Physics Informed Neural Networks (PINNs) Approach for Numerical Solution of Heat Equation

Gargi Rathore[1], Arushi Sharma[1] and Ishwar Kant[1]

[1] Department of Physics and Astronomical Science
Central University of Himachal Pradesh, Dharamshala, Bharat.

Submitted on 05-Feb-2026

Abstract

Heat conduction is a fundamental physical process governing thermal energy transport across diverse media. While traditional numerical methods (e.g., Finite Difference) are physically powerful, they often require complex and can be computationally expensive for high dimension. PINNs utilize the universal approximation power of neural networks constrained by the underlying physical laws. We present a Python-based implementation demonstrating how PINNs can be used to visualize thermal diffusion. A physics-informed neural networks is formulated by embedding the governing heat equation, along with the prescribed initial and boundary conditions, into the loss function of a neural network. Automatic differentiation is employed to compute the required spatial and temporal derivatives. The network is trained using collocation points sampled across the space–time domain, without the need for labeled training data. The PINNs-based solution exhibits excellent agreement with the actual solution of the heat equation, correctly reproducing the spatial temperature profiles and their temporal evolution. The study highlights the effectiveness of PINNs as a reliable alternative to traditional numerical methods.

1 The Heat Equation

The heat equation is a very special case of the general linear second-order partial differential equation[3]. The heat equation is a parabolic partial differential equation and occurs in the characterization of diffusion progress. The one-dimensional heat equation is:

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

where $u = u(x,t)$ is the dependent variable, and α is a constant coefficient. Equation 1 is a model of transient heat conduction in a slab of material with thickness L. The solution is defined on a strip of width L that extends infinitely in time. The material property α is the thermal diffusivity. In a practical computation, the solution is obtained only for a finite time, say t_{max}. Solution to Equation 1 requires specification of bound-

ary conditions at x =0 and x =L, and initial conditions at t = 0. In numerical calculations, the solution is evaluated only up to a finite time t_{max}. Equation 1 must be supplemented with boundary conditions at x =0 and x =L, along with an initial condition at t=0.

Simple boundary and initial conditions are

$u(0,t) = u_0, \quad u(L,t) = u_L \quad u(x,0) = f_0(x).$

Other boundary conditions, e.g. gradient (Neumann) or mixed conditions or dritchlet condition, can be specified.

For a homogeneous and isotropic medium where the material properties are uniform and identical in all directions and assuming constant thermal properties, the 2D heat conduction equation is expressed as

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{2}$$

where

- The function $u(x, y, t)$ denotes the temperature at position (x, y) at time t. It describes the thermal state of the medium and its temporal evolution, showing how the temperature varies due to internal and external influences.
- The parameter α represents the thermal diffusivity of the medium, which characterizes the rate at which heat spreads through the material.
- $\frac{\partial u}{\partial t}$ signifies the rate of change of temperature with respect to time, indicating the dynamic evolution of the thermal state. This term is essential for understanding transient heat conduction, where temperature changes occur over time.
- $\frac{\partial^2 u}{\partial x^2}$ and $\frac{\partial^2 u}{\partial y^2}$ are the second-order partial derivatives of temperature with respect to the spatial coordinates x and y, respectively. They characterize the local curvature of the temperature profile and indicate how variations in spatial gradients govern heat transport in the medium.

These equations serves as a mathematical model to describe the temporal and spatial distribution of temperature in a two-dimensional medium. It is crucial for understanding how heat propagates within materials and across various mediums, impacting a wide range of applications from industrial processes to environmental science.

Traditional numerical approaches, such as the finite difference and finite element methods, are based on discretizing the governing physical equations. In the finite difference method, derivatives in the differential equation are replaced by appropriate difference approximations[5]. For the heat equation, this involves discretization in both time and space. Different choices of grid points in these approximations lead to different numerical methods, each with its own convergence behavior. The rate at which the numerical solution converges to the exact solution depends on the method employed. Moreover, certain practical formulations may become unstable for inappropri-

ate choices of the spatial and temporal step sizes, Δx and Δt.

This work investigates the simulation of one- and two-dimensional heat conduction problems using Physics-Informed Neural Networks. The objective is to develop a robust PINN-based framework for modeling one- and two-dimensional heat conduction with boundary conditions, while exploiting neural networks to embed the underlying physical laws into the learning process.[6, 7]

Assume that we have a thermal system $u(x,t)$, where x and t stand for the spatial and temporal coordinates, respectively. To minimize the residual of the heat equation, $\frac{\partial T}{\partial t} - \alpha \nabla^2 T = 0$, a neural network is trained using the PINNs approach[6]. Even in 2D environments where conventional grid-based approaches would necessitate complicated indexing, using PINNs, this results in a highly accurate global approximation of the temperature field[6, 7, 2].

2 Physics Informed Neural Networks(PINNs)

In a traditional neural network, the model functions as a universal functional approximator, mapping inputs to outputs through a purely data-driven regime [1, 2]. Traditional Neural Network (TNN) learns patterns solely from labeled datasets without any intrinsic knowledge of the underlying governing equations. Three different kinds of layers make up the architecture:

- **Input Layer:** This layer gets the raw coordinate data, like time (t) and space (x, y). The number of neurons corresponds directly to the dimensionality of the input space.
- **Hidden Layers:** Non-linear transformations are carried out by these intermediate layers. Each layer consists of "neurons" that extract features from the input data. To balance computational efficiency and approximation power, we employ 64-neuron hidden layers.
- **Output Layer:** $\hat{u}$, is the temperature generated by the network's output layer.

In order to minimize the difference between predictions and observations, the network learns by modifying its internal parameters:

- **Weights and Biases:** Every connection between neurons has an associated weight W that determines how strongly one neuron influences another. In addition, each neuron has a bias b, which is a trainable constant added to the weighted sum of inputs to allow the neuron to shift its activation. These parameters are adjusted during training .

$$\mathbf{a} = \sigma\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) \tag{3}$$

Where:

  - $\mathbf{x}$ is the input coordinate vector.
  - $\mathbf{W}$ is the weight matrix.
  - $\mathbf{b}$ is the bias vector.

  - σ is the non-linear activation function.
  - **a** is the output activation of the layer.

- **Activation Function:** The hyperbolic tangent (*tanh*) function is utilized. Because physical problems often require computing second-order derivatives, a smooth, infinitely differentiable activation function is necessary.[7]

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

- **Loss Function:** The NN model is trained by minimizing the Mean Squared Error (MSE):

$$\mathcal{L}_{data} = \frac{1}{N}\sum_{i=1}^{N} |\hat{u}_i - u_i|^2 \tag{4}$$

The Traditional Neural Network does not ensure physical consistency between training points because the NN objective is restricted to minimizing $\mathcal{L}_{data}$.

PINNs, on the other hand, follows the rules of physics by minimizes a composite loss function. [6]. This is made possible by Automatic Differentiation (AD), which enables the network to precisely compute its own gradients[7].

Over a time interval $[0, t_{max}]$, the field $u(\mathbf{x}, t)$ in a 2D spatial domain Ω changes in accordance with:

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0, \quad \mathbf{x} \in \Omega, \quad t \in [0, t_{max}] \tag{5}$$

where the thermal diffusivity is represented by α. The system's initial state and boundary behavior must be specified in order to produce a unique solution[3].

- **Initial Condition (IC):** The state of the system at $t = 0$.

$$u(\mathbf{x}, 0) = h(\mathbf{x}) \tag{6}$$

where $h(\mathbf{x})$ usually denotes a particular distribution or a localized heat source.

- **Boundary Conditions (BC):** When the field value is fixed at the domain edges $\partial\Omega$, we use Dirichlet boundary conditions.

$$u(\mathbf{x}, t) = g(\mathbf{x}, t), \quad \mathbf{x} \in \partial\Omega \tag{7}$$

The sum of the Mean Squared Errors (MSE) from three different sources is the total objective function $\mathcal{L}$:

$$\mathcal{L}_{Total} = \omega_f \mathcal{L}_f + \omega_{bc} \mathcal{L}_{bc} + \omega_{ic} \mathcal{L}_{ic} \tag{8}$$

Where:

- **Physics Residual Loss($\mathcal{L}_f$):**
  - **1-D Case:**

$$\mathcal{L}_f = \frac{1}{N_f}\sum_{i=1}^{N_f} \left| \frac{\partial \hat{u}}{\partial t} - \alpha \frac{\partial^2 \hat{u}}{\partial x^2} \right|^2 \tag{9}$$

  - **2-D Case:**

$$\mathcal{L}_f = \frac{1}{N_f}\sum_{i=1}^{N_f} \left| \frac{\partial \hat{u}}{\partial t} - \alpha \left(\frac{\partial^2 \hat{u}}{\partial x^2} + \frac{\partial^2 \hat{u}}{\partial y^2} \right) \right|^2 \tag{10}$$

- **Boundary Condition Loss ($\mathcal{L}_{bc}$):**

$$\mathcal{L}_{bc} = \frac{1}{N_{bc}}\sum_{j=1}^{N_{bc}} |\hat{u} - g|^2 \tag{11}$$

- **Initial Condition Loss ($\mathcal{L}_{ic}$):**

$$\mathcal{L}_{ic} = \frac{1}{N_{ic}} \sum_{k=1}^{N_{ic}} |\hat{u} - h|^2 \qquad (12)$$

Nevertheless, the method is susceptible to Spectral Bias, in which the network learns low-frequency, smooth patterns more quickly than acute thermal gradients[2]. Our use of 64-neuron hidden layers and a learning rate of 10^{-3} helps to alleviate this, guaranteeing a balance between detail and stability[7].

3 Computational Implementation

Five different computational phases are used to solve the Heat Equation using Physics-Informed Neural Networks (PINNs), moving from raw domain sampling to a trained continuous solution manifold[6, 2].

3.1 Computational Procedure

Step 1: Point Sampling and Domain Initialization The continuous (x, t) domain is discretized into point sets prior to training:

- **Collocation Points:** A uniform random distribution is used to select 1,000 points throughout the domain in order to assess the PDE residual.
- **Constraint Points:** For the Initial Condition (IC), 200 points are fixed at $t = 0$, and for the Boundary Conditions (BC), 200 points are fixed at $x = 0$ and $x = 1$.

Step 2: Neural Network Architecture Setup
The initialization of the model $u(x, t) \approx NN(x, t; \theta)$ is as follows[1, 7]:

- **Input:** Input layer for coordinates (x, t) are connected to the hidden layers, each with 64 neurons.
- **Activation Function:** *tanh* functions to guarantee the heat equation's smooth, twice-differentiable curvature[7].
- **Output:** The temperature u is predicted by the output layer of the network.

Step 3: Physics-Informed Loss Construction
A composite loss function $\mathcal{L}_{total} = \mathcal{L}_f + \lambda(\mathcal{L}_i + \mathcal{L}_{bc})$ directs the "Learning": enforces the PDE using automatic differentiation: $u_t - ku_{xx} = 0$.

- **Residual Loss ($\mathcal{L}_f$):** Enforces the PDE $u_t - ku_{xx} = 0$ via Automatic Differentiation.
- **Initial Loss ($\mathcal{L}_i$):** Enforces $u(x, 0) = 0$.
- **Boundary Loss ($\mathcal{L}_{bc}$):** Enforces $u(0, t) = 1.0$ and $u(1, t) = 0.0$.

Step 4: Iterative Optimization
It consists of these two steps:

- **Optimizer:** It uses the **Adam** algorithm for stochastic weight updates[1].
- **Weighting:** To guarantee boundary convergence, $\mathcal{L}_i$ and $\mathcal{L}_{bc}$ are given a penalty factor ($\lambda = 10$). To achieve convergence, the procedure is iterated 4,000 times.

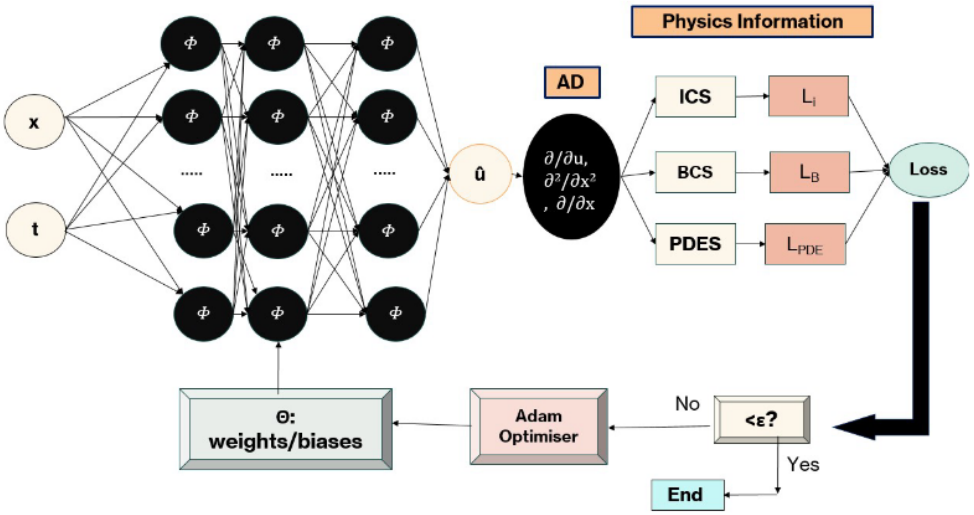


Figure 1: Architecture of a Physics-Informed Neural Network (PINN)

4 1D Heat Equation Results: Rod with Single-End Heater

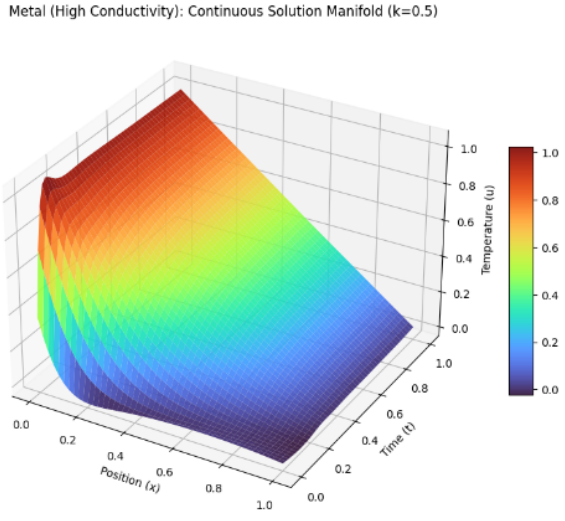
With a constant heater connected to the boundary at $x = 0$, the 1D simulation simulated a rod of length $L = 1.0$, keeping the temperature at $u(0,t) = 1.0$ for all $t > 0$. $u(x,0) = 0$ was the initial condition across the remainder of the rod. The continuous solution manifold over a temporal domain of $t \in [0,1]$ has to be resolved using the PINNs.

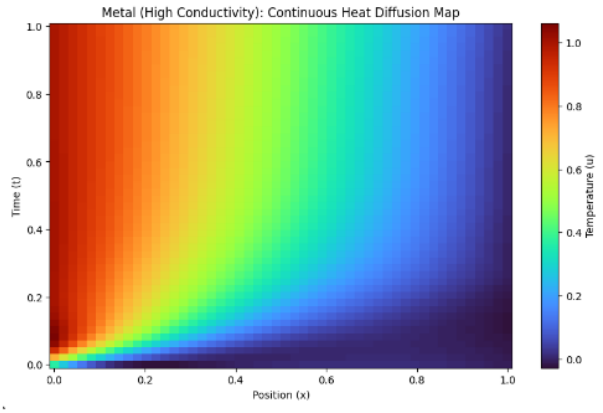
4.0.1 High-Conductivity Material (Metal)

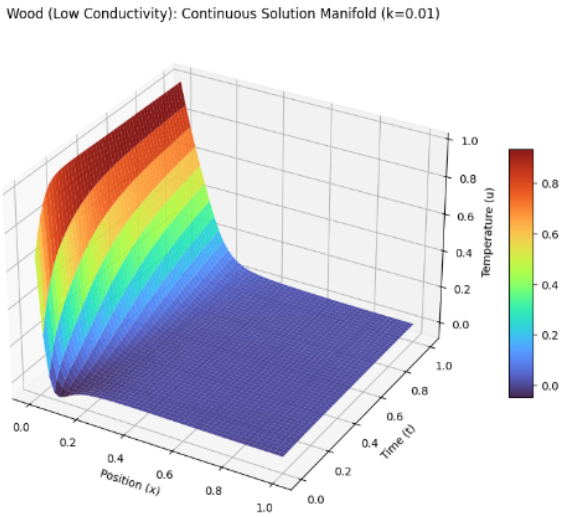
Rapid thermal propagation across the spatial domain was successfully captured by the PINNs for the metal simulation (set up with $k = 0.5$). As seen in the 3D solution manifold and the continuous heat diffusion map:

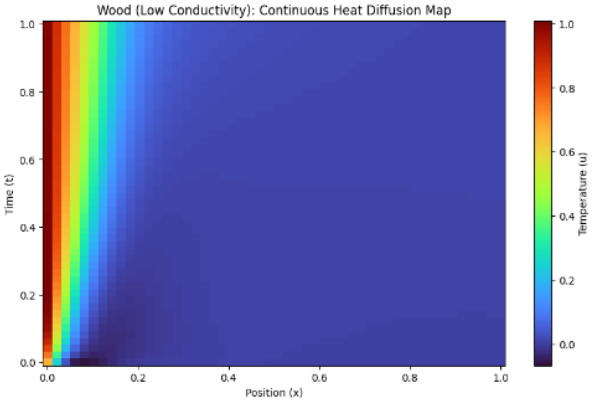
Efficient energy transfer is shown by the temperature profile's sweeping, smooth curvature. Even at the furthest points, the heater's thermal influence has reached the distal end of the rod ($x = 1.0$) by $t = 1.0$, with temperatures rising noticeably above the initial state.

Heatmap: Metal









4.0.2 Low-Conductivity Material (Wood)

The hardwood log (designed with $k = 0.01$) exhibits high localization and thermal resistance. The temperature increase is firmly limited to the immediate area of the heater ($x < 0.2$), according to the heat diffusion map. Near the source, the temperature rise but the rod stays at its starting ambient temperature for most of the time after $x = 0.2$. The PINNs remained stable and faithfully depicted the insulating behavior of the wood in spite of the steep gradients that usually provide a hurdle to conventional numerical solvers[7].

Heatmap: Wood

5 2D Heat Conduction in a Plate

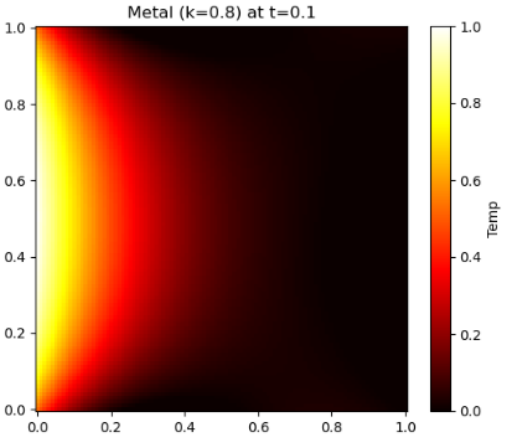
The simulation was extended to a 2D square plate of unit dimensions. The left edge ($x = 0$) was subjected to a continual heat source ($T = 1.0$), while the other limits were kept at $T = 0$. For two different materials, the PINNs successfully mapped the spatial-temporal evolution of heat.

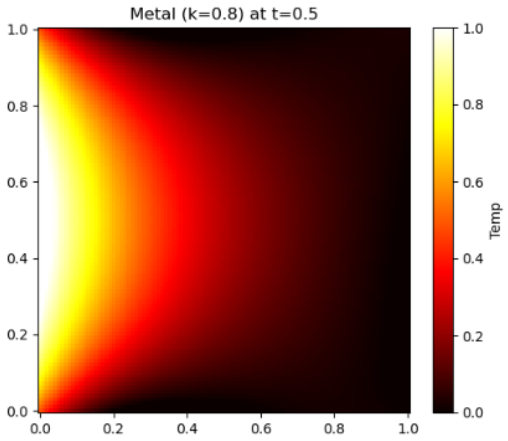
5.0.1 High-Conductivity Material (Metal)

The PINNs recorded a quick and wide thermal expansion, as seen in the heatmaps for

Metal ($k = 0.8$). The heat front had already made considerable inroads into the plate's center at $t = 0.1$. The efficient energy transfer feature of metallic structures is demonstrated by the thermal gradient being significantly smoother at $t = 0.5$, with the 0.2 temperature contour extending past the midpoint ($x = 0.6$).

Heatmap: Metal

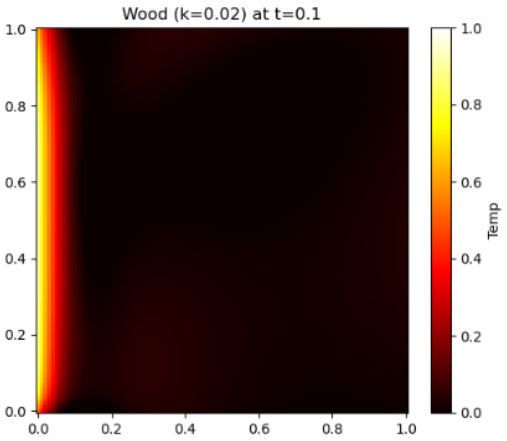


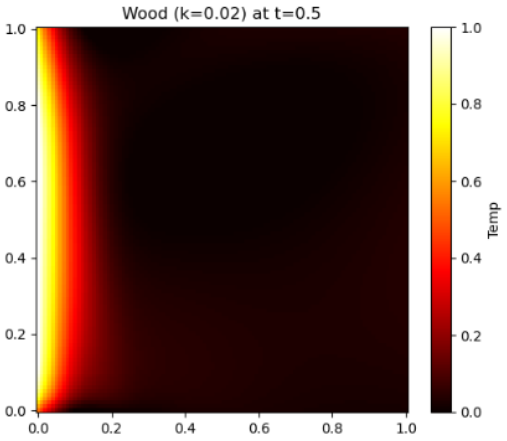


5.0.2 Low-Conductivity Material (Wood)

The Wood ($k = 0.02$) simulation, on the other hand, showed a very localized temperature response. The heat is still mostly contained in the area around the heated edge ($x < 0.2$) at both $t = 0.1$ and $t = 0.5$. The PINNs ability to resolve steep gradients in insulating materials without numerical instability is demonstrated by the abrupt change from the heater temperature to the ambient temperature.

Heatmap: Wood





The findings show that isotherms dispersed evenly over the domain and that the **metal plate** rapidly achieved a quasi-steady state. The PINNS correctly predicted a large thermal lag in the **wooden plate** simulation;

Table 1: Material Parameters used in 2D PINNs Simulations

Property	Metal	Wood
Conductivity (k)	0.8	0.02
BC ($x = 0$)	$T = 1.0$	$T = 1.0$
IC ($t = 0$)	$T = 0.0$	$T = 0.0$

during the simulation, the heat stayed focused along the $x = 0$ axis.

6 Conclusion

In this paper, we systematically show how well Physics-Informed Neural Networks (PINNs) solve the one- and two-dimensional heat equation. By incorporating the governing heat-conduction equation directly into the neural network loss function, the suggested PINN framework effectively captures the spatiotemporal evolution of the temperature field without the need for either temporal or spatial discretization. Compared to traditional numerical solvers like the Finite Difference Method (FDM) and the Finite Element Method (FEM), the PINN approach is more adaptable when dealing with complicated boundary and initial conditions. The findings show that PINNs offer a reliable and effective substitute for simulating heat conduction issues, especially when conventional grid-based approaches become computationally costly or challenging to use.

References

[1] Yoshua Bengio, Ian Goodfellow, Aaron Courville, et al. *Deep learning*, volume 1. MIT press Cambridge, MA, USA, 2017.

[2] Miaomiao Chen, Ruiping Niu, and Wen Zheng. Adaptive multi-scale neural network with resnet blocks for solving partial differential equations. *Nonlinear Dynamics*, 111(7):6499–6518, 2023.

[3] Lawrence C Evans. *Partial differential equations*, volume 19. American mathematical society, 2022.

[4] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[5] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM, 2007.

[6] M Raissi, P Perdikaris, GE Karniadakis, and Bhavesh Shrimali. Cs598: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear pdes. 2021.

[7] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[8] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.