Otimização de Hiperparâmetros de Redes Neurais guiadas pela Física em Problema Convectivo-Difusivo

Ricardo Ervilha Silva[1], **José J. Camata**[1]

[1] Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora (UFJF)
Caixa Postal 20.010 – 36.036-230 – Juiz de Fora – MG – Brazil.

ricardo.ervilha@estudante.ufjf.br, camata@ice.ufjf.br

Abstract. *Physics-Informed Neural Networks (PINN) are an innovative technique for solving problems governed by Differential Equations, with applications in science and engineering. This study investigates the application of this method to Transient Convection-Diffusion equations. Through the DeepXDE library, which supports frameworks such as TensorFlow and PyTorch, the optimization of hyperparameters in the mentioned problem was analyzed. The experiments conducted were compared with finite element numerical methods, and the results demonstrate that this approach is promising, offering a new path compared to traditional techniques.*

Resumo. *Redes Neurais Guiadas pela Física (PINN) são uma técnica inovadora para resolver problemas governados por Equações Diferenciais, com aplicações em ciência e engenharia. Este estudo investiga a aplicação desse método em equações Convectivos-Difusivos Transientes. Através da biblioteca DeepXDE, que suporta frameworks como TensorFlow e PyTorch, analisou-se a otimização de hiperparâmetros no problema citado. Os experimentos realizados foram comparados com métodos numéricos de elementos finitos, e os resultados demonstram que essa abordagem é promissora, oferecendo um novo caminho em relação às técnicas tradicionais.*

1. Introdução

Redes Neurais Profundas (DNNs) são utilizadas com eficácia em diversos problemas, incluindo classificação de imagens, processamento de linguagens naturais e mecanismos de recomendação [Zhang et al. 2023]. Recentemente, uma abordagem emergente envolve as Redes Neurais guiadas pela Física (PINN, do inglês, *physics-informed neural networks*), que estão revolucionando a solução de problemas governados por equações diferenciais parciais (EDPs) tanto na ciência quanto na engenharia [Raissi et al. 2019]. Essas redes incorporam conhecimento físico no treinamento, adicionando componentes adicionais à função de perda que refletem o resíduo das equações diferenciais e suas condições de ínicio e contorno.

PINNs podem ser desenvolvidas e treinadas utilizando frameworks amplamente conhecidos, como TensorFlow[1], Keras[2] e PyTorch[3]. Este trabalho explora a ferramenta

[1] https://www.tensorflow.org/?hl=pt-br
[2] https://keras.io/
[3] https://pytorch.org/

DeepXDE [Lu et al. 2021], uma biblioteca de Machine Learning Científico que se baseia nesses frameworks de Deep Learning. DeepXDE[4] oferece uma codificação mais compacta, facilitando a formulação de problemas em uma abordagem semelhante às tradicionais expressões matemáticas.

Adicionalmente, algoritmos de Machine Learning (ML) costumam ter diversos hiperparâmetros, como número de épocas de treinamento, *batch size*, número de camadas ocultas, taxa de aprendizado, entre outras. Tais hiperparâmetros costumam assumir diferentes valores e, *a priori*, são independentes uns dos outros. Isso causa uma grande dificuldade para encontrar a melhor combinação, impactando fortemente na qualidade do modelo treinado. Para lidar com isso, a otimização Bayesiana baseada em Processos Gaussianos vem demonstrando bons resultados [Hansen et al. 2022, Escapil-Inchauspé and Ruz 2023, Bhaumik et al. 2024].

Assim, este trabalho explora o potencial das PINNs para a resolução de problemas Convectivos-Difusivos Transientes que são essenciais para descrever fenômenos onde partículas, energia ou outras quantidades são transferidas em um sistema devido a processos de difusão e convecção [Atangana 2018]. A otimização Bayesiana é empregada para explorar diversas combinações de hiperparâmetros, utilizando o histórico da função de perda como aliado na predição futura das melhores combinações no espaço de busca.

Portanto, o principal objetivo deste trabalho é explorar o potencial das PINNs na solução de problemas convectivos-difusivos transientes, realizando uma comparação com as soluções obtidas por meio do método de elementos finitos. Essa comparação permitirá avaliar a eficácia das PINNs frente a métodos numéricos tradicionais.

O restante deste artigo está estruturado da seguinte forma: na Seção 2, discute-se a teoria sobre PINNs. Aspectos cruciais da busca por hiperparâmetros são abordados na Seção 3. Enquanto isso, a Seção 4 é apresenta o problema de interesse, a configuração dos testes e a discussão dos resultados. Finalmente, a Seção 5 sumariza as principais conclusões e possíveis próximas etapas de pesquisa.

2. Visão Geral Sobre Redes Neurais Guiadas pela Física

Conforme apresentado em [Cuomo et al. 2022], Redes Neurais Guiadas pela Física (PINNs) podem resolver problemas utilizando poucos dados e medições experimentais. As PINNs resolvem equações diferenciais que podem ser genericamente expressas como:

$$\begin{aligned} \mathcal{N}(u(z);\gamma) &= f(z), \quad \text{para } z \in \Omega \\ \mathcal{B}(u(z)) &= g(z), \quad \text{para } z \in \partial\Omega \end{aligned} \tag{1}$$

onde $\Omega \subset \mathbb{R}^d$ é o domínio e $\partial\Omega$ é seu contorno. Aqui, z representa as coordenadas espaço-temporais, u é a solução desconhecida, γ são os parâmetros físicos, f é a função descritiva do problema, e $\mathcal{N}$ é um operador diferencial não linear. Conforme ilustrado em [Cuomo et al. 2022], a equação (1) pode descrever diversos sistemas, incluindo tanto problemas diretos quanto inversos.

No contexto das PINNs, $u(z)$ é predito computacionalmente por um modelo de rede neural (NN), parametrizado por parâmetros θ, fornecendo a seguinte aproximação:

$$\hat{u}_\theta(z) \approx u(z) \tag{2}$$

[4]https://github.com/lululxvi/deepxde/

onde $\hat{u}_\theta$ denota a aproximação obtida pela rede com parâmetros θ. Redes neurais são usadas para aproximar as equações diferenciais encontrando os melhores parâmetros θ^* que minimizam uma função de perda, a qual depende diretamente da equação diferencial $\mathcal{L}_\mathcal{N}$, das condições de contorno $\mathcal{L}_{\mathcal{BC}}$, das condições iniciais $\mathcal{L}_{\mathcal{BI}}$, e eventualmente de dados de simulação $\mathcal{L}_{\mathcal{DATA}}$. Equacionando tudo, pode-se denotar formalmente o seguinte:

$$\theta^* = \arg\min_\theta(\mathcal{L}_\mathcal{N} + \mathcal{L}_{\mathcal{BC}} + \mathcal{L}_{\mathcal{BI}} + \mathcal{L}_{\mathcal{DATA}}) \tag{3}$$

Sobre a equação (3), baseado em [Wang et al. 2021], PINNs tem uma falha estrutural no balanceamento dos gradientes que levam a problemas de qualidade na solução. Nesse caso, a rede pode focar demasiadamente em uma parcela da *loss*, por ter maior ordem de magnitude, e negligenciar as demais. A fim de mitigar essa falha, recorre-se a fatores de escalamento, no intuito de ajudar a nivelar as contribuições para o gradiente final, e garantir que todos os termos são corretamente levados em consideração. Com a adição de tais artifícios, a função perda é reescrita como:

$$\theta^* = \arg\min_\theta(\omega_\mathcal{N}\mathcal{L}_\mathcal{N} + \omega_{\mathcal{BC}}\mathcal{L}_{\mathcal{BC}} + \omega_{\mathcal{BI}}\mathcal{L}_{\mathcal{BI}} + \omega_{\mathcal{DATA}}\mathcal{L}_{\mathcal{DATA}}) \tag{4}$$

Outro aspecto importante de ser destacado, conforme citado por [Markidis 2021], é que o treinamento da PINN não requer necessariamente dados rotulados. Fornecer dados para a rede é possível, conforme citado na equação (3) e, assim, ter uma perda específica para ajuste de dados de simulações ($\mathcal{L}_{\mathcal{DATA}}$). Essa abordagem supervisionada é frequentemente usada para resolver problemas mal definidos quando, por exemplo, não temos condições de contorno explicitadas. Como esse não é o caso neste estudo, o foco está apenas na PINN básica, sem fornecer dados adicionais para treinamento.

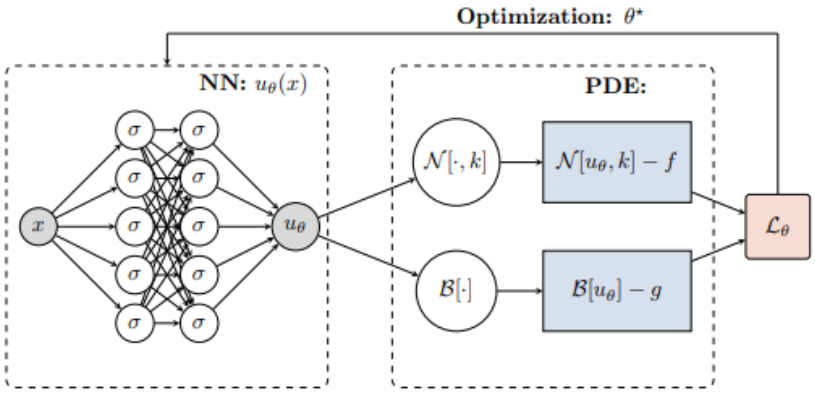

Figura 1. Representação ilustrativa de uma PINN. A equação diferencial é resolvida levando em consideração o resíduo, e a rede neural obtém os parâmetros θ otimizados via treinamento [Escapil-Inchauspé and Ruz 2023].

Para finalizar essa discussão, a Figura 1 resume os pontos discutidos nesta seção. As PINNs são compostas por três módulos principais: a rede neural, a rede informada pela física, e os mecanismos de feedback. Mais detalhes pode ser encontrado em [Cuomo et al. 2022]

3. Otimização de Hiperparâmetros (HPO)

Conforme dito por [Hansen et al. 2022], existem vários métodos para ajuste automático de hiperparâmetros, sendo os mais comuns: *Grid Search*, *Random Search* e otimização Bayesiana. A otimização Bayesiana é aplicada neste trabalho para identificar os melhores hiperparâmetros da PINN, ao ter apresentado bom desempenho [Escapil-Inchauspé and Ruz 2023]. A otimização bayesiana é especialmente vantajosa para problemas onde a avaliação da função é complexa, não convexa e computacionalmente custosa de avaliar [Asrav and Aydin 2023]. Nesse trabalho, os hiperparâmetros avaliados foram:

1. Taxa de Aprendizado (α)
2. Profundidade: Número de Camadas Ocultas (L)
3. Largura: Número de Neurônios por Camada (N)
4. Função de ativação (σ)

Conforme apresentado em [Escapil-Inchauspé and Ruz 2023], Λ pode ser definido como um espaço formado pelo produto cartesiano de todos os hiperparâmetros utilizados. Cada $\lambda \in \Lambda$ é denotado por:

$$\lambda = [\alpha; N; L; \sigma] \tag{5}$$

Sendo assim, o problema em questão pode ser definido como um problema de otimização em dois níveis: Encontrar o melhor λ tal que:

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \mathcal{L}_{\theta^*}^{test}[\lambda] \tag{6}$$

Com:

$$\theta^* = \arg\min_{\theta \in \Theta} \mathcal{L}_{\theta}^{train}[\lambda] \tag{7}$$

Em que Θ representa o espaço formado por todas as combinações possíveis de pesos na Rede Neural. Sobre o otimizador do HPO, foi-se escolhido Otimização Bayesiana usando processos gaussianos. O mesmo consiste em utilizar uma regressão para prever quais serão as futuras melhores configurações [Escapil-Inchauspé and Ruz 2023].

Algoritmo 1: Algoritmo para escolha dos hiperparâmetros

Entrada: λ_0 (HP's iniciais), g_{acq} (Função de Aquisição)
Saída : λ^*, $loss(\lambda^*)$

begin
  for m ***in*** 0 ***until*** $M-1$ **do**
    $loss(\lambda_m) \leftarrow \arg\min_{\theta \in \Theta} \mathcal{L}_{\theta}^{train}[\lambda_m]$
    Aplicação da Regressão para os pontos $(\lambda_i, loss(\lambda_i))$, onde $i \in [0, m]$
    $\Rightarrow loss(\lambda) \sim GP$
    $\lambda_{m+1} \leftarrow \arg\min_{\lambda} g_{acq}$
  end
end

Conforme apresentado no Algoritmo 1, o procedimento utilizado para a escolha dos hiperparâmetros é iterativo e baseado na minimização da perda com os diferentes λ. Sobre a função de aquisição, utilizou-se a melhoria esperada negativa $(-EI)$, conforme discutido por [Escapil-Inchauspé and Ruz 2023]. Tal função é importante para determinar

qual o próximo conjunto será usado para avaliação na função objetivo. Além disso, define-se a menor $loss$ do teste como erro para uso como função de perda (externa) do HPO. Para facilitar a implementação, foi empregada a função `gp_minimize` disponível no Scikit-optimize[5].

4. Resultado e Discussão

Para esse estudo, foi utilizado um problema de transporte de espécies regida por uma equação de convecção-difusão transiente a fim de testar a aplicabilidade das redes neurais e a avaliação dos hiperparâmetros. Conforme descrito por [Valli et al. 2015], a equação governante é dada por

$$\frac{\partial u}{\partial t} + v\nabla u - \nabla \cdot (k\nabla u) = s, \text{ em } \Omega\times]0, T_f] \tag{8}$$

sendo u a variável de interesse, como a concentração da espécie, v a velocidade do meio, k o coeficiente de difusão e s o termo fonte. Além disso, para caracterizar o problema completamente, condição de contorno de Dirichlet g_D e inicial u_0 devem ser prescritas.

Os experimentos foram realizados usando a biblioteca DeepXDE na versão 1.11.1 [Lu et al. 2021] executado no ambiente *Lightning AI* [6], a qual disponibilza um sistema isolado com GPU NVIDIA L4 tensor core, contendo 24GB de VRAM, 16 CPU's (não encontrado marca e modelo) e memória RAM de 64GB. O DeepXDE aproveita o suporte natural ao paralelismo em GPU oferecido pelo TensorFlow e PyTorch, acelerando significativamente o cálculo dos gradientes e a atualização dos pesos da rede neural.

Para a resolução do problema, os pontos de treino e teste são gerados aleatoriamente usando pontos de uma sequência de Hammersley, sendo o número de pontos amostrados para o interior do domínio igual a 3000, para as condições de contorno 750, para a condição inicial 1500, e para o teste 1500. Para cada iteração dentro do loop interno utilizou-se o número de épocas de treino sendo 10000 com inicialização dos pesos usando o esquema *Glorot Uniform*, *batch size* igual a 16 e o otimizador sendo o Adam. A otimização dos hiperparâmetros rodou com M = 50 iterações, e para fins de reprodutibilidade, a *seed* utilizada foi 9298745. Os parâmetros padrões usados foram:

$$\lambda_0 = [10^{-3}; 5; 60; tanh] \tag{9}$$

A Tabela 1 sumariza o espaço de valores utilizados para cada hiperparâmetro. Estão listados cada parâmetro conjuntamente ao intervalo de variação. A variação da taxa de aprendizado segue uma log-uniforme; e a profundidade da rede, além do número de neurônios por camada, assumem quaisquer valores inteiros dentro do intervalo.

Tabela 1. Tabela compilando a variação dos hiperparâmetros.

Hiperparâmetro	Variação
Taxa de Aprendizado (α)	$[10^{-4}, 5 \times 10^{-2}]$
Profundidade (L)	$[1, 10]$
Largura (N)	$[10, 120]$
Função de Ativação (σ)	$[sigmoid, tanh, Swish, sin]$

[5]https://scikit-optimize.github.io/stable/modules/generated/skopt.gp_minimize.html
[6]https://lightning.ai/

Sobre os pesos relacionados a cada parcela da função de perda (4), é importante mencionar que esses valores foram determinados empiricamente. Foram testados diversos exemplos separadamente, ajustando e verificando os valores que melhor se adaptavam a cada caso. Para o problema descrito em 8 utilizou-se $\omega_{\mathcal{PDE}} = 100$, $\omega_{\mathcal{BTOP}} = \omega_{\mathcal{BRIGHT}} = \omega_{\mathcal{BBOTTOM}} = \omega_{\mathcal{BLEFT}} = 0,1$ e $\omega_{\mathcal{BI}} = 10$.

4.1. Problema de convecção-difusão-transiente

O problema retrata a rotação de um pulso gaussiano ao redor do centro de um domínio quadrado. A equação é definida em um domínio espacial $[0, 10] \times [0, 10]$ e domínio temporal $]0, 2\pi]$. A condição inicial do problema é dada pelo pulso posicionado em $(x, y) = (5, 0; 7, 5)$, ou seja, a função que representa a condição inicial do problema é dada por $u_0(x; y) = e^{-0,5r}$ tal que $r = (x - 5)^2 + (y - 7, 5)^2$. Para condição de contorno é aplicada condição de Dirichlet nula e não há termo fonte. A velocidade de advecção do pulso é dada por

$$\begin{aligned} v_x &= -y + 5, \\ v_y &= x - 5. \end{aligned} \tag{10}$$

No tempo final da simulação $T = 2\pi$ o pulso retorna para a posição inicial. A taxa de difusão foi $k = 10^{-8}$ e por apresentar uma difusão próxima de zero, a solução numérica desse problema requer estabilizações para evitar oscilações espúrias e discretizações refinadas, o que demanda maior custo computacional nos métodos numéricos.

A análise HPO desse problema atinge a solução ótima na 16ª iteração, resultando nos seguintes parâmetros:

$$\lambda^* = [1, 12 \times 10^{-3}; 10; 120; Swish] \tag{11}$$

A Tabela 2 compara as 10 configurações melhores classificadas. Observa-se que, em geral, o erro da PINN é baixo, com uma média de $6, 62 \times 10^{-4}$. Também pode-se identificar alguns padrões nas melhores configurações. Por exemplo, apenas duas funções de ativação aparecem, sendo que o $Swish$ está presente nas quatro primeiras posições. Nota-se que, em geral, as configurações relacionadas à arquitetura da rede tendem a ter maior profundidade e largura. Além disso, a taxa de aprendizado fica em torno de $2, 3 \times 10^{-3}$ na média com um desvio padrão de $1, 62 \times 10^{-3}$. No mais, a partir da 34ª posição da classificação, o erro fica na magnitude 10^{-1}, mostrando que nem sempre para um dado λ a PINN converge da forma ideal, e o HPO ajuda a contornar tal dificuldade. Por fim, o tempo médio para executar um conjunto de parâmetros foi de $191, 65$ segundos. No tempo de treinamento, percebe-se um padrão já natural, em que o tempo aumenta com profundidade e largura, e também com uma baixa taxa de aprendizado.

Iteração	Taxa de Aprendizado	Profundidade	Largura	Função de Ativação	Erro	Tempo de Treinamento (s)
16	$1, 12 \times 10^{-3}$	10	120	$Swish$	$2, 64 \times 10^{-4}$	$309, 60$
3	$7, 50 \times 10^{-4}$	9	76	$Swish$	$2, 73 \times 10^{-4}$	$194, 67$
11	$2, 27 \times 10^{-3}$	10	120	$Swish$	$2, 86 \times 10^{-4}$	$309, 56$
2	$5, 20 \times 10^{-3}$	7	92	$Swish$	$3, 77 \times 10^{-4}$	$170, 50$
30	$6, 60 \times 10^{-4}$	10	120	sin	$3, 84 \times 10^{-4}$	$200, 40$
41	$2, 03 \times 10^{-3}$	8	120	$Swish$	$3, 91 \times 10^{-4}$	$246, 15$
31	$1, 53 \times 10^{-3}$	7	120	sin	$7, 34 \times 10^{-4}$	$141, 98$
4	$2, 07 \times 10^{-3}$	4	107	sin	$1, 20 \times 10^{-3}$	$81, 04$
9	$5, 11 \times 10^{-3}$	5	49	$Swish$	$1, 22 \times 10^{-3}$	$104, 71$
10	$2, 28 \times 10^{-3}$	8	114	sin	$1, 50 \times 10^{-3}$	$157, 86$

Tabela 2. Convecção-Difusão - Análise dos Hiperparâmetros.

A Figura 2 compara soluções obtidas em $t = 2\pi$ usando PINN com a solução numérica pelo método de elementos finitos obtida pela biblioteca numérica `libMesh`[7]. A solução numérica usa uma malha com 128×128 elementos quadrados lineares e um passo de tempo $\Delta t \approx 0,0157$. Para isso, utiliza-se um recorte específico da função $u(x, y, t)$ em $x = 5$. Encontram-se na figura o melhor resultado da PINN (#01) e o primeiro a atingir magnitude no erro de 10^{-1} (#34). Na legenda, o "Iter" representa a iteração no HPO, "LR" representa Taxa de Aprendizado, "Layers" número de camadas ocultas, "Nodes" número de neurônios e "Activ." é a função de ativação.

Percebe-se que a melhor solução da PINN consegue fornecer uma aproximação semelhante ao obtido usando FEM, visto que ambas as curvas estão bem sobrepostas. Isso é reforçado pela norma l^2 ter ficado com valor de apenas $2,17 \times 10^{-1}$. Em contrapartida, o outro resultado também disponível no gráfico mostra-se bem inferior ao esperado, evidenciando que os hiperparâmetros impactam fortemente na qualidade dos resultados da Rede Neural.

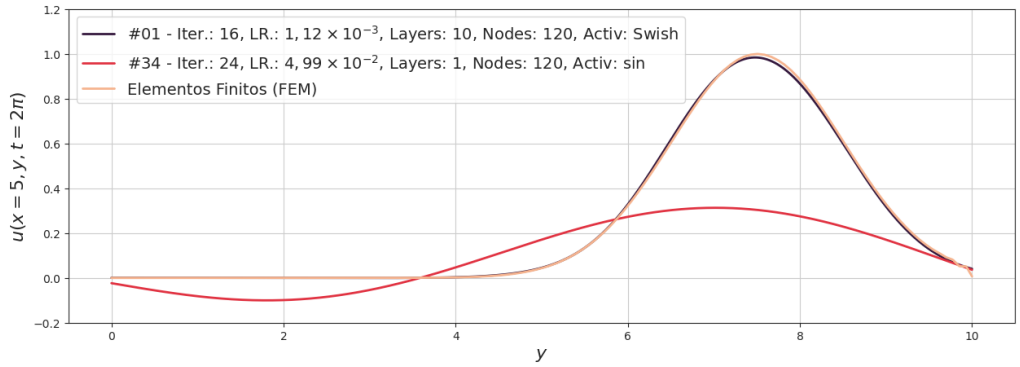

Figura 2. Comparação da PINN com FEM usando malha de 128×128 elementos.

5. Conclusões

Este trabalho propõe uma análise de otimização de hiperparâmetros em PINNs para a resolução de um problema convectivo-difusivo. Verifica-se que as Redes Neurais, quando configuradas com os hiperparâmetros corretos, conseguem aproximar esse problema com alta precisão, apresentando uma solução de qualidade comparável com a solução numérica. No entanto, novos estudos são necessários para lidar com problemas cujas soluções não sejam tão suaves e que apresentem altos gradientes. Como pesquisa futura, pretende-se investir em métodos mais refinados para calibrar os pesos das métricas da função de perda, visto que essa definição envolve uma otimização multiobjetivo. Além disso, é interessante também testar PINNs em outras equações diferenciais e até mesmo em problemas inversos para verificar outros tipos de comportamentos. Por fim, pode ser explorado a utilização de arquiteturas mais recentes, como Redes Neurais Lagrangianas [Cranmer et al. 2020], que, conforme a referência mencionada, podem aprender simetrias e leis de conservação de energia.

Agradecimentos

Os autores agradecem o apoio financeiro fornecido pela FAPERJ (APQ-01123-21) .

[7]https://libmesh.github.io/

Referências

Asrav, T. and Aydin, E. (2023). Physics-informed recurrent neural networks and hyper-parameter optimization for dynamic process systems. *Computers Chemical Engineering*, 173:108195.

Atangana, A. (2018). Chapter 3 - groundwater pollution. In Atangana, A., editor, *Fractional Operators with Constant and Variable Order with Application to Geo-Hydrology*, pages 49–72. Academic Press.

Bhaumik, B., De, S., and Changdar, S. (2024). Deep learning based solution of nonlinear partial differential equations arising in the process of arterial blood flow. *Mathematics and Computers in Simulation*, 217:21 – 36. Cited by: 2.

Cranmer, M. D., Greydanus, S., Hoyer, S., Battaglia, P. W., Spergel, D. N., and Ho, S. (2020). Lagrangian neural networks. *CoRR*, abs/2003.04630.

Cuomo, S., Cola, V. S. D., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. (2022). Scientific machine learning through physics-informed neural networks: Where we are and what's next. *CoRR*, abs/2201.05624.

Escapil-Inchauspé, P. and Ruz, G. A. (2023). Hyper-parameter tuning of physics-informed neural networks: Application to helmholtz problems. *Neurocomputing*, 561:126826.

Hansen, L. D., Stokholm-Bjerregaard, M., and Durdevic, P. (2022). Modeling phosphorous dynamics in a wastewater treatment process using bayesian optimized lstm. *Computers Chemical Engineering*, 160:107738.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. (2021). DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228.

Markidis, S. (2021). The old and the new: Can physics-informed deep-learning replace traditional linear solvers? *Frontiers in Big Data*, 4.

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

Valli, A. M. P., Catabriga, L., Santos, I. P., Coutinho, A. L., and Almeida, R. C. (2015). Predictor-multicorrector scheme for the dynamic diffusion method. In *Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 3, N. 2, 2015*.

Wang, S., Teng, Y., and Perdikaris, P. (2021). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081.

Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. `https://D2L.ai`.