dérivées d'ordre élevé. Les expériences en 1D, 2D et 3D montrent qu'à précision donnée, un maillage plus grossier suffit, ce qui réduit le coût de calcul, en particulier dans le cadre paramétrique. Ce travail a été publié dans la revue *ESAIM : Mathematical Modelling and Numerical Analysis* [8].

**Seconde contribution.** Elle repose sur un principe différent, puisqu'au lieu d'enrichir un espace éléments finis avec un a priori neuronal fixé, le solveur lui-même (Galerkin discontinu, éléments finis ou $\varphi$-FEM) est rendu différentiable, donc apprenable. Elle comprend d'abord la bibliothèque `ScimBa`, développée par l'équipe-projet Inria MACARON, dans sa version en `JAX`, dont j'ai implémenté les modules d'un schéma de Galerkin différentiable et le formalisme permettant d'entraîner un réseau à travers le solveur, par différentiation implicite et méthode adjointe. Elle utilise ensuite cette bibliothèque pour revenir sur le manque d'expressivité des espaces classiques, en enrichissant les bases de Galerkin par un réseau de neurones, d'abord pour une valeur fixée du paramètre, puis pour toute une famille de paramètres.

Cette contribution diffère de la précédente de deux façons. Premièrement, le réseau n'est plus un a priori pré-entraîné puis figé, mais appris à travers le solveur. Deuxièmement, il ne s'agit plus d'un espace enrichi dont l'erreur est contrôlée a priori, mais d'une base apprise pour laquelle nous ne disposons d'aucune estimation d'erreur. Les résultats sont donc expérimentaux, obtenus en 1D et 3D, et constituent une étude de faisabilité de l'enrichissement appris plutôt qu'un solveur compétitif.

**Collaborations et contribution personnelle.** La première contribution est le fruit d'une collaboration entre trois équipes-projets Inria (MIMESIS, MAKUTU et MACARON) et l'IMAG à Montpellier, et j'en suis la principale contributrice. Pour la seconde, j'ai implémenté, en étroite collaboration avec Emmanuel Franck, Victor Michel-Dansac et Rémi Imbach, l'ensemble des modules spécifiques au schéma Galerkin discontinu, ainsi que ses extensions aux éléments finis et à $\varphi$-FEM, et le formalisme permettant d'entraîner un réseau à travers le solveur, par différentiation implicite et méthode adjointe. Cet ensemble s'appuie sur des briques déjà présentes dans `ScimBa` et développées par d'autres membres de l'équipe, qui ne font pas partie de ma contribution, avec en particulier les outils d'optimisation (optimiseurs et architectures de réseaux). À la date de rédaction de ce résumé, un article présentant ces développements est en préparation.

**Organisation du manuscrit.** Le manuscrit comporte six chapitres organisés de la manière suivante :

1. L'introduction.
2. Le problème paramétrique considéré et les méthodes classiques sur lesquelles reposent les contributions (éléments finis continus, Galerkin discontinu, $\varphi$-FEM), ainsi que les PINN.
3. La première contribution avec l'enrichissement d'un espace éléments finis par un a priori neuronal fixé.
4. La bibliothèque `ScimBa` et le formalisme apprenable, qui constituent le fondement de la seconde contribution.
5. Les méthodes de Galerkin à bases enrichies, qui en constituent l'application.
6. La conclusion avec les limites et perspectives de ces travaux.

Ce résumé suit cet ordre : les sections 3 à 6 correspondent aux chapitres 2 à 5 du manuscrit, et la section 7 à la conclusion. La section 8 présente les publications et communications associées à ces travaux.

# 3 Cadre et outils

*(voir manuscrit, chapitre 2)*

**Problème considéré.** Dans toute la thèse, nous considérons une EDP elliptique linéaire paramétrique sur un domaine $\Omega \subset \mathbb{R}^d$ de frontière lipschitzienne $\Gamma$. Pour un paramètre $\boldsymbol{\mu} \in \mathcal{M} \subset \mathbb{R}^p$, on cherche $u(\cdot, \boldsymbol{\mu})$ tel que

$$R\,u + \boldsymbol{C} \cdot \nabla u - \frac{1}{\mathrm{Pe}} \nabla \cdot (D \nabla u) = f \quad \text{dans } \Omega,$$

où $R$ est le coefficient de réaction, $\boldsymbol{C}$ le champ de convection, $D$ la matrice de diffusion symétrique définie positive, Pe le nombre de Péclet et $f$ le terme source. Ces termes sont tous des fonctions de $\boldsymbol{x} \in \Omega$ et peuvent dépendre de $\boldsymbol{\mu}$, de même que les conditions aux limites de Dirichlet, Neumann ou Robin.

**Éléments finis continus.** Pour une valeur fixée de $\boldsymbol{\mu}$ et des conditions de Dirichlet homogènes, la formulation variationnelle s'écrit, avec $a$ la forme bilinéaire et $l$ la forme linéaire associées :

$$\text{Trouver } u \in H^1_0(\Omega) \text{ tel que } a(u, v) = l(v) \text{ pour tout } v \in H^1_0(\Omega).$$

Sur un maillage $\mathcal{T}_h$ de taille $h$ dont le bord est ajusté à $\Omega$, on la discrétise dans l'espace des éléments finis de Lagrange continus de degré $k$, noté $V_h^0$, dans lequel on cherche la solution $u_h$. Si $u \in H^{s+1}(\Omega)$ avec $1 \leqslant s \leqslant k$, le lemme de Céa, l'estimation d'interpolation et la régularité elliptique (argument d'Aubin–Nitsche) donnent, sur un polytope convexe,

$$|u - u_h|_{H^1} \leqslant C\, h^s\, |u|_{H^{s+1}}, \qquad \|u - u_h\|_{L^2} \leqslant C\, h^{s+1}\, |u|_{H^{s+1}}.$$

Ces estimations sont celles que nous retrouverons, modifiées par l'enrichissement, dans la première contribution.

**Galerkin discontinu.** La méthode de Galerkin discontinu (DG) cherche la solution dans l'espace des polynômes par morceaux, sans continuité aux interfaces. Le terme de diffusion est écrit comme un système du premier ordre, avec une variable de flux auxiliaire, et les traces ambiguës aux interfaces sont remplacées par des flux numériques. Une formulation unifiée [3, 4] regroupe les différentes variantes. Nous utilisons, pour la diffusion, les méthodes à pénalisation intérieure symétrique (SIPG) et non symétrique (NIPG), ainsi qu'un flux décentré pour la convection. Pour le problème de Poisson, SIPG est d'ordre optimal en norme $H^1$ brisée et en norme $L^2$, tandis que NIPG perd un ordre en norme $L^2$ (sans surpénalisation).

**La méthode $\varphi$-FEM.** Lorsque le domaine est complexe, construire un maillage ajusté est coûteux. La méthode $\varphi$-FEM [20] plonge $\Omega$ dans un maillage de fond cartésien, sans que son bord y soit ajusté (figure 1). Le domaine est donné par une fonction level-set $\varphi$, négative dans $\Omega$ et nulle sur $\Gamma$, et la solution est cherchée sous la forme $u = \varphi w$, de sorte que la condition de Dirichlet homogène est vérifiée quel que soit $w$. Toutes les intégrales sont calculées sur des éléments entiers du maillage de fond, avec les quadratures usuelles, et une stabilisation sur les éléments coupés préserve la coercivité et le conditionnement. Pour le problème de Poisson, l'erreur est d'ordre optimal $k$ en semi-norme $H^1$ et d'ordre $k + 1/2$ en norme $L^2$.

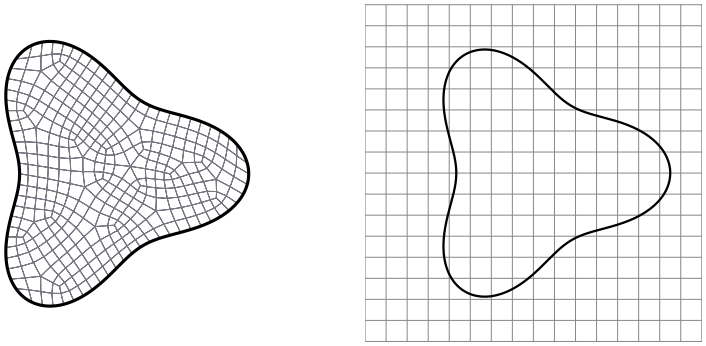


FIGURE 1 – Maillages ajusté (à gauche) et non ajusté (à droite) pour un même domaine à trois lobes.

**Réseaux informés par la physique.** Les PINNs cherchent la solution sous la forme de la sortie d'un réseau $u_\theta$, ici un réseau entièrement connecté prenant en entrée la variable d'espace $\boldsymbol{x}$ et le paramètre $\boldsymbol{\mu}$. Résoudre l'EDP revient à trouver les poids

$$\theta^\star = \operatorname*{argmin}_\theta \ \omega_r \mathcal{J}_r(\theta) + \omega_b \mathcal{J}_b(\theta) + \omega_{\text{data}} \mathcal{J}_{\text{data}}(\theta),$$

où $\omega_r$, $\omega_b$ et $\omega_{\text{data}}$ sont des poids, $\mathcal{J}_r$ l'intégrale du carré du résidu de l'EDP, $\mathcal{J}_b$ celle du carré de l'écart à la condition au bord et $\mathcal{J}_{\text{data}}$ un terme optionnel d'ajustement à des données de référence. Les intégrales, y compris celle sur l'espace des paramètres, sont estimées par une méthode de Monte-Carlo sur des points de collocation, dont la vitesse de convergence ne dépend pas de la dimension. Un PINN paramétrique est ainsi entraîné une seule fois pour toute la famille de paramètres.

Un PINN classique ne vérifie pas exactement les conditions aux limites, ne contrôle pas ses dérivées d'ordre élevé et souffre du biais spectral. Or ces dérivées jouent un rôle clé dans la première contribution, ce qui nous conduit à utiliser trois améliorations de la littérature. La première impose strictement la condition de Dirichlet en cherchant $u_\theta = \varphi\, w_\theta + g$ [25, 32], où $g$ prolonge la donnée au bord. La deuxième est l'entraînement de Sobolev [46], qui ajoute à la fonction de coût le gradient du résidu et contraint ainsi indirectement les dérivées de $u_\theta$. La troisième contre le biais spectral par des caractéristiques de Fourier à fréquences apprises [48], ce qui permet de traiter des solutions haute fréquence.

# 4 Contribution 1 : enrichir les éléments finis par un a priori neuronal

*(voir manuscrit, chapitre 3, publié dans [8])*

## 4.1 Principe

Nous supposons que nous disposons d'un a priori de la solution de l'EDP, notée $u_\theta$ et appelée « prior ». Les premières sections du chapitre traitent un paramètre $\boldsymbol{\mu}$ fixé et un prior quelconque. Dans la pratique, ce prior est la prédiction d'un PINN paramétrique. La méthode se déroule en deux phases (figure 2). Hors ligne, le PINN est entraîné une seule fois pour toute la famille de paramètres. En ligne, pour chaque valeur de $\boldsymbol{\mu}$, il est évalué pour fournir le prior, qui est injecté dans l'espace d'approximation éléments finis de Lagrange continus, puis la résolution éléments finis enrichie est effectuée de manière classique. Deux enrichissements sont proposés, l'un additif, l'autre multiplicatif.

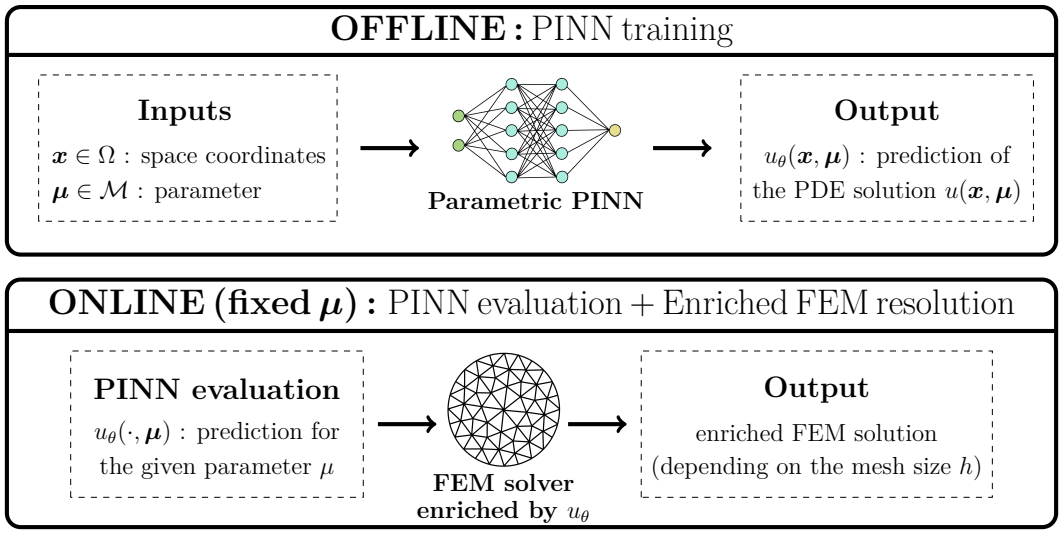



FIGURE 2 – Pipeline de la méthode enrichie. En haut, phase hors ligne (entraînement du PINN). En bas, phase en ligne (évaluation du PINN et résolution éléments finis enrichie).

## 4.2 Enrichissement additif

Pour un prior $u_\theta \in H^{s+1}(\Omega) \cap H_0^1(\Omega)$, l'espace d'essai enrichi est défini par :

$$V_h^+ := \left\{ u_h^+ = u_\theta + p_h^+,\ p_h^+ \in V_h^0 \right\},$$

et l'espace $V_h^0$ des éléments finis de Lagrange classiques est conservé pour les fonctions test. Le problème discret revient à trouver $p_h^+ \in V_h^0$ tel que

$$a(p_h^+, v_h) = l(v_h) - a(u_\theta, v_h), \qquad \forall v_h \in V_h^0,$$

c'est-à-dire à une approximation de Petrov–Galerkin classique avec un second membre modifié.

**Théorème 1.** *Soit $u \in H^{s+1}(\Omega)$ la solution du problème continu et $1 \leqslant s \leqslant k$. Alors*

$$|u - u_h^+|_{H^1} \leqslant C\, C_{gain}^+ h^s\, |u|_{H^{s+1}}, \qquad \|u - u_h^+\|_{L^2} \leqslant C\, C_{gain}^+ h^{s+1}\, |u|_{H^{s+1}},$$

*avec $C_{gain}^+ := |u - u_\theta|_{H^{s+1}}/|u|_{H^{s+1}}$.*

Les ordres de convergence classiques sont donc conservés. Le gain par rapport aux éléments finis standard est contrôlé explicitement par $C^+_{\text{gain}}$, c'est-à-dire par la qualité des dérivées d'ordre $s+1$ du prior, et non par sa seule valeur ponctuelle. Ce résultat guide le choix et l'entraînement du prior.

## 4.3 Enrichissement multiplicatif

Pour que le prior ne s'annule jamais, le problème est rehaussé d'une constante $M > 0$, d'inconnue $u_M := u + M$, et le prior devient $u_{\theta,M} := u_\theta + M$, où $M$ est choisie assez grande pour que ce prior rehaussé ne s'annule pas. L'espace d'essai est alors défini par :

$$V_h^\times := \left\{ u_{h,M}^\times = u_{\theta,M}\, p_h^\times,\ p_h^\times \in 1 + V_h^0 \right\},$$

dont les fonctions de forme sont $\phi_i^\times = u_{\theta,M}\, \phi_i$, avec $\phi_i$ les fonctions de forme de Lagrange classiques. La solution du problème initial est retrouvée par $u_h^\times = u_{h,M}^\times - M$.

**Théorème 2.** *Sous les mêmes hypothèses que théorème 1, pour $1 \leqslant s \leqslant k$,*

$$|u - u_h^\times|_{H^1} \leqslant C\, C_{gain,H^1}^{\times,M}\, h^s\, |u|_{H^{s+1}}, \qquad \|u - u_h^\times\|_{L^2} \leqslant C\, C_{gain,L^2}^{\times,M}\, h^{s+1}\, |u|_{H^{s+1}},$$

*avec*

$$C_{gain,H^1}^{\times,M} := \left| \frac{u_M}{u_{\theta,M}} \right|_{H^{s+1}} \frac{\|u_{\theta,M}\|_{W^{1,\infty}}}{|u|_{H^{s+1}}},$$
$$C_{gain,L^2}^{\times,M} := C_{\theta,M} \left| \frac{u_M}{u_{\theta,M}} \right|_{H^{s+1}} \frac{\|u_{\theta,M}\|_{W^{1,\infty}}^2}{|u|_{H^{s+1}}},$$

*où*

$$C_{\theta,M} := \|u_{\theta,M}^{-1}\|_{L^\infty} + 2|u_{\theta,M}^{-1}|_{W^{1,\infty}} + |u_{\theta,M}^{-1}|_{W^{2,\infty}}.$$

Comme dans le cas additif, les ordres de convergence classiques sont conservés, et les constantes de gain dépendent des dérivées d'ordre élevé du prior, de sorte qu'un bon prior doit approcher correctement les dérivées de la solution. La principale différence avec l'approche additive tient au choix de la constante de rehaussement $M$, dont l'influence est étudiée ci-dessous.

## 4.4 Comparaison des deux enrichissements

Le théorème suivant montre que les deux approches sont asymptotiquement équivalentes.

**Théorème 3.** *Les deux constantes de gain multiplicatives tendent vers la constante de gain additive quand le paramètre de rehaussement tend vers l'infini :*

$$C^{\times,M}_{gain,H^1} \underset{M\to\infty}{\longrightarrow} C^{+}_{gain}, \qquad C^{\times,M}_{gain,L^2} \underset{M\to\infty}{\longrightarrow} C^{+}_{gain}.$$

Numériquement, sur un problème de Poisson 1D, les expériences confirment cette limite. Pour l'approche additive, le gain théorique (environ 34,01) correspond bien aux gains observés. Pour l'approche multiplicative, les constantes s'accordent bien quand $M$ est grand, mais les estimations sous-estiment le gain pour de petites valeurs de $M$ : pour $M = 3$, le gain théorique en semi-norme $H^1$ vaut environ 8,22, contre 30 obtenus en pratique. Dans les deux premiers cas tests 1D, l'approche multiplicative ne donne de meilleurs résultats que l'approche additive que dans des cas très particuliers et pour un bon choix de $M$.

## 4.5 Mise en œuvre

Pour être efficaces, les méthodes dépendent de la qualité de l'approximation des dérivées du prior et de son intégration précise sur le domaine. Les dérivées du prior sont calculées par différentiation automatique, ce qui donne des dérivées exactes à la précision machine près. L'intégration doit se faire avec une règle de quadrature de degré plus élevé que pour les éléments finis classiques, notamment pour le terme $l(v_h) - a(u_\theta, v_h)$ de l'approche additive, qui est calculé en pratique sous forme forte (par exemple $\int_\Omega (f + \Delta u_\theta)\, v_h$ pour le laplacien).

Les conditions de Dirichlet sont imposées comme en éléments finis classiques, par élimination des degrés de liberté essentiels. Si le PINN les impose exactement par une fonction level-set, elles deviennent homogènes pour la correction additive ($p_h^+ = 0$ sur le bord si le prior est exact au bord) et se réduisent à $p_h^\times = 1$ pour la correction multiplicative. Le PINN est entraîné avec la version `PyTorch` de `ScimBa`, et les éléments finis sont implémentés avec `FEniCS` ou `FEniCSx`. À titre d'ordre de grandeur, pour les cas tests scalaires et linéaires présentés ci-dessous, l'entraînement du PINN prend moins de dix minutes sur une carte NVIDIA RTX 2000 (8 Go de VRAM), et reste sous une heure quand l'optimiseur LBFGS est utilisé.

## 4.6 Résultats numériques

Les expériences sont des instances du problème paramétrique en dimension $d \in \{1, 2, 3\}$, de complexité croissante, pour lesquelles les théorèmes précédents s'appliquent (tableau 1), à l'exception du problème de transmission, dont la solution n'est pas assez régulière. Elles vont du problème de Poisson à des cas plus complexes : un problème à

gradient discontinu (transmission 1D), un cas haute fréquence traité avec des features de Fourier, un problème elliptique anisotrope, et une géométrie courbe avec des conditions aux limites mixtes (anneau). Dans chacune d'elles, un PINN paramétrique est entraîné pour construire le prior, et l'on compare l'approche enrichie aux éléments finis standard sur un ensemble $\mathcal{D}$ de jeux de paramètres, de taille $N_\mu$. Les erreurs sont des erreurs relatives en norme $L^2$ par rapport à une solution de référence analytique ou calculée sur un maillage très fin avec un degré polynomial élevé. Le gain d'une méthode enrichie est le rapport entre l'erreur des éléments finis standard et celle de la méthode enrichie, que l'on résume sur $\mathcal{D}$ par son minimum, son maximum, sa moyenne et son écart-type.

| | | | | Gain moyen | |
|---|---|---|---|---|---|
| Cas test | Dim. | $N_\mu$ | Maillage | Additif | Multiplicatif |
| Poisson | 1D | 100 | $N = 20$ | 141 | 43 – 141 |
| Système elliptique général, régime dominé par la convection | 1D | 50 | $N = 20$ | 3,4 | 1,6 – 187 |
| Problème de transmission non lisse | 1D | 50 | $N = 21$ | 139 | – |
| Poisson, basse fréquence | 2D | 50 | $N = 20$ | 269 | – |
| Poisson, basse fréquence, entraînement de Sobolev | 2D | 50 | $N = 20$ | 768 | – |
| Poisson, haute fréquence | 2D | 50 | $N = 20$ | 350 | – |
| Problème elliptique anisotrope | 2D | 50 | $N = 20$ | 36 | – |
| Poisson sur un anneau, conditions mixtes | 2D | 50 | $h \approx 0{,}13$ | 56 | – |
| Poisson dans un cube | 3D | 50 | $N = 20$ | 137 | – |

TABLEAU 1 – Gain moyen en erreur relative $L^2$ des approches additive et multiplicative par rapport aux éléments finis $\mathbb{P}_1$ standard. L'approche multiplicative n'est testée qu'en 1D, et la plage indiquée correspond aux variantes testées : $M \in \{3, 100\}$ pour le problème de Poisson, et condition imposée ou non sur la correction multiplicative au bord pour le système elliptique ($M = 0$). $N_\mu$ est le nombre de jeux de paramètres de l'ensemble $\mathcal{D}$ et $N$ le nombre de nœuds du maillage par direction (le maillage de l'anneau, non structuré, est caractérisé par sa taille $h$).

**Gain global.** Dans tous les cas testés, l'approche additive améliore en moyenne les éléments finis standard, avec un gain moyen de 3,4 à 768 selon le cas (supérieur à 100 dans six cas sur neuf, tableau 1), dont l'ampleur dépend de la qualité du prior. Pour des éléments $\mathbb{P}_1$, les résultats moyennés sur un ensemble de paramètres montrent que l'approche enrichie atteint la précision des éléments finis standard avec des maillages de 5 (problème anisotrope 2D) à 28 (problème de Poisson basse fréquence avec entraînement de Sobolev) fois plus grossiers, ce qui réduit le coût de calcul en ligne.

**Rôle des dérivées du prior.** Les expériences vérifient d'abord les ordres de convergence prévus par les théorèmes et valident le rôle des dérivées du prior. En 1D, un prior obtenu par apprentissage sur des données approche mieux la solution qu'un PINN, mais

approche moins bien ses dérivées, en particulier les dérivées secondes. Comme attendu par les analyses théoriques, celui-ci donne de moins bons résultats une fois injecté dans les éléments finis, ce qui guide le choix des PINNs. Le problème de transmission 1D, dont la solution a un gradient discontinu, sort du cadre théorique. Il est traité avec une architecture de réseau adaptée à l'interface, et les dérivées d'ordre élevé y jouent encore le rôle principal. En 2D, l'entraînement de Sobolev, qui pénalise le gradient du résidu, améliore nettement le gain par rapport à un entraînement standard (de 269 à 768 en moyenne sur le maillage le plus grossier du tableau 1), ce qui confirme que la qualité des dérivées est déterminante.

**Influence de l'entraînement du PINN.** L'initialisation des poids du PINN influence l'ampleur du gain, mais un gain existe systématiquement : sur le problème de Poisson 1D, pour 30 initialisations, le gain minimal de l'approche additive vaut en moyenne 23 (écart-type 6) pour $N = 20$. Sur le problème de Poisson 3D, en entraînant neuf PINNs qui ne diffèrent que par le nombre d'époques et de points de collocation, on observe que le nombre de points de collocation influence fortement le gain, alors que le nombre d'époques joue un rôle moindre.

**Approche multiplicative.** L'approche multiplicative n'est testée qu'en 1D. Sur le problème de Poisson, son gain dépend de $M$ (43 pour $M = 3$) et rejoint celui de l'approche additive quand $M$ grandit (141 pour $M = 100$). Sur le système elliptique dominé par la convection, c'est elle qui donne les meilleurs résultats sur l'ensemble de paramètres (187 contre 3,4 pour l'approche additive), mais seulement si aucune condition n'est imposée sur la correction au bord : avec $p_h^{\times} = 1$, le gain tombe à 1,6. L'approche multiplicative ne fait donc mieux que l'approche additive que dans des cas très particuliers, et pour un bon choix de $M$ ou de l'imposition des conditions au bord. Les autres cas tests sont par conséquent traités avec l'approche additive seule.

**Le cas 3D et le coût paramétrique.** Le cas 3D (problème de Poisson dans un cube) donne un gain moyen de 137 sur l'ensemble de paramètres pour $N = 20$, et examine les temps de calcul dans le cadre paramétrique complet. Pour atteindre une erreur relative de $10^{-3}$, les éléments finis standard exigent $N = 152$ nœuds par direction (temps d'exécution 76,5 s), contre $N = 12$ pour la méthode additive (0,052 s), soit une accélération d'environ 1471 fois en ligne. Il faut toutefois tenir compte du coût hors ligne, l'entraînement du PINN ayant pris 707,84 s. En le comptant, la méthode additive devient plus avantageuse que les éléments finis standard dès 19 jeux de paramètres à résoudre pour une erreur de $10^{-3}$, et dès 5 jeux pour une erreur de $5 \cdot 10^{-4}$. C'est dans ce cadre paramétrique que l'approche prend tout son intérêt.

## 4.7 Au-delà du cadre théorique : les équations de Navier–Stokes

Le manuscrit propose aussi une extension purement heuristique à un problème non linéaire à plusieurs champs, les équations de Navier–Stokes stationnaires incompressibles

couplées à la température par l'approximation de Boussinesq, sur la cavité carrée chauffée. Les paramètres sont la viscosité et la conductivité thermique, dans $[0{,}01; 0{,}1]^2$, et la difficulté croît avec le nombre de Rayleigh. Cette extension n'est couverte ni par la théorie ni par la publication [8]. Un PINN paramétrique fournit le prior, utilisé soit pour initialiser le solveur de Newton, comme dans [1, 40], soit comme enrichissement additif. Les résultats montrent que le solveur de Newton initialisé naturellement n'aboutit pas sur le maillage le plus grossier pour le deuxième jeu de paramètres et sur aucun maillage pour le plus grand nombre de Rayleigh, alors que l'initialisation par le PINN et l'approche additive aboutissent partout, en 2 à 3 itérations au lieu de 6 pour le premier jeu et en 3 au lieu de 8 à 10 pour le deuxième. Cette robustesse vient de l'initialisation par le PINN, commune aux deux approches. À maillage donné, l'approche additive est plus précise que les éléments finis standard, mais avec des gains modestes comparés aux cas scalaires : de 2,9 à 6,1 sur les erreurs en $u_1$, $u_2$ et $T$ pour les deux premiers jeux de paramètres, et seulement de 1,1 à 2,3 pour le plus grand nombre de Rayleigh, où la précision du PINN diminue. Le coût d'une résolution est comparable à celui des éléments finis initialisés par le PINN. Ces résultats rejoignent les conclusions des autres cas tests sur le rôle de la qualité du prior.

## 4.8 Bilan et limites

La première contribution fournit une méthode d'enrichissement dont l'erreur est contrôlée a priori, avec un gain explicite qui dépend de la qualité des dérivées du prior, et qui réduit le coût de résolution dans le cadre paramétrique. Ses limites sont que la théorie reste restreinte aux problèmes elliptiques linéaires scalaires et que le gain se dégrade quand la précision du prior diminue, comme le montre le cas de Navier–Stokes à grand nombre de Rayleigh. Les pistes d'amélioration identifiées sont un meilleur prior (opérateurs neuronaux, pénalisation des dérivées dans la fonction de coût), puis l'extension aux problèmes instationnaires, non linéaires et aux géométries complexes.

# 5 Contribution 2a : `ScimBa` et schémas de Galerkin apprenables

*(voir manuscrit, chapitre 4)*

La seconde contribution part d'un constat : pour apprendre une composante d'un schéma de Galerkin (une base, une application de maillage, un champ de l'EDP) en minimisant une fonction de coût qui porte sur la solution approchée, il faut pouvoir différentier à travers l'assemblage et la résolution du schéma. Cette section présente la bibliothèque qui rend cela possible, `ScimBa`, dans sa version en `JAX`, développée par l'équipe-projet MACARON, ainsi que le formalisme d'apprentissage qui s'y appuie. Elle ne discute pas encore l'intérêt de paramétrer telle ou telle composante, qui fait l'objet de la section suivante.

## 5.1 Programmation différentiable

La programmation différentiable [12] étend la différentiation automatique au-delà des réseaux de neurones, à des programmes arbitraires (boucles, branchements, structures de données). Parmi les trois façons de calculer une dérivée numériquement [10], la différentiation symbolique est exacte mais limitée aux expressions en forme close, et les différences finies, applicables à tout programme, introduisent une erreur de troncature et exigent une évaluation par dimension, ce qui devient prohibitif dès que le nombre de paramètres est grand. La différentiation automatique a les avantages des deux : elle est exacte et s'applique à des programmes arbitraires. Elle propose deux modes, avec des produits Jacobien-vecteur (JVP) et vecteur-Jacobien (VJP), ce dernier étant le plus adapté lorsque la sortie est un scalaire, comme une fonction de coût dépendant d'un grand nombre de paramètres.

La bibliothèque `JAX` [13] fournit ces transformations (`grad`, `jvp`, `vjp`), ainsi que la compilation (`jit`) et la vectorisation (`vmap`). Elles opèrent sur des *pytrees*, c'est-à-dire des structures dont les feuilles sont des tableaux `JAX`. Dans `ScimBa`, chaque module est enregistré comme un pytree, en séparant ses attributs dynamiques (les tableaux sur lesquels on différentie ou on vectorise) de ses attributs statiques. Un maillage, une base, une application ou un réseau sont ainsi tous des objets différentiables de même nature, et un schéma de Galerkin complet peut être traité comme une seule fonction composée.

## 5.2 Architecture modulaire d'un schéma de Galerkin discontinu

L'architecture est construite autour du schéma de Galerkin discontinu, qui travaille cellule par cellule, sans aucune relation imposée entre les degrés de liberté de deux cellules voisines. Elle se compose des modules suivants, chacun pouvant être rendu apprenable.

**Géométrie et intégration.** Le module de quadrature construit, sur l'hypercube de référence $\hat{K} = [0,1]^d$ et sur ses faces, des points et des poids par tensorisation de quadratures 1D (Gauss–Legendre par exemple). Le maillage est défini à partir d'un maillage logique de $[0,1]^d$, composé d'hypercubes, que l'on déforme par une application $\mathcal{G}_\theta$ (analytique, ou apprise par un réseau) pour obtenir le maillage physique $\mathcal{T}_h$. Le transport de la quadrature jusqu'à la cellule physique passe par les transformations affines locales puis par $\mathcal{G}_\theta$, dont le Jacobien intervient dans la transformation des poids de quadrature. Le choix d'un maillage d'hypercubes, et non de simplexes, conduit à des bases $\mathbb{Q}_k$ plutôt que $\mathbb{P}_k$. Il permet de réutiliser tel quel le même maillage logique et la même quadrature pour le schéma de Galerkin discontinu et pour ses extensions aux éléments finis continus et à $\varphi$-FEM, présentées plus bas, ainsi que la même application $\mathcal{G}_\theta$ pour le DG et les éléments finis.

**Fonctions de base.** Trois familles de bases sont disponibles : des bases analytiques (Lagrange, Taylor), et deux types de bases neuronales qui enrichissent la base analytique

par un réseau prenant les coordonnées physiques en entrée, soit un unique réseau commun à toutes les cellules (base *patchwise*), soit un réseau par cellule (base *cellwise*, coûteuse et réservée aux maillages grossiers). Contrairement au cas polynomial, rien ne garantit alors que les $N_b$ fonctions de base restent linéairement indépendantes sur une cellule : cela dépend des paramètres appris.

**Variable DG et post-traitement.** La solution discrète est reconstruite cellule par cellule à partir des degrés de liberté locaux $\boldsymbol{\beta}_c$ et des fonctions de base, $w_{h,\theta}|_{K_c} = \sum_i \beta_{c,i}\phi^{\theta}_{c,i}$. Un post-traitement optionnel $\mathcal{P}_\theta$, analytique ou neuronal, linéaire ou non, agit ensuite sur cette variable pour donner la solution $u_{h,\theta} = \mathcal{P}_\theta(w_{h,\theta})$. Les applications, les bases et le post-traitement sont les trois sources possibles de paramétrisation ; $\theta$ désigne l'ensemble de leurs paramètres.

**Schéma, flux et conditions aux limites.** Les formes faibles de l'EDP, les flux numériques (SIPG, NIPG, décentré), les conditions aux limites (Dirichlet, Neumann, Robin) et les champs de l'EDP sont des classes composables. L'assemblage procède en trois passes indépendantes (volume, faces intérieures, faces de bord), vectorisées sur les cellules et sur les fonctions test, et produit le résidu $\boldsymbol{R}_h(\boldsymbol{\beta}_h)$ du schéma. Ce résidu est affine en $\boldsymbol{\beta}_h$ tant que les champs de l'EDP ne dépendent pas de $u$ et que le post-traitement est linéaire, et non linéaire sinon.

**Résolution.** Un résidu affine se résout par une seule étape de Newton. Dans le cas non linéaire, `ScimBa` propose une méthode de Newton dense, et une variante sans matrice (Newton–Krylov) dans laquelle le système linéaire de chaque étape est résolu par une méthode de Krylov (gradient conjugué, GMRES, BiCGStab [44]) qui n'accède au Jacobien que par des produits matrice-vecteur obtenus par différentiation automatique. Plusieurs préconditionneurs sans matrice sont disponibles, dont un bloc-Jacobi qui exploite la structure par cellule du DG. Enfin, `vmap` permet de résoudre en parallèle, dans un seul programme compilé, un lot de problèmes qui ne diffèrent que, par exemple, par le paramètre $\boldsymbol{\mu}$ de l'EDP. Sur le problème de Poisson 2D sur le carré, la résolution d'un lot de $N_\mu = 256$ problèmes est jusqu'à 12,6 fois plus rapide qu'une boucle compilée.

## 5.3 Extensions aux éléments finis et à $\varphi$-FEM

Les éléments finis continus et $\varphi$-FEM sont construits au-dessus de cette pile, en ajoutant des modules de variables et d'assemblage sans modifier les modules de géométrie, de bases et de résolution.

**Éléments finis continus.** Le schéma DG est ici le cas général, et les éléments finis continus en sont un cas particulier, obtenu en ajoutant une unique contrainte de continuité qui identifie les degrés de liberté partagés par des cellules voisines. Le module `VariablesFE` remplace `VariablesDG` : il construit, une fois pour toutes à partir de la

topologie du maillage, une table de connectivité qui associe à chaque degré de liberté local un indice global, tout en conservant la formule de reconstruction cellule par cellule. L'assemblage se réduit alors à une passe de volume et, si le bord porte des conditions de Neumann ou de Robin, à une passe sur les faces de bord, sans les termes de saut du DG.

**$\varphi$-FEM.** Le bord $\Gamma$ n'est plus ajusté par le maillage, qui est maintenant un maillage cartésien de fond $\mathcal{T}_h^{\mathcal{O}}$. Une classe de classification géométrique étiquette chaque cellule (intérieure, extérieure ou coupée par $\Gamma$) d'après le signe de $\varphi$ à ses sommets, et ne conserve pour l'assemblage que les cellules actives (intérieures ou coupées), ainsi que les faces à stabiliser. L'inconnue $w_{h,\theta}$ est reconstruite comme pour les éléments finis continus, puis multipliée par $\varphi$ par un post-traitement dédié, $u_{h,\theta} = \varphi\, w_{h,\theta}$, et les termes de stabilisation propres à $\varphi$-FEM sont ajoutés au schéma. La présentation se limite au problème de Poisson avec condition de Dirichlet homogène, comme dans [20].

Dans l'état actuel de la bibliothèque, le cas d'usage apprenable naturel de $\varphi$-FEM, l'optimisation de forme par une fonction level-set apprise, se heurte à un problème de recompilation de `JAX` et est laissé à des travaux futurs. Seuls le DG et les éléments finis sont donc disponibles comme schémas apprenables.

## 5.4 Formalisme d'apprentissage à travers le solveur

*(voir manuscrit, chapitre 4, section « Learnable Galerkin schemes »)*

Soit $\theta$ l'ensemble des paramètres apprenables, de taille $N_\theta$, qui peuvent paramétrer un champ de l'EDP (coefficient de diffusion ou de réaction, par exemple), les poids d'un réseau qui enrichit les bases, l'application $\mathcal{G}_\theta$ ou le post-traitement $\mathcal{P}_\theta$. Les degrés de liberté $\boldsymbol{\beta}_h(\theta) \in \mathbb{R}^{N_{\text{dofs}}}$ dépendent de $\theta$ puisqu'ils sont obtenus en résolvant le schéma. Pour une fonction de coût $\mathcal{J}$ mesurant la qualité de $u_{h,\theta}$ (par exemple la fonction de coût d'un PINN, ou un terme d'ajustement à des données), l'apprentissage est un problème d'optimisation à deux niveaux, écrit sous forme contrainte :

$$\theta^\star = \operatorname*{argmin}_{\theta}\ \mathcal{J}(\boldsymbol{\beta}_h(\theta);\theta) \quad \text{sous la contrainte} \quad \boldsymbol{R}_h(\boldsymbol{\beta}_h(\theta);\theta) = \mathbf{0}.$$

Il ne s'agit pas d'une optimisation alternée : à chaque itération sur $\theta$, le schéma est résolu à nouveau, et $\boldsymbol{\beta}_h$ est toujours une fonction déterministe de $\theta$.

Pour calculer le gradient de $\mathcal{J}$ par rapport à $\theta$, la règle de dérivation en chaîne donne

$$\frac{d\mathcal{J}}{d\theta} = \frac{\partial \mathcal{J}}{\partial \theta}\Big|_{\boldsymbol{\beta}_h} + \frac{\partial \mathcal{J}}{\partial \boldsymbol{\beta}_h}\,\frac{d\boldsymbol{\beta}_h}{d\theta},$$

où le premier terme (dépendance directe) est calculé par différentiation automatique en « oubliant » le chemin qui passe par $\boldsymbol{\beta}_h$. Le second terme (dépendance indirecte) traduit la façon dont la solution du schéma varie quand $\theta$ change. Il exige la sensibilité $d\boldsymbol{\beta}_h/d\theta$ des degrés de liberté, qui n'est pas accessible directement puisque $\boldsymbol{\beta}_h$ n'est défini

qu'implicitement, par la résolution du schéma. En dérivant la contrainte, on obtient

$$d\boldsymbol{\beta}_h/d\theta = -J_{\boldsymbol{R}_h}^{-1}\,\partial \boldsymbol{R}_h/\partial\theta,$$

avec $J_{\boldsymbol{R}_h} = \partial \boldsymbol{R}_h/\partial\boldsymbol{\beta}_h$ le Jacobien du résidu. Appliquer $J_{\boldsymbol{R}_h}^{-1}$ à la matrice $N_{\text{dofs}} \times N_\theta$ requerrait $N_\theta$ résolutions de systèmes linéaires, ce qui est prohibitif pour un réseau de neurones. Or seul le produit de $d\boldsymbol{\beta}_h/d\theta$ par le vecteur $\partial\mathcal{J}/\partial\boldsymbol{\beta}_h$ est nécessaire. Avec la variable adjointe $\boldsymbol{\lambda}_h$, solution de

$$J_{\boldsymbol{R}_h}(\boldsymbol{\beta}_h;\theta)^T\,\boldsymbol{\lambda}_h = \Big(\frac{\partial\mathcal{J}}{\partial\boldsymbol{\beta}_h}\Big)^T,$$

le gradient total s'écrit

$$\frac{d\mathcal{J}}{d\theta} = \frac{\partial\mathcal{J}}{\partial\theta}\Big|_{\boldsymbol{\beta}_h} - \boldsymbol{\lambda}_h^T\,\frac{\partial \boldsymbol{R}_h}{\partial\theta}(\boldsymbol{\beta}_h;\theta),$$

et ne demande qu'une seule résolution linéaire, quelle que soit la dimension de $\theta$. Le produit $\boldsymbol{\lambda}_h^T\,\partial\boldsymbol{R}_h/\partial\theta$ est obtenu par un VJP de l'assemblage du résidu. Le coût total est donc celui de quelques étapes de Newton (problème direct) et d'une résolution linéaire (problème adjoint), qui font intervenir le même Jacobien, à une transposition près : le solveur sans matrice du problème direct peut être réutilisé tel quel pour le problème adjoint. Ce formalisme est valable pour les schémas DG, éléments finis et $\varphi$-FEM, puisqu'il ne repose que sur l'équation de résidu générique.

## 5.5 Validations numériques

*(voir manuscrit, chapitre 4, section « Numerical validations »)*

Sept cas tests de complexité croissante valident la bibliothèque. Les six premiers vérifient, pour $\theta$ fixé, les ordres de convergence prévus par la théorie en norme $L^2$ et en semi-norme $H^1$ (cassée pour le DG) ; le dernier valide le formalisme apprenable lui-même. Les cas 1D et le premier cas 2D s'inspirent de ceux du chapitre précédent, avec des bases $\mathbb{Q}_k$ et une comparaison entre DG et éléments finis. Ces six cas sont le problème de Poisson 1D avec conditions de Dirichlet, puis de Dirichlet et Neumann, un problème elliptique 1D à convection dominante, un problème de Poisson non linéaire 1D, et le problème de Poisson 2D sur un carré, puis sur un domaine à trois lobes. Le dernier est un problème inverse 1D.

Plusieurs enseignements se dégagent de ces cas. En DG, la parité du degré entre en jeu : pour NIPG, l'ordre optimal $k+1$ en norme $L^2$ n'est atteint que pour $k$ impair sur un maillage uniforme en 1D, phénomène connu, propre au cas 1D sur maillage uniforme [33], et sans lien avec le type de condition aux limites. Dans le régime à convection dominante, on retrouve la dégradation de SIPG seul en norme $L^2$ pour $k$ pair, corrigée par le flux décentré. Sur le domaine à trois lobes, l'application apprise par un réseau fournit un maillage dont la qualité limite la précision. L'erreur $L^2$ plafonne vers $9 \cdot 10^{-3}$, limitée par l'erreur géométrique de l'application ($\approx 1{,}1 \cdot 10^{-3}$ au bord), et l'erreur supplémentaire par rapport à l'application analytique est localisée au centre

du domaine, là où les cellules sont les plus déformées, alors que $\varphi$-FEM, qui n'a pas besoin d'ajuster le bord, conserve les ordres optimaux (les erreurs étant calculées sur le domaine $\Omega_h$ couvert par le maillage actif, comme dans [20]).

**Un problème inverse.** Le dernier cas valide le formalisme apprenable. On cherche à retrouver, pour le problème de réaction-diffusion $-\partial_{xx}u + Ru = f$ sur $(0,1)$ avec $u = 2$ au bord, un coefficient de réaction inconnu $R$, à partir de $N_{\text{data}} = 60$ mesures sans bruit de la solution exacte $u(x) = 2 + \sin(2\pi x)$, le coefficient à retrouver étant $R(x) = 10\,x(1-x)$. Le coefficient est représenté par un réseau entièrement connecté $R_\theta$ (2 couches cachées de 16 neurones, activation tanh, $N_\theta = 321$ paramètres), inséré comme champ apprenable dans un schéma SIPG ($N_{\text{cells}} = 40$, $\mathbb{Q}_1$). Seul le terme d'ajustement aux données est minimisé, par une méthode de gradient naturel sans matrice [39] sur 100 itérations, le gradient étant calculé par différentiation implicite et méthode adjointe. Les erreurs $L^2$ finales sont de $6{,}77 \cdot 10^{-4}$ pour la solution $u_{h,\theta}$ et de $3{,}92 \cdot 10^{-2}$ pour le coefficient $R_\theta$, ce dernier étant naturellement moins précis puisqu'il n'est retrouvé qu'indirectement à travers 60 mesures. L'apprentissage prend environ 11 secondes sur un GPU NVIDIA RTX 2000 (figure 3).

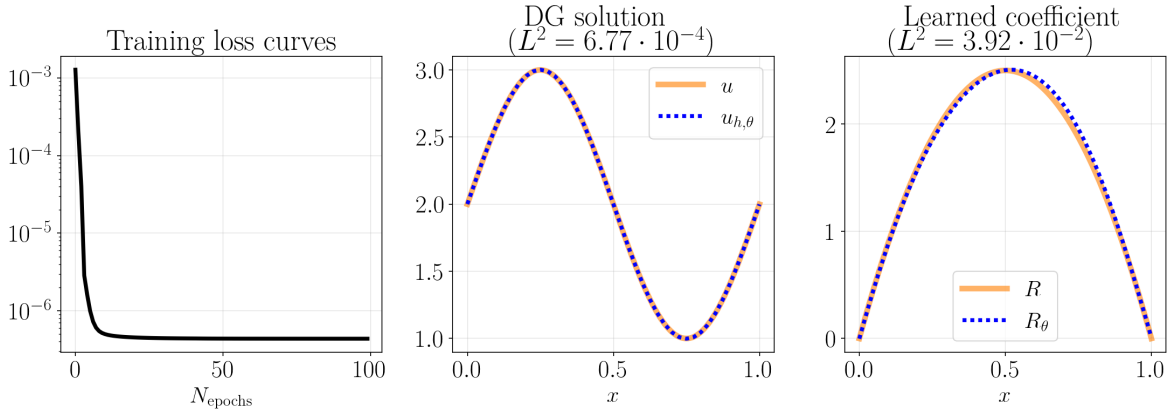



FIGURE 3 – Problème inverse 1D. Gauche : courbe de la fonction de coût. Milieu : solution DG $u_{h,\theta}$ et solution exacte $u$. Droite : coefficient appris $R_\theta$ et coefficient exact $R$.

## 5.6 Bilan

Ce chapitre fournit les briques d'un schéma de Galerkin différentiable (DG, éléments finis, $\varphi$-FEM) dont les ordres de convergence théoriques sont retrouvés, et un formalisme d'apprentissage à deux niveaux dont le coût du gradient ne dépend pas de la dimension de $\theta$. Il n'apporte pas à lui seul de gain de précision : l'intérêt de paramétrer certaines composantes du schéma, en particulier les bases, est étudié dans la section suivante.

# 6 Contribution 2b : bases de Galerkin enrichies par un réseau

*(voir manuscrit, chapitre 5)*

Cette section utilise `ScimBa` et le formalisme de la section précédente pour revenir sur la troisième limite identifiée en introduction, le manque d'expressivité des espaces d'approximation classiques, qui exigent un maillage très fin pour représenter des structures locales comme une couche limite. Nous étudions si l'on peut, à nombre de degrés de liberté $N_{\text{dofs}}$ fixé, obtenir une solution plus précise en enrichissant la base de Galerkin par un réseau de neurones, d'abord pour une valeur fixée du paramètre $\boldsymbol{\mu}$ (en 1D puis 3D), puis pour toute une famille de paramètres (en 1D). Il s'agit d'une étude de faisabilité et non d'un solveur compétitif : contrairement à la première contribution, aucune estimation d'erreur n'est disponible pour la base apprise.

Dans toute cette section, comme pour la première contribution, le gain est le rapport entre l'erreur relative en norme $L^2$ de la méthode classique et celle de la méthode enrichie, à $N_{\text{dofs}}$ fixé. Il ne tient pas compte du coût de l'entraînement, discuté séparément, ni du nombre $N_\theta$ de poids du réseau, compté à part.

## 6.1 Motivation : une étude dimensionnelle

Les bases locales classiques deviennent coûteuses, en nombre de degrés de liberté et donc en mémoire, quand la dimension $d$ augmente : c'est la malédiction de la dimension [15]. À l'inverse, un réseau peut tirer parti de la structure de la solution [7]. Pour l'illustrer, nous considérons sur $\Omega = [0,1]^d$ le problème de Poisson $-\Delta u = f$ avec la solution manufacturée $u(\boldsymbol{x}) = \sin\left(2\pi \sum_{i=1}^{d} x_i\right)$, une onde plane dans la direction diagonale. Cette solution est volontairement favorable aux réseaux, puisqu'il suffit d'aligner les poids de la première couche avec cette direction pour n'avoir plus qu'un profil 1D à représenter, alors qu'un maillage cartésien ne voit pas cette direction. L'étude ne prétend donc pas montrer que les réseaux échappent en général à la malédiction de la dimension, qui les affecte aussi pour des fonctions plus générales [27, 53].

Nous résolvons le problème par éléments finis et par Galerkin discontinu ($\mathbb{Q}_1$, maillages cartésiens uniformes, $d \in \{1,2,3,4\}$), et nous l'approchons par un réseau entièrement connecté à 3 couches cachées, entraîné soit par ajustement de la solution exacte (data-fitting), soit avec un PINN, avec l'optimiseur de gradient naturel (ENG) [39] et 10 initialisations. Le tableau 2 compare, pour atteindre une même erreur relative $L^2$ de $6 \cdot 10^{-2}$ (proche de la meilleure précision obtenue en 4D avec notre carte graphique de 8 Go), le nombre de degrés de liberté des méthodes classiques et le nombre de poids d'un réseau de largeur 16 ainsi que l'erreur moyenne qu'il atteint.

Le nombre de degrés de liberté croît exponentiellement avec $d$ (de 9 à 38 416 pour les éléments finis et de 16 à 65 536 pour le DG), alors que le nombre de poids n'augmente que linéairement, avec une pente égale à la largeur du réseau, et que l'erreur du réseau ne se dégrade pas notablement avec $d$. Les deux nombres ne sont toutefois pas comparables :

l'un est le coût d'une résolution linéaire, l'autre celui d'une optimisation non convexe, dont la qualité dépend de l'initialisation et des hyperparamètres, sans contrôle a priori et sans garantie sur l'erreur. Les réseaux présentent donc un avantage net en mémoire, mais pas de garanties, à l'inverse des méthodes de Galerkin, ce qui motive une hybridation : une base de Galerkin conserve le cadre variationnel et le contrôle de l'erreur, tandis qu'un réseau est chargé de capturer la structure de la solution. Les sections suivantes étudient cette hybridation sous la forme d'un enrichissement, sur des solutions qui présentent une structure locale raide (couche limite ou pic) que les bases classiques représentent mal.

| | $N_{\text{dofs}}$ pour $6 \cdot 10^{-2}$ | | Réseau de largeur 16 | | |
|---|---|---|---|---|---|
| $d$ | EF | DG | $N_\theta$ | erreur data-fitting | erreur PINN |
| 1 | 9 | 16 | 593 | $8 \cdot 10^{-5}$ | $1 \cdot 10^{-5}$ |
| 2 | 121 | 256 | 609 | $7 \cdot 10^{-5}$ | $3 \cdot 10^{-5}$ |
| 3 | 1 728 | 4 096 | 625 | $1 \cdot 10^{-4}$ | $8 \cdot 10^{-5}$ |
| 4 | 38 416 | 65 536 | 641 | $1 \cdot 10^{-4}$ | $2 \cdot 10^{-5}$ |

TABLEAU 2 – Problème de Poisson en dimension $d$. Degrés de liberté nécessaires aux éléments finis (EF) et au Galerkin discontinu (DG) pour atteindre une erreur relative $L^2$ de $6 \cdot 10^{-2}$, et nombre de poids et erreur relative $L^2$ moyenne (sur 10 initialisations) d'un réseau de largeur 16 entraîné par data-fitting ou comme PINN.

## 6.2 Principe de l'enrichissement multiplicatif

Dans les trois sections qui suivent, la fonction de base apprise est le produit de la fonction de base de Lagrange locale $\phi_{c,i}$ par un multiplicateur $m_\theta > 0$, construit à partir d'un réseau entièrement connecté $v_\theta$ à sortie scalaire :

$$\phi^\theta_{c,i}(\boldsymbol{x}) = m_\theta(\boldsymbol{x})\, \phi_{c,i}(\boldsymbol{x}),$$

pour une base *patchwise* (un réseau pour tout le domaine) et $\phi^\theta_{c,i} = m_{c,\theta}\, \phi_{c,i}$ pour une base *cellwise* (un réseau par cellule). La solution $u_{h,\theta}$ est la solution du schéma de Galerkin (éléments finis ou DG) dans l'espace engendré par cette base. Un multiplicateur constant redonne l'espace classique, et le réseau est initialisé de sorte que l'entraînement parte de cette situation. Ce produit rappelle l'espace multiplicatif de la première contribution, avec deux différences : le multiplicateur n'est pas préentraîné puis figé, mais appris à travers le solveur, et sa positivité, qui joue le rôle de la condition de non-annulation de l'a priori (imposée dans la première contribution par un rehaussement), est ici garantie par construction.

L'entraînement résout le problème d'optimisation à deux niveaux de la section précédente avec une fonction de coût de type PINN, dont le résidu est estimé par Monte Carlo sur des points de collocation tirés à chaque époque :

$$\min_\theta \ \omega_r \mathcal{J}_r(\theta) + \omega_b \mathcal{J}_b(\theta) \quad \text{sous la contrainte} \quad \boldsymbol{R}_h(\boldsymbol{\beta}_h(\theta); \theta) = \mathbf{0}.$$

Ici, $\mathcal{J}_r$ et $\mathcal{J}_b$ sont les termes de résidu et de bord, pondérés par $\omega_r$ et $\omega_b$. La solution de référence n'intervient jamais dans l'entraînement, seulement dans le calcul des erreurs. L'espace appris dépend du maillage et du degré polynomial : un entraînement est donc nécessaire pour chaque triplet $(\boldsymbol{\mu}, k, N_{\text{cells}})$.

## 6.3 Cas non paramétrique en 1D

Nous considérons sur $\Omega = (0,1)$ le problème d'advection-diffusion $-\frac{1}{\text{Pe}}u'' + (\mu_1 + \mu_2 x)u' = 1$ avec $u = 0$ au bord et $\text{Pe} = 20$, pour trois paramètres fixés, $\boldsymbol{\mu}^{(1)} = (0{,}8; 0{,}8)$, $\boldsymbol{\mu}^{(2)} = (1; 1)$ et $\boldsymbol{\mu}^{(3)} = (1{,}2; 1{,}2)$. Le transport est dirigé vers $x = 1$ et la solution présente une couche limite en ce point, plus raide quand $\boldsymbol{\mu}$ augmente. Faute de solution analytique, l'erreur est mesurée par rapport à une solution de référence obtenue par éléments finis $\mathbb{Q}_4$ sur un maillage uniforme de 128 cellules, dont la précision est de l'ordre de $10^{-8}$ ou mieux : les gains très élevés sont donc des bornes inférieures.

**Choix du multiplicateur.** Nous avons d'abord pris le multiplicateur le plus simple, affine en la sortie du réseau, $m_\theta = 2 + 0{,}1\, v_\theta$ (réseau de 2 couches cachées de largeur 16, soit $N_\theta = 321$ poids dans une première étude). L'étude à budget fixé ($N_{\text{dofs}} = 25$ et réseaux plus petits, $134 \leq N_\theta \leq 151$, 20 initialisations pour chacune des 20 configurations de $(k, N_{\text{cells}})$ et de forme de réseau, soit 400 entraînements) montre que 288 entraînements sur 400 font mieux que les éléments finis classiques, tandis que 101 font moins bien et que 45 explosent (la loss devient NaN, le réseau retenu restant celui de plus faible fonction de coût avant l'explosion). L'analyse de la forme et du signe des multiplicateurs appris explique l'issue de chaque entraînement :

- un multiplicateur constant redonne l'erreur classique, puisqu'il ne modifie pas l'espace éléments finis ;
- un multiplicateur qui change de signe ne fait jamais mieux (59 entraînements), puisque toute fonction de l'espace enrichi s'annule alors là où le multiplicateur s'annule, alors que la solution exacte est strictement positive à l'intérieur du domaine ;
- un multiplicateur avec une marche (variation très raide) ne fait jamais mieux non plus (8 entraînements), car il rend les fonctions de l'espace quasi discontinues, ce que les degrés de liberté ne peuvent pas corriger.

Ces deux causes d'échec conduisent à deux modifications successives du multiplicateur, étudiées au même budget. La première impose la positivité, avec $m_\theta = \exp(2 \tanh v_\theta)$, ce qui supprime les changements de signe mais pas les marches, qui restent la principale cause d'échec et que l'on observe juste avant l'explosion dans tous les cas où celle-ci se produit. La seconde borne les poids de chaque couche par un soft clipping à quatre fois leur norme à l'initialisation, ce qui limite la raideur du réseau et empêche les marches. Avec ce dernier multiplicateur, aucun entraînement n'explose ni ne fait moins bien que les éléments finis classiques, et 376 entraînements sur 400 font mieux, contre 288 et 290 pour les deux premiers multiplicateurs. En contrepartie, la positivité et la borne restreignent l'ensemble des fonctions représentables : le multiplicateur affine, moins

fiable, atteint les gains les plus élevés. Nous retenons le multiplicateur positif à poids bornés.

**Gains obtenus.** La figure 4 montre, pour $\boldsymbol{\mu}^{(1)}$, $k = 3$ et $N_{\text{cells}} = 4$, la solution classique, qui oscille et dépasse la solution exacte près de la couche limite, et la solution à base apprise de la meilleure initialisation (celle d'erreur la plus faible parmi les initialisations dont la fonction de coût passe sous le seuil de $10^{-3}$), indiscernable de la référence à l'échelle de la figure, dont l'erreur reste au plus de l'ordre de $10^{-5}$ contre jusqu'à environ $10^{-1}$ pour la solution classique (gain de 7 014, l'erreur classique étant de $5{,}1 \cdot 10^{-2}$). Plus généralement, pour les trois valeurs de $\boldsymbol{\mu}$, $k \in \{3, 4\}$ et $N_{\text{cells}} \in \{2, 4, 8\}$, le gain de la meilleure initialisation va de 195 à 104 590 avec le multiplicateur retenu, sauf pour les deux configurations ($k = 3$, $N_{\text{cells}} = 2$, pour $\boldsymbol{\mu}^{(2)}$ et $\boldsymbol{\mu}^{(3)}$) où aucune initialisation n'atteint ce seuil. Le gain diminue en général quand $\boldsymbol{\mu}$ augmente, la couche limite étant plus raide. Le degré $k$ et le nombre de cellules sont les deux leviers déterminants, sans que le gain soit pour autant monotone en chacun d'eux, alors que la largeur du réseau a peu d'influence sur ce cas.

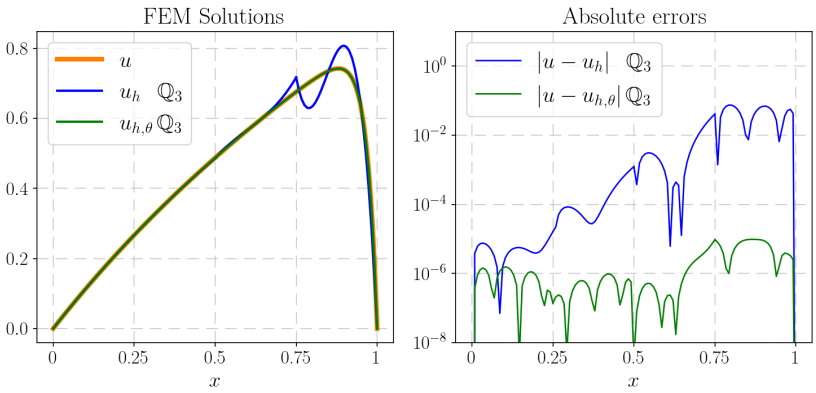



FIGURE 4 – Advection-diffusion 1D avec $\boldsymbol{\mu}^{(1)}$, éléments finis $\mathbb{Q}_3$ sur $N_{\text{cells}} = 4$ cellules (13 degrés de liberté), base *patchwise* avec le multiplicateur positif à poids bornés. Gauche : solution exacte $u$, solution classique $u_h$ et solution à base apprise $u_{h,\theta}$ (meilleure initialisation). Droite : erreurs absolues.

**Variantes et limites.** Les quatre variantes étudiées (éléments finis ou DG, base *patchwise* ou *cellwise*) donnent des résultats du même ordre, avec des gains de $10^2$ à $10^5$ pour la meilleure initialisation. La base *cellwise*, dont le nombre de poids croît avec le nombre de cellules, n'apporte pas de gain systématique par rapport à la base *patchwise* et est moins robuste sur les maillages grossiers, où moins d'initialisations atteignent le seuil. Ces résultats doivent toutefois être nuancés par quatre limites, qui concernent le cas $k = 1$, le choix de la meilleure initialisation, la base de comparaison avec les méthodes classiques et le coût de l'entraînement. Pour $k = 1$ avec les éléments finis continus, aucune initialisation n'atteint le seuil fixé sur la fonction de coût, quel que soit le multiplicateur, ce qui est regrettable puisque c'est le cas le moins coûteux mais également celui où l'erreur classique est la plus grande, même si, dans l'étude à budget fixé, 70 initialisations sur 80 font mieux que les éléments finis classiques

avec le multiplicateur retenu. L'entraînement dépend de l'initialisation, et la meilleure initialisation est choisie à l'aide de la solution de référence, indisponible en pratique (les erreurs médianes sur les initialisations qui atteignent le seuil, critère qui n'utilise pas la référence, sont néanmoins de plusieurs ordres de grandeur inférieures à l'erreur classique). La comparaison se fait à $N_{\text{dofs}}$ égal, sur maillages uniformes, alors que pour une couche limite le concurrent classique naturel est un maillage raffiné de façon adaptée [2] ou un raffinement *hp* [9] : les gains sont donc des majorants du bénéfice par rapport à la meilleure méthode classique. Enfin, le coût de l'entraînement n'est pas pris en compte. Il revient à remplacer une résolution classique sur un maillage fin par un entraînement sur un maillage plus grossier, dont chaque époque coûte l'ordre de deux résolutions du schéma grossier avec un optimiseur de gradient, davantage avec ENG. L'enrichissement ne peut donc être rentable en coût que sur de gros problèmes, ce qui n'est pas le cas en 1D, et motive le cas 3D.

## 6.4 Cas non paramétrique en 3D

Nous étendons l'étude au problème d'advection-diffusion sur le cube unité, avec $\mathrm{Pe} = 20$ et un champ de transport porté par la diagonale du cube, pour $\boldsymbol{\mu}^{(1)} = (0{,}8; 0{,}8)$. La solution manufacturée $u(\boldsymbol{x}) = \prod_{i=1}^{3} x_i\big(1 - e^{-\mathrm{Pe}(\mu_1+\mu_2)(1-x_i)/\sqrt{3}}\big)$ présente une couche limite le long des trois faces $x_i = 1$, la plus raide près du coin $(1, 1, 1)$. Nous employons des éléments finis $\mathbb{Q}_2$ sur un maillage de $4 \times 4 \times 4$ cellules ($N_{\text{dofs}} = 729$), avec un unique réseau de 3 couches cachées de largeur 10 ($N_\theta = 271$ poids) et le multiplicateur $m_\theta = \exp(6 \tanh v_\theta)$, dont l'amplitude est plus grande qu'en 1D (réglage trouvé par essais préliminaires, avec une borne sur les poids de 16 au lieu de 4). L'entraînement dure 70 époques, avec 6000 points de collocation redessinés à chaque époque, pour 3 initialisations. Les éléments finis classiques sont calculés sur des maillages de $2^3$ à $40^3$ cellules (jusqu'à environ $5 \cdot 10^5$ degrés de liberté), avec le même solveur.

Pour $N_{\text{dofs}} = 729$, l'erreur des trois initialisations est inférieure à $5 \cdot 10^{-4}$, contre environ 0,15 pour les éléments finis classiques, qui ne résolvent pas la couche de coin, beaucoup plus fine qu'une cellule : le gain est de deux à trois ordres de grandeur. À erreur égale, les éléments finis classiques exigeraient de l'ordre de $4 \cdot 10^5$ à $3 \cdot 10^6$ degrés de liberté, soit de quelques centaines à quelques milliers de fois plus que la base apprise (figure 5). Le multiplicateur appris croît de façon monotone le long de la diagonale, sur plus de trois ordres de grandeur, avec un maximum près du coin $(1, 1, 1)$.

La comparaison à temps de calcul comparable est plus nuancée. À un temps total d'environ une minute (construction du maillage, entraînement et résolution finale, compilation comprise), la meilleure initialisation atteint $5{,}7 \cdot 10^{-5}$ avec 729 degrés de liberté en 59 s, contre $3{,}5 \cdot 10^{-4}$ avec 531 441 degrés de liberté en 51 s pour les éléments finis classiques sur le maillage le plus fin que la mémoire permet de calculer : l'erreur est environ 6 fois plus faible. Le gain à temps égal est donc bien plus modeste que celui obtenu à $N_{\text{dofs}}$ fixé et dépend de l'initialisation, mais la base apprise garde plusieurs centaines de fois moins de degrés de liberté, sans compter les 271 poids du réseau. Hors compilation, la base apprise demande de 6,8 à 10,1 s contre 26,6 à 49,5 s pour les éléments finis classiques sur les maillages les plus fins, le temps de ces derniers étant

dominé par la résolution. Les limites sont voisines de celles du cas 1D : une seule valeur de $\boldsymbol{\mu}$, un seul maillage, trois initialisations et un seul degré, avec des maillages uniformes comme concurrent.

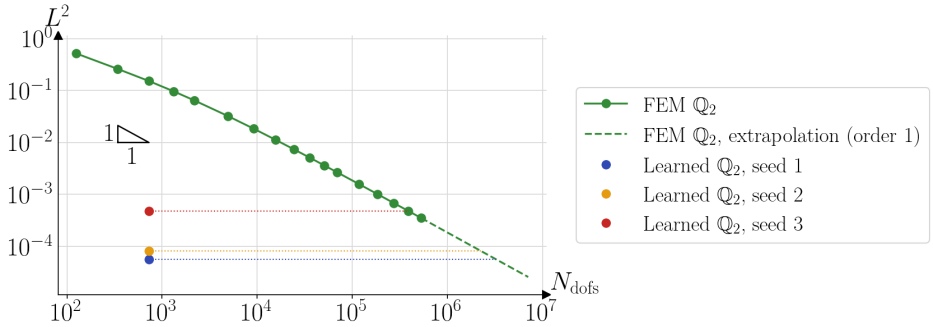



FIGURE 5 – Advection-diffusion 3D avec $\boldsymbol{\mu}^{(1)}$. Erreur relative $L^2$ en fonction de $N_{\text{dofs}}$ pour les éléments finis $\mathbb{Q}_2$ classiques (trait plein), son extrapolation d'ordre 1 (tirets) et les trois bases apprises sur le maillage $4 \times 4 \times 4$. Une ligne pointillée relie l'erreur de chaque base apprise au nombre de degrés de liberté auquel l'erreur classique l'atteint.

## 6.5 Cas paramétrique en 1D : bases apprises par un opérateur neuronal

Nous passons au cadre paramétrique du problème de la section 3 : un seul réseau est entraîné une fois sur toute une famille de valeurs de $\boldsymbol{\mu} \in \mathcal{M}$, puis la base apprise est utilisée, sans nouvel entraînement, pour résoudre le problème avec des valeurs de $\boldsymbol{\mu}$ jamais vues. Nous nous restreignons au schéma DG avec flux SIPG et à des bases *patchwise*, et nous comparons trois façons de prendre en compte le paramètre :

- la base *partagée*, $\phi^{\theta}_{c,i} = m_\theta(x)\,\phi_{c,i}$, identique pour toutes les valeurs de $\boldsymbol{\mu}$ ;
- la base *dépendante de* $\boldsymbol{\mu}$, $\phi^{\theta}_{c,i} = m_\theta(x, \boldsymbol{\mu})\,\phi_{c,i}$, dont le réseau reçoit aussi $\boldsymbol{\mu}$ en entrée ;
- la base *dépendante du champ*, $\phi^{\theta}_{c,i} = m_\theta(x, \boldsymbol{z})\,\phi_{c,i}$ avec $\boldsymbol{z} = \mathcal{E}_\theta\big(C(\cdot, \boldsymbol{\mu})\big)$, où le champ de transport $C(\cdot, \boldsymbol{\mu})$ de l'EDP est résumé par un second réseau, l'encodeur $\mathcal{E}_\theta$, entraîné conjointement.

Les deux dernières relèvent de l'apprentissage d'opérateurs neuronaux, à ceci près que la sortie de l'opérateur est ici la base, et la solution $u_{h,\theta}(\cdot\,; \boldsymbol{\mu})$ est obtenue en résolvant le schéma DG dans cette base, de sorte que c'est le solveur qui porte la physique. Il n'existe pas de base de référence, et l'entraînement est donc supervisé à travers le solveur, par la fonction de coût

$$\mathcal{J}(\theta) = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \left| u_{h,\theta}\big(x^{(i)}; \boldsymbol{\mu}^{(i)}\big) - u^{(i)}_{\text{ref}} \right|^2,$$

où $u^{(i)}_{\text{ref}}$ est la valeur au point $x^{(i)}$ de la solution de référence pour le paramètre $\boldsymbol{\mu}^{(i)}$, calculée numériquement (DG, $\mathbb{Q}_4$, $N_{\text{cells}} = 128$). Le réseau est entraîné sur 32 valeurs de

$\boldsymbol{\mu}$ avec l'optimiseur SS-BFGS [49], 400 époques et 10 initialisations. Chaque époque porte sur un lot de 16 valeurs de $\boldsymbol{\mu}$, chacune évaluée en 101 points de $\Omega$, soit $N_{\text{data}} = 16 \times 101$ données dans la fonction de coût. Le réseau est testé sur des valeurs tirées dans $\mathcal{M}$ et distinctes de celles de l'entraînement.

**Transport linéaire.** Le premier problème est $-\frac{1}{\text{Pe}}u'' + (\mu_1 + \mu_2 x)u' = 1$ avec $\text{Pe} = 20$ et $\boldsymbol{\mu} \in \mathcal{M} = [0{,}8; 1{,}2]^2$, testé sur 500 valeurs, avec le multiplicateur $\exp(2 \tanh v_\theta)$. La couche limite en $x = 1$ reste à la même place : le paramètre n'en change que l'amplitude. Les trois bases ont respectivement 321, 353 et 1 381 poids (dont 996 pour l'encodeur). Le tableau 3 donne les gains médians sur le jeu de test selon la configuration $(N_{\text{cells}}, k)$. La base partagée fait déjà mieux que la base classique pour tous les paramètres de test, avec des gains médians de 14 à 167. La base dépendante de $\boldsymbol{\mu}$ est bien plus précise, avec des gains médians de l'ordre de $10^2$ pour $k = 1$ et de plusieurs milliers pour $k \geq 2$ sur les maillages de 8 et 12 cellules (jusqu'à 7 143 pour $k = 3$ et 8 cellules), pour seulement 32 poids de plus. Pour tous les degrés $k \geq 2$, son gain minimal dépasse même le gain maximal de la base partagée. La base dépendante du champ donne des résultats du même ordre que la précédente : pour ce champ affine, l'encodeur retrouve l'information du paramètre à partir du seul champ, mais ne fait pas mieux que le paramètre lui-même, avec environ quatre fois plus de poids.

**Point tournant.** Le second problème est $-\frac{1}{\text{Pe}}u'' + (\mu - x)u' = 1$ avec $\text{Pe} = 1000$ et $\mu \in \mathcal{M} = [0{,}3; 0{,}7]$, testé sur 32 valeurs, avec le multiplicateur $\exp(4 \tanh v_\theta)$ (le pic, plus fin qu'une cellule, impose une plus grande amplitude). Le champ change de signe en $x = \mu$ : les deux bords sont des bords d'entrée et la solution présente un pic étroit en $x = \mu$, autrement dit le paramètre déplace le pic. La base partagée, qui doit convenir à toutes les positions du pic, n'apporte presque rien : son gain médian est compris entre 1,0 et 1,5 pour la plupart des configurations, son gain minimal descend à 0,4 (la solution apprise est alors plus de deux fois moins précise que la solution classique) et ses gains ne dépassent 4 que sur le maillage de 4 cellules avec $k \in \{1, 2\}$, où l'erreur classique est la plus grande. La base dépendante de $\mu$ (337 poids) fait mieux que la base classique pour tous les paramètres de test, avec des gains médians d'au moins 9 et allant jusqu'à 211 (tableau 3). La dépendance de la base vis-à-vis du paramètre semble donc indispensable quand la structure de la solution se déplace avec lui.

**Limites.** L'étude se limite à des problèmes 1D, au schéma DG avec une base *patchwise* et à un espace de paramètres de dimension au plus 2, avec 32 valeurs d'entraînement, sans test d'extrapolation hors de $\mathcal{M}$. Les entraînements dépendent de l'initialisation et d'un budget de 400 époques, et seule la meilleure initialisation est rapportée. Certaines configurations (point tournant avec $k = 3$) semblent limitées par l'optimisation plus que par la base. La comparaison se fait à $N_{\text{dofs}}$ égal, sans compter le coût de l'entraînement, qui fait maintenant intervenir la résolution du schéma pour tout un lot de paramètres à chaque époque. Ces résultats ne sont pas directement comparables à ceux du cas non paramétrique, puisque la fonction de coût (supervisée), l'optimiseur (SS-BFGS au lieu d'ENG) et le nombre d'initialisations changent en même temps.

| Problème | Base | $N_\theta$ | Gain médian (min.–max.) |
|---|---|---|---|
| Transport linéaire | partagée | 321 | 14–167 |
| | dépendante de $\boldsymbol{\mu}$ | 353 | 116–7 143 |
| | dépendante du champ | 1 381 | 47–6 379 |
| Point tournant | partagée | 321 | 1,0–4,4 |
| | dépendante de $\mu$ | 337 | 9,4–211 |

TABLEAU 3 – Gains médians sur le jeu de test pour la meilleure initialisation, DG $\mathbb{Q}_k$ ($k \in \{1, 2, 3\}$) et $N_{\text{cells}} \in \{4, 8, 12\}$ (neuf configurations), par rapport au DG classique à même $N_{\text{dofs}}$. L'intervalle donne le plus petit et le plus grand gain médian parmi les configurations.

## 6.6 Bilan

Les bases de Galerkin enrichies par un réseau réduisent l'erreur $L^2$ de plusieurs ordres de grandeur à $N_{\text{dofs}}$ fixé en 1D et, en 3D, donnent des erreurs du même ordre ou jusqu'à 6 fois plus faibles à temps de calcul comparable avec des centaines de fois moins de degrés de liberté. En contexte paramétrique, un unique réseau entraîné une fois améliore la solution classique pour des paramètres jamais vus, à condition que la base dépende du paramètre lorsque le pic de la solution se déplace. Ces résultats restent exploratoires : peu de cas tests, entraînement sensible à l'initialisation, paramétrisation du multiplicateur choisie à la main, aucune initialisation sous le seuil pour $k = 1$ en éléments finis non paramétrique, comparaison principalement à $N_{\text{dofs}}$ égal et sur maillages uniformes.

# 7 Conclusion et perspectives

*(voir manuscrit, chapitre 6)*

Cette thèse porte sur la résolution d'EDP paramétriques, en particulier elliptiques, qui doivent être résolues un grand nombre de fois pour différentes valeurs de paramètres. L'introduction a identifié trois limites des méthodes haute fidélité classiques : la difficulté de construire un maillage conforme à une géométrie complexe, le coût prohibitif de résolutions complètes répétées dès que le problème dépend de paramètres, et l'expressivité limitée des espaces d'approximation polynomiaux, qui exigent un maillage très fin pour représenter des structures comme les couches limites. Les méthodes neuronales répondent en partie à ces limites, mais n'offrent ni garanties théoriques ni estimateurs d'erreur fiables, et leur entraînement peut être très sensible aux hyperparamètres. Plutôt que d'opposer ces deux familles de méthodes, nous avons choisi de les hybrider en conservant la structure des méthodes de Galerkin, de deux façons complémentaires.

## 7.1 Bilan des contributions

**Première contribution.** Elle enrichit l'espace éléments finis de Lagrange continus par un a priori neuronal fixé, un PINN paramétrique entraîné une seule fois hors ligne pour toute la famille de paramètres, puis injecté en ligne de façon additive ou multiplicative. Nous avons démontré que, dans les deux cas, les ordres de convergence classiques sont conservés et que le gain dépend explicitement de la qualité de l'a priori, en particulier de la précision de ses dérivées. Les expériences en 1D, 2D et 3D valident ces résultats : à précision donnée et en moyenne sur un ensemble de paramètres, l'approche additive avec des éléments $\mathbb{P}_1$ permet d'utiliser des maillages de 5 à 28 fois plus grossiers que la méthode standard, selon les cas tests. En 3D, pour atteindre une erreur relative $L^2$ de $10^{-3}$, elle devient plus avantageuse, entraînement du réseau compris, dès que 19 jeux de paramètres doivent être résolus. La comparaison des deux enrichissements montre, théoriquement, que le gain de l'approche multiplicative tend vers celui de l'approche additive quand son paramètre de rehaussement tend vers l'infini et, numériquement, qu'elle ne fait mieux que l'approche additive que dans des cas très particuliers, pour un bon choix de ce paramètre.

**Seconde contribution.** Elle rend à l'inverse le solveur lui-même différentiable, donc apprenable. Nous avons développé, dans la version `JAX` de `ScimBa`, un schéma de Galerkin discontinu différentiable, étendu aux éléments finis continus et à $\varphi$-FEM, et le formalisme d'apprentissage associé (différentiation implicite et méthode adjointe), qui permet d'entraîner un réseau à travers le solveur. Nous l'avons d'abord utilisé contre le manque d'expressivité des espaces polynomiaux, en enrichissant les bases de Galerkin par un réseau. Pour une valeur fixée du paramètre, l'erreur $L^2$ est réduite de plusieurs ordres de grandeur en 1D à $N_{\text{dofs}}$ fixé. En 3D, où la résolution classique devient coûteuse, nous obtenons des erreurs du même ordre à temps de calcul comparable, avec beaucoup moins de degrés de liberté. Nous l'avons ensuite utilisé contre le coût des résolutions répétées : un unique réseau, entraîné sur des solutions de référence à la manière d'un opérateur neuronal, construit la base enrichie pour toute une famille de paramètres et s'applique sans nouvel entraînement à des valeurs jamais vues. En 1D, la base dépendant du paramètre améliore la solution classique pour tous les paramètres de test, avec des gains médians de l'ordre de $10^2$ à plusieurs milliers pour le transport linéaire et de 9 à 211 pour le point tournant, où la structure de la solution se déplace avec le paramètre.

**Comparaison.** Les deux contributions apportent des réponses différentes aux limites initiales. Toutes deux enrichissent l'espace d'approximation pour pallier son manque d'expressivité, et toutes deux visent à réduire le coût d'une résolution répétée pour de nombreux paramètres, ce qui est établi numériquement pour la première et encore préliminaire mais prometteur pour la seconde. Elles diffèrent surtout par le compromis qu'elles offrent entre garanties et expressivité. Dans la première, l'a priori est entraîné puis figé et l'erreur du schéma enrichi est contrôlée a priori, mais le gain reste borné par la qualité de l'a priori. Dans la seconde, la base est apprise à travers le solveur et s'adapte à la solution du schéma au lieu de dépendre d'un a priori figé, mais sans estimation d'erreur, et chaque itération d'entraînement exige de résoudre le schéma.

## 7.2 Limites

Pour la première contribution, le gain se dégrade quand la précision de l'a priori diminue, comme le montre le cas de Navier–Stokes à nombre de Rayleigh élevé, la théorie reste limitée aux problèmes elliptiques linéaires scalaires, et l'approche multiplicative n'est testée qu'en 1D.

Pour la seconde, les résultats restent exploratoires et constituent une étude de faisabilité plutôt qu'un solveur compétitif. Ils reposent sur peu de cas tests, l'entraînement est sensible à l'initialisation, la paramétrisation du multiplicateur est fixée à la main, et aucune initialisation n'atteint le seuil pour $k = 1$ dans l'étude 1D non paramétrique avec des éléments finis continus. La comparaison avec les méthodes classiques se fait principalement à nombre de degrés de liberté égal et sur des maillages uniformes. Le coût de l'entraînement n'est discuté que qualitativement en 1D et ignoré dans le cas paramétrique, et une comparaison à temps de calcul comparable n'est faite qu'en 3D, pour une seule valeur du paramètre.

## 7.3 Perspectives

Pour la première contribution, plusieurs pistes se dégagent : améliorer l'a priori (opérateurs neuronaux, pénalisation des dérivées dans la fonction de coût d'entraînement), puis étendre la méthode aux problèmes instationnaires, non linéaires et aux géométries complexes, par exemple en combinant l'a priori neuronal avec $\varphi$-FEM.

Pour la seconde, il reste à comparer plus systématiquement les bases apprises aux méthodes classiques à coût de calcul égal, entraînement compris, et à des maillages adaptés ou progressifs, et à étudier davantage de cas tests, en particulier en 2D dans le cadre paramétrique. Le moyen le plus direct de retrouver, au moins en partie, des garanties sur l'erreur de la base apprise est de figer le multiplicateur une fois appris. La base enrichie coïncide alors avec celle de l'approche multiplicative de la première contribution, de sorte que son analyse a priori s'applique et donne une estimation de type Céa, dont le gain dépend des dérivées de la solution et du multiplicateur. C'est la perspective qui relie le plus directement les deux contributions.

Du côté de la bibliothèque, des extensions aux problèmes instationnaires et aux maillages structurés sont en cours au sein de l'équipe MACARON, et l'optimisation de forme avec $\varphi$-FEM, où la fonction level-set devient elle-même un objet appris, est un cas d'usage naturel de ce cadre différentiable.

Plus généralement, ce travail montre qu'injecter de l'information neuronale dans la structure des méthodes de Galerkin permet de tirer parti de l'expressivité des réseaux tout en conservant, pour la première contribution, les garanties qui font la force des méthodes classiques, et en cherchant à en retrouver une partie pour la seconde.

# 8 Publications et communications

## Publication

Une publication a été rédigée et publiée au cours de la thèse. Elle décrit le pipeline de résolution ainsi que les résultats théoriques et numériques de la première contribution (section 4).

— Hélène Barucq, Michel Duprez, Florian Faucher, Emmanuel Franck, Frédérique Lecourtier, Vanessa Lleras, Victor Michel-Dansac, Nicolas Victorion. *Enriching continuous Lagrange finite element approximation spaces using neural networks.* ESAIM : Mathematical Modelling and Numerical Analysis (M2AN), vol. 60, n° 2, p. 541–603, 2026.

Les travaux de la seconde contribution (sections 5 et 6), menés en collaboration avec l'équipe MACARON qui développe la bibliothèque `ScimBa`, n'ont pas encore été publiés. Un article est en préparation, qui présente l'enrichissement des bases de Galerkin par un réseau et l'entraînement de ce réseau à travers le solveur par différentiation implicite.

## Présentations orales

— NeuralFEM Workshop 2026, IRMA, Strasbourg, 14 décembre 2026 : *Hybrid finite element and neural network methods for parametric PDEs.*
— Séminaire EDP, IRMA, Strasbourg, 24 novembre 2026 : *Hybrid finite element and neural network methods for parametric PDEs.*
— Séminaire MACS, IMAG, Montpellier, 10 novembre 2026 : *Hybrid finite element and neural network methods for parametric PDEs.*
— SciCADE 2026, Édimbourg, 3 juillet 2026 : *Enriching continuous Lagrange finite element approximation spaces using neural networks.*
— CANUM 2026, Saint-Jacut-de-la-Mer, 4 juin 2026 : *Enriching continuous Lagrange finite element approximation spaces using neural networks.*
— Séminaire des doctorants, Montpellier, 17 décembre 2025 : *Enriching continuous Lagrange finite element approximation spaces using neural networks.*
— ICOSAHOM 2025, Montréal, 15 juillet 2025 : *Enriching continuous Lagrange finite element approximation spaces using neural networks.*
— DTE & AICOMAS 2025, Paris, 20 février 2025 : *Combining Finite Element Methods and Neural Networks to Solve Elliptic Problems on 2D Geometries.*
— Projet Exama, réunion du WP2, 26 mars 2024 : *How to work with complex geometries in PINNs ?*
— Retraite MACARON/TONUS, 6 février 2024 : *Mesh-based methods and physically informed learning.*
— Réunion d'équipe MIMESIS, 12 décembre 2023 : *Development of hybrid finite element/neural network methods to help create digital surgical twins.*

### Posters

— EMS-TAG-SciML 2025, Milan, 24 mars 2025 : *Enriching continuous Lagrange finite element approximation spaces using neural networks.*
— CJC-MA 2024, Lyon, 29 octobre 2024 : *Combining Finite Element methods and Neural Networks to solve elliptic problem on complex 2D geometries.*
— Journée poster MSII, Strasbourg, 24 octobre 2024 : *Combining Finite Element methods and Neural Networks to solve elliptic problem on complex 2D geometries.*
— SciML 2024, Strasbourg, 8 juillet 2024 : *Combining Finite Element methods and Neural Networks to solve elliptic problem on complex 2D geometries.*

## Références


[1] J. Aghili, E. Franck, R. Hild, V. Michel-Dansac, and V. Vigon. « Accelerating the convergence of Newton's method for nonlinear elliptic PDEs using Fourier neural operators ». In: *Communications in Nonlinear Science and Numerical Simulation* 140 (2025), p. 108434.

[2] M. Ainsworth and J. T. Oden. « A posteriori error estimation in finite element analysis ». In: *Computer Methods in Applied Mechanics and Engineering* 142.1–2 (1997), pp. 1–88.

[3] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. « Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems ». In: *SIAM Journal on Numerical Analysis* 39.5 (2002), pp. 1749–1779.

[4] B. Ayuso and L. D. Marini. « Discontinuous Galerkin Methods for Advection-Diffusion-Reaction Problems ». In: *SIAM Journal on Numerical Analysis* 47.2 (2009), pp. 1391–1420.

[5] I. Babuška and J. M. Melenk. « The partition of unity method ». In: *International Journal for Numerical Methods in Engineering* 40.4 (1997), pp. 727–758.

[6] S. Badia, W. Li, and A. F. Martín. « Finite element interpolated neural networks for solving forward and inverse problems ». In: *Computer Methods in Applied Mechanics and Engineering* 418 (2024), p. 116505.

[7] A. R. Barron. « Universal approximation bounds for superpositions of a sigmoidal function ». In: *IEEE Transactions on Information Theory* 39.3 (1993), pp. 930–945.

[8] H. Barucq, M. Duprez, F. Faucher, E. Franck, F. Lecourtier, V. Lleras, V. Michel-Dansac, and N. Victorion. « Enriching continuous Lagrange finite element approximation spaces using neural networks ». In: *ESAIM: Mathematical Modelling and Numerical Analysis* 60.2 (2026), pp. 541–603.

[9] C. E. Baumann and J. T. Oden. « A discontinuous hp finite element method for convection-diffusion problems ». In: *Computer Methods in Applied Mechanics and Engineering* 175.3–4 (1999), pp. 311–341.

[10] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. « Automatic Differentiation in Machine Learning: a Survey ». In: *Journal of Machine Learning Research* 18.153 (2018), pp. 1–43.

[11] G. Berkooz, P. Holmes, and J. L. Lumley. « The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows ». In: *Annual Review of Fluid Mechanics* 25.1 (1993), pp. 539–575.

[12] M. Blondel and V. Roulet. *The Elements of Differentiable Programming*. 2024. arXiv: 2403.14606.

[13] J. Bradbury, R. Frostig, P. Hawkins, et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: http://github.com/jax-ml/jax.

[14] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods*. 3rd ed. Vol. 15. Texts in Applied Mathematics. Springer New York, 2008.

[15] H.-J. Bungartz and M. Griebel. « Sparse grids ». In: *Acta Numerica* 13 (2004), pp. 147–269.

[16] E. Burman and P. Hansbo. « Fictitious domain finite element methods using cut elements: I. A stabilized Lagrange multiplier method ». In: *Computer Methods in Applied Mechanics and Engineering* 199.41–44 (2010), pp. 2680–2686.

[17] C. Canuto, M. Hussaini, A. Quarteroni, and T. Zang. *Spectral Methods: Fundamentals in Single Domains*. Springer Berlin Heidelberg, 2006.

[18] F. Chinesta, P. Ladeveze, and E. Cueto. « A Short Review on Model Order Reduction Based on Proper Generalized Decomposition ». In: *Archives of Computational Methods in Engineering* 18.4 (2011), pp. 395–404.

[19] G. Cybenko. « Approximation by superpositions of a sigmoidal function ». In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314.

[20] M. Duprez and A. Lozinski. « $\phi$-FEM: A Finite Element Method on Domains Defined by Level-Sets ». In: *SIAM Journal on Numerical Analysis* 58.2 (2020), pp. 1008–1028.

[21] W. E and B. Yu. « The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems ». In: *Communications in Mathematics and Statistics* 6.1 (2018), pp. 1–12.

[22] A. Ern and J.-L. Guermond. *Theory and Practice of Finite Elements*. Springer New York, 2004.

[23] R. Eymard, T. Gallouët, and R. Herbin. « Finite volume methods ». In: *Solution of Equation in $\mathbb{R}^n$ (Part 3), Techniques of Scientific Computing (Part 3)*. Vol. 7. Handbook of Numerical Analysis. Elsevier, 2000, pp. 713–1018.

[24] X. Feng, H. Shangguan, T. Tang, X. Wan, and T. Zhou. *A hybrid FEM-PINN method for time-dependent partial differential equations*. 2024. arXiv: 2409.02810.

[25] E. Franck, V. Michel-Dansac, and L. Navoret. « Approximately well-balanced Discontinuous Galerkin methods using bases enriched with Physics-Informed Neural Networks ». In: *Journal of Computational Physics* 512 (2024), p. 113144.

[26] T. G. Grossmann, U. J. Komorowska, J. Latz, and C.-B. Schönlieb. « Can physics-informed neural networks beat the finite element method? » In: *IMA Journal of Applied Mathematics* 89.1 (2024), pp. 143–174.

[27] Z. Hu, K. Shukla, G. E. Karniadakis, and K. Kawaguchi. « Tackling the curse of dimensionality with physics-informed neural networks ». In: *Neural Networks* 176 (2024), p. 106369.

[28] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. « Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement ». In: *Computer Methods in Applied Mechanics and Engineering* 194.39–41 (2005), pp. 4135–4195.

[29] J. Jung and E. Constantinescu. *Learning differentiable weak-form corrections to accelerate finite element simulations*. 2026. arXiv: 2601.20019.

[30] S. Kang and E. M. Constantinescu. *Differentiable DG with neural operator source term correction*. 2023. arXiv: 2310.18897.

[31] E. Kharazmi, Z. Zhang, and G. E. M. Karniadakis. « *hp*-VPINNs: Variational physics-informed neural networks with domain decomposition ». In: *Computer Methods in Applied Mechanics and Engineering* 374 (2021), p. 113547.

[32] I. E. Lagaris, A. Likas, and D. I. Fotiadis. « Artificial neural networks for solving ordinary and partial differential equations ». In: *IEEE Transactions on Neural Networks* 9.5 (1998), pp. 987–1000.

[33] M. G. Larson and A. J. Niklasson. « Analysis of a family of discontinuous Galerkin methods for elliptic problems: the one dimensional case ». In: *Numerische Mathematik* 99.1 (2004), pp. 113–130.

[34] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007.

[35] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. « Fourier Neural Operator for Parametric Partial Differential Equations ». In: *International Conference on Learning Representations (ICLR)*. 2021.

[36] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar. « Physics-informed neural operator for learning partial differential equations ». In: *ACM/IMS Journal of Data Science* 1.3 (2024), pp. 1–27.

[37] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. « Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators ». In: *Nature Machine Intelligence* 3.3 (2021), pp. 218–229.

[38] N. Margenberg, R. Jendersie, C. Lessig, and T. Richter. « DNN-MG: A hybrid neural network/finite element method with applications to 3D simulations of the Navier-Stokes equations ». In: *Computer Methods in Applied Mechanics and Engineering* 420 (2024), p. 116692.

[39] J. Müller and M. Zeinhofer. « Achieving High Accuracy with PINNs via Energy Natural Gradient Descent ». In: *Proceedings of the 40th International Conference on Machine Learning (ICML)*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 25471–25485.

[40] A. Odot, R. Haferssas, and S. Cotin. « DeepPhysics: A physics aware deep learning framework for real-time simulation ». In: *International Journal for Numerical Methods in Engineering* 123.10 (2022), pp. 2381–2398.

[41] A. T. Patera and E. M. Rønquist. « Reduced basis approximation and *a posteriori* error estimation for a Boltzmann model ». In: *Computer Methods in Applied Mechanics and Engineering* 196.29–30 (2007), pp. 2925–2942.

[42] C. S. Peskin. « Flow patterns around heart valves: a numerical method ». In: *Journal of Computational Physics* 10.2 (1972), pp. 252–271.

[43] M. Raissi, P. Perdikaris, and G. E. Karniadakis. « Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations ». In: *Journal of Computational Physics* 378 (2019), pp. 686–707.

[44] Y. Saad. *Iterative Methods for Sparse Linear Systems*. 2nd ed. Society for Industrial and Applied Mathematics, 2003.

[45] K. Škardová, A. Daby-Seesaram, and M. Genet. « Finite element neural network interpolation: Part I—interpretable and adaptive discretization for solving PDEs ». In: *Computational Mechanics* 77.2 (2026), pp. 547–567.

[46] H. Son, J. W. Jang, W. J. Han, and H. J. Hwang. « Sobolev Training for Physics Informed Neural Networks ». In: *Communications in Mathematical Sciences* 21.6 (2023), pp. 1679–1705.

[47] T. Strouboulis, K. Copps, and I. Babuška. « The generalized finite element method ». In: *Computer Methods in Applied Mechanics and Engineering* 190.32–33 (2001), pp. 4081–4193.

[48] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. « Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains ». In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. Curran Associates, Inc., 2020, pp. 7537–7547.

[49] J. F. Urbán, P. Stefanou, and J. A. Pons. « Unveiling the optimization process of physics informed neural networks: How accurate and competitive can PINNs be? » In: *Journal of Computational Physics* 523 (2025), p. 113656.

[50] D. Wang, H. Li, and Q. Zhang. « General enrichments of stable GFEM for interface problems: Theory and extreme learning machine construction ». In: *Applied Numerical Mathematics* 214 (2025), pp. 143–159.

[51] W. Xiong, X. Long, S. P. A. Bordas, and C. Jiang. « The deep finite element method: A deep learning framework integrating the physics-informed neural networks with the finite element method ». In: *Computer Methods in Applied Mechanics and Engineering* 436 (2025), p. 117681.

[52] T. Xue, S. Liao, Z. Gan, C. Park, X. Xie, W. K. Liu, and J. Cao. « JAX-FEM: A differentiable GPU-accelerated 3D finite element solver for automatic inverse design and mechanistic data science ». In: *Computer Physics Communications* 291 (2023), p. 108802.

[53] D. Yarotsky. « Error bounds for approximations with deep ReLU networks ». In: *Neural Networks* 94 (2017), pp. 103–114.