PINN's FOR INVERSE HEAT TRANSFER PROBLEMS

By Logan Levine, Cole McCaleb, Thomas Pelosi

Table of contents

01

Methodology

What are IHTP’s?

- **Heat Transfer Problems (HTPs)** involve determining temperature data based on unknown thermal parameters such as thermal conductivity, heat flux, or boundary conditions
- Unlike normal HTPs, where inputs (ex. heat sources) are known and outputs (ex. temperature distribution) are calculated, Inverse HTPs (IHTPs) work in reverse using temperature measurements to infer the underlying causes.
- In IHTPs, small errors or noise in temperature measurements can also lead to significant inaccuracies in the estimated parameters, making the problem sensitive and unstable.

Real-World Applications

- **Cryosurgery:** Analyzing the thermal fields during cryosurgery to ensure accurate and effective tissue destruction.
- **Electronics Cooling**: Determining heat fluxes to design effective cooling systems.
- **Aerospace Engineering**: Estimating the heat flux on a spacecraft's heat shield during atmospheric re-entry
- **Material Science**: Evaluating thermal properties of new materials under varying conditions.

Why Use Physics-Informed Neural Networks?

Limitations of Traditional Inverse Methods

- **Tikhonov regularization** and **Conjugate Gradient** methods are common, but:
    - Require **manual tuning** of regularization parameters
    - Are **highly sensitive to noise**, especially with **limited or sparse data**
    - **Struggle in complex or high-dimensional geometries**

Why PINNs Are Better Suited

- **Embed physical laws** (e.g., the heat equation) directly into training
- Naturally handle **sparse, noisy, or indirect data**
- Improve **robustness** and **generalization** without manual regularization
- Enable **data efficient learning** by enforcing physical consistency

Governing Equations

- **1D Steady-State**: d/dx(k(x)du/dx)=0

-Solves for **spatially varying** conductivity k(x)
-k(x) is modeled as a **differentiable function**, not via a neural net, but recovered implicitly via PDE residuals and smoothness regularization
-Objective: infer k(x) and u(x) from sparse temperature measurements

- **1D Transient**: Ut=k*Uxx

-Time-dependent temperature evolution
-Scalar k is treated as a **trainable parameter** (learned via gradient descent)
-PINN jointly predicts u(x,t) and infers the best-fit k

- **2D Steady-State**: $\nabla^2 U$

-Solves for equilibrium temperature distribution over a 2D plate
-**Assumes a fixed, uniform k** in PDE form, but used as context (not estimated)

- **2D Transient**: $k*\nabla^2 U$

-Most complex case: full space-time PINN
-Scalar k is treated as a **trainable parameter**, estimated via optimization
-Uses dynamic loss balancing to help prioritize data fitting early, then physics enforcement

PINN ARCHITECTURE

Neural Network Structure

- **Model Type**: Fully Connected Feedforward Neural Network
- **Layers**: 2-4 hidden layers
- **Neurons**: 50–128 per layer
- **Optimizer:** Adam
- **Activation**: Tanh (smooth, differentiable for PDEs)
- **Input**:
    - 1D: x, or x,t
    - 2D: x,y, or x,y,t
- **Output**: Temperature u and Inferred thermal conductivity k

Purpose:

- Use physics + observed temperatures to recover **unknown conductivity k**
- **u** is predicted to enforce the PDE; **k** is the inverse target

Total Loss Function

Total Loss = Data Loss + Physics Loss + Regularization + Boundary Loss

Loss Components

- **Data Loss:**

  Mean Squared Error (MSE) between predicted and observed temperatures
  Ensures the model fits the available measurements
- **Physics Loss:**
  Enforces the heat equation using autograd

  Example: ut - k * u_xx or Laplace residual in 2D
- **Regularization**
  Smoothness penalty on k(x) in 1D steady case
  Prior-based penalty on log(k) in 2D transient model
- **Boundary Loss**
  Penalty for violating boundary or anchor conditions
  Especially important in transient models

Loss Weighting Strategy

- Most models use static weights
- 2D transient model uses dynamic weighting:
  Start with physics loss scaled down (0.1x)
  Increase to 5.0x after 1000 epochs to emphasize physics

02

Results and Analysis

Experiment Setup

Setup:

- **PINN 1D Setup**:
  - The initial temperature is given by an analytical solution:
    - u(x,t)=sin(πx)e^(−kπ2t)
  - Training points sampled randomly in (x,t)
  - Goal: Recover scalar thermal conductivity *k*

- **PINN 2D Setup**:
  - The top edge is set to 100°C
  - The other edges are set to 0°C.
  - Goal: Recover uniform (constant) k.

Experiment Results & Analysis

Case	True k	Predicted k̂	Error
1D (Steady)	0.10	0.0999	0.1%
1D (Transient)	0.10	~0.0999	0.1%
2D (Steady)	0.50	~0.1640	67.2%
2D (Transient)	0.50	~0.5999	19.98%

1D Steady: from PINN_1D_Inverse_Heat.ipynb, very low error

1D Transient: PINN_1D_Inverse_Heat_Time.ipynb, predicts ~0.0999 from 0.10

2D Steady: 2D_PINN.ipynb (first part), gets stuck around 0.1640

2D Transient: 2D_PINN.ipynb (second part), overshoots to ~0.5999 (true = 0.50)

Additional Experiments & Analysis

We analyzed logs from three PINN implementations to evaluate consistency and performance across different formulations and geometries.

- PINN_1D_Inverse_Heat
- PINN_1D_Inverse_Heat_Time
- 2D_PINN

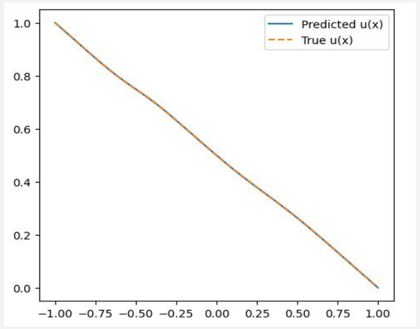
PINN_1D_Inverse_Heat

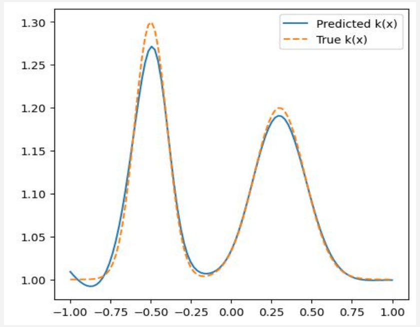
Experiment 1 - PINN_1D_Inverse_Heat:

- **Training**: 10000 epochs
- **Total Loss**: ↓ from 1.23 → 4.76e-04 (0.000476)
- **Physics Loss**: ↓ from 4.08e-04 → 6.02e-06 (0.00000602)
- **Thermal Conductivity Error**: ~7.58e-05 (0.0000758)

Takeaways:

- Excellent fit to the governing PDE
- High accuracy in estimating conductivity
- Smoothing loss increased → highlights need for *regularization balancing*





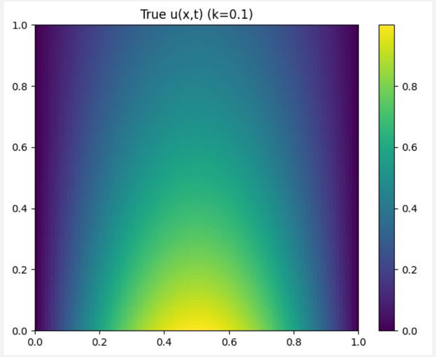
PINN_1D_Inverse_Heat_Time

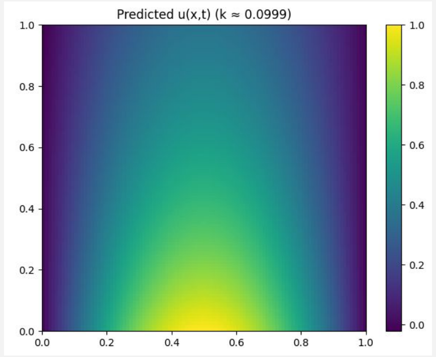
Experiment 2 - PINN_1D_Inverse_Heat_Time:

- **Training**: 5000 epochs
- **True k:** 0.10
- **Initial Inferred k:** 0.0510
- **Final Inferred k:** 0.0999
- **Final Loss:** 7.27e-06 (0.00000727)

Takeaways:

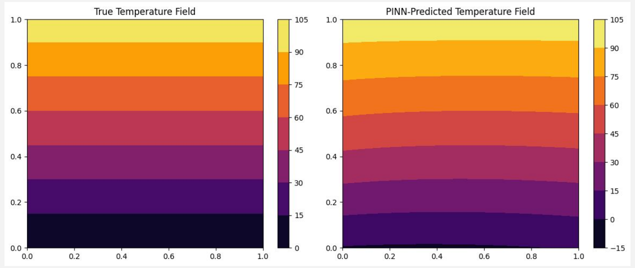
- Smooth, stable convergence
- Sub-percent error despite noisy synthetic data
- Shows PINNs' *robustness over time* and ability to infer evolving parameters



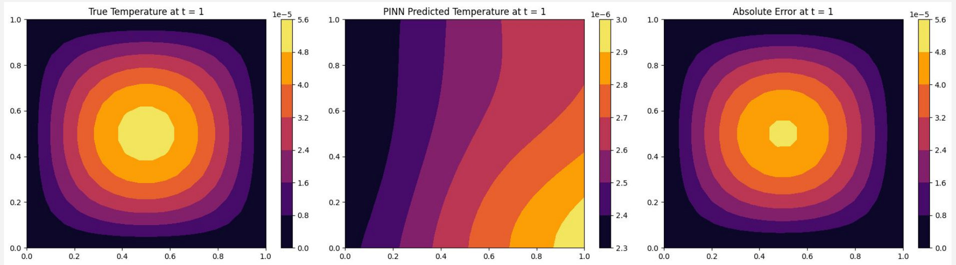


2D_PINN

Steady



Transient



Experiment 3 - 2D_PINN:

- **Steady:**
  - **Training:** 5000 epochs
  - **True k:** 0.5
  - **Initial Inferred k:** 1.0
  - **Final Inferred k:** 0.1640 → stuck in local minimum
- **Transient:**
  - **Training:** 5000 epochs
  - **True k:** 0.5
  - **Initial Inferred k:** 0.5
  - **Final Inferred k:** 0.5914

Takeaways:

- 2D PINNs can recover parameters, but…
  - Higher Dimensionality = Higher Complexity
  - Sensitive to *Initialization* & *Gradient Scaling*

Future Work

- **Limitations:**
  - Performance degrades with extreme noise/sparse data.
  - Simultaneous estimation of multiple parameters (e.g., k and heat capacity) is difficult.
  - Training time scales with dimensionality.

- **Proposed Improvements:**
  - Adaptive Collocation Sampling
  - Transfer Learning
  - Uncertainty Quantification
  - Advanced Training Techniques

Conclusion

Physics-Informed Neural Networks (PINNs) effectively solve inverse heat transfer problems by combining data with governing physical laws.

Models accurately recovered thermal conductivity and full temperature fields across:

- 1D steady & transient problems
- 2D steady & transient setups

Dynamic loss balancing helped improve training stability, especially in complex 2D cases.

Limitations

- Performance drops with extreme noise or very sparse data
- Learning multiple parameters simultaneously remains challenging
- Training time increases significantly in higher dimensions