Friedrich-Alexander-Universität
Erlangen-Nürnberg

Physics-Informed Neural Nets for approximating the Heat Equation

Introduction to Control and Machine Learning - WiSe 24/25

Pedro Blöss, Bilal Bostan, Haiyun Ning, Katharina Merzlikin, Zeyu Tan, Symon Islam

[1]Friedrich-Alexander Universität Erlangen-Nürnberg, Department Mathematik

January 13, 2025

Introduction

Heat Equation

- Models how heat spreads in a medium over time, with applications in engineering, environmental science, and other fields.
- Efficient solutions are critical for applications including insulation design and thermal management.

What are PINNs?

- *Physics-Informed Neural Networks (PINNs)* embed physical laws (e.g., PDEs) into neural network training.
- Leverage the known physics to constrain the learning process, improving accuracy with limited data.

How PINNs Solve the Heat Equation

- Incorporate the heat equation into the loss function.
- Minimize data loss and physics residuals to ensure consistency with physical laws and observed data.

Relationship to Control Theory

- Residuals act as *feedback*, guiding training like error correction in control systems.
- The loss function serves as a *control mechanism* to enforce physical constraints, boundary, and initial conditions.

Heat Equation

The Heat Equation

$$\frac{\partial u}{\partial t} - \alpha \Delta u = f(x, y, t) \tag{1}$$

Explanation of Variables

- $u(x, t)$: Temperature at position x and time t.
- $\frac{\partial u}{\partial t}$: Rate of change of temperature with respect to time.
- $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$: The Laplacian, representing how temperature diffuses through space.
- $f(x, y, t)$: external heat source
- α: Thermal diffusivity, indicating how quickly heat spreads.

Physical Interpretation

- The heat equation models how thermal energy moves through a material, which is important for designing efficient thermal management systems.
- It can also be used to predict temperature distribution over time in various physical systems.

PINNs: Scheme

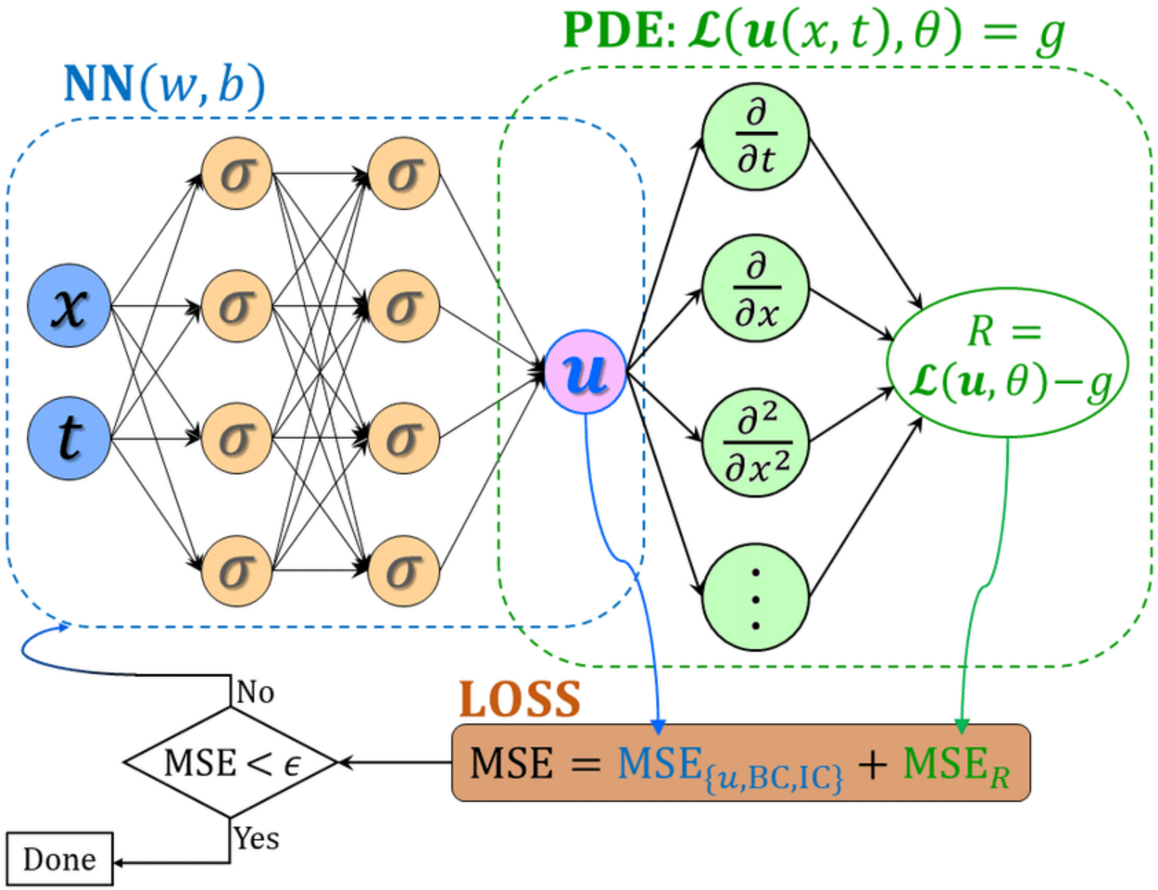


Figure: Illustration by (Meng, Li, Zhang, and Karniadakis 2020)

Approximation to the Heat Equation

We consider the **2D heat equation** with an external source $f(x,y,t)$ in a spartial domain $\Omega = (0, L_X) \times (0, L_y)$ and the time interval $t \in [0, T]$:

$$\frac{\partial u}{\partial t} - \alpha \Delta u = f(x,y,t), \quad (x,y,t) \in \Omega \times (0,T),$$

where $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$.

Boundary Conditions (homogeneous for simplicity)

$$u(x,0,t) = 0, \quad u(x,L_y,t) = 0,$$
$$u(0,y,t) = 0, \quad u(L_x,y,t) = 0 \quad \forall t \in [0,T]$$

Initial Condition specifies the temperature distribution at $t = 0$:

$$u(x,y,0) = u_0(x,y) \quad \forall (x,y) \in \Omega,$$

where $u_0(x,y)$ is the given initial temperature distribution.

Designing the Loss function

Generally, a PINN loss function for approximating the heat equation can be designed by a composition of the terms:

- PDE Loss: $\mathcal{L}_{\text{PDE}} = \sum_{i=1}^{N_\Omega} \left(\frac{\partial u}{\partial t} - \alpha \Delta u - f(x, y, t) \right)^2$ with $\lambda_{\text{PDE}} = \frac{1}{N_\Omega}$
- Boundary Conditions Loss: $\mathcal{L}_{\text{BC}} = \sum_{i=1}^{N_{\partial\Omega}} (u_\theta(x, y, t) - g_{\text{BC}}(x, y, t))^2$ with $\lambda_{\text{BC}} = \frac{1}{N_{\partial\Omega}}$
- Initial Conditions Loss: $\mathcal{L}_{\text{IC}} = \sum_{i=1}^{N_0} (u_\theta(x, y, 0) - g_{\text{ic}}(x, y))^2$ with $\lambda_{\text{IC}} = \frac{1}{N_0}$

So the Total Loss is:

$$\mathcal{L} = \lambda_{\text{PDE}} \mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}} \mathcal{L}_{\text{BC}} + \lambda_{\text{IC}} \mathcal{L}_{\text{IC}} \tag{2}$$

with weights $\lambda_{\text{PDE}}, \lambda_{\text{BC}}, \lambda_{\text{IC}}$.

Analysis: Existence and Uniqueness of solutions

Given that the Heat Equation is a linear and parabolic PDE, the **Lax-Milgram theorem** can be applied to guarantee the **existence and uniqueness of solutions in** $H_0^1(\Omega)$.

Proof.

Given the problem $-\Delta u = g$, with $g = f - u_t$, the weak formulation yiels (after integrating by parts)

$$\int_\Omega \nabla u \cdot \nabla v dx = \int_\Omega g v dx \quad , \forall v \in H_0^1(\Omega)$$

Note that the bilinear form $a(u,v) = \int_\Omega \nabla u \cdot \nabla v dx$ is coercive and continuous, and that the linear functional $L(v) = \int_\Omega g v dx$ is bounded. □

Theorem (Lax-Milgram (Alt 2016))

Let X *be a Hilbert space, the bilinear functional* $a : X \times X \to \mathbb{R}$*, and the linear functional* $L : X \to \mathbb{R}$*. If* $\forall u, v \in X$,

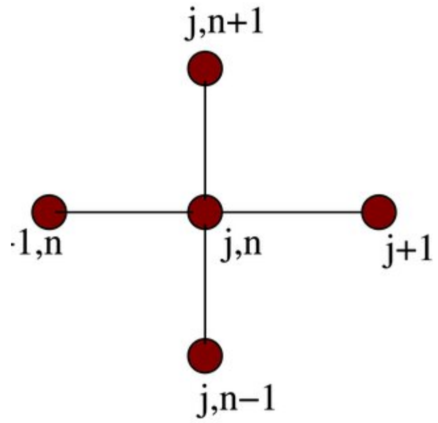
- *(coercivity)* $\quad a(u,u) \geq \alpha \|u\|_X^2, \quad \alpha > 0$
- *(continuity)* $\quad |a(u,v)| \leq C\|u\|_X \|v\|_X$
- *(boundedness)* $\quad L(v) \leq M\|v\|_X, \quad M > 0$

Then, there exists a unique solution $u \in X$ *to the weak problem* $a(u,v) = L(v), \quad \forall v \in X$

Baseline: Finite-Difference method

We used the **Forward Time Centered Space (FCTS) Finite Difference Method (FDM)** (Recktenwald 2004) as a **baseline to compare with the PINN results**.
This is a general explicit numerical method for PDE solutions, with the following recurrence equation:

$$\frac{u(t,i,j)-u(t-1,i,j)}{\alpha \Delta t} = \frac{u(t-1,i+1,j)-2u(t-1,i,j)+u(t-1,i-1,j)}{\Delta x^2} + \frac{u(t-1,i,j+1)-2u(t-1,i,j)+u(t-1,i,j-1)}{\Delta y^2}$$



PINNs convergence

Some factors introduce challenges into the theoretical formulation of PINNs solutions:

- Non-convexity of the NNs loss function;
- Nonlinearity of NNs;

Furthermore, the **convergence of PINNs** requires functional analysis tools, for example:

- "CONVERGENCE AND ERROR ANALYSIS OF PINNS" (Doumèche, Biau, and Boyer 2023) **States that the convergence depend on Sobolev inequalities**.
- "On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs" (Shin, Darbon, and Karniadakis 2020) **shows strong convergence with a Schauder approach adaptation**. More specifically, "Theorem 3.3 shows that neural networks that minimize the Holder regularized empirical losses (3.3) converge to the unique classical solution to the PDE"

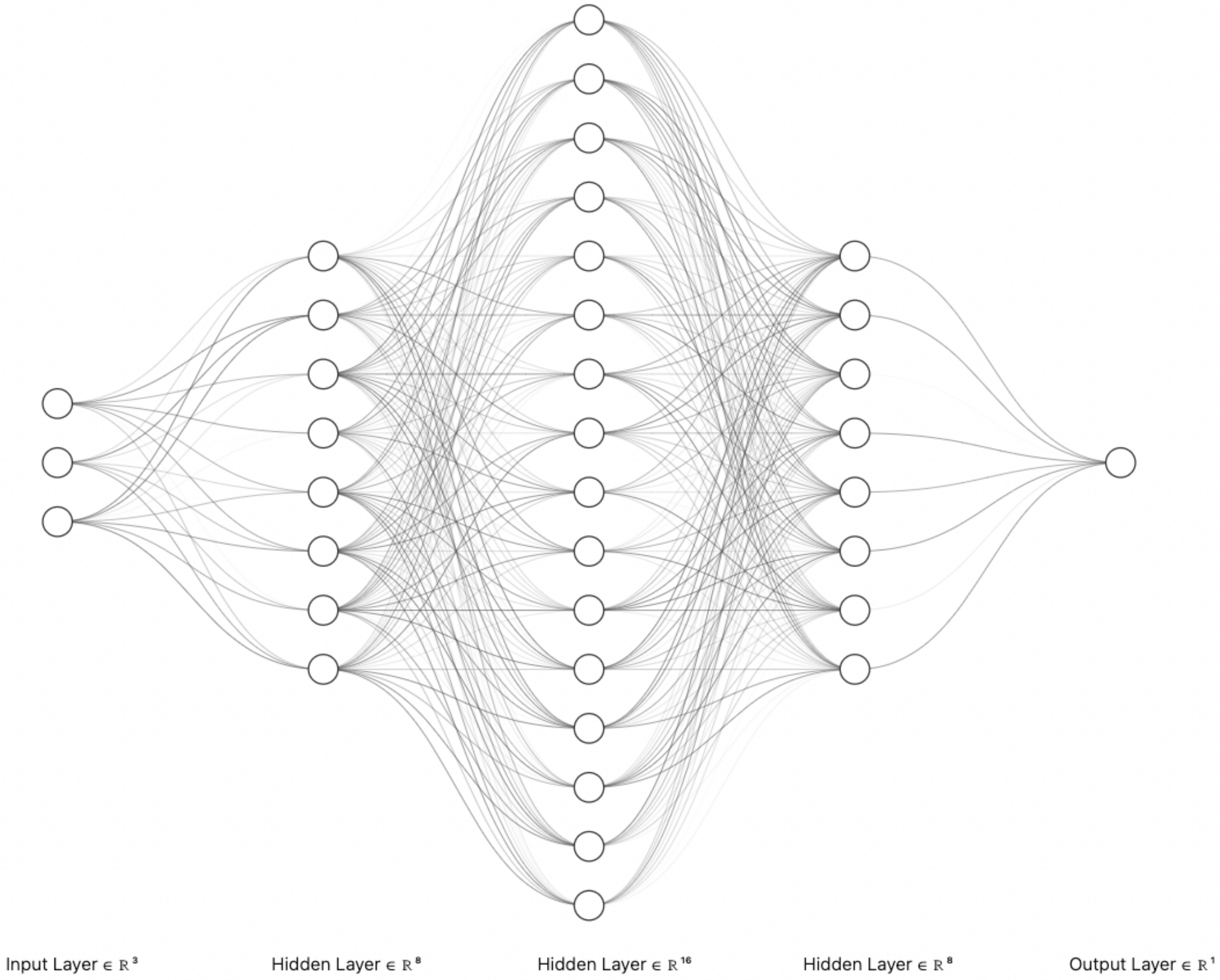
Simulation Setting

Network: Architecture with three hidden layers [8, 16, 8], with:

- **Input:** 3 features (spatial and temporal).
- **Active for hidden:** tanh better for a non-linear approximation.

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- **Output:** Scalar value representing heating.
- **Training:** 1000 epochs, learning rate: 0.001, optimizer: Adam, Thermal diffusivity: 0.1



Loss and Notation:

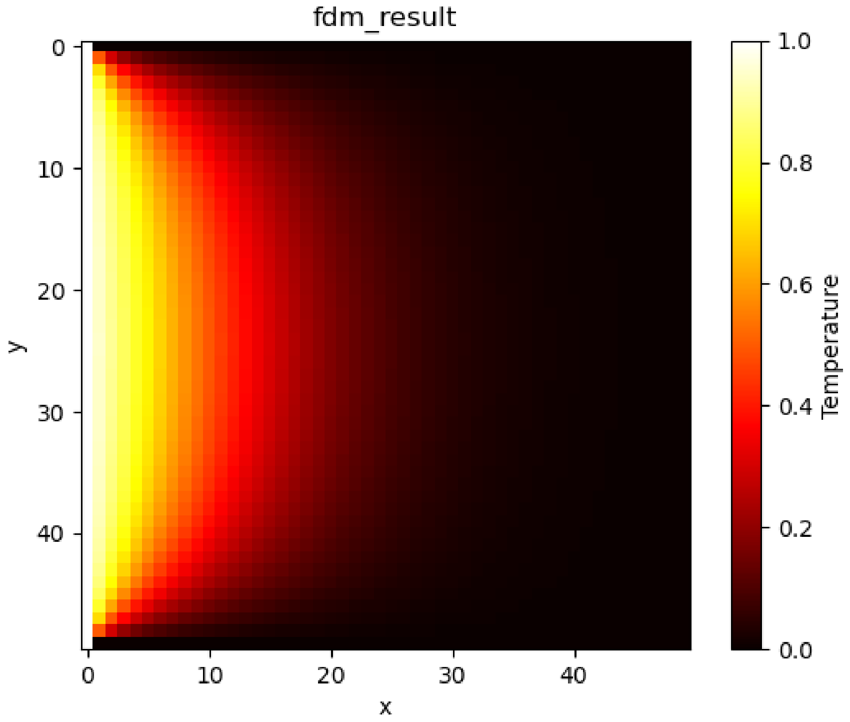
- $\mathbf{x}_b, \mathbf{y}_b, \mathbf{t}_b$: Boundary points (spatial and temporal).
- u_b: True boundary values at these points.
- $\mathbf{x}_i, \mathbf{y}_i, \mathbf{t}_i$: Interior points of the domain (spatial and temporal).
- $u(\mathbf{x}, \mathbf{y}, t)$: Neural network's predicted scalar output (e.g., temperature).
- $\frac{\partial u}{\partial t}$: First-order time derivative of u.
- $\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}$: First-order spatial derivatives of u.
- $\frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial y^2}$: Second-order spatial derivatives of u.
- α: Thermal diffusivity constant (or similar parameter).
- $\mathbf{x}_0, \mathbf{y}_0$: Spatial points at the initial time $t = 0$.
- u_0: True initial condition values at $t = 0$.
- u_d: True condition values at a future time $t = 1$ (true value from finite difference method).
- N_b: Number of boundary points.
- N_i: Number of interior points.
- N_0: Number of points at the initial condition and finite difference result.

$$\text{Total Loss} = \frac{1}{N_b}\sum_{j=1}^{N_b}\left(u(\mathbf{x}_b^j, \mathbf{y}_b^j, t_b^j) - u_b^j\right)^2 + \frac{1}{N_i}\sum_{i=1}^{N_i}\left(\frac{\partial u}{\partial t_i} - \alpha\left(\frac{\partial^2 u}{\partial x_i^2} + \frac{\partial^2 u}{\partial y_i^2}\right)\right)^2$$

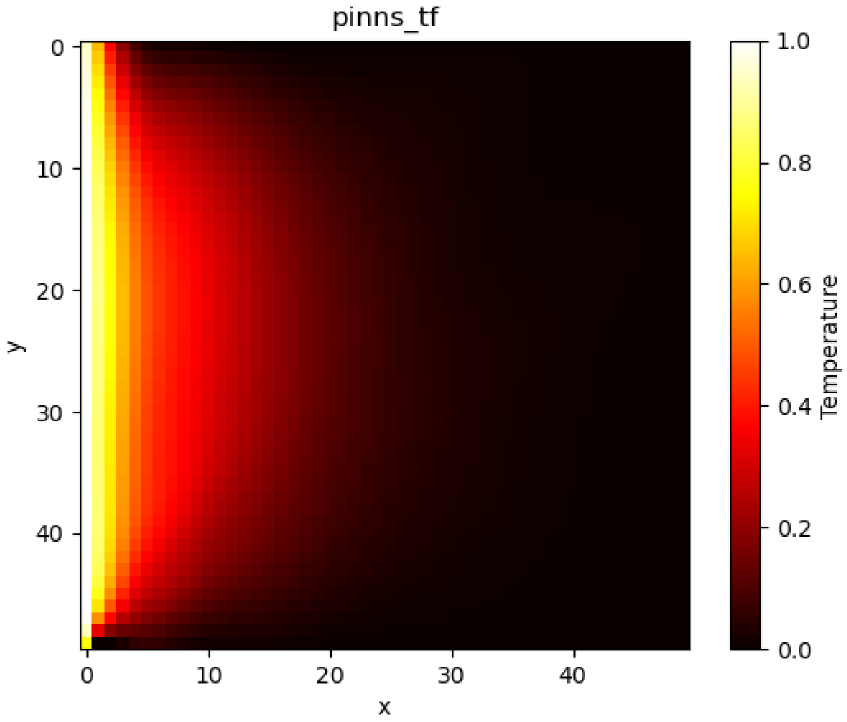
$$+\frac{1}{N_0}\sum_{k=1}^{N_0}\left(u(\mathbf{x}_0^k, \mathbf{y}_0^k, t=0) - u_0^k\right)^2 + \frac{1}{N_f}\sum_{m=1}^{N_f}\left(u(\mathbf{x}_f^m, \mathbf{y}_f^m, t_f^m) - u_{\text{FDM}}^m\right)^2$$

Result

Boundary Setting: The left boundary is set to 1, while all other boundaries are set to 0.



Finite Difference method result

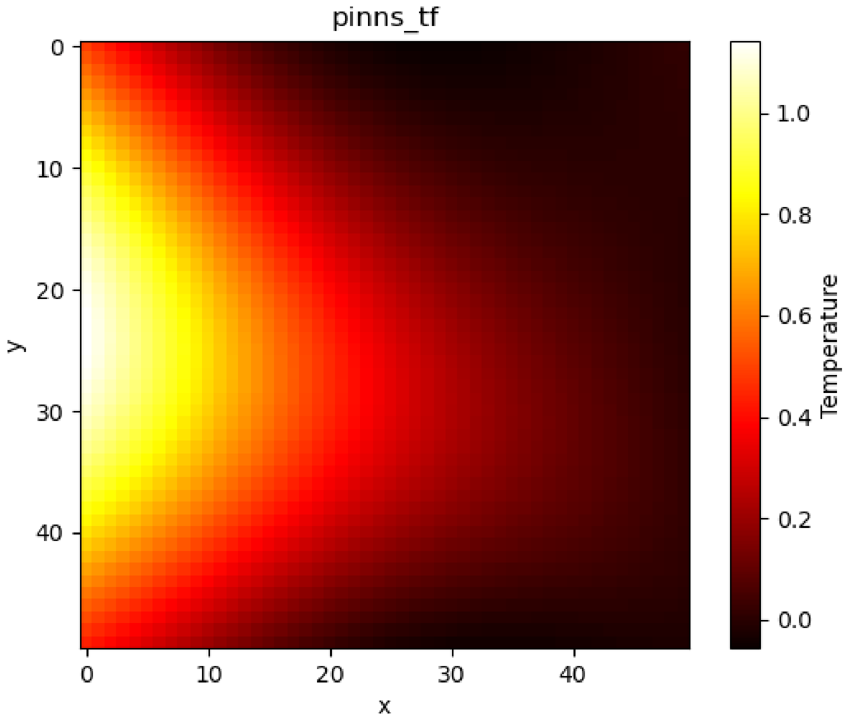


PINN result with tanh activation

PINNs without the initial condition

When the initial condition is excluded, the approximation improves compared to when it is included.

- $||u_{\text{FDM}} - u_{\text{PINN, no IC}}||_2 = 0.01655$
- $||u_{\text{Fourier}} - u_{\text{PINN, no IC}}||_2 = 0.02175$



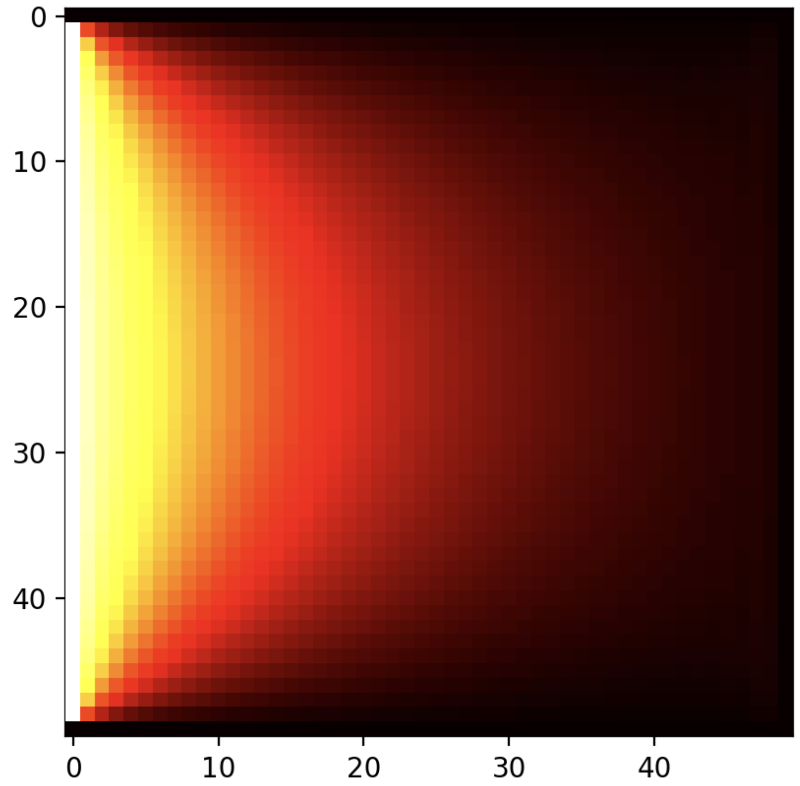
$$\mathcal{L} = \lambda_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}} + \cancel{\lambda_{\text{IC}}\mathcal{L}_{\text{IC}}}$$

Benchmark

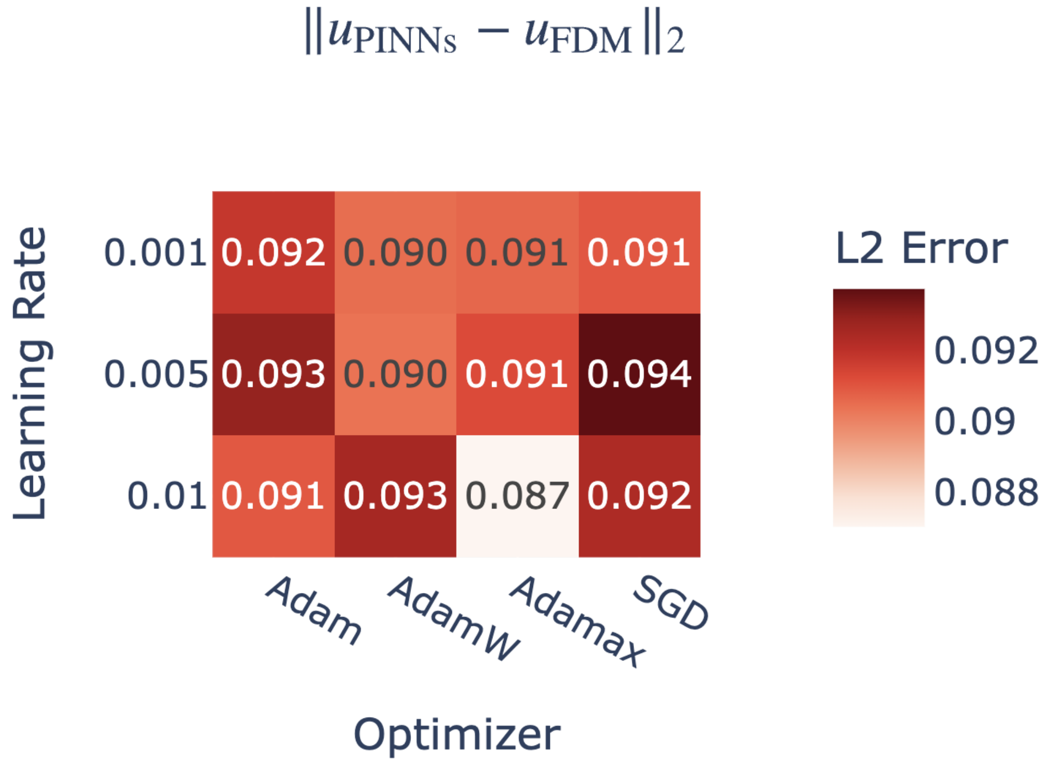
Fourier method: Using the Fourier method to simulate the heat equation(same boundary condition as before).

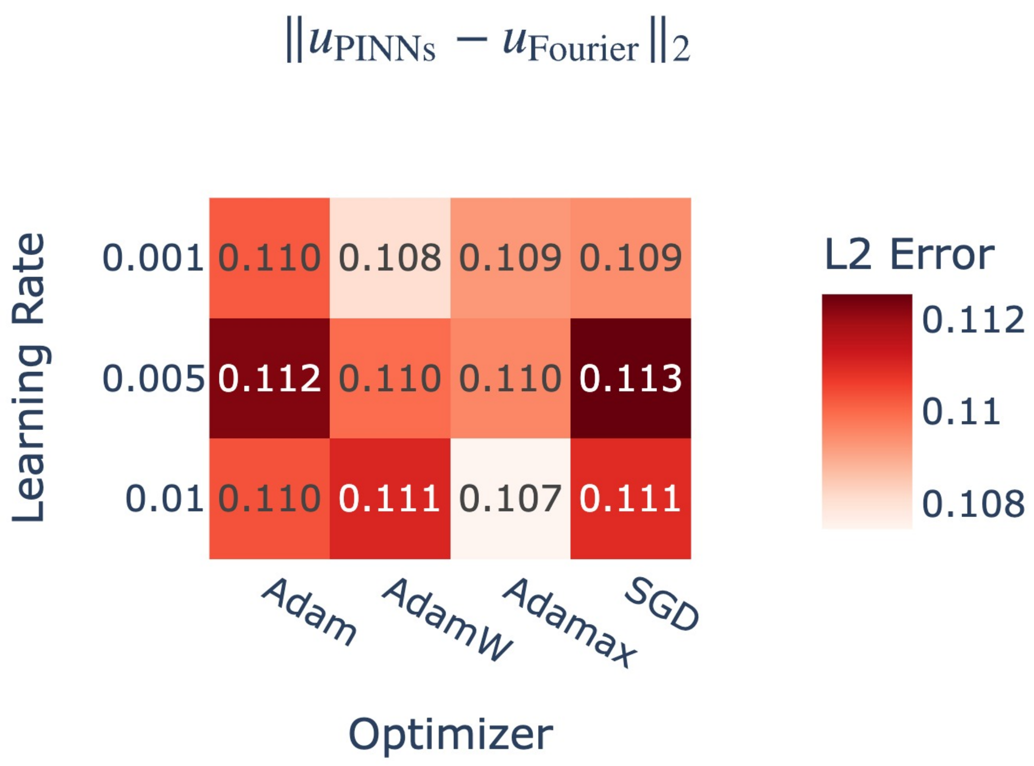
$$u(x,y,t) = \frac{1}{(2\pi)^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{u}(k_x, k_y, 0) \exp\left(-\alpha\left(k_x^2 + k_y^2\right) t\right) e^{i(k_x x + k_y y)} \, dk_x \, dk_y$$

The detailed simulation settings and mathematical derivations, which are quite lengthy, have been omitted here to focus on the results. The full details can be found at https://github.com/Haiyun314/intro-control-ml.



Testing different Optimizers and learning rates





- Adam showed smoother approximations, and is generally more suitable (adaptative learning).
- SGD requires careful tuning and may introduce irregularities in the heat distribution.

5. Conclusions and Discussions

Conclusions

- **Approximation error**: The simulation results present reasonable errors compared to baseline models.
- **Computational cost**
  - FDM time: 2.79855 s.
  - Fourier time: 0.08019 s.
  - PINN time: 62.563114 s.
- **Convergence**: Several references leverage the convergence of PINNs to PDE problems. (Doumèche, Biau, and Boyer 2023) (Shin, Darbon, and Karniadakis 2020) (Lorenz, Bacho, and Kutyniok 2024)
- **Time dependance**: PINNs approximate the entire domain at once, so they do not "evolve" solutions step-by-step like explicit FDM.
- **Advantages of PINNs**:
  - Mesh-free - discretization into a mesh is not required (Cuomo et al. 2022),
  - Automatic differentiation (Baydin, Pearlmutter, Radul, and Siskind 2018) - eliminating numerical errors,
  - Handling high-dimensionality data,
  - Flexibility of boundary conditions and loss.

References I

Alt, H. W. (2016). "Linear functional analysis". In: *An Application-oriented Introduction*.

Baydin, A. G., B. A. Pearlmutter, A. A. Radul, and J. M. Siskind (2018). "Automatic differentiation in machine learning: a survey". In: *Journal of machine learning research* 18.153, pp. 1–43.

Cuomo, S., V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli (2022). "Scientific machine learning through physics–informed neural networks: Where we are and what's next". In: *Journal of Scientific Computing* 92.3, p. 88.

Doumèche, N., G. Biau, and C. Boyer (2023). "Convergence and error analysis of PINNs". In: *arXiv preprint arXiv:2305.01240*.

Lorenz, B., A. Bacho, and G. Kutyniok (2024). "Error Estimation for Physics-informed Neural Networks Approximating Semilinear Wave Equations". In: *arXiv preprint arXiv:2402.07153*.

Meng, X., Z. Li, D. Zhang, and G. E. Karniadakis (2020). "PPINN: Parareal physics-informed neural network for time-dependent PDEs". In: *Computer Methods in Applied Mechanics and Engineering* 370, p. 113250.

Recktenwald, G. W. (2004). "Finite-difference approximations to the heat equation". In: *Mechanical Engineering* 10.01.

Shin, Y., J. Darbon, and G. E. Karniadakis (2020). "On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs". In: *arXiv preprint arXiv:2004.01806*.