Projet de recherche en Mathématiques

Approximation des solutions d'équations différentielles partielles par apprentissage automatique

Problématique :

"Peut-on utiliser des techniques simples de machine learning pour approximer les solutions d'EDP ?"

Présenté par :
Leon Lafosse, M1 MIDS

Université Paris-Cité

M1 Mathématiques, Modélisation, Apprentissage Statistique

Année: 2024-2025

Ce projet revisite l'utilisation de méthodes de machine learning pour approximer les solutions d'équations différentielles partielles classiques comme l'équation de diffusion de chaleur.

Sommaire

1 Introduction

1.1 Contexte et motivation

Une équation différentielle partielle (EDP) est une équation différentielle dans laquelle les solutions sont des fonctions inconnues, dépendant de plusieurs variables.
Ces EDP sont des outils mathématiques essentiels pour modéliser des phénomènes qui dépendent, par exemple, du temps et de l'espace, tels que la diffusion de la chaleur ou la propagation des ondes en sciences physique.

Cependant, la résolution exacte de ces équations est souvent complexe. Pour des EDO, la méthode de la séparation des variables s'applique aisément alors que pour des EDP, il faut d'abord vérifier si elles sont séparables, c'est-à-dire si leur solution s'écrit comme un produit ou une somme de fonctions d'une seule variable.

Ainsi, les méthodes numériques, telles que les différences finies, constituent une approche efficace afin d'approcher les solutions d'une EDP, si elles existent.
Toutefois, ces méthodes peuvent être coûteuses en termes de calcul (complexité) et sensibles aux conditions limites et aux discrétisations choisies : une discrétisation trop grossière peut entraîner une perte de régularité.

Dans notre contexte, avec l'essor des techniques d'apprentissage automatique, on se rend compte que l'idée d'exploiter des méthodes simples d'apprentissage automatique pour approximer les solutions d'EDP mérite d'être explorée. On éviterait, par exemple, les erreurs liées à la discrétisation.

L'objectif de ce projet est donc d'évaluer si des approches simples, comme les modèles de réseaux de neurones, peuvent fournir des résultats efficaces pour la résolution des EDP. Nous comparerons les performances de chaque méthode.

1.2 Problématique

La problématique centrale de ce projet est la suivante :

"Peut-on utiliser des méthodes de machine learning pour approximer les solutions d'EDP ?"

L'application à la résolution d'EDP reste un champ de recherche actif, donc, on ne prétendra pas pouvoir apporter une solution générale à ce problème, on se limitera à une exploration modeste.
J'ai choisi ce sujet car les EDP me passionnent et je souhaite approfondir leur utilisation en master, notamment dans le traitement d'image, étant le domaine où je souhaite m'orienter.

2 Définition d'une équation aux dérivées partielles (EDP)

Une équation aux dérivées partielles est une équation où l'inconnue est une fonction dépendant de plusieurs variables, et qui met en jeu les dérivées partielles de cette fonction. Elle peut s'écrire sous la forme générale :

- L'inconnue : $u : \mathbb{R}^n \to \mathbb{R}$.
- L'équation : $F(x, u(x), Du(x), \ldots, D^p u(x)) = 0, \quad \forall x \in \mathbb{R}$.

Un exemple classique est l'équation des ondes :

$$\frac{\partial^2 u}{\partial t^2}(x,t) - \frac{\partial^2 u}{\partial x^2}(x,t) = 0, \quad \forall x \in \Omega \subset \mathbb{R}^n, \forall t > 0. \tag{1}$$

Si l'on considère une EDP d'ordre inférieur ou égal à 2 avec des coefficients constants, on peut écrire :

$$a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial x \partial y} + c\frac{\partial^2 u}{\partial y^2} + d\frac{\partial u}{\partial x} + e\frac{\partial u}{\partial y} + fu = 0, \tag{2}$$

avec a, b, c, d, e, f des coefficients réels donnés.

Si l'on écrit la forme quadratique associée, on peut en déduire la classification de l'EDP :

$$q(x, y) = ax^2 + bxy + cy^2 + dx + ey + f. \tag{3}$$

On peut alors distinguer trois types d'EDP :

- Si q est une ellipse, l'EDP est elliptique.
- Si q est une hyperbole, l'EDP est hyperbolique.
- Si q est une parabole, l'EDP est parabolique.

La classification des EDP est importante car elle détermine le comportement des solutions et nous guide pour le choix des méthodes de résolution.
Par exemple, les EDP elliptiques modélisent des phénomènes à l'équilibre, les paraboliques décrivent des évolutions dans le temps, et les hyperboliques concernent des phénomènes de propagation.

3 Exemples et classification des EDP d'ordre inférieur ou égal à 2

On distingue principalement trois grandes classes d'EDP :

3.1 Équations de type elliptique

L'exemple typique est l'équation de Poisson :

$$-\Delta u(x) = \sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2}(x) = f(x), \quad \forall x \in \Omega \subset \mathbb{R}^n. \tag{4}$$

3.2 Équations de type parabolique

L'exemple standard est l'équation de la chaleur :

$$\frac{\partial T}{\partial t}(x,t) - \alpha \Delta T(x,t) = 0, \quad \forall x \in \Omega \subset \mathbf{R}^n, \forall t > 0, \quad \alpha > 0. \tag{5}$$

Cette équation modélise l'évolution d'un système en fonction du temps.

3.3 Équations de type hyperbolique

Un exemple typique est l'équation de transport :

$$\frac{\partial u}{\partial t}(x,t) + a\frac{\partial u}{\partial x}(x,t) = 0, \quad \forall x \in \Omega \subset \mathbb{R}^n, \forall t > 0, \quad a \in \mathbb{R}. \tag{6}$$

Cette équation décrit la propagation d'une grandeur sans diffusion.

Cette classification va ainsi influencer les approches numériques. Les différences finies s'appliquent bien aux EDP paraboliques et hyperboliques tandis que pour des EDP elliptiques nous opterons plutôt pour la méthode des éléments finis.

4 Approche numérique d'une EDP par différences finies

La méthode des différences finies est l'une des techniques numériques les plus couramment utilisées pour approximer les solutions des équations différentielles. Cette méthode repose sur la discrétisation des dérivées en utilisant des expressions discrètes basées sur des points d'un maillage.

4.1 Principe des différences finies

Pour une fonction $u(x)$ définie sur un intervalle, on commence par discrétiser l'espace en une grille de pas h. Les points du maillage sont donnés par :

$$x_i = x_0 + ih, \quad i = 0, 1, \ldots, N,$$

où h est le pas spatial et N est le nombre de points de discrétisation.

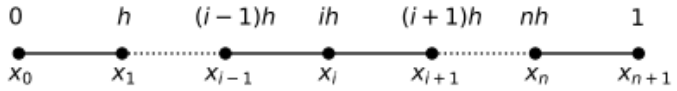

Figure 1: Maillage uniforme de [0,1]. Noter que x_0 et x_{n+1} sont au bord et qu'il y a n points à l'intérieur du segment.

4.1.1 Approximation des dérivées premières

Les dérivées de la fonction $u(x)$ peuvent être approximées en utilisant les différences finies. Voici les approximations les plus courantes :

- **Différence avant** : La dérivée première peut être approximée par la différence avant :
$$\frac{du}{dx} \approx \frac{u(x_{i+1}) - u(x_i)}{h}.$$
Cette approximation est d'ordre 1 en h, ce qui signifie que l'erreur est proportionnelle à h.

- **Différence arrière** : La dérivée première peut également être approximée par la différence arrière :
$$\frac{du}{dx} \approx \frac{u(x_i) - u(x_{i-1})}{h}.$$
Cette approximation est également d'ordre 1 en h.

- **Différence centrée** : Pour une approximation plus précise, on utilise la différence centrée :
$$\frac{du}{dx} \approx \frac{u(x_{i+1}) - u(x_{i-1})}{2h}.$$
Cette approximation est d'ordre 2 en h, ce qui signifie que l'erreur est proportionnelle à h^2.

4.1.2 L'approximation de la dérivée seconde

Pour démontrer l'approximation de la dérivée seconde, partons du développement en série de Taylor de $u(x_{i+1})$ et $u(x_{i-1})$ autour de x_i :

$$u(x_{i+1}) = u(x_i) + h\frac{du}{dx}(x_i) + \frac{h^2}{2}\frac{d^2u}{dx^2}(x_i) + \frac{h^3}{6}\frac{d^3u}{dx^3}(x_i) + \mathcal{O}(h^4),$$
$$u(x_{i-1}) = u(x_i) - h\frac{du}{dx}(x_i) + \frac{h^2}{2}\frac{d^2u}{dx^2}(x_i) - \frac{h^3}{6}\frac{d^3u}{dx^3}(x_i) + \mathcal{O}(h^4).$$

En additionnant ces deux équations, on obtient :

$$u(x_{i+1}) + u(x_{i-1}) = 2u(x_i) + h^2\frac{d^2u}{dx^2}(x_i) + \mathcal{O}(h^4).$$

En réarrangeant, on trouve l'approximation de la dérivée seconde :

$$\boxed{\frac{d^2u}{dx^2}(x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2}}$$

Cette approximation est d'ordre 2 en h, ce qui signifie que l'erreur est proportionnelle à h^2.

4.2 Application à l'équation de la chaleur

L'équation de la chaleur en une dimension est un exemple classique d'EDP parabolique. Elle s'écrit :

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2},$$

où $u(x,t)$ est la température au point x et au temps t, et α est la diffusivité thermique.
Les conditions limites sont souvent imposées sur les bords de l'intervalle spatial :

$$u(0,t) = u_L \quad \text{et} \quad u(L,t) = u_R,$$

où u_L et u_R sont des températures constantes aux extrémités de l'intervalle $[0, L]$.
La condition initiale est donnée pour la température à $t = 0$:

$$u(x,0) = f(x),$$

où $f(x)$ est une fonction donnée représentant la distribution initiale de la température.

4.2.1 Discrétisation de l'équation de la chaleur

Pour résoudre numériquement cette équation, nous discrétisons le domaine spatial et temporel. Le domaine spatial est divisé en N_x points avec un pas h, et le domaine temporel est divisé en N_t points avec un pas Δt. Les points du maillage sont donnés par :

$$x_i = ih, \quad i = 0, 1, \ldots, N_x,$$

$$t^n = n\Delta t, \quad n = 0, 1, \ldots, N_t.$$

En utilisant les différences finies pour approximer les dérivées, l'équation de la chaleur devient :

$$\boxed{\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2}}.$$

En réarrangeant cette équation, on obtient un schéma explicite pour calculer la température au temps t^{n+1} en fonction de la température au temps t^n:

$$u_i^{n+1} = u_i^n + \lambda(u_{i+1}^n - 2u_i^n + u_{i-1}^n), \quad \lambda = \frac{\alpha \Delta t}{h^2}.$$

4.2.2 Condition de stabilité

Le schéma explicite utilisé pour discrétiser l'équation de la chaleur est simple à implémenter, mais il peut être instable si on choisit mal les paramètres h et Δt. Concrètement, si ces valeurs ne respectent pas une certaine condition, la solution devient complètement fausse, avec des résultats qui explosent au fil du temps.
Pour éviter cela, il faut respecter une condition de stabilité. Elle impose une contrainte sur Δt en fonction du pas spatial h :

$$\lambda = \frac{\alpha \Delta t}{h^2} \leq \frac{1}{2}.$$

Démonstration Considérons le schéma explicite obtenu précédemment :

$$u_i^{n+1} = u_i^n + \lambda \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n \right).$$

On suppose une solution sous la forme d'une onde discrète :

$$u_i^n = A^n e^{ikx_i},$$

où k est le nombre d'onde. En injectant cette expression dans l'équation discrète, on obtient :

$$A^{n+1} e^{ikx_i} = A^n e^{ikx_i} + \lambda A^n \left(e^{ik(x_i+h)} - 2e^{ikx_i} + e^{ik(x_i-h)} \right).$$

En factorisant par $A^n e^{ikx_i}$, on a :

$$A^{n+1} = A^n \left(1 + \lambda \left(e^{ikh} - 2 + e^{-ikh}\right)\right).$$

En utilisant la formule d'Euler : $e^{ikh} + e^{-ikh} = 2\cos(kh)$, on peut réécrire cette expression sous la forme :

$$A^{n+1} = A^n \left(1 + 2\lambda \left(\cos(kh) - 1\right)\right).$$

Pour que la solution ne diverge pas, le facteur $\left|\frac{A^{n+1}}{A^n}\right|$ doit être inférieur ou égal à 1, ce qui donne la condition :

$$|1 + 2\lambda \left(\cos(kh) - 1\right)| \leq 1.$$

Comme $\cos(kh) - 1$ est toujours négatif, on impose :

$$-1 \leq 1 + 2\lambda \left(\cos(kh) - 1\right) \leq 1.$$

En simplifiant, on obtient :

$$-2\lambda \leq 0 \quad \text{et} \quad 2\lambda \left(\cos(kh) - 1\right) \geq -2.$$

Ce qui donne finalement :

$$\lambda \leq \frac{1}{2}.$$

■

L'idée, c'est que pour que la simulation soit fiable, l'information ne doit pas se propager trop vite par rapport au maillage. Si Δt est trop grand par rapport à h^2, les erreurs numériques s'accumulent et deviennent incontrôlables, ce qui fait diverger la solution.

4.3 Implémentation numérique

Voici une implémentation en Python du schéma explicite pour l'équation de la chaleur :

```
import numpy as np
import matplotlib.pyplot as plt

# Paramètres
Nx = 50  # nb de points en espace
Nt = 100  # nb de points en temps
L = 1.0  # longueur du domaine
T = 0.1  # temps final
alpha = 0.01  # diffusivité thermique

h = L / (Nx - 1)
dt = T / (Nt - 1)
lmbda = alpha * dt / h**2

x = np.linspace(0, L, Nx)
u = np.zeros((Nt, Nx))

def initial_condition(x):
    return np.sin(np.pi * x)

u[0, :] = initial_condition(x)

for n in range(Nt - 1):
    for i in range(1, Nx - 1):
        u[n+1, i] = u[n, i] + lmbda * (u[n, i+1] - 2*u[n, i] + u[n, i-1])

# courbe de la température à t = T/2
plt.plot(x, u[Nt//2, :], label='t = T/2', color = 'red')

plt.xlabel('Position x')
plt.ylabel('Température u(x, t)')
plt.title('Température à t = T/2')
plt.legend()
plt.grid(True)
plt.show()
```

Nous obtenons ainsi :

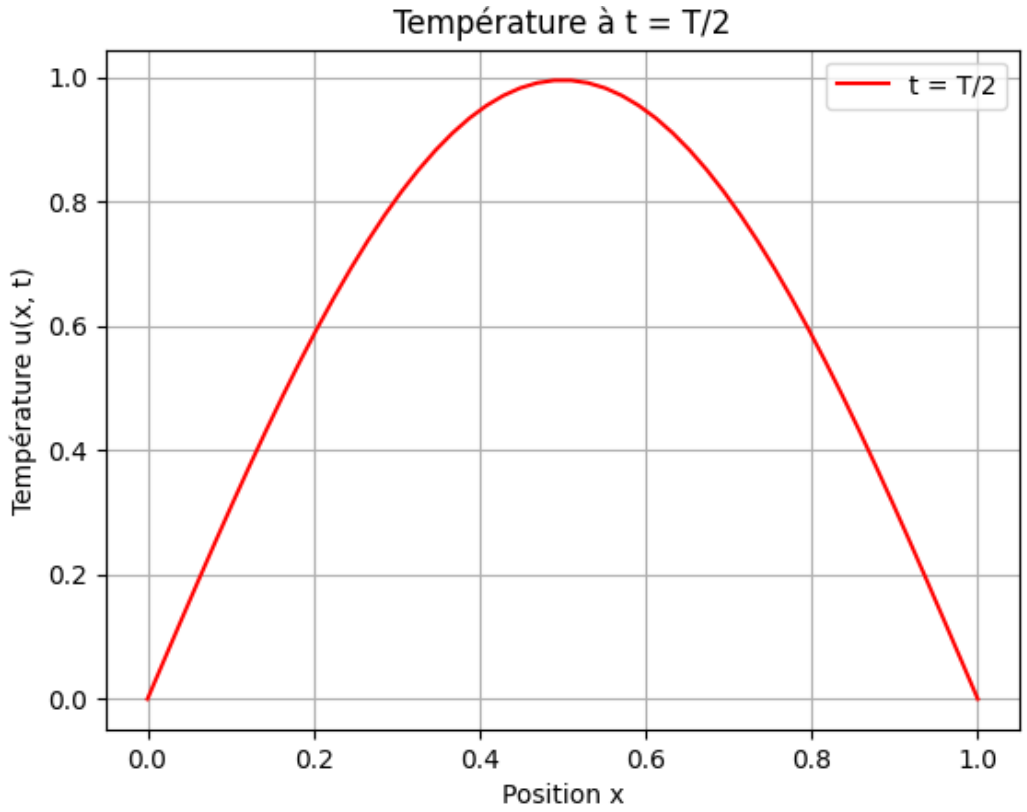

Figure 2: Evolution de la température $u(x,t)$ à $t = T/2$

L'image montre l'évolution de la température $u(x,t)$ à $t = T/2$. La courbe en rouge représente la température en fonction de la position x. Elle suit une forme parabolique, avec un maximum au centre et des valeurs nulles aux bords, conformément aux conditions aux limites. Cette répartition traduit la diffusion thermique au cours du temps, issue de l'initialisation $u(x,0) = \sin(\pi x)$.

5 Approximation des solutions d'EDP par les PINN

Les méthodes numériques comme les différences finies présentées précédemment sont efficaces pour résoudre des EDP, mais elles présentent certaines limitations si l'on discute en termes de dimensionnalité.
Nous nous baserons encore sur l'équation de chaleur. Bien évidemment, la méthode des différences finies est suffisante pour approximer les solutions de l'équation de chaleur. Il s'agit ici, de présenter une nouvelle approche en utilisant les réseaux de neurones.

5.1 Principe des Physics-Informed Neural Networks

Les Physics-Informed Neural Networks (PINN) sont une méthode innovante qui associe les principes des EDP avec la capacité des réseaux de neurones à faire des approximations.
Cette méthode, proposée par Raissi en 2019, permet d'intégrer directement les connaissances physiques dans le modèle d'apprentissage.

5.1.1 Formulation du problème

Dans l'approche PINN, nous approximons la solution $u(x,t)$ par un réseau de neurones $u_\theta(x,t)$, où θ représente les paramètres du réseau (poids et biais).

5.1.2 Architecture et différentiation automatique

L'élément clé des PINN est la différentiation automatique, qui permet de calculer les dérivées partielles du réseau de neurones $u_\theta(x,t)$ par rapport à ses entrées (x et t). Ces dérivées sont nécessaires pour évaluer les termes différentiels dans l'EDP.

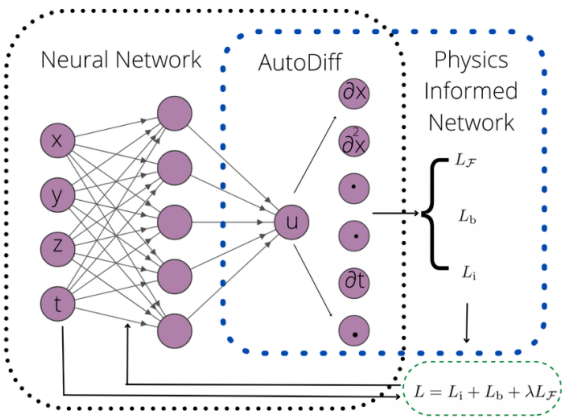

Figure 3: Architecture de réseau de neurones informée par la physique

L'architecture d'un PINN comprend généralement :

- Un réseau feedforward multicouche avec des entrées représentant les coordonnées spatio-temporelles (x, t)
- Une couche de sortie produisant la valeur de la solution $u(x, t)$
- Un mécanisme de différentiation automatique (autodiff dans l'image) pour calculer les dérivées partielles

5.2 Fonction de perte physiquement contrainte

La fonction de perte d'un PINN combine plusieurs termes :

$$\mathcal{L}(\theta) = \lambda_r \mathcal{L}_r(\theta) + \lambda_b \mathcal{L}_b(\theta) + \lambda_i \mathcal{L}_i(\theta), \tag{7}$$

où :

- $\mathcal{L}_r(\theta)$ est le terme de résidu de l'EDP, évalué sur des points de collocation à l'intérieur du domaine
- $\mathcal{L}_b(\theta)$ est le terme d'erreur des conditions aux limites
- $\mathcal{L}_i(\theta)$ est le terme d'erreur des conditions initiales
- λ_r, λ_b, et λ_i sont des coefficients de pondération

Pour l'équation de la chaleur $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$, ces termes s'écrivent :

$$\mathcal{L}_r(\theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \frac{\partial u_\theta}{\partial t}(x_i^r, t_i^r) - \alpha \frac{\partial^2 u_\theta}{\partial x^2}(x_i^r, t_i^r) \right|^2, \tag{8}$$

$$\mathcal{L}_b(\theta) = \frac{1}{N_b} \sum_{i=1}^{N_b} \left| u_\theta(x_i^b, t_i^b) - g(x_i^b, t_i^b) \right|^2, \tag{9}$$

$$\mathcal{L}_i(\theta) = \frac{1}{N_i} \sum_{i=1}^{N_i} \left| u_\theta(x_i^i, 0) - f(x_i^i) \right|^2, \tag{10}$$

où (x_i^r, t_i^r), (x_i^b, t_i^b), et $(x_i^i, 0)$ sont des points d'échantillonnage à l'intérieur du domaine, sur les frontières, et aux conditions initiales, respectivement.

5.3 Avantages des PINN par rapport aux méthodes numériques

Les PINN offrent plusieurs avantages par rapport aux méthodes numériques traditionnelles :

- **Solution continue** : Contrairement aux méthodes de différences finies qui fournissent des approximations discrètes, les PINN produisent une solution continue et différentiable
- **Maillage** : Les PINN ne nécessitent pas de maillage.
- **Problèmes de haute dimension** : Les PINN sont plus flexibles pour résoudre des problèmes multidimensionnels, là où les méthodes numériques deviennent trop coûteuses en terme de complexité.

6 Conclusion générale sur le projet de recherche

Malheureusement, je manque de temps pour finaliser ce projet dans les délais. Actuellement, le projet est en cours et je me documente afin d'implémenter une telle méthode en Python avec la bibliothèque torch.
Une fois cette étape terminée, il aurait été essentiel de comparer les solutions obtenues, d'évaluer les coûts d'entraînement et d'évaluation des modèles et enfin d'étudier le cas de la dimensionnalité.

Cependant, cette approche a de nombreux avantages : elle permet d'obtenir une solution continue, sans nécessiter de maillage comme pour les différences finies. En revanche, il y a aussi des inconvénients, il est nécessaire d'avoir de grandes quantités de données pour l'entraînement. Il faut également penser à la gestion de la complexité des réseaux de neurones et le temps de calcul souvent élevé pour des problèmes à grande échelle. Pour l'instant, c'est principalement le temps de calcul qui constitue un obstacle dans mon code.

Je poursuivrai bien évidemment ce projet, car les EDP sont un domaine qui me passionne et que je souhaite approfondir durant le master. L'introduction des EDP dans le traitement d'image permet de considérer les images comme des signaux continus, ainsi, en combinant ces deux domaines, je pense que ce projet sera non seulement utile pour approfondir mes connaissances en EDP, mais également pour explorer les applications pratiques dans le domaine du traitement d'image. En effet, dans la finalité, c'est vers le traitement d'image que je souhaite m'orienter.

7 Références

[1] **Méthodes d'apprentissage automatique pour la résolution de problèmes de contrôle stochastique et d'équations aux dérivées partielles en grande dimension**
Par Maximilien Germain, Présentée et soutenue publiquement le 20/06/2022.

[2] **Résolution Numérique de l'Equation de la chaleur**
http://www.lmm.jussieu.fr/ lagree/COURS/MECAVENIR/cours7$_e qchal_n um.pdf$

[3] **Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations**
Maziar Raissi, Paris Perdikaris, and George Em Karniadakis.
https://arxiv.org/pdf/1711.10561

[4] **PDEs using PINNs (Raissi)**
PDEs using PINNs (Raissi)

[5] **Figure 3: Physics-informed neural network architecture**
https://www.researchgate.net/figure/Physics-informed-neural-network-architecture$_f ig2_367565$