Physical-informed Neural Networks for the simulation of nonlinear partial differential equations in fluid dynamics

Alejandro Salazar Arango[1]

Advisor(s):
Cristhian David Montoya Zambrano[2]

Research practice 3
Research proposal
Mathematical Engineering
School of Applied Sciences and Engineering
Universidad EAFIT

August 2023

[1]Mathematical Engineering Student. Universidad EAFIT asalazara1@eafit.edu.co
[2]School of Applied Sciences and Engineering. Universidad EAFIT cdmontoyaz@eafit.edu.co

Abstract

The purpose of this paper is to analyze the performance of Physical Informed Neural Networks for simulating fluid dynamics equations. This is made by the implementation of multiple architectures for solving multiples cases of both the Navier Stokes Equations and the Burgers Equations For later comparing the behavior of the L^2 Error of the outputs of each model.

Keywords: Computational fluid dynamics, Machine Learning, PINN, Nonlinear Partial Differential Equations

1 Introduction

Partial differential equations (PDEs) are one of the most powerful tools for the study of physical and biological phenomena. They allow us to represent in a simplified way, the dynamics between the spatial and temporal variables of any phenomenon and to take these dynamics to a mathematical model, whose solution describes in a great way the behavior of the studied system[1]. This has led partial differential equations to become a tool of great use in areas such as fluid dynamics, electromagnetism, optics, magnetism, among others[2]; these areas being dominated by PDEs such as Maxwell's Equations and Navier-Stokes Equations.

However, PDEs, given their complex structure, are usually models whose analytical solutions are difficult or impossible to find, being even the solutions already found for some PDEs so complex that it is preferred to use other methods to solve the problem[3]. This is why, in recent years, the development of multiple numerical methods to solve this type of equations has been one of the major topics of study of the scientific community. In addition, numerical methods for solving PDEs are a really important topic of study given that although there are already multiple methods developed, if these are not applied correctly can lead to solutions far from the real solution of the problem and therefore lead to erroneous conclusions about the behavior of the system under study.

One of the most commonly used numerical method for solving Partial Differential Equations is the Finite Element Method (FEM), a numerical method consisting on the discretization of the problem's domain, to later solve a variational formulation of the problem with determined test functions, to obtain an algebraic system of equations, which, when solved, gives the approximated solution of the problem. However, due to the substantial number of degrees of freedom it needs in order to correctly solve the PDE, it remains as an extremely expensive method both in terms of CPU and memory demand, therefore not being too useful for real-time contexts[4] [5]. Consequently, models such as physically informed neural networks (PINNs) have been developed and have become important methods in the study of PDEs.

PINNs are quite useful in this type of problems since, by incorporating the physical information of the system under study in the training of the neural networks, allowing the imposition of boundary conditions and governing equations directly on the network architecture, it facilitates the adaptation of the network to different conditions and geometries on which to solve the problem, in addition to potentially even improving the accuracy of the numerical solutions.

Therefore this project focuses on the implementation of multiple Physical Informed Neural Networks (PINNs) Architectures for solving Fluid Dynamic problems, seeking to observe and compare the performance of these architectures, aiming for a deeper understanding of the advantages and limitations of PINNs in the context of Fluid dynamics simulations.

2 State of the art

Given its important on a great amount of biological, medical, and physical fields, PDEs are one of the most studied topics in applied mathematics, and therefore, a large body of literature on the subject can be found. The following is a review of some of the literature consulted on PINNs and RBMs and their applications in EDPs, where a collection of both theoretical and applied papers are presented.

On the other hand, although PINNs are new techniques in the solution of PDEs, they have also been shown to be a topic of great interest to the scientific community and therefore it is not difficult to find literature on the subject, not only for forward problems but for inverse problems as well. On this matter, Berg & Nyström [6] , Meng & Karniadakis [7] and Raissi, Perdikaris & Karniadakis [8] present three works of interest in which they use PINNs as a solution method for inverse problems with multiple examples and making a focus on the explanation of how PINNs work on every presented examples, as well as a mathematical background on the method itself.

Some works about the usage of PINN for fluid dynamics problems can be found as well. A first interesting work on this topic can be found in the paper from Moschou *et al.* [9] who present an application of PINNs on the modeling of astrophysical shocks, focusing on the solar termination shock, a phenomenon described using the fluid dynamics conservation laws. In agreement Ciao *et al.* [10] developed an application of PINN for the turbulent mixing induced by the Rayleigh-Taylor instability, using the Reynolds-Averaged Navier–Stokes Equations for modeling the problem. And finally, Lino *et al.* [11] exhibit a review of the current and emerging deep-learning methods used on fluid dynamics simulation.

3 Proposed methodology

As a means to achieve the objectives set for this project, a 3-stage methodology was proposed in which various mathematical methods were used. In the first stage a version of the Broyden–Fletcher–Goldfarb–Shanno algorithm, the L-BFGS-B optimization method, was implemented, which is the optimization method used for the training of the PINNs implemented. The second stage consisted of the implementation, training, and validation of multiple PINN architecture, which were used for solving both Burgers and Navier Stokes equations with a variety of cases for comparing the capacity of these Neural Networks for solving these problems. Finally, the third stage was the analysis of results and the comparison of the performances of the models implemented on the previous stage.

Below both L-BFGS-B and PINN are explained with more detail for the reader to understand the nature of both algorithms and how they work.

3.1 The Broyden–Fletcher–Goldfarb–Shanno algorithm

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm is an iterative method for numerical optimization, specifically used for finding the minimum of a function. It falls under the category of quasi-Newton methods and is particularly well-suited for problems where the objective function is smooth and has continuous derivatives. The main goal of the BFGS algorithm is to iteratively update an approximation of the inverse Hessian matrix, which represents the local curvature of the objective function, to efficiently converge towards the minimum.

At each iteration, the BFGS algorithm computes the gradient of the objective function with respect to the parameters, in order to subsequently update the current estimate of the inverse Hessian matrix based on the difference in gradients and parameter updates between iterations as shown in Equation 1. By utilizing this information, the algorithm generates a sequence of parameter values that progressively move closer to the optimal solution. Unlike Newton's method, BFGS avoids explicitly computing the Hessian matrix, which can be computationally expensive for large problems, by iteratively updating an approximation of it. [12].

$$\begin{aligned} s_k &= x_{k+1} + x_k \\ y_k &= \nabla f(x_{k+1}) - \nabla f(x_k) \\ H_{k+1}^{-1} &= (I - \frac{sy^T}{y^T s}) H_k^{-1} \frac{ys^T}{y^T s} + \frac{ss^T}{y^T s} \end{aligned} \tag{1}$$

BFGS' use of an approximation of the inverse Hessian matrix allows for a balance between efficiency and accuracy, making it a popular choice for optimizing smooth and high-dimensional functions. However, BFGS does not guarantee convergence to a global minimum and may converge to a local minimum based on the starting point and the characteristics of the objective function.

3.2 Physical Informed neural Networks

Physics-informed neural networks (PINNs) have emerged as a promising approach to solve PDEs with remarkable efficiency and accuracy. These combine the power of neural networks with the governing equations of physical systems, offering a seamless integration of data-driven learning and domain-specific physics. They work by incorporating the residual error of the governing equations into the neural network training process, gaining a unique ability to capture the underlying physics of the model while learning from sparse and noisy data[8].

Unlike traditional methods, PINNs avoid the need for explicit mesh discretization and provide continuous solutions throughout the domain, dynamically adapting to complex geometries. Their versatility, combined with the ability to leverage existing deep learning frameworks, positions PINNs as a transformative tool for advancing EDP-based modeling and simulation in interdisciplinary domains.

For this project, the training of the PINNs was performed using the initial and boundary data of each problem as a set of training points and selecting a random set of points in the problem domain for the evaluation of the residual of the underlying equations, thus not requiring a previous solution of the equation for the training of the neural network.

4 Results

For this project three set of equations were considered with multiple scenarios and Architectures each. Each experiment was implemented on a Virtual Machine provided by Google Cloud platform with an e2-highmem-16 architecture which has 16 vCPU and 128 GB of RAM [13]. The main results obtained for each set of equations is presented below.

4.1 1-dimentional Burgers' Equation

For the 1-dimentional Burgers0 Equation, three cases were implemeneted each with the listed three architectures:

- Architecture 1: 5 hidden layers with 20 neurons each
- Architecture 2: 5 hidden layers with 50 neurons each
- Architecture 3: 8 hidden layers with 20 neurons each

Case 1:

For the first case the following exact solution $u(x,t)$ was considered

$$u(x,t) = e^{-t}\sin(\pi x)$$

which, for $\nu = \frac{1}{\pi^2}$, gives the PDE below

$$u_t + uu_x - \frac{1}{\pi^2} u_{xx} = \pi e^{-2t} \sin(\pi x) \cos(\pi x) \tag{2}$$

When solving Equation 2 with the architectures explained above, the following results were obtained

Time	**MSE**	L^2 **error**
0.0	4.1430×10^{-8}	2.0354×10^{-4}
0.1	1.7357×10^{-8}	1.3175×10^{-4}
0.2	1.0056×10^{-8}	1.0028×10^{-4}
0.3	7.8224×10^{-9}	8.8444×10^{-5}
0.4	5.2395×10^{-9}	7.2384×10^{-5}
0.5	2.1801×10^{-9}	4.6691×10^{-5}
0.6	1.2080×10^{-9}	3.4756×10^{-5}
0.7	2.7061×10^{-9}	5.2020×10^{-5}
0.8	2.9751×10^{-9}	5.4544×10^{-5}
0.9	8.7641×10^{-10}	2.9604×10^{-5}
1.0	1.0935×10^{-8}	1.0457×10^{-4}

Table 1: Results for the first architecture in the first case of Burgers Equations

Time	**MSE**	L^2 **error**
0.0	4.9834×10^{-9}	7.0593×10^{-5}
0.1	8.7334×10^{-9}	9.3453×10^{-5}
0.2	1.4264×10^{-8}	1.1943×10^{-4}
0.3	1.4884×10^{-8}	1.2200×10^{-4}
0.4	1.0369×10^{-8}	1.0183×10^{-4}
0.5	5.4275×10^{-9}	7.3671×10^{-5}
0.6	4.2357×10^{-9}	6.5083×10^{-5}
0.7	5.5986×10^{-9}	7.4824×10^{-5}
0.8	4.3080×10^{-9}	6.5635×10^{-5}
0.9	1.3539×10^{-9}	3.6795×10^{-5}
1.0	2.6247×10^{-8}	1.6201×10^{-4}

Table 2: Results for the second architecture in the first case of Burgers Equations

Time	**MSE**	L^2 **error**
0.0	2.7108×10^{-9}	5.2065×10^{-5}
0.1	1.2982×10^{-9}	3.6030×10^{-5}
0.2	4.2802×10^{-9}	6.5424×10^{-5}
0.3	7.1932×10^{-9}	8.4813×10^{-5}
0.4	6.6241×10^{-9}	8.1388×10^{-5}
0.5	4.1758×10^{-9}	6.4621×10^{-5}
0.6	2.0452×10^{-9}	4.5224×10^{-5}
0.7	8.8635×10^{-10}	2.9772×10^{-5}
0.8	4.2484×10^{-10}	2.0612×10^{-5}
0.9	3.2664×10^{-10}	1.8073×10^{-5}
1.0	1.0409×10^{-9}	3.2263×10^{-5}

Table 3: Results for the third architecture in the first case of Burgers Equations

Case 2:

For the second case the following exact solution $u(x,t)$ was considered

$$u(x,t) = e^t(x - x^2)$$

which, for $\nu = 1$, gives the PDE below

$$u_t + uu_x - u_{xx} = e^t(x - x^2) + e^{2t}(x + x^2)(1 - 2x) + 2e^t \tag{3}$$

When solving Equation 3 with the architectures explained above, the following results were obtained

Time	**MSE**	L^2 **error**
0.0	3.9986×10^{-9}	6.3234×10^{-5}
0.1	3.0841×10^{-9}	5.5534×10^{-5}
0.2	8.1619×10^{-10}	2.8569×10^{-5}
0.3	4.8544×10^{-9}	6.9673×10^{-5}
0.4	1.4056×10^{-8}	1.1856×10^{-4}
0.5	1.4323×10^{-8}	1.1968×10^{-4}
0.6	4.6711×10^{-9}	6.8346×10^{-5}
0.7	3.4578×10^{-9}	5.8803×10^{-5}
0.8	1.9748×10^{-8}	1.4053×10^{-4}
0.9	2.2694×10^{-8}	1.5064×10^{-4}
1.0	1.0653×10^{-8}	1.0321×10^{-4}

Table 4: Results for the first architecture in the second case of Burgers Equations

Time	**MSE**	L^2 **error**
0.0	5.5798×10^{-8}	2.3622×10^{-4}
0.1	5.7194×10^{-8}	2.3915×10^{-4}
0.2	1.3271×10^{-7}	3.6430×10^{-4}
0.3	1.1661×10^{-7}	3.4148×10^{-4}
0.4	4.7579×10^{-8}	2.1813×10^{-4}
0.5	6.0910×10^{-9}	7.8045×10^{-5}
0.6	2.9455×10^{-8}	1.7162×10^{-4}
0.7	8.0868×10^{-8}	2.8437×10^{-4}
0.8	7.7610×10^{-8}	2.7859×10^{-4}
0.9	1.4173×10^{-8}	1.1905×10^{-4}
1.0	3.4448×10^{-7}	5.8693×10^{-4}

Table 5: Results for the second architecture in the second case of Burgers Equations

Time	**MSE**	L^2 **error**
0.0	5.8925×10^{-8}	2.4274×10^{-4}
0.1	6.7472×10^{-8}	2.5975×10^{-4}
0.2	7.6371×10^{-8}	2.7635×10^{-4}
0.3	3.8459×10^{-8}	1.9611×10^{-4}
0.4	6.7915×10^{-8}	2.6061×10^{-4}
0.5	1.4779×10^{-7}	3.8444×10^{-4}
0.6	1.5428×10^{-7}	3.9279×10^{-4}
0.7	6.3770×10^{-8}	2.5253×10^{-4}
0.8	1.3139×10^{-8}	1.1462×10^{-4}
0.9	3.3779×10^{-8}	1.8379×10^{-4}
1.0	8.7593×10^{-8}	2.9596×10^{-4}

Table 6: Results for the third architecture in the second case of Burgers Equations

Case 3:

For the third case the following exact solution $u(x,t)$ was considered

$$u(x,t) = te^{x-x^2}$$

which, for $\nu = 1$, gives the PDE below

$$u_t + uu_x - u_{xx} = e^{x-x^2} + t^2 e^{2(x-x^2)}(1-2x) - te^{x-x^2}(4x^2 - 4x - 1) \tag{4}$$

When solving Equation 4 with the architectures explained above, the following results were obtained

Time	**MSE**	L^2 **error**
0.0	1.8636×10^{-9}	4.3169×10^{-5}
0.1	1.1989×10^{-9}	3.4625×10^{-5}
0.2	4.6511×10^{-9}	6.8199×10^{-5}
0.3	1.0917×10^{-8}	1.0449×10^{-4}
0.4	1.0227×10^{-8}	1.0113×10^{-4}
0.5	2.8382×10^{-9}	5.3275×10^{-5}
0.6	2.3910×10^{-9}	4.8898×10^{-5}
0.7	1.4141×10^{-8}	1.1892×10^{-4}
0.8	1.9828×10^{-8}	1.4081×10^{-4}
0.9	5.3809×10^{-9}	7.3355×10^{-5}
1.0	4.0183×10^{-8}	2.0046×10^{-4}

Table 7: Results for the first architecture in the third case of Burgers Equations

Time	**MSE**	L^2 **error**
0.0	5.2489×10^{-9}	7.2449×10^{-5}
0.1	4.4637×10^{-9}	6.6811×10^{-5}
0.2	1.3846×10^{-8}	1.1767×10^{-4}
0.3	1.2864×10^{-8}	1.1342×10^{-4}
0.4	4.8487×10^{-9}	6.9632×10^{-5}
0.5	2.0942×10^{-9}	4.5762×10^{-5}
0.6	9.1584×10^{-9}	9.5699×10^{-5}
0.7	1.5329×10^{-8}	1.2381×10^{-4}
0.8	7.8916×10^{-9}	8.8835×10^{-5}
0.9	2.3779×10^{-9}	4.8764×10^{-5}
1.0	7.9247×10^{-8}	2.8151×10^{-4}

Table 8: Results for the second architecture in the third case of Burgers Equations

Time	**MSE**	L^2 **error**
0.0	2.1867×10^{-8}	1.4788×10^{-4}
0.1	6.7413×10^{-9}	8.2105×10^{-5}
0.2	4.0810×10^{-9}	6.3883×10^{-5}
0.3	2.5088×10^{-8}	1.5839×10^{-4}
0.4	3.4817×10^{-8}	1.8659×10^{-4}
0.5	1.5709×10^{-8}	1.2534×10^{-4}
0.6	3.1670×10^{-9}	5.6276×10^{-5}
0.7	2.1016×10^{-8}	1.4497×10^{-4}
0.8	3.4413×10^{-8}	1.8551×10^{-4}
0.9	1.0148×10^{-8}	1.0074×10^{-4}
1.0	4.2785×10^{-8}	2.0684×10^{-4}

Table 9: Results for the third architecture in the third case of Burgers Equations

4.2 1-dimentional Navier-Stokes Equations

For the 1-dimentional Navier-Stokes Equations, two cases were implemeneted each with the listed four architectures:

- Architecture 1: 5 hidden layers with 20 neurons each
- Architecture 2: 5 hidden layers with 50 neurons each
- Architecture 3: 8 hidden layers with 20 neurons each
- Architecture 4: 8 hidden layers with 50 neurons each

Case 1:

For the first case the following exact solution $u(x,t)$ and $p(x,t)$ was considered

$$u(x,t) = e^{-t}\sin(\pi x)$$
$$p(x,t) = -\sin(\pi x)$$

which, for $\nu = \frac{1}{\pi^2}$, gives the PDE below

$$u_t + uu_x - \frac{1}{\pi^2}u_{xx} + p_x = \pi e^{-2t}\left(\sin(\pi x)\cos(\pi x) - \cos(\pi x)\right) \tag{5}$$

When solving Equation 5 with the architectures explained above, the following results were obtained

	$u(x,t)$		$p(x,t)$	
Time	**MSE**	L^2 **error**	**MSE**	L^2 **error**
0.0	1.9714×10^{-7}	4.4401×10^{-4}	9.8200×10^{-6}	3.1337×10^{-3}
0.1	6.8281×10^{-7}	8.2633×10^{-4}	1.2453×10^{-5}	3.5289×10^{-3}
0.2	1.7221×10^{-6}	1.3123×10^{-3}	1.6410×10^{-5}	4.0509×10^{-3}
0.3	3.2257×10^{-6}	1.7960×10^{-3}	2.1246×10^{-5}	4.6094×10^{-3}
0.4	5.4655×10^{-6}	2.3378×10^{-3}	2.7291×10^{-5}	5.2241×10^{-3}
0.5	8.6866×10^{-6}	2.9473×10^{-3}	3.4859×10^{-5}	5.9041×10^{-3}
0.6	1.3048×10^{-5}	3.6123×10^{-3}	4.3914×10^{-5}	6.6268×10^{-3}
0.7	1.8638×10^{-5}	4.3172×10^{-3}	5.3911×10^{-5}	7.3424×10^{-3}
0.8	2.5494×10^{-5}	5.0491×10^{-3}	6.3783×10^{-5}	7.9865×10^{-3}
0.9	3.3628×10^{-5}	5.7990×10^{-3}	7.2126×10^{-5}	8.4927×10^{-3}
1.0	4.3075×10^{-5}	6.5632×10^{-3}	7.7514×10^{-5}	8.8042×10^{-3}

Table 10: Results for the first architecture in the first case of the 1-dimensional Navier-Stokes Equations

	$u(x,t)$		$p(x,t)$	
Time	**MSE**	L^2 **error**	**MSE**	L^2 **error**
0.0	9.6309×10^{-9}	9.8137×10^{-5}	2.4499×10^{-6}	1.5652×10^{-3}
0.1	2.4987×10^{-7}	4.9987×10^{-4}	2.4866×10^{-6}	1.5769×10^{-3}
0.2	7.4725×10^{-7}	8.6443×10^{-4}	3.3586×10^{-6}	1.8326×10^{-3}
0.3	1.4043×10^{-6}	1.1850×10^{-3}	5.0944×10^{-6}	2.2571×10^{-3}
0.4	2.2494×10^{-6}	1.4998×10^{-3}	7.7559×10^{-6}	2.7849×10^{-3}
0.5	3.3342×10^{-6}	1.8260×10^{-3}	1.1329×10^{-5}	3.3658×10^{-3}
0.6	4.7054×10^{-6}	2.1692×10^{-3}	1.5696×10^{-5}	3.9618×10^{-3}
0.7	6.3922×10^{-6}	2.5283×10^{-3}	2.0645×10^{-5}	4.5437×10^{-3}
0.8	8.3984×10^{-6}	2.8980×10^{-3}	2.5889×10^{-5}	5.0881×10^{-3}
0.9	1.0697×10^{-5}	3.2707×10^{-3}	3.1097×10^{-5}	5.5765×10^{-3}
1.0	1.3233×10^{-5}	3.6377×10^{-3}	3.5934×10^{-5}	5.9945×10^{-3}

Table 11: Results for the second architecture in the first case of the 1-dimensional Navier-Stokes Equations

	$u(x,t)$		$p(x,t)$	
Time	**MSE**	**L^2 error**	**MSE**	**L^2 error**
0.0	6.0640×10^{-7}	7.7872×10^{-4}	1.1935×10^{-4}	1.0925×10^{-2}
0.1	1.0510×10^{-5}	3.2419×10^{-3}	2.1964×10^{-4}	1.4820×10^{-2}
0.2	3.9396×10^{-5}	6.2766×10^{-3}	3.3119×10^{-4}	1.8199×10^{-2}
0.3	8.3766×10^{-5}	9.1524×10^{-3}	4.4435×10^{-4}	2.1080×10^{-2}
0.4	1.3836×10^{-4}	1.1763×10^{-2}	5.5427×10^{-4}	2.3543×10^{-2}
0.5	1.9779×10^{-4}	1.4064×10^{-2}	6.5851×10^{-4}	2.5661×10^{-2}
0.6	2.5804×10^{-4}	1.6064×10^{-2}	7.5561×10^{-4}	2.7488×10^{-2}
0.7	3.1720×10^{-4}	1.7810×10^{-2}	8.4431×10^{-4}	2.9057×10^{-2}
0.8	3.7555×10^{-4}	1.9379×10^{-2}	9.2328×10^{-4}	3.0386×10^{-2}
0.9	4.3517×10^{-4}	2.0861×10^{-2}	9.9122×10^{-4}	3.1484×10^{-2}
1.0	4.9947×10^{-4}	2.2349×10^{-2}	1.0470×10^{-3}	3.2358×10^{-2}

Table 12: Results for the third architecture in the first case of the 1-dimensional Navier-Stokes Equations

	$u(x,t)$		$p(x,t)$	
Time	**MSE**	**L^2 error**	**MSE**	**L^2 error**
0.0	1.2035×10^{-7}	3.4692×10^{-4}	1.1034×10^{-5}	3.3217×10^{-3}
0.1	3.4170×10^{-7}	5.8455×10^{-4}	7.1700×10^{-6}	2.6777×10^{-3}
0.2	9.7516×10^{-7}	9.8750×10^{-4}	5.3049×10^{-6}	2.3032×10^{-3}
0.3	2.1471×10^{-6}	1.4653×10^{-3}	5.0461×10^{-6}	2.2464×10^{-3}
0.4	4.0106×10^{-6}	2.0026×10^{-3}	6.1677×10^{-6}	2.4835×10^{-3}
0.5	6.6570×10^{-6}	2.5801×10^{-3}	8.5381×10^{-6}	2.9220×10^{-3}
0.6	1.0104×10^{-5}	3.1787×10^{-3}	1.2098×10^{-5}	3.4782×10^{-3}
0.7	1.4311×10^{-5}	3.7829×10^{-3}	1.6868×10^{-5}	4.1071×10^{-3}
0.8	1.9196×10^{-5}	4.3813×10^{-3}	2.2987×10^{-5}	4.7945×10^{-3}
0.9	2.4648×10^{-5}	4.9647×10^{-3}	3.0768×10^{-5}	5.5469×10^{-3}
1.0	3.0525×10^{-5}	5.5249×10^{-3}	4.0787×10^{-5}	6.3865×10^{-3}

Table 13: Results for the fourth architecture in the first case of the 1-dimensional Navier-Stokes Equations

Case 2:

For the first case the following exact solution $u(x,t)$ and $p(x,t)$ was considered

$$u(x,t) = e^t(x - x^2)$$

$$p(x,t) = -\sin(\pi x)$$

which, for $\nu = 1$, gives the PDE below

$$u_t + uu_x - u_{xx} + p_x = e^t(x - x^2) + e^{2t}(x + x^2)(1 - 2x) + 2e^t - \pi\cos(\pi x) \tag{6}$$

When solving Equation 6 with the architectures explained above, the following results were obtained

	$u(x,t)$		$p(x,t)$	
Time	**MSE**	**L^2 error**	**MSE**	**L^2 error**
0.0	1.5241×10^{-6}	1.2345×10^{-3}	8.6635×10^{-3}	9.3078×10^{-2}
0.1	6.5389×10^{-6}	2.5571×10^{-3}	1.4381×10^{-2}	1.1992×10^{-1}
0.2	2.1865×10^{-5}	4.6760×10^{-3}	2.2393×10^{-2}	1.4964×10^{-1}
0.3	4.9040×10^{-5}	7.0028×10^{-3}	3.3283×10^{-2}	1.8244×10^{-1}
0.4	9.0032×10^{-5}	9.4885×10^{-3}	4.7784×10^{-2}	2.1859×10^{-1}
0.5	1.4730×10^{-4}	1.2137×10^{-2}	6.6815×10^{-2}	2.5849×10^{-1}
0.6	2.2385×10^{-4}	1.4962×10^{-2}	9.1519×10^{-2}	3.0252×10^{-1}
0.7	3.2333×10^{-4}	1.7981×10^{-2}	1.2330×10^{-1}	3.5115×10^{-1}
0.8	4.5008×10^{-4}	2.1215×10^{-2}	1.6389×10^{-1}	4.0483×10^{-1}
0.9	6.0920×10^{-4}	2.4682×10^{-2}	2.1535×10^{-1}	4.6406×10^{-1}
1.0	8.0655×10^{-4}	2.8400×10^{-2}	2.8017×10^{-1}	5.2931×10^{-1}

Table 14: Results for the first architecture in the second case of the 1-dimensional Navier-Stokes Equations

	$u(x,t)$		$p(x,t)$	
Time	**MSE**	**L^2 error**	**MSE**	**L^2 error**
0.0	8.5628×10^{-7}	9.2535×10^{-4}	2.8796×10^{-3}	5.3662×10^{-2}
0.1	2.4453×10^{-6}	1.5638×10^{-3}	3.2165×10^{-3}	5.6714×10^{-2}
0.2	3.6890×10^{-6}	1.9207×10^{-3}	3.7059×10^{-3}	6.0876×10^{-2}
0.3	4.4040×10^{-6}	2.0986×10^{-3}	4.3900×10^{-3}	6.6257×10^{-2}
0.4	5.3140×10^{-6}	2.3052×10^{-3}	5.3268×10^{-3}	7.2985×10^{-2}
0.5	6.9950×10^{-6}	2.6448×10^{-3}	6.5958×10^{-3}	8.1214×10^{-2}
0.6	9.9864×10^{-6}	3.1601×10^{-3}	8.3072×10^{-3}	9.1144×10^{-2}
0.7	1.5002×10^{-5}	3.8732×10^{-3}	1.0615×10^{-2}	1.0303×10^{-1}
0.8	2.2719×10^{-5}	4.7664×10^{-3}	1.3734×10^{-2}	1.1719×10^{-1}
0.9	3.3337×10^{-5}	5.7738×10^{-3}	1.7967×10^{-2}	1.3404×10^{-1}
1.0	4.6836×10^{-5}	6.8437×10^{-3}	2.3737×10^{-2}	1.5407×10^{-1}

Table 15: Results for the second architecture in the second case of the 1-dimensional Navier-Stokes Equations

	$u(x,t)$		$p(x,t)$	
Time	**MSE**	**L^2 error**	**MSE**	**L^2 error**
0.0	1.7706×10^{-6}	1.3306×10^{-3}	5.9847×10^{-3}	7.7361×10^{-2}
0.1	2.4228×10^{-5}	4.9222×10^{-3}	2.0570×10^{-2}	1.4342×10^{-1}
0.2	1.0285×10^{-4}	1.0141×10^{-2}	4.6340×10^{-2}	2.1527×10^{-1}
0.3	2.6243×10^{-4}	1.6200×10^{-2}	8.6240×10^{-2}	2.9367×10^{-1}
0.4	5.3076×10^{-4}	2.3038×10^{-2}	1.4432×10^{-1}	3.7990×10^{-1}
0.5	9.3961×10^{-4}	3.0653×10^{-2}	2.2584×10^{-1}	4.7522×10^{-1}
0.6	1.5272×10^{-3}	3.9080×10^{-2}	3.3733×10^{-1}	5.8080×10^{-1}
0.7	2.3419×10^{-3}	4.8394×10^{-2}	4.8673×10^{-1}	6.9766×10^{-1}
0.8	3.4460×10^{-3}	5.8703×10^{-2}	6.8356×10^{-1}	8.2678×10^{-1}
0.9	4.9188×10^{-3}	7.0134×10^{-2}	9.3924×10^{-1}	9.6914×10^{-1}
1.0	6.8590×10^{-3}	8.2819×10^{-2}	1.2677	1.1259

Table 16: Results for the third architecture in the second case of the 1-dimensional Navier-Stokes Equations

	$u(x,t)$		$p(x,t)$	
Time	**MSE**	**L^2 error**	**MSE**	**L^2 error**
0.0	2.3821×10^{-6}	1.5434×10^{-3}	6.6235×10^{-3}	8.1385×10^{-2}
0.1	3.3026×10^{-5}	5.7468×10^{-3}	2.2045×10^{-2}	1.4848×10^{-1}
0.2	1.3701×10^{-4}	1.1705×10^{-2}	5.0095×10^{-2}	2.2382×10^{-1}
0.3	3.4115×10^{-4}	1.8470×10^{-2}	9.4570×10^{-2}	3.0752×10^{-1}
0.4	6.7963×10^{-4}	2.6070×10^{-2}	1.6022×10^{-1}	4.0027×10^{-1}
0.5	1.1959×10^{-3}	3.4581×10^{-2}	2.5301×10^{-1}	5.0300×10^{-1}
0.6	1.9448×10^{-3}	4.4100×10^{-2}	3.8045×10^{-1}	6.1681×10^{-1}
0.7	2.9953×10^{-3}	5.4729×10^{-2}	5.5208×10^{-1}	7.4302×10^{-1}
0.8	4.4333×10^{-3}	6.6583×10^{-2}	7.8000×10^{-1}	8.8318×10^{-1}
0.9	6.3649×10^{-3}	7.9780×10^{-2}	1.0797	1.0391
1.0	8.9200×10^{-3}	9.4446×10^{-2}	1.4709	1.2128

Table 17: Results for the fourth architecture in the second case of the 1-dimensional Navier-Stokes Equations

4.3 2-dimentional Navier-Stokes Equations

For the 2-dimentional Navier-Stokes Equations, only one case was implemeneted with the listed five architectures:

- Architecture 1: 5 hidden layers with 50 neurons each
- Architecture 2: 8 hidden layers with 20 neurons each

- Architecture 3: 8 hidden layers with 50 neurons each
- Architecture 3: 8 hidden layers with 70 neurons each
- Architecture 5: 8 hidden layers with 100 neurons each

Case 1:

For this case, a stream-function forulation was used, were a function $\psi(x,t)$ was defined as

$$U = \nabla \times \psi(x,t)$$

therefore

$$u = \psi_y$$
$$v = -\psi_x$$

For this case, the following exact solution $\psi(x,t)$ and $p(x,t)$ was considered

$$\psi(x,t) = (1 - x^2y^2)e^{-t}$$
$$p(x,t) = xy$$

which, for $\nu = 1$, gives the PDE below

$$\begin{aligned} u_t + uu_x + vu_y - u_{xx} - u_{yy} + p_x &= 2x^2ye^{-t} + 12x^3y^2e^{-t} + 4ye^{-t} + y \\ v_t + uv_x + vv_y - v_{xx} - v_{yy} + p_y &= -2xy^2e^{-t} + 4x^2y^3e^{-t} - 4ye^{-t} + x \end{aligned} \tag{7}$$

When solving Equation 7 with the architectures explained above, the following results were obtained

	$\psi(x,t)$		$p(x,t)$	
time	**MSE**	L^2 **error**	**MSE**	L^2 **error**
0.0	1.7342×10^{-5}	8.3287×10^{-3}	2.6630×10^{-2}	3.2637×10^{-1}
0.1	1.1646×10^{-3}	6.8252×10^{-2}	1.0959×10^{-2}	2.0937×10^{-1}
0.2	3.8980×10^{-3}	1.2487×10^{-1}	8.0858×10^{-3}	1.7984×10^{-1}
0.3	7.4127×10^{-3}	1.7219×10^{-1}	1.0201×10^{-2}	2.0200×10^{-1}
0.4	1.1247×10^{-2}	2.1210×10^{-1}	1.4190×10^{-2}	2.3824×10^{-1}
0.5	1.5156×10^{-2}	2.4622×10^{-1}	1.9096×10^{-2}	2.7637×10^{-1}
0.6	1.9022×10^{-2}	2.7584×10^{-1}	2.4764×10^{-2}	3.1473×10^{-1}
0.7	2.2797×10^{-2}	3.0198×10^{-1}	3.1217×10^{-2}	3.5337×10^{-1}
0.8	2.6465×10^{-2}	3.2536×10^{-1}	3.8418×10^{-2}	3.9201×10^{-1}
0.9	3.0017×10^{-2}	3.4651×10^{-1}	4.6221×10^{-2}	4.2998×10^{-1}
1.0	3.3443×10^{-2}	3.6575×10^{-1}	5.4403×10^{-2}	4.6649×10^{-1}

Table 18: Results for the first architecture for the 2-dimesnional Navier Stokes Equations

	$\psi(x,t)$		$p(x,t)$	
time	**MSE**	L^2 **error**	**MSE**	L^2 **error**
0.0	9.9230×10^{-5}	1.9923×10^{-2}	1.5095×10^{-2}	2.4572×10^{-1}
0.1	8.7404×10^{-4}	5.9128×10^{-2}	8.5973×10^{-3}	1.8544×10^{-1}
0.2	2.6156×10^{-3}	1.0229×10^{-1}	7.6538×10^{-3}	1.7497×10^{-1}
0.3	5.0194×10^{-3}	1.4170×10^{-1}	9.5980×10^{-3}	1.9594×10^{-1}
0.4	7.8569×10^{-3}	1.7728×10^{-1}	1.3048×10^{-2}	2.2845×10^{-1}
0.5	1.0956×10^{-2}	2.0934×10^{-1}	1.7481×10^{-2}	2.6443×10^{-1}
0.6	1.4186×10^{-2}	2.3821×10^{-1}	2.2916×10^{-2}	3.0276×10^{-1}
0.7	1.7444×10^{-2}	2.6415×10^{-1}	2.9685×10^{-2}	3.4459×10^{-1}
0.8	2.0647×10^{-2}	2.8738×10^{-1}	3.8272×10^{-2}	3.9127×10^{-1}
0.9	2.3724×10^{-2}	3.0805×10^{-1}	4.9202×10^{-2}	4.4363×10^{-1}
1.0	2.6613×10^{-2}	3.2627×10^{-1}	6.2971×10^{-2}	5.0188×10^{-1}

Table 19: Results for the second architecture for the 2-dimesnional Navier Stokes Equations

	$\psi(x,t)$		$p(x,t)$	
time	**MSE**	L^2 **error**	**MSE**	L^2 **error**
0.0	2.1447×10^{-5}	9.2622×10^{-3}	2.3688×10^{-2}	3.0782×10^{-1}
0.1	1.1856×10^{-3}	6.8865×10^{-2}	1.1292×10^{-2}	2.1253×10^{-1}
0.2	3.8643×10^{-3}	1.2433×10^{-1}	8.9173×10^{-3}	1.8886×10^{-1}
0.3	7.1762×10^{-3}	1.6943×10^{-1}	1.0742×10^{-2}	2.0729×10^{-1}
0.4	1.0635×10^{-2}	2.0625×10^{-1}	1.4303×10^{-2}	2.3919×10^{-1}
0.5	1.4016×10^{-2}	2.3678×10^{-1}	1.8837×10^{-2}	2.7449×10^{-1}
0.6	1.7252×10^{-2}	2.6269×10^{-1}	2.4307×10^{-2}	3.1181×10^{-1}
0.7	2.0354×10^{-2}	2.8534×10^{-1}	3.0877×10^{-2}	3.5144×10^{-1}
0.8	2.3374×10^{-2}	3.0577×10^{-1}	3.8664×10^{-2}	3.9327×10^{-1}
0.9	2.6369×10^{-2}	3.2477×10^{-1}	4.7642×10^{-2}	4.3654×10^{-1}
1.0	2.9401×10^{-2}	3.4293×10^{-1}	5.7633×10^{-2}	4.8014×10^{-1}

Table 20: Results for the third architecture for the 2-dimesnional Navier Stokes Equations

	$\psi(x,t)$		$p(x,t)$	
time	**MSE**	L^2 **error**	**MSE**	L^2 **error**
0.0	2.0811×10^{-5}	9.1239×10^{-3}	1.4456×10^{-2}	2.4046×10^{-1}
0.1	1.0758×10^{-3}	6.5599×10^{-2}	8.7353×10^{-3}	1.8693×10^{-1}
0.2	3.4794×10^{-3}	1.1797×10^{-1}	8.1228×10^{-3}	1.8025×10^{-1}
0.3	6.4367×10^{-3}	1.6046×10^{-1}	1.0359×10^{-2}	2.0356×10^{-1}
0.4	9.4849×10^{-3}	1.9478×10^{-1}	1.4200×10^{-2}	2.3833×10^{-1}
0.5	1.2373×10^{-2}	2.2247×10^{-1}	1.8950×10^{-2}	2.7532×10^{-1}
0.6	1.4979×10^{-2}	2.4478×10^{-1}	2.4203×10^{-2}	3.1114×10^{-1}
0.7	1.7252×10^{-2}	2.6269×10^{-1}	2.9710×10^{-2}	3.4473×10^{-1}
0.8	1.9177×10^{-2}	2.7696×10^{-1}	3.5310×10^{-2}	3.7582×10^{-1}
0.9	2.0757×10^{-2}	2.8814×10^{-1}	4.0887×10^{-2}	4.0441×10^{-1}
1.0	2.2000×10^{-2}	2.9665×10^{-1}	4.6361×10^{-2}	4.3063×10^{-1}

Table 21: Results for the fourth architecture for the 2-dimesnional Navier Stokes Equations

	$\psi(x,t)$		$p(x,t)$	
time	**MSE**	**L^2 error**	**MSE**	**L^2 error**
0.0	1.6969×10^{-5}	8.2386×10^{-3}	1.9618×10^{-2}	2.8013×10^{-1}
0.1	9.4158×10^{-4}	6.1370×10^{-2}	9.3919×10^{-3}	1.9382×10^{-1}
0.2	3.1448×10^{-3}	1.1216×10^{-1}	8.0040×10^{-3}	1.7893×10^{-1}
0.3	5.9426×10^{-3}	1.5418×10^{-1}	1.0275×10^{-2}	2.0273×10^{-1}
0.4	8.9039×10^{-3}	1.8872×10^{-1}	1.4243×10^{-2}	2.3869×10^{-1}
0.5	1.1787×10^{-2}	2.1714×10^{-1}	1.9205×10^{-2}	2.7717×10^{-1}
0.6	1.4485×10^{-2}	2.4071×10^{-1}	2.4883×10^{-2}	3.1549×10^{-1}
0.7	1.6976×10^{-2}	2.6059×10^{-1}	3.1112×10^{-2}	3.5277×10^{-1}
0.8	1.9294×10^{-2}	2.7781×10^{-1}	3.7755×10^{-2}	3.8862×10^{-1}
0.9	2.1502×10^{-2}	2.9327×10^{-1}	4.4696×10^{-2}	4.2283×10^{-1}
1.0	2.3676×10^{-2}	3.0774×10^{-1}	5.1843×10^{-2}	4.5538×10^{-1}

Table 22: Results for the fifth architecture for the 2-dimesnional Navier Stokes Equations

5 Conclussions

When looking at tables 1 - 9, the first thing that one can visualize is how, for all three cases, the PINNs implemented had a really good performance, approximating the exact proposed solution very closely and obtaining very positive indicators of both MSE and $L^2 Error$. However, it is interesting to see how for each case, it can be seen how the architecture implemented had no effect on the error evolution through the time domain, allowing to conclude that for small enough problems, the implemented architecture does not affect the proposed solution when the architecture is big enough for it to capture the nature of the problem. Another interesting conclusion that can be drawn from these results is that, although the architecture becomes larger, it does not lead to the common problem of overfitting, since by the nature of PINNs, they are trained mainly on the residuals of the underlying equations of the phenomenon and therefore, overfitting to the training data is no longer a possible problem.

Now when looking at both 1-dimensional and 2-dimensional Navier-Stokes Equations in tables 10 - 22, there are multiple points to address. Firstly, it is interesting to see how, here we have a similar behavior to other numerical methods for solving time-dependent PDEs, in which the error grows as the time domain progresses, this can be due to a great concentration of data points on the initial scenario, for which there is not only the points used for the training process using the residual of the Equations, but also the initial data provided by the problem itself.

Finally, it is worth noting the difference between the results for the pressure field and the velocity field, in which it can be seen how the neural network obtains a fairly acceptable approximation for the velocity field while presenting relatively large errors for the pressure.

This may be due to the fact that, since the network is trained with initial and boundary data for the velocity, while it is only trained with boundary values for the pressure, it can focus mainly on the correct estimation of the velocity, without really worrying about a very good fit for the pressure. Therefore, the possibility is raised, as a future research, to implement not one but three PINNs, following a methodology similar to the one proposed by Chorin [14] in which the problem is divided into three stages, the first one an estimation of a candidate velocity, the second one the estimation of the pressure field based on the projection of this candidate velocity to the divergence-free field and the third one the corrected velocity estimation.

Acknowledgements

I have no words to thank my brother Samuel Salazar, who has accompanied me on this path of research and thanks to whom I was motivated to do this research in the field of fluid dynamics. I also want to thank my father Wilson Salazar and my mother Patricia Arango for being those who throughout my life have always fanned the flames of curiosity in me and finally I want to thank all my teachers of the career for being those who have motivated in me the desire to investigate and seek new problems on which to work in this beautiful field that is applied mathematics.

References

[1] Logan JD. Applied partial differential equations. Springer; 2014.

[2] Farlow SJ. Partial differential equations for scientists and engineers. Courier Corporation; 1993.

[3] Strauss WA. Partial differential equations: An introduction. John Wiley & Sons; 2007.

[4] Hesthaven JS, Ubbiali S. Non-intrusive reduced order modeling of nonlinear problems using neural networks. Journal of Computational Physics. 2018;363:55-78.

[5] Quarteroni A. Physics-Based and Data-Driven-Based Algorithms for the Simulation of the Heart Function; 2023. .

[6] Berg J, Nyström K. Neural networks as smooth priors for inverse problems for PDEs. Journal of Computational Mathematics and Data Science. 2021;1:100008.

[7] Meng X, Karniadakis GE. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems. Journal of Computational Physics. 2020;401:109020.

[8] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics. 2019;378:686-707.

[9] Moschou SP, Hicks E, Parekh R, Mathew D, Majumdar S, Vlahakis N. Physics-Informed Neural Networks for modeling astrophysical shocks. Machine Learning: Science and Technology. 2023.

[10] Xiao MJ, Yu TC, Zhang YS, Yong H. Physics-informed neural networks for the Reynolds-Averaged Navier–Stokes modeling of Rayleigh–Taylor turbulent mixing. Computers & Fluids. 2023:106025.

[11] Lino M, Fotiadis S, Bharath AA, Cantwell CD. Current and emerging deep-learning methods for the simulation of fluid dynamics. Proceedings of the Royal Society A. 2023;479(2275):20230058.

[12] Zhu C, Byrd RH, Lu P, Nocedal J. Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization. ACM Trans Math Softw. 1997 dec;23(4):550–560. Available from: `https://doi.org/10.1145/279232.279236`.

[13] Google. General-purpose machine family for Compute Engine;. Accessed: 2023-11-10. `https://cloud.google.com/compute/docs/general-purpose-machines#e2-high-mem`.

[14] Chorin AJ. Numerical solution of the Navier-Stokes equations. Mathematics of computation. 1968;22(104):745-62.