UFSM

CIÊNCIA e NATURA

ISSN 2179-460X
OPEN ACCESS

Ci.e Nat., Santa Maria, v. 45, spe. n. 3, e74615, p. 01-24, 2023 • https://doi.org/10.5902/2179460X74615
Submissão: 03/20/2023 • Aprovação: 03/23/2023 • Publicação: 12/01/2023

Special Edition

Parameter estimation in the pollutant dispersion problem with Physics-Informed Neural Networks

Estimação de parâmetros no problema de dispersão de poluentes com Redes Neurais Informadas por Física

Roberto Mamud Guedes da Silva[I], **Helio dos Santos Migon**[I,II], **Antônio José da Silva Neto**[II]

[I] Universidade Federal do Rio de Janeiro, Macaé, RJ, Brazil
[II] Rio de Janeiro State University, Nova Friburgo, RJ, Brazil

ABSTRACT

In this work[1], the inverse problem of parameter estimation in the advection-dispersion-reaction equation, modeling the pollutant dispersion in a river, is studied with a Neural Network approach. In the direct problem, the dispersion, velocity and reaction parameters are known and then the initial and boundary value problems are solved by classical numerical methods, where they are used as input datasets for the inverse problem and formulation. In the inverse problem, we know the dispersion and velocity parameters and also the information about the pollutant concentration from the synthetic experimental data, and then the aim is to estimate the reaction parameter in the advection-dispersion-reaction equation. This inverse problem is solved by a usual Artificial Neural Network (ANN) and by a Physics-Informed Neural Network (PINN), which is a special type of neural network that includes in its formulation the physical laws that describe the phenomena involved. Numerical experiments with both the ANN and PINN are presented, demonstrating the feasibility of the approach considered.

Keywords: Inverse problem; Parameter estimation; Physics-Informed Neural Network

RESUMO

Neste trabalho, é estudado o problema inverso de estimação de parâmetros na equação de advecção-dispersão-reação, modelando a dispersão de poluentes em um rio, com uma abordagem por Redes Neurais. No problema direto, os parâmetros de dispersão, velocidade e reação são conhecidos e, então, o problema de valor inicial e de fronteira é resolvido empregando métodos numéricos clássicos, onde esta solução é usada como dado de entrada para a formulação do problema inverso. No problema

[1] The original version of this work was presented in and published in the Proceedings of the joint events XXV ENMC – Encontro Nacional de Modelagem Computacional, XIII ECTM – Encontro de Ciências e Tecnologia de Materiais, 9º MCSul - Conferência Sul em Modelagem Computacional e IX SEMENGO - Seminário e Workshop em Engenharia Oceânica, October 19-21, 2022

inverso, são conhecidos os parâmetros de dispersão e velocidade, bem como a informação sobre a concentração do poluente a partir de dados experimentais sintéticos, onde o objetivo é estimar o parâmetro de reação na equação de advecção-dispersão-reação. Este problema inverso é resolvido por uma Rede Neural Artificial (ANN - Artificial Neural Network) e por uma Rede Neural Informada por Física (PINN - Physics-Informed Neural Network), sendo esta um tipo especial de rede neural que inclui, em sua formualção, as leis físicas que descrevem o fenômeno envolvido. Experimentos numéricos relacionados com ambas ANN e PINN são apresentados, demonstrando a viabilidade da abordagem considerada.

Palavras-chave: Problema Inverso; Estimação de parâmetros; Redes Neurais Informadas por Física

1 INTRODUCTION

Over the last centuries, with the development of industrial production, starting with the industrial revolution, the world was capable of producing in larger quantities, improving the quality process, and reducing the time of goods production. However, with this increasing industrial process, pollution has also increased, generating, in many cases, contaminants in the air, soil, and water.

Unfortunately, the leakage of pollutants into bodies of water like rivers, lakes, or oceans is a common occurrence nowadays. This kind of pollution source includes industrial waste, oil refineries, and waste water treatment facilities. Here, we consider the model posed by the advection-dispersion-reaction equation, which allows the computation of the concentration of a pollutant in a river.

In this context, the area of inverse problems emerges as a tool in the study of these leakages which is instrumental in the prevention and/or design and planning of remediation measures to reduce their impact. For example, when it is observed that there is a pollutant in some region of the river, it is important to understand how this concentration is evolving along the river. In the Inverse Problem framework, we say that from some measurements, we are interested in estimating the reaction parameter of the advection-dispersion-reaction equation. This kind of inverse problem is posed in contrast with the so-called direct problem (Moura Neto and Silva Neto, 2013). In the example above, the direct problem can be formulated as, given the parameters of the

medium, we are interested in measure the pollution concentration in any region of the river.

The direct problem is a well-posed problem in the Hadamard sense (Hadamard, 1902), while the inverse problem is usually ill-posed in view of the lack of uniqueness (Isakov, 1990). Besides, the numerical methods employed in each formulation (direct and inverse) must be different to avoid the so-called Inverse Crime. According to (Colton and Kress, 2013), it is crucial that the synthetic data obtained by a forward solver must not have any connection to the inverse solver under consideration.

In this work, we employ classical numerical methods to solve the direct problem and two kinds of Neural Network to solve the inverse problem, the Artificial Neural Network (ANN) and the Physics-Informed Neural Network (PINN). The PINN is a recent type of neural networks introduced in a recent work (Raissi; Perdikaris; Karniadakis, 2019). These nets have the characteristic that they take into account the physical laws that describe the phenomena involved, along with the process of training to solve some learning tasks. It can be made by modifying the mean squared error loss (the function which we are interested in minimizing to obtain the weights and biases for the neural network). This modification is made by adding a term representing the physical laws of the model through linear or non-linear Partial Differential Equations (PDE's), where the derivatives of the network are computed using Automatic Differentiation (Baydin *et al.*, 2018). In other words, the PINN can be derived by applying the chain rule to differentiate compositions of functions using the automatic differentiation technique. Besides, this recent network and that one that represents the function solution have shared parameters, in which can be learned by minimizing the mean squared error loss (Raissi; Perdikaris; Karniadakis, 2019).

In this work, we study the concentration distribution of a pollutant in a river governed by the advection-dispersion-reaction equation. The forward problem is studied through classical numerical methods with the goal of generating an artificial dataset to be used in the inverse problem framework with neural networks that take

into account the model equation. In Section 2, we consider the formulation of direct and inverse problems for the pollutant concentration computation and the reaction coefficient determination, respectively. In Section 3, we study the direct problem and the dataset generated, as well as the corresponding inverse problem approach by ANN. The Section 4 is devoted to the formulation of the PINN for the inverse problem. In Section 5, we present numerical experiments related to the parameter estimation of the reaction term in the advection-dispersion-reaction equation. Conclusions and extensions are described in Section 6.

2 THE POLLUTANT DISPERSION DIRECT AND INVERSE PROBLEMS

In this section, we establish the forward and inverse problem formulations for the pollution source concentration computation and the reaction coefficient determination, respectively.

2.1 The Problem Considered

Consider the following one-dimensional transient problem:

$$\begin{cases} \partial_t c - D\partial_{xx} c + V\partial_x c + Rc = 0, \text{ for } x \in (0, L), t \in (0, T) \\ c(x, 0) = g_0(x), \text{ for } x \in [0, L] \\ c(0, t) = g_l, \text{ for } t \in (0, T] \\ c(L, t) = g_r, \text{ for } t \in (0, T], \end{cases} \tag{1}$$

where ∂_t, ∂_x and ∂_{xx} represent the partial derivatives of concentration c [g/m^3] with respect to time, t [s], and space x [m], respectively. Beside this, L [m] is the length of the domain, D [m^2/s] is the dispersion coefficient, V [m/s] is the velocity, R [$1/s$] is the reaction coefficient, g_o is the initial concentration and g_l and g_r are the known concentration values at the boundary, in which are valid the compatibility conditions $g_l(0) = g_o(0)$ and $g_r(0) = g_o(L)$. This problem has a unique solution, provided that u, g_o, g_l and g_r belong to appropriated function spaces, see for example (Evans, 1998). Note that in the above problem, we consider constant boundary concentrations $g_l \geq 0$ and $g_r \geq 0$, but it could be considered concentrations that change with time.

This kind of problem can represent a leakage scenario along the river represented by the function $g_0(x)$.

2.2 The Direct and Inverse Frameworks Overview

The Direct Problem for Eq. (1) is posed as: given the medium parameters D, V and R, the initial and boundary conditions, we must find the concentration solution $c(z)$, for all $z = (x, t) \in (0, L) \times (0, T]$.

On the other hand, the Inverse Problem studied here is posed as: given the medium parameters of dispersion and velocity in Eq. (1), D and V, the initial and boundary conditions g_o, g_l and g_r, and some measurements of concentrations from sensors, we are interested in determining the reaction parameter, R.

These frameworks will be detailed in the following sections.

3 THE ARTIFICIAL NEURAL NETWORK APPROACH

In this section, we present the direct problem and the dataset generated, the usual Artificial Neural Network (ANN), formed by a Multi-Layer Perceptrons (MLP), and the inverse problem considered for this neural net.

3.1 Direct Problem and the Dataset Generated

Consider the Direct Problem for Eq. (1) with initial condition $g_o(x)$ given by:

$$g_0(x) = (1 - \cos(2\pi x/L))0.5, \quad \text{for } x \in [0, L]. \tag{2}$$

Consider also N_R different reaction parameters R_j, $j = 1, 2, N_R$, with values between 0 and 1 s^{-1}, where the parameters D and V are supposed to be fixed. So, for each R_j, the direct problem is solved with the Finite Element Method built-in functions in *Mathematica'* software, using the routine NDSolve[2], and then generating a numerical solution $c(z, R_j)$. Figure 1 shows an example of this numerical solution for different values of R.

[2] https://reference.wolfram.com/language/tutorial/NDSolvePDE.html

In this way, consider the sets $\{z_i\}_{i=1}^{N_d}$ and $\{R_j\}_{j=1}^{N_R}$, where each element z_i and R_j are sampled from a uniform distribution over the sets $[0, L] \times [0, T]$ and $[0, 1]$, respectively. So, for each R_j, we can determine the numerical solution evaluated in each node z_i, for $i = 1, 2,, N_d$, called $\{c(z_i, R_j)\}_{i=1}^{N_d}$.

Therefore, the full dataset, D, is given by:

$$\mathcal{D} = \left\{ \left\{ \{c(z_i, R_j)\}_{i=1}^{N_d}, R_j \right\} \right\}_{j=1}^{N_R}. \tag{3}$$

Observe that the size of D is $(N_d + 1) \times N_R$.

3.2 The Artificial Neural Network - ANN

Consider a multi-layer perceptron (MLP) fully-connected neural network with $Q \geq 1$ hidden layers as the surrogate model, see, for example, (Goodfellow; Bengio, Y.; Courville, 2017), (Nielsen, 2015), and (Chollet, 2018). The k-th hidden layer, $\mathbf{l}_k \in \mathrm{R}^{N_k \times 1}$, for $k = 1, 2, ..., Q$, and the output of the neural network, $R_{NN} \in \mathrm{R}^1$, are given by:

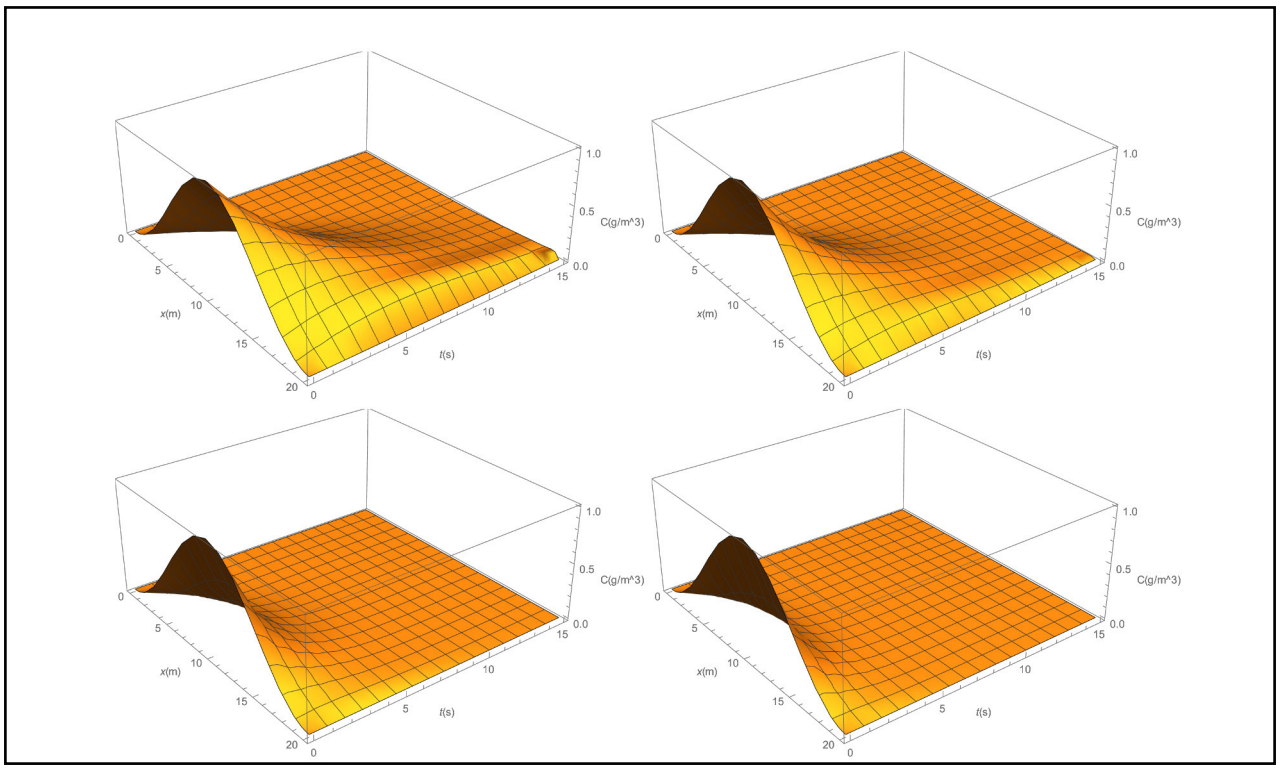
$$\mathbf{l}_0 = \{c(z_i, R_j)\}_{i=1}^{N_d} \tag{4}$$

$$\mathbf{l}_k = \sigma\left(\mathbf{W}_{k-1}.\mathbf{l}_{k-1} + \mathbf{b}_{k-1}\right), \ k = 1, 2, ..., Q, \tag{5}$$

$$R_{NN} = \mathbf{W}_Q.\mathbf{l}_Q + \mathbf{b}_Q, \tag{6}$$

where $\mathbf{W}_k$ are the weight matrices and $\mathbf{b}_k$ the biases vectors for the k-th hidden layer, $\sigma(\cdot)$ is the nonlinear activation function, where we use the hyperbolic tangent function and the Leaky Rectified Linear Unit (Leaky ReLU) function (Maas hannun; NG, 2013). In Section 5, we consider a fully-connected neural network with input of size N_d, four hidden layers and different numbers of neurons. For example, if we consider $Q = 4$ hidden layers with 40 neurons each and $N_d = 100$, then $\mathbf{W}_1 \in \mathrm{R}^{40\times100}$, $\mathbf{b}_1 \in \mathrm{R}^{40\times1}$, $\mathbf{W}_2 \in \mathrm{R}^{40\times40}$, $\mathbf{b}_2 \in \mathrm{R}^{40\times1}$, $\mathbf{W}_3 \in \mathrm{R}^{40\times40}$, $\mathbf{b}_3 \in \mathrm{R}^{40\times1}$, $\mathbf{W}_4 \in \mathrm{R}^{40\times1}$, $\mathbf{b}_4 \in \mathrm{R}^1$, resulting in 40 × 100 +2.(40 × 40) +40 = 7240 weights and 40 + 2.(40) + 1 = 121 biases.

Figure 1 – Numerical Solution of the direct problem (1), for different values of R. Top left: $R = 0.2\,s^{-1}$; Top right: $R = 0.3\,s^{-1}$; Bottom left: $R = 0.6\,s^{-1}$; Bottom right: $R = 0.9\,s^{-1}$. In all cases, we fixed $D = 0.06\,m^2s^{-1}$, $V = 0.7\,ms^{-1}$, $L = 20\,m$, $T = 15\,s$, $g_l = g_r = 0$ and g_o given by Eq. (2). Observe that increasing the value of *R*, the concentration tends to vanish in a shorter time



Source: Author's

Observe that, in the full dataset defined previously in Eq. (3), we can split it in another two sets:

$$\mathcal{D} = \left\{\left\{\{c(z_i, R_j)\}_{i=1}^{N_d}, R_j\right\}\right\}_{j=1}^{Ntrain} \bigcup \left\{\left\{\{c(z_i, R_j)\}_{i=1}^{N_d}, R_j\right\}\right\}_{j=Ntrain+1}^{Ntrain+1+Ntest}, \tag{7}$$

with $Ntrain$ as the number of patterns to be used in the training process and $Ntest$ as the number of patterns to be used after in the validation process, where $Ntest = N_R - Ntrain$. Furthermore, in this work, the training set considered is 70% of the full dataset, i.e., $Ntrain = \lfloor (0.7).N_R \rfloor$ and, therefore, $Ntest = \lfloor (0.3).N_R \rfloor$, where $\lfloor a \rfloor$ is the least integer function that gives the greatest integer that is less than or equal to a.

The training set is defined as a vector of association

$$\{c(z_i, R_j)\}_{i=1}^{N_d} \rightarrow R_j, \tag{8}$$

for $j = 1, 2, ..., N_R$. This set will be used to inform the neural net that, when we have a pollutant concentration $c(z_i, R_j)$ along the river, initially distributed as in Eq. (2), then the reaction parameter is given by R_j. For simplicity in the code for the MLP, used in this work, we write the input as a vector of length N_d. In the case of considering a matrix form, it is possible to use convolutional type network (Lecun; Bengio; Hinton. (2015), Li *et al*. (2022)).

The test set is defined in a similar way with the remaining dataset. This set is important to evaluate or "to test" if the weights and biases found in the optimization process from the training set is good enough to predict a value that is comparable to those in the test set. In other words, after the training process, we can compare $|R^{predict} - R^{exact}|$, where $R^{predict}$ is the computed output of the net, Eq. (6), when the input is $c(z_i, R_j)$ (from the test set) and R^{exact} is the expected correct value (from the test set). If this difference is not small enough, then the net continues the optimization procedure.

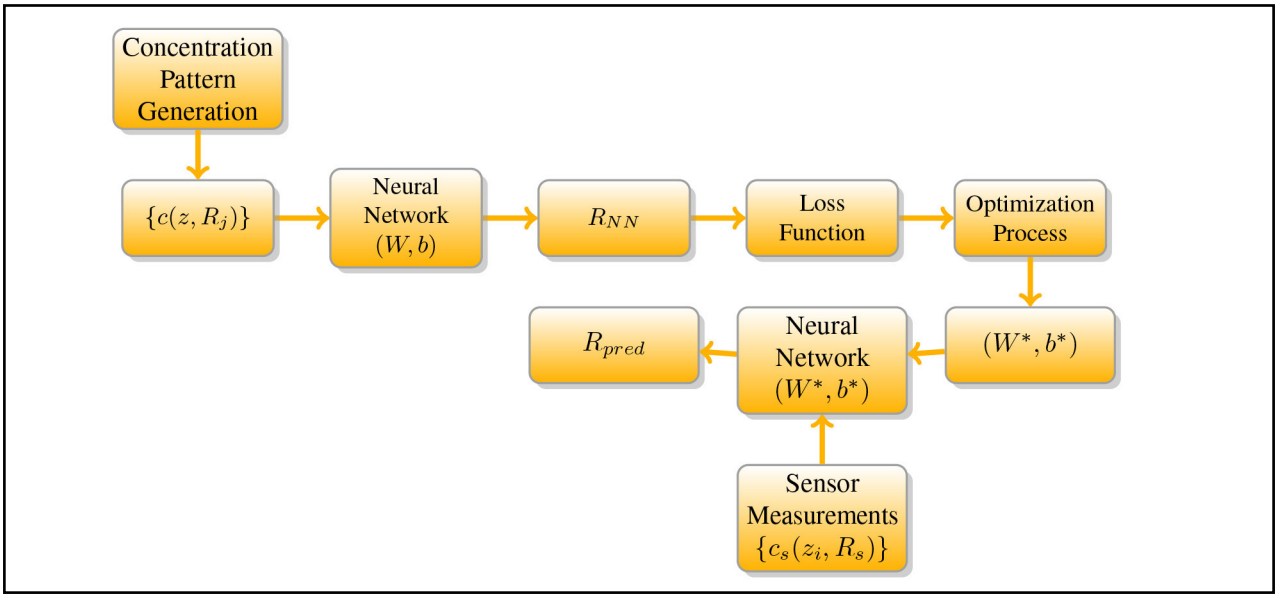
After the process described above, we find the weight matrix, $\mathbf{W}$, and the biases vector, $\mathbf{b}$, that minimizes the loss function. Consequently, we find the output R_{NN}, given by Eq. (6).

The details about the topology of the neural network considered in this work, as well as the activation and the loss functions employed here, will be presented in Section 5.

3.3 The ANN Approach for the Inverse Problem Solution

As presented before, the goal in this work is to estimate the reaction parameter R from a pollutant leakage in a river of length L, using concentration measurements acquired with sensors.

Figure 2 – Flowchart for the Parameter Estimation using the ANN approach. The neural network is created with the list of concentration patterns as input, the list of R-values as target and the function R_{NN} as output. The loss function is created based on the output R_{NN} and the target R-values. After the training process, the optimal values of weights and biases, $W^*$and b^*, are obtained. The predicted value, R_{pred}, is computed through the neural network with sensor measurements as input and the already calculated optimal weights and biases as the parameters



Source: Author's

Here, we consider a total of N_s measurements which is obtained by multiplying the number of sensors by the number of information in time provided by each sensor. In another words, let N_{xs} be the number of sensors distributed over the interval $[0, L]$, where each one is capable of providing information as many times instants as we want. In this way, suppose that a single sensor can provide, for example, N_{ts} measured values of pollutant concentration, that is, we have a total of $N_{xs} \times N_{ts} = N_s$ concentration measurements.

On the other hand, we observe that, since the input dimension of the neural net is N_d, we must consider that the number of sensors, N_{xs}, and the number of measurements provided by each one, N_{ts}, must satisfy the relation

$$N_s = N_{xs} \times N_{ts} = N_d. \tag{9}$$

The procedure to solve the reaction parameter estimation problem by using an ANN is described in Fig. 2, with each step being described next.

In first place, we create a list of reaction coefficient values, $\{R_j\}$, in view of generating a set of concentration patterns with *Mathematica*' software, using the routine NDSolve, $\{c(z, R_j)\}$. This is the dataset to be used.

In second place, we create a neural network that has as input the concentration values and as output the reaction coefficient, R_{NN}. It is important to highlight that the loss function considered here is the mean squared error. After the optimization process, the optimal weights W^* and biases b^* are obtained.

Third, the N_s concentration measurements from sensors are considered. This collection can be obtained from an experimental or a synthetic approach. In this work, we generate the observational data synthetically with Mathematica' Solver, considering a fixed R_s, to be estimated, that is, we solve the direct problem with the value R_s for the reaction coefficient, in order to obtain the numerical solution, computed in N_s nodes, which are randomly spaced in the domain, i.e. $\{c_s(z_i, R_s)\}$.

Fourth, the sensor measurements are used as the input in the neural network with the optimal weights and biases, W^* and b^* .The computed output R_{pred} is the predicted value of R_s.

Since we are dealing with random quantities, we realized the third and fourth steps thirty times, aiming at obtaining more reliable results, and calculated the mean and standard deviation values for R_{pred} considering all runs, called $\overline{R_{pred}}$ and σ_{Rpred}, respectively, and computed the relative error $\frac{|\overline{R_{pred}} - R_s|}{R_s}$ Numerical experiments are presented in Subsection 5.1.

4 THE PHYSICS-INFORMED NEURAL NETWORK APPROACH

As mentioned in the Introduction, the Physics-Informed Neural Network (PINN) is a recent type of neural network which has the feature that, in the optimization process, we search for the weights and biases that minimizes a special modified loss function.

In this section, we present the main ideas about the PINN, and how we use them to solve the inverse problem of parameter estimation.

4.1 The PINN approach for solve the Inverse Problem

Here we will point out the main differences between construct an Artificial Neural Network and a Physics-Informed Neural Network.

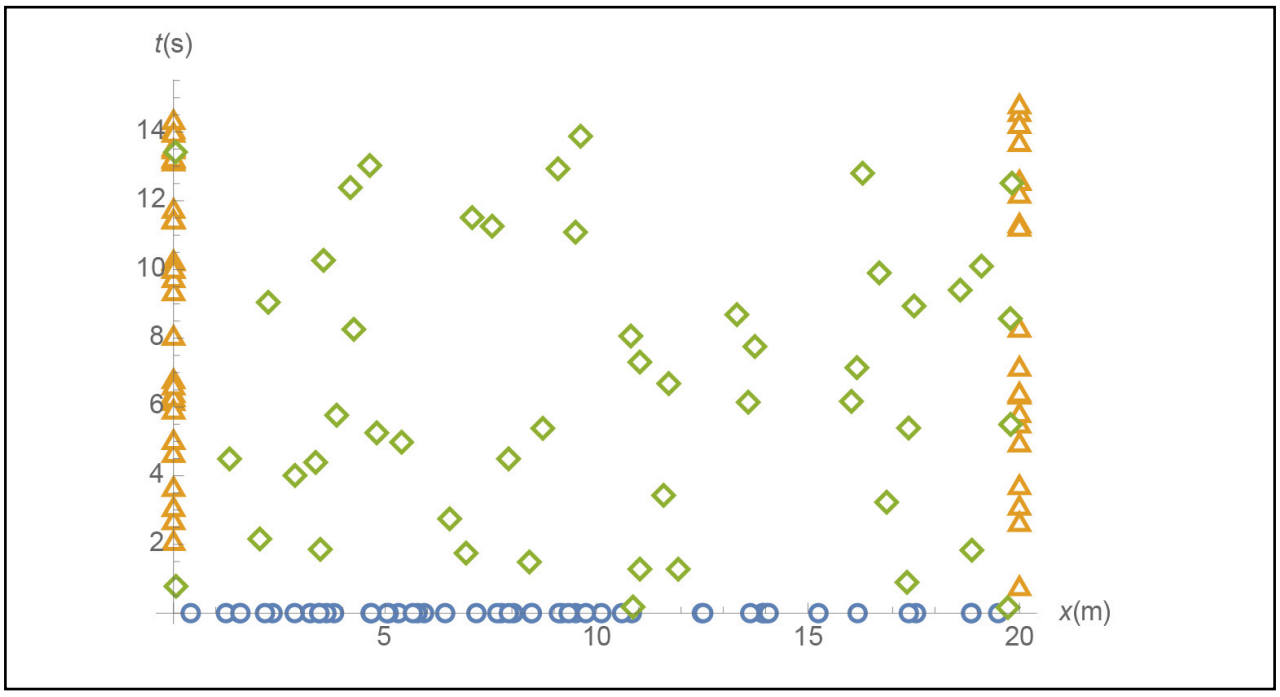
The ANN is composed by a collection of neurons (or nodes) distributed in layers that apply different transformations on their inputs. This neurons are weighted to adjust the training process. The aim in the training phase of an ANN is to find a set of optimal weights that minimizes the error between the prediction (output) and the target values.

When we are dealing with the ANN architecture, we have certain “freedom” to define what are the inputs, the targets and the output of the net. Depending on the architecture, we can change from an usual ANN to a Convolutional Neural Network or a Recurrent Neural Network, for example. In the previous sections, we defined that the pollutant concentration at some nodes would be considered as input and the reaction parameter as the output. In the PINN approach, this architecture will change slightly.

The main difference between the ANN and the PINN are the target values. In the ANN, we provide a list of R-values that was used for adjusting the loss function of Mean Squared Error type. On the other hand, in the PINN, there are no target values because we are interested in minimizing the new loss function that incorporates the physics of the problem through the differential equation that govern the observed data (Raissi *et al.*, 2019).

Consider the problem of pollutant dispersion in a river, posed by Eq. (1). Observe that we have information about the pollutant concentration initially (with the function $g_0(x)$) and on the boundary (with the functions $g_l(z)$ and $g_r(z)$). Furthermore, in the inverse problem context, we can suppose that, after the pollutant leakage, we have information about the concentration from observational data provided by sensors distributed along the river. Correlated to this data from sensors there is a reaction parameter (unknown) R_s that we are interested in estimating. Figure 3 shows an example of points distribution over the domain where we have information about the pollutant concentration. In this example, the points are sampled from a Uniform Distribution over the intervals $x \in [0, 20]$ and $t \in [0, 15]$.

Figure 3 – Example of points distribution in the ANN case. The blue circles represent the data from the initial condition, namely $\{z_i^0\}_{i=1}^{N_0}$. The orange triangles represent the data from the boundary conditions, namely $\{z_i^b\}_{i=1}^{N_b}$. The green diamonds represent the observational data from sensors whose values are correlated to the reaction parameter,R_s, to be determined, namely $\{z_i^s\}_{i=1}^{N_s}$.



Source: Author's

Let us take into account another ANN whose inputs are the nodes (z_i), for $i = 1, 2, \ldots, N_d$, distributed as in Fig.3. We observe that the set of points $\{z_i\}_{i=1}^{N_d}$ can be seen as the collection:

$$\{z_i\}_{i=1}^{N_d} = \{z_i^0\}_{i=1}^{N_0} \bigcup \{z_i^b\}_{i=1}^{N_b} \bigcup \{z_i^s\}_{i=1}^{N_s}, \tag{10}$$

where N_0, N_b and N_s correspond to the number of domain points that generate the initial condition (blue circles as in Fig.3), boundary conditions (orange triangles as in Fig.3) and sensor measurements (green diamonds as in Fig.3), respectively. So, the total number of domain points, N_d, can be view as the sum $N_d = N_0 + N_b + N_s$. Observe that the splitting in Eq.(10) will generate different terms in the loss function definition. Besides, the targets are the corresponding concentrations $c(z_i)$, that is, the initial condition $c(z_i^0) = c(x_i^0, 0) = g_0(x_i)$, for $i = 1, 2, \ldots N_0$, the boundary conditions

$$c(z_i^b) = c(x_i^b, t_i^b) = \begin{cases} g_l(t_i^b), & \text{if } x_i^b = 0 \\ g_r(t_i^b), & \text{if } x_i^b = L, \end{cases} \tag{11}$$

and the sensors measurements $c_s(z_i^s, R_s)$, for $i = 1, 2, \ldots N_s$. The output of this ANN is the function $c_{NN}(z_i, \theta)$, where $\theta = \{\mathbf{W}, \mathbf{b}\}$ is the vector of all parameters of the network.

Therefore, the loss function, $L_d(\theta)$, of Mean Squared Error (MSE) type for the ANN, uses only the data from the problem and from the sensors and is given by:

$$L_d(\theta) = L_0(\theta) + L_b(\theta) + L_s(\theta) \tag{12}$$

that is,

$$L_d(\theta) = \frac{1}{N_0}\sum_{i=1}^{N_0} |c_{NN}(z_i^0, \theta) - g_0(x_i)|^2 + \frac{1}{N_b}\sum_{i=1}^{N_b} |c_{NN}(z_i^b, \theta) - c(z_i^b)|^2 + \frac{1}{N_s}\sum_{i=1}^{N_s} |c_{NN}(z_i^s, \theta) - c_s(z_i^s, R_s)|^2. \tag{13}$$

For the purpose of defining the new loss function for the PINN case, we need to define the following functional, namely residue of Eq. (1):

$$F(u, R) := \partial_t u - D\partial_{xx} u + V\partial_x u + Ru, \tag{14}$$

where u is an appropriated function and $R > 0$ is a constant. We note that if $F(c, R) = 0$, then we say that $u = c$ is a solution (general) of the PDE in Eq. (1), with known values for the parameters D, V and R.

When we are approximating the pollutant concentration solution $c(z, R)$ by a neural network $c_{NN}(z, \theta)$, then it will result in a different approximation with a neural network for F, namely a Physics-Informed Neural Network (PINN) (Raissi *et al*., 2019). This method involves training a neural network to approximate the concentration data in Eq.(8), and, simultaneously, the solution of the PDE in Eq. (1) by minimizing a new loss function that incorporates the residue given by Eq.(14) in a new summation term.

It is important to point out that we are supposing that both in the neural networks' output c_{NN} and in the residue $F(c_{NN}, R)$ the set of parameters θ is the same. So, the parameters θ and R can be optimized by minimizing the new loss function, $L(\theta, R)$, defined by:

$$L(\theta, R) = L_d(\theta) + L_r(\theta, R) \tag{15}$$

where $L_d(\theta)$ is the loss term due to the data, given by Eq.(13), and $L_r(\theta, R)$ is the loss term due to the residue, defined by

$$L_r(\theta, R) = \frac{1}{N_r} \sum_{i=1}^{N_r} |F(c_{NN}(z_i^r, \theta), R)|^2, \tag{16}$$

where $F(c_{NN}(z_i^r, \theta), R)$ is given by (14), substituting u by $c_{NN}(z_i^r, \theta)$, and $\{z_i^r\}_{i=1}^{N_r}$ are collocation points distributed over the entire domain. We observe, that for the PINN case, we append a new set of points, the collocation points. The collocation points are different points, and they are chosen so that the approximation to the solution satisfies the differential equation at those points. With this new set of points, we have the following collection, where an example of this points distribution can be observed in Fig.4:

$$\{z_i\}_{i=1}^{N_d} = \{z_i^0\}_{i=1}^{N_0} \bigcup \{z_i^b\}_{i=1}^{N_b} \bigcup \{z_i^s\}_{i=1}^{N_s} \bigcup \{z_i^r\}_{i=1}^{N_r}, \quad (17)$$

This approach for the construction of the loss function is slightly different from those presented in (Raissi; Perdikaris; Karniadakis, 2019), in which it only considers the data from sensor measurements in the loss term due to the data. In this work, we consider, besides the sensor measurements, the data from initial and boundary conditions in the corresponding loss term.

Figure 5 represents the procedure to solve the reaction parameter estimation problem with PINN, where each step is described next:

In the first place, we create a neural network with $z = (x, t)$ as input and the concentration $c_{NN}(z, \theta)$ as output. We remember that the loss function, see Eqs. (13), (15) and (16), is formed by four summations.

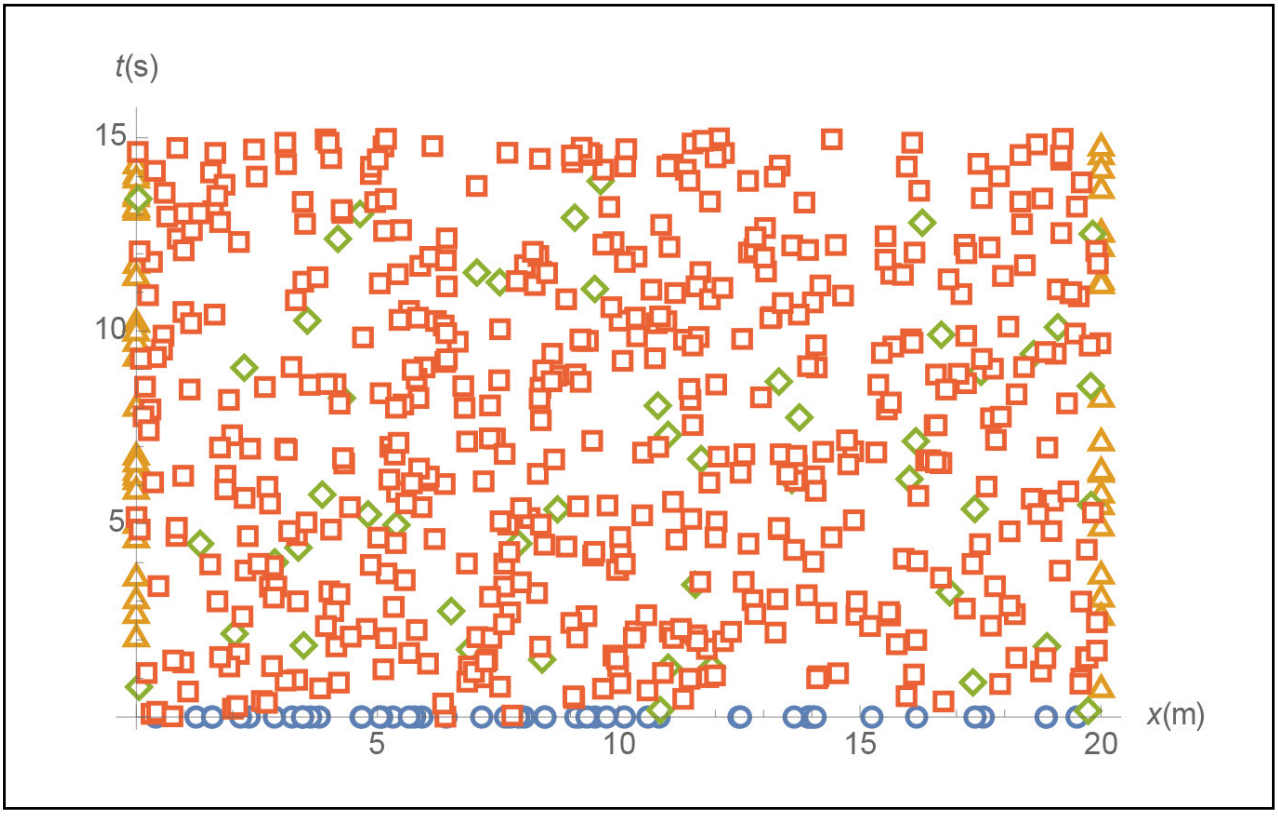
Second, the N_0 points from the initial condition, N_b points from the boundary conditions and the N_s concentration observational data from the sensors are incorporated in the loss term $L_d(\theta)$. This is another difference between both approaches, ANN and PINN, with respect to the observational data. Here we consider this observational data as a constrain for the numerical model. On the other hand, in the ANN case, this observational data is considered just after the optimization process. The measurements from sensors are correlated to a reaction parameter R_s, to be determined.

Third, in view of the defined new loss term $L_r(\theta, R)$ in Eq. (14), we need to compute the derivative of neural networks' output $c_{NN}(z, \theta)$ with respect to the input and model parameters. This is done trough the technique known as Automatic Differentiation. According (Baydin *et al.*, 2018), it "is a family of techniques similar to, but more general than, backpropagation for efficiently and accurately evaluating derivatives of numeric functions expressed as computer programs". This technique is well established in a Tensorflow routine.

Fourth, we sum up all the loss terms and proceed the optimization process in view of minimizing this cost function for the parameters θ and R. After this process,

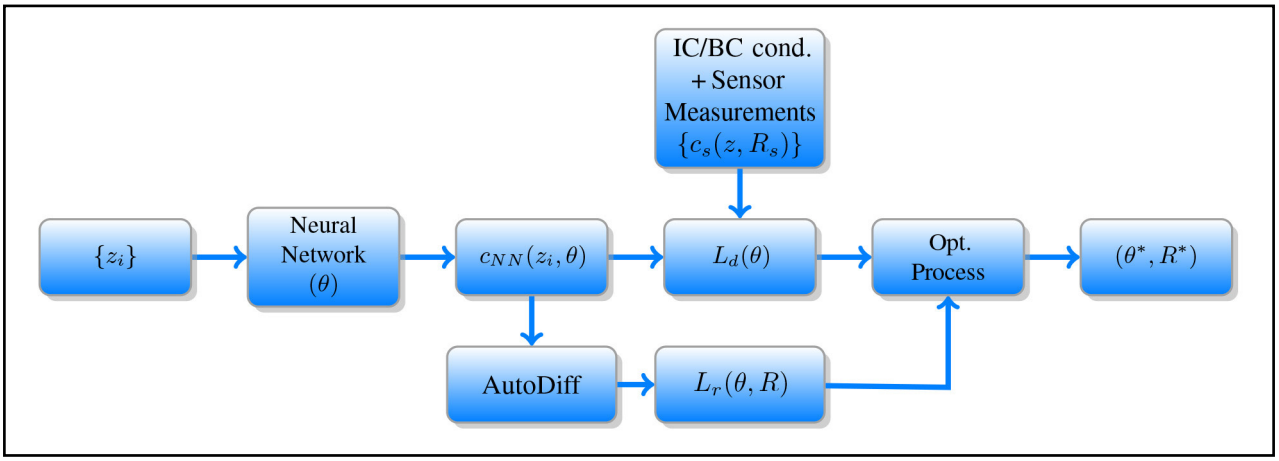
we find the arguments θ^* and R^*. The computed R^* is the prediction of the unknown reaction term R_s from the observational data obtained with sensors, that is, $R_{pred} = R^* \approx R_s$. Since we are dealing with random quantities, we realized the third and fourth steps thirty times, aiming to obtain more reliable results, and take the mean and standard deviation of the R_{pred} considering all runs, called $\overline{R_{pred}}$ and σ_{Rpred}, respectively. Numerical experiments are presented in Subsection 5.2.

Figure 4 – Example of points distribution in the PINN case. The blue circles represent the data from the initial condition, namely $\{z_i^0\}_{i=1}^{N_0}$.The orange triangles represent the data from the boundary conditions, namely $\{z_i^b\}_{i=1}^{N_b}$.The green diamonds represent the observational data from sensors whose values are correlated to reaction parameter,R_s, to be determined, namely $\{z_i^s\}_{i=1}^{N_s}$.The red squares represent the collocation points where we compute the residue of the neural network' output, namely $\{z_i^r\}_{i=1}^{N_r}$.



Source: Author's

Figure 5 – Flowchart for the Parameter Estimation using the PINN approach. The neural network is created with the list of concentration patterns as input, the list of R-values as target and the function R_{NN} as output. The loss function is created based on the output R_{NN} and the target R-values. After the training process, the optimal values of weights and biases, W^* and b^*, are obtained. The predicted value, R_{pred}, is computed through the neural network with sensor measurements as input and the already calculated optimal weights and biases as the parameters



Source: Author's

5 NUMERICAL EXPERIMENTS

In this section, we present the numerical experiments considered, which are related to the estimation of the reaction parameter of pollutant source in a river, posed by the homogeneous advection-dispersion-reaction equation.

5.1 Parameter Estimation with the ANN

In this subsection, we implement the reaction parameter estimation in the ANN context. In the following experiments, we consider the coefficients $D = 0.06\, m^2 s^{-1}$ and $V = 0.7\, ms^{-1}$ in the PDE given by Eq. (1) as well as the length of the river $L = 20\, m$ and total time considered $T = 15\, s$. Besides, we consider an initial pollutant concentration in the river given by Eq. (2), but null pollutant concentration on the boundary. We also

consider 70% of the full dataset as a training set and the remaining 30%, as the test set.

The experiments for ANN were implemented in the *Mathematica*' software, version 12.2.0.0, in a macOS 10.13.6, with 8GB RAM, without a GPU.

In the first experiment, we consider that the synthetic experimental data provided by sensors are correlated to the reaction parameter $R_s = 0.3\ s^{-1}$, in which we are interested in estimate. The N_{xs} sensor locations are sampled from a Uniform Distribution in the interval given by the length of the river $L = 20\,m$, where the number of measurements in time by each sensor, N_{ts}, is also sampled from a Uniform Distribution in the time interval $T = 15\,s$. With the aim to provide more reliable results, we generate 50 samples of observable data from sensors with the same $R_s = 0.3\ s^{-1}$ and take the mean value of the results.

In view of generating a training/test set, we consider $N_R = 50$ reaction parameters, R_j, sampled from a Uniform Distribution on the interval $[0, 1\ s^{-1}]$, $N_x = 50$ and $N_t = 50$ equally spaced points over the entire domain $[0, L] \times [0, T]$.

Following the ideas from Section 3, we create the dataset, see Eq. (3), for this experiment and the ANN with input of dimension $(N_x).(N_t)$ and output of dimension 1 and activation function of hyperbolic tangent type.

In Table 1, we fixed 4 hidden layers and varied the number of neurons, the number of epochs fixed in 2000 and batch size of 35. We are considering no noise in this measurements. We observe that the network with 40 neurons in each hidden layer provided the best estimation in this experiment.

Table 1 – Estimation of the Reaction Parameter R with the ANN - 4 Layers and variable number of neurons - Noiseless data - $N_s = 50$, $N_o = N_b = 50$

Layers	Neurons	Exact Parameter	Predicted Parameter	Standard Deviation	Rel. Error
4	20	0.3	0.289212	0.0298367	3.60 %
4	30	0.3	0.287926	0.0524926	4.02 %
4	40	0.3	0.307266	0.0530582	2.42 %
4	50	0.3	0.287413	0.0466505	4.20 %

Source: Author's

In the second experiment for ANN, we consider that the neural network with 4 layers and 40 neurons and that the observational data from sensors can be noisy. In other words, if $c_s(z_s, R_s)$ is the concentration measurement in spatial and time position $z^s = (x^s, t^s)$, correlated to the reaction parameter R_s (to be estimated), then, in this experiment, we will consider the following new observational data from sensors:

$$c_s(z^s, R_s) + \epsilon, \quad \epsilon \sim N(0, \sigma^2), \tag{18}$$

where $N(0, \sigma^2)$ is the Normal distribution with zero mean and variance σ^2. Observe that the variance σ^2 can be seen as the level of uncertainties in the measurements. Again, we run 50 times in view of generating the noisy observational data and, after substituting this as an argument in the already trained neural network, we take the mean and standard deviation of the predicted reaction parameter results, R_{pred}. In Table 2 the results are presented, in which we observe that the larger the noise level the greater the relative error between R_{pred} and R_s.

Table 2 – Estimation of the Reaction Parameter R with the ANN - 4 Layers and 40 Neurons - Data with different levels of noise - $N_s = 50$, $N_o = N_b = 50$

Noise	Exact Parameter	Predicted Parameter	Standard Deviation	Rel. Error
0 %	0.3	0.307266	0.0530582	2.42 %
0.1 %	0.3	0.289112	0.0483208	3.63 %
0.5 %	0.3	0.285646	0.0533316	4.78 %
1 %	0.3	0.284946	0.0525375	5.02 %

Source: Author's

5.2 Parameter Estimation with the PINN

In this subsection, we implement the reaction parameter estimation with PINN. We consider the same PDE parameters, domain and initial and boundary conditions. The PINN was implemented in the Python language using the well known open-source library for machine learning, TensorFlow (Abadi *et al.*, 2015). However, in view of limited CPU, we choose to use the Google Colab, which is a way of writing and executing Python

codes directly in the internet browser, even without any Python software installed in the computer, and with access to GPUs free of charges in a free account[3].

In the first example of this subsection, we consider a few measurements from sensors, that is, $N_s = 50$. It is important to observe the difference between the two approaches considered in this work, i.e. ANN and PINN. Here, in the PINN approach, we incorporate the observational data from sensors in the training process, while in the ANN approach, we use these measurements just after the training process. In Tables 3 and 4, we consider only the network with 4 layers in view of the observation that in the ANN approach this generated better results.

Table 3 – Estimation of the Reaction Parameter with the PINN - 4 Layers and variable number of neurons - Noiseless data - $N_s = 50$, $N_o = N_b = 50$, $N_r = 1000$

Layers	Neurons	Exact Parameter	Predicted Parameter	Standard Deviation	Rel. Error
4	20	0.3	0.25128	0.14861	16.24 %
4	30	0.3	0.33838	0.029058	12.79 %
4	40	0.3	0.29732	0.045473	0.89 %
4	50	0.3	0.18453	0.032688	38.49 %

Source: Author's

In Table 3, we consider the parameter estimation generated by different number of neurons, besides the activation function as the Leaky RELu function, with slope $\alpha = 0.1$, $N_x = 50$ and $N_t = 50$ equally spaced points in the entire domain $[0, L] \times [0, T]$, and $N_r = 1000$ collocation points sampled from the domain. In all experiments, we use ADAM optimizer (Kingma and Ba, 2015), with learning rate varying accordingly of number of epochs in the following way

$$\delta(n) = 0.1(1 - H(n - 2000)) + 0.01(H(n - 2000) - H(n - 4000)) + 0.001H(n - 4000), \quad (19)$$

where $H(n) = 0$, for $n < 0$, and $H(n) = 1$, for $n \geq 0$, is the Heaviside or step function. The number of epochs is fixed in 5000 and the initial guess of $R^{(0)} = 0.5\, s^{-1}$, the median value

[3]https://colab.research.google.com/

of the range considered for the reaction parameter $[0, 1\ s^{-1}]$. We observe that the best estimation was obtained considering 40 neurons. Therefore, in the next experiments, we will consider only networks of 4 layers with 40 neurons.

In Table 4, we now consider noisy measurements. The noise, σ, is introduced here in the same way as in the ANN case, that is, $c_s(z^s, R_s) + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$.

Table 4 – Estimation of the Reaction Parameter with the PINN - 4 Layers and 40 neurons - Data with different levels of noise - $N_s = 50$, $N_o = N_b = 50$, $N_r = 1000$

Noise	Exact Parameter	Predicted Parameter	Standard Deviation	Rel. Error
0.0 %	0.3	0.29732	0.045473	0.89 %
0.1 %	0.3	0.30550	0.056082	1.83 %
0.5 %	0.3	0.30988	0.031200	3.29 %
1 %	0.3	0.26628	0.049348	11.24 %

Source: Author's

We note that the PINN generated results better than the ones obtained with the ANN for small values of noise. When we consider 1% of noise, the parameter reconstruction with PINN seems more sensible than the one with the ANN. As a future work, a possible strategy is to manipulate the number of parameters to optimize or even to weight each one of the loss terms in Eq. (15), in which (Rojas; Bittencourt; Boldrini, 2021) include these loss weights as another parameter in the training process.

Table 5 – Estimation of the Reaction Parameter with the PINN - 4 Layers and 40 neurons - Data with different levels of noise - $N_s = 500$, $N_o = N_b = 100$, $N_r = 5000$

Noise	Exact Parameter	Predicted Parameter	Standard Deviation	Rel. Error
0.0 %	0.3	0.31087	0.18086	3.62%
0.1 %	0.3	0.32993	0.94682	9.98%
0.5 %	0.3	0.33602	0.17518	12.01%
1 %	0.3	0.37351	0.12039	24.50%

Source: Author's

In the second experiment, we increase the size of the sets to verify if we obtain better results in the estimation. We consider $N_s = 500, N_o = N_b = 100, N_r = 5000$. The results are shown in Table 5, However, we see that there is no substantial improvement with the increasing of the number of measurements, where it can result from the effect of overfitting. In fact, the results decrease the accuracy.

6 CONCLUSIONS

In this work, we study the reaction parameter estimation in the advection-dispersion-reaction equation modelling the pollutant concentration in a river. This inverse problem was solved by using Artificial Neural Networks and Physics-Informed Neural Networks.

The direct problem was solved by classical numerical methods with the goal of generating a dataset to be used in the neural network approach for the inverse problem. This dataset was split into two sets, a training set and a test set, where the first set will be used to train the network for the inverse problem.

The inverse problem was solved by two kinds of neural network. The first type considered, the ANN, was capable of recovering the reaction parameter from multiple sensors' measurements even in the presence of noise. The second one, with the PINN, despite also considering the noisy measurements, it proved to reconstruct the parameter with a lower relative error than in the ANN case, despite of being possibly more sensible to higher noise levels.

ACKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, National Council for Scientific and Technological Development - CNPq and FAPERJ - Carlos Chagas Filho Foundation for the Research Support of the State of Rio de Janeiro. H.S.M. also

acknowledge the Rio de Janeiro State University (UERJ), for the PAPD Program and FAPERJ - Carlos Chagas Filho Foundation for the Research Support of the State of Rio de Janeiro for the Emeritus Visiting Researcher grant.

REFERENCES

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; ... Zheng, X ; (2015). *Tensorflow: Large-scale machine learning on heterogeneous systems*. Software available from tensorflow.org.

Baydin, A. G.; Pearlmutter, B. A.; Radul, A. A.; Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, *18*, 1–43.

Chollet, F. (2018). *Deep Learning with Python*, 1st edn. Manning Publications, Shelter Island, NY.

Colton, D.; Kress, R. (2013). *Inverse Acoustic and Electromagnetic Scattering Theory*, 3rd edn. Springer, New York.

EVANS, L. C. (1998). *Partial Differential Equations*, 1st edn. American Mathematical Society.

Goodfellow, I.; Bengio, Y.; Courville, A. (2016). *Deep Learning*, 1st edn. MIT Press, Cambridge, MA.

Hadamard, J. (1902). Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton University Bulletin*, 13, pp. 49–52.

Isakov, V. (1990). *Inverse Source Problem*, 1st edn. American Mathematical Society, Providence, Rhode Island.

Kingma, D. P.; Ba, J. L. (2015). Adam: A method for stochastic optimization. In: *Proceedings of the International Conference on Learning Representations - ICLR 2015*, San Diego, California.

Lecun, Y.; Bengio, Y.; Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444.

Li, Z.; Liu, F.; Yang, W.; Peng, S.; Zhou, J. (2022). A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(12), 6999–7019.

Maas, A. L.; Hannun, A. Y.; Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In: *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, Georgia, USA.

Moura Neto, F. D.; Silva Neto, A. J. (2013). *An Introduction to Inverse Problems with Applications*, 1st edn. Springer, NIELSEN, M. A. (2015). *Neural Networks and Deep Learning*, 1st edn. Determination Press. New York.

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.

Rojas, C. J.; Bittencourt, M. L.; Boldrini, J. L. (2021). *Parameter identification for a damage model using a physics informed neural network*. *arXiv:210708781v1*, pp. 1–31.

Authorship contribution

1 – Roberto Mamud Guedes da Silva

Bachelor and Master in Mathematics, Doctor of Science in Nuclear Engineering and professor at Polytechnic Institute
https://orcid.org/0000-0002-9060-2640 • rmamud@macae.ufrj.br
Contribution: Writing – original draft and Writing – review & editing

2 – Helio dos Santos Migon

M.Sc. and doctorate in statistics and emeritus professor
https://orcid.org/0000-0003-4913-3210 • migon@im.ufrj.br
Contribution: Writing, Review & editing, Visualization, project administration, supervision

3 – Antônio José da Silva Neto

Ph.D. in Mechanical Engineering, M.Sc. in Nuclear Engineering and Professor in the Department of Mechanical Engineering and Energy
https://orcid.org/0000-0002-9616-6093• ajsneto@iprj.uerj.br
Contribution: Writing, Review & editing, Visualization, project administration, supervision

How to quote this article

Silva, R. M. G. da., Migon, H. dos. S., & Neto, A. J. da. S.(2023) Parameter estimation in the pollutant dispersion problem with Physics-Informed Neural Networks. *Ciência e Natura*, Santa Maria, *45(*spe. 3) e74615. DOI: https://doi.org/10.5902/217946074615. Available in: https://periodicos.ufsm.br/cienciaenatura/article/view/74615. Accessed in: day month abbr. year.