Physics-Informed Neural Networks for Solving the Heat Equation

Anthony Merlin

April 2025

Abstract

This report explores Physics-Informed Neural Networks (PINNs) as a novel scientific machine learning approach to solving partial differential equations, specifically focusing on the heat equation. Unlike traditional numerical methods that rely solely on discretization, PINNs incorporate physical laws directly into neural network training through custom loss functions. This project implements PINNs to solve both forward and inverse heat equation problems, and compares the results with traditional finite element methods (FEM). The findings demonstrate that while FEM achieves slightly better accuracy for forward problems with constant coefficients, PINNs excel in solving inverse problems where the diffusion coefficient must be inferred from sparse measurements - a significant advantage over traditional approaches.

1 Introduction

The heat equation is a fundamental partial differential equation (PDE) that describes how heat diffuses through a material over time:

$$u_t = (\kappa(x)u_x)_x \tag{1}$$

where $u(t, x)$ represents the temperature at position x and time t, and $\kappa(x)$ is the thermal conductivity of the material. Traditional approaches to solving this equation include finite difference, finite element, and finite volume methods, which discretize the domain and approximate the solution at specific points.

Physics-Informed Neural Networks (PINNs), introduced by Raissi et al. (2019), offer an alternative approach by leveraging neural networks' universal approximation capabilities while enforcing physical constraints. PINNs represent the solution as a neural network and minimize a custom loss function that incorporates both data fitting terms and PDE residuals, effectively embedding the physics into the learning process.

This project explores two key problems:

1. **Forward Problem**: Given a known diffusion coefficient $\kappa(x)$, find the temperature field $u(t, x)$

2. **Inverse Problem**: Given sparse measurements of the temperature field $u(t, x)$, recover both the temperature field and the unknown diffusion coefficient $\kappa(x)$

2 Methodology

2.1 Physics-Informed Neural Networks Framework

PINNs use neural networks to approximate the solution to PDEs while incorporating physical laws into the training process. The key components include:

- A neural network $u_\theta(t, x)$ parameterized by weights θ to approximate the solution
- For inverse problems, a separate network $\kappa_\phi(x)$ to approximate the unknown coefficient
- A composite loss function that enforces:
  - Initial conditions: $u(0, x) = u_0(x)$
  - Boundary conditions: $u(t, 0) = u(t, 1) = 0$
  - PDE residual: $u_t - (\kappa(x)u_x)_x = 0$
  - Data fitting (for inverse problems): match sparse measurements

The networks are trained using gradient-based optimization to minimize this composite loss function.

2.2 Implementation Details

For both the forward and inverse problems, we implemented:

1. **Neural Network Architectures**:
   - Temperature field network: 4 hidden layers with 20 neurons per layer
   - Diffusion coefficient network (inverse problem): 2 hidden layers with 10 neurons per layer
   - Hyperbolic tangent (tanh) activation functions
2. **Training Data**:
   - Initial condition points: $u(0, x) = \sin(\pi x)$
   - Boundary condition points: $u(t, 0) = u(t, 1) = 0$
   - Collocation points: random $(t, x) \in [0, 1] \times [0, 1]$
   - For the inverse problem: 100 randomly sampled measurements from the forward solution

3. **Loss Function Components**:
    - Initial condition loss: $L_{IC} = \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} |u(0, x_i) - u_\theta(0, x_i)|^2$
    - Boundary condition loss: $L_{BC} = \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} |u(t_i, 0) - u_\theta(t_i, 0)|^2 + |u(t_i, 1) - u_\theta(t_i, 1)|^2$
    - PDE residual loss: $L_{PDE} = \frac{1}{N_f} \sum_{i=1}^{N_f} |u_t(t_i, x_i) - (\kappa(x_i)u_x(t_i, x_i))_x|^2$
    - Data loss (inverse problem): $L_{data} = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} |u(t_i, x_i) - u_\theta(t_i, x_i)|^2$
4. **Optimization**:
    - Adam optimizer with initial learning rate of 0.001
    - Exponential learning rate decay (gamma = 0.999)
    - 5000 epochs for forward problem, 2000 epochs for inverse problem

2.3 Finite Element Method

For comparison, we implemented a finite element solution using an implicit backward Euler scheme:

1. Discretized the spatial domain into 100 elements
2. Used a tridiagonal matrix system to solve the heat equation efficiently
3. Applied the same initial and boundary conditions as in the PINN

3 Results and Analysis

3.1 Forward Problem

The forward problem involved solving the heat equation with a known constant diffusion coefficient $\kappa(x) = 1$. The PINN was successfully trained with the loss function decreasing steadily over 5000 epochs, with the initial condition loss showing the most significant improvement (from $\sim$0.15 to $\sim$0.000003).

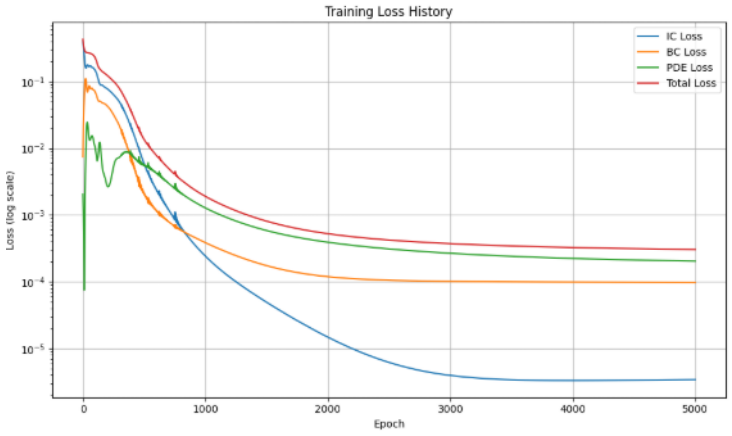

Figure 1: Training loss history for the forward problem showing the convergence of different loss components over 5000 epochs. The log-scale plot shows all loss components steadily decreasing, with the initial condition loss showing the most dramatic improvement.

The trained network accurately captured the temperature evolution, showing the expected diffusion behavior where the initial sine wave decays over time while maintaining its shape. The visualization of the solution shows:

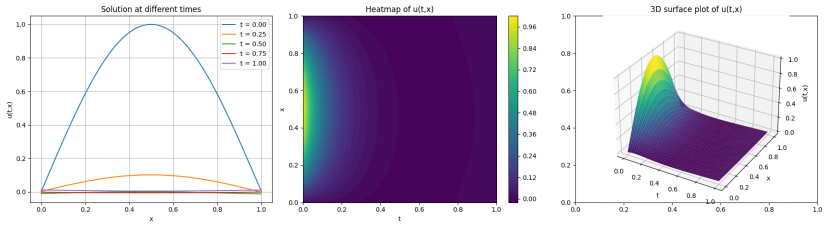

Figure 2: Solution visualization showing temperature profiles at different times (left), heatmap of the solution (middle), and 3D surface plot (right). The plots demonstrate how the initial sine wave diffuses over time while maintaining its shape.

Quantitatively, the PINN solution achieved:

- Maximum error vs. analytical solution: 0.026427
- Mean error vs. analytical solution: 0.006583

The error distribution showed characteristic patterns with higher errors near the initial time step and along the domain boundaries, consistent with the challenges in enforcing sharp transitions with neural networks:

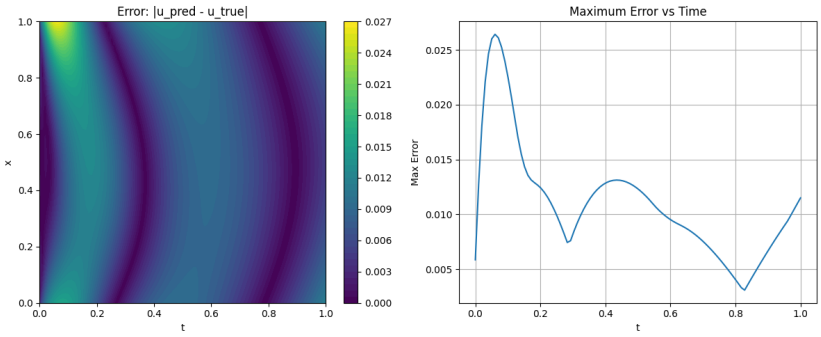

Figure 3: Error analysis showing the difference between predicted and analytical solutions, with a heatmap of the error (left) and maximum error plotted against time (right).

3.2 Inverse Problem

The inverse problem used 100 randomly sampled points from the forward solution to recover both the temperature field and the unknown diffusion coefficient. The sampling approach ensured good coverage of the space-time domain while using only a small fraction (1%) of the total solution points.

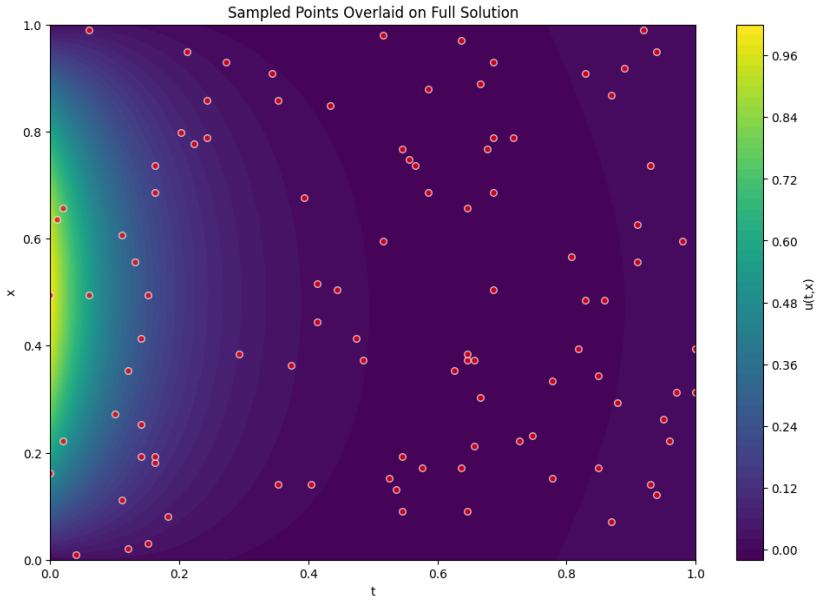

Figure 4: Visualization of the sampled points used for the inverse problem, showing their distribution in the space-time domain and their values. These sparse measurements (100 points) are used to recover both the temperature field and diffusion coefficient.

Training convergence was achieved after 2000 epochs, with all loss components showing satisfactory reduction:

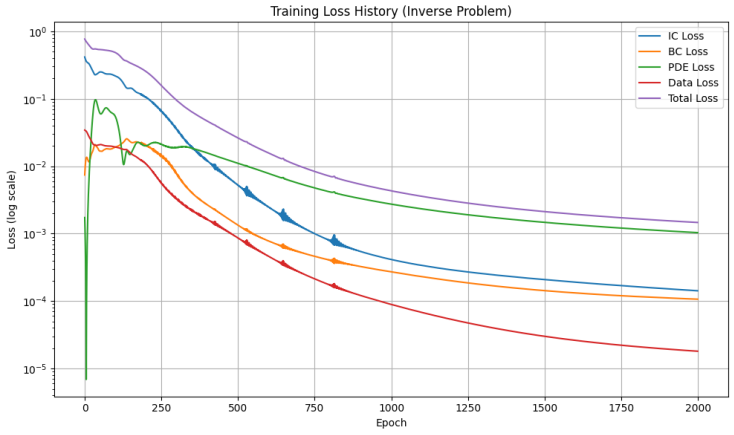

Figure 5: Training loss history for the inverse problem showing the convergence of all loss components over 2000 epochs, including the data fitting loss used to match the sparse measurements.

Most impressively, the network successfully recovered the diffusion coefficient:

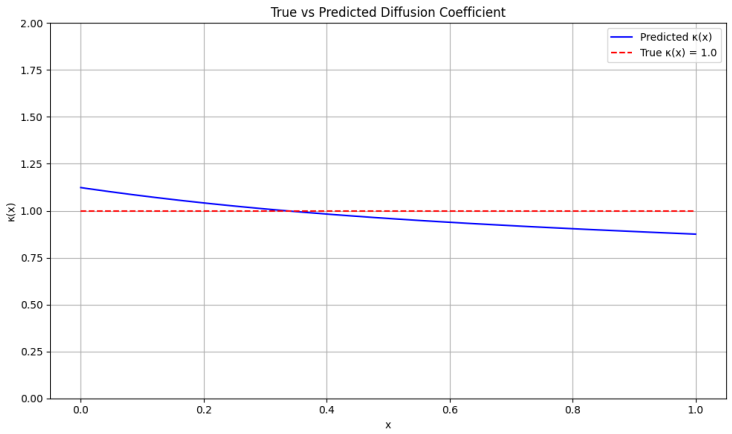

Figure 6: Comparison of true vs. predicted diffusion coefficient. The network accurately recovered the constant coefficient $\kappa(x) = 1.0$ with relatively small deviations.

- Mean absolute error: 0.066057
- Maximum absolute error: 0.124866

The recovered temperature field closely matched the analytical solution:

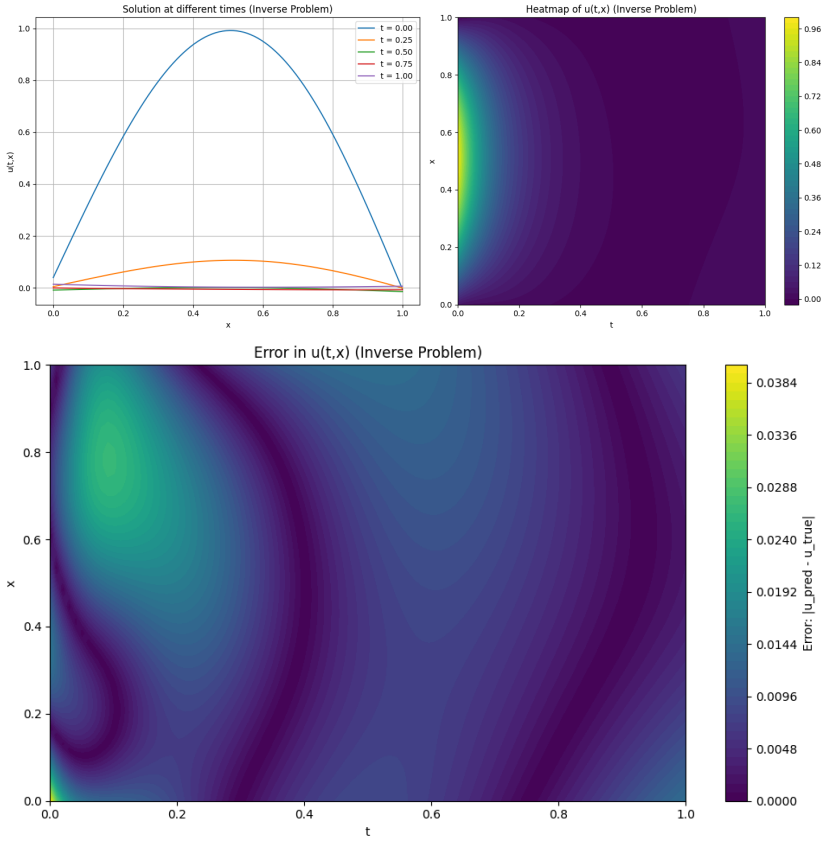

Figure 7: Visualization of the recovered temperature field from the inverse problem, showing solution profiles at different times and a heatmap of the solution, along with error analysis.

Notably, the inverse solution was achieved using only sparse measurements, demonstrating the power of physics-informed constraints.

3.3 Comparison with Finite Element Method

The FEM solution provided a useful benchmark for the PINN approach. Key comparisons include:

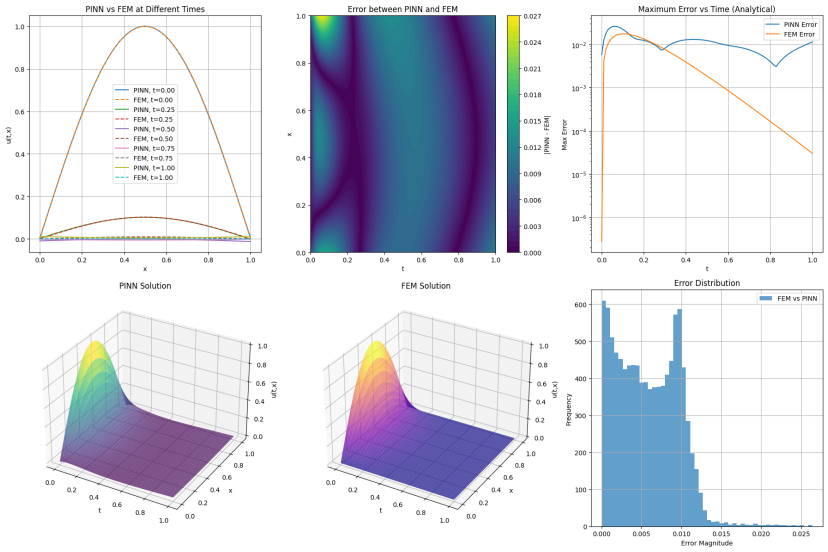

Figure 8: Comparison between PINN and FEM solutions, showing solution profiles, error distributions, and 3D visualizations of both approaches.

Method	Max Error	Mean Error	Parameters
PINN	0.026427	0.006583	1341
FEM	0.017638	0.003102	100

Table 1: Quantitative comparison between PINN and FEM solutions.

The FEM solution achieved slightly better accuracy with fewer parameters for the forward problem with constant diffusion. However, for the inverse problem, the PINN approach offered significant advantages:

- **Parameter recovery**: PINNs directly recovered the unknown diffusion coefficient
- **Continuous solution**: PINNs provided a fully differentiable solution across the entire domain
- **Sparse data handling**: PINNs effectively utilized limited measurement data

4 Discussion

4.1 Advantages and Limitations of PINNs

Advantages:

1. **Physical law incorporation**: By encoding the PDE directly in the loss function, PINNs ensure that solutions satisfy the underlying physics
2. **Inverse problem capability**: PINNs naturally handle inverse problems by simultaneously optimizing for unknown parameters and solution fields
3. **Mesh-free approach**: No need for structured meshes, allowing application to complex geometries
4. **Continuous, differentiable solutions**: Solutions are continuously differentiable across the entire domain

Limitations:

1. **Computational cost**: Training neural networks is more computationally intensive than traditional methods for forward problems
2. **Hyperparameter sensitivity**: Performance depends on careful selection of network architecture and training parameters
3. **Loss landscape challenges**: Multiple competing loss terms can create difficult optimization landscapes

4.2 Comparison with Traditional Methods

For the simple heat equation with constant diffusion coefficient, FEM provides more efficient and slightly more accurate solutions. However, PINNs offer distinct advantages for more complex scenarios:

1. **Inverse problems**: Traditional methods typically require iterative optimization loops to solve inverse problems, while PINNs handle them directly
2. **Complex geometries**: PINNs can easily adapt to irregular domains without specialized meshing
3. **Unknown physics**: PINNs can simultaneously discover governing equations and their solutions

The quantitative comparison shows:

```
Comparison Summary:
-------------------------------------------------
Method     | Max Error       | Mean Error      | Parameters
-------------------------------------------------
PINN       | 0.026427        | 0.006583        | 1341
FEM        | 0.017638        | 0.003102        | 100
-------------------------------------------------
```

4.3 Practical Considerations

Implementation challenges for PINNs include:

1. **Balancing loss terms**: Proper weighting of different loss components is crucial
2. **Training stability**: Managing gradient-based optimization for composite loss functions
3. **Computational resources**: Neural network training can be resource-intensive

The training logs from our implementation show the progression of the optimization process:

```
Epoch 100/5000, Loss: 0.227991, IC Loss: 0.148057, BC Loss: 0.069497, PDE Loss: 0.010437, Ti
Epoch 500/5000, Loss: 0.013552, IC Loss: 0.005905, BC Loss: 0.002132, PDE Loss: 0.005514, Ti
Epoch 1000/5000, Loss: 0.001901, IC Loss: 0.000242, BC Loss: 0.000384, PDE Loss: 0.001274, T
Epoch 5000/5000, Loss: 0.000304, IC Loss: 0.000003, BC Loss: 0.000097, PDE Loss: 0.000203, T
```

For the inverse problem:

```
Epoch 100/2000, Loss: 0.487283, IC Loss: 0.219074, BC Loss: 0.019189, PDE Loss: 0.055562, Da
Epoch 500/2000, Loss: 0.026396, IC Loss: 0.005250, BC Loss: 0.001346, PDE Loss: 0.011089, Da
Epoch 1000/2000, Loss: 0.004327, IC Loss: 0.000415, BC Loss: 0.000272, PDE Loss: 0.002745, D
Epoch 2000/2000, Loss: 0.001467, IC Loss: 0.000142, BC Loss: 0.000107, PDE Loss: 0.001036, D
```

5 Conclusion

This project demonstrated the successful application of Physics-Informed Neural Networks to both forward and inverse heat equation problems. The results show that while traditional FEM is slightly more accurate for forward problems with constant coefficients, PINNs excel in inverse problems where unknown parameters need to be recovered from sparse data.

The ability to incorporate physical laws directly into the learning process makes PINNs a powerful tool for scientific machine learning, especially in scenarios with limited data or unknown parameters. Future work could explore extensions to more complex PDEs, higher dimensions, or non-linear problems.

References

[1] Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 686-707.