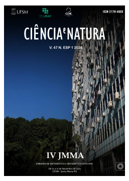
CIÊNCIAeNATURA

ISSN 2179-460X
OPEN ACCESS

Ci. e Nat., Santa Maria, v. 47, spe. 1, e89888, 2025• https://doi.org/10.5902/2179460X89888
Submitted: 11/22/24 • Approved: 11/29/24 • Published: 01/15/25



Mathematics

Burgers' PINNs with transfer learning by θ-scheme

PINNs para resolução da equação de Burgers com transferência de aprendizagem pelo esquema θ

Vitória Biesek[I], **Pedro Henrique de Almeida Konzen**[I]

[I]Universidade Federal do Rio Grande do Sul, RS, Brazil

ABSTRACT

The Burgers equation is a well-established test case in the computational modeling of several phenomena, such as fluid dynamics, gas dynamics, shock theory, cosmology and others. In this work, we present the application of physics-informed neural networks (PINNs) with a transfer learning approach using the θ-scheme to solve the Burgers' equation. The proposed approach consists of searching for a discrete solution in time through a sequence of artificial neural networks (ANNs). At each time step, the previous ANN transfers its learning to the next network model, which learns the solution in the current time by minimizing a loss function based on the θ-scheme approximation of the Burgers' equation. To test this approach, we present its application to two benchmark problems with known analytical solutions. Compared to usual PINN models, the proposed approach has the advantage of requiring smaller neural network architectures with similar accurate results and potentially decreasing computational costs.

Keywords: Burgers' equation; Physics-informed neural network; Explicit Euler method; Implicit Euler method; Crank-Nicolson method

RESUMO

A equação de Burgers é um caso de teste bem estabelecido na modelagem computacional de diversos fenômenos, como dinâmica de fluidos, dinâmica de gases, teoria do choque, cosmologia e outros. Neste trabalho, apresentamos a aplicação de rede neural informada pela física (PINNs, do inglês, *physics-informed neural networks*) com uma abordagem de transferência de aprendizagem pelo esquema θ para resolver a equação de Burgers. A abordagem proposta consiste em buscar uma solução discreta no tempo por meio de uma sequência de redes neurais artificiais (ANNs, do inglês, *artificial neural networks*). A cada passo de tempo, a ANN anterior transfere seu aprendizado para o próximo modelo de rede, que aprende a solução no tempo corrente pela minimização de uma função de perda baseada na aproximação pelo esquema θ da equação de Burgers. Para testar esta abordagem, apresentamos sua aplicação a dois problemas padrões com soluções analíticas conhecidas. Em comparação com os modelos clássicos de PINNs, a abordagem proposta tem a vantagem de exigir arquiteturas de redes neurais menores com precisão semelhante e potencialmente diminuir os custos computacionais.

Palavras-chave: Equação de Burgers; Redes neurais informadas pela física; Método de Euler explícito; Método de Euler implícito; Método de Crank-Nicolson

1 INTRODUCTION

Burgers' equation is found in modeling fluid dynamics, gas dynamics, shock theory, cosmology, among others (Bonkile et al., 2018). It is widely used for testing the mathematical and numerical analysis of convective-diffusive differential equations (Konzen et al., 2017). We consider the viscous Burgers equation with initial and Dirichlet boundary conditions, given by

$$u_t + uu_x = \nu u_{xx},\ (t,x) \in (0,t_f] \times (0,1) \tag{1a}$$

$$u(0,x) = u_0(x),\ x \in [0,1], \tag{1b}$$

$$u(t,0) = u(t,1) = 0,\ t \in [0,t_f], \tag{1c}$$

For example, in a fluid dynamics application, it models the velocity $u = u(t,x)$ (m/s) in time t (s) and point x (m), of a fluid with kinematic viscosity ν (m^2/s).

In this work, we investigate the application of physics-informed neural networks (PINNs) with a transfer learning approach using the θ-scheme to solve the Burgers' equation. PINNs (Raissi et al., 2019) are deep learning techniques (Goodfellow et al., 2016) for solving partial differential equations (PDEs). Recently, they have been applied to solve many important problems, such as incompressible Navier-Stokes equations (Jin et al., 2021; Raissi et al., 2018), Euler equations for high-speed aerodynamic flows (Mao et al., 2020), heat transfer problems (Cai et al., 2021), and the advection equation (Vadyala et al., 2022).

For a time-dependent PDE, the classical PINN consists of a multilayer perceptron (MLP) neural network (Haykin, 2009) that learns the solution $u(t,x)$ from the governing equations. Usually, the MLP has input (t,x) and output $u(t,x)$, or input x and output $\boldsymbol{u}(x) = \left\{u\left(t^{(k)},x\right)\right\}_{k=0}^{n_t}$ for given n_t steps in time. Here, we investigate the application of an adaptation approach to PINN, in which the solution is given by a sequence of MLPs, one at each time step $t^{(k)}$. Given the initial condition, a first MLP $\mathcal{N}^{(0)}$ learns the solution at $t^{(0)} = 0$ by training to estimate the initial condition $u_0 = u_0(x)$. Then, given the time step size $h_t > 0$, the model $\mathcal{N}^{(0)}$ transfers its learning to a second MLP $\mathcal{N}^{(1)}$, which learns the solution in $t^{(1)} = h_t$ by training based on the θ-scheme of the Burgers' equation. This is an iterative process, where the k-th model $\mathcal{N}^{(k)}$ transfers its learning to start the next model $\mathcal{N}^{(k+1)}$, which learns the solution in time $t^{(k+1)}$ through the θ scheme.

Next, we present details about the PINNs approach with transfer learning by the θ scheme. Then, results for test cases are discussed. Finally, we present some final considerations.

2 PINNs

PINNs are deep learning techniques for solving partial differential equations (PDEs). The solution is learned by an artificial neural network (ANN) following a supervised learning approach that incorporates the PDE (governing equation, initial and boundary conditions) into the loss function. Here, we describe an alternative PINN approach, where a sequence of multilayer perceptrons (MLPs), one for each discrete time, is trained with a θ transfer learning scheme.

2.1 Multilayer perceptron

In this work, we consider an MLP type neural network

$$\tilde{u} = \mathcal{N}\left(x; \left\{\left(W^{(l)}, \boldsymbol{b}^{(l)}, \boldsymbol{f}^{(l)}\right)\right\}_{l=1}^{n_l}\right) \tag{2}$$

where the triple $\left(W^{(l)}, \boldsymbol{b}^{(l)}, \boldsymbol{f}^{(l)}\right)$ denotes the weights $W^{(l)}$, the biases $\boldsymbol{b}^{(l)}$ and the activation function $\boldsymbol{f}^{(l)}$ in the l-th layer of the network, $l = 1, 2, \ldots, n_l$, $n_l = n_h + 1$, with n_h a given number of hidden layers. As a deep learning technique, the output of the network is computed through successive compositions

$$\boldsymbol{a}^{(l)} = \boldsymbol{f}^{(l)}\left(W^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}\right), \tag{3}$$

with input $a^{(0)} = x$, output $a^{(n_l)} = \tilde{u}$ and $l = 1, 2, \ldots, n_l$. Assuming the necessary smoothness of $\boldsymbol{f}^{(l)}$, the derivatives $\tilde{u}_x$ and $\tilde{u}_{xx}$ can be computed by automatic differentiation as an application of the chain rule (Ketkar & Moolayil, 2021).

2.2 θ-scheme

The θ-scheme for Eq. (1) consists in the iteration

$$\tilde{u}^{(0)}(x) = u_0(x), \tag{4a}$$

$$\tilde{u}^{(k)} = \tilde{u}^{(k-1)} + (1-\theta)h_t\left(\nu\tilde{u}_{xx}^{(k-1)} - \tilde{u}^{(k-1)}\tilde{u}_x^{(k-1)}\right) + \theta h_t\left(\nu\tilde{u}_{xx}^{(k)} - \tilde{u}^{(k)}\tilde{u}_x^{(k)}\right), \tag{4b}$$

$$\tilde{u}^{(k)}(0) = \tilde{u}^{(k)}(1) = 0, \tag{4c}$$

where $\tilde{u}^{(k)}(x) \approx u\left(t^{(k)}, x\right)$ at each discrete time $t^{(k)} = kh_t$, $k = 0, 1, 2, \ldots, n_t$, with time step $h_t = 1/n_t$, for a given number of time steps n_t. Choosing $\theta = 0$ we have the explicit Euler scheme, with $\theta = 0.5$ we have the Crank-Nicolson scheme and for $\theta = 1$ we have the implicit Euler scheme.

The proposed PINN with θ transfer learning scheme consists of training a sequence of MLPs $\tilde{u}^{(k)}(x) = \mathcal{N}^{(k)}(x)$, $k = 0, 1, 2, \ldots, n_t$, with the input x and the output being the estimate of $\tilde{u}^{(k)}(x)$.

For the initial condition Eq. (4a), the neural network $\mathcal{N}^{(0)}$ is trained to minimize the loss function

$$\mathcal{L}_0 := \frac{1}{n_s}\sum_{s=1}^{n_s}\left|\tilde{u}^{(0)}(x_s) - u_0(x_s)\right|^2, \tag{5}$$

whith n_s samples $0 \leq x_s \leq 1$. Sequentially, $k = 1, 2, \ldots, n_t$, the knowledge of $\mathcal{N}^{(k-1)}$ is transferred to $\mathcal{N}^{(k)}$, which is trained to minimize the loss function

$$\mathcal{L} := \frac{1}{n_s - 2}\sum_{s=1}^{n_s-2}\left|\mathcal{R}\left(x; \tilde{u}^{(k)}, \tilde{u}^{(k-1)}\right)\right|^2 + \frac{1}{2}\left(\left|\tilde{u}^{(k)}(0)\right|^2 + \left|\tilde{u}^{(k)}(1)\right|^2\right), \tag{6}$$

where $\mathcal{R}$ denotes the residual

$$\mathcal{R}\left(x; \tilde{u}^{(k)}, \tilde{u}^{(k-1)}\right) := \tilde{u}^{(k)} - (1-\theta)h_t\left(\nu\tilde{u}_{xx}^{(k-1)} - \tilde{u}^{(k-1)}\tilde{u}_x^{(k-1)}\right) - \theta h_t\left(\nu\tilde{u}_{xx}^{(k)} - \tilde{u}^{(k)}\tilde{u}_x^{(k)}\right). \tag{7}$$

The derivatives are computed directly from the neural network by automatic differentiation.

The basic algorithm of PINN with transfer learning by the θ-scheme is summarized as follows:

0. Set the number n_t and the size h_t of the time step.

1. Set the architecture $\mathcal{N}^{(0)}$.

2. Train $\mathcal{N}^{(0)}$ to minimize the initial loss function Eq. (5).

3. For $k = 1, 2, \ldots, n_t$:

    a. Set θ.

    b. (Transfer learning.) $\mathcal{N}^{(k)} \leftarrow \mathcal{N}^{(k-1)}$.

    c. Train $\mathcal{N}^{(k)}$ to minimize the loss function Eq. (6).

In the end, the approach provides the sequence of MLPs $\{\mathcal{N}^{(k)}\}_{k=0}^{n_t}$, each providing the estimated solution $\tilde{u}^{(k)}(x) \approx u\left(t^{(k)}, x\right)$ of the Burgers' equation (1). Normally, there is no need to store the entire sequence, and the algorithm requires the storage only of $\mathcal{N}^{(k)}$ and $\mathcal{N}^{(k-1)}$ in each iteration.

2.3 Implementation details

We carried out implementations in Python language of neural network models with the help of the PyTorch library (Stevens et al., 2020). The model considered is an MLP with an architecture $1 - n_n \times n_h - 1$, one input, n_h hidden layers, each with n_n units, and one output. A study on the choices of n_h and n_n was presented in the work of (Biesek & Konzen, 2023). In the hidden layers, the hyperbolic tangent is used as the activation function and in the output layer the identity function. The models are trained using the Adam method (Kingma & Ba, 2014) as an optimizer. As a training stopping criteria, we assume $\mathcal{L}_0, \mathcal{L} < 10^{-6}$, and a maximum number of 5000 epochs (the calculation was performed using 32-bit float arithmetic).

3 RESULTS

Here, we present two benchmark problems to test the proposed approach. In the first, the solution of the Burgers' equation is a shock wave. The second is a rarefaction problem.

3.1 Problem 1: Shock wave problem

The firs problem has the initial condition

$$u_0(x) = -\sin(\pi x). \tag{8}$$

In the article by (Basdevant et al., 1986), the following analytical solution is given

$$u(x) = \frac{-\int_{-\infty}^{\infty} \sin(\pi(x-\eta)) f(x-\eta) e^{-\frac{\eta^2}{4\nu t}}\, d\eta}{\int_{-\infty}^{\infty} f(x-\eta) e^{-\frac{\eta^2}{4\nu t}}\, d\eta}, \tag{9}$$

where $f(y) = e^{-\cos(\pi y)/(2\pi\nu)}$. We assume $\nu = 0.01/\pi$.

We consider an MLP with structure $1-50\times 4-1$, one input, 4 hidden layers with 50 units in each layer, and one output. The model $\mathcal{N}^{(k)}$ receives the initial parameters of $\mathcal{N}^{(k-1)}$ and its training depends on the time step h_t and the number of spatial samples n_s. In order to analyze the influence of such parameters, we performed numerical tests varying them, for each scheme $\theta = 0$ (explicit Euler), $\theta = 0.5$ (Crank-Nicolson) and $\theta = 1$ (implicit Euler). For $\theta = 0$, Table 1 shows the results $n_e \setminus \varepsilon_{\text{rel}}$, with n_e being the total number of epochs and ε_{rel} final value of the relative L^2 error

$$\varepsilon_{\text{rel}}\left(t^{(k)}\right) := \frac{\left\|\tilde{u}^{(k)}(x) - u\left(t^{(k)}, x\right)\right\|_2}{\left\|u\left(t^{(k)}, x\right)\right\|_2}, \tag{10}$$

for $k = n_t$, i.e. $t = t_f = 1$. Due to the stochastic nature of the training method, each test has been performed three times, and the average of the results are tabulated. In Table 2 results are presented with the scheme $\theta = 0.5$. Results for the $\theta = 1$ scheme are presented in Table 3.

Table 1 – Problem 1: Test of parameters with the explicit Euler scheme

$h_t \backslash n_s$	20	200	2000
1E−1	$-x-$	$-x-$	$-x-$
1E−2	4E+4 \ 7.5E−4	6E+4 \ 2.1E−3	5E+4 \ 3.0E−4
1E−3	16E + 4 \ 1.7E−2	14E+4 \ 1.8E−2	1.3E+5 \ 9.0E−3

Source: the authors (2024)

Table 2 – Problem 1: Test of parameters with the Crank-Nicholson scheme

$h_t \backslash n_s$	20	200	2000
1E−1	1E+4 \ 2.8E−4	6E+4 \ 3.1E−5	4E+4 \ 4.2E−5
1E−2	4E+4 \ 1.4E−2	4E+4 \ 1.5E−3	3E+4 \ 1.5E−3
1E−3	15E+4 \ 4.0E−2	13E+4 \ 6.9E−3	14E+4 \ 4.2E−3

Source: the authors (2024)

Table 3 – Problem 1: Test of parameters with the implicit Euler scheme

$h_t \backslash n_s$	20	200	2000
1E−1	$-x-$	4E+4 \ 4E−3	4E+4 \ 4E−3
1E−2	3E+4 \ 1E−3	3E+4 \ 2E−4	5E+4 \ 1E−3
1E−3	2E+5 \ 1E−2	14E+4 \ 3E−3	14E+4 \ 5E−3

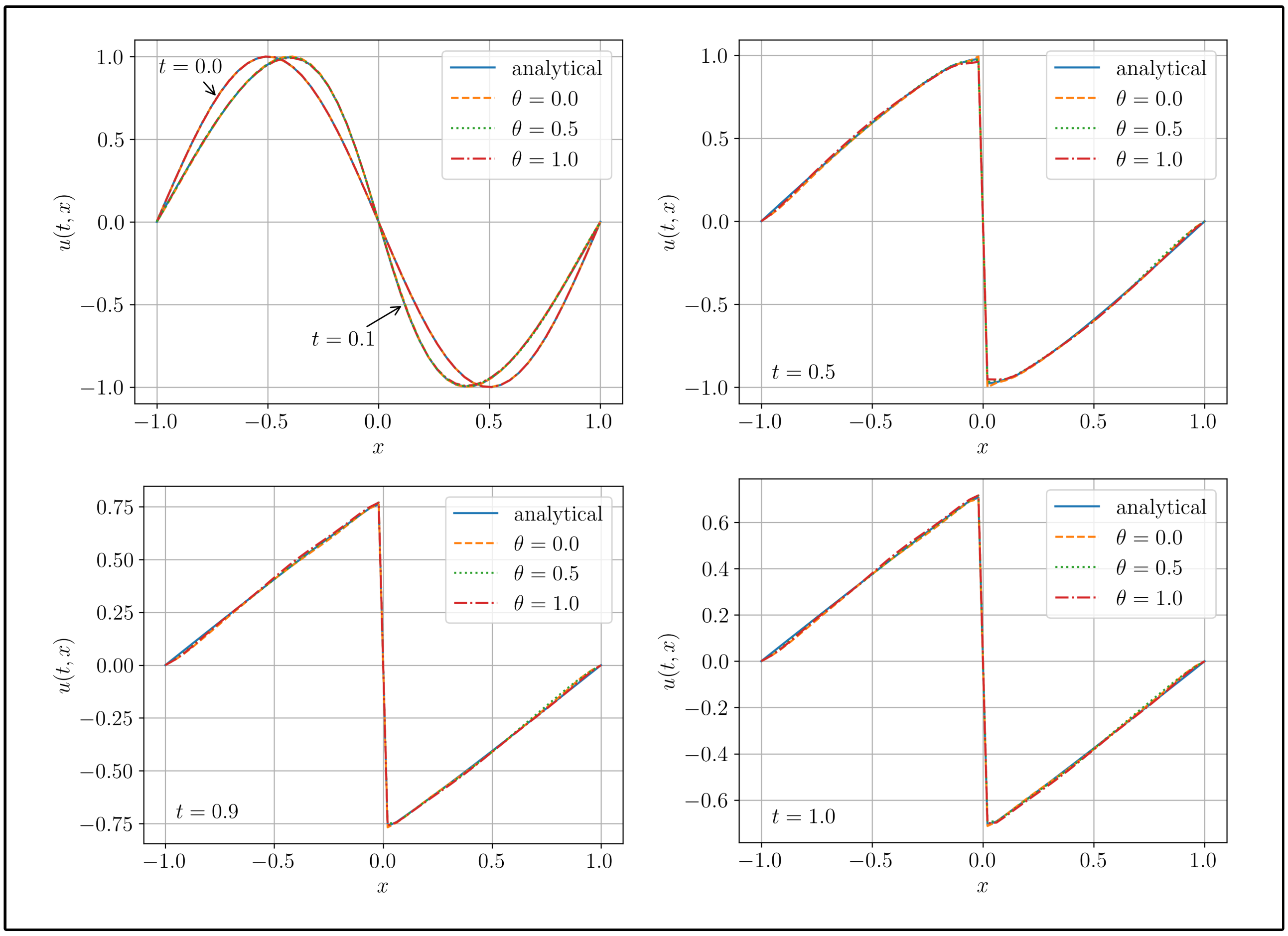
Source: the authors (2024)

The results presented in Tables 1-3 indicate that transfer learning was effective with all time steps in the Crank-Nicolson scheme. Euler's schemes required an adequate time step, and in the case of $h_t = 10^{-1}$, the learning of the networks failed to converge. From the analysis of these results, we observed that $h_t = 10^{-2}$ and $n_s = 200$ present a good relationship between accuracy and learning cost.

Figure 1 shows the plots of PINNs solutions (with transfer learning schemes $\theta = 0$, 0.5 and 1), with $n_s = 200$ and $h_t = 10^{-2}$, *versus* the analytical solution (solid line) of the problem for times $t = 0.0$, 0.1, 0.5, 0.9 and 1.0.

In comparison with the estimations reported by (Raissi et al., 2019) for this same problem, we note that our proposed transfer learning approach produce similar results with a similar MLP architecture. In the paper of (Raissi et al., 2019), an MLP $2 - 20 \times 9 - 1$ was found to produce good estimations. In the next subsection we discuss on a problem with discontinuous initial condition, where our proposed transfer learning approach has feasible advantages in comparison with classical PINN schemes.

Figure 1 – Problem 1: Comparisons of PINNs solutions with transfer learning *versus* analytical solutions



Note: Solid line: analytical solution. Dashed line: PINN with explicit Euler scheme. Dotted line: PINN with Crank-Nicolson scheme. Dash-dotted line: PINN with implicit Euler scheme
Source: the authors (2024)

3.2 Problem 2: Rarefaction problem

The second test case is a rarefaction problem with initial condition

$$u_0(x) = 2\,\mathrm{sign}\,(x) \quad , \quad x \in [a, b] = [-1, 1]. \tag{11}$$

Assuming $\nu = 1$, the analytical solution is given by (Benton & Platzman, 1972)

$$u(t, x) = 2\frac{G(t, x) - G(t, -x)}{G(t, x) + G(t, -x)} \quad , \tag{12}$$

where $G(t, x) = \frac{1}{2}e^{t-x}\mathrm{erfc}\frac{2t - x}{2\sqrt{t}}$. We consider the boundary conditions $u(t, a) = u_a$ and $u(t, b) = u_b$.

This problem involves a discontinuity at $x = 0$ in the initial condition, which often leads to MLP estimations with spurious oscillations near that point. To address this issue, we opt not to train the $\mathcal{N}^{(0)}$ model directly with the initial condition during the training of $\mathcal{N}^{(1)}$. Instead, we assume $\tilde{u}^{(0)} = u_0$. While this approach could be applied to previous problems as well, it is noted that it comes with increased computational demands. Hence, the preferable strategy is to initially train $\mathcal{N}^{(0)}$ and then transfer its acquired knowledge to $\mathcal{N}^{(1)}$, particularly when the initial condition u_0 exhibits sufficient smoothness. In summary, when the initial condition is not smooth, the approach always perform the first time step with the implicit Euler scheme.

Here, an MLP with architecture $1 - 30 \times 3 - 1$, one input, 3 hidden layers with 30 units per layer and one output, and fixing the time step as $h_t = 10^{-3}$, was enough to achieve the expected tolerance for $\mathcal{L} < 10^{-5}$ with $\theta = 0.5$ and 1. We observe that it wasn't possible to achieve the expected tolerance with the explicit Euler scheme, nor with a time step bigger then $h_t = 10^{-3}$.

By performing the same tests as in the previous problem, we obtained the results $n_e \setminus \varepsilon_{\text{rel}}$ shown in Table 4, where n_e is the total number of epochs and ε_{rel} the final value of the relative error L^2 given by (10). Again, the average results of three rounds are tabulated. The results show that the Crank-Nicolson and implicit Euler schemes have similar accuracies.

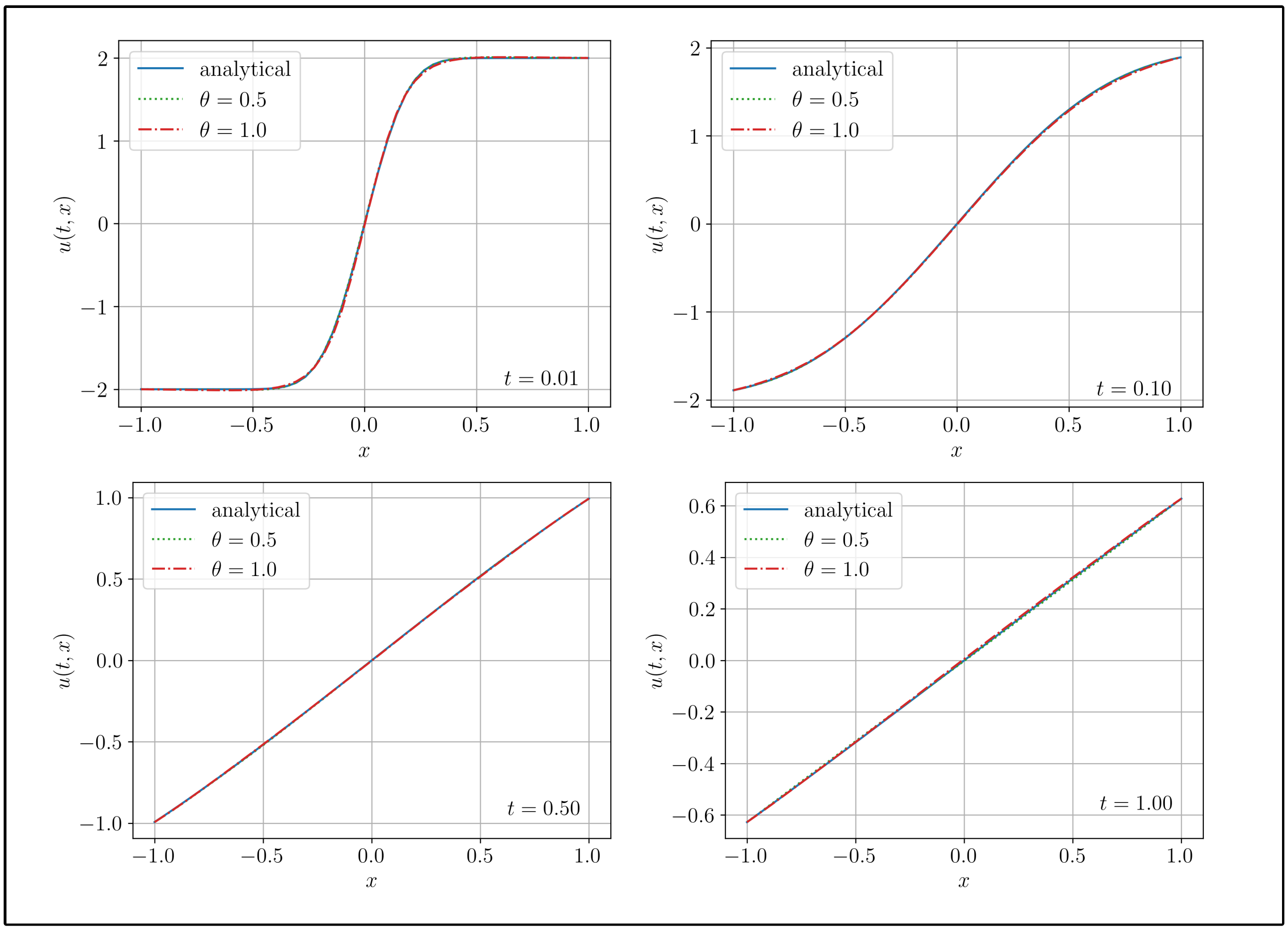
Table 4 – Problem 2: Tests of Crank-Nicolson (CN) and implicit Euler (IE) schemes with $h_t = 10^{-3}$.

n_s	20	200	400
CN	1E+5 \ 4E−4	1E+5 \ 5E−5	1E+5 \ 5E−5
IE	1E+5 \ 2E−4	1E+5 \ 6E−5	1E+5 \ 9E−5

Source: the authors (2024)

Figure 2 shows the graph of PINNs solutions (with transfer learning scheme $\theta = 0.5$ and 1), with $n_s = 200$ and $h_t = 10^{-3}$, *versus* the analytical solution (solid line) of the problem for times $t = 0.01$, 0.10, 0.50, and 1.0.

Figure 2 – Problem 2: Comparisons of PINNs solutions with transfer learning *versus* analytical solutions



Note: Solid line: analytical solution. Dotted line: PINN with Crank-Nicolson scheme. Dash-dotted line: PINN with implicit Euler scheme
Source: the authors (2024)

3.2.1 Comparison on Classical PINNs

An classical PINN approach will seek the solution for the the Problem 2 by assuming a MLP with inputs t, x and output $\tilde{u} \approx u(t,x)$. A common choice for the loss function will involve the estimation of the initial condition. Because its discontinuity, this is a not suitable task for a MLP, which is expected to produce estimations of u with spurious oscillations for t near 0. An alternative also explored in the paper of (Raissi et al., 2019), is to assume an MLP with input x and discrete time estimations as outputs $\tilde{\boldsymbol{u}}(x) \approx \left(u^{(k+1)}(x)\right)_{k=0}^{n_t-1}$, avoiding the estimations of the initial condition.

This last PINN approach estimates the solution just on discrete time steps, and it can be trained by minimizing the residual of a discrete time method as the θ-scheme, some Runge-Kutta method or some other. By assuming the θ-scheme, one could impose the following loss function

$$\begin{aligned}\mathcal{L}_{\text{PINN}} := \frac{1}{(n_s-2)n_t}\sum_{s=1}^{n_s-2}\left(\left|\mathcal{R}_{\text{PINN}}^{(1)}(x_s)\right|^2+\sum_{k=1}^{n_t}\left|\mathcal{R}_{\text{PINN}}^{(k+1)}(x_s)\right|^2\right)\\ +\frac{1}{2n_t}\sum_{k=0}^{n_t-1}\left(\left|u\left(t^{(k+1)},0\right)-\tilde{u}^{(k+1)}(0)\right|^2+\left|u\left(t^{(k+1)},1\right)-\tilde{u}^{(k+1)}(1)\right|^2\right),\end{aligned} \tag{13}$$

where the implicit Euler would need to be imposed for $k=0$, i.e.

$$\mathcal{R}_{\text{PINN}}^{(1)}(x_s) := \tilde{u}^{(1)}(x_s)-u_0(x_s)-h_t\left(\nu\tilde{u}_{xx}^{(1)}-\tilde{u}^{(1)}\tilde{u}_x^{(1)}\right). \tag{14}$$

For $k>0$, the θ-scheme will be applied by

$$\begin{aligned}\mathcal{R}_{\text{PINN}}^{(k+1)}(x_s) := \tilde{u}^{(k+1)}(x_s)-\tilde{u}^{(k)}(x_s)+(1-\theta)h_t\left(\nu\tilde{u}_{xx}^{(k)}-\tilde{u}^{(k)}\tilde{u}_x^{(k)}\right)\\ -\theta h_t\left(\nu\tilde{u}_{xx}^{(k+1)}-\tilde{u}^{(k+1)}\tilde{u}_x^{(k+1)}\right).\end{aligned} \tag{15}$$

In comparison to our proposed transfer learning PINN, this approach has a notable disadvantage. It requires bigger MLPs, since the number of outputs are n_t, which usually will be of order 10^2-10^4 (giving $h_t=10^{-2}-10^{-4}$). Based on the results previous shown, we performed numerical tests with MLP of architectures $1-30\times 3-n_t$, $1-30\times 6-n_t$, and $1-60\times 3-n_t$ for $n_t=100, 1000$ and all failed to produce suitable estimations with the classical PINN approach discussed above. It indicates that a much bigger architecture is required, demanding a higher computational cost (processing and memory), much related to the high cost of the automatic differentiation computations.

4 CONCLUSIONS

In this work, we proposed the application of PINNs with transfer learning using the θ-scheme to solve the viscous Burgers' equation with homogeneous initial and Dirichlet boundary conditions. The method consists of using MLPs to estimate the solution in discrete time steps. As an iterative process, a first neural network model is trained to learn the initial condition. Then, the knowledge is transferred to the next model, which learns the solution in the next time step, minimizing the residual of the θ-scheme of the Burgers' equation. The result is a sequence of neural network models that estimate solutions to the problem in discrete times.

Compared to usual PINN models, the proposed approach has the potential to require smaller neural network architectures with similar accurate results and, consequently, reduce computational costs. The comparison between the schemes $\theta = 0.0$, 0.5 and 1, showed that all have a similar accuracy when convergence is achieved. The explicit Euler scheme converged only with small enough time steps. For the problem with discontinuous initial condition the approach has only converged with the implicit and the Crank-Nicolson schemes.

ACKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

Basdevant, C., Deville, M., Haldenwang, P., Lacroix, J., Ouazzani, J., Peyret, R., Orlandi, P., & Patera, A. (1986). Spectral and finite fifference solutions of the burgers' equation. *Computers $\&$ Fluids*, *14*(1):23–41.

Benton, E. R. & Platzman, G. W. (1972). A table of solutions of the one-dimensional Burgers equation. *Quarterly of Applied Mathematics*, *30*(2):195–212.

Biesek, V. & Konzen, P. H. A. (2023). Burgers' PINNs with Implicit Euler Transfer Learning. *Anais do Encontro Nacional de Modelagem Computacional e Encontro de Ciência e Tecnologia dos Materiais*, Nova Friburgo, RJ, Brasil, 16.

Bonkile, M., Awasthi, A., Nair, L., Mukundan, V., & Vs, A. (2018). A systematic literature review of burgers' equation with recent advances. *Pramana*, *90*(69):1–21.

Cai, S., Wang, Z., Wang, S., Perdikaris, P., & Karniadakis, G. E. (2021). Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, *143*(6):060801.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. Recovered from: http://www.deeplearningbook.org.

Haykin, S. (2009). *Neural networks and learning machines*. (3th ed). Upper Saddle River: Pearson Education.

Jin, X., Cai, S., Li, H., & Karniadakis, G. E. (2021). Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, *426*:109951.

Ketkar, N. & Moolayil, J. (2021). *Deep learning with Python*. Apress.

Kingma, D. & Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*, *9*:1–15.

Konzen, P. H. A., Azevedo, F. S., Sauter, E., & Zingano, P. R. A. (2017). Numerical simulations with the galerkin least squares finite element method for the burgers' equation on the real line. *TEMA*, *18*(2):287–304.

Mao, Z., Jagtap, A. D., & Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, *360*:112789.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (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.

Raissi, M., Wang, Z., Triantafyllou, M. S., & Karniadakis, G. E. (2018). Deep learning of vortex induced vibrations. *Journal of Fluid Mechanics*, *861*:119–137.

Stevens, E., Antiga, L., & Viehmann, T. (2020). *Deep learning with PyTorch*. Manning Publications.

Vadyala, S. R., Betgeri, S. N., & Ph.D, N. P. B. (2022). Physics-informed neural network method for solving one-dimensional advection equation using pytorch. *Array*, *13*:100110.

Author contributions

1 – Vitória Biesek (Corresponding Author)
Master student in Applied Mathematics
https://orcid.org/0009-0002-9716-3764 • vitoriabiesek@gmail.com
Contribution: Writing & Editing

2 – Pedro Henrique de Almeida Konzen
Doctor in Applied Mathematics
https://orcid.org/0000-0002-0411-1563 • pedro.konzen@ufrgs.br
Contribution: Writing - Review & Editing

How to cite this article

Biesek, V., & Konzen, P. H. de A. (2024). Burgers' PINNs with transfer learning by θ-scheme. *Ciência e Natura*, Santa Maria, v. 47, spe. 1, e89888. DOI: https://doi.org/10.5902/2179460X89888.