Méthode de quadrature pour les PINNs fondée théoriquement sur la hessienne des résiduels

Antoine CARADOT Rémi EMONET Amaury HABRARD Abdel-Rahim MEZIDI Marc SEBBAN
Laboratoire Hubert Curien, UMR CNRS 5516, Inria, Bâtiment F 18 Rue du Professeur Benoît Lauras, 42000 Saint-Étienne, France

Résumé – Les réseaux de neurones informés par la physique (PINNs) sont apparus comme un moyen efficace d'apprendre des solveurs d'EDPs en incorporant le modèle physique dans la fonction de perte et en minimisant ses résiduels par différenciation automatique à des points dits de collocation. Originellement sélectionnés de manière uniforme, le choix de ces derniers a fait l'objet d'avancées récentes en échantillonnage adaptatif. Nous proposons ici une nouvelle méthode de quadrature pour l'approximation d'intégrale basée sur la hessienne de la fonction considérée, pour laquelle nous dérivons une borne d'erreur d'approximation. Nous exploitons cette information de second ordre pour guider la sélection des points de collocation durant l'apprentissage des PINNs.

Abstract – Physics-informed Neural Networks (PINNs) have emerged as an efficient way to learn surrogate neural solvers of PDEs by embedding the physical model in the loss function and minimizing its residuals using automatic differentiation at so-called collocation points. Originally uniformly sampled, the choice of the latter has been the subject of recent advances leading to adaptive sampling refinements. In this paper, we propose a new quadrature method for approximating definite integrals based on the hessian of the considered function that we leverage to guide the selection of the collocation points during the training process of PINNs.

1 Introduction

Malgré des avancées scientifiques importantes en simulation numérique, résoudre efficacement des EDPs demeure un problème complexe et coûteux. En incorporant l'équation dans la fonction de perte et en minimisant ses résiduels en des points de *collocation*, les réseaux de neurones informés par la physique (PINNs) [6] sont apparus comme une solution séduisante pour apprendre efficacement des solveurs neuronaux. Malgré leur efficacité, les PINNs sont encore mal compris, et il est crucial d'étudier leurs fondements théoriques ainsi que leurs propriétés algorithmiques afin d'avoir une bonne compréhension de leurs capacités et limites. En effet, des études récentes ont montré que les PINNs peuvent être sujets à des comportements pathologiques, conduisant à des résiduels nuls, donc plausibles du point de vue physique, mais correspondant à des solutions incorrectes [1, 3]. La caractérisation de ces « modes d'échec » a mené à une recherche active abordant la question sous deux angles différents : un premier théorique visant à établir dans un contexte d'échantillonnage uniforme des points de collocation (e.g., grille uniforme équidistante ou tirage aléatoire uniforme), des garanties de consistance et de convergence sous forme de bornes d'estimation, ou d'approximation (e.g., [3, 4]) ; un deuxième améliorant l'échantillonnage des points de collocation. Plutôt que de les tirer de manière uniforme, plusieurs stratégies ont émergé dans la littérature, suggérant d'orienter la sélection des points au cours de l'apprentissage en fonction de l'amplitude ou du gradient des résiduels de l'EDP. Ceci a donné lieu à une nouvelle famille de méthodes d'échantillonnage adaptatif pour les PINNs (e.g., [2, 7, 8, 9]). Cependant, il convient de noter que, bien que ces méthodes aient démontré des performances remarquables en pratique, elles ont en commun l'absence de garanties théoriques quant à leur avantage par rapport à un échantillonnage uniforme.
L'objectif de ce papier est de répondre à cette limitation. Partant du fait que la minimisation d'une perte empirique en apprentissage peut être abordée d'un point de vue mathématique à travers l'approximation de l'intégrale d'une fonction f, nous proposons une nouvelle règle de quadrature simple basée sur la hessienne de f. Nous dérivons une borne supérieure sur l'erreur d'approximation et montrons sa meilleure précision par rapport à celle issue d'une grille uniforme régulière. Ce résultat théorique nous conduit à concevoir une méthode d'échantillonnage adaptatif pour les PINNs, où f prend la forme de la fonction de perte basée sur les résiduels. Cette stratégie sélectionne les points de collocation dans le domaine spatio-temporel aux endroits où la hessienne varie le plus. Les expériences menées sur deux EDPs en 2D mettent en évidence les propriétés intéressantes de notre méthode.

2 Notations

On considère les EDPs de la forme $\frac{\partial u}{\partial t} + \mathcal{N}[u] = 0$, où $\mathcal{N}[\cdot]$ est un opérateur différentiel en temps et en espace, et où $u(t, \mathbf{x})$ est la solution avec $t \in [0, T]$ et $\mathbf{x} \in \Omega$. Cette équation est généralement accompagnée de conditions initiales et limites : $\forall \mathbf{x} \in \Omega$, $\mathcal{I}[u](0, \mathbf{x}) = 0$, et $\forall \mathbf{x} \in \partial\Omega, t \in [0, T]$, $\mathcal{B}[u](t, \mathbf{x}) = 0$, où $\mathcal{B}$ est l'opérateur de frontière appliqué à $\partial\Omega$ et $\mathcal{I}$ est l'opérateur initial à $t = 0$. L'objectif d'un PINN [6] est d'apprendre une approximation $u_\theta(t, \mathbf{x})$ de la solution $u(t, \mathbf{x})$ en optimisant les paramètres θ d'un réseau de neurones par la minimisation d'une fonction de perte $\mathcal{L}(\theta)$ (voir Problème 1) composée des termes non négatifs suivants :

$$\begin{aligned}
\mathcal{L}_{\mathcal{N}}(\theta) &= \int_{[0,T]\times\Omega} \left(\frac{\partial u_\theta}{\partial t} + \mathcal{N}[u_\theta] \right)^2 dt d\mathbf{x} \\
\mathcal{L}_{\mathcal{I}}(\theta) &= \int_{\Omega} (\mathcal{I}[u_\theta](0, \mathbf{x}))^2 d\mathbf{x} \\
\mathcal{L}_{\mathcal{B}}(\theta) &= \int_{[0,T]\times\partial\Omega} (\mathcal{B}[u_\theta](t, \mathbf{x}))^2 \, dt d\mathbf{x}
\end{aligned}$$

$$\min_\theta \mathcal{L}(\theta)$$
$$= \min_\theta(\mathcal{L}_\mathcal{N}(\theta) + \lambda_1\mathcal{L}_\mathcal{I}(\theta) + \lambda_2\mathcal{L}_\mathcal{B}(\theta) + \lambda_3 R(\theta)), \quad (1)$$

où $\lambda_1, \lambda_2, \lambda_3$ sont des hyperparamètres et $R(\theta)$ est un terme de régularisation. En pratique, les intégrales $\mathcal{L}_\mathcal{N}(\theta)$, $\mathcal{L}_\mathcal{I}(\theta)$, et $\mathcal{L}_\mathcal{B}(\theta)$ sont approximées par des espérances calculées à partir de $N_\mathcal{N}$ points de collocation, $N_\mathcal{I}$ points initiaux et $N_\mathcal{B}$ points aux bords, respectivement.
D'un point de vue mathématique, les intégrales de Eq. (1) s'obtiennent en approximant l'intégrale d'une fonction $f : \mathscr{D} \to \mathbb{R}$ à partir de N mesures de l'intégrande par une quadrature numérique comme suit : $\sum_{i=1}^N w_i f(\mathbf{x}_i) \approx \int_\mathscr{D} f(\mathbf{x})d\mathbf{x}$, où $w_i \geqslant 0$ sont des poids de quadrature.

Dans la section suivante, nous présentons une nouvelle méthode de quadrature reposant sur la hessienne de f et dérivons une borne supérieure sur l'erreur d'approximation, plus serrée que celle d'une méthode sélectionnant de manière régulière les $N = N_\mathcal{N} + N_\mathcal{I} + N_\mathcal{B}$ points de quadrature.

3 Bornes d'erreurs de quadrature

Soient $a, b \in \mathbb{R}$ et une fonction $f : [a, b] \longrightarrow \mathbb{R}$. Pour résoudre la quadrature de f, on choisit $x_0, \ldots, x_N \in [a, b]$ et les poids $w_0, \ldots, w_N$ peuvent être obtenus en approximant f par des fonctions polynomiales (méthode dite de Newton-Cotes). Afin d'éviter le phénomène de Runge, où l'interpolation polynomiale présente des piques aux bords de l'intervalle à cause de la croissance de $\max_{x\in[a,b]} |f^{(n)}(x)|$ comme une fonction de n, on propose ici de contrôler l'expressivité de l'approximation en utilisant une interpolation par de simples trapèzes.

3.1 Quadrature à partir d'une grille uniforme

Déterminons tout d'abord l'erreur qui serait obtenue par une règle basée sur des trapèzes construits à partir d'une grille uniforme de points de quadrature. On approxime tout d'abord f sur $[x_1, x_2]$, avec $x_1, x_2 \in [a, b]$, par la droite passant par $(x_1, f(x_1))$ et $(x_2, f(x_2))$. On pose $h = x_2 - x_1$. L'interpolation $p(x)$ est donnée par :

$$p(x) = \frac{x - x_1}{h} f(x_2) - \frac{x - x_2}{h} f(x_1).$$

Comme il s'agit d'un polynôme de degré 1, la formule d'interpolation de Lagrange implique que $\exists \xi \in [x_1, x_2]$ tel que

$$f(x) - p(x) = \frac{f''(\xi)}{2}(x - x_1)(x - x_2).$$

Divisons désormais $[a, b]$ en N sous-intervalles de longueur $h = \frac{b-a}{N}$. On pose $x_0 = a$ et $x_i = x_0 + hi$ pour $1 \leqslant i \leqslant N$. La fonction f est approximée sur chaque $[x_i, x_{i+1}]$ par une droite, et donc $p(x)$ devient une fonction linéaire par morceaux. L'intégrale de f peut alors être approximée par l'intégrale des N trapèzes formés par $p(x)$. En exploitant [5, Th.20.5.1], on peut prouver le résultat suivant :

Proposition 3.1. *Soient $a, b \in \mathbb{R}$, $N \in \mathbb{N}^*$, et $f : [a, b] \longrightarrow \mathbb{R}$ une fonction de classe C^2, i.e., avec une dérivée seconde continue. Alors l'erreur totale $E_{\text{tot,unif}}$ de la quadrature uniforme de f par N trapèzes est bornée par :*

$$E_{\text{tot,unif}} \leqslant B_{\text{tot,unif}} = \frac{1}{12}\frac{(b-a)^3}{N^2} \max_{x\in[a,b]} |f''(x)|. \quad (2)$$

3.2 Quadrature basée sur la hessienne

Plutôt que de choisir les points de quadrature sur une grille uniforme, nous proposons une *méthode raffinée* où la sélection s'adapte aux variations de la dérivée seconde de $f : [a, b] \to \mathbb{R}$ de classe C^2. On divise $[a, b]$ en k sous-intervalles I_j, $1 \leqslant j \leqslant k$, de longueur respective $l = \frac{b-a}{k}$. Pour permettre une comparaison avec la méthode uniforme, l'interpolation est faite telle que le nombre total de trapèzes sur tous les I_j est N. On divise chaque I_j en n_j sous-intervalles où

$$n_j = \left\lceil N \frac{\sqrt{M_j}}{\sum_{p=1}^k \sqrt{M_p}} \right\rceil \quad (3)$$

avec $M_j = \max_{x\in I_j} |f''(x)|$. En raison de la fonction $\lceil . \rceil$, $\sum_{j=1}^k n_j \approx \sum_{j=1}^k N \frac{\sqrt{M_j}}{\sum_{p=1}^k \sqrt{M_p}} = N$ avec un écart au plus k, celui-ci devenant négligeable quand $N \gg k$. Pour chaque I_j, on effectue une approximation de l'intégrale de $f_{|_{I_j}} : I_j \longrightarrow \mathbb{R}$ par n_j trapèzes puis on somme les résultats. Nous pouvons désormais énoncer le résultat principal :

Théorème 3.1. *Soient $a, b, \in \mathbb{R}$, $k, N \in \mathbb{N}^*$ avec $k \leqslant N$, et $f : [a, b] \longrightarrow \mathbb{R}$ une fonction de classe C^2. Alors le majorant $B_{\text{tot,refined}}$ de l'erreur totale $E_{\text{tot,refined}}$ de notre méthode de quadrature raffinée de l'intégrale de f est plus fine que celle de Eq.* (2) *pour un même nombre de trapèzes :*

$$B_{\text{tot,refined}} = \sum_{j=1}^k \frac{l^3}{12} \frac{1}{\left(\left\lceil N \frac{\sqrt{M_j}}{\sum_{p=1}^k \sqrt{M_p}} \right\rceil\right)^2} |f''(\xi_j)| \leqslant B_{\text{tot,unif}}.$$

Pour chaque $1 \leqslant j \leqslant k$, ξ_j est un élément bien choisi de I_j. En particulier, plus f'' varie et plus l'inégalité précédente est stricte, et donc en faveur de notre méthode raffinée.

Preuve (sketch). Pour tout $1 \leqslant j \leqslant k$, on pose $h_j = \frac{l}{n_j}$ et on applique [5, Th.20.5.1] pour montrer qu'il existe $\xi_j \in I_j$ tel que l'erreur totale de quadrature vérifie : $E_{\text{tot,refined}} = \sum_{j=1}^k -\frac{1}{12} l h_j^2 f''(\xi_j)$. En posant $M = \max_{1\leqslant j\leqslant k} M_j$, alors $E_{\text{tot,refined}}$ est majorée par :

$$\begin{aligned}
B_{\text{tot,refined}} &= \sum_{j=1}^k \frac{l^3}{12} \frac{1}{\left(\left\lceil N \frac{\sqrt{M_j}}{\sum_{p=1}^k \sqrt{M_p}} \right\rceil\right)^2} |f''(\xi_j)| \\
&\leqslant \frac{l^3}{12N^2} \sum_{j=1}^k \left(\sum_{p=1}^k \frac{\sqrt{M_p}}{\sqrt{M_j}} \right)^2 |f''(\xi_j)| \\
&\leqslant \frac{l^3}{12N^2} \sum_{j=1}^k k^2 \frac{M}{M_j} |f''(\xi_j)| \\
&\leqslant \frac{l^3 k^3}{12N^2} M = B_{\text{tot,unif}}.
\end{aligned}$$

On note que la dernière inégalité est obtenue en utilisant $|f''(\xi_j)| \leqslant M_j$, et donc plus f'' varie, plus cette inégalité devient stricte en faveur de notre règle de quadrature. □

3.3 Illustration de la borne du Théorème 3.1

Illustrons ici le comportement de notre méthode de quadrature pour la fonction $f(x) = \sin(\frac{1}{\sqrt{x}})$ sur $[0.1, 1]$. Afin de détermi-

ner M_j pour chaque $1 \leqslant j \leqslant k$, on prend $S = 100$ points régulièrement espacés $x_{j,s} \in I_j$, on calcule $M_j = \max_{1 \leqslant s \leqslant S} |f''(x_{j,s})|$ et on déduit n_j d'après Eq. (3). Notons ici que le coût de calcul de M_j n'a pas d'importance. L'objectif est de montrer que sélectionner N points à partir de f'' donne une plus petite erreur de quadrature que d'utiliser N points régulièrement espacés. En exploitant cette idée dans les PINNs, le même budget de points de collocation sera bien sûr utilisé pour toutes les méthodes dans le processus d'entraînement.

Pour la méthode uniforme, on sélectionne $N + 1$ points régulièrement espacés entre a et b inclus. Ces points forment les extrémités de N trapèzes. On calcule alors leurs aires respectives comme vu en Section 3.1 avant de les sommer.

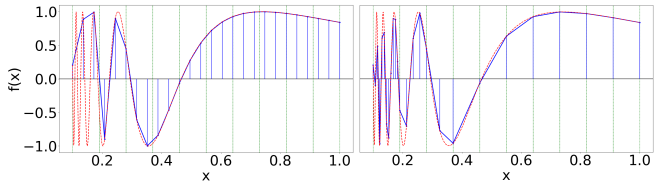

FIGURE 1 : $f(x) = \sin(\frac{1}{\sqrt{x}})$ (en rouge) et ses approximations (en bleu) avec $N = 25$; (à gauche) : méthode uniforme ; (à droite) : méthode raffinée avec $k = 10$.

La Fig. 1 illustre le comportement des deux méthodes de quadrature pour $N = 25$ et $k = 10$, et notamment celui pathologique de la méthode uniforme qui ne peut capturer les fortes variations de f (courbe rouge) sur de petits intervalles. Au contraire, notre méthode utilise seulement une petite partie du budget pour approximer la partie droite de la fonction, et garde la majorité des points de collocation pour les zones où les variations sont fortes, réduisant ainsi considérablement l'erreur de quadrature de 16.4% à 1.89%.

4 Échantillonnage adaptatif et PINNs

Pour comparer les principales méthodes d'échantillonnage adaptatif pour les PINNs, nous utilisons le cadre de la méthode RAD [8], où les N points de collocation sont tirés selon une distribution proportionnelle à un *critère d'intérêt*. Ce dernier peut prendre la forme des **résiduels** de l'EDP comme dans la version originelle RAD [8], être adapté au **gradient** des résiduels [7], à la **hessienne** des résiduels pour notre méthode, ou encore une **distribution uniforme** comme utilisé dans un PINN standard [6]. Soit la distribution générique suivante :

$$d(\mathbf{x}) \quad \propto \quad \frac{\gamma(\mathbf{x})^\tau}{\mathbb{E}[\gamma(\mathbf{x})^\tau]} + c, \tag{4}$$

où τ et c sont des hyperparamètres permettant de contrôler la concentration des points. Les méthodes d'échantillonnage de l'état de l'art peuvent toutes être vues comme des cas spéciaux de Eq. (4), i.e., comme des instanciations d'un algorithme générique que nous appelons dans la suite ★-RAD, où res-RAD, grad-RAD, hessian-RAD, et unif-RAD, correspondent respectivement à une méthode basée sur les résiduels (i.e., où $\gamma(\mathbf{x}) = f(\mathbf{x})$), leur gradient ($\gamma(\mathbf{x}) = f'(\mathbf{x})$), la hessienne ($\gamma(\mathbf{x}) = f''(\mathbf{x})$) et la distribution uniforme (PINN standard obtenu avec $\tau = 0$ et $c \to \infty$). Si les dérivées de f ne sont pas à valeurs scalaires, on utilise la norme du vecteur ou de la matrice correspondante. Le pseudo-code de ★-RAD est présenté dans l'Algorithme 1. Par ailleurs, dans ce qui suit, on utilise $\lambda_3 = 0$ (cf. Eq. (1)).

Algorithme 1 : ★-RAD

Fixer ★ ∈ {"res", "grad", "hessian", "unif"}, τ, c, N et $\#epochs$;
Sélectionner aléatoirement un ensemble de points S ;
Entraîner un PINN pour un nombre d'époques donné ;
while $\#epochs$ *n'est pas atteint* **do**
  Construire une distribution $d(\mathbf{x})$ de Eq. (4) pour ★ à partir d'un ensemble de points aléatoires ;
  $S \leftarrow$ nouvel ensemble de N points i.i.d. $\sim d(\mathbf{x})$;
  Entraîner le PINN pour un nombre d'époques ;
end

Nous présentons dans ce qui suit les résultats obtenus avec les quatre méthodes d'échantillonnage sur deux EDPs en 2D[1].

4.1 Équation de Poisson 2D

L'équation de Poisson est une EDP elliptique du second ordre utilisée en physique théorique et définie comme suit : $\Delta u = F(x, y)$, où $(x, y) \in [0, 1]^2$, et où F est prise telle que $u(x, y) = 2^{4a} x^a (1-x)^a y^a (1-y)^a$ avec $a = 10$ est la solution analytique (Fig. 2 en haut à gauche). Un PINN est appris en utilisant la fonction d'activation `tanh` sur un réseau entièrement connecté et composé de 3 couches cachées de 20 neurones. Les paramètres suivants sont utilisés : $\#epochs = 20000$, le taux d'apprentissage $\eta = 10^{-3}$, $N = 400$ tirés selon $d(\mathbf{x})$ approximée à partir de 40000 candidats, $\tau = 1/2$ et $c = 0$. Le rééchantillonnage est réalisé toutes les 1000 itérations.

La remarque la plus frappante que l'on peut faire à partir de la Fig. 2 (en haut à droite) est que notre méthode hessian-RAD profite pleinement des variations abruptes des solutions de Poisson pour converger beaucoup plus rapidement que les autres. Environ 1000 itérations suffisent, tandis que les stratégies concurrentes nécessitent beaucoup plus d'itérations pour se stabiliser. Fait intéressant, même après 20000 itérations, lorsque les méthodes ont convergé vers une solution exacte, l'écart en termes d'erreur de prédiction en faveur de notre méthode est important, comme l'illustrent les quatre heatmaps de la Fig. 2 (partie inférieure). Les erreurs L_2 sont 6×10^{-5}, 5×10^{-6}, 2×10^{-6} et 7×10^{-7} respectivement pour unif-RAD, res-RAD, grad-RAD et hessian-RAD. Bien que le calcul de la hessienne soit plus coûteux, cette charge supplémentaire est raisonnable et donc compensée par une meilleure prédiction.

4.2 Équation de diffusion-réaction 2D

L'équation de diffusion-réaction est définie par : $\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial x^2} + F(x, t)$, $x \in [-\pi, \pi]$, $t \in [0, 1]$, $u(x, t)$ est la concentration de soluté, $D = 1$ représente le coefficient de diffusion, et $F(x, t) = e^{-t}\left[\frac{3}{2}\sin(2x) + \frac{8}{3}\sin(3x) + \frac{15}{4}\sin(4x) + \frac{63}{8}\sin(8x)\right]$ est la réaction chimique. Les conditions initiales et aux bords sont $u(x, 0) = \sum_{i=1}^{4} \frac{\sin(ix)}{i} + \frac{\sin(8x)}{8}$ et $u(-\pi, t) = u(\pi, t) = 0$. La solution est décrite sur la Fig. 3 (haut gauche). La même architecture de PINN qu'en 4.1 est utilisée avec les paramètres $\#epochs = 100000$, $\eta = 10^{-4}$, $N = 50 \sim d(\mathbf{x})$

[1] Le code est disponible sur ce dépôt GitHub.

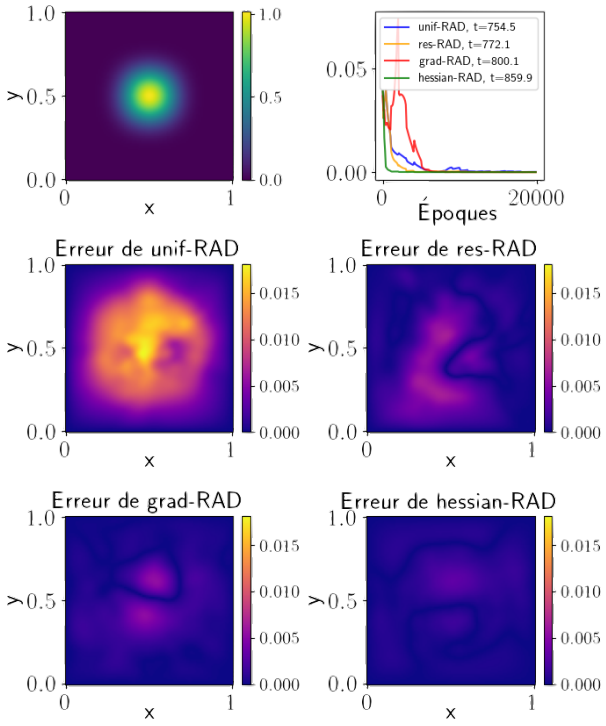

FIGURE 2 : (Haut gauche) Solution analytique de l'équation de Poisson ; (Haut droit) Erreur test L_2 au cours des 20000 premières itérations, et temps de calcul (en s) ; (Milieu et bas) Heatmaps des erreurs des 4 méthodes après 20000 itérations.

approximée à partir de 5000 candidats, $\tau = 1/2$ et $c = 0$. Étonnamment sur cette EDP, le niveau des résiduels n'est que très peu informatif pour bien échantillonner les points de collocation, probablement du fait d'un *loss landscape* très peu lisse (contrairement à la Section 4.1) lié aux fortes variations de la solution. La méthode res-RAD est ainsi bien moins efficace que les autres avec une erreur 5 à 10 fois supérieure (3×10^{-2}), alors que celles d'unif-RAD, grad-RAD et hessian-RAD sont 6×10^{-3}, 3×10^{-3} et 3×10^{-3} respectivement. Les deux méthodes basées sur les dérivées d'ordre 1 et 2 (voir heatmaps d'erreur sur Fig. 3 - bas) capturent bien mieux les variations de la loss, tout comme la méthode uniforme (haut droite) qui est un peu moins performante mais permet néanmoins de bien couvrir le domaine. On notera que les 3 distributions d'erreurs sont très différentes, celle d'unif-RAD tendant à être répartie sur l'ensemble du domaine, celles de grad-RAD et hessian-RAD étant plus localisées autour de zones spécifiques.

5 Conclusion

Nous avons présenté une méthode de quadrature basée sur les dérivées secondes. L'exploitation de la hessienne des résiduels montre également des résultats prometteurs dans une méthode d'échantillonnage adaptatif pour les PINNs. Mais l'utilisation de f'' peut devenir coûteuse en haute dimension. Une direction porte sur une approche stochastique, où à chaque rééchantillonnage, les éléments de la hessienne à calculer seraient échantillonnés. Une autre direction consiste à s'appuyer sur le fait que des méthodes comme gPINN [9] réalisent déjà une grande partie des calculs nécessaires pour la hessienne, et peuvent donc être combinées avec peu de coût supplémentaire.

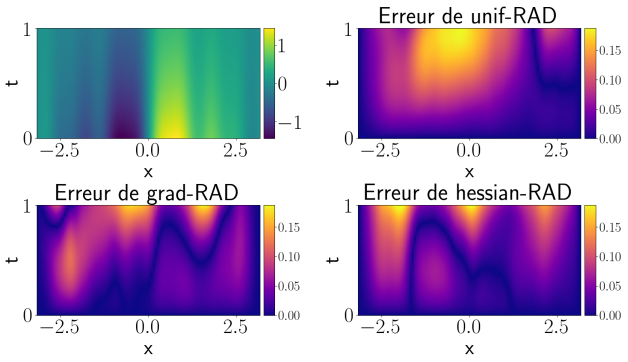

FIGURE 3 : (Haut gauche) Solution analytique de l'équation de diffusion-réaction ; (Autres) Heatmaps des erreurs des méthodes basées sur les résiduels après 100000 itérations.

Remerciements. Ce travail a été financé par l'ANR sous le projet "France 2030", avec la référence EUR MANUTECH SLEIGHT - ANR-17-EURE-0026.

Références

[1] C. Bajaj, L. McLennan, T. Andeen et A. Roy : Recipes for when physics fails : recovering robust learning of physics informed neural networks. *Machine Learning : Science and Technology*, 4(1):015013 (2023).

[2] A. Daw, J. Bu, S. Wang, P. Perdikaris et A. Karpatne : Mitigating propagation failures in physics-informed neural networks using retain-resample-release (R3) sampling. *PMLR* vol. 202 (2023).

[3] N. Doumèche, G. Biau et C. Boyer : Convergence and error analysis of PINNs. arXiv:2305.01240 (2023).

[4] B. Girault, R. Emonet, A. Habrard, J. Patracone et M. Sebban : Approximation Error of Sobolev Regular Functions with tanh Neural Networks : Theoretical Impact on PINNs. *ECML* (2024).

[5] R. W. Hamming : Numerical methods for scientists and engineers. 2nd ed. International Series in Pure and Applied Mathematics (1973).

[6] M. Raissi, P. Perdikaris et G.E Karniadakis : Pinns : A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics* (2019).

[7] S. Subramanian, R. M. Kirby, M.W. Mahoney et A. Gholami : Adaptive self-supervision algorithms for physics-informed neural networks. *ECAI* (2023).

[8] C. Wu, M. Zhu, Q. Tan, Y. Kartha et L. Lu : A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, vol. 403 (2023).

[9] J. Yu, L. Lu, X. Meng et G. Karniadakis : Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering*, vol. 393 (2022).