XLIII
CILAMCE
21-25, November, 2022
FOZ DO IGUAÇU - PR - BRAZIL

Evaluating Different Neural Networks Architectures for the Solution of Heat Conduction Problems in NVIDIA Modulus

Felipe M. Eler[1], Anaximandro A. P. M. de Souza[1], Paulo Couto[1], Alvaro L. G. A. Coutinho[1]

[1]*Dept. of Civil Engineering, COPPE/Federal University of Rio de Janeiro*
Ilha do Fundão - R. Moniz Aragão, Nº 360 - Bloco 4 - Cidade Universitária, 21941-594, RJ, Brazil
felipe.eler@petroleo.ufrj.br, anaximandro@nacad.ufrj.br, pcouto@petroleo.ufrj.br, alvaro@nacad.ufrj.br

Abstract. The use of neural networks to address engineering problems is increasing considerably. A limitation of using neural networks is the need for large amounts of data to fit nonlinear problems with acceptable accuracy. An alternative to the purely data-driven approach is the physics-informed neural networks, which add physical constraints that significantly reduce the amount of data needed to achieve acceptable accuracy. In this work, we solve two simple heat conduction problems using PINNs, evaluating the complexity of different neural network architectures. A direct comparison to the analytical solution proved the PINNs to be good solvers for the evaluated partial differential equations. A fully connected neural network (FCN) handles the problem well for the steady-state case. However, a gated recursive unit (GRU) architecture is needed to solve a transient problem. For both problems, an architecture of 6 layers with 64 units each is sufficient to achieve good results.

Keywords: Physics-informed neural network (PINN), Heat conduction, Machine learning

1 Introduction

Machine Learning (ML) has been a topic of increasing interest in recent decades. According to Evsukoff [1], this growth is related to the increasing amount of data being stored digitally instead of analogically, the increasing processing power, the broader internet access, and new paradigms such as the multiple layers error propagation [2] and deep neural networks [3].

Physics-Informed Neural Networks (PINN) are a particular category of ML where physical restrictions are added to the loss function instead of a purely data-driven approach. According to Karniadakis et al. [4], numerical discretization of partial differential equations (PDEs) is still complex and prohibitively expensive, while merely data-driven machine learning may not find enough training data to achieve the required precision. In this context, PINN can use available data and enforce the physical laws to improve the predictions of a neural network (NN).

In this work, we used NVIDIA Modulus [1] to solve heat transfer problems and to test the NN architecture's effectiveness in solving transient problems. According to Nvidia, Modulus is a neural network framework that blends the power of physics in the form of governing partial differential equations (PDEs) with data to build high-fidelity, parameterized surrogate models with near-real-time latency. Modulus trains its NN trying to satisfy the PDEs and its boundary conditions. If the network can minimize this loss function, then it will, in effect, solve the given differential equation [5].

2 Heat conduction problems

We tackle two cases in the present work. The cases are presented in increasing order of complexity:

- 2D Steady State Heat Transfer in a Solid;
- 1D Transient Heat Transfer in a Semi-Infinite Solid.

[1]https://developer.nvidia.com/modulus

2.1 2D Steady State Heat Transfer in a Solid

The first case describes the 2D heat conduction in a thin rectangular plate or a long rectangular rod (Bergman et al. [6]. Figure 1 shows the geometry schematic, where the rectangle's sides and bottom surfaces are kept at a temperature $T_1$and the top surface is kept at a temperature T_2, higher than T_1.

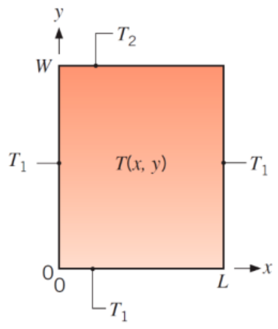

Figure 1. Two-dimensional conduction in a thin rectangular plate. Adapted from Bergman et al. [6]

The governing equation for this problem is the two-dimensional, steady-state heat conduction with no generation and constant thermal conductivity for rectangular coordinates, which takes the following form:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{1}$$

and Dirichlet boundary conditions:

$$T(0, y) = T_1, \quad T(L, y) = T_1, \quad T(x, 0) = T_1, \quad T(x, W) = T_2 \tag{2}$$

For this problem, $L = H = 1m$, $T_1 = 50°C$ and $T_2 = 150°C$ The analytical solution (Equations 3 and 4) is used to create the validation domain. The first 100 terms of the summation are considered.

$$\theta = \frac{T - T_1}{T_2 - T_1} \tag{3}$$

$$\theta = \frac{2}{\pi} \sum_{n=1}^{\infty} \frac{(-1)^{n+1} + 1}{n} \sin \frac{n\pi x}{L} \frac{sinh(n\pi y/L)}{sinh(n\pi W/L)} \tag{4}$$

2.2 1D Transient Heat Transfer in a Semi-Infinite Solid

The second case describes the 1D heat conduction in a semi-infinite solid from Bergman et al. [6]. This solid extends to infinity in all but one direction and is characterized by a single identifiable surface (Figure 2).

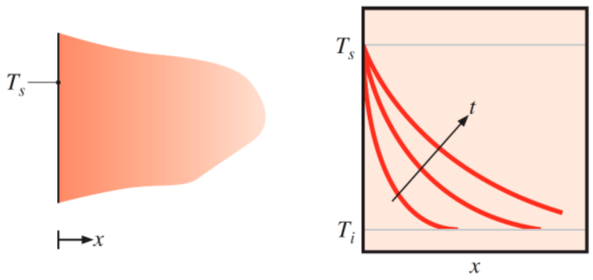

Figure 2. Semi-infinite solid schematic and transient temperature distributions. Adapted from Bergman et al. [6]

The governing equation for this problem is the one-dimensional, transient heat conduction with no generation and constant thermal conductivity for rectangular coordinates, which takes the following form:

$$\frac{1}{\alpha} \frac{\partial T}{\partial t} - \frac{\partial^2 T}{\partial x^2} = 0 \tag{5}$$

The initial condition is:

$$T(0, x) = T_i \tag{6}$$

The boundary conditions are of the first type:

$$T(t, 0) = T_s, \quad T(t, \infty) = T_i \tag{7}$$

Despite being an idealized geometry, Bergman et al. [6] highlights that the semi-infinite solid has several practical applications. For our problem, $L = 0.1m$, $T_s = 25°C$ and $T_i = 100°C$. The analytical solution (Equation 8) is used to create the validation domain.

$$\frac{T(t, x) - T_s}{T_i - T_s} = erf\left(\frac{x}{2\sqrt{\alpha t}}\right) \tag{8}$$

2.3 Physics-Informed NN Modeling

In the first problem, we tested four PINNs to model the problem. The difference between those PINNs is the architecture, which followed the logic of n layers with 2^n neurons each. The networks were fully connected, and we kept all the other parameters the same; many of them were Modulus defaults:

Table 1. Summary of the PINN architecture for the first case.

Hyper-parameter	Used
Solver	Adam
Activation function	Swish
Initial Learning Rate	10^{-3}
Decay Type	Exponential
Decay Rate	0.95
Number of Steps to Decay	8×10^3
Maximum Steps	3×10^5
Architecture	n layers and 2^n neurons

In Modulus, one must choose how many points she/he wants to sample in the geometry interior and on each boundary. We choose 100 points in each boundary and 4000 points in the geometry interior. As aforementioned, Modulus does not use external data in the loss function, it only tries to minimize the PDE residual; for this case:

$$L = \frac{1}{n}\sum_{i}^{n}\left(\left[\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right] T_{PINN}(x_i, y_i)\right)^2 \tag{9}$$

The second problem adds a bit of complexity by introducing transient effects. Four fully connected PINNs (FCN) were used to model the problem, but the results were unsatisfactory. We tried to increase the architecture's complexity to match the problem complexity. We tested four PINNs with recurrent neural networks (RNN) architecture and four PINNs with gated recurrent unit (GRU) architecture, totalizing 16 PINNs.

For each architecture class, we tested four PINNs following the logic of n layers with 2^n neurons each. We kept all other parameters the same; many of them are Modulus defaults (Table 1). All layers of the RNN are recursive, and a fixed number of 3 recursive layers are used in the GRU. Both GRU and RNN divided the time domain into ten steps. For this case, we choose 50 points in each boundary and initial condition and 100 points in the geometry interior. The loss is as follows:

$$L = \frac{1}{n}\sum_{i}^{n}\left(\left[\frac{\partial}{\partial t} + \frac{\partial^2}{\partial x^2}\right] T_{PINN}(t_i, x_i)\right)^2 \tag{10}$$

3 Results

In this section, we present and discuss the main results. The criteria used to choose the best architecture for each problem were the loss and the relative error of the validation domain.

3.1 2D Steady state heat transfer in a solid results

Table 2 presents the performance of each architecture tested. Figures 3 and 4 present the evolutions of the loss function and of the validation domain error, respectively. For this case, the 6x64 PINN presented the best results.

Table 2. Final loss and validation domain relative error for the first case.

Architecture Size	Loss after 3×10^5 steps	Validation domain relative error
4x16	34.83	0.08361
5x32	21.39	0.06448
6x64	10.51	0.04698
7x128	33.95	0.04563

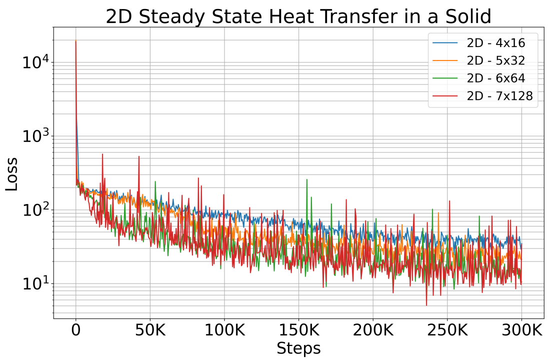

Figure 3. Loss for 2D Steady State heat transfer in a solid

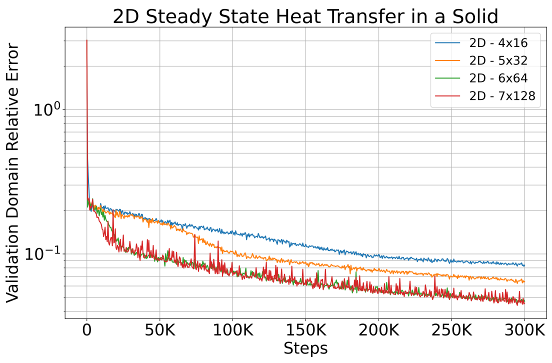

Figure 4. Validation domain error for 2D Steady State heat transfer in a solid

Another interesting plot is shown in Figure 5 for the 6x64 PINN, where we compare the Modulus solution to the analytical solution. It is possible to observe small differences in the majority of the domain (98.2% of the domain has absolute differences smaller than $0.2°C$). The higher differences are present at (x, y) equal to $(0, W/2)$ and $(L/2, W/2)$, due to the singularity in the boundary conditions. To speed convergence, higher weights were given to the center $(x = 0)$ and smaller eights were given as $abs(x) \to L/2$ (close to the singularities).

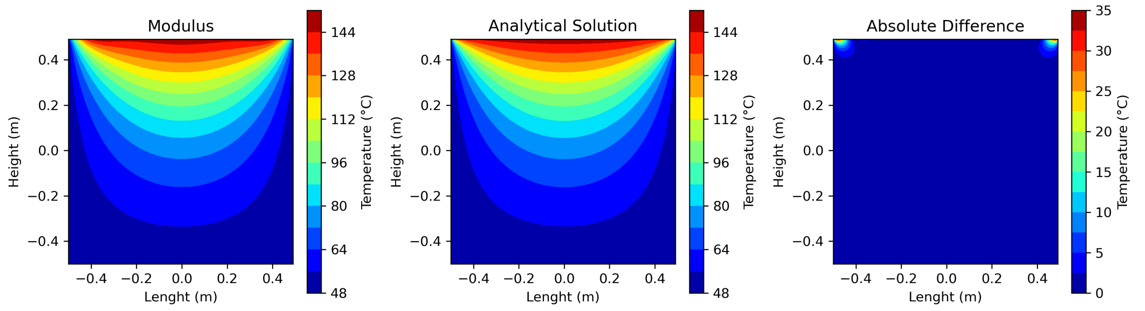

Figure 5. Results for the 6x64 PINN

3.2 1D Transient heat transfer in a semi-infinite solid results

Table 3 presents the performance of each architecture evaluated. Figures 6 to 11 presents the evolutions of the loss function and of the validation domain relative error evolution for each architecture (FCN, RNN, and GRU). It is worth noting that all cases had the same random seed to avoid initialization biases.

Table 3. Final loss and validation domain relative error for the second case.

Architecture	Loss after 3×10^5 steps			Validation domain relative error		
	FCN	RNN	GRU	FCN	RNN	GRU
4x16	0.7492	0.03165	0.4127	0.1438	0.03952	0.03413
5x32	0.6209	0.184	0.08192	0.07279	0.02206	0.0161
6x64	0.02452	0.1918	0.01617	0.02215	0.01905	9.75E-03
7x128	0.01934	0.07261	0.05803	0.01331	0.01521	0.01746

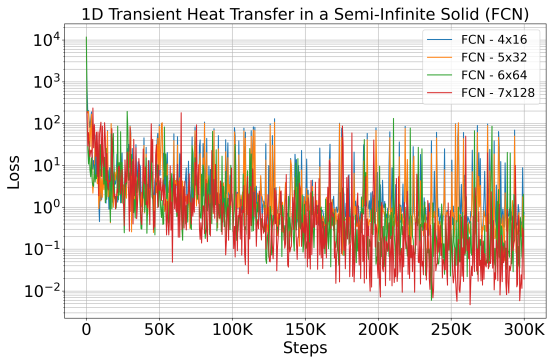

Figure 6. Loss evolution for the different FCNs

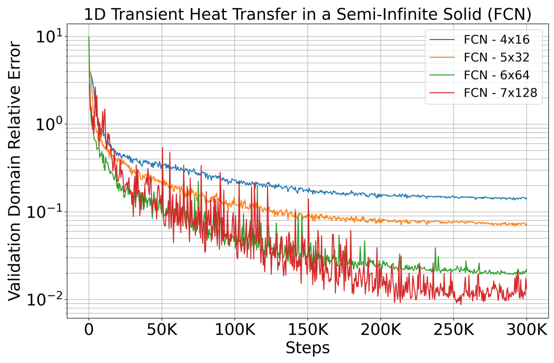

Figure 7. Validation Domain Relative Error evolution for the different FCNs

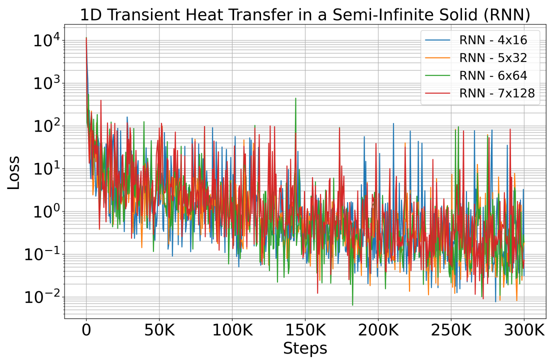

Figure 8. Loss evolution for the different RNNs

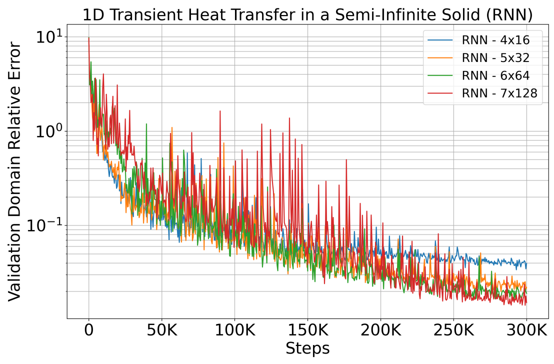

Figure 9. Validation Domain Relative Error evolution for the different RNNs

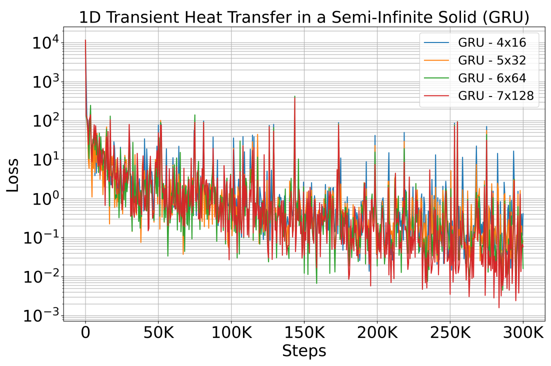

Figure 10. Loss evolution for the different GRUs

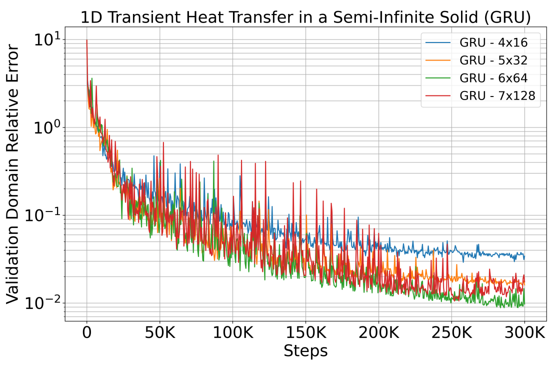

Figure 11. Validation Domain Relative Error evolution for the different GRUs

We compare Modulus prediction to the analytical solution for the three classes of PINNs using the 6x64 size. Figure 12 presents the analytical solution to the problem. Figures 13, 14 and 15 presents the results obtained for FCN, RNN, and GRU, respectively. GRU performs the best for the same network size, followed by RNN and FCN. GRU presents the smallest loss and the smallest relative error in the validation domain. Figures 13, 14, and 15 show that the minimum and maximum differences decrease as we walk from FCN to GRU. Table 4 presents the metrics used to quantify the differences. The FCN and the RNN could not eliminate differences in the region with the steepest gradients.

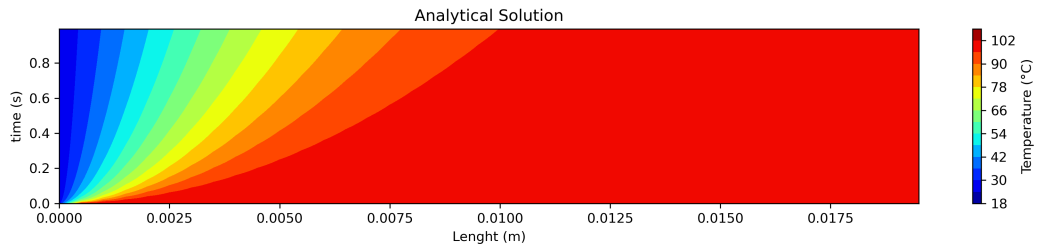

Figure 12. Analytical solution for 1D Transient heat transfer in a semi-infinite solid. This image is a magnification of the domain region with the steepest gradients.

Table 4. Metrics to compare the PINN classes.

Architecture	Minimum Difference ($^\circ C$)	Maximum Difference ($^\circ C$)	% of the domain with difference smaller than $0.2^\circ C$
FCN 6x64	-6.76	17.41	89.1%
RNN 6x64	-9.74	12.79	80.0%
GRU 6x64	-9.94	1.91	99.1%

It is possible to observe that there is a small gain when comparing FCN to RNN, but there are significant improvements when changing to GRU.

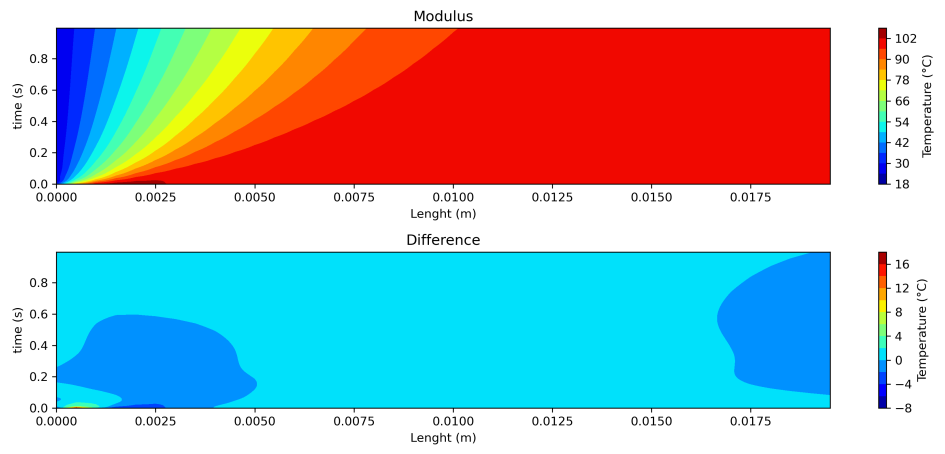

Figure 13. Results for the 6x64 FCN: a) Modulus Prediction; b) Difference. This image is a magnification of the domain region with the steepest gradients.

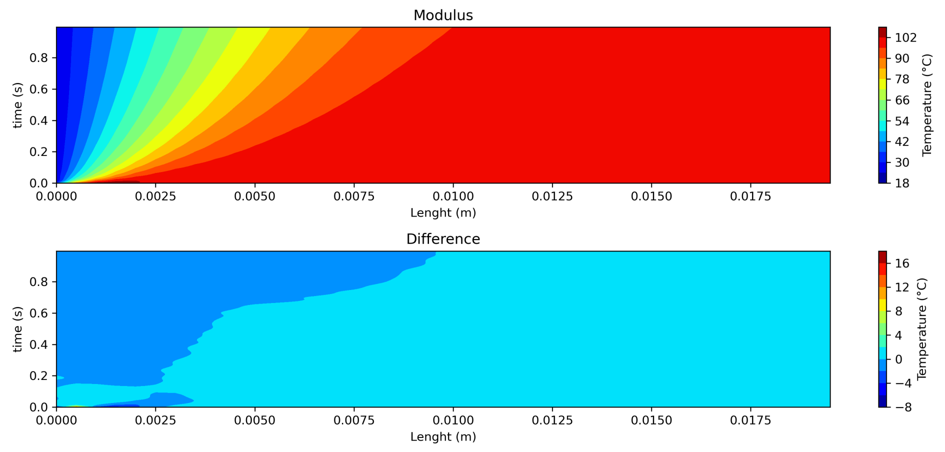

Figure 14. Results for the 6x64 RNN: a) Modulus Prediction; b) Difference. This image is a magnification of the domain region with the steepest gradients.

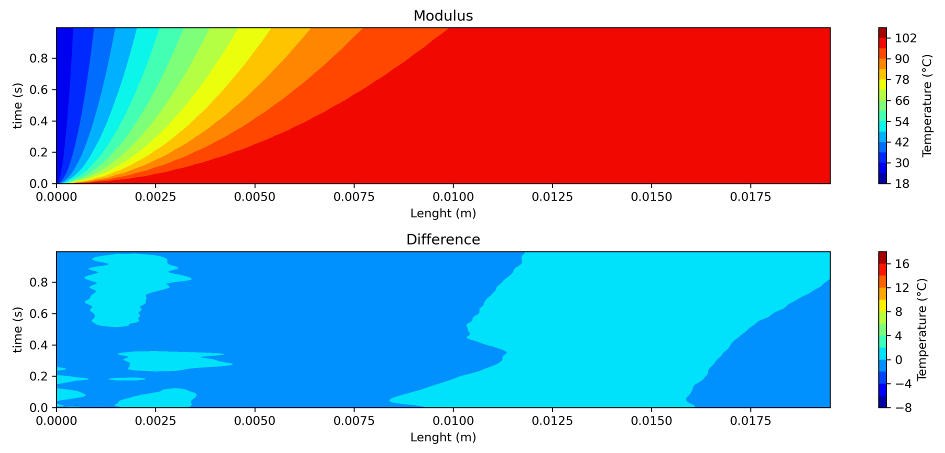

Figure 15. Results for the 6x64 GRU: a) Modulus Prediction; b) Difference. This image is a magnification of the domain region with the steepest gradients.

4 Conclusions

In this work, we solved two heat transfer problems with increasing complexity. In both cases, NVIDIA Modulus proved to be a good solver. It is worth noting that the Modulus approach to solve PINNs does not need data for forward problems. For steady-state problems, the conventional fully connected NN (FCN) proved to be enough to achieve good precision. For transient problems, results suggest that more advanced NN classes/types should be tested. The GRU PINN has the best results for the particular problem in this work. For the two problems, a 6x64 PINN is an optimal architecture with no further improvement migrating to a 7x128 PINN.

Acknowledgements. The authors gratefully acknowledge sponsorship from Shell Brasil Petróleo Ltda with the ongoing RD project LRAP (Laboratório de Recuperação Avançada de Petróleo) at UFRJ via the Shell Brasil sponsored RD project ANP nº 20163-2, "Análise Experimental da Recuperação de Petróleo para os Carbonatos do Pré-Sal do Brasil através de Injeção Alternada de CO2 e Água" (UFRJ/Shell Brasil/ANP) all made possible by the support of the ANP through the RD levy as "Compromisso de Investimentos com Pesquisa e Desenvolvimento". The authors also would like to thank ANP for the support via PRH-ANP scholarship. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES)—Finance Code 001. This work is also partially supported by CNPq, FAPERJ.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] A. G. Evsukoff. *Inteligência Computacional: Fundamentos e aplicações*, 2020.
[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, vol. 323, n. 6088, pp. 533–536, 1986.
[3] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`, 2016.
[4] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, vol. 3, n. 6, pp. 422–440, 2021.
[5] O. Hennigh, S. Narasimhan, M. A. Nabian, A. Subramaniam, K. Tangsali, Z. Fang, M. Rietmann, W. Byeon, and S. Choudhry. Nvidia simnet™: An ai-accelerated multi-physics simulation framework. In *International Conference on Computational Science*, pp. 447–461. Springer, 2021.
[6] T. L. Bergman, T. L. Bergman, F. P. Incropera, D. P. Dewitt, and A. S. Lavine. *Fundamentals of heat and mass transfer*. John Wiley & Sons, 2011.