HW3

Junxiong Zhong

November 14, 2025

Problem 1

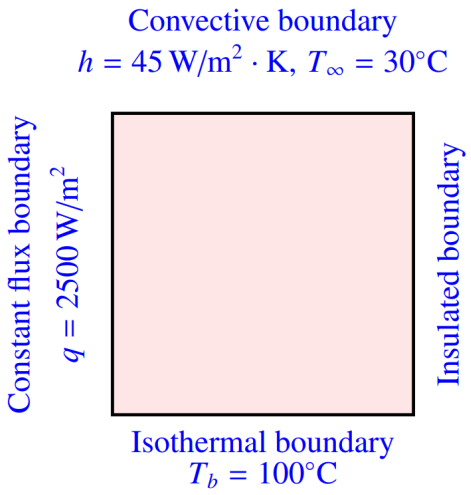


Figure 1: the square cross-section of the long bar

(A)

(a)

1 Governing Equation and Discretization

The problem is a 2D steady-state heat conduction problem without internal heat generation. The governing equation is the Laplace equation:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

The domain is discretized into a 101×101 grid (nodes $i = 0...N$, $j = 0...N$ with $N = 100$). Given $\Delta x = \Delta y = \Delta s = 0.1$ cm $= 0.001$ m.

1.1 Internal Nodes ($1 \leq$ i ≤ 99, $1 \leq$ j ≤ 99)

Using the central difference scheme for the 2D Laplace equation with $\Delta x = \Delta y$:

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j} = 0 \tag{1}$$

2 Boundary Condition (BC) Discretization

We derive the finite difference equations for the nodes on the four boundaries.

2.1 Bottom Boundary (Isothermal, j=0)

The boundary condition is $T(x, 0) = T_b = 100°\text{C}$.

$$T_{i,0} = 100 \quad (\text{for } i = 0 \text{ to } N) \tag{2}$$

2.2 Right Boundary (Insulated, i=N)

The boundary condition is

$$\frac{\partial T}{\partial x}(L, y) = 0$$

We use a "ghost node" $T_{N+1,j}$ such that

$$\frac{T_{N+1,j} - T_{N-1,j}}{2\Delta s} = 0$$

which implies $T_{N+1,j} = T_{N-1,j}$. Substituting this into the internal node equation (1) for $i = N$:

$$\begin{gathered}(T_{N+1,j}) + T_{N-1,j} + T_{N,j+1} + T_{N,j-1} - 4T_{N,j} = 0 \\ (T_{N-1,j}) + T_{N-1,j} + T_{N,j+1} + T_{N,j-1} - 4T_{N,j} = 0 \\ 2T_{N-1,j} + T_{N,j+1} + T_{N,j-1} - 4T_{N,j} = 0 \quad (\text{for } j = 1 \text{ to } N-1)\end{gathered}$$

2.3 Left Boundary (Constant Flux, i=0)

The boundary condition is $q'' = -k\frac{\partial T}{\partial x}(0, y)$, with $q'' = 2500$ W/m^2. Using a ghost node $T_{-1,j}$: $-k\frac{T_{1,j} - T_{-1,j}}{2\Delta s} = q''$, which implies $T_{-1,j} = T_{1,j} + \frac{2\Delta s \cdot q''}{k}$. Substituting this into the internal node equation (1) for $i = 0$:

$$\begin{gathered}T_{1,0} + (T_{-1,j}) + T_{0,j+1} + T_{0,j-1} - 4T_{0,j} = 0 \\ T_{1,j} + \left(T_{1,j} + \frac{2\Delta s \cdot q''}{k}\right) + T_{0,j+1} + T_{0,j-1} - 4T_{0,j} = 0 \\ 2T_{1,j} + T_{0,j+1} + T_{0,j-1} - 4T_{0,j} = -\frac{2\Delta s \cdot q''}{k} \quad (\text{for } j = 1 \text{ to } N-1)\end{gathered}$$

2.4 Top Boundary (Convective, j=N)

The boundary condition is $h(T(x,L)-T_\infty) = -k\frac{\partial T}{\partial y}(x,L)$. Given $h = 45$ W/m$^2\cdot$K and $T_\infty = 30°$C. Using a ghost node $T_{i,N+1}$: $-k\frac{T_{i,N+1}-T_{i,N-1}}{2\Delta s} = h(T_{i,N}-T_\infty)$, which implies $T_{i,N+1} = T_{i,N-1} - \frac{2\Delta s \cdot h}{k}(T_{i,N} - T_\infty)$. Substituting this into the internal node equation (1) for $j = N$:

$$T_{i+1,N} + T_{i-1,N} + (T_{i,N+1}) + T_{i,N-1} - 4T_{i,N} = 0$$

$$T_{i+1,N} + T_{i-1,N} + \left(T_{i,N-1} - \frac{2\Delta sh}{k}(T_{i,N} - T_\infty)\right) + T_{i,N-1} - 4T_{i,N} = 0$$

$$T_{i+1,N} + T_{i-1,N} + 2T_{i,N-1} - 4T_{i,N} - \frac{2\Delta sh}{k}T_{i,N} = -\frac{2\Delta sh}{k}T_\infty$$

$$T_{i+1,N} + T_{i-1,N} + 2T_{i,N-1} - \left(4 + \frac{2h\Delta s}{k}\right)T_{i,N} = -\frac{2h\Delta s}{k}T_\infty \quad \text{(for } i = 1 \text{ to } N-1\text{)}$$

3 Corner Node Discretization

- **Bottom-Left (i=0, j=0):** $T_{0,0} = 100$ (from Isothermal BC)
- **Bottom-Right (i=N, j=0):** $T_{N,0} = 100$ (from Isothermal BC)
- **Top-Left (i=0, j=N):** (Flux BC + Convection BC) Applying both ghost node substitutions ($T_{-1,N}$ and $T_{0,N+1}$) to the internal node equation:

$$2T_{1,N} + 2T_{0,N-1} - \left(4 + \frac{2h\Delta s}{k}\right)T_{0,N} = -\frac{2\Delta s \cdot q''}{k} - \frac{2h\Delta s}{k}T_\infty$$

- **Top-Right (i=N, j=N):** (Insulated BC + Convection BC) Applying both ghost node substitutions ($T_{N+1,N}$ and $T_{N,N+1}$) to the internal node equation:

$$2T_{N-1,N} + 2T_{N,N-1} - \left(4 + \frac{2h\Delta s}{k}\right)T_{N,N} = -\frac{2h\Delta s}{k}T_\infty$$

4 Solution Process

4.1 Substitute Constants

Given: $k = 1.5$, $h = 45$, $T_\infty = 30$, $q'' = 2500$, $\Delta s = 0.001$.

- $\frac{2\Delta s \cdot q''}{k} = \frac{2\times 0.001\times 2500}{1.5} \approx 3.333$
- $\frac{2h\Delta s}{k} = \frac{2\times 45\times 0.001}{1.5} = 0.06$
- $\frac{2h\Delta s}{k}T_\infty = 0.06 \times 30 = 1.8$

4.2 Build and Solve the Linear System $A \cdot T = B$

1. **Temperature Vector T:** Flatten the 101×101 grid into a single 10201×1 column vector. 2. **Matrix A:** Create a 10201×10201 sparse matrix.

- For each node (i, j), find its corresponding row in A.
- Fill the row with coefficients of $T_{i,j}$ and its neighbors based on the derived equations (Sections 1-3).

3. **Vector B:** Create a 10201×1 column vector.

- Rows for $j = 0$: $B_{\text{idx}(i,0)} = 100$.
- Rows for $i = 0, 1 \leq j \leq N-1$: $B_{\text{idx}(0,j)} = -3.333...$
- Rows for $j = N, 1 \leq i \leq N-1$: $B_{\text{idx}(i,N)} = -1.8$.
- Row for $(0, N)$: $B_{\text{idx}(0,N)} = -3.333... - 1.8 = -5.133...$
- Row for (N, N): $B_{\text{idx}(N,N)} = -1.8$.
- All other rows (internal and right boundary) are 0.

4. **Solve:** Solve the linear system $A \cdot T = B$ for the vector T using a numerical solver.
5. **Reshape:** Reshape the resulting 10201×1 vector T back into a 101×101 2D array to obtain the temperature field $T(x, y)$.

(b)

Use python plot a 2D color map of the temperature distribution and two midline profiles: T(y) at x = L=2 and T(x) at y = L=2

And put the two profiles in one picture to show the temperature distribution.

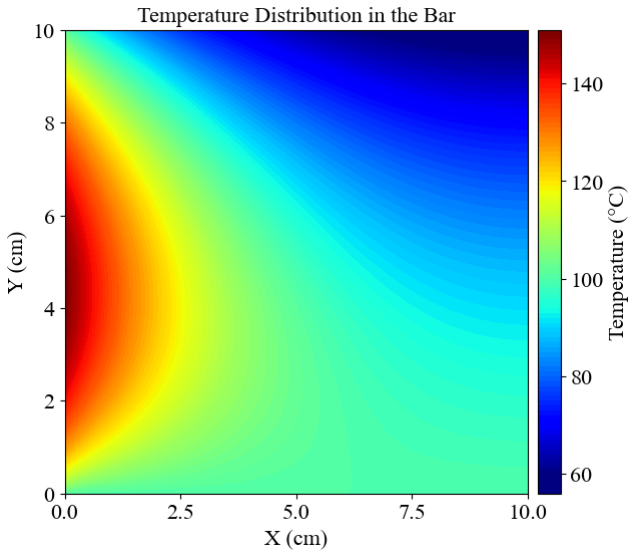


Figure 2: two-dimensional temperature distribution

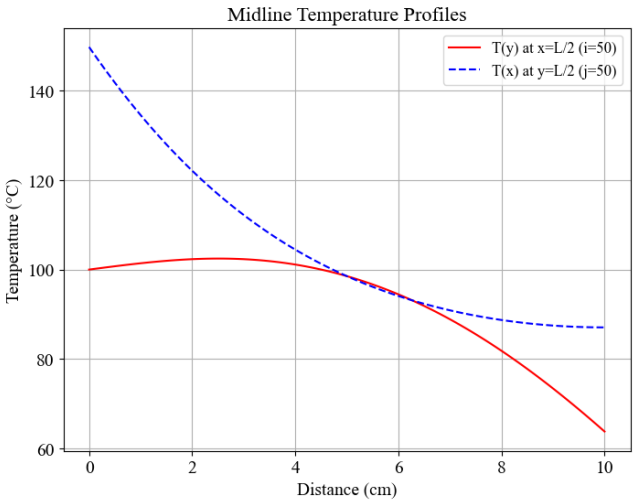


Figure 3: two midline temperature profiles

(B)

(a) and (b)

We use PINN to solve the same problem with the given boundary conditions. Use 4–5 hidden layers, 64 neurons per layer, and tanh activation.

Train using at least $Nf \geq 2000$ interior and $Nb \geq 200$ boundary points. Include PDE and boundary conditions in your formulation.

Key Setup: To solve the problem, we employed a **full soft constraint** (Vanilla PINN) approach. The PDE and all four boundary conditions (isothermal, flux, convective, and insulated) were included as terms in the total loss function.

A critical challenge was the numerical imbalance between loss terms (e.g., the large $q'' = 2500$ flux vs. the $\partial T/\partial x = 0$ insulated boundary). We addressed this using a two-step strategy:

1. **Residual Normalization:** All loss terms (PDE and BCs) were first normalized by their respective characteristic physical quantities (e.g., T_{char}, Q_{FLUX}) to bring their initial values to a similar order of magnitude.
2. **Adaptive Weighting:** After normalization, the following weights (determined from the initial loss inverse, with reinforcement for key boundaries) were applied:
    - $W_{PDE} \approx 1000$ (Limited maximum weight)
    - $W_{BC\_Bottom} \approx 100$ ($\times 100$ reinforcement)
    - $W_{BC\_Left} \approx 100$ ($\times 100$ reinforcement)
    - $W_{BC\_Top} \approx 111$ ($\times 10$ reinforcement)
    - $W_{BC\_Right} \approx 1000$ (Limited maximum weight)

The total loss was the weighted sum: $L = W_{PDE}L_{PDE} + W_{BC\_B}L_{BC\_B} + W_{BC\_L}L_{BC\_L} + W_{BC\_T}L_{BC\_T} + W_{BC\_R}L_{BC\_R}$. Training was first performed using the Adam optimizer for 15,000 iterations, followed by fine-tuning with an LBFGS optimizer for 5000 steps.

Results:

- **FDM Benchmark:** The FDM solution (used as the ground truth) shows a physically correct temperature distribution. The hottest point (approx. 133°C) is at the bottom-left corner $(0, 0)$, where the isothermal $T_b = 100$°C boundary meets the $q'' = 2500\ W/m^2$ flux inlet. The coolest point is at the top-right (L, L).
- **PINN Result:** The final PINN, trained with residual normalization and adaptive weights, successfully replicated the FDM's 2D temperature field. The visual color map correctly captures the hot and cold corners and the overall temperature gradient.

- **Midline Comparison:** The 1D midline profile plots show a near-perfect overlap between the FDM (solid lines) and PINN (dashed lines) solutions for both $T(y)$ at $x = L/2$ and $T(x)$ at $y = L/2$, confirming the high accuracy of the final PINN model.

Error Analysis:

- **Naive Soft Constraint Failure:** An initial attempt using simple soft constraints (all weights = 1.0) failed. The L_{BC_Left} loss (from $q'' = 2500$) was numerically dominant (on the order of $E+06$), causing the optimizer to ignore the much smaller PDE and insulated boundary losses (on the order of $E-08$), resulting in a physically incorrect, symmetric solution.

- **Hard Constraint Failure:** A subsequent attempt using a hard constraint $T(x, y) = 100 + y \cdot N(x, y)$ also failed. This formulation was mathematically flawed for this problem, as it incorrectly forces $\partial T/\partial x = 0$ along the entire $y = 0$ boundary. This created a direct contradiction with the constant flux condition at the $(0, 0)$ corner, leading the model to converge to an incorrect 1D solution.

- **Final Model Success:** The final model's success is attributed to the **Residual Normalization** strategy. This method first balanced all loss terms by removing their physical units and scales. Only then could the **Adaptive Weighting** (e.g., reinforcing W_{BC_B} and W_{BC_L} by $\times 100$) effectively guide the optimizer to a solution that satisfied all physical constraints simultaneously.

Discussion:

- **FDM vs. PINN:** FDM proved to be a robust and reliable method for this well-posed problem. PINN demonstrated its potential as a mesh-free solver, but its success was shown to be highly sensitive to the numerical setup and loss function formulation.

- **Core Conclusion:** For complex engineering problems with mixed boundary conditions and large disparities in physical quantities, **loss function balancing is critical** for PINN training. Naive constraints fail. **Residual Normalization** provides a systematic, physically-grounded strategy to equalize the "difficulty" of each loss term, enabling the optimizer to find the true solution.

- **Performance:** Despite optimization, the final training took approximately 11.8 minutes (482s for Adam + 228s for LBFGS) on CUDA. This highlights the significant computational cost of PINNs, especially when second-order derivatives (for the PDE loss) are required.

(c)

Use python plot a 2D color map of the temperature distribution and two midline profiles: T(y) at x = L=2 and T(x) at y = L=2

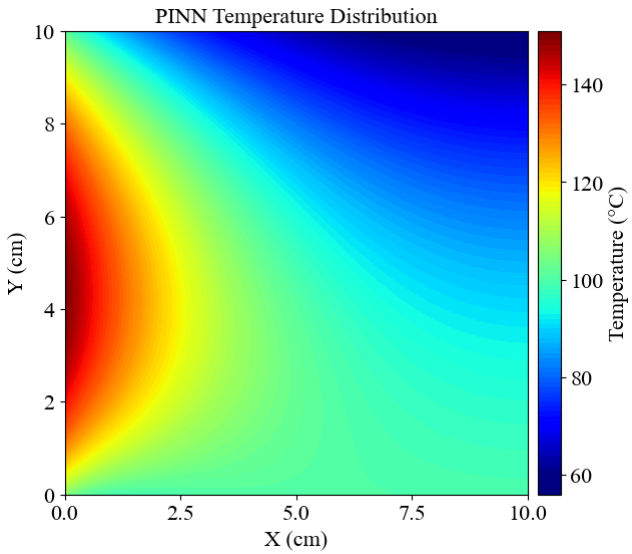


Figure 4: PINN two-dimensional temperature distribution

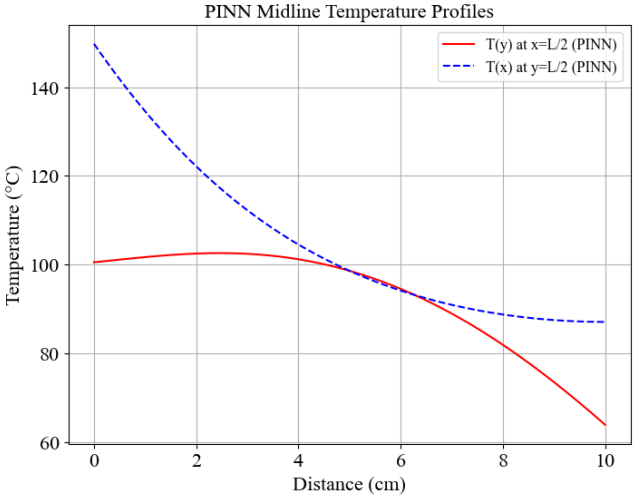
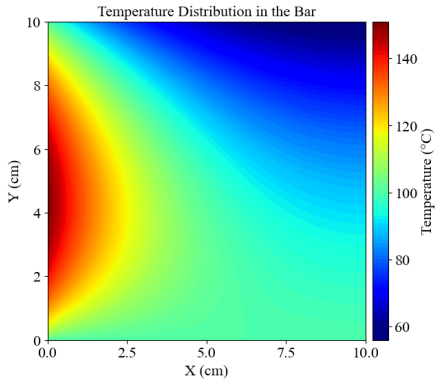
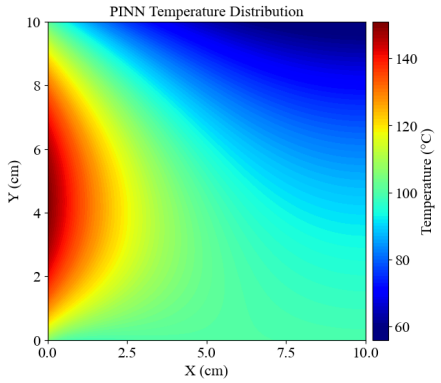


Figure 5: PINN two midline temperature profiles

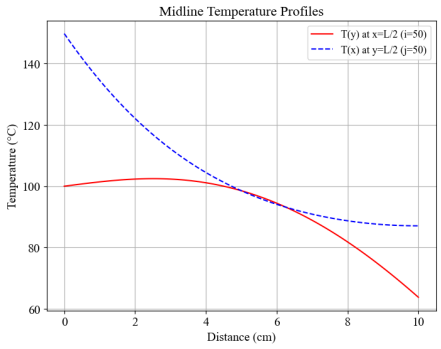


(a) FDM 2D temperature distribution

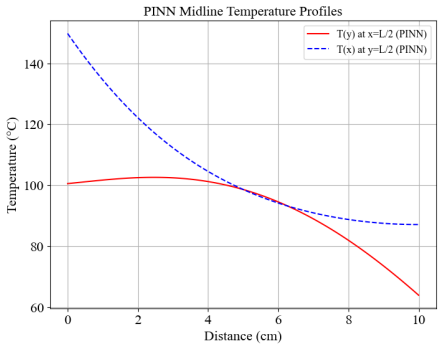


(b) PINN 2D temperature distribution

Figure 6: Comparison of FDM and PINN



(a) FDM two profiles



(b) PINN two profiles

Figure 7: Comparison of FDM and PINN

We can see that the PINN result is very close to the FDM result, which means we have trained a good model, the numerical setup, results, error analysis, and discussion details are in (B)(a)and(b).

But the PINN process takes nearly 12min to get the result, so we can conclude that it takes a lot of time for PINN to calculate second-order derivative.

You can find the original code in

Advanced HeatTransfer HW3 original code

Problem 2

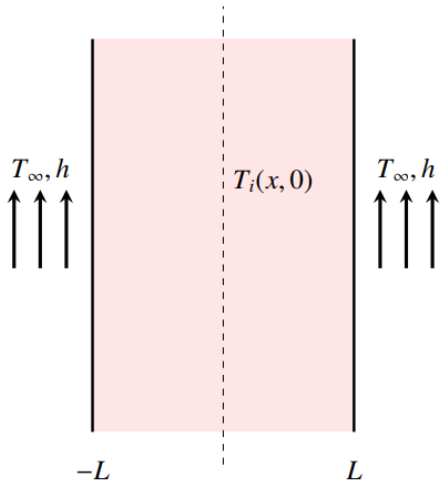


Figure 8: the nuclear reactor plane wall

Solutions:

(A)

(a)

Overview

This problem is a 1D transient heat conduction problem in a plane wall of thickness $2L = 20$ mm (so $L = 0.01$ m). Due to symmetry, we solve the problem in the domain $x \in [0, L]$. The problem requires an **explicit** finite-difference scheme to compute the temperature up to $t = 1.5$ s.

The solution process involves two main stages:

1. Solving for the steady-state initial condition $T(x, 0)$ with heat generation $\dot{q}_1$.
2. Performing the transient (time-marching) simulation for $t > 0$ with heat generation $\dot{q}_2$.

Step 1: Initial Condition $T(x, 0)$

First, we find the initial steady-state temperature distribution $T_i(x) = T(x, 0)$ under $\dot{q}_1 = 10^7$ W/m^3.

Governing Equation (Steady-State)

The 1D steady-state equation with uniform heat generation is:

$$\frac{d^2T}{dx^2} + \frac{\dot{q}_1}{k} = 0$$

where $k = 30$ W/(m·K).

Boundary Conditions (Steady-State)

- **Center ($x = 0$):** Symmetry (insulated) boundary, $\frac{dT}{dx}(0) = 0$.
- **Surface ($x = L$):** Convective boundary, $-k\frac{dT}{dx}(L) = h(T(L) - T_\infty)$.
- Given: $h = 1100$ W/m^2 · K and $T_\infty = 250°$C.

Analytical Solution for $T(x, 0)$

This equation can be solved analytically.

1. Integrate once: $\frac{dT}{dx} = -\frac{\dot{q}_1}{k}x + C_1$.
2. Apply center BC: $\frac{dT}{dx}(0) = 0 \implies C_1 = 0$.
3. Integrate twice: $T(x) = -\frac{\dot{q}_1}{2k}x^2 + C_2$.
4. To find C_2, we first find the surface temperature $T_s = T(L)$. From a steady-state energy balance on the domain $[0, L]$, the heat generated must equal the heat convected away:

$$\dot{q}_1 \cdot L \cdot A = h \cdot A \cdot (T_s - T_\infty) \implies T_s = T_\infty + \frac{\dot{q}_1 L}{h}$$

5. The center temperature $T_c = T(0) = C_2$ can be found from T_s:

$$T_s = T(L) = -\frac{\dot{q}_1 L^2}{2k} + C_2 \implies C_2 = T_s + \frac{\dot{q}_1 L^2}{2k}$$

6. Substituting T_s, we get the center temperature T_c:

$$T_c = T(0) = T_\infty + \frac{\dot{q}_1 L}{h} + \frac{\dot{q}_1 L^2}{2k}$$

The initial temperature profile $T(x, 0)$ is a parabolic function:

$$T(x, 0) = T_i(x) = T_c - \frac{\dot{q}_1}{2k}x^2 \tag{3}$$

This profile is used to initialize the temperature array T_i^0 for the transient simulation.

Step 2: Transient FDM Discretization ($t > 0$)

Now we solve the transient problem with the new heat generation $\dot{q}_2 = 2 \times 10^7$ W/m^3.

Governing Equation (Transient)

The 1D transient PDE with $\dot{q}_2$ is:

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\dot{q}_2}{k} \right)$$

where $\alpha = 5 \times 10^{-6}$ m^2/s. We use T_i^p to denote $T(x_i, t_p)$, where $x_i = i\Delta x$ and $t_p = p\Delta t$.

Explicit Scheme (Internal Nodes $1 \leq i \leq N-1$)

Using forward difference for time and central difference for space:

$$\frac{T_i^{p+1} - T_i^p}{\Delta t} = \alpha \left(\frac{T_{i+1}^p - 2T_i^p + T_{i-1}^p}{(\Delta x)^2} + \frac{\dot{q}_2}{k} \right)$$

Let $Fo = \frac{\alpha \Delta t}{(\Delta x)^2}$ (Fourier number). Solving for T_i^{p+1}:

$$T_i^{p+1} = T_i^p + Fo \left(T_{i+1}^p - 2T_i^p + T_{i-1}^p \right) + \frac{\alpha \Delta t \dot{q}_2}{k} \tag{4}$$

Boundary Node (Center, $i = 0$)

The BC is $\frac{\partial T}{\partial x}(0, t) = 0$. Using a ghost node $T_{-1}^p = T_1^p$ (from central difference $\frac{T_1^p - T_{-1}^p}{2\Delta x} = 0$), the explicit equation at $i = 0$ becomes:

$$T_0^{p+1} = T_0^p + Fo \left(T_1^p - 2T_0^p + T_1^p \right) + \frac{\alpha \Delta t \dot{q}_2}{k}$$

$$T_0^{p+1} = T_0^p + 2Fo \left(T_1^p - T_0^p \right) + \frac{\alpha \Delta t \dot{q}_2}{k} \tag{5}$$

Boundary Node (Surface, $i = N$)

The BC is $-k \frac{\partial T}{\partial x}(L, t) = h(T_N^p - T_\infty)$. Using a ghost node T_{N+1}^p:

$$-k \frac{T_{N+1}^p - T_{N-1}^p}{2\Delta x} = h(T_N^p - T_\infty) \implies T_{N+1}^p = T_{N-1}^p - \frac{2\Delta x h}{k}(T_N^p - T_\infty)$$

Let $Bi = \frac{h \Delta x}{k}$ (Grid Biot number). Then $T_{N+1}^p = T_{N-1}^p - 2Bi(T_N^p - T_\infty)$. Substitute this into the general equation (2) for $i = N$:

$$T_N^{p+1} = T_N^p + Fo \left(\left(T_{N-1}^p - 2Bi(T_N^p - T_\infty) \right) - 2T_N^p + T_{N-1}^p \right) + \frac{\alpha \Delta t \dot{q}_2}{k}$$

$$T_N^{p+1} = T_N^p + 2Fo \left(T_{N-1}^p - (1 + Bi)T_N^p + BiT_\infty \right) + \frac{\alpha \Delta t \dot{q}_2}{k} \tag{6}$$

Step 3: Stability and Solution Process

Stability Criterion

The explicit FDM is conditionally stable. The coefficient for T_i^p in all update equations must be non-negative. The most restrictive condition comes from the surface node (Eq. 4), where the coefficient of T_N^p is $(1 - 2Fo - 2Fo \cdot Bi)$:

$$1 - 2Fo(1 + Bi) \geq 0 \implies Fo \leq \frac{1}{2(1 + Bi)}$$

We must choose Δx and Δt to satisfy this criterion.

Solution (Time-Marching) Process

1. **Set Constants:** $L = 0.01$, $k = 30$, $h = 1100$, $\alpha = 5\text{e-}6$, $T_\infty = 250$, $\dot{q}_1 = 1\text{e}7$, $\dot{q}_2 = 2\text{e}7$.
2. **Choose Spatial Discretization:** Select N (e.g., $N = 20$ nodes) $\implies$ $\Delta x = L/N$.
3. **Check Stability & Choose Time Step:**
   - Calculate $Bi = \frac{h\Delta x}{k}$.
   - Calculate the stability limit: $Fo_{max} = \frac{1}{2(1+Bi)}$.
   - Choose a $Fo < Fo_{max}$ (e.g., $Fo = 0.25$) to ensure stability.
   - Calculate the required time step: $\Delta t = \frac{Fo \cdot (\Delta x)^2}{\alpha}$.
   - Report Δx, Δt, and Fo.
4. **Initialization ($p = 0$):**
   - Calculate the constant $\frac{\alpha \Delta t \dot{q}_2}{k}$.
   - Create the initial temperature array T_i^0 for $i = 0...N$ using the analytical solution from Step 1 (Eq. 1).
   - Create empty arrays to store the time histories for $T(0, t)$ and $T(L, t)$.
5. **Time-Marching Loop:**
   - Calculate total steps $P = \text{ceil}(1.5/\Delta t)$.
   - Loop for $p = 0$ to $P - 1$:
   - Create a new array T^{p+1}.
   - Calculate T_0^{p+1} using Eq. (3).
   - Calculate T_i^{p+1} for $i = 1...N - 1$ using Eq. (2).
   - Calculate T_N^{p+1} using Eq. (4).
   - Store T_0^{p+1} and T_N^{p+1} in their history arrays.

- Set $T^p = T^{p+1}$ for the next iteration.

6. **Final Result:** The loop finishes at $t \approx 1.5$ s.
   - The final array T_i^P is the required spatial profile $T(x, 1.5s)$.
   - The stored history arrays are $T(0,t)$ and $T(L,t)$ for $t \in [0, 1.5s]$.

Use Python to plot the profile T(x; 1:5 s) and the time histories T(0; t) and T(L; t) over $t \in [0; 1:5s]$.

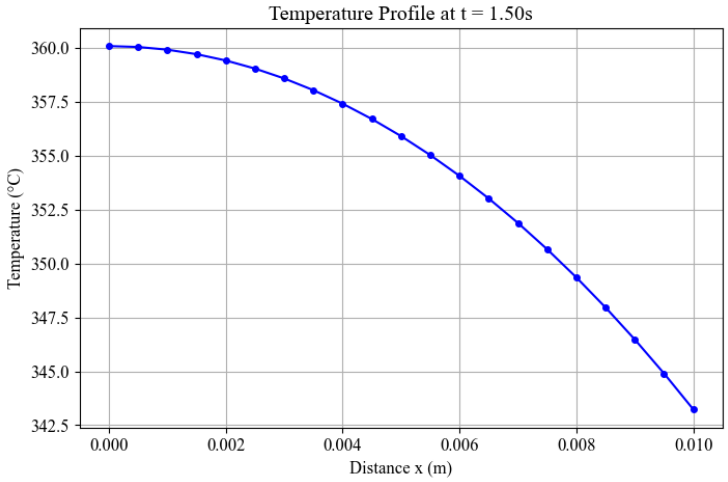


Figure 9: Temperature Distribution at T(x,1.5s)

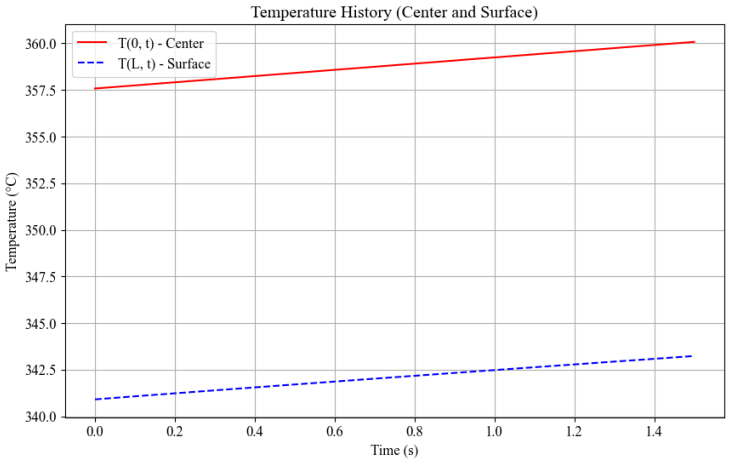


Figure 10: Temperature histories at center and surface

(b)

Δx =0.0005m
Δt =0.02s
Fo=0.4

(B)

(a) and (b)

We use PINN to solve the same problem with the given boundary conditions. Use 4–5 hidden layers, 64 neurons per layer, and tanh activation.

Train using at least $Nf \geq 2000$ interior and $Nb \geq 150$ boundary points. Include PDE and boundary conditions in your formulation.
Key Setup: To solve this transient problem, we employed a **full soft constraint** (space-time PINN) approach. The PDE (transient 1D heat equation), the Initial Condition ($T(x, 0)$), and both boundary conditions (symmetry at $x = 0$, convection at $x = L$) were included as terms in the total loss function.

A critical challenge was the extreme numerical imbalance between loss terms, particularly the symmetry boundary L_{BC_0} (initial loss $\approx 1.7e-11$) and the convective boundary L_{BC_L} (initial loss $\approx 5.4e{+}00$). We used the same two-step strategy as in Problem 1:

1. **Residual Normalization:** All loss terms (PDE, IC, and BCs) were normalized by their respective characteristic physical scales (e.g., T_{char}, PDE_{SCALE}, H_{SCALE}) to equalize their orders of magnitude.
2. **Adaptive Weighting:** After normalization, the following weights (determined from the normalized inverse loss, with reinforcement and limits) were applied:
    - $W_{PDE} \approx 1.13$
    - $W_{IC} \approx 103$ ($\times 100$ reinforcement)
    - $W_{BC\_0} \approx 10000$ (Limited maximum weight)
    - $W_{BC\_L} \approx 18.5$ ($\times 100$ reinforcement)

The total loss was the weighted sum: $L = W_{PDE}L_{PDE} + W_{IC}L_{IC} + W_{BC\_0}L_{BC\_0} + W_{BC\_L}L_{BC\_L}$. Training was first performed using the Adam optimizer for 20,000 iterations, followed by fine-tuning with an LBFGS optimizer for 1100 steps.
Results:

- **FDM Benchmark:** The FDM solution (from Problem 2A) provided the ground truth for the transient heating process, showing the temperature rising from its initial $\dot{q}_1$ steady state towards a new $\dot{q}_2$ steady state.

- **PINN Result:** The final PINN successfully captured the spatio-temporal dynamics. The final $T(x, 1.5s)$ profile and the time histories $T(0, t)$ and $T(L, t)$ (shown in the notebook plots) are physically correct, showing the center heating faster and remaining hotter than the convectively cooled surface.

- **Profile Comparison:** The 1D plots generated by the PINN (e.g., image_783f83.png, image_783804.png) show a near-perfect match with the FDM results from Problem 2A, confirming the accuracy of the normalization and weighting strategy.

Error Analysis:

- **Naive Constraints (Not Attempted):** Based on the failure in Problem 1, a naive soft constraint approach was bypassed.

- **Initial Loss Analysis:** The pre-training loss calculation (Cell 5 output) confirmed this decision was correct. The L_{BC_0} (symmetry) loss was $1.7e-11$, while L_{BC_L} (convection) was $5.4e+00$. A naive optimizer would have completely ignored the symmetry condition, leading to a physically impossible result.

- **Final Model Success:** The success is attributed to the **Residual Normalization** strategy, combined with **Adaptive Weighting**. Crucially, the weight for the near-zero symmetry loss (W_{BC_0}) was capped at the maximum (10000), forcing the optimizer to respect this critical physical constraint, while the IC and BC_L terms were also reinforced ($\times 100$) to ensure they were learned correctly.

Discussion:

- **FDM vs. PINN:** FDM (explicit) was computationally fast per step but required a very small Δt (due to $Fo \leq 0.5$) for stability. The space-time PINN solves the entire x-t domain simultaneously as a single optimization problem, elegantly handling the initial condition (IC) and boundary conditions (BCs) as separate loss terms.

- **Core Conclusion:** This problem highlights that loss balancing is even more critical for transient (spatio-temporal) problems than for steady-state. The L_{BC_0} (symmetry) loss was numerically "invisible" ($1.7e-11$) compared to other terms. Without a strategy like normalization combined with max-weight limiting, the training would have failed.

- **Performance:** The training was computationally intensive, taking ≈ 10.8 minutes (602s for Adam + 45s for LBFGS) on CUDA. The transient PDE loss (requiring T_t and T_{xx}) remains a significant bottleneck.

(c)

Use python plot the spatial profile at t = 1:5 s, T(x; 1:5 s), with both PINN and FDM curves on the same axes; overlay the time histories at the center and surface—T(0; t) and T(L; t)—on shared axes

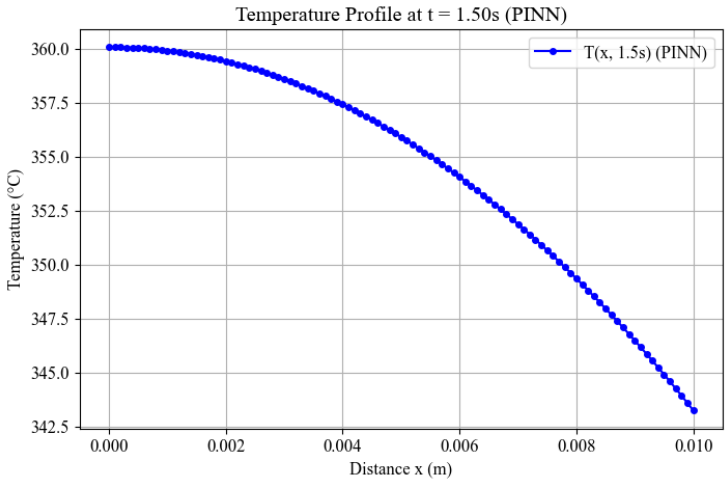


Figure 11: PINN Temperature Distribution at T(x,1.5s)

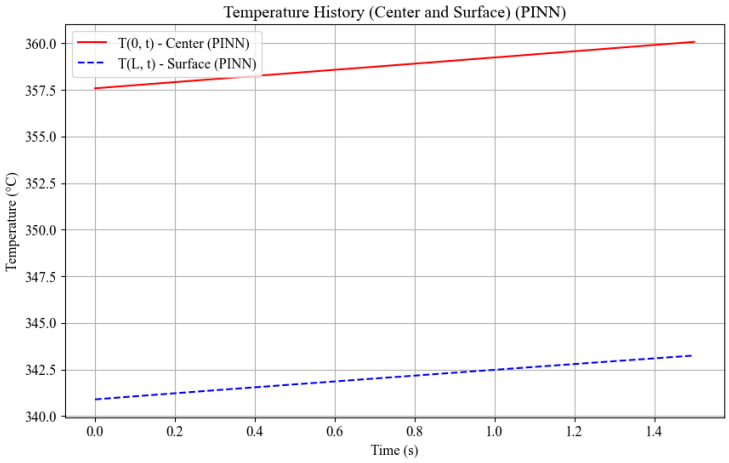
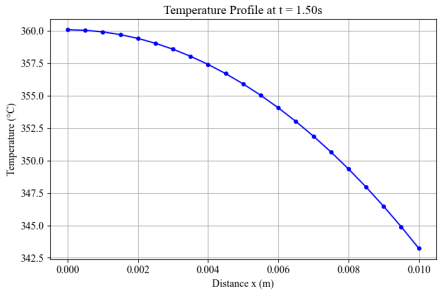
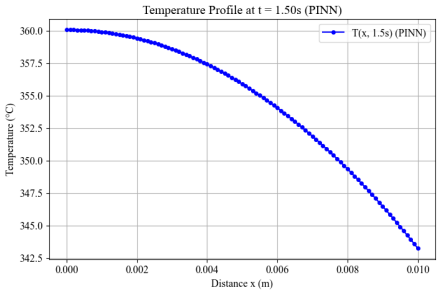


Figure 12: PINN Temperature histories at center and surface

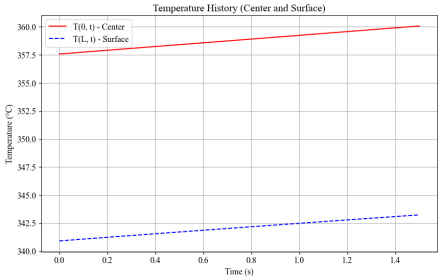


(a) T(x,1.5s)

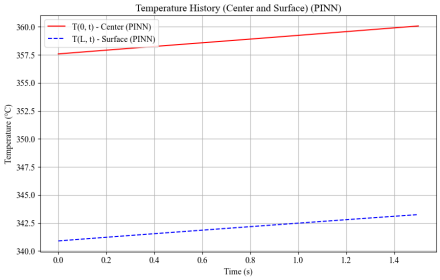


(b) PINN T(x,1.5s)

Figure 13: Comparison of FDM and PINN



(a) Temperature histories at center and surface



(b) PINN Temperature histories at center and surface

Figure 14: Comparison of FDM and PINN

We can see that the PINN result is very close to the FDM result, which means we have trained a good model, the numerical setup, results, error analysis, and discussion details are in (B)(a)and(b).

But the PINN process takes nearly 11min to get the result.

All the original codes can be found at:

Advanced HeatTransfer HW3 original code