DLCP 2026 · Machine Learning in Natural Sciences

A Physics-Informed Neural Network Framework for Multidimensional Heat Transfer

Three benchmark problems, one methodology: from a moving 1D pulse to curved 2D Robin cooling to a discontinuous 3D conductivity field

1D · Transient | 2D · Steady State | 3D · Transient

Kiura Stanley Matheka · Dmitry Efremenko · Fedor Buzaev
Russian State Social University · HSE University

10th International Conference on Deep Learning in Computational Physics

ROADMAP

What we'll cover

01

Why rethink the mesh?

The classical bottleneck PINNs are built to sidestep.

02

Anatomy of a PINN

How physics becomes a loss function via automatic differentiation.

03

Three benchmark problems, fully solved

1D moving source · 2D Robin-cooled hole · 3D discontinuous conductivity — each with FDM/FEM cross-validation.

04

Design principles that generalize — and where they don't

Activation choice, adaptive weighting, hard vs. soft constraints, warm-starts, two-stage training.

05

The verdict

Where each method wins, honestly, with numbers — and a practical advantages/disadvantages checklist.

MOTIVATION

The mesh is where the difficulty lives

Finite-difference, finite-volume, and finite-element solvers ties accuracy directly to grid resolution.

Computational cost grows rapidly in higher dimensions, or with irregular geometries, internal sources, and mixed boundary conditions.

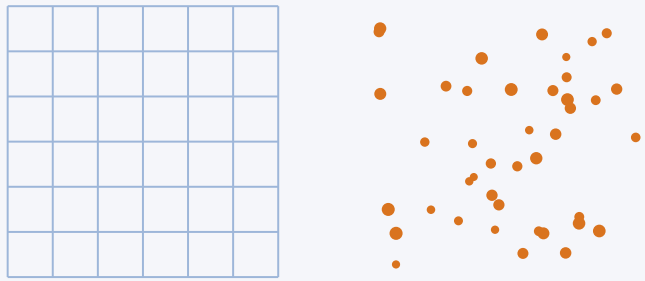
A Physics-Informed Neural Network takes a different route:

No stencils — automatic differentiation supplies exact derivatives at any point

No remeshing — the same architecture handles new geometries

A continuous, evaluable-anywhere temperature field, not a lookup table on a grid

Grid of stencil nodes vs. scattered collocation points



discretize differentiate

Mesh-based methods: error scales with Δx, Δt — refine to converge.

PINNs: error scales with training budget — same architecture, any point.

METHODOLOGY

Anatomy of a physics-informed network

Loss, $\mathcal{L} = \lambda_1 \cdot \mathcal{L}_{pde} + \lambda_2 \cdot \mathcal{L}_{bc} + \lambda_3 \cdot \mathcal{L}_{ic}$ **(** $+ \lambda_4 \cdot \mathcal{L}_{data}$**, optional)**

PDE term

Residual on collocation points, computed via 2nd-order automatic differentiation.

Boundary term

Dirichlet / Neumann / Robin — enforced softly (loss) or exactly (hard constraint).

Initial term

Matches T(x,·,0) — same soft/hard choice as boundaries.

Automatic differentiation forms every derivative in the PDE exactly — the network is trained so the residual vanishes everywhere in the domain, not just at data points.

THE BENCHMARK SUITE

Three problems, three kinds of difficulty

1D · TRANSIENT

Moving Gaussian Source

A rod of length 18 m, initially at 25°C, heated by a Gaussian pulse that travels along its length and decays in intensity.

Difficulty: a source that moves in space and time

2D · STEADY STATE

Robin-Cooled Quarter-Disk

A quarter-disk with a circular cooling hole. Insulated straight walls and outer arc, convective (Robin) cooling at the curved hole.

Difficulty: a curved, mixed boundary condition

3D · TRANSIENT

Discontinuous Conductivity

A cube split into octants with conductivity jumping 10× across material interfaces, heated by a fixed Gaussian source.

Difficulty: a conductivity field with real discontinuities

All three are solved twice — once by a classical mesh-based method (FDM/FEM) and once by a PINN — and cross-validated against each other.

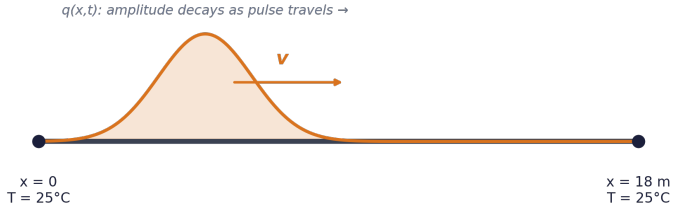
CASE 1 · 1D TRANSIENT

A heat source on the move

$$\partial T/\partial t = \alpha\, \partial^2 T/\partial x^2 + q(x,t)$$

$$q(x,t) = A \cdot \exp[-(x - vt)^2 / (2\sigma^2)] \cdot \exp(-\beta t)$$

Parameter	Value
Domain / initial / boundary	Rod, L = 18 m; T(x,0) = 25°C; T(0,t)=T(L,t)=25°C
Diffusivity α	1.0 m²/s (k = ρ = Cp = 1)
Source amplitude A	4500 W/m³
Pulse velocity v	1.0 m/s
Spatial width σ	1.2 m
Decay rate β	0.08 s⁻¹
Simulated horizon	t ∈ [0, 20] s



Why this is hard for a plain PINN

A narrow (σ=1.2m out of L=18m), fast, translating bump — the classic spectral-bias failure case for MLP-PINNs.

The network must track a feature in motion, not just interpolate a static field.

CASE 1 · PINN DESIGN

Soft constraints only — what went wrong

T ≡ 25°C everywhere satisfies IC + BC almost perfectly — an easy attractor for gradient descent from small-weight init.

PDE loss froze at its untrained baseline (R = −q, since $\partial T/\partial t = \partial^2 T/\partial x^2 = 0$ for a flat field) — 0.065, and stayed there.

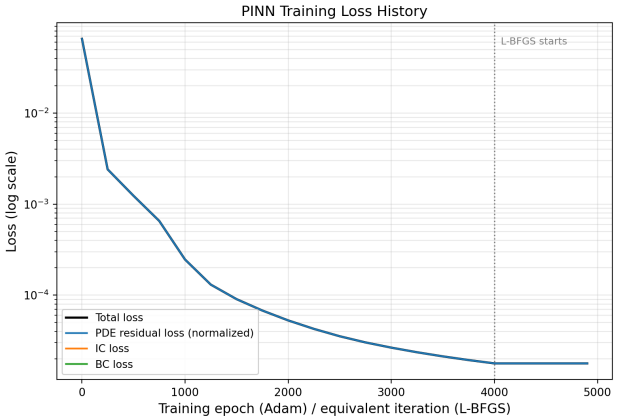
Raising w_pde 10–20× and training thousands more epochs did not escape the plateau.

The fix: bake IC/BC into the architecture

$$T(x,t) = T_{am}^{\,b} + T\_scale \cdot (x/L)(1-x/L) \cdot [1-e^{-t/t^*}] \cdot N\theta(x,t)$$

Satisfies $T(x,0)=T(0,t)=T(L,t)=T_{am}^{\,b}$ exactly, for any weights — a classic fix (Lagaris et al., 1998) that removes the competing attractor entirely.

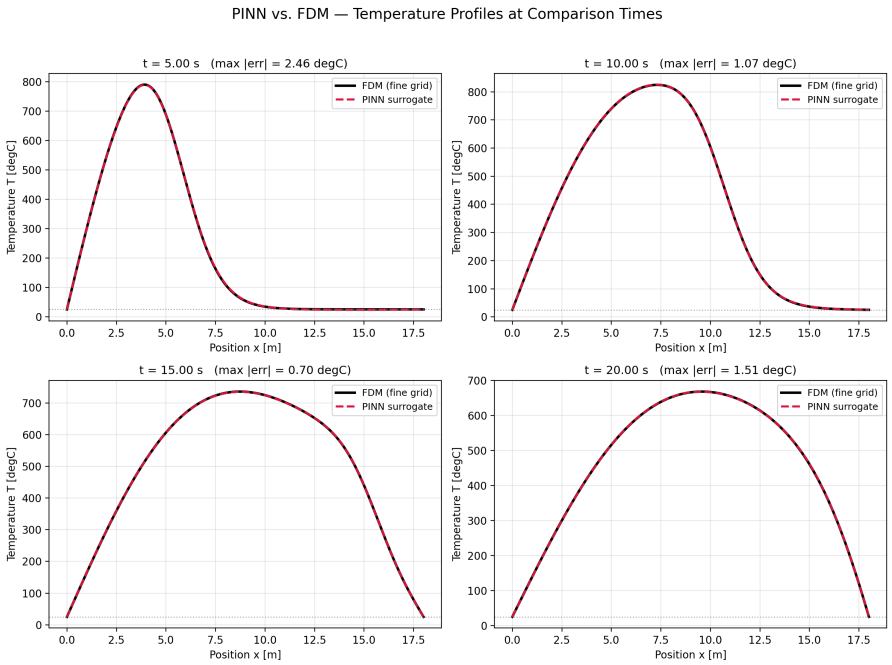
	Soft-constraint only	Hard-constraint
PDE loss, ~2–3k epochs	stuck ≈ 0.065	1.8×10^{-5}
IC / BC loss	→ 0 (but PDE frozen)	exactly 0.0 (by construction)



PDE loss falls ~3,700× (0.0656 → 1.78×10^{-5}) once the trivial-solution attractor is removed — below the 10^{-4} target.

CASE 1 · RESULTS

The PINN tracks the moving pulse to 0.23% error



FDM (solid) vs. PINN (dashed) at t_c = 5, 10, 15, 20 s — curves overlap almost exactly.

0.227%

L2 relative error, whole space-time field

848.6 vs 849.0 °C

FDM vs. PINN peak temperature

7.72 vs 7.73 s

FDM vs. PINN time of peak

t	L2 rel. err.	Max |err|
5 s	0.35%	2.46 °C
10 s	0.13%	1.07 °C
15 s	0.06%	0.70 °C
20 s	0.12%	1.51 °C

Timing, reported honestly: FDM fine-grid solve 50 ms · PINN training 479 s (one-time, single CPU core, no GPU) · PINN inference on 905,181 points 2.05 s. For this cheap 1D linear PDE, classical FDM is simply faster — the PINN's case rests on mesh-free evaluation and extensibility to parametric surrogates, not raw speed here.

CASE 2 · 2D STEADY STATE

Cooling a disk through a curved hole

$\nabla^2 T = -4$ on the quarter-disk Ω

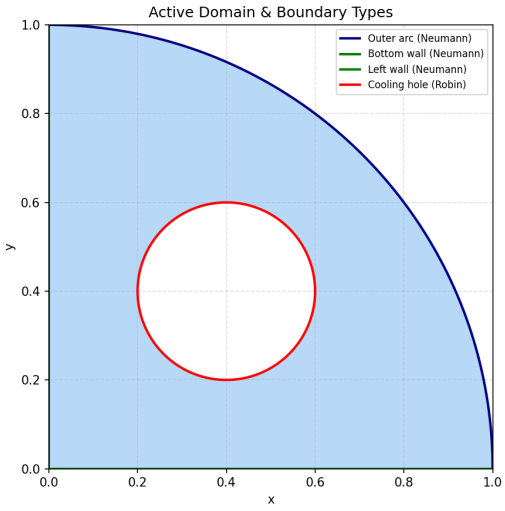
Ω = quarter-disk of unit radius, minus a circular hole of radius 0.2 centred at (0.4, 0.4)

Boundary	Type	Condition
Straight walls x=0, y=0	Neumann	∂T/∂n = 0 (insulated)
Outer arc $x^2+y^2=1$	Neumann	∂T/∂n = 0 (insulated)
Cooling hole boundary	Robin	K ∂T/∂n = −T (K=1, H=1)

Conservation identity (exact validation check):

$$K\partial T/\partial n \, ds = \int \Omega \, 4 \, dA = 4(\pi/4 - \pi \times 0.04) \approx 2.639$$

All heat generated inside Ω must exit through the hole — satisfied by both solvers to within a fraction of a percent.



Active domain: Neumann walls, outer arc, Robin hole

CASE 2 · PINN DESIGN

Warm-starting the network with physics

Particular-solution warm start: $T_\theta(x,y) = (1 - x^2 - y^2) + NN_\theta(x,y)$

The leading term already solves $\nabla^2 T = -4$ and both wall Neumann conditions exactly — the network only learns the correction.

Loss term	N pts	Weight	AD order
PDE: $\nabla^2 T_\theta + 4 = 0$	2000	1.0	2nd
Wall: $\partial T_\theta / \partial n$	300	5.0	1st
Arc: $\partial T_\theta / \partial n$	300	5.0	1st
Hole (Robin)	300	10.0	1st

Two-stage optimisation: Adam (900 epochs, lr 10^{-3}, ×0.5 @ 300/600) → L-BFGS (30 steps, strong-Wolfe, resampled every 3 steps). Total loss falls 4 orders of magnitude; L-BFGS alone cuts it 10× (0.012→0.0014) in ~30s.

Outward normal at the hole points into the hole: $-(\cos\theta, \sin\theta)$. First implementation used the opposite sign.

Training converged confidently to $T<0$ everywhere — low loss, wrong physics. Caught only via a controlled sign-flip test against a known analytic solution.

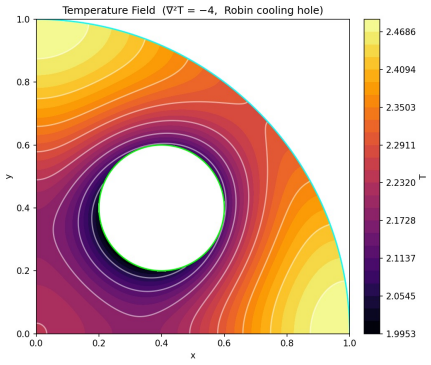
Using (x^2, y^2, xy) as network inputs hard-constrains the wall Neumann BCs by symmetry — mathematically valid.

In practice it distorted the loss landscape and converged to wrong, small-amplitude solutions. Abandoned in favor of the additive warm-start + soft Neumann losses above.

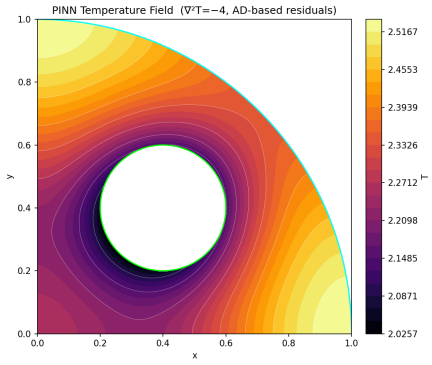
Contrast with Case 1: hard constraints there fixed a stuck optimizer; here, an over-parameterized hard constraint created a new one. The right tool depends on how the constraint is baked in.

CASE 2 · RESULTS

Two solvers, one temperature field



FDM



PINN

Quantity	FDM	PINN	FEM ref.
T_min	1.995	2.025	2.026
T_max	2.488	2.537	2.538
∫ T dA	1.513	1.542	1.535
Flux balance err.	~1.4%	0.05%	—

T_max error vs FEM: FDM 2.0% → PINN 0.04%
Flux balance error: FDM ~1.4% → PINN 0.05%
Solution time (CPU): FDM <2s → PINN ~210s

The PINN's exact, pointwise autograd evaluation of the curved Robin boundary avoids the staircase error of the Cartesian FDM grid — at ~100× the compute cost.

CASE 3 · 3D TRANSIENT

A conductivity field with real jumps

$$\partial T/\partial t = \nabla\cdot(k(x,y,z)\nabla T) + Q(x,y,z)$$

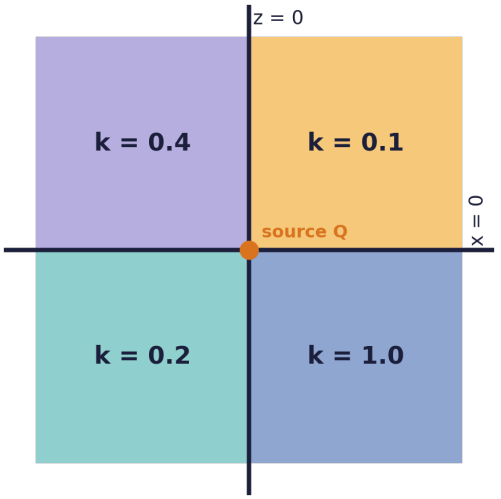
Cube $[-1,1]^3$, $t \in [0,3]$; cold start $T(x,y,z,0)=0$; $T=0$ on all six faces.

Fixed Gaussian source $Q = 10\cdot\exp(-(x^2+y^2+z^2))$ at the origin.

Conductivity k is piecewise-constant across 4 octant-pairs split by the x=0, z=0 planes — a 10× jump that makes ∇k a delta function.

The problem for autodiff: $\nabla\cdot(k\nabla T) = \nabla k\cdot\nabla T + k\nabla^2 T$ needs ∇k, which is undefined at a hard jump.

Fix: mollify the jump with a tanh transition of half-width ε=0.02 (1% of the domain half-width); compute ∇k in closed form from the blend, not via autograd.



Octant conductivities (y-independent cross-section)

CASE 3 · PINN DESIGN

Sampling and weighting for a stiff, discontinuous field

Architecture & loss

6 hidden layers × 64 tanh neurons (~21,185 params); time rescaled to match spatial coordinate range.

Loss = 1·Loss_PDE + 20·Loss_BC + 20·Loss_IC

BC/IC up-weighted — they're the 'easy' exact constraints a pure PDE loss can under-satisfy.

Importance-sampled collocation (8,000/iter)

50% uniform

25% concentrated near the Gaussian source

15% concentrated near the x=0 / z=0 conductivity interfaces

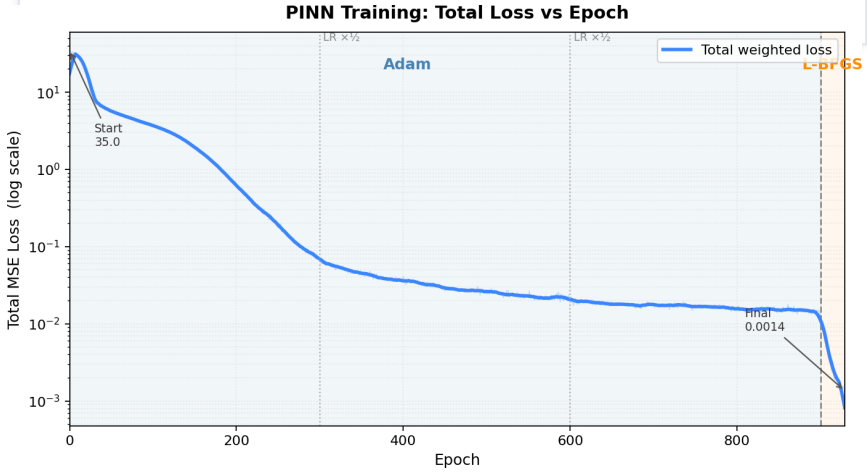
10% skewed toward t=0 to resolve the initial transient

1,200 BC (200/face) + 1,200 IC points, all resampled every iteration

Two-phase training

Adam: 6,000 epochs, lr 10^{-3} halved every 1,000 — loss 32.6 → 2.4

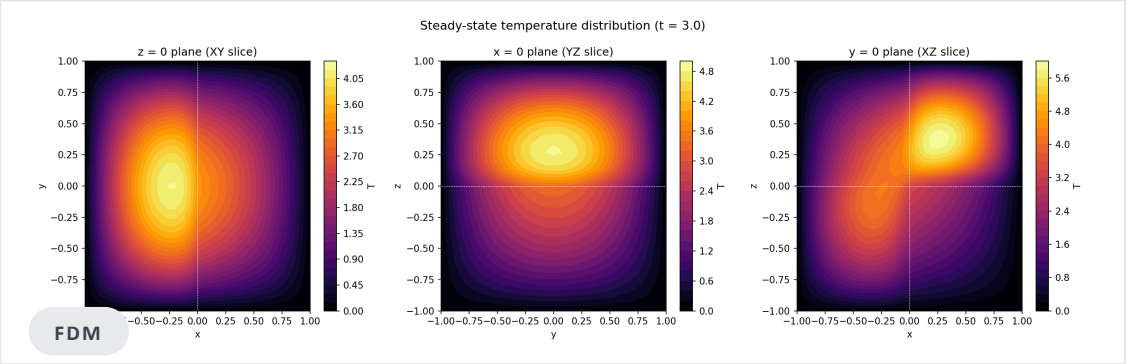
L-BFGS: 1,953 closure evals (6 chunked runs) — final loss 0.1548 (PDE 0.1097, BC 0.0016, IC 0.00066)

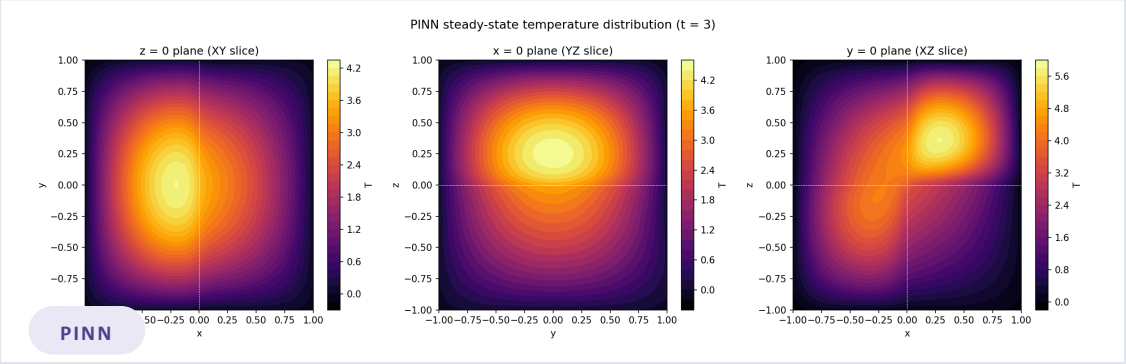


Total loss vs. epoch: Adam (0–6,000) then L-BFGS — falls >2 orders of magnitude.

CASE 3 · RESULTS

Matching physics across a material jump





5.92 vs 5.80

Global peak T, FDM vs PINN (−2.0%)

3.37 vs 3.65

Center-point T, FDM vs PINN (+8.4%)

13.5s vs 41.3min

Compute time, FDM vs PINN (≈184×)

Honest read

8-corner monitors over-predict by 8–25% (mean ~15%); traced to soft BC, ε-mollified k, and a compute-limited (1 CPU, no GPU) budget — not a methodological flaw.

WHAT CARRIED ACROSS ALL THREE CASES

Design principles that generalize

Smooth activations

tanh keeps the network differentiable to the order the PDE needs — no kinks for autodiff to trip over.

Adaptive / careful loss weighting

Up-weighting boundary and initial terms (w_hole=10 in Case 2; λ_bc=λ_ic=20 in Case 3) keeps 'easy' exact constraints from being drowned out by the PDE residual.

Hard constraints: powerful, but not universal

Fixed a stuck optimizer in Case 1 (trivial-solution attractor removed). Backfired in Case 2 when baked in via input features, distorting the landscape — an additive warm-start worked instead.

Two-stage Adam → L-BFGS

Adam handles large early residuals cheaply; L-BFGS's curvature estimate then sharpens final convergence — seen in all three cases (10×–100× loss drops in the L-BFGS phase alone).

Warm-start with particular solutions

Seeding a model with an exact or approximate particular solution (Case 2's $1-x^2-y^2$) gives training a physically sensible starting point — safer than encoding constraints via input features.

Importance-sampled collocation

Concentrating points near sources, interfaces, and t=0 (Case 1: 40% near the pulse path; Case 3: 25%/15%/10% near source/interfaces/t=0) resolves exactly where residuals concentrate.

THE VERDICT

Where each method wins

FDM / FEM wins when

- The domain is simple, structured, low-dimensional
- The same operator is reused every time step (factorize once — Case 3's 13.5s)
- You need a convergence guarantee, not just a good fit
- Wall-clock time is the binding constraint

PINN wins when

- The boundary is curved or the geometry is irregular (Case 2: 0.05% vs 1.4% flux balance)
- You need T evaluable anywhere, not just at grid nodes
- The problem is inverse — inferring k or a source from data
- One trained network must answer many related (parametric) queries

Across 1D, 2D, and 3D, both methods converged to the same physics. The PINN pulled ahead precisely where geometry got curved or discontinuous and exact, mesh-free boundary evaluation mattered (Case 2); classical FDM stayed 100–200× cheaper everywhere the domain was simple and structured (Cases 1 and 3). Neither result is a surprise once you see the mechanism — which is the point.