SOLVING PARTIAL DIFFERENTIAL EQUATIONS USING PHYSICS INFORMED NEURAL NETWORKS

Raj Gopal Nannapaneni

Abstract

In this project, the usage of physics informed neural networks (PINNs) to solve partial differential equations (PDEs) is investigated. The solutions of PDEs from PINNs are compared with analytical solutions for 1D and 2D problems and solution from a computational software COMSOL.

1. Introduction

Transient problems like heat conduction and wave propagation have significant importance in the field of engineering. These problems are challenging to solve analytically and require numerical solutions. There are many numerical methods that engineers use to solve these problems which have been developed over the decades. With the advancement in computational resources and solvers made it possible to solve large scale and complex problems with relative ease. One such numerical method to solve the governing equations of heat conduction and wave propagation is based on the use of Neural networks.

Physics informed neural networks (PINN) have been developed recently to solve partial differential equations (PDEs) as an alternative to other numerical techniques like finite difference and finite element methods [1–3]. These methods come with an attractive feature of incorporating physical phenomenon into the model and be able to solve PDEs with high accuracy. To establish the accuracy of the solutions from PINNs, we solve two problems heat equation, wave equation and compare with the analytical solutions of the PDE. We will study the influence of number of neural networks, algorithms on the accuracy of the solution of the PDEs. We use NeuralPDE [4] in Julia [5], with modules like Flux[6], ModelingToolkit, DiffEqFlux.

Though the method of solving PDEs using NN has not been developed to the compete with the already existing methods, in the coming years with increase in computational resources it can be possible to solve multiple complex problems.

2. Methodology of solving PDEs'

Conventional methods for numerically solving PDEs require the PDE to be written in variational form and discretization of the problem. PINNs' have an attractive feature of solving the PDE directly in its strong form using *automatic differentiation*. The solution procedure is based on gradient based optimization of the loss function. For this reason the convergence of the solution is not always guaranteed. The optimization solver have major impact on the accuracy and rate of convergence of the solution. In this project two optimization solvers ADAM and BFGS were used to solve heat and wave equations respectively. Another important parameter that effects the solution is the number of iterations used. Keeping these two as the hyperparameters of we solve the two types of PDEs.

3. Results

Any numerical method solving a problem must be tested for its accuracy. In that spirit we use 1D and 2D benchmark problems to test the accuracy of solution obtained from PIINs' by comparing with analytical solution and solutions from commercial FEM software like COMSOL. For this purpose we choose two classes of time dependent PDEs' — ***heat equation, wave equation*** in 1D and 2D.

3.1. 1D Heat and Wave Equations

In this section we solve 1D heat and wave equations to verify the accuracy of solution of PINN vs the analytical solutions.

3.1.1. Problem: 1D Heat Equation

Consider a PDE as shown below:

$$\begin{aligned} &u_t = k u_{xx} \quad x \in [0, L], \quad t \in [0, T] \\ &u(x,0) = f(x) \quad u(0,t) = 0 \quad u(L,t) = 0 \\ &f(x) = 6 \sin\left(\frac{\pi x}{L}\right) \end{aligned} \tag{1}$$

here k is the thermal conductivity. The solution [7] of the equation is:

$$u(x,t) = 6 \sin\left(\frac{\pi x}{L}\right) e^{-k\left(\frac{\pi}{L}\right)^2 t} \tag{2}$$

For this problem, the length of the domain $(L = 1)$ and time of the simulation $(t = 1)$.

[Figure 1 about here.]

3.1.2. Problem: 1D Heat Equation

Consider a heat equation with in-homogeneous boundary conditions:

$$\begin{aligned} &u_t = u_{xx} \quad x \in [0, 2], \quad t \in [0, T] \\ &u(x,0) = f(x) \quad u(0,t) = 0 \quad u(2,t) = 8 \\ &f(x) = 2x^2 \end{aligned} \tag{3}$$

The solution of the equation is [8] :

$$u(x,t) = 4x - \sum_{k=1(odd)}^{N} \frac{64}{k^3 \pi^3} e^{-\frac{9}{4}\pi^2 k^2 t} \sin \frac{k\pi x}{2} \tag{4}$$

[Figure 2 about here.]

3.1.3. Problem: 1D Wave Equation

Consider a 1D wave equation as shown below:

$$\begin{aligned} &u_{tt} = c^2 u_{xx}, \quad x \in [0,1], \quad t \in [0,1] \\ &u(0,t) = u(1,t) = 0 \\ &u(x,0) = x(1-x) \\ &u_t(x,0) = 0 \end{aligned} \tag{5}$$

here c is the wave velocity. Analytical solution [9] of the above equation is:

$$u(t,x) = \sum_{n=1}^{N} \frac{8}{(n\pi)^3} \sin(n\pi x) \cos(\mathbf{c}\, n\pi t) \tag{6}$$

Solving the equation using PINN from Julia, we can see the results compared with the analytical solution in **Fig.** 3.

[Figure 3 about here.]

3.1.4. Problem: 1D Wave Equation

$$\begin{aligned} &u_{tt} = \mathbf{4} u_{xx}, \quad x \in [0,1], \quad t \in [0,1] \\ &u(0,t) = u(1,t) = 0 \\ &u(x,0) = \sin(\pi x) \\ &u_t(x,0) = 0 \end{aligned} \tag{7}$$

True solution of the above equation is:

$$u(t,x) = \sin(\pi x) \cos(2\pi t) \tag{8}$$

Comparison of analytical and NN solution is presented in **Fig.** 4.

[Figure 4 about here.]

3.2. 2D Heat and Wave Equations

For the purpose of numerical verification of the accuracy of PDEs in 2D we use method of manufactured solutions. The methodology is that a solution to a PDE which satisfies the PDE is assumed and the boundary conditions and initial conditions are constructed accordingly.

3.2.1. Problem: 2D Heat Equation

The heat equation in 2D is

$$u_t = k(u_{xx} + u_{yy}), \quad (x,y) \in [0,1], \quad t \in [0,1] \tag{9}$$

Let the solution of the PDE is

$$u(x,y,t) = e^{x+y} \cos(x + y + 4t) \tag{10}$$

For the given solution of the PDE the initial and boundary conditions are

$$\begin{aligned} &u(x,y,0) = e^{x+y}\cos(x+y) \\ &u(0,y,t) = e^{y}\cos(y+4t) \qquad u(1,y,t) = e^{1+y}\cos(1+y+4t) \\ &u(x,0,t) = e^{x}\cos(x+4t) \qquad u(x,1,t) = e^{x+1}\cos(x+1+4t) \end{aligned} \tag{11}$$

The comparison of analytical and PINN solution is **Fig. 5**.

[Figure 5 about here.]

3.2.2. Problem: 2D Wave Equation

A 2D wave equation with velocity c is

$$u_t = c^2(u_{xx} + u_{yy}), \quad (x,y) \in [0,1], \quad t \in [0,1] \tag{12}$$

Let the solution of the PDE is

$$u(x,y,t) = \sin(\pi x)\sin(\pi y)\cos(2\pi t) \tag{13}$$

For the given solution of the PDE the initial and boundary conditions are

$$\begin{aligned} &u(x,y,0) = \sin(\pi x)\sin(\pi y) \\ &u_t(x,y,0) = 0 \\ &u(0,y,t) = 0 \quad u(1,y,t) = 0 \\ &u(x,0,t) = 0 \quad u(x,1,t) = 0 \end{aligned} \tag{14}$$

[Figure 6 about here.]

3.3. Comparison with FEM solutions

In this section, we solve a transient heat equation on a 2D square domain. The initial temperatures of 10C and 1C are prescribed on the boundary of a square as shown in **Fig. 7**. The thermal conductivity of the square domain is assumed to be 1. The result from PINN shown in **Fig. 8** is compared with the solution from comsol (*attached comsolresult.gif file in the folder*).

[Figure 7 about here.]

[Figure 8 about here.]

4. Conclusions

Though using NN can be advantageous for solving PDEs it gets computationally more expensive for a larger domains and larger time intervals. There is a no standard procedure to solve a PDE to get accurate results since the NN has to be trained for every PDE. Solvers used for the optimization problem should be chosen carefully since this choice can impact the accuracy and execution time. In this project ADAM solver performed better for hear equation and BFGS gave better results for wave equation.

References

[1] Chengping Rao, Hao Sun, and Yang Liu. Physics informed deep learning for computational elastodynamics without labeled data. *arXiv preprint arXiv:2006.08472*, 2020.

[2] Lu Lu, Xuhui Meng, Zhiping Mao, and George E Karniadakis. Deepxde: A deep learning library for solving differential equations. *arXiv preprint arXiv:1907.04502*, 2019.

[3] Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A deep learning framework for solution and discovery in solid mechanics. *arXiv preprint arXiv:2003.02751*, 2020.

[4] Christopher Rackauckas and Qing Nie. Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia. *The Journal of Open Research Software*, 5 (1), 2017. doi: 10.5334/jors.151.

[5] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[6] Mike Innes. Flux: Elegant machine learning with julia. *Journal of Open Source Software*, 3(25): 602, 2018. doi: 10.21105/joss.00602.

[7] Paul Dawkins. *Solving The Heat Equation*, 2019. URL https://tutorial.math.lamar.edu/classes/de/solvingheatequation.aspx.

[8] Joel Feldman. *Solution of the Heat Equation by Separation of Variables*, 2007. URL https://www.math.ubc.ca/~feldman/m267/heatSln.pdf.

[9] Joel Feldman. *Solution of the Wave Equation by Separation of Variables*, 2007. URL https://www.math.ubc.ca/~feldman/m267/separation.pdf.

Figures

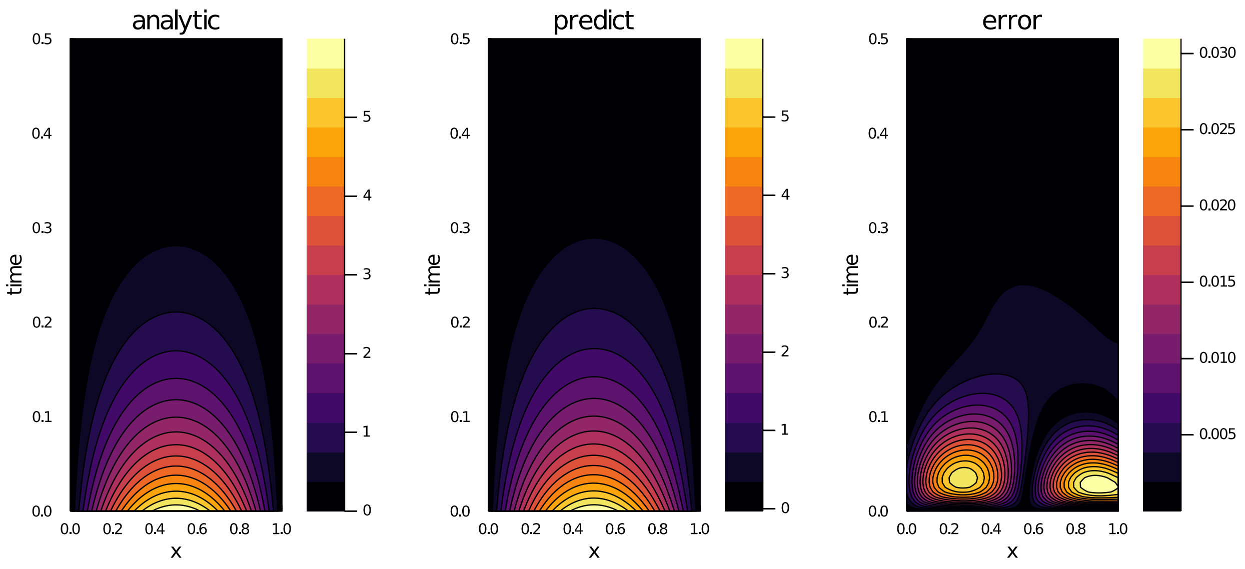

Figure 1: Neural network solution for 1D heat equation compared with the analytical solution using Julia.

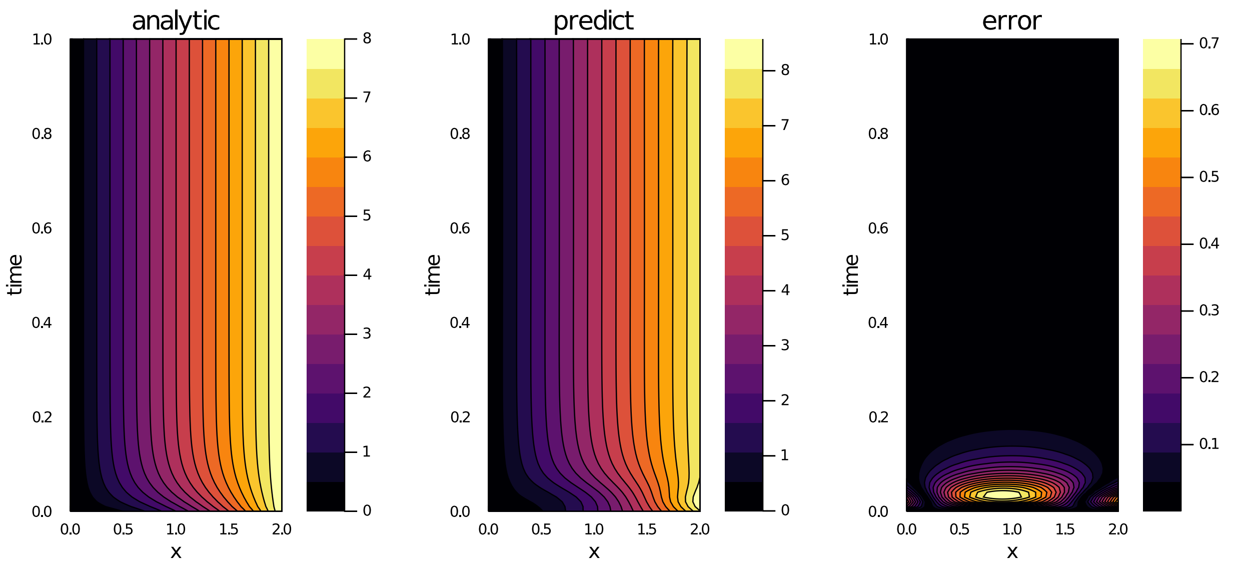

Figure 2: Neural network solution for 1D heat equation compared with the analytical solution using Julia.

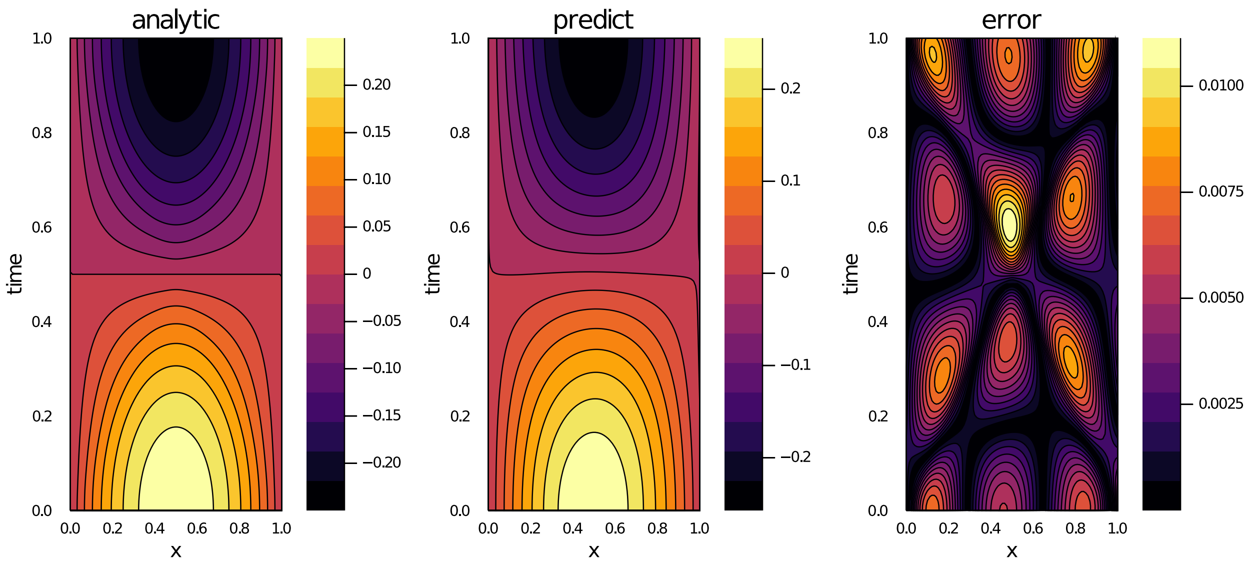

Figure 3: Neural network solution for 1D wave equation compared with the analytical solution using Julia.

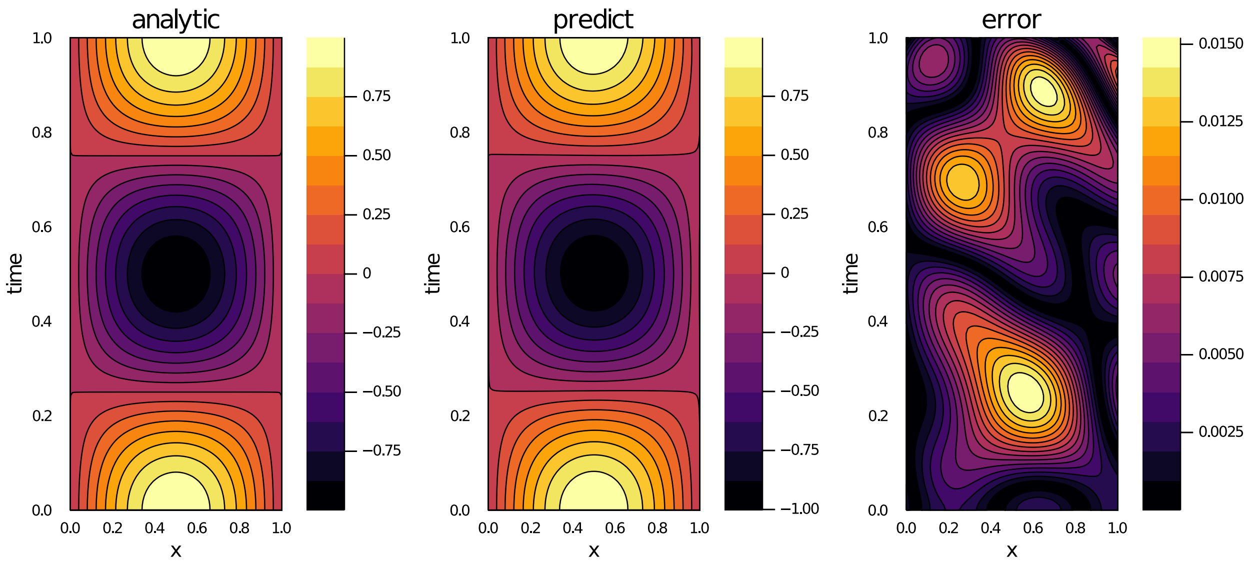

Figure 4: Neural network solution for 1D wave equation compared with the analytical solution using Julia.

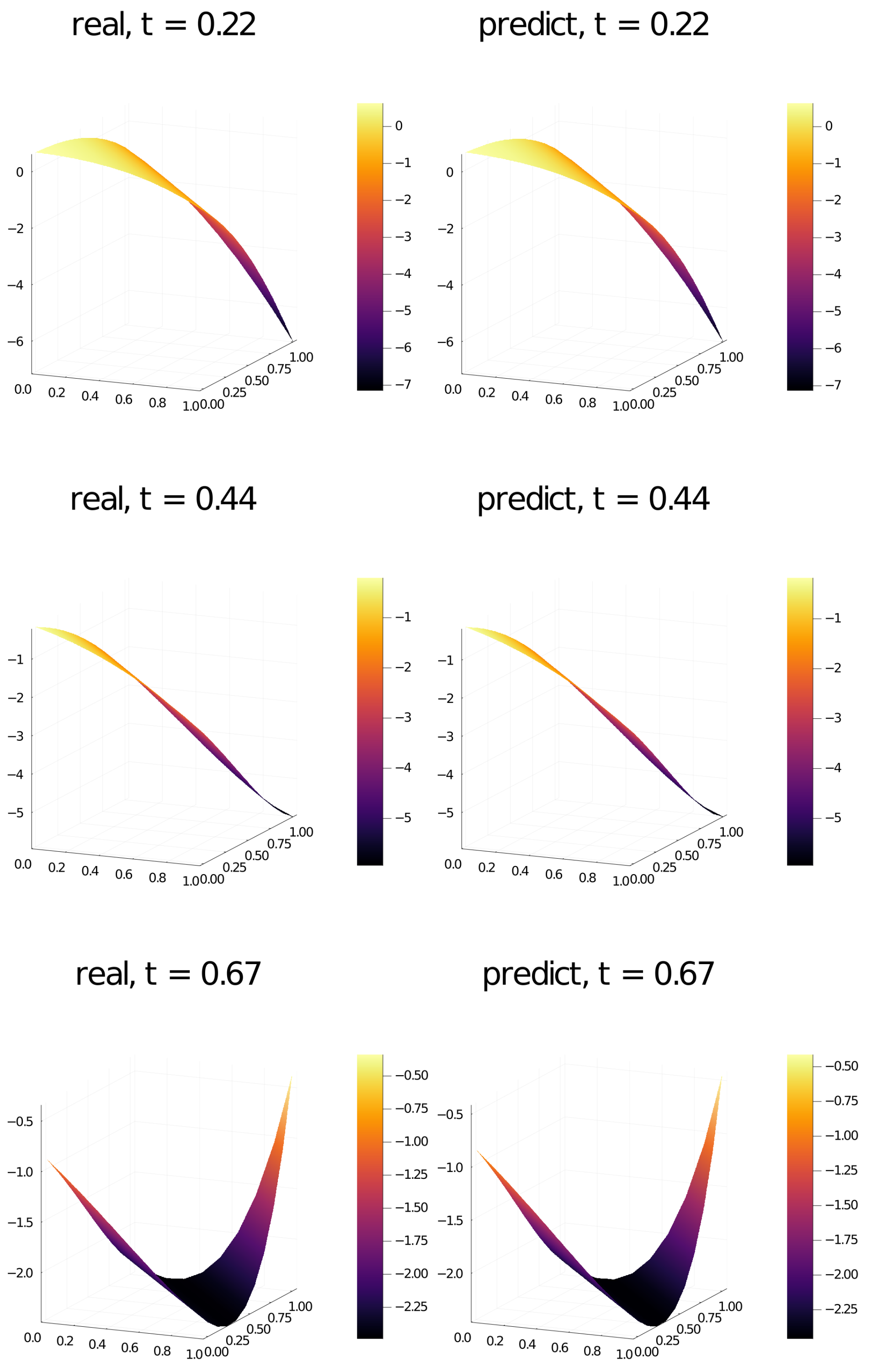

Figure 5: 2D heat equation results

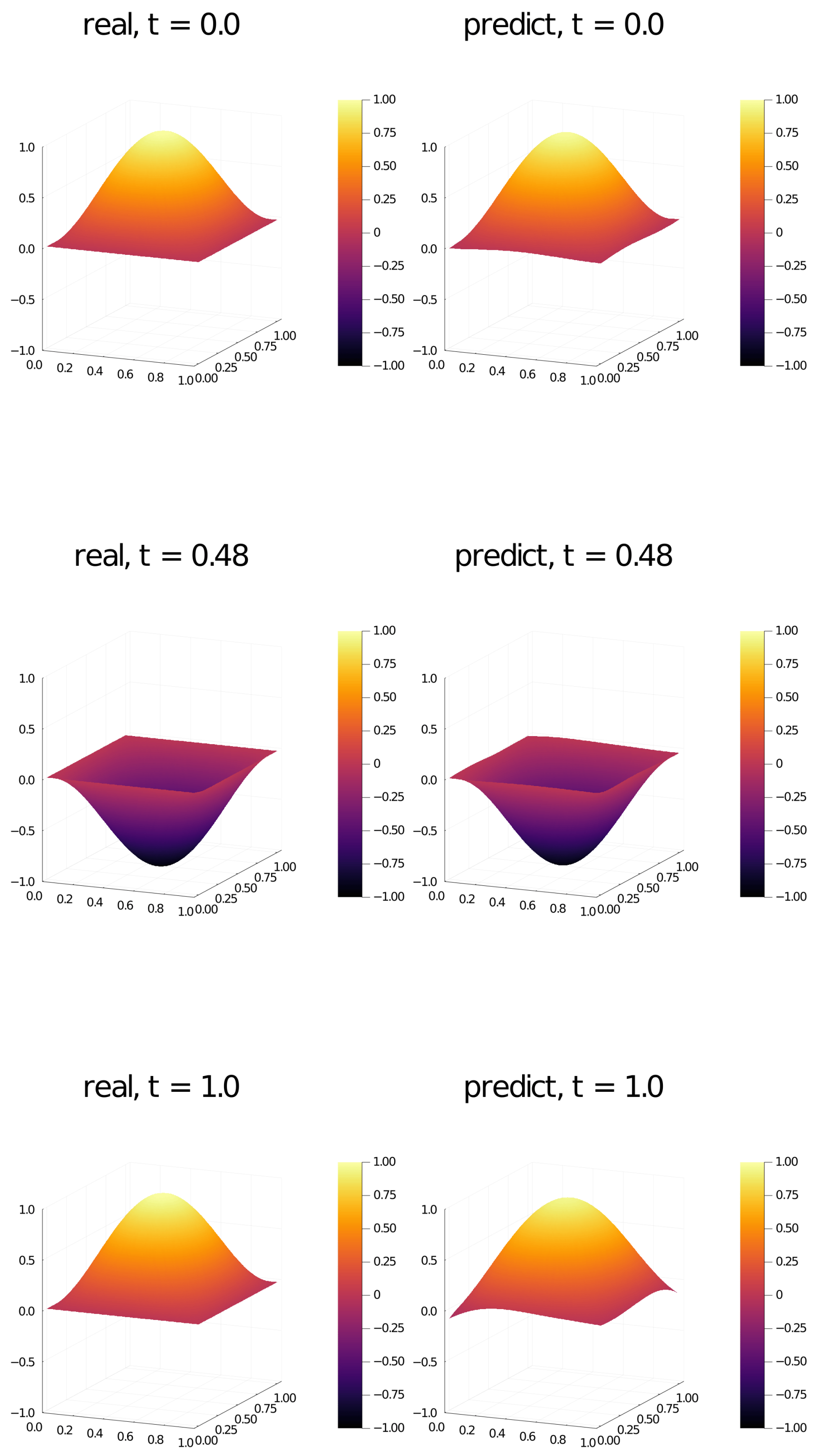

Figure 6: 2D wave equation results

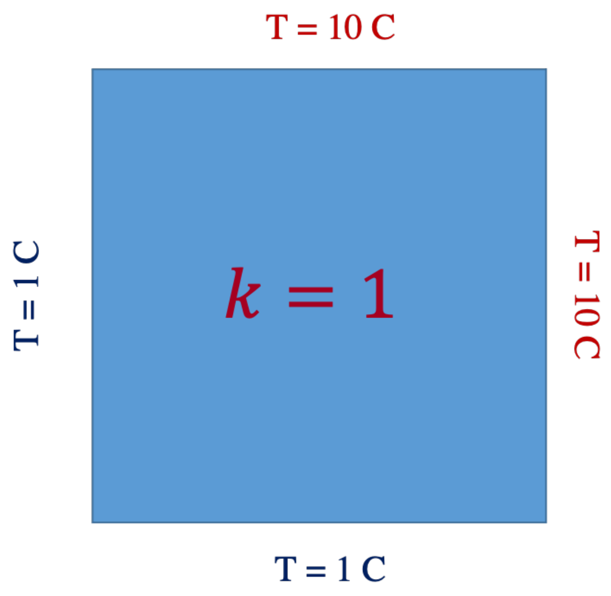

Figure 7: 2D domain with boundary temperatures

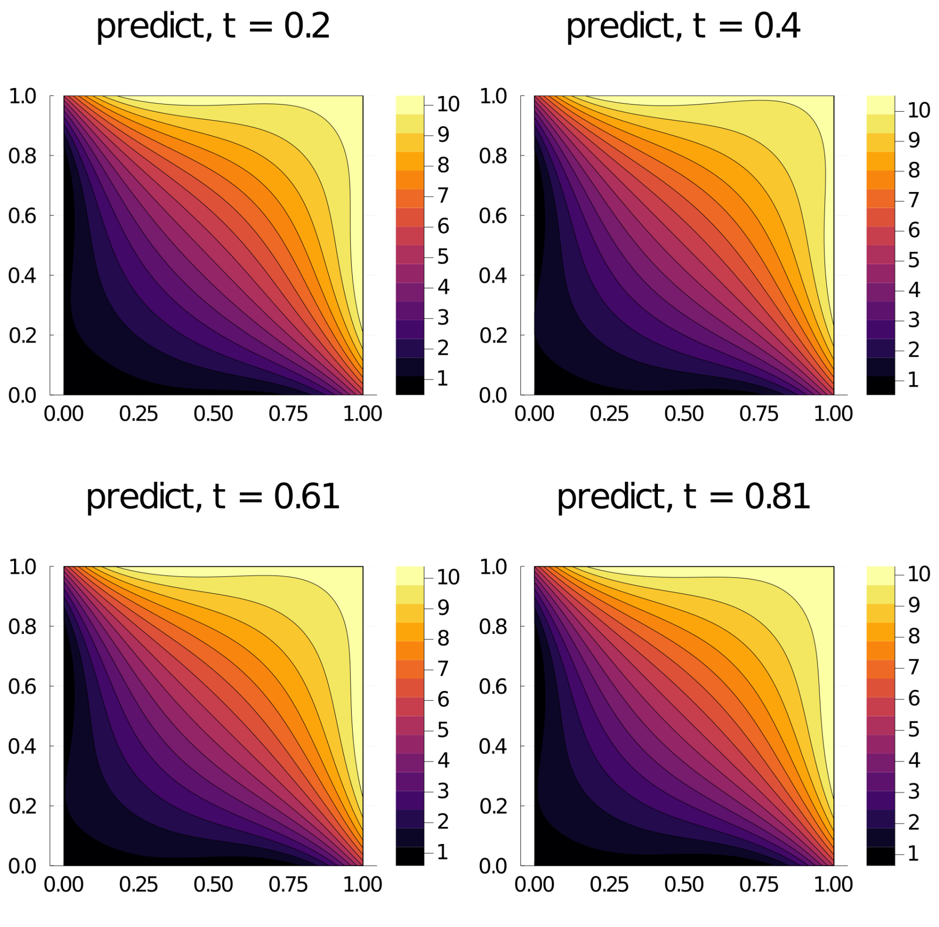

Figure 8: PINN solution for the problem