DATA AUGMENTATION OF HEAT-LIKE EQUATIONS USING PHYSICS-INFORMED NEURAL NETWORK

Bin Hoang and Migara Jayasingha

Univeristy of Rochester

ABSTRACT

The past decade has seen an exponential growth of machine learning applications in all areas of science. In recent years, there has been mounting interest in combining machine learning techniques with traditional science to tackle difficult problems where only partial domain knowledge and limited data measurements are available. Such problems are frequently encountered in medical science, where the measurements of the fluid flows in the domain of interests such as the heart and brain are limited due to safety concerns. We propose that physics informed neural network, or PINN, a new type of physics-guided neural network, can be used to tackle these problems.

***Index Terms*—** Machine Learning, Physics Informed Neural Network, Fluid Dynamics

1. INTRODUCTION

PINN stands for Physics-Informed Neural Networks, a type of machine learning algorithm that combines deep neural networks with partial differential equations to solve complex physical problems. PINNs are used to model physical systems, such as fluid dynamics or structural mechanics, by incorporating knowledge of the underlying physical laws into the neural network architecture.

The main difference between traditional neural networks and PINNs is that traditional neural networks are purely data-driven, while PINNs incorporate prior knowledge of physical laws into the neural network architecture. By doing so, PINNs are able to learn from limited data, and also can generalize to unseen scenarios that obey the same physical laws. The physical laws are enforced into the network learning process by defining error terms, which are often referred to as *"physics-driven"* loss terms. In addition to the data-driven error term in general NNs, PINNs utilize both physics and data-driven errors to minimize the cost function.

2. METHODOLOGY

The overview of our approach is as follows. Synthetic data $u(x,t)$ is generated using finite differencing over the entire simulation domain. Data points within the simulation domain and at the boundary are randomly selected to create a training set $u_s(x,t)$. In our approach, the PINN is a fully connected neural net (FCC) that takes space-time inputs (x,t) and produces measurement predictions $u_p(x,t)$ over the entire simulation domain. The network learns using a combination of two different loss components. The first loss component is the measurement loss, given by the MSE of $u_p(x,t)$ and the corresponding ground truths from $u(x,t)$. The second loss component is the physics-guided loss, given by the square of the residual given by the governing equations. An example of a PINN procedure is given in Fig 1, where the network is used to simulate fluid flow obeying Burgers' equation:

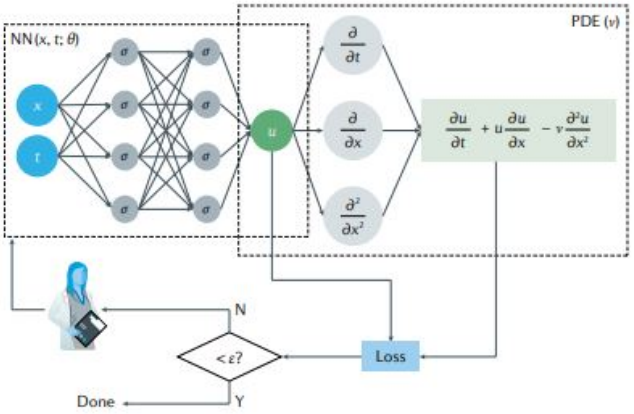


Figure 1: Example of PINN for simulating fluid flow using Burgers' equation. Source: George Karniakadis, 2021 [1].

For all the PINN models in this study, we use the L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shann) optimizer, which is a widely used optimization algorithm in PINNs. The L-BFGS optimizer is well-suited for PINNs because it is a quasi-Newton method that uses the Hessian approximation to update the weights of the neural network, making it computationally efficient for problems with large numbers of parameters. For data preprocessing, we follow a similar procedure to the public source codes given in [2, 3, 4] to properly format the physics ground truth data and compute the physics loss for neural network training.

2.1. Burgers' equation

The Burgers' equation is an important equation that occurs in many areas of fluid dynamics. Due to its inherent nonlinearity, Burgers' equation can be used to model shock waves and turbulence. Given space-time coordinate $(\mathbf{x}, t)$, the general 1D Burgers' equation describes the evolution of wave amplitude $u(x, t)$ controlled by a nonlinear convection term and a diffusivity term:

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} + \alpha\frac{\partial u}{\partial x^2}. \tag{1}$$

Here $u\frac{\partial u}{\partial x}$ is the convection term describing bulk flow driven by a pressure gradient, while $\alpha\frac{\partial u}{\partial x^2}$ is the diffusivity term describing the spread of fluid due to random motion, where α is the diffusion speed.

Due to its nonlinearity, general solutions for the Burgers' equation are often not available. Thus, we simulate the ground truth data using finite differencing, according to the strong Wolfe condition:

$$\delta_t \leq \frac{\delta_x^2}{2\alpha}. \tag{2}$$

In our simulation, the initial condition is chosen to be a square pulse wave reside in the left half of the domain, with $\alpha = 0.1$. To initialize PINN, we create three separate types of input:

- IC input: The entire initial condition is passed to PINN as input to ensure PINN predicts the correct initial state.
- BC input: The boundary points $u(x, t)_{BC}$ at all time domain is passed to make sure boundary conditions are obeyed at all time.
- Collocation input: Randomly sample points inside the domain that has to obey physical laws. In this case, we only pass the space-time coordinates $(x, t)_{col}$ and not their measurements $u(x, t)_{col}$ to simulate the cases where $u(x, t)_{col}$ inside the domain are not measurable.

From these inputs, we generate 3 corresponding losses from our network's prediction u_p:

- The initial loss is defined as:

$$L_{IC} = \sum_x \Big(u(x, t = 0) - u_p(x, t = 0)\Big)^2. \tag{3}$$

- The boundary loss is defined as:

$$L_{BC} = \sum_t \Big(u_p(x = 0, t)^2 - 0\Big) + \Big(u_p(x = L, t)^2 - 0\Big)^2. \tag{4}$$

- The physics loss is defined as:

$$L_{phys} = \sum_{x,t} \Big(\frac{\partial u_p}{\partial t} + u\frac{\partial u_p}{\partial x} - \alpha\frac{\partial u_p}{\partial x^2}\Big)^2. \tag{5}$$

In training, the neural network will optimize u_p according to the combination loss using tanh activation.

2.2. Diffusion

In the cases where the wave speed is small and the diffusion term dominates, the Burgers' equation reduces to the diffusion equation. The diffusion equation is a partial differential equation that describes the time evolution of a quantity that diffuses, such as heat, mass, or particles, in a physical system. The equation for non-dimensional diffusion is given by:

$$\frac{\partial}{\partial t}u(\mathbf{x}, t) = \alpha\nabla^2 u(\mathbf{x}, t), \tag{6}$$

where $u(\mathbf{x}, t)$ is the concentration of the diffusing quantity at position $\mathbf{x}$ and time t, α is the diffusion coefficient, and ∇^2 is the Laplacian operator, which describes the spatial variation of u.

Solutions to the diffusion equation can be obtained using analytical techniques, such as separation of variables, or numerical methods, such as finite differences or finite element methods. In this study we use finite differences to generate data for 1D and 2D diffusion equation. To ensure numerical stability, the following Wolfe condition is applied:

$$\delta_t \leq \frac{\delta_x^2}{4\alpha}, \tag{7}$$

where δ_t and δ_x are the numerical step sizes in time and space domains respectively.

As aforementioned, the PINN architecture is fully-connected neural net with tanh activation functions. For diffusion cases, the physics loss given by equation 6 is:

$$\frac{\partial}{\partial t}u_p(\mathbf{x}, t) - \alpha\nabla^2 u_p(\mathbf{x}, t) = \text{residual} \tag{8}$$

We create two data sets using the finite difference method for two cases of initial conditions with parameters as given below:

- Pulse at the center of the lattice:

  radius of the pulse = 4, temperature of the pulse at the beginning = 100

- Set temperatures at the edges:

  temperature at the left and bottom edges = 100, temperature at the top and right = 0

3. RESULTS

3.1. 1D models

We simulate the 1D Burger equation with notation $\Psi(x) = u(x,t)$, $\delta_x = 0.1$ and $\delta_t = 0.004$, with a 1D coefficient diffusion $\alpha = 0.1$ (Fig. 1). The initial state is given as a square pulse wave located in the left half of the domain. To check for robustness of PINN output, we also induce Gaussian noise into the ground truth and test the performance of the net work (bottom row). MSE errors is approximately 0.001 for the no and low noise (noise strength 0.01) case, and 0.0922 (noise strength 0.3) for the high noise case.

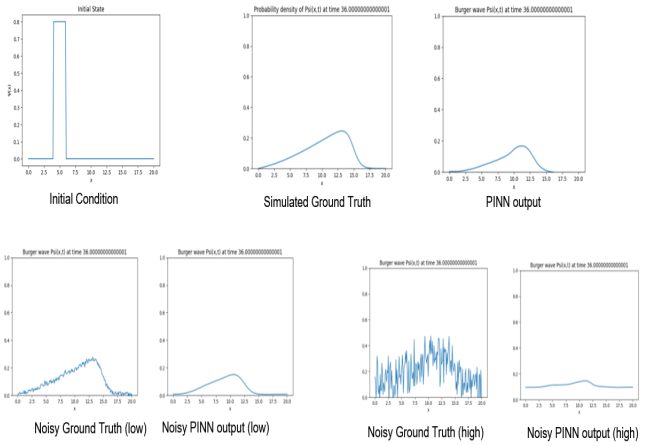


Fig. 1. Snapshot of PINN for simulating fluid flow using Burgers' equation at $t = 36s$. Top row, left to right: Initial state, simulated ground truth, PINN prediction. Bottom row, left to right: Ground truth with low noise corruption and corresponding PINN output, ground truth with high noise corruption and corresponding PINN output.

3.2. 1D Diffusion Equation

Using the same parameters as the Burgers' equation, we simulate the 1D Diffusion equation with $\Psi(x) = u(x,t)$, $\delta_x = 0.1$ and $\delta_t = 0.004$, with a 1D coefficient diffusion $\alpha = 0.1$. Similarly, the initial state is given as a square pulse wave located in the left half of the domain. Results with both the no noise and low noise (noise strength = 0.01) are given in Fig. 2.

3.3. 2D Diffusion Equation

For the generalization of the model, the 2D diffusion coefficient α is set to 1. The square lattice is generated with a step size of $\delta_t = 0.001, \delta_x = \delta_y = 1$, and when generating data, we have used two different initial conditions. The simulated data at a random time and the PINN output at the same time for the two initial boundary conditions are shown in the following figures.

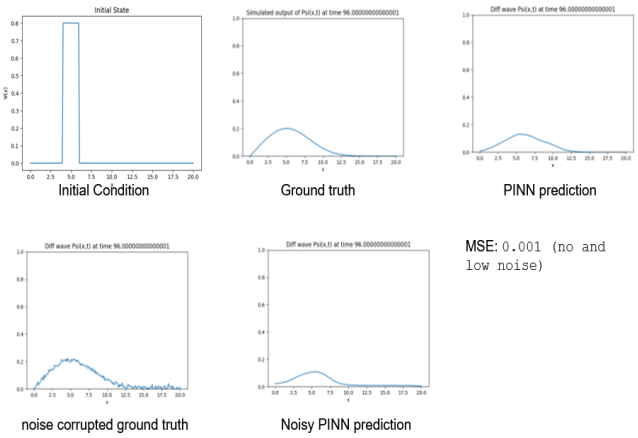
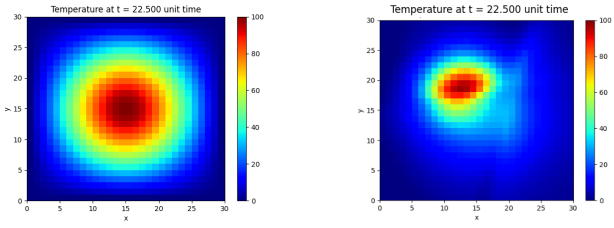


Fig. 2. Snapshot of PINN for simulating fluid flow using diffusion equation at $t = 96s$. Top row, left to right: Initial state, simulated ground truth, PINN prediction. Bottom row, left to right: Ground truth with low noise corruption (noise strength = 0.1) and corresponding PINN output

For the PINN, we use 6 dense layers each with $\tanh$ activation function. The input layer has 3 neurons to input positional (x, y) and time (t) features and the output layer is just one neuron to get the model-calculated concentration u_p.

3.3.1.

Set each edge at zero temperature and create a temperature pulse in the middle of the lattice.

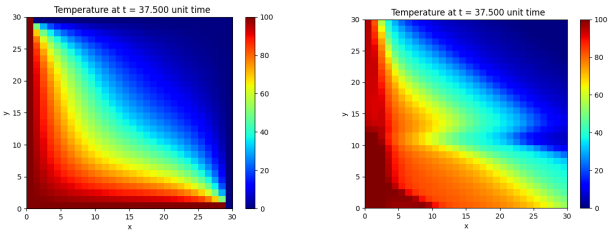


(b) Simulation result (c) PINN output

Fig. 3. The comparison of the simulated result (expected output) and the model output at a random temperature $t = 22.5$

3.3.2.

Set two edges (left and bottom) to a high value while keeping the other two edges at zero temperature.



(b) Simulation result (c) PINN output

Fig. 4. The comparison of the simulated result (expected output) and the model output at a random temperature $t = 37.5$

4. DISCUSSION

4.1. Data inference results

4.1.1. 1D PINN:

We see that for the 1D case, PINN is able to achieve good performance for both Burgers' and the diffusion equations. In the case of no and low noise (noise strength < 0.2), PINN is able to infer accurate physical predictions throughout the entire domain (MSE ≈ 0.001) with only the initial and boundary points as measurement inputs. For the case of higher noise (noise strength ≥ 0.2), PINN's prediction suffers from noise-induced artifacts. Nevertheless, PINN are able to capture the significant features of the system and showcase strong denoising capability in all 1D cases.

4.1.2. 2D PINN:

In general, both PINN models have shown acceptable results for both cases of the diffusion equation in 2D by capturing the general diffusivity of temperature in the two systems. However, the model has performed better in the system with initial case 1 (high temperature pulse at the center of the lattice) than in case 2 (L-shaped temperature edges), as shown in figures 3 and 4. One possible reason for this is that in case 2, the solution is more sensitive to the boundary error propagation because the initial heat source is located at the boundary.

4.2. Limitations and possible solutions

Our models show that in addition to the physics loss, initial and boundary losses are essential for a PINN architecture to produce accurate prediction. Moreover, the current architecture could be expanded to accommodate conservation laws, such as conservation of mass, momentum, and energy. We believe adding these addition constraints will be beneficial, especially in the case of significant noise corruption.

5. CONCLUSION

PINN performs well as a hybrid method particularly for data inference and denoising. Nevertheless, there are some limitations that might reduce the overall performance of the model. Such limitations include the background noise and the choice of the initial and boundary conditions. Inclusion of the initial and boundary condition losses can greatly improve the model's performance. Additionally, the accuracy of the model can be increased by introducing new losses governed by physics, such as conservation of energy, mass, and momentum. Choosing the correct physics model is crucial when creating PINN models, because the wrong model choice will output misleading predictions. Despite its limitations, our results have demonstrated that PINN has a lot of potentials in tackling real world problems and could open many opportunities for future research.

6. REFERENCES

[1] Karniadakis et al, "Physics-informed machine learning," in *Nature Review Physics*, 2021, vol. III, p. 422–440.

[2] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.

[3] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis, "Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10566*, 2017.

[4] Maziar Raissi, Paris Perdikaris, and George E Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.