Machine Learning Project 2: Solving partial differential equations with neural networks

Carolina Alvarez, Gaston Cisterna, Sebastian Quevedo, Marti Rovira, Ania Tato

February 2024

Contents

I. Introduction

Partial differential equations (PDEs) play a crucial role in modeling various physical phenomena, ranging from heat conduction to fluid dynamics. Solving PDEs accurately and efficiently is essential for understanding and predicting the behavior of these systems.

In this project, we explore the solution of the diffusion equation in one dimension using both traditional finite difference schemes and neural networks. The physical system to study will be a rod of length L = 1, and the physical phenomenon being studied is the diffusion of heat along the rod, which obeys the partial differential equation:

$$K\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}, \quad t > 0, \quad x \in [0, L] \tag{1}$$

Where $u(x,t)$ represents the temperature gradient along the rod, which fulfills the diffusion equation, and evolves over time and space; L represents the length of the rod, and K is a positive coefficient called the thermal diffusivity of the medium. The initial temperature distribution is given by $u(x,0) = sin(\pi x)$, reflecting a sinusoidal variation along the rod. Two different boundary conditions will be considered: Dirichlet conditions (on $u(x_0,t)$ and $u(x_N,t)$), and Neumann conditions (on $\frac{du(x,t)}{dx}|_{x_0}$ and $\frac{du(x,t)}{dx}|_{x_N}$).

The first scenario can represent a rod whose ends are submerged in ice at 0 degrees, where the temperature at the boundaries is known. In this situation heat transfer within the rod is primarily influenced by the initial distribution of temperature along its length, represented by a sinusoidal shape. Over time, heat redistributes along the rod as it seeks to equalize the temperature throughout its length. In contrast, systems described by the heat equation with Neumann boundary conditions involve boundaries where the rate of heat flow or heat flux is specified. This is the case of a one-dimensional insulated rod with adiabatic ends. Physically, this implies that there is no net heat flow across the boundaries of the rod, and heat transfer within the rod occurs solely through internal processes, such as conduction. In this case, this kind of problem refers to a re-distribution of the energy within the material as an isolated system.

The problem focuses on understanding how heat propagates through the rod and how the temperature distribution evolves over time, given the specified boundary conditions.

Our goal is twofold: first, to implement a standard explicit finite difference scheme to solve the diffusion equation numerically, and second, to employ neural networks to tackle the same problem.

By comparing the results obtained from traditional numerical methods with those obtained from neural networks, we aim to evaluate the effectiveness and efficiency of both approaches in solving the diffusion equation. This project serves as an exploration into the intersection of computational physics and machine learning, showcasing the versatility and potential of these techniques in solving complex physical problems.

II. Explicit forward Euler algorithm

To solve the partial differential equation (PDE) describing the temperature gradient within the rod, we first implemented the explicit scheme algorithm known as the forward Euler method. This method is a simple yet effective approach for solving time-dependent problems by discretizing both the spatial and temporal domains.

i. Analytical solution

We start with the diffusion equation:

$$u_t = Ku_{xx}, \quad 0 \leq x \leq L, \quad t > 0 \tag{2}$$

with the initial condition:

$$u(x, 0) = \sin(\pi x), \quad 0 \leq x \leq L \tag{3}$$

and Dirichlet boundary conditions:

$$u(0, t) = u(L, t) = 0, \quad t > 0 \tag{4}$$

To solve this equation, we use separation of variables. Let $u(x, t) = X(x)T(t)$. Substituting this into the diffusion equation, we get:

$$\frac{T'}{KT} = \frac{X''}{X} = -\lambda \tag{5}$$

where λ is a separation constant.

Solving the time part of the equation, we get:

$$T' - \lambda KT = 0 \implies T(t) = e^{-\lambda Kt} \tag{6}$$

Now, solve the spatial part:

$$X'' + \lambda X = 0 \tag{7}$$

with boundary conditions $X(0) = X(L) = 0$.

The general solution to this ODE is:

$$X(x) = A\cos\left(\sqrt{\lambda}x\right) + B\sin\left(\sqrt{\lambda}x\right) \tag{8}$$

Applying the boundary conditions at $x = 0$, we find that $A = 0$. Now with $X(L) = 0$, we obtain:

$$B\sin\left(\sqrt{\lambda}L\right) = 0 \tag{9}$$

Excluding the trivial case of $B = 0$, we find:

$$\sin\left(\sqrt{\lambda}L\right) = 0 \tag{10}$$

Therefore, we find that $\lambda = n^2\pi^2/L^2$, where $n = 1, 2, 3, \ldots$. Thus, the eigenfunctions are:

$$X_n(x) = \sin\left(\frac{n\pi x}{L}\right) \tag{11}$$

Therefore, considering that L=1 the solution to the diffusion equation is given by:

$$u(x,t) = \sum_{n=1}^{\infty} B_n e^{-n^2\pi^2 Kt}\sin\left(n\pi x\right) \tag{12}$$

where the coefficients B_n can be determined using the initial condition $u(x,0) = \sin(\pi x)$.

Now, let's find the coefficients B_n using the initial condition:

$$u(x,0) = \sum_{n=1}^{\infty} B_n \sin\left(n\pi x\right) \tag{13}$$

$$= \sin(\pi x) \tag{14}$$

Comparing coefficients of sine functions on both sides, we see that $B_1 = 1$, and $B_n = 0$ for $n \geq 2$.

Therefore, the solution to the diffusion equation with the given initial and boundary conditions is:

$$u(x,t) = e^{-\pi^2 Kt}\sin(\pi x) \tag{15}$$

Taking a parallel approach, for the second scenario under consideration (Neumann), the boundary conditions would be:

$$\frac{du(x,t)}{dx}|_{x_0} = \frac{du(x,t)}{dx}|_{x_N} = 0 \quad t > 0 \tag{16}$$

with the initial condition:

$$u(x,0) = \cos(\pi x), \quad 0 \leq x \leq L \tag{17}$$

This results in the analytical solution:

$$u(x,t) = e^{-\pi^2 K t} \cos(\pi x) \tag{18}$$

ii. Explicit scheme

Numerically this problem can be solved using the explicit scheme, in which an iterative approach is used to obtain the solutions to the 1D diffusion equation starting from the boundary conditions. Considering that this is a numerical approach, there is a need to discretize the x and t variables. Step length can be introduces for the space variable x:

$$\Delta x = \frac{1}{n}$$

with n being the number of segments in which the space is divided into. Taking into account the stability condition for the explicit scheme $\frac{\Delta t}{\Delta x^2} = 1/2$, we have for t:

$$\Delta t = 0.5 \cdot \Delta x^2$$

Therefore, the discrete evolution of the variables follows:

$$t_j = j\Delta t, \quad j \geq 0$$
$$x_i = i\Delta x, \quad 0 \leq i \leq n$$

Now, using standard approximation expressions for the derivatives we obtain:

$$u_t \approx \frac{u(x_i, t_j + \Delta t) - u(x_i, t_j)}{\Delta t} \approx \frac{u_{i,j+1} - u_{i,j}}{\Delta t} \tag{19}$$

$$u_{xx} \approx \frac{u(x_i+, t_j) - 2u(x_i, t_j) + u(x_i-, t_j)}{\Delta x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} \tag{20}$$

Therefore, defining $\alpha = \frac{\Delta t}{\Delta x^2}$, we can write the PED in its discrete form:

$$K \cdot \frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} \tag{21}$$

With some algebraic steps it is possible to arrive at the explicit scheme expression:

$$u_{i,j+1} = K\alpha u_{i-1,j} + (1 - 2K\alpha)u_{i,j} + K\alpha u_{i+1,j} \tag{22}$$

Now the Dirichlet boundary conditions for the discrete system are:

$$u_{i,0} = \sin(\pi x_i)$$
$$u_{0,t} = u_{n,t} = 0$$

From the spatial boundary condition it is possible to obtain the solutions for the hole length at $t = 0$, and using equation 22 and the temporal boundary conditions it is possible to obtain all the subsequent solutions only using the information from the previous step. This same approach can be used with Neumman boundary conditions.

iii. Numerical setup

We conducted tests of the solution for two different spatial resolutions, $\Delta x = \frac{1}{10}$, and $\Delta x = \frac{1}{100}$, to observe how the accuracy of the solution varies with grid spacing. Additionally, we ensured that the time step satisfied the stability criterion of the explicit scheme, which demands that $\Delta t/(\Delta x)^2 \leq \frac{1}{2}$. We examined the solutions at two distinct time points, denoted as t_1 and t_2 , to capture different stages of the temperature evolution within the rod. At t_1 the temperature gradient distribution $u(x, t_1)$ exhibits significant curvature while remaining smooth, representing an intermediate stage of the transient process. Conversely, $u(x, t_2)$ is nearly linear and approaches the stationary state, indicating that the system has reached thermal equilibrium.

III. Neural Network

The Universal Approximation Theorem asserts that a neural network, consisting of a single hidden layer alongside input and output layers, can approximate any function to any desired degree of precision. In our case, the function to recover with the neural network is $u(x, t)$, whose analytical form has been obtained in the previous section.

i. Network design

To construct the required tools in order to build the network, we must first build a trial solution $g_t(x, t, P)$, depending on the 2 variables from the PDE, and which also incorporates the neural network parameters.

$$g_t(x,t,P) = h_1(x,t) + h_2((x,t), N(x,t,P)) \tag{23}$$

$h_1(x)$ is a function that depends solely on the PDE variables, $N(x,P)$ is a neural network with weights and biases described by P, and $h_2(x, N(x,P))$ is a function of the neural network as well as the PDE variables. This trial function must fulfill the boundary conditions, and this will be done by making $h_1(x)$ alone satisfy these conditions, while making $h_2(x) = 0$ at the boundaries, so that the output of the neural network doesn't play a role at these points. Additionally, the initial condition must be fulfilled by $g_t(x,t,P)$

For the case of Dirichlet boundary conditions 4, the trial function used was:

$$g_t(x,t,P) = (1-t)\cdot u(x,t) + x(1-x)\cdot t \cdot N(x,t,P) \tag{24}$$

Alternatively, when incorporating Neumann conditions, adjustments were made to the $h_2(x, N(x,t,P))$ term in the trial function in order to maintain fulfillment of the boundary conditions, resulting in:

$$g_t(x,t,P) = u(x,t) + t \cdot N(x,t,P) \tag{25}$$

The machine learning algorithm will consist on finding the set of weights and biases P in the neural network such that the trial solution (24 or 25) satisfies the PDE (1).

i. The cost function

The next step consists on defining the cost function, which quantifies the extent to which the neural network output satisfies the PDE. The cost function will be calculated at each of the x_i and t_i coordinates resulting from the discretization in these variables. For Dirichlet boundary conditions, the cost function will be defined by:

$$Cost(P,x,t) = \frac{\sum_i^{N_x}\sum_j^{N_t}\left(k\frac{\partial^2 g_t(x_i,t_j,P)}{\partial x^2} - \frac{\partial g_t(x_i,t_j,P)}{\partial t}\right)^2}{N_x \cdot N_t} \tag{26}$$

Where the derivatives can easily be calculated using the jacobian and hessian functions included in the autograd python library.

On the other hand, in order to study the Neumann boundary conditions, some changes must be made in 26 for optimal performance of the network, and the cost function will now be defined by:

$$\begin{aligned} Cost_{Inner} &= \frac{\sum_i^{N_x}\sum_j^{N_t}\left(k\frac{\partial^2 g_t(x_i,t_j,P)}{\partial x^2} - \frac{\partial g_t(x_i,t_j,P)}{\partial t}\right)^2}{N_{INNER}} \\ Cost_{BC} &= \frac{\sum_{x=x_0,x_N}\sum_j^{N_t}\left(\frac{\partial g_t(x,t_j,P)}{\partial x}\right)^2}{N_{BC}} \\ Cost(P,x,t) &= \gamma_{Inner}\cdot Cost_{Inner} + \gamma_{BC}\cdot Cost_{BC} \end{aligned} \tag{27}$$

Where N_{INNER} and N_{BC} stand for the inner points and the boundary condition points. In this instance, the error within the cost function originates from two sources. The first source concerns the alignment of the trial function with the equation, while the second is associated with the gradient of the trial function itself at the boundaries. The weights assigned to each of these components will be adjusted by the parameters γ_{Inner} and γ_{BC}, which we adjusted as $\gamma_{Inner} = 0.02$, $\gamma_{BC} = 0.98$, as proposed in [1].

The aim in both of the described scenarios is to minimize the cost function concerning the weights and biases of the network, represented by P. This study will utilize and contrast three different methods for the optimization process: Gradient Descent (GD), Gradient Descent with Momentum (GDM), and ADAM. Detailed explanations of these algorithms are provided in the subsequent sections.

ii. Gradient Descent

The concept behind the gradient descent algorithm is to iteratively adjust the parameters P bringing the cost function to a minimum. The update of the parameters is defined by:

$$\begin{aligned} g_i &= \nabla_P Cost(P_i, x_i, t_i) \\ P_{i+1} &= P_i - \lambda \cdot g_i \end{aligned} \tag{28}$$

The gradient is taken with respect to the parameters P_i (sets of weights and biases of the network corresponding to the iteration i), and this operation is repeated for a chosen number of iterations. The parameter λ is known as the learning rate, and it determines the size of the steps taken in the direction of the gradient trough the minimization. For the Gradient descent method, the learning rate is kept constant; and only local minima of the cost function will be reached. The convergence of the optimization methods as a function of the learning rate will be studied in this project.

iii. Gradient Descent with Momentum

The Gradient Descent with Momentum method bears resemblance to the previously discussed approach but introduces an additional variable to the update of weights and biases, known as momentum m:

$$\begin{aligned} g_i &= \nabla_P Cost(P_i, x_i, t_i) \\ m_{i+1} &= \beta \cdot m_i + (1 - \beta) \cdot g_i \\ P_{i+1} &= P_i - \lambda \cdot m_{i+1} \end{aligned} \tag{29}$$

The momentum can be interpreted as the algorithm having memory about previous steps, and the update will be done on both the parameters of the neural network and the momentum for each iteration. As a result, the learning rate, also known as the step size, varies, and an extra coefficient, denoted as β, is introduced in the update.

iv. ADAM Method

The final optimization method evaluated is called Adaptive Moment Estimation, or ADAM. This technique calculates averages of both the first and second moments of the gradient (m and v respectively) and utilizes this data to dynamically adjust the learning rate for individual parameters.

$$\begin{aligned} g_i &= \nabla_P Cost(P_i, x_i, t_i) \\ m_{i+1} &= \beta_1 \cdot m_i + (1-\beta_1) \cdot g_i \\ v_{i+1} &= \beta_2 \cdot m_i + (1-\beta_2) \cdot g_i^2 \\ \hat{m}_{i+1} &= \frac{m_{i+1}}{1-\beta_1^t} \\ \hat{v}_{i+1} &= \frac{v_{i+1}}{1-\beta_2^t} \\ P_{i+1} &= P_i - \frac{\lambda \cdot \hat{m}_{i+1}}{\sqrt{\hat{v}_{i+1}} + \epsilon} \end{aligned} \tag{30}$$

β_1 and β_2 set the memory lifetime of the first and second moment and are typically taken to be 0.9 and 0.99 respectively; and $\epsilon \sim 10^{-8}$ is a small regularization constant to prevent divergences. In this case, the learning rate is reduced in directions where the norm of the gradient is consistently large, which greatly speeds up the convergence by allowing to use a larger learning rate for flat directions.

ii. Network parameters

The remaining ingredients in setting up the network are its activation function, hidden layer architecture, and number of iterations to be performed by the network. The activation function used is the sigmoid, which gives an output between 0 and 1:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{31}$$

On the other hand, the neural network to be employed in this work consists of 3 layers, the input layer having 100 nodes, one hidden layer with 25 nodes, and the output layer, consisting on 1 node, which gives the output calculated function for a given (x,t).

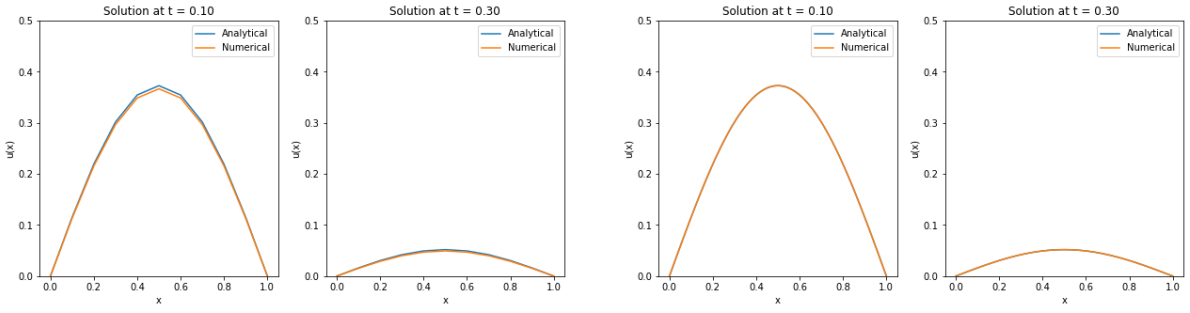
Regarding the variable discretization, the step size to be taken in both the time and position coordinates will be $\Delta x = \Delta t = 0.01$.

IV. Results

i. Forward Euler

i. Dirichlet boundaries

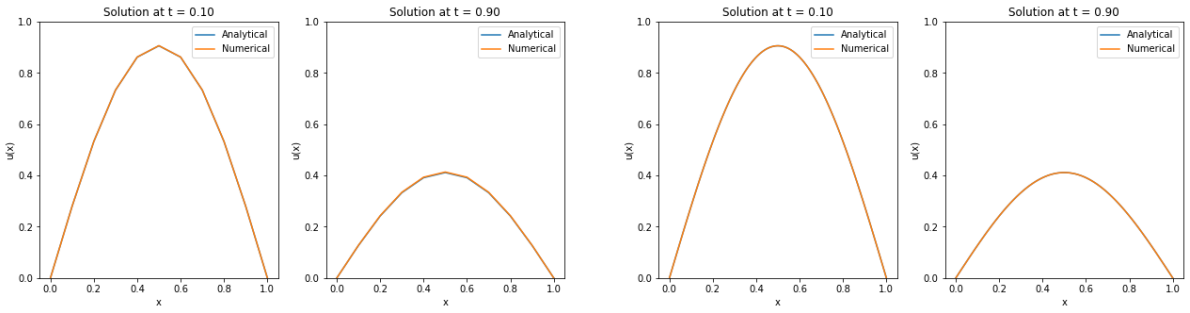
Below it is possible to see the solution obtained with the explicit scheme applying Dirichlet boundary conditions. It was observed that the speed at which the solutions go to zero is mediated by the K variable, therefore results for K=1 and K=0.1 are provided in order to aid in comparing this variable's effect. Staring with K=1 it can be seen from Figure 1 that the numerical solutions are in considerable agreement with the analytical solutions. As expected, as Δx becomes smaller, the solution is smoother an more accurate compared to the analytical function.



(a) Solutions with $\Delta x = 0.1$ (b) Solutions with $\Delta x = 0.01$

Figure 1: Obtained solutions for K = 1

Now, considering the solution obtained with K=0.1, these are presented in Figure 2, which shows the same improvement in accuracy as in the case of K=1. The solutions improve as the step length in the spatial (and time) coordinate become smaller.

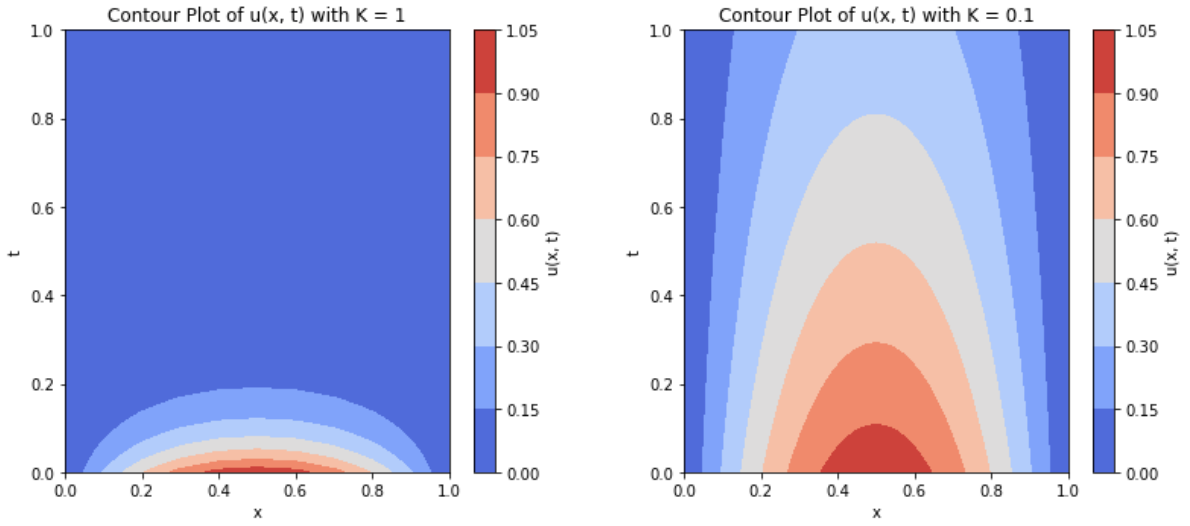


(a) Solutions with $\Delta x = 0.1$ (b) Solutions with $\Delta x = 0.01$

Figure 2: Obtained solutions for K = 0.1

It is important to notice when comparing Figures 1 and 2 that the upper limit of the

y-axis is different, as the magnitude of the solution changes considerably with K. This can be more clearly seen from the contour plots of time evolution in Figure 3.For K=1 the solutions rapidly go to zero meaning that the diffusion occurs quickly and the heat dissipates along the rod efficiently. On the other hand, when K=0.1 the solutions at the final time show a clear decrease, but not as prominent as in the previous case. It can be seen that at the final time the middle region has a higher value compared to the extremes.



(a) Solutions with K=1

(b) Solutions with K=0.1

Figure 3: Contour plot showing the time evolution of the PDEs solution

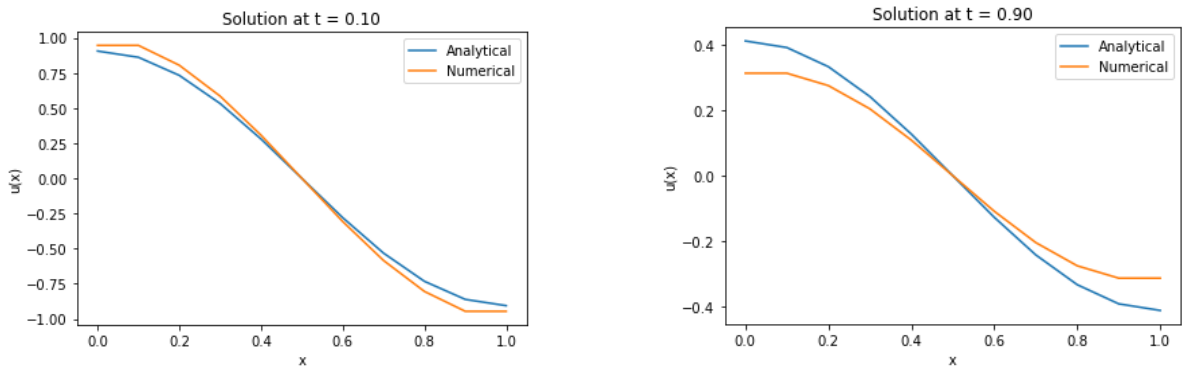
From the previous results it is clear that the solutions become more accurate as the number of points in the spatial coordinate increase, and that the constant K mediates the rate of diffusion as time evolves.

ii. Neumann boundaries

For the second case of study, with to boundary conditions on the derivative, the obtained results for both of the considered spatial resolutions can be seen in Figures 4 and 5.

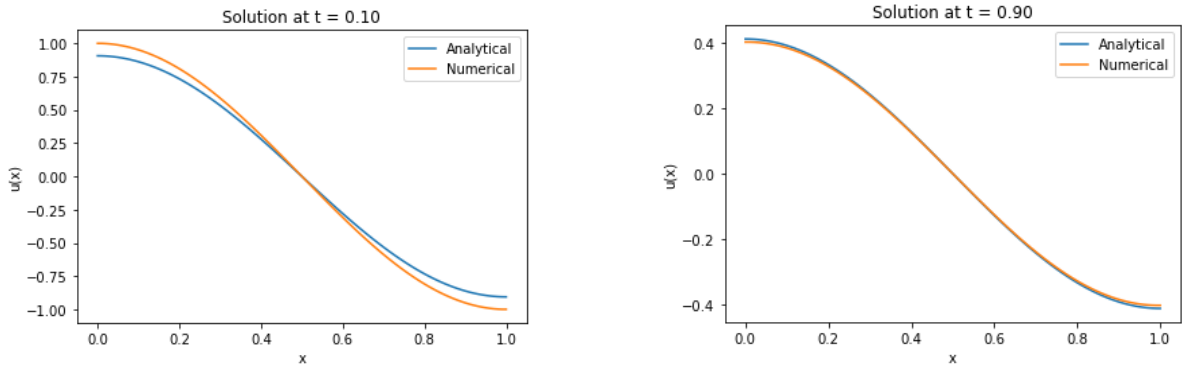
The numerical solution achieves greater accuracy with a smaller spatial resolution of $\Delta x = 0.01$, as expected. This enhancement can be attributed to the increased amount of points near the boundaries, where the discrepancy between the analytical and numerical solutions reaches its maximum, as can be clearly observed in Figure 4b. In the case of $\Delta x = 0.1$, this difference increases with each time step, increasing the separation between the numerical and analytical solutions, as opposed to $\Delta x = 0.01$, where the difference progressively decreases.

On the other hand, Figure 6 depicts the contour plots corresponding to both spatial reso-



(a) (b)

Figure 4: Obtained solutions for a spatial resolution $\Delta x = 0.1$



(a) (b)

Figure 5: Obtained solutions for a spatial resolution $\Delta x = 0.01$

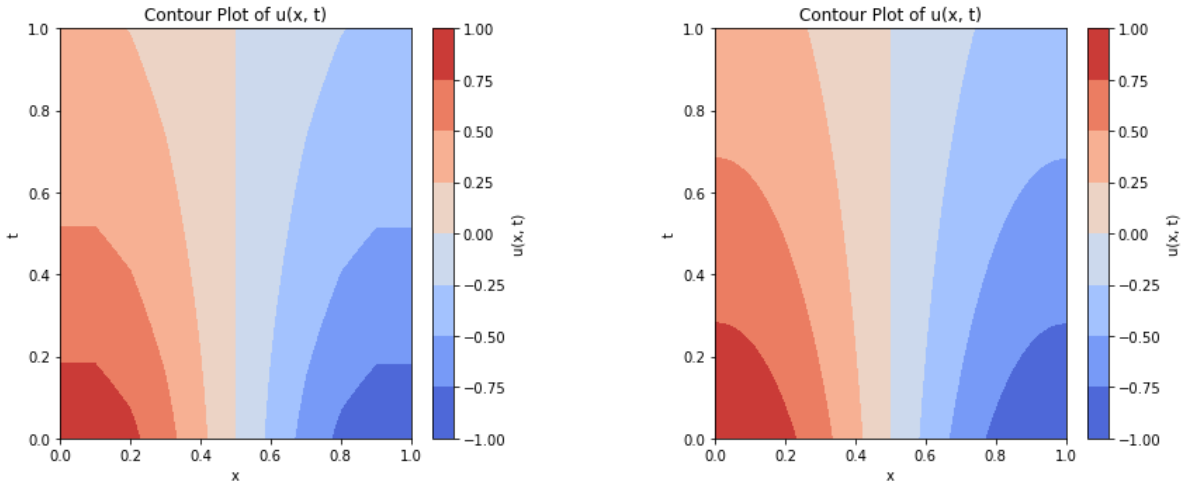
lutions. We can observe an overall decrease in the temperature as time progresses, which is uniform throughout the whole rod, and now affects the borders as well. Additionally, the profile shape remains constant, since losses are uniform, and affect every position on the rod equally.

ii. Neural Network

i. Dirichlet boundaries

The examination of the performance of the NN will commence with its implementation utilizing various gradient descent methods, including Stochastic Gradient Descent, Gradient Descent with Momentum, and the ADAM method, as discussed in the previous section (refer to III). Each method will be evaluated using three learning rates, denoted as $\lambda = \{0.001, 0.01, 0.1\}$, to determine the optimal performance of the neural network.

The results for each method are presented in Table 1. It is observed that, after 250



(a) (b)

Figure 6: Contour plot

iterations, each method achieves different final values of the cost function, with the ADAM method showing superior performance compared to the others. All methods begin with the same initial cost function value of 0.30.

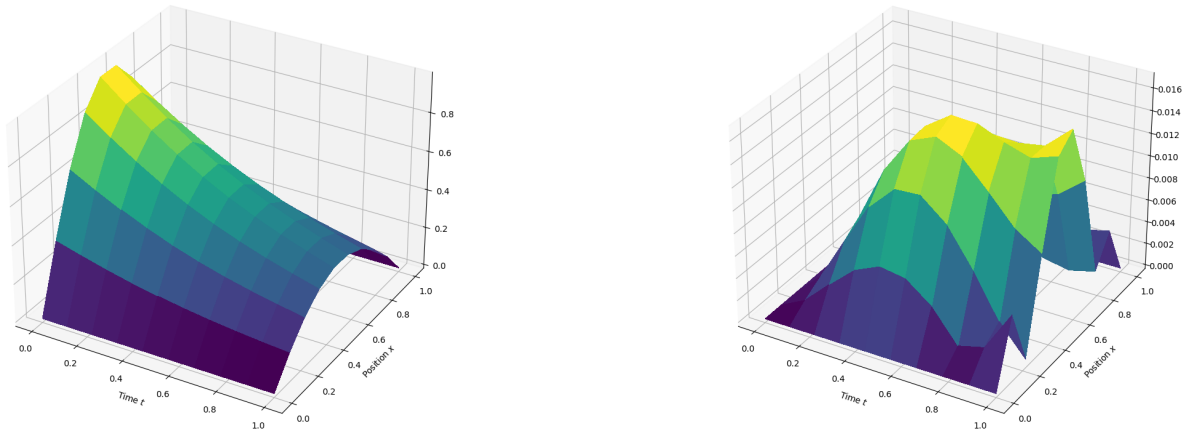
Method / λ	$\lambda = 0.001$	$\lambda = 0.01$	$\lambda = 0.1$
Stochastic Gradient Descent: Final Value Cost Function	0.086	0.022	0.0027
Gradient descent with Momentum: Final Value Cost Function	0.081	0.018	0.0022
ADAM: Final Value Cost Function	0.0004	0.0001	0.02

Table 1: Final values of the Cost Functions for each gradient descent method and learning rate λ. In the case of the G.D. with Momentum and ADAM, the learning rate represent the initial one with which iterations start. The initial value of the Cost function for all the cases is 0.3. The worst and the better learning rate are in red and green respectively, for each method.

Varying the learning rate has yielded diverse performance outcomes for the neural network, as measured by the final value of the cost function. The behaviors of the GD and GDM methods are comparable. In both instances, the most favorable outcome is attained with the largest learning rate. The ADAM method on the other hand gets better result with $\lambda = 0.01$.

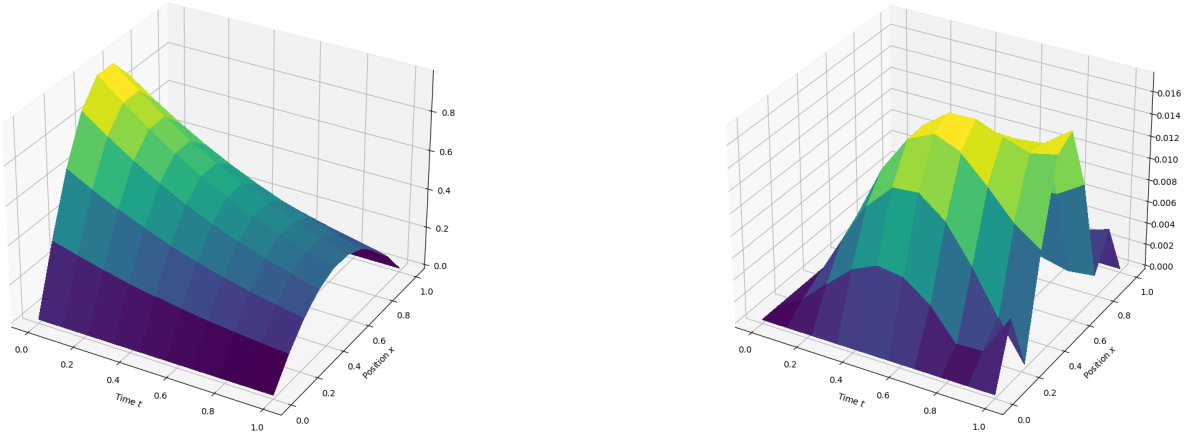
Then, the NN solutions to the partial differential equations for each GD method considered are plotted.

Figures 7 a),8 a) and 9 a) illustrate the NN solutions for the three GD methods discussed. In the three cases, the temperature profile diminishes as time progresses. The overall shape remains consistent, with the peak temperature located at the center of the uni



(a) (b)

Figure 7: Stochastic Gradient Method



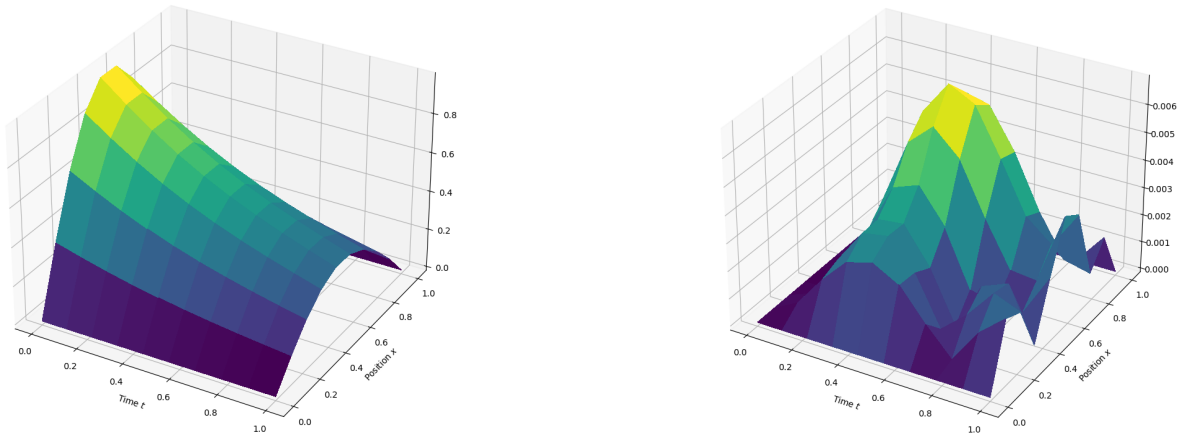
(a) (b)

Figure 8: Gradient Method with Momentum

dimensional geometry, as dictated by the initial sinusoidal condition. Furthermore, the temperature at the edges remains at 0 for all times as established by the boundary conditions. This behaviour correctly mimics the analytical solution.

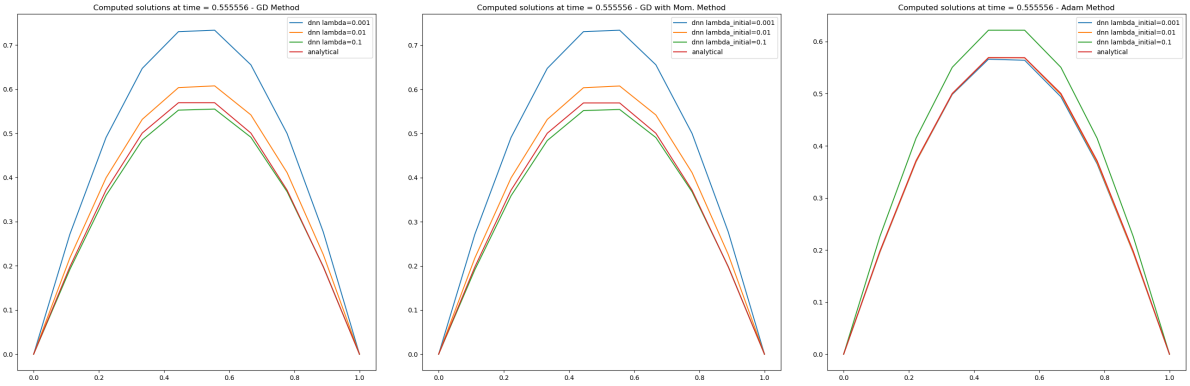
On the other hand, Figures 7 b), 8 b), and 9 b) illustrate the temperature difference between the analytical and computational results. To provide a thorough examination of the disparities between the analytical and neural network solutions, Figure 10 presents the solutions at the intermediate time points for each GD method and learning rate under study.

Lastly, in Figure 11, the solutions of the PDE using NN at intermediate time, for each GD method. Each one of them has been implemented with the best learning rate. It is evident that ADAM outperforms the others.



(a) (b)

Figure 9: ADAM Method



(a) (b) (c)

Figure 10: Numerical result obtained by the neural network for several values of λ and different methods.

ii. Neumann boundaries

When dealing with Neumann boundary conditions, the numerical result was obtained only for the ADAM method, with initial learning rate $\lambda = 0.01$, since it has proven to be the most accurate. In this case, unlike Diritchlet Boundary Conditions, penalization is implemented in the cost function itself when points belonging to the boundaries move away from the correct values (refer to Section III).

The system's behavior is to neutralize gradients of temperature due to diffusion and the isolation of the system, resulting in a uniform temperature distribution. The final temperature T_{final} can be already known without solving the PDE.

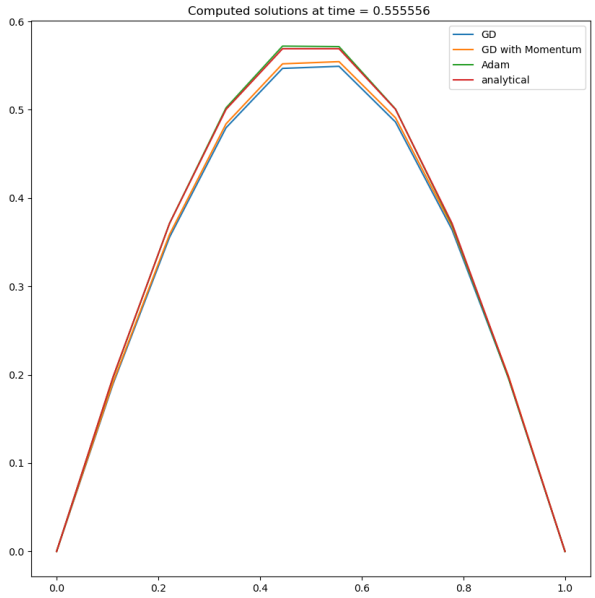


Figure 11: Comparison of solutions to PDE using NN for different gradient descent methods. ADAM demonstrates superior performance over other methods.

As the system is isolated, the total energy must be conserved.

$$E_{initial}^{Total} = \int_0^1 \rho\, Cp\ \left(U(x,0) - T_{ref}\right)\ dx \tag{32}$$

$$E_{Final}^{Total} = \rho\, Cp\ \left(T_{final} - T_{ref}\right) \tag{33}$$

$$E_{initial}^{Total} = E_{final}^{Total} \tag{34}$$

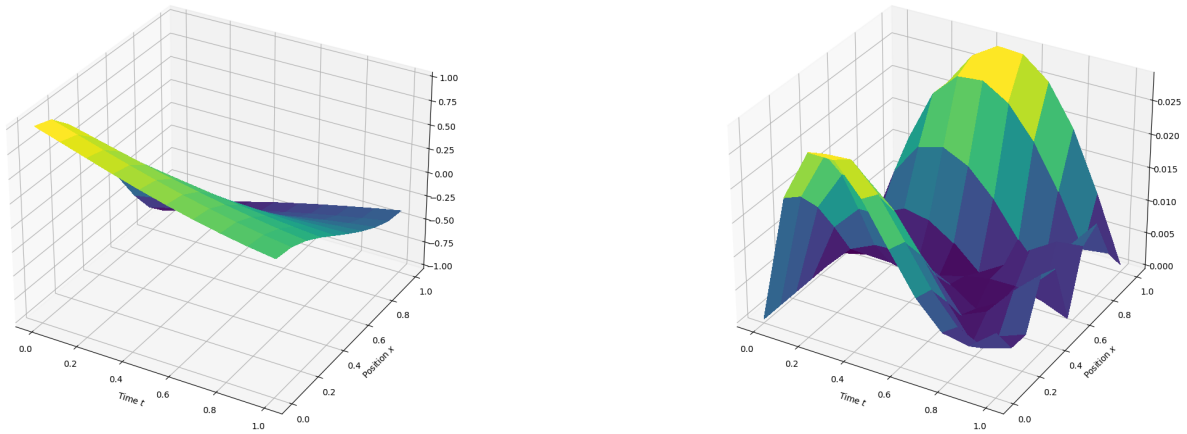
$$\int_0^1 \rho\, Cp\ \left(U(x,0) - T_{ref}\right)\ dx = \rho\, Cp\ \left(T_{final} - T_{ref}\right) \tag{35}$$

As $U(x,0) = cos(\pi\, x)$, the integral of eq. 35 gives 0. T_{ref} is the reference point for the temperature, normally chosen as 0. Thus, the final temperature T_{final} is equal to 0.

Consequently, we are looking for solutions which the temperature profile decreases over time, with the temperatures at the boundaries also diminishing as they are no longer fixed.

Figure 12 a) presents the NN solution of the PDE with the Neumann boundary conditions discussed. This solution meets the criteria mentioned. As time progresses, the temperature distribution tends to a final constant temperature of 0. On the other hand, the derivative

in the extremes goes to 0, as dictated by the boundary conditions. Figure 12 b) presents the difference between the analytical and NN solution. It can be observed that in the boundary conditions points there is greater error. This can be fixed by varying the parameters γ_1 and γ_2 (refer to III) in the cost function. The problem this solution might bring is worsening in the difference in the inner points. Nevertheless, there is a good and acceptable agreement between the solutions.



(a) (b)

Figure 12: ADAM method with Neumann BCs.

Lastly, in Figure 13, the solutions at intermediate time are depicted. The derivatives at the extremes are close to 0, and the temperature profile essentially resembles that at $t = 0$, albeit decreased.

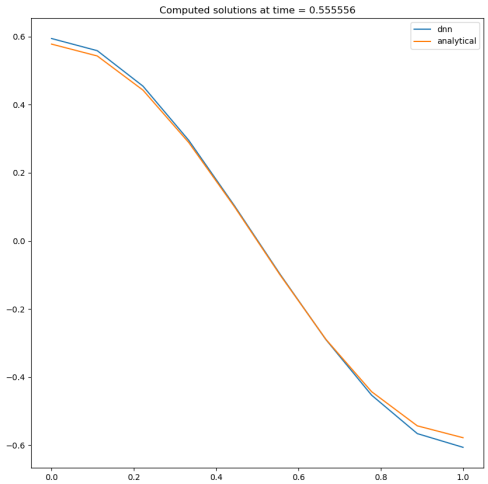


Figure 13: Comparison of solutions to PDE using NN for different gradient descent methods. ADAM demonstrates superior performance over other methods.

V. Conclusions

From the previous work it is apparent that partial differential equations can be accurately solved using numerical tools, such as the explicit approach and neural networks. Considering the former, an in depth study was performed which yielded accurate results for the solution of the diffusion equation considering two distinct values of the K constant. Moreover, Dirichlet and Neummann boundary conditions were explored and in both cases it was found that more accurate solutions could be obtained by decreasing the size of the steps in the spatial coordinate, albeit with a increase in the processing time. For the considered system, the increase in time was not substantial but this may not hold true for more complex systems. Additionally, it was shown the the constant K in the PDE mediates the speed at which diffusion occurs as time evolves.

Then, our study compared different methods for training neural networks (NN) to solve partial differential equations (PDEs) with varying boundary conditions (BCs). We found that the ADAM method consistently outperformed Stochastic Gradient Descent (SGD) and Gradient Descent with Momentum (GDM), minimizing the cost function, and thus, indicating its effectiveness in training the network. This superiority may be attributed to several features of ADAM. For instance, it utilizes adaptive learning rates, allowing for more efficient convergence in various parameter space directions compared to Stochastic Gradient Descent. Additionally, ADAM employs adaptive moments by computing both first and second moments of gradients, facilitating adaptive updates to parameters. Moreover, ADAM demonstrates less sensitivity to hyperparameter choices, such as learning rate and momentum coefficient, making it easier to tune and more robust in practice.

When dealing with Dirichlet BCs, we observed that adjusting the learning rate affected the network's performance. In spite of Gradient Descent with Momentum and ADAM methods change the learning rate from one iteration to the next one, initial values of it lead to different results. Higher rates worked better for SGD and DGM, while a moderate rate was optimal for ADAM.

For Neumann BCs, ADAM method was implemented due to the better performance with Dirichlet BCs. In this particular case, the penalization with regard to the BC points is implemented at the cost function level, unlike the Dirichlet BCs. Despite some discrepancies, the overall agreement between the neural network and analytical solutions was satisfactory.

References

[1] X. Li, J. Deng, J. Wu, S. Zhang, W. Li, Y.-G. Wang, Physical informed neural networks with soft and hard boundary constraints for solving advection-diffusion equations using fourier expansions (2023). arXiv:2306.12749.