O Uso de Aprendizado de Máquina na Solução de Problemas Físicos Descritos por Equações Diferenciais com Introdução de Um modelo Híbrido Envolvendo Redes Neurais

Reconhecimento de Padrões

Aluno: Yago Pereira dos Anjos Santos
Profs. Drs. Carlos Cristiano H. Borges, Raul Fonseca Neto, Saulo Moraes Villela

Universidade Federal de Juiz de Fora
Programa de Pós-graduação em Modelagem Computacional

13 de janeiro de 2026

UNIVERSIDADE
FEDERAL DE JUIZ DE FORA

Sumário

Sumário

Introdução ao Problema e Levantamento Bibliográfico

- **O problema:** A busca de soluções de EDPs para simulação de sistemas físicos (Ex.: Engenharia estrutural, dinâmica dos fluidos, etc).
- **Métodos comumente adotados:** Diferenças Finitas, Elementos Finitos, Volumes Finitos, etc.
- **Motivação:** Desenvolvimento de modelos generalizáveis que estejam em conformidade com a física do problema e que utilizem dados de maneira eficiente.
- **Objetivo:** Introdução do modelo híbrido FEM-NN (Combinação Elementos Finitos + Redes Neurais).

- Uma treliça é governada pela equação $\frac{d}{dx}\left[AE\frac{du(x)}{dx}\right] = 0$, onde A é a área da seção transversal E é o módulo de elasticidade e u é o deslocamento.
- O equilíbrio de forças é dado por $AE\frac{du(x)}{dx} = T$, onde T representa a força axial sobre a treliça.
- Uma estrutura de treliça com n barras é resolvido com a união da forma discretizada de cada membro para formar o sistema completo de equações.

- Finite element method-enhanced neural network for forward and inverse problems, Meethal et al. [4]
- Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems, Laura et al. [8].
- Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, Raissi et al. [7].
- On automatic differentiation and algorithmic linearization, Griewank et al. [2].
- Resultados promissores em ciência computacional para diversos problemas: transferência de calor [1], sistemas de potência [5], EDPs fracionárias [6].

Sumário

Problema modelo:

$$-\frac{d^2u}{dx^2} = f(x), \quad x \in [0, 2].$$

Condições de contorno:

- Dirichlet: adequadas para solução analítica $u(0) = a$, $u(b) = b$.
- Representa fenômenos de difusão/equilíbrio.

Dois casos de estudo:

1. Termo fonte $f_1(x) = 6(x-1)^5 + 2(x-1) \rightarrow$ Solução polinomial, $a = \frac{10}{21}$ e $b = -\frac{10}{21}$.
2. Termo fonte $f_2(x) = -2\pi^2 \cos(2\pi x) \rightarrow$ Solução oscilatória, $a = b = 0$.

Dataset 1: Solução Polinomial

$$u_1(x) = -\frac{(x-1)^7}{7} - \frac{(x-1)^3}{3}$$

Características:

- Comportamento suave
- Monotônico
- Fácil para interpolação
- $n = 1000$ amostras uniformes em [0,2]

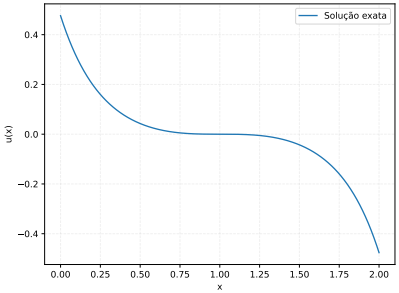


Figura 1: Gráfico da solução exata $u_1(x)$.

Dataset 2: Solução Oscilatória

$$u_2(x) = \sin^2(\pi x)$$

Características:

- Múltiplos picos/vales
- Alta frequência
- Testa capacidade de aproximação
- $n = 1000$ amostras uniformes em [0,2]

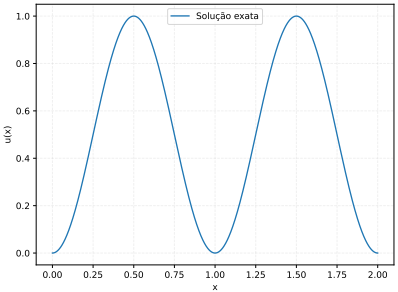


Figura 2: Gráfico da solução exata $u_2(x)$.

Mesmo domínio, comportamento mais complexo

Configuração Experimental - Etapa 1

Metodologia

- **Seed fixa:** 42 (reprodutibilidade)
- **Validação:** validação cruzada com 10-fold
- **Métricas:** MSE, MAE, R^2
- **Algoritmos:** SVR, Random Forest, XGBoost + Ensemble

Versões das bibliotecas utilizadas

- **Scikit-learn v1.7.2:** engloba funções KFold, cross_validate, SVR, RandomForestRegressor e VotingRegressor.
- **XGBoost v3.1.2:** engloba a função XGBRegressor.
- **Numpy v2.3.5**, **Pandas v2.3.3**, **Matplotlib v3.10.6**.

SVR: Support Vector Regression

Ideia Principal:

- Encontrar função que se desvie no máximo ϵ dos dados.
- Minimizar norma dos pesos + penalizar violações.
- Kernel trick para não-linearidade (Radial Basis Function - RBF).

Parâmetro	Valor	Função	Escolha
`kernel`	`rbf`	Mapeia para espaço não-linear	Padrão para problemas complexos
`C`	100	Penalidade de violações	Balancear precisão $\times$ generalização
`gamma`	0.1	Influência de cada exemplo	Não muito pequeno (evitar underfit)
`epsilon`	0.1	Margem de erro aceitável	Valor padrão (scikit-learn)

Tabela 1: Parâmetros de configuração do algoritmo SVR.

Random Forest: Ensemble por Bagging

Ideia Principal:

- Múltiplas árvores de decisão (floresta)
- Cada árvore treina com:
  - Bootstrap dos dados (bagging)
  - Subconjunto aleatório de features
- Previsão final: média das árvores

Parâmetro	Valor	Função	Escolha
`n_estimators`	100	Número de árvores	Bom custo-benefício
`max_depth`	`None`	Profundidade máxima	Cresce até pureza (controlado por outras restrições)
`min_samples_split`	2	Mínimo para dividir	Valor padrão
`random_state`	42	Seed para reprodutibilidade	Fixado para comparar igualmente

Tabela 2: Parâmetros de configuração utilizados para o algoritmo Random Forest.

XGBoost: Gradient Boosting Otimizado

Ideia Principal:

- Sequência de árvores fracas (boosting)
- Cada árvore corrige erros da anterior
- Minimização de loss com gradiente
- Implementação otimizada (sklearn)

Parâmetro	Valor	Função	Escolha
`n_estimators`	100	Número de árvores	Balancear tempo $\times$ performance
`learning_rate`	0.1	Taxa de aprendizado	Valor conservador (evitar overfitting)
`max_depth`	6	Profundidade máxima	Limitar complexidade (árvores fracas)
`subsample`	0.8	Fração de amostras	Regularização por aleatorização
`random_state`	42	Seed para reprodutibilidade	Comparação justa entre métodos

Tabela 3: Parâmetros de configuração do algoritmo XGB.

No contexto de ensemble com ponderação:

- Definição de pesos para os algoritmos: (SVR, RF, XBG) = (1, 2, 2).

Modelo	**MSE**	**MSE stdev**	**MAE**	**MAE stdev**	**R^2**
SVM	4.959891×10^{-3}	5.256133×10^{-4}	0.063255	0.004676	0.796004
RF	7.333740×10^{-7}	2.776495×10^{-7}	0.000520	0.000094	0.999972
XGB	5.756622×10^{-6}	2.133678×10^{-6}	0.001373	0.000293	0.999780
Ensemble	5.522621×10^{-4}	5.794388×10^{-5}	0.021101	0.001565	0.977299
Ens. Ponderado	1.998666×10^{-4}	2.066300×10^{-5}	0.012681	0.000941	0.991791

Tabela 4: Resultados para o contexto polinomial usando comitês com e sem ponderação.

Resultados para o contexto do termo fonte oscilatório

Modelo	MSE	MSE stdev	MAE	MAE stdev	R^2
SVM	2.776495×10^{-1}	1.275787×10^{-2}	0.278407	0.018540	0.086981
RF	6.852028×10^{-6}	1.005624×10^{-6}	0.002163	0.000143	0.999943
XGB	5.751968×10^{-5}	1.258333×10^{-5}	0.005680	0.000623	0.999522
Ensemble	1.239357×10^{-2}	1.430318×10^{-3}	0.092913	0.006272	0.898451
Ens. Ponderado	4.472686×10^{-3}	5.195076×10^{-4}	0.055846	0.003815	0.963355

Tabela 5: Resultados para o contexto oscilatório usando comitês com e sem ponderação.

Comparativo de Modelos: Ensemble vs. Modelos Base

Gráfico comparativo entre os modelos base e Ensemble não poderado

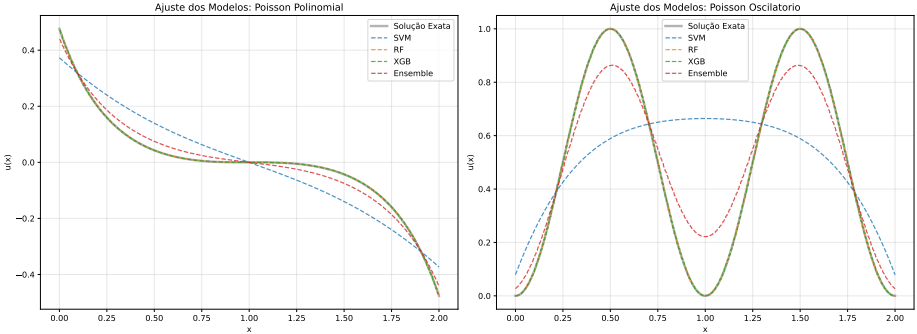


Figura 3: Plotagem dos resultados a partir do ajuste dos modelos. Dataset 1 à esquerda, dataset 2 à esquerda.

Comparativo de Modelos: Ensemble Ponderado vs. Modelos Base

Gráfico comparativo entre os modelos base e Ensemble poderado

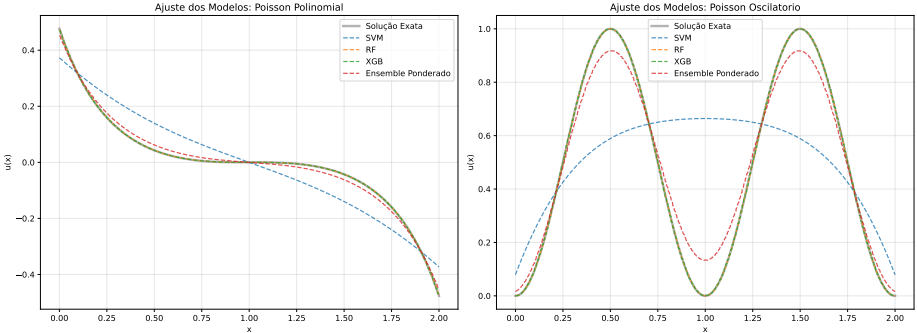


Figura 4: Plotagem dos resultados a partir do ajuste dos modelos. Dataset 1 à esquerda, dataset 2 à esquerda.

Sumário

Etapa 2: Modelo de Linguagem (Gemini) como Regressor

Objetivo: Avaliar a capacidade do modelo de linguagem Gemini na tarefa de regressão matemática, comparando estratégias *Zero-Shot* e *Few-Shot*.

Configuração experimental:

- **Mesmos subconjuntos** da Etapa 1 (10-fold, 1º fold, 10 amostras de teste)
- **Modelo**: `models/gemini-2.5-flash` (versão disponível via API)
- **Prompt base** (Zero-Shot):
  `"Atue como um regressor matemático preciso. Tarefa: Estimar u(x) para uma [descrição]. x = [valor]. Responda APENAS o número decimal."`
- **Few-Shot**: Adiciona 5 exemplos (x, u(x)) extraídos do conjunto de treino
- **Mapeamento de saída**: Extração numérica via regex e conversão para `float`

Dataset	Estratégia	MSE	MSE stdev	MAE	MAE stdev	R^2
Polinonial	Zero-Shot	0.104612	0.036368	0.318875	0.054135	-30.116766
Polinomial	Few-Shot	0.106322	0.039850	0.320875	0.057982	-30.625659
Oscilatório	Zero-Shot	0.014439	0.012014	0.102189	0.063218	-2.612955
Oscilatório	Few-Shot	0.014439	0.012014	0.102189	0.063218	-2.612955

Tabela 6: Resultados obtidos com o modelo de linguagem Gemini.

Observações:

- Resultados mostram limitações do LLM na regressão precisa (R^2 negativo)
- Pouca diferença entre Zero-Shot e Few-Shot
- Custo computacional e tempo elevados (limitações de quota da API)

Sumário

Introdução do Modelo Híbrido FEMM-NN

Arquitetura da Rede Neural (Fully Connected)

- **Entrada:** 1 neurônio (coordenadas espaciais x).
- **Camadas ocultas:** $n\_hidden = 10$ neurônios, repetidas $n\_layer = 3$ vezes.
- **Ativação:** Tanh, inicialização Xavier.
- **Saída:** 1 neurônio ($u(x)$).

Funções de Perda

1. **PINN (Physics-Informed Neural Network)**
   - Resíduo da EDP + condições de contorno.
   - Derivadas calculadas via autograd do **PyTorch v2.9.1+cu130**.
2. **FEM-NN (Finite Element Method-enhanced Neural Network)**
   - Utiliza a formulação fraca do Método dos Elementos Finitos (FEM)
   - Minimiza o resíduo do sistema FEM ($Au - F$).

O Modelo Híbrido FEM-NN

Estratégias de Treinamento

- Otimizador: Adam (lr = 0.0003).
- **PINN:** Minimiza perda física e de contorno.
- **FEM-NN:** Minimiza erro da discretização FEM.

Limitação PINN:

- Faz com que problemas físicos se tornem mal condicionados devido a regularização suave [3].

Vantagem FEM-NN

- Preserva a boa condição física do problema usando diretamente as matrizes após aplicação das condições de contorno.

O Modelo Híbrido FEM-NN

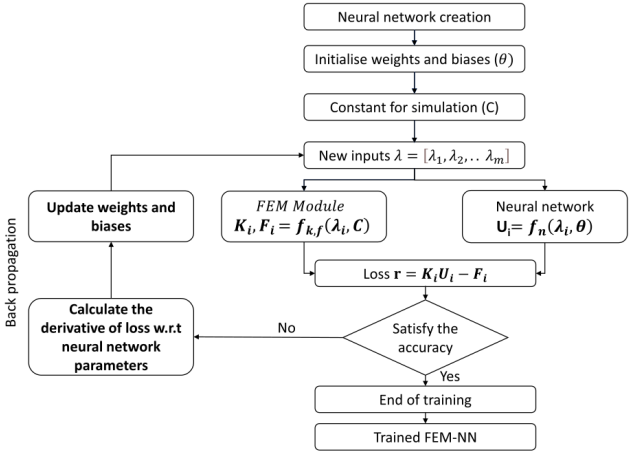


Figura 5: Fluxograma detalhando o uso da função perda baseada em FEM. Adaptado de [4].

Dataset 1 - Fonte Polinomial

Biblioteca PyTorch com seed fixo 43

Para montagem da matriz FEM:

- Base de Lagrange linear.
- Número de elementos: 9.
- Total de nós: 10 (usados como pontos de colocação para PINN).

Treinamento em épocas

- Inicialmente 1900 épocas, depois para 2200 épocas.
- Comparação das perdas ao longo das épocas para PINN e FEM-NN.
- Plotagem comparativa das aproximações para malhas mais refinadas.

Dataset 1 - Fonte Polinomial

Comparação das Perdas entre PINN e FEM-NN

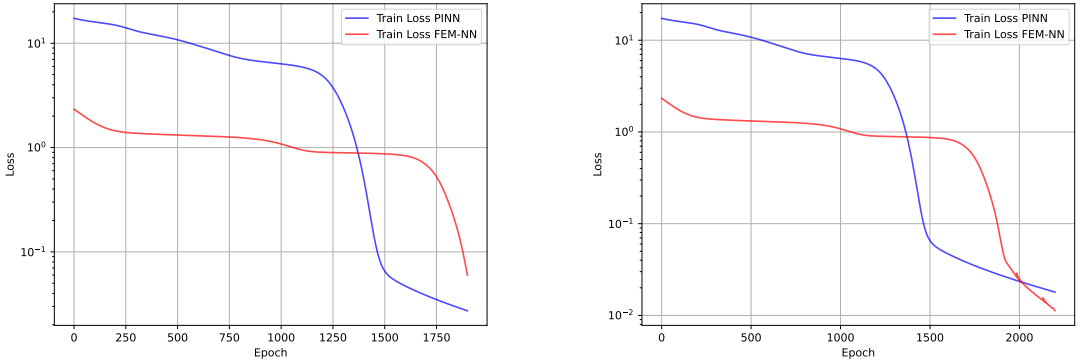
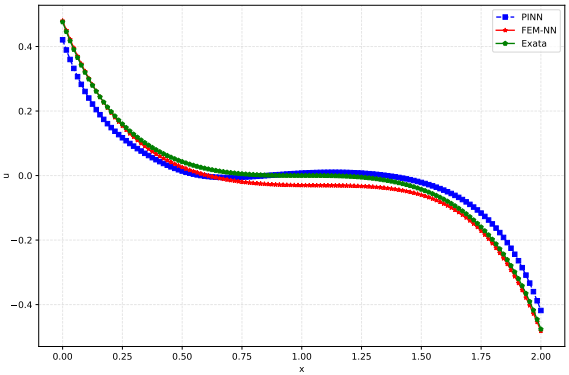


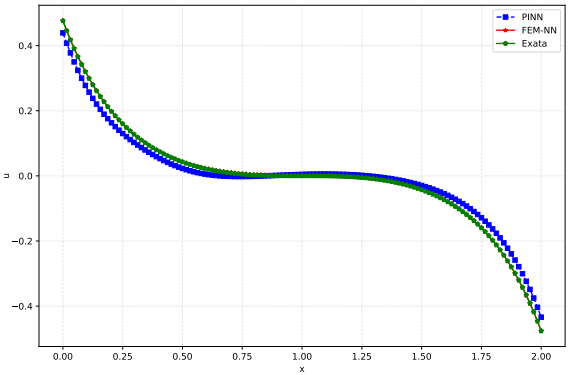
Figura 6: Comparativo de evolução das perdas: (a) 1900 épocas; (b) 2200 épocas.

Dataset 1 - Fonte Polinomial

Comparativo de convergência das redes



(a)



(b)

Figura 7: Comparando convergência das redes para uma malha de 129 elementos: (a) 1900 épocas; (b) 2200 épocas.

Dataset 2 - Fonte Oscilatória

Mesma configuração de arquitetura de rede

Para montagem da matriz FEM:

- Base de Lagrange de ordem 4.
- Número de elementos: 9.
- Total de nós: 37 (usados como pontos de colocação para PINN).

Treinamento em épocas

- Inicialmente 3000 épocas, depois para 5000 épocas.
- Comparação das perdas ao longo das épocas para PINN e FEM-NN.
- Plotagem comparativa das aproximações para malhas mais refinadas.

Dataset 2 - Fonte Oscilatória

Comparação das Perdas entre PINN e FEM-NN

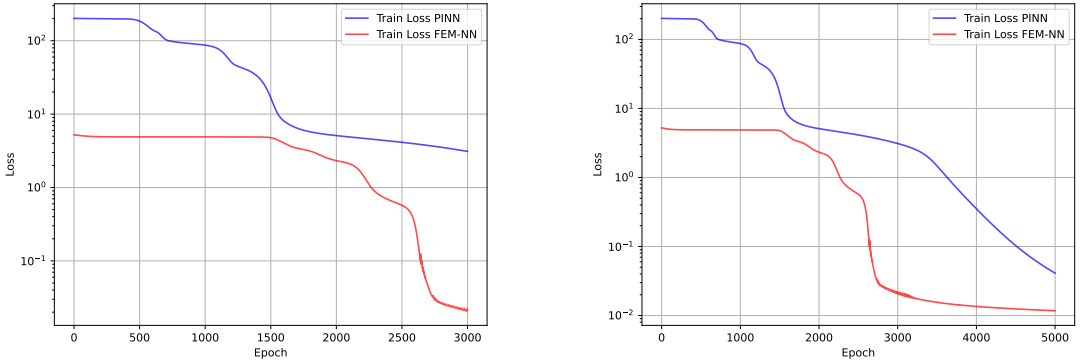
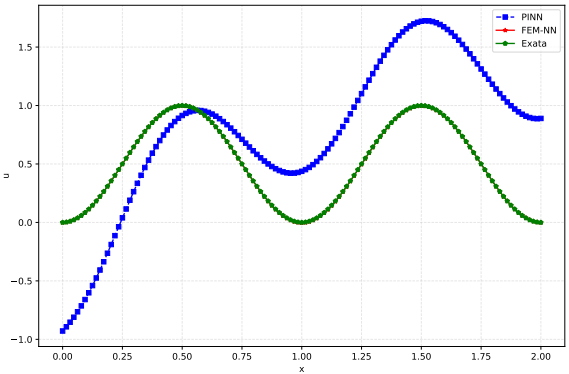


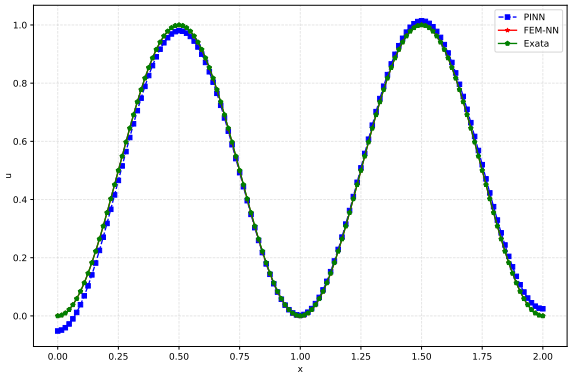
Figura 8: Comparativo de evolução das perdas: (a) 3000 épocas; (b) 5000 épocas.

Dataset 2 - Fonte Oscilatória

Comparativo de convergência das redes



(a)



(b)

Figura 9: Comparando convergência das redes para uma malha de 129 elementos: (a) 3000 épocas; (b) 5000 épocas.

Dataset 1: termo fonte polinomial (2200 épocas)

Modelo	MSE	MAE	ER	R^2
PINN	5.537601×10^{-4}	1.939084×10^{-2}	1.419372×10^{-1}	0.979854
FEM-NN	7.647131×10^{-7}	7.190360×10^{-4}	5.274542×10^{-3}	0.999972

Tabela 7: Métricas para o modelo polinomial para 2200 épocas.

Dataset 2: termo fonte oscilatório (5000 épocas)

Modelo	MSE	MAE	ER	R^2
PINN	4.908194×10^{-4}	1.828439×10^{-2}	3.631910×10^{-2}	0.996103
FEM-NN	3.202030×10^{-7}	4.121474×10^{-4}	9.276560×10^{-4}	0.999997

Tabela 8: Métricas para o modelo oscilatório para 5000 épocas.

Sumário

Conclusão e Próximos Passos

Modelos Clássicos

- Requerem número maior de dados para aprendizado.
- RF e XGB apresentam desempenho superior ao modelo SVR.
- Levam em consideração os dados puros e não a física do problema.

Modelo GPT

- Não é adequado para o tipo de problema proposto, apresentando desempenho inferior ao dos modelos clássicos, inclusive SVR.
- Alto custo computacional (tempo) em relação ao baseline.

Conclusão e Próximos Passos

Modelo Híbrido FEM-NN

- Resolve problema de mal-condicionamento da física ocorrida na PINN.
- Menos dados necessários para treinar o modelo.
- Apredinzado em conformidade com a física do problema.
- Natureza empírica da seleção da arquitetura da rede neural pode levar a consumo de tempo de pesquisa.
- Necessidade de investigação para problemas não lineares.

Próximos Passos

- Incluir o modelo FEM-NN para pesquisa de doutorado na sua versão para resolver problemas inversos: Avaliação de danos estruturais.

Referências I

[1] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

[2] Andreas Griewank. On automatic differentiation and algorithmic linearization. *Pesquisa Operacional*, 34(3):621–645, 2014.

[3] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

[4] Rishith E Meethal, Anoop Kodakkal, Mohamed Khalil, Aditya Ghantasala, Birgit Obst, Kai-Uwe Bletzinger, and Roland Wüchner. Finite element method-enhanced neural network for forward and inverse problems. *Advanced modeling and simulation in engineering sciences*, 10(1):6, 2023.

Referências II

[5] George S Misyris, Andreas Venzke, and Spyros Chatzivasileiadis. Physics-informed neural networks for power systems. In *2020 IEEE power & energy society general meeting (PESGM)*, pages 1–5. IEEE, 2020.

[6] Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

[7] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[8] Laura Von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, et al. Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):614–633, 2021.

Agradecimentos

Muito Obrigado!