ICCFD13-2026-306

Uncertainty Quantification for Inverse Heat Transfer via Machine Learning and Quantum Monte Carlo

Ramesh Kolluru, Aditi Lal, Ferdin Don Bosco, Alex Khan,
Rut Lineswala, and Abhishek Chopra
BosonQ Psi Corp., Syracuse, New York 13202, USA

Abstract

We present an integrated framework for uncertainty-aware inverse heat transfer tailored to thermal protection system characterization. The approach targets recovery of spatially varying thermal conductivity from noisy transient temperature measurements on a one-dimensional slab with mixed boundary conditions, representative of certification-by-analysis workflows for thermal protection materials. To address the cost and fragility of traditional gradient- and adjoint-based calibration in large-scale settings, we train an inverse physics-informed neural operator on synthetic ensembles generated by a high-fidelity implicit finite-volume solver; the default inverse operator uses a compact convolutional backbone, with an optional Fourier-neural-operator backbone used for controlled architecture comparisons. This learned inverse map is coupled with Karhunen–Loève representations of stochastic input fields to enable efficient uncertainty quantification in the inferred conductivity. A quantum Monte Carlo (QMC) estimator is then used to accelerate sampling-based uncertainty propagation while remaining compatible with near-term quantum hardware. Together, these components deliver near-real-time, uncertainty-aware inverse characterization uncertainty propagation, not uncertainty quantification because KL prior samples are propagated, MC are applied to scalar QoIs so No posterior uncertainty over k(x) is actually inferred for thermal protection system–like problems, bridging the gap between high-fidelity Bayesian methods and deployable certification workflows.

Keywords: uncertainty quantification (UQ); machine learning (ML); Monte Carlo (MC); quantum Monte Carlo (QMC); inverse heat transfer; inverse heat conduction problem (IHCP); physics-informed neural operator (PINO); SimplePINO-inverse; Fourier neural operator (FNO); Karhunen–Loève (KL) expansion; thermal protection systems (TPS).

Acronyms (selected). BFGS: Broyden–Fletcher–Goldfarb–Shanno; CNN: convolutional neural network; CPU / GPU: central / graphics processing unit; FNO: Fourier neural operator; HTC: heat transfer coefficient; IHCP: inverse heat conduction problem; KL: Karhunen–Loève; L-BFGS-B: limited-memory BFGS with bound constraints; MAE / RMSE: mean absolute error / root mean square error; MC: Monte Carlo; PDE: partial differential equation; PINN: physics-informed neural network; **QMC**: quantum Monte Carlo; the estimator implemented here follows the modewise quantum Monte Carlo construction of Agarwal et al. [1] (distinct from classical spatial Monte Carlo unless noted); QoI: quantity of interest; R^2: coefficient of determination; SSE / SSE_T: sum of squared errors (sensor domain, when subscripted); TPS: thermal protection system; UQ: uncertainty quantification.

1 Introduction

Machine learning (ML) is widely used in computational physics and engineering because it can learn useful input–output maps from data. In many problems, classical calibration methods based on gradients or adjoints can be expensive or unstable [2]. After training, ML models such as neural networks and neural operators can give fast predictions for forward and inverse maps [3, 4]. This is helpful when the same model must be evaluated many times, for example in design studies or certification workflows.

Many engineering tasks are inverse problems: we want to infer unknown model inputs from measured outputs [5]. In heat transfer, this includes inverse heat conduction, where temperature measurements are

used to infer quantities such as conductivity or heat flux [6]. These inverse problems are often ill-posed and sensitive to noise. Physics-informed ML methods (e.g., PINNs and PINOs) help by adding the governing equations to the training loss, which can improve stability compared with purely data-driven models.

Uncertainty quantification (UQ) is important because inverse solutions are not unique and measurements are noisy [7]. For ML-based inverse solvers, UQ can capture both measurement noise and model uncertainty [8, 9]. Uncertainty estimates also help plan new experiments by showing which measurements would reduce uncertainty the most, which reduces iterative design process in critical applications like Thermal Protection System (TPS) studies for space re-entry [10].

In this work, we estimate a spatially varying thermal conductivity $k(x)$ from noisy transient temperature data $T(x, t)$ on a one-dimensional slab with mixed boundary conditions [11]. We train a physics-informed inverse model on synthetic data generated with a high-fidelity implicit finite-volume solver. Our baseline inverse model uses a convolutional network, and we also test a Fourier neural operator (FNO) backbone to enable controlled architecture comparisons. To represent uncertain inputs, we use a Karhunen–Loève (KL) expansion and report uncertainty measures for the recovered conductivity. As a diagnostic check for sampling-based UQ, we also run a quantum Monte Carlo (QMC) estimator.

2 Related Work

Inverse heat conduction methods fit models to temperature data to estimate quantities that are hard to measure directly, such as heat flux or spatial conductivity [12–15]. These methods are well understood, but they can be slow because each new case may require many forward or adjoint solves. Applications continue to motivate inverse identification in fin–tube exchangers [16], bubbly jets [17], vertical-plate cooling [18], transient property-estimation benchmarks [19], and quenching-related analyses [20]. Another significant application is the Inverse Heat Conduction Problem (IHCP), which has been thoroughly investigated in online summaries [21].

Scientific machine learning aims to learn these maps once and then reuse them for fast prediction [22]. Neural operators extend this idea by learning mappings between functions; examples include Fourier neural operators [23] and DeepONet-type models [24, 25]. Heat-transfer-relevant operator-learning developments by Lu and co-authors include multifidelity neural operators for fast inverse design in nanoscale heat transport [26] and operator learning for multiscale bubble growth dynamics [27]. These models can act as surrogates for families of PDE problems [28] and have also been applied to instability-wave prediction in high-speed boundary layers [29].

Uncertainty remains a key need. Under standard assumptions for i.i.d. Monte Carlo with finite variance, the RMSE typically decays like $O(N^{-1/2})$, so reaching RMSE ε often requires $N = O(\varepsilon^{-2})$ samples. Quantum amplitude estimation can improve this scaling in theory [30, 31], but practical QMC estimators still need careful, budget-matched comparison with classical MC [1]. For inverse heat transfer, careful methodology is also important before claiming one approach is better than another [32–34].

3 Problem Formulation

3.1 Forward conduction model

We consider transient one-dimensional conduction on $x \in [0, L]$:

$$\rho c_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k(x)\frac{\partial T}{\partial x}\right), \tag{1}$$

with initial condition $T(x, 0) = T_0(x)$ and thermal boundary data drawn from Dirichlet, Neumann, or Robin families consistent with the synthetic benchmark generator (see Figure 1).

3.2 Inverse objective

Let $\mathcal{F} : k \mapsto T$ denote the implicit finite-volume forward operator that maps a conductivity field and boundary forcing to space–time temperatures. We approximate the inverse map by a parametric operator

$$\hat{k}(x) = \mathcal{G}_\theta\big(\widetilde{T}(x,t)\big), \tag{2}$$

where $\widetilde{T}$ stacks noisy observables on the primary benchmark grid (shape $N_t \times N_x$ after tensor reshaping inside the network). Unless noted otherwise, $\mathcal{G}_\theta$ denotes the default **SimplePINO-inverse** map; Section 5.3 documents an optional spectral **PINO-inverse** variant, included to enable controlled architectural comparisons and to remain consistent with the spectral (FNO-based) forward PINO solver. Since many conductivity fields can explain the same temperature history, the inverse map is ill-posed unless k is constrained; here, well-posedness is improved by restricting k to the finite-dimensional KL subspace in (3).

3.3 KL prior conditioning

Random conductivity fields are represented through a truncated Karhunen–Loève (KL) expansion:

$$k(x,\omega) = k_{\text{mean}} + \sum_{i=1}^{N} \sqrt{\lambda_i}\, \phi_i(x)\, \xi_i(\omega), \tag{3}$$

where $k_{\text{mean}} = \mathbb{E}[k(x,\omega)]$ and the KL coefficients ξ_i are independent and identically distributed (i.i.d.) standard normal random variables on $(\Omega, \mathcal{F}, \mathbb{P})$. The eigenpairs (λ_i, ϕ_i) are obtained from the covariance operator associated with an exponential kernel. Unless stated otherwise, we truncate to $N = 10$ modes in the slab benchmark experiments.

4 Integrated framework overview

A simplified one-dimensional heat transfer representation, shown in Figure 1, is utilized to demonstrate the application of the proposed framework. The governing equation is the transient heat equation (1) with spatially varying thermal conductivity $k(x)$, constant material density ρ, and specific heat capacity c_p. The system is subject to Dirichlet, Neumann, or Robin boundary conditions.

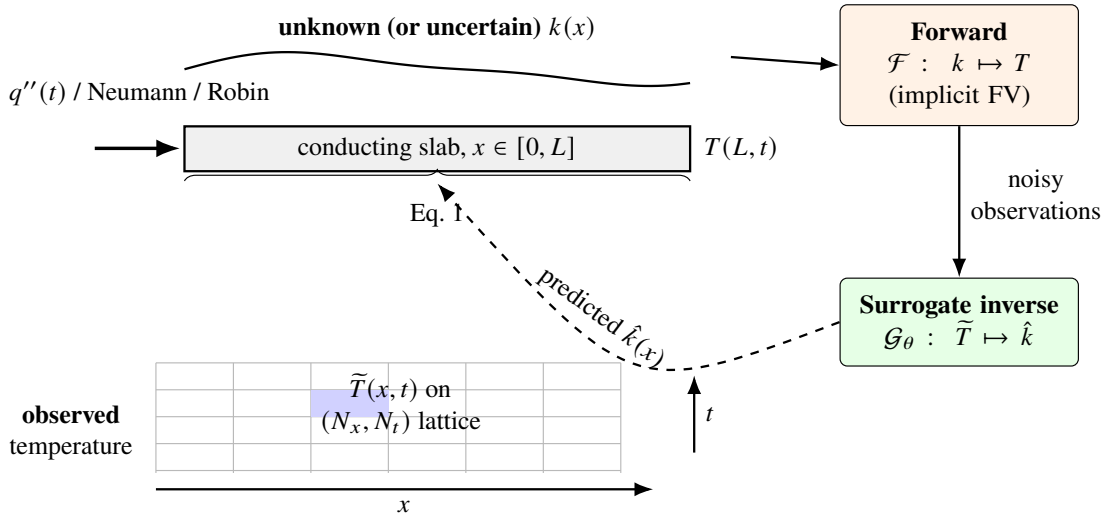


Figure 1: Schematic of the primary inverse problem: uncertain or unknown conductivity $k(x)$ enters transient conduction (1); the forward map $\mathcal{F}$ generates temperatures on a space–time grid; noisy full-field data $\widetilde{T}$ feed the operator $\mathcal{G}_\theta$ that reconstructs $\hat{k}$.

The inverse problem statement seeks to determine $k(x)$ given noisy $T(x,t)$. Conductivity is represented as a random field using the KL expansion (3). Performing UQ then entails propagating this stochastic input through the forward model and computing statistical moments, a task that becomes costly when relying on classical Monte Carlo (MC) sampling. Our contribution is an end-to-end pipeline comprising: (1) a forward solver coupled with KL-based UQ for ensemble generation; (2) a physics-informed inverse operator; and (3) MC and QMC estimators for UQ, enabling fair comparisons at equal computational cost or equal target error. The overall workflow, shown in Figure 2, consists of three tightly coupled stages: training, parameter estimation, and uncertainty quantification.

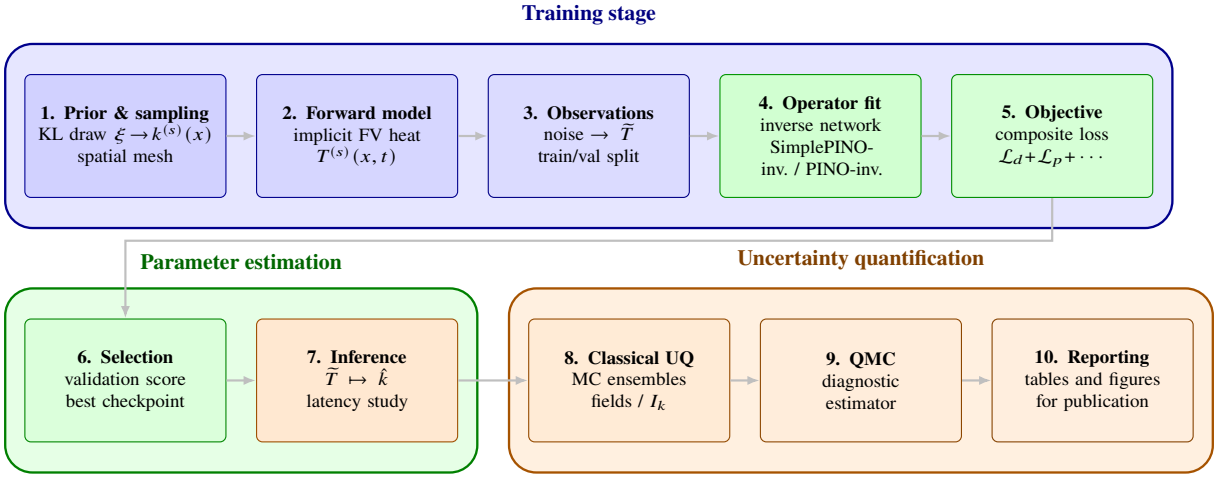


Figure 2: End-to-end workflow for the inverse benchmark and uncertainty diagnostics, grouped into three stages (training; parameter estimation; uncertainty quantification): synthetic prior draws and forward labels, supervised operator training with composite losses, deployment-time inference, classical Monte Carlo reference bands, optional finite-budget QMC comparison on I_k, and aggregation into tables and figures.

During the training stage, spatial variability in thermal conductivity is generated using a Karhunen–Loève expansion (3), together with variations in the imposed thermal boundary conditions. This ensemble of random conductivity fields and boundary conditions is propagated through a high-fidelity implicit finite-volume solver to produce the corresponding transient temperature fields. These paired input–output fields constitute the training data used to fit the learned operators (forward surrogate, when used, and the inverse operator reported here) over the prescribed stochastic space.

Unlike forward-only UQ settings [1], the present setting targets the inverse problem with spatially varying coefficients, so that inferred $k(x)$ and its uncertainty feed into downstream certification quantities (e.g., surface heat flux, wall temperature). In the parameter estimation stage, the trained inverse operator maps measured or simulated temperature histories to conductivity on the training stencil. The **default** architecture is **SimplePINO-inverse** (purely convolutional; Figure 2); for controlled comparisons we also train **PINO-inverse**, which adds an FNO1d spectral backbone on x after a (t,x) convolutional encoder (Figure 4 and Table 1). Training may follow a two-phase schedule: an initial data-only warmup, followed by a phase that gradually introduces physics. The composite loss includes data misfit, a finite-difference PDE residual on the observation grid, boundary-condition terms, and regularization (Section 5.4). We consider temperature data with complete spatial coverage at discrete time points (assuming temperature data are available on the chosen (x,t) grid) and measurement noise levels typical of experiments (e.g., relative errors of $\leq 5\%$). Under these conditions, the documented SimplePINO-inverse run attains about 2–3% mean relative MAE on $k(x)$ over held-out validation fields with high R^2 on most cases (Tables 4–3; Section 7).

Finally, in the uncertainty quantification stage, we compare classical Monte Carlo (MC) and quantum Monte Carlo (QMC) estimators for the conductivity-integral quantity of interest (QoI), $I_k = \int_0^L k(x)\,\mathrm{d}x$, following [1]. QMC is implemented with finite depth/shot schedules and evaluated under classical simulation. We report both a fixed-budget comparison and a matched-accuracy RMSE ladder on the

same KL-profile ensemble (Tables 6, 7; Figure 6). At matched budget, classical MC achieves lower mean absolute error, whereas in the matched-accuracy study QMC reaches selected RMSE targets with fewer oracle calls than nested MC requires samples. These results are presented as estimator-level diagnostics for this benchmark, not as a claim of hardware quantum speedup.

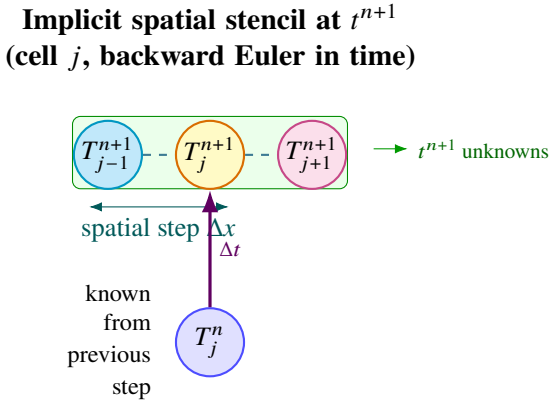
5 Methodology

5.1 Forward data generation

The process starts with a finite volume solver generating data with various initial and boundary conditions which match either on an experimental or theoretical basis. For each random seed draw $\omega^{(s)}$, coefficients $\{\xi_i^{(s)}\}$ induce a conductivity sample $k^{(s)}(x)$ via (3); an implicit finite-volume integrator advances (1) to produce paired temperatures $T^{(s)}(x,t)$. Observational noise is applied only after forward solves to emulate imperfect sensing while keeping latent conductivities noise-free.

Spatial discretization adopts the conservative flux form; Figure 3 illustrates the spatial index stencil and the time step Δt between T_j^n and T_j^{n+1}; the discrete heat balance reads

$$\rho c_p \frac{T_j^{n+1} - T_j^n}{\Delta t} = \frac{1}{\Delta x}\left[k_{j+\frac{1}{2}} \frac{T_{j+1}^{n+1} - T_j^{n+1}}{\Delta x} - k_{j-\frac{1}{2}} \frac{T_j^{n+1} - T_{j-1}^{n+1}}{\Delta x} \right]. \tag{4}$$



Indices $j = 0, \ldots, N_x - 1$ and $n = 0, \ldots, N_t - 1$ match the exported observation lattice. Equation 4 couples three neighbors at t^{n+1} (backward Euler in time); the training-time differentiable residual uses the same $(\Delta x, \Delta t)$ grid with centered spatial and forward temporal differences as noted in Section 5.4.

Figure 3: Discrete notation for (4): uniform spacing Δx along x and step Δt in time. Implicit fluxes at t^{n+1} link T_{j-1}^{n+1}, T_j^{n+1}, and T_{j+1}^{n+1}; the explicit history T_j^n enters the left-hand side.

Face conductivities $k_{j\pm\frac{1}{2}}$ use the harmonic mean of neighboring cell-centered values to preserve flux continuity across heterogeneous $k(x)$. Adaptive Δt selection maintains stable Newton solves across the conductivity ensemble, and the temporal spacing is aligned with the N_t snapshots exported to the inverse learner.

5.2 Noisy temperature observations

What the inverse model sees. SimplePINO-inverse takes temperature data on the same (x,t) grid used by the forward solver. In the main benchmark we assume *full-field* measurements (temperatures at all N_x locations and N_t times). This keeps the focus on the inverse mapping itself, rather than on where sensors are placed. To mimic measurement error we add Gaussian noise:

$$\widetilde{T}_{i,n} = T_{i,n} + \epsilon_{i,n}, \qquad \epsilon_{i,n} \sim \mathcal{N}\big(0, (\sigma_{\text{noise}}\,\hat{\sigma}_T)^2\big), \tag{5}$$

where $\hat{\sigma}_T$ is the standard deviation of T over space and time for that realization and we use $\sigma_{\text{noise}} = 0.01$ unless stated otherwise. The same noise rule is used for both training and validation.

Important implication for physics losses. In our physics-informed training loss (used for both **SimplePINO-inverse** and **PINO-inverse**), we compute the finite-difference PDE residual directly on the noisy temperature field $\widetilde{T}$ (Section 5.4). If the noise level is not small, this residual can mainly penalize measurement noise (via finite differences of $\widetilde{T}$) rather than reliably constraining the inferred conductivity through the PDE. As a result, mixed data+physics training can perform worse than data-only training unless we tune the residual weight or denoise $\widetilde{T}$.

Why the problem becomes manageable. We do not allow completely arbitrary conductivity fields. Instead, each $k(x)$ is generated from a truncated KL expansion (ten random coefficients in the default setup). This low-dimensional constraint makes the inverse problem much better behaved; changing the KL correlation length, variance, or the noise level changes the achievable accuracy and is left for sensitivity studies.

5.3 Spectral variant: PINO-inverse

PINO-inverse is the Fourier-backbone inverse operator used for controlled comparisons against the default **SimplePINO-inverse**. It builds a two-channel pointwise input by concatenating the normalized temperature field and a **normalized spatial coordinate** $x \in [0, 1]$ (linearly spaced along each column), yielding $(B, N_t, N_x, 2)$, then permutes to $(B, 2, N_t, N_x)$. A **temporal encoder** applies two `Conv2d` layers with kernel $(5, 1)$, padding $(2, 0)$, channel path $2 \to 64$ then $64 \to 64$, each followed by `GELU`. **Temporal pooling** uses `AdaptiveAvgPool2d` to $(1, N_x)$, collapsing time and producing $(B, 64, N_x)$ after squeezing. The core is an **FNO1d** stack on x: a width-preserving 1×1 lifting `Conv1d`, followed by L=4 identical Fourier blocks. Each block sums a truncated **spectral convolution** (real FFT along x, complex multiplication on the first m modes, inverse FFT) with a **pointwise** `Conv1d` ($64 \to 64$, k=1), then applies `GELU`. A projection sub-network applies two 1×1 convolutions $64 \to 64 \to 64$ with `GELU` between. Finally, a **material head** uses three `Conv1d` layers $64 \to 64 \to 32 \to 1$ (k=1, `GELU` between hidden layers); the last layer bias is initialized to zero and the last weights use a small Xavier gain. Unless noted in a controlled sensitivity study, defaults are m=16 retained modes per spectral layer, hidden width 64, and L=4 Fourier blocks. Figure 2 shows where the operator sits in the workflow; Fig. 4 traces tensors for the default slab grid (N_x=50, N_t=20). Table 1 lists the corresponding layer dimensions. To report physical conductivity, outputs are denormalized with training-set statistics $\bar{k}$ and σ_k (W/(m·K)). The head does not apply a built-in positivity clamp; normalized supervision keeps mapping smooth, and rare slightly negative $\hat{k}$ after denormalization can appear when training is purely data-driven. An optional physics penalty may discourage denormalized values outside $[50, 500]$ W/(m·K).

Table 1: PINO-inverse default architecture (matches implementation defaults).

Component	Specification
Input assembly	T plus normalized $x \in [0, 1]$ per column $\Rightarrow (B, N_t, N_x, 2)$; permute to $(B, 2, N_t, N_x)$.
Temporal encoder	Two `Conv2d`: kernel $(5, 1)$, padding $(2, 0)$; $2 \to 64$ then $64 \to 64$; `GELU` after each.
Temporal pooling	`AdaptiveAvgPool2d` to $(1, N_x)$; squeeze $\Rightarrow (B, 64, N_x)$.
FNO1d core	Lift `Conv1d` $64 \to 64$ (k=1); L=4 Fourier blocks: truncated spectral conv along x (m=16 modes) + `Conv1d` $64 \to 64$ (k=1), `GELU`; project $64 \to 64 \to 64$ with `GELU`.
Material head	`Conv1d` $64 \to 64 \to 32 \to 1$ (k=1); `GELU` between 64 channels; last layer small Xavier gain, bias 0.
Output	$\hat{k}(x_i)$ (normalized).

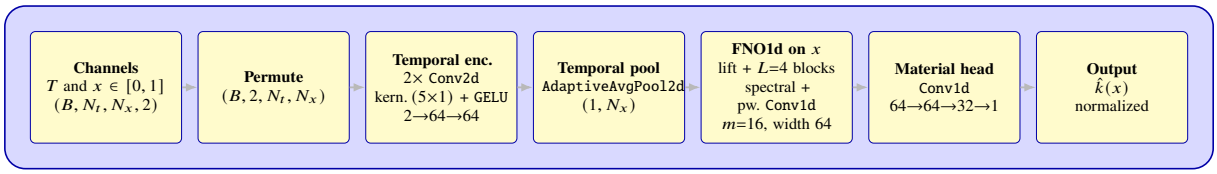


Figure 4: PINO-inverse as implemented: coordinate-augmented field, $(5, 1)$ temporal convolutions, pooling, then FNO1d spectral blocks on x and a 1×1 conductivity head (Table 1).

5.4 Training objective and schedule

Learning minimizes a weighted sum of supervision and physics-aware penalties,

$$\mathcal{L} = \lambda_d \mathcal{L}_{\text{data}} + \lambda_p \mathcal{L}_{\text{PDE}} + \lambda_b \mathcal{L}_{\text{BC}} + \lambda_r \mathcal{L}_{\text{reg}}. \tag{6}$$

Supervised conductivity loss. $\mathcal{L}_{\text{data}}$ averages squared errors between predicted and ground-truth *normalized* conductivity vectors on the spatial stencil. This term anchors the inverse map to KL-generated labels.

Differentiable PDE residual. $\mathcal{L}_{\text{PDE}}$ penalizes how much the predicted $\hat{k}$ and the observed (noisy) temperature $\widetilde{T}$ violate the heat equation, using finite differences on the same grid. We use centered differences in space and forward differences in time. Since we compute this residual on $\widetilde{T}$ (not on a denoised temperature), the residual can also penalize measurement noise, especially in time derivatives. This is one reason why mixed data+physics training can perform worse than data-only training if the weight λ_p is not scheduled carefully. We keep the residual differentiable, so it is not the same as the fully implicit Newton solver used to generate the dataset; instead, it provides a lightweight physics constraint during training.

Boundary and regularization. $\mathcal{L}_{\text{BC}}$ softly enforces consistency between predicted boundary traces and the boundary-condition templates embedded in training data. $\mathcal{L}_{\text{reg}}$ adds mild ℓ_2 control on normalized conductivities to regularize the decoder head.

Optimization proceeds in stages: data-only warm starts stabilize early gradients, scheduled ramps gradually activate physics weights, and validation scores dictate checkpoint retention. Recorded logs preserve $(\lambda_d, \lambda_p, \lambda_b, \lambda_r)$ trajectories for reproducibility when comparing mixed-loss experiments.

5.5 Uncertainty propagation and diagnostic estimators

Classical Monte Carlo baselines. Conductivity ensembles originate either from direct KL sampling or, when documented, from perturbations around inferred fields ($\hat{k}$ from **SimplePINO-inverse** in the main study). Each realization enters the forward solver when spatial temperature uncertainty or flux summaries matter; empirical moments and percentile envelopes summarize variability after N_{MC} pushes (typically $\mathcal{O}(10^2)$ samples for band widths cited alongside Table 2).

QMC diagnostic. Parallel experiments estimate integral QoIs via shallow quantum circuits following [1]: Grover-depth ladders $m \in \{0, 1, 2, 3\}$, five-qubit grids, 64 shots per depth unless varied for scaling plots, and classical Monte Carlo matched through oracle tallies (Section 7).

Budget accounting uses

$$C_{\text{oracle}} = \sum_{\ell=1}^{L} s_\ell (2m_\ell + 1), \tag{7}$$

where s_ℓ is shot count at depth stage m_ℓ.

6 Experimental Protocol

6.1 Overview

We evaluate the inverse models on a clean synthetic benchmark (in-distribution) and on a small set of diagnostic cases that mimic common experimental issues. We keep the data splits, noise assumptions, and target representation consistent across all comparisons.

6.2 Main benchmark: 1D slab

We use a 1D slab of length $L = 10$ mm discretized with $N_x = 50$ spatial points and $N_t = 20$ time snapshots. The conductivity field is modeled with a Gaussian prior with mean $k_{\text{mean}} = 167$ W/(m·K), standard deviation $k_{\text{std}} = 16.7$ W/(m·K), and 10 KL modes. Synthetic temperature data are generated with an implicit heat-conduction solver and controlled noise injection.

6.3 Diagnostic Tier-A cases

To test robustness, we also run a small Tier-A diagnostic set.

- **A: forward-solver cross-check (mild mismatch).** We generate "truth" temperatures with a separate forward-solver configuration and then map them onto the training grid before applying the learned inverse operator. In the configuration reported here ($N_x^{\text{truth}} = 50$ and backward Euler with $\theta = 1$), the mapping is effectively the identity and the solver choice largely matches the dataset generator, so this test should be viewed as a mild consistency check. A stronger mismatch test would change the truth resolution, time step, or boundary-condition family.
- **B: sparse sensors.** To mimic thermocouples, we keep temperatures only at a small number of space–time locations (e.g., 8 sensor positions and 10 time stamps), add Gaussian noise, and then reconstruct a full (N_t, N_x) grid by linear interpolation in space and time. This lets the learned operator run unchanged, but the reconstruction is approximate. Results should therefore be reported versus the number of sensors and the reconstruction method; without reconstruction, the model must be trained to accept sparse masks.
- **C: calibration check.** Using a calibration subset, we pool absolute errors $|\hat{k} - k|$ to build symmetric "tube" intervals and report empirical coverage at different nominal levels. This is a quick distribution-free sanity check, but it is not a formal conformal guarantee for the full spatial field.

6.4 Data splits

We use 200 samples to tune the conductivity inversion model (160 train / 40 validation). This dataset size is chosen for a controlled, low-cost benchmark (full-field observations and a low-dimensional KL parameterization); larger training sets can further improve accuracy and are a straightforward extension. Uncertainty aggregation uses 100-case ensembles. Additional MC/QMC profile studies use 100 profiles. All model comparisons use matched partitions.

Experiment	N	Train	Val.	Eval
Conductivity inversion tuning	200	160	40	held-out val.
Conductivity UQ aggregate	100	—	—	100-case ens.
MC/QMC profile study	100	—	—	100 profiles

Table 2: Consolidated sample counts and split protocol used across experiments.

6.5 Metrics and reporting

Inverse accuracy is reported with MAE, RMSE, relative MAE, and R^2. Agreement in the temperature (observation) space is measured using SSE_T. We report per-sample inference latency together with the CPU/GPU context. For uncertainty quantification, we report the sample mean, standard deviation, standard error, and percentile bands.

Model / setup	**Rel. MAE on k**	**Runtime/sample**
SimplePINO-inverse (best val. checkpoints, repr. sweeps)	3–6%	< 1 ms (GPU)
SimplePINO-inverse (default / weaker configs, repr.)	~5–8%	~1 ms (GPU)
CNN baseline	~7–12%	~1 ms (GPU)
BFGS optimization inverse	~2–4%	~5 s (CPU)

Table 3: Representative inverse baseline comparison aggregated over validation studies. The first two rows refer to the convolutional **SimplePINO-inverse**; **PINO-inverse** (spectral) is discussed in Section 5.3 without a separate band here. **BFGS-inverse:** deterministic inversion with regularized misfit on k using L-BFGS-B, bounded conductivity box consistent with the prior support, and gradients obtained through the differentiable discrete forward map; runtimes reflect $O(10^2)$–$O(10^3)$ forward-equivalent evaluations per case on CPU until stopping tolerances on the misfit gradient are met. Hyperparameters were tuned to reach near-minimal error within that budget so the latency comparison is not an intentionally weak classical baseline.

7 Results and Discussion

We report (i) conductivity inversion accuracy, (ii) uncertainty summaries, and (iii) a simple integral quantity of interest (QoI) to compare classical Monte Carlo (MC) and quantum Monte Carlo (QMC). On the synthetic validation set, the documented SimplePINO-inverse run gives about 2–3% mean relative MAE on $k(x)$ over held-out fields, and about 1.8% relative MAE for a representative single case (Tables 4–5; Figure 5). Across broader sweeps, the relative MAE typically falls in the 3–6% range (Table 3). After training, inference takes about a millisecond per case on the reported hardware (Table 3).

Conductivity inversion. Table 4 summarizes all $N_{\mathrm{val}} = 40$ validation cases for the documented run (200 training samples; encoder 64–128–256–128–64; AdamW $7{\times}10^{-4}$ with weight decay 10^{-4}; seed 42; best validation checkpoint after 35 epochs). Table 5 shows one held-out example from the same run (relative MAE 1.80%, R^2=0.888); three cases attain $R^2 \le 0$ under this single-seed run and are excluded from the R^2 summary in Table 4.

Model / setup	**Val. cases**	**Rel. MAE on k (%)**			**R^2**		**Run**
		mean	std	med.	mean†	std†	
SimplePINO-inverse (doc. run)	40	2.32	1.01	1.93	0.83	0.18	< 1 ms
CNN / BFGS baselines	—	Repr. bands: Table 3					—

Table 4: **SimplePINO-inverse** (documented run): aggregate validation over $N_{\mathrm{val}} = 40$ held-out conductivity fields (best checkpoint after 35 epochs). $^\dagger R^2$ statistics are computed over the 37/40 cases with $R^2 > 0$.

Deterministic BFGS-inverse takes seconds per case on CPU (Table 3). This is why we use the trained inverse model for fast screening, and reserve optimization-based inversion for selected cases when needed.

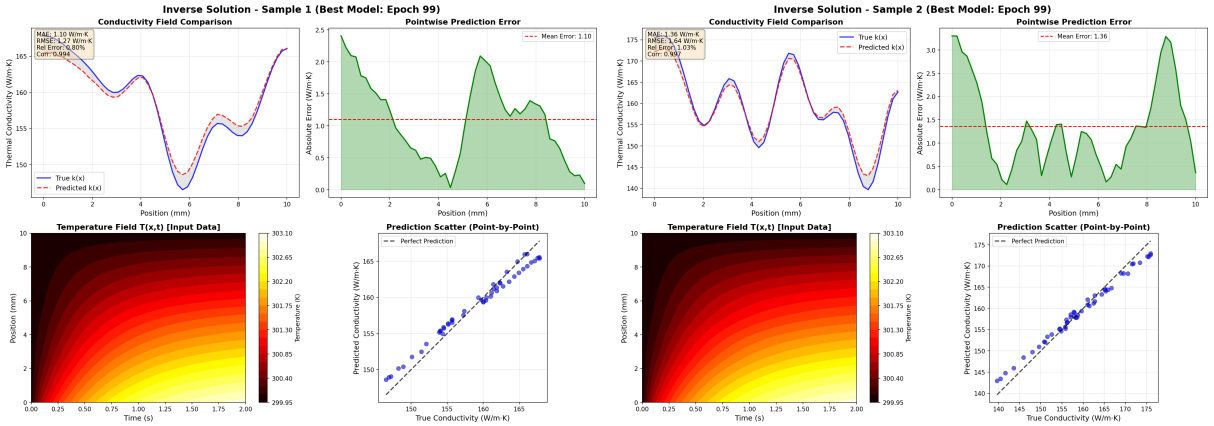


Figure 5: Representative inverse conductivity reconstructions (two held-out samples; SimplePINO-inverse, best validation checkpoint). Shaded bands illustrate an ensemble spread when classical Monte Carlo perturbations are applied around the conductivity prior as in Section 5.5 (schematic visualization).

Primary tuning experiment metric	**Value**
Inverse MAE on $k(x)$	3.27 W/(m·K)
Inverse RMSE on $k(x)$	3.92 W/(m·K)
Inverse relative MAE	1.80%
Inverse R^2	0.888
k uncertainty mean std across x	2.52 W/(m·K)
k uncertainty max std across x	3.82 W/(m·K)

Table 5: Primary tuning metrics for **one** held-out conductivity inversion draw (200-sample study, **SimplePINO-inverse**, documented run).

Uncertainty on conductivity. Using the ensemble procedure in Section 5.5, we compute the spread of the inferred conductivity. For the documented SimplePINO-inverse run, the mean and maximum standard deviation of $\hat{k}(x)$ across x are 2.52 and 3.82 W/(m·K), respectively.

Calibration. In this short paper we report ensemble mean and standard deviation. A simple coverage check for $k(x_i)$ and for I_k is included in the Tier-A diagnostics, but we do not include full reliability curves here.

Integral QoI I_k and MC versus QMC. For the scalar QoI

$$I_k = \int_0^L k(x)\,\mathrm{d}x, \tag{8}$$

where $L = 0.01$ m (10 mm) in our slab setup, so I_k has units of W/K when k is in W/(m·K). Figure 7 compares trapezoidal integration, classical MC, and QMC for one representative profile (implementation details are in Section 7). Classical MC uses 16,384 uniform-in-x samples per profile. QMC is run with the shot/depth schedule listed in Section 7. For this setup, classical MC is closer to the trapezoid reference on average, while QMC is better on 26 out of 100 profiles (Table 6).

QMC implementation (brief). The trapezoidal reference is computed as $I_k^{\mathrm{trap}} = \sum_i w_i k(x_i)$ on the discrete mesh. The classical MC baseline uses the same uniform-in-x estimator with 16,384 samples per profile in the 100-profile table (Table 6) and in the representative two-panel figure (Figure 7).

For the **figure** case, the QMC pipeline uses eight retained Fourier modes, a five-qubit grid, Grover depths in fixed order $(m_1, m_2, m_3, m_4) = (0, 1, 2, 3)$, and s=64 shots at each depth. Oracle accounting follows `cumulative_oracle_calls` in the integration driver: stage ℓ adds $s(2m_\ell{+}1)$ uses, i.e. $\{64, 192, 320, 448\}$ for the four stages, with cumulative totals $\{64, 256, 576, 1024\}$ through the ladder (final per-mode tally 1024). Classical MC is then matched for that illustration by drawing $N_{\mathrm{MC}} = 2 \times (\text{Fourier terms}) \times 1024 = 16384$ uniform x-samples on $[0, L]$ (the same pairing rule used in the Section 6 figure generator).

We implement the QMC estimator described by Agarwal et al. [1]. Optionally, before the Fourier/Grover step, we remove a straight-line trend between the two endpoints of $k(x)$ and later add back the exact integral of that removed line. All QMC results in this paper are from **classical simulation** (we do not run on quantum hardware). After fixing a missing physical scaling factor in an earlier version of our code, the remaining QMC error mainly comes from Fourier truncation and finite-shot noise.

Table 6 summarizes the MC and QMC errors for I_k over 100 synthetic KL profiles. The reference value I_k^{ref} is computed using the trapezoid rule on the mesh. Classical MC uses $N_{\mathrm{MC}} = 16384$ uniform samples in x per profile. QMC uses the same settings described in Section 7. **Take-away.** At the settings in Table 6,

Metric (100 profiles)	**Classical MC**	**QMC**
Mean absolute error vs. trapezoid ref.	8.0×10^{-4}	1.8×10^{-3}
Median absolute error vs. trapezoid ref.	6.0×10^{-4}	1.3×10^{-3}
Max absolute error vs. trapezoid ref.	2.4×10^{-3}	1.28×10^{-2}
Mean signed error ($\hat{I}_k - I_k^{\mathrm{ref}}$)	-1.0×10^{-4}	$+1.0 \times 10^{-4}$
RMSE of signed error	1.0×10^{-3}	2.6×10^{-3}
Profiles with $\lvert\hat{I}_k^{\mathrm{QMC}} - I_k^{\mathrm{ref}}\rvert < \lvert\hat{I}_k^{\mathrm{MC}} - I_k^{\mathrm{ref}}\rvert$	—	26/100

Table 6: MC and QMC error summary for the scalar QoI I_k over 100 KL profiles.

MC is closer to the trapezoid reference on average, but QMC is better for 26 out of 100 profiles. All QMC results here are from **classical simulation**, so we are not claiming any quantum hardware speedup.

Table 7 and Figure 6 answer a different question: how much computation is needed to reach a target RMSE. In this benchmark, QMC reaches the listed RMSE targets using fewer reported QMC calls than the number of MC samples. At the largest tested setting, QMC also has a slightly smaller RMSE (2.78×10^{-4} W/K) than MC (3.18×10^{-4} W/K at $N_{\mathrm{MC}} = 131072$).

RMSE ladder (100 KL profiles). We increase the MC sample count and the QMC shots step by step (roughly doubling each time) on the same 100 profiles. Table 7 shows the first setting that reaches each RMSE target, and Figure 6 shows the full curves.

Target RMSE vs. trapz (W/K)	**MC: N_{MC}**	**QMC: shots**	**QMC: oracle calls (shots×depth)**
2.0×10^{-3}	4096	128	2048
1.0×10^{-3}	16384	512	8192
5.0×10^{-4}	65536	1024	16384

Table 7: First setting that reaches each RMSE target (100 KL profiles).

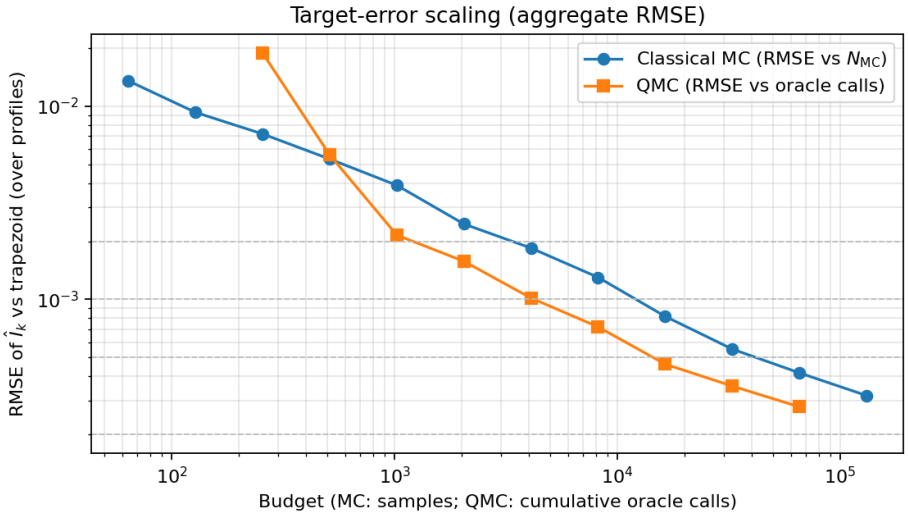


Figure 6: RMSE of $\hat{I}_k$ vs. trapezoid reference (100 KL profiles): MC vs. N_{MC} and QMC vs. reported QMC oracle calls.

Note on cost. MC samples and QMC oracle calls are not the same real hardware cost metric, and our QMC runs are classical simulations. We use these results only to compare errors under the stated settings.

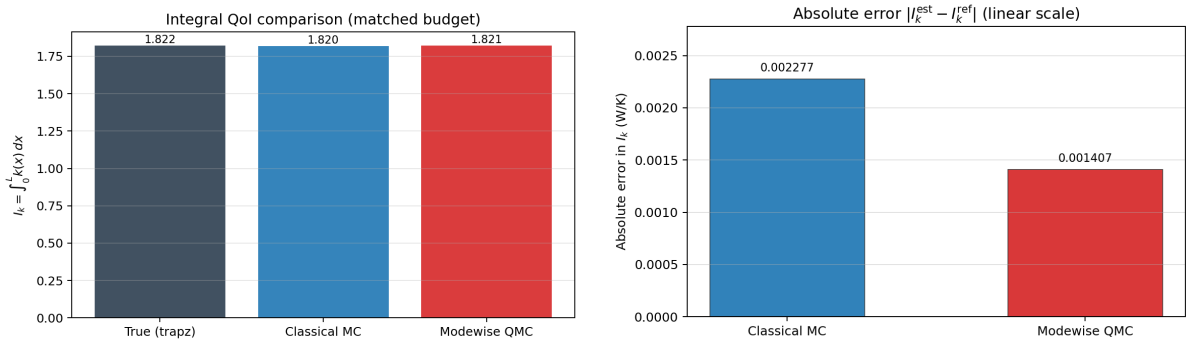


Figure 7: Example I_k comparison on one profile: trapezoid reference, MC, and QMC (left), and their absolute errors (right). Implementation details are in Section 7, and aggregate results are in Table 6.

8 Sensitivity Analysis and Future Controlled Comparisons

Three simple observations from this benchmark are:

1. If the loss weights are not tuned well, data-only training can perform better than mixed data+physics training.

2. Sensor layout and measurement noise strongly affect both inversion accuracy and the width of uncertainty bands.

3. The prior choice (KL kernel, variance, and number of modes) directly affects how identifiable $k(x)$ is, so it should always be reported.

These points suggest future work on better loss-weight schedules, sensor studies, and prior-sensitivity sweeps.

Noisy data in the PDE residual. Our PDE residual is computed using the noisy temperatures $\widetilde{T}$. When noise is not small, this term can punish the noise itself, not just physics mismatch. This helps explain

why mixed data+physics training may underperform unless we schedule the physics weight carefully or denoise $\widetilde{T}$ before computing residuals.

Spectral PINO-inverse. We also tested a spectral inverse model (Figure 4) for controlled comparisons. For the default slab setup it did not give a consistent improvement over SimplePINO-inverse, so we do not include a separate results table here.

9 Discussion

We use the workflow in two steps: (i) SimplePINO-inverse gives fast estimates of $k(x)$ for many cases, and (ii) for a few selected cases we can run a slower optimization-based inverse solver (BFGS-inverse) as a verification step.

Sparse sensors. The main benchmark assumes temperatures are available on the full (N_t, N_x) grid. With thermocouples, we may only have a few sensors and irregular times. A simple workaround is to fill a full grid by interpolation and then apply the trained model. This can add extra error at low sensor counts, so future studies should report results versus the number of sensors and the interpolation method.

Boundary conditions. Here we assume the boundary-condition family (Dirichlet/Neumann/Robin) used to generate $T(x,t)$ is known. If boundary inputs like $q''(t)$ or Robin parameters are wrong, the inverse problem can change a lot. This should be tested in controlled sweeps or handled as a joint estimation problem.

Prior shift. The inverse model is trained on a KL prior. If the real conductivity fields do not follow this prior, accuracy can drop. For practical use, we should test changes in correlation length, variance, and mode count.

MC vs QMC for I_k. We compare MC and QMC in two ways. First, at the fixed settings in Table 6, MC has lower mean absolute error on average, but QMC is better for 26 out of 100 profiles. Second, in Table 7 we increase computation until we reach a target RMSE, and QMC reaches the listed RMSE targets at smaller reported QMC call counts than the number of MC samples. In all cases, QMC results are from classical simulation, so they should be read as algorithmic diagnostics, not as a hardware speedup claim.

10 Limitations

This study has six main limitations. (1) The results use a synthetic 1D slab with fixed KL and noise settings, and we do not yet include a full prior-sensitivity study. (2) The forward and inverse models use the same discretized physics, which may make uncertainty look smaller than it would in experiments with contact resistance, emissivity errors, or sensor bias. (3) We do not yet compare against experimental temperature data. (4) Training with mixed data and physics losses is sensitive to the loss-weight schedule. (5) The QMC results are for a finite budget and are obtained from classical simulation of the quantum estimator; at the reported budget, QMC is less accurate than matched classical MC on average for I_k (Section 7). (6) Extended implementation notes, code scope, and supplementary numerical tables for the RES-VSSC study are documented separately [35].

11 Conclusions

This paper presents a workflow for uncertainty-aware inverse heat transfer. We combine (i) a high-fidelity forward solver with KL-based random conductivity fields, (ii) a physics-informed inverse operator—**SimplePINO-inverse** by default, with optional **PINO-inverse** for spectral comparisons—to estimate $k(x)$ from noisy temperature data $T(x,t)$, and (iii) classical MC and a QMC-based estimator to study uncertainty at controlled budgets. The framework is demonstrated on a one-dimensional synthetic slab that represents key features of thermal protection system testing.

In this benchmark, SimplePINO-inverse gives fast estimates of $k(x)$, and classical MC is used as the main reference for uncertainty estimates. We also report QMC results for the integral QoI I_k (Tables 6

and 7, Figure 6); all QMC results are from classical simulation. Future work will include tests on experimental data, better tuning of mixed data+physics losses, and broader tests under distribution shift.

References

[1] Rochisha Agarwal, Ferdin Don Bosco, Ramesh Kolluru, Rut Lineswala, Abhishek Chopra, and René Steijl. Utilization of quantum monte carlo algorithm for uncertainty quantification. In *AIAA Aviation Forum and ASCEND 2025*, Las Vegas, Nevada, July 2025. AIAA 2025-3041.

[2] Kyriakos C. Giannakoglou, Dimitrios I. Papadimitriou, Evangelos M. Papoutsis-Kiachagias, and Carsten Othmer. Adjoint methods in CFD-based optimization—gradient computation & beyond. In *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, pages 10–24, 2012.

[3] Zongyong Pan and Xiaomin Pan. Deep learning and adjoint method accelerated inverse design in photonics: a review. *Photonics*, 10(7):852, 2023.

[4] Dhruvil Panchigar, Kunal Kar, Shashank Shukla, Rhea Mary Mathew, Utkarsh Chadha, and Senthil Kumaran Selvaraj. Machine learning-based CFD simulations: a review, models, open threats, and future tactics. *Neural Computing and Applications*, 34(24):21677–21700, 2022.

[5] J. Nathan Kutz. Machine learning for parameter estimation. *Proceedings of the National Academy of Sciences*, 120(12):e2300990120, 2023.

[6] Navid Zobeiry and Keith D. Humfeld. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Engineering Applications of Artificial Intelligence*, 101:104232, 2021.

[7] Apostolos F. Psaros, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 477:111902, 2023.

[8] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[9] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural Safety*, 31(2):105–112, 2009.

[10] Karthik Reddy Lyathakula. Statistical design of thermal protection system using physics-informed machine learning. arXiv preprint arXiv:2501.10825, 2025.

[11] Silvester Sabathiel, Hèlios Sanchis-Alepuz, Andrew S. Wilson, Jacob Reynvaan, and Monika Stipsitz. Neural network-based reconstruction of steady-state temperature systems with unknown material composition. *Scientific Reports*, 14(1):22265, 2024.

[12] J. V. Beck, B. Blackwell, and C. R. Clair. *Inverse Heat Conduction: Ill-Posed Problems*. Wiley-Interscience, 1985.

[13] M. N. Özisik and H. R. B. Orlande. *Inverse Heat Transfer: Fundamentals and Applications*. Taylor & Francis, 2000.

[14] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-Posed Problems*. John Wiley & Sons, 1977.

[15] O. M. Alifanov, E. A. Artyukhin, and S. V. Rumyantsev. *Extreme Methods for Solving Ill-Posed Problems with Applications to Inverse Heat Transfer Problems*. Begell House, 2013.

[16] M. Mobtil, D. Bougeard, and S. Russeil. Experimental study of inverse identification of unsteady heat transfer coefficient in a fin and tube heat exchanger assembly. *International Journal of Heat and Mass Transfer*, 125:17–31, 2018. doi: 10.1016/j.ijheatmasstransfer.2018.04.028.

[17] Honeyeh Razzaghi, Farshad Kowsary, and Mohammad Layeghi. Inverse boundary problem in estimating heat transfer coefficient of a round pulsating bubbly jet: design of experiment. *Applied Mathematics in Science and Engineering*, 30(1):210–234, 2022. doi: 10.1080/27690911.2022.2045983.

[18] B. Hadała, Z. Malinowski, A. Gołdasz, and A. Cebo-Rudnicka. Inverse solution to the vertical plate cooling by radiation and convection in air. *Archives of Metallurgy and Materials*, pages 1167–1178, 2022. doi: 10.24425/amm.2022.139717.

[19] Lauren B. Tomanek and Daniel S. Stutts. Data on the validation to determine the material thermal properties estimation via a one-dimensional transient convection model. *Data in Brief*, 40:107632, 2022. doi: 10.1016/j.dib.2021.107632.

[20] Martha Janeth Sanabria Guerrero. Comparison study of numerical methods applied for estimation of heat transfer coefficient during quenching. https://www.academia.edu/24923676/Comparison_Study_of_Numerical_Methods_Applied_for_Estimation_of_Heat_Transfer_Coefficient_During_Quenching, 2016. Accessed: 2026-04-26.

[21] Heatlab. Inverse heat conduction problem. https://www.heatlab.cz/research/inverse-heat-conduction-problem/, 2026. Accessed: 2026-04-26.

[22] M. Raissi, P. Perdikaris, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.

[23] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

[24] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021.

[25] Shanshan Deng et al. Approximation and learning rates for deeponet-type operator networks. Manuscript / preprint—verify before camera-ready, 2022.

[26] Lu Lu et al. Multifidelity neural operators for nanoscale heat transport. Manuscript / preprint—verify before camera-ready, 2022.

[27] C. Lin et al. Deep operator networks for multiscale bubble growth dynamics. Manuscript / preprint—verify before camera-ready, 2021.

[28] N. B. Kovachki, S. Lanthaler, and S. Mishra. Neural operator: Learning maps between function spaces. *Acta Numerica*, 32:1–97, 2023.

[29] P. C. Di Leoni et al. Neural operators for instability-wave prediction in high-speed boundary layers. Manuscript / preprint—verify before camera-ready, 2023.

[30] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[31] Ashley Montanaro. Quantum speedup of Monte Carlo methods. *Proceedings of the Royal Society A*, 471(2181):20150301, 2015. doi: 10.1098/rspa.2015.0301.

[32] James V. Beck, B. Blackwell, and Charles R. St. Clair. Methodology for comparison of inverse heat conduction methods. *Journal of Heat Transfer*, 110(1):30–39, 1988. URL https://asmedigitalcollection.asme.org/heattransfer/article/110/1/30/413155/Methodology-for-Comparison-of-Inverse-Heat.

[33] Helcio R. B. Orlande. Inverse problems in heat transfer: New trends on solution methodologies and applications. In *14th International Heat Transfer Conference (IHTC14), Volume 8*, pages 379–398, Washington, DC, USA, 2010. doi: 10.1115/IHTC14-23349.

[34] J. Krejsa, L. Slama, J. Horsky, M. Raudensky, and B. Patikova. A comparison of traditional and non-classical methods for solving inverse heat conduction problem. In *Advanced Computational Methods in Heat Transfer*, volume 12 of *WIT Transactions on Engineering Sciences*. WIT Press, 1996. URL https://www.witpress.com/Secure/elibrary/papers/HT96/HT96043FU.pdf.

[35] Ramesh Kolluru. Inverse heat transfer with physics-informed neural operators (pino): Technical report, code scope, results, and work plan. Technical report, RES-VSSC-2025-025, February 2026.