Thermal Management in Microprocessors using PINNs

Zentiq.AI SciML Challenge

Achieved **Rank 16** with **RMSE = 0.329**
Challenge Link: Zentiq.AI SciML Challenge

Problem Statement

The aim of this challenge is to leverage neural network-based partial differential equation solvers to predict and prevent thermal damage in next-generation microprocessors before it occurs.

Geometry and Physical Conditions

A unit square domain $[0,1] \times [0,1]$ models a 1 cm $\times$ 1 cm processor die.

- Primary cooling fan: $b_y = 3$ cm/s (y-direction)
- Secondary cooling fan: $b_x = 2$ cm/s (x-direction)
- Thermal diffusion coefficient: $\varepsilon = 10^{-4}$

Governing Equation

The convection-diffusion equation:

$$-\varepsilon \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + b_x \frac{\partial u}{\partial x} + b_y \frac{\partial u}{\partial y} = f(x,y)$$

Heat Generation Function

$$\begin{aligned} f(x,y) = {} & 2\varepsilon \left(-x + \exp\left(\frac{2(x-1)}{\varepsilon} \right) \right) + xy^2 + 6xy \\ & - x \exp\left(\frac{3(y-1)}{\varepsilon} \right) - y^2 \exp\left(\frac{2(x-1)}{\varepsilon} \right) + 2y^2 \\ & - 6y \exp\left(\frac{2(x-1)}{\varepsilon} \right) - 2 \exp\left(\frac{3(y-1)}{\varepsilon} \right) + \exp\left(\frac{2x+3y-5}{\varepsilon} \right) \end{aligned}$$

Boundary Conditions

Dirichlet boundary condition:

$$u(x,y) = 0 \quad \text{for} \quad (x,y) \in \partial\Omega$$

Evaluation Metric

$$l_2 = \sqrt{\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \left(u_{\text{pred}}(x_i, y_i) - u_{\text{true}}(x_i, y_i)\right)^2}$$

PINN Implementation Code (Python)

We solve a two-dimensional convection-diffusion partial differential equation (PDE) over the unit square domain $\Omega = (0,1) \times (0,1)$. The PDE is given by:

$$-\epsilon \Delta u + \mathbf{b} \cdot \nabla u = f(x,y), \quad (x,y) \in \Omega, \tag{1}$$

where:

- $u(x,y)$: the unknown solution,
- $\epsilon = 10^{-4}$: diffusion coefficient,
- $\Delta u = u_{xx} + u_{yy}$: Laplacian,
- $\mathbf{b} = (b_x, b_y) = (2.0, 3.0)$: convection velocity vector,
- $\nabla u = (u_x, u_y)$: gradient,
- $f(x,y)$: forcing function.

Expanding the terms, the PDE becomes:

$$-\epsilon(u_{xx} + u_{yy}) + b_x u_x + b_y u_y = f(x,y). \tag{2}$$

The forcing function is defined as:

$$f(x,y) = 2e^{-x+e^{2(x-1)/\epsilon}} + xy^2 + 6xy - xe^{3(y-1)/\epsilon} - y^2 e^{2(x-1)/\epsilon} + 2y^2. \tag{3}$$

0.1 Boundary Conditions

The PDE is supplemented with homogeneous Dirichlet boundary conditions on the boundary $\partial\Omega$:

$$u(x,y) = 0, \quad (x,y) \in \partial\Omega, \tag{4}$$

where $\partial\Omega$ consists of the edges $x = 0$, $x = 1$, $y = 0$, and $y = 1$.

0.2 Domain

The computational domain is:

$$\Omega = (0,1) \times (0,1), \tag{5}$$

with boundary:

$$\partial\Omega = \{(x,y) \mid x = 0, 0 \le y \le 1\} \cup \{x = 1, 0 \le y \le 1\} \cup \{y = 0, 0 \le x \le 1\} \cup \{y = 1, 0 \le x \le 1\}. \tag{6}$$

1 Physics-Informed Neural Network (PINN) Approach

A physics-informed neural network (PINN) is used to approximate the solution $u(x,y)$. The PINN is a neural network $\hat{u}(x,y;\theta)$, parameterized by weights and biases θ, trained to satisfy the PDE and boundary conditions.

1.1 Neural Network Architecture

The neural network is a fully connected feedforward network with:

- Input layer: 2 neurons (for coordinates (x, y)),
- Hidden layers: 4 layers with 64 neurons each,
- Output layer: 1 neuron (for $\hat{u}$),
- Activation function: tanh,
- Initialization: Xavier normal for weights, zero for biases.

The network maps:

$$\hat{u} : \mathbb{R}^2 \to \mathbb{R}, \quad (x, y) \mapsto \hat{u}(x, y; \theta). \tag{7}$$

1.2 Loss Function

The PINN is trained by minimizing a composite loss function:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{PDE}}(\theta) + 10\mathcal{L}_{\text{BC}}(\theta), \tag{8}$$

where:

- **PDE Loss**: Enforces the PDE at interior collocation points:

$$\mathcal{L}_{\text{PDE}}(\theta) = \frac{1}{N_{\text{collo}}} \sum_{i=1}^{N_{\text{collo}}} \left| -\epsilon(\hat{u}_{xx} + \hat{u}_{yy}) + b_x \hat{u}_x + b_y \hat{u}_y - f(x_i, y_i) \right|^2, \tag{9}$$

  with $N_{\text{collo}} = 2000$ points $\{(x_i, y_i)\}$ sampled randomly in Ω.

- **Boundary Loss**: Enforces the boundary condition:

$$\mathcal{L}_{\text{BC}}(\theta) = \frac{1}{N_{\text{BC}}} \sum_{i=1}^{N_{\text{BC}}} \left| \hat{u}(x_i, y_i; \theta) \right|^2, \tag{10}$$

  with $N_{\text{BC}} = 200$ points sampled on $\partial\Omega$, approximately 50 points per edge.

1.3 Sampling Strategy

- **Interior Points**: $N_{\text{collo}} = 2000$ points sampled uniformly in $(0, 1) \times (0, 1)$ at each epoch.
- **Boundary Points**: $N_{\text{BC}} = 200$ points, with $N_{\text{BC}}/4 \approx 50$ points per edge, sampled uniformly.

1.4 Optimization

The parameters θ are optimized using the Adam optimizer with learning rate $\eta = 10^{-3}$. Training runs for $N_{\text{epochs}} = 30,000$ iterations, with gradients computed via backpropagation and automatic differentiation.

2 Training Process

The training loop:

1. Samples N_{collo} interior points and computes $\mathcal{L}_{\text{PDE}}$.
2. Samples N_{BC} boundary points and computes $\mathcal{L}_{\text{BC}}$.
3. Combines losses: $\mathcal{L} = \mathcal{L}_{\text{PDE}} + 10\mathcal{L}_{\text{BC}}$.
4. Updates θ using Adam.
5. Monitors losses every 2000 epochs.

3 Prediction

After training, the PINN predicts $\hat{u}(x, y; \theta)$ at test points $\{(x_i, y_i)\}$, typically from a file (e.g., `test.csv`), for generating a submission file with columns ID, x, y, u.

4 Mathematical Insights

- The small $\epsilon = 10^{-4}$ makes the PDE convection-dominated, leading to sharp boundary layers near $x = 1$ and $y = 1$.
- PINNs avoid meshing, but small ϵ challenges gradient accuracy.
- The boundary loss weight (10) emphasizes boundary conditions, requiring tuning.
- Random collocation sampling introduces stochasticity, aiding generalization but potentially slowing convergence.