OPERATOR SENSITIVITY AS AN INDUCTIVE PRIOR FOR UNCERTAINTY QUANTIFICATION IN DEEPONETS

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate whether the input–output sensitivity of a learned operator can provide an inductive prior for the spatial structure of predictive uncertainty. We introduce Jacobian-DeepONet, which parameterizes asymmetric prediction intervals using the normalized pointwise Jacobian norm and two additional trainable scalars, jointly optimized with the operator under mean-squared-error and pinball losses. We also introduce Jacobian Conformal Bands (JCB), which calibrate sensitivity-shaped intervals on a frozen operator without retraining. Both methods exploit the Cartesian DeepONet factorization to compute Jacobian norms exactly through a latent Gram matrix using K reverse-mode vector–Jacobian products, without materializing the full input–output Jacobian. On viscous Burgers, Darcy flow, and compressible Navier–Stokes, the methods achieve competitive interval widths after calibration and stronger alignment with reference physical sensitivities than constant-width and learned-quantile baselines. Monte Carlo perturbations validate the first-order variance approximation, while experiments with noisy inputs show improved coverage retention for JCB relative to conditional quantile bands under the tested perturbations. Our results suggest that operator sensitivity provides useful, problem-dependent structure for prediction intervals in DeepONets, complementing uncertainty models learned from residuals.

1 INTRODUCTION

Partial differential equations (PDEs) govern a wide range of physical phenomena, yet their numerical solution remains computationally expensive when repeated evaluations under varying parameters are required. Neural operators, notably Deep Operator Networks (DeepONets) (Lu et al., 2021) and Fourier Neural Operators (Li et al., 2020), learn mappings between infinite-dimensional function spaces, amortizing computational costs for real-time simulation, design optimization, and data-driven control.

While neural operators deliver fast surrogate evaluations, their standard predictions remain deterministic point estimates devoid of confidence measures. In safety-critical engineering, predictions without reliable uncertainty quantification (UQ) present severe operational risks. Established UQ methodologies, such as variational Bayesian approximations (Garg & Chakraborty, 2023) or post-hoc conformal calibration (Moya et al., 2025), either demand expensive sampling or produce spatially uniform bounds that ignore where physical sensitivities concentrate. Conversely, sensitivity analysis characterizes how outputs respond to input variations without producing calibrated prediction intervals.

In this work, we propose that the analytical input-output sensitivity (Jacobian) of a learned operator serves as an effective inductive prior for the spatial distribution of predictive uncertainty. In nonlinear dynamical systems, regions exhibiting elevated parametric sensitivity empirically align with areas of heightened surrogate error, evidenced by a rank correlation of 0.746 between Jacobian-derived widths and absolute prediction errors in the viscous Burgers equation. Rather than treating physical sensitivity and prediction error as identical, we investigate whether sensitivity geometry can be harnessed to structure predictive intervals.

A major computational impediment to sensitivity-based UQ in neural operators is that materializing the full input-output Jacobian $\mathcal{J}_u \in \mathbb{R}^{N_{\text{nodes}} \times N_{\text{in}}}$ is intractable for dense spatial grids. We resolve

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

this by exploiting the Cartesian product factorization of DeepONets: because spatial coordinates are isolated within the trunk network, all parametric differentiation reduces to the branch network. Pointwise variance can thus be extracted through a compact $K \times K$ latent Gram matrix via K reverse-mode vector-Jacobian products (VJPs), reducing memory complexity by a factor of N_{nodes}/K ($41\times$ on Darcy, $128\times$ on Burgers) without materializing the full Jacobian.

Building on this sensitivity prior, we introduce two complementary operational methods:

1. **Method 1: Jacobian-DeepONet (End-to-End).** We parameterize asymmetric prediction intervals directly by the normalized Jacobian norm scaled by two learnable scalars, trained jointly under mean-squared error and pinball loss. Backpropagating through the Jacobian norm actively shapes operator representations and induces an implicit second-order derivative regularization. This method is suited for training new models where joint interval optimization and derivative control are desired, incurring an approximate $4\times$ training cost.
2. **Method 2: Jacobian Conformal Bands (JCB, Post-Hoc).** For large, pretrained operators where retraining is computationally prohibitive, JCB extracts the identical sensitivity template from a frozen deterministic operator and rescales it using split conformal prediction with a non-zero relative floor. JCB requires zero retraining and adds zero parameters, providing immediate, input-conditioned uncertainty bounds for off-the-shelf surrogate models.

We systematically benchmark both formulations against constant-width, conditional-quantile, and conformal baselines across three canonical PDE benchmarks: 1D viscous Burgers, 2D Darcy flow, and 2D compressible Navier-Stokes. At matched empirical coverage, the Jacobian prior follows the true physical sensitivity profile more closely than unconditioned or data-driven baselines ($\rho_{\text{Sens}} = 0.628$ vs 0.477 on Burgers, 0.612 vs 0.573 on Darcy, 0.525 vs 0.479 on Navier-Stokes). Furthermore, Jacobian-DeepONet achieves interval widths comparable to flexible deep quantile heads while incurring a substantially lower penalty on central prediction accuracy (14% vs 21% on Burgers), and on frozen operators JCB yields intervals 5% and 7% narrower than the best constant-shape conformal band on Burgers and Navier-Stokes.

In summary, our principal contributions are:

1. **Operator Sensitivity as an Inductive Prior:** We propose the input–output Jacobian norm as an inductive prior for the spatial shape of prediction intervals. We show that Cartesian DeepONet separability enables exact evaluation of these norms through a latent Gram matrix using K reverse-mode vector–Jacobian products, without forming the full input–output Jacobian.
2. **Two Complementary Methods and Trade-off Analysis:** We propose (i) **Jacobian-DeepONet**, an end-to-end framework coupling a mean squared error (MSE) objective and pinball loss with implicit derivative regularization, and (ii) **Jacobian Conformal Bands (JCB)**, a post-hoc conformalization method for frozen operators requiring zero retraining. We characterize their respective trade-offs in computational cost, representational adaptation, and operational flexibility.
3. **Rigorous Same-Backbone Empirical Validation:** We conduct an extensive audit across three PDE regimes under matched backbones, training budgets, and conformalized coverage over three seeds, confirming that the analytical sensitivity prior outperforms unconditioned and learned-width baselines in physical alignment, at central accuracy equal to or better than the learned-width head on two benchmarks of three.

2 Related Work

Neural operators. Neural operators learn mappings between infinite-dimensional function spaces rather than solutions of a single boundary value problem, amortizing the cost of repeated parametric evaluation. Fourier Neural Operators (Li et al., 2020) parameterize the kernel in spectral space, while DeepONets (Lu et al., 2021; Clark Di Leoni et al., 2022) realize the universal operator approximation theorem by decoupling the encoding of the input function (branch) from the evaluation coordinates (trunk). That separability is what this work exploits: it confines parametric differentiation to the branch network and makes the latent Jacobian extractable in K reverse-mode vector-Jacobian products, independent of the number of evaluation nodes (Subsection 3.1).

Uncertainty quantification for neural operators. Predictive uncertainty is usually split into aleatoric and epistemic components (Kendall & Gal, 2017; Zhang, 2020). For DeepONets the established routes are Bayesian and conformal: VB-DeepONet (Garg & Chakraborty, 2023) places priors over the weights and approximates the posterior variationally, and Conformalized-DeepONet (Moya et al., 2025) applies inductive conformal prediction (Vovk et al., 2004; Romano et al., 2019) for finite-sample, distribution-free coverage. Both calibrate or sample; neither uses the operator's own parametric sensitivity to decide where the band should be wide. Closest to the present formulation, Magnani et al. (2024) linearize a trained operator into a function-valued Gaussian process, and O'Leary-Roseberry et al. (2024) show that derivative-informed training improves operator surrogates; Stent & Boullé (2026) combine conformal guarantees with operator surrogates. We differ in using the operator Jacobian as an inductive prior on interval *shape*, operationalized both via joint end-to-end training and through post-hoc conformal calibration on frozen operators.

Deep quantile regression. Quantile regression (Koenker, 2005) builds intervals without a Gaussian assumption, and its deep variants (Tagasovska & Lopez-Paz, 2019; Padilla et al., 2020) learn the width directly from data. Applied to PDE operators it is the natural competitor to a sensitivity prior, and the fair-comparison baselines of Subsection 5.1 are exactly this family at matched backbone and budget. Unconstrained quantile heads can cross, and an unconditioned head collapses to a spatially uniform width, which is the failure the sensitivity prior is meant to avoid.

Sensitivity, automatic differentiation and implicit regularization. Parametric sensitivity is classically obtained by adjoint methods or automatic differentiation (AD) (Ma et al., 2021; Rackauckas et al., 2020). Sobolev training (Czarnecki et al., 2017) penalizes the error between network and target derivatives, but requires ground-truth derivative labels. With respect to the use of sensitivity, SC-FNO (Behroozi et al., 2025) and its extension (Behroozi et al., 2026) constrain a Fourier neural operator by matching selected sensitivities supplied by a differentiable solver or a discrete adjoint, which improves forward accuracy and, more markedly, gradient-based inverse reconstruction. We differ at both ends: the sensitivity we use is the network's own, obtained by AD and never supervised, and it parameterizes the width of a prediction interval rather than the accuracy of the central field. Our formulation induces a related second-order effect without any derivative supervision: backpropagating through the Jacobian norm that scales the pinball loss constrains the network's internal derivatives as a by-product of fitting the interval (Appendix A).

3 Methodology

3.1 Analytical Operator Sensitivity as an Inductive Prior

We adopt the Cartesian Product DeepONet (Lu et al., 2021) as our backbone architecture. Unlike pointwise formulations that evaluate coordinates sequentially, Cartesian DeepONets evaluate the full solution field over a spatial grid $X = \{x_j\}_{j=1}^{N_{\text{nodes}}}$ in a single forward pass:

$$\mathcal{G}_\theta(u)(X) = B_\theta(u) \cdot T_\theta(X)^T + b, \tag{1}$$

where $B_\theta(u) \in \mathbb{R}^{N_b \times K}$ is the branch network encoding N_b input parameter fields, $T_\theta(X) \in \mathbb{R}^{N_{\text{nodes}} \times K}$ is the trunk network evaluated across spatial coordinates, K is the shared latent dimension, and $b \in \mathbb{R}$ is a scalar bias.

Physical sensitivity via the Delta Method. Consider an input function subject to perturbation $\tilde{u} = u + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_{N_{\text{in}}})$. Linearizing the operator via a first-order multivariate Taylor expansion (Oehlert, 1992) yields $\mathcal{G}_\theta(u + \varepsilon)(x_j) \approx \mathcal{G}_\theta(u)(x_j) + \mathcal{J}_u^{(j)} \cdot \varepsilon$, where $\mathcal{J}_u^{(j)} = \partial \mathcal{G}_\theta(u)(x_j)/\partial u \in \mathbb{R}^{1 \times N_{\text{in}}}$. The pushforward predictive variance evaluates to first order in σ^2 as:

$$\text{Var}_\Delta(x_j) = \mathbb{E}[(\mathcal{J}_u^{(j)} \cdot \varepsilon)^2] = \sigma^2 \|\mathcal{J}_u^{(j)}\|_2^2. \tag{2}$$

Thus, the analytical Jacobian norm $\sigma^2 \|\mathcal{J}_u\|_2^2$ provides a first-order estimate of physical pushforward variance without test-time stochastic sampling (verified in Appendix F).

Latent Jacobian extraction in $\mathcal{O}(K)$. Forming the dense Jacobian $\mathcal{J}_u \in \mathbb{R}^{N_{\text{nodes}} \times N_{\text{in}}}$ at each step is computationally prohibitive for high-resolution meshes. Cartesian separability resolves this bottleneck. Because the trunk depends only on spatial coordinates X, parametric differentiation is confined strictly to the branch network: $\mathcal{J}_u = T(X)\mathcal{J}_{\text{branch}}$, where $\mathcal{J}_{\text{branch}} = \partial B_\theta(u)/\partial u \in \mathbb{R}^{K \times N_{\text{in}}}$. The pointwise variance reduces to the diagonal of a quadratic form over the compact $K \times K$ latent Gram matrix $\mathbf{G} = \mathcal{J}_{\text{branch}}\mathcal{J}_{\text{branch}}^T$:

$$V_{\text{pred}} = \sigma^2 \operatorname{diag}\left(T(X)\mathbf{G}T(X)^T\right), \qquad \sigma_{\text{pred}} = \sqrt{V_{\text{pred}}}. \tag{3}$$

We compute $\mathcal{J}_{\text{branch}}$ using K reverse-mode vector-Jacobian products (VJPs) through the branch network, vectorized via `vmap(jacrev)` in a single fused pass. The memory requirement drops by a factor of N_{nodes}/K ($41\times$ on Darcy, $128\times$ on Burgers) without materializing the dense $N_{\text{nodes}} \times N_{\text{in}}$ Jacobian (derivation in Appendix A). This provides a fast, input-dependent spatial sensitivity profile $s_{\text{J}}(\mathbf{x})$.

3.2 Method 1: End-to-End Jacobian-DeepONet

Residuals of nonlinear PDE operators exhibit heavy tails and asymmetry near shocks or interfaces that violate Gaussian assumptions. To capture this structure, we use the normalized sensitivity field as an inductive spatial prior for asymmetric prediction intervals trained under the pinball loss (Koenker, 2005):

$$s_{\text{J}}(\mathbf{x}) = \frac{\sigma_{\text{pred}}(\mathbf{x})}{\overline{\sigma}_{\text{EMA}}}, \qquad y_{\text{low/up}} = \mathcal{G}_\theta(u) \mp \text{Softplus}(w_{\text{low/up}}) \cdot s_{\text{J}}(\mathbf{x}), \tag{4}$$

where $\overline{\sigma}_{\text{EMA}}$ is an exponential moving average (EMA) of the sensitivity scale detached from the computational graph, guaranteeing batch independence at inference (ablation in Appendix H), and $w_{\text{low}}, w_{\text{up}} \in \mathbb{R}$ are two trainable scalars. The operator and its uncertainty bounds are optimized jointly:

$$\mathcal{L}_{\text{total}} = \text{MSE}(\mathcal{G}_\theta(u), y_{\text{true}}) + \lambda\left[\mathcal{L}_{\text{pinball}}(y_{\text{low}}, y_{\text{true}}, \alpha/2) + \mathcal{L}_{\text{pinball}}(y_{\text{up}}, y_{\text{true}}, 1-\alpha/2)\right], \tag{5}$$

where $\mathcal{L}_{\text{pinball}}(q, y, \tau) = \max(\tau(y-q), (\tau-1)(y-q))$.

Properties and trade-offs. The normalized Jacobian norm $s_{\text{J}}(\mathbf{x})$ acts as a structural prior allocating width where the operator is sensitive, while the learnable scalars scale the template to nominal coverage $(1-\alpha)$. Backpropagating through the Jacobian norm enables the pinball loss to actively shape the branch network representations. This induces an implicit second-order derivative regularization (Appendix A) without derivative supervision. In exchange, the higher-order autodiff computation incurs an approximate $4\times$ training wall-clock penalty and requires tuning the weight λ.

3.3 Method 2: Post-Hoc Jacobian Conformal Bands (JCB)

While Method 1 reshapes representations during training, the latent Gram matrix $\mathbf{G}(u)$ is an intrinsic feature of any differentiable Cartesian DeepONet, including models trained solely under mean squared error. For large-scale pretrained operators where retraining is prohibitive, JCB extracts the identical sensitivity prior post-hoc without modifying operator weights or adding parameters.

Sensitivity extraction and flooring. Given frozen weights θ, we evaluate $\mathbf{G}(u)$ via K first-order VJPs at inference with no gradient tracking (`torch.no_grad()`). Because physical sensitivity vanishes at Dirichlet boundary nodes, a raw sensitivity divisor would diverge. We therefore introduce a relative floor parameterized by $\beta > 0$:

$$h(u, \mathbf{x}) = \sigma_{\text{pred}}(u, \mathbf{x}) + \beta \,\text{median}(\sigma_{\text{pred}}). \tag{6}$$

Split conformal calibration. We calibrate the global scale $\hat{q}$ post-hoc on a calibration set $\mathcal{C}$ disjoint from test data using normalized nonconformity scores (Papadopoulos et al., 2008; Lei & Wasserman, 2014):

$$s_i = \frac{|y_i - \mathcal{G}_\theta(u)_i|}{h(u, \mathbf{x}_i)}, \qquad \hat{q} = \text{Quantile}_{1-\alpha}\left(\{s_i\}_{i \in \mathcal{C}}\right), \qquad y_{\text{low/up}} = \mathcal{G}_\theta(u) \mp \hat{q}\, h(u, \mathbf{x}). \tag{7}$$

Properties and trade-offs. JCB provides input-conditioned spatial uncertainty bands for off-the-shelf surrogate models at zero retraining cost and zero parameter overhead, requiring only a small calibration set. However, because the operator remains frozen, JCB cannot reshape representations or induce derivative regularization; it relies entirely on the natural sensitivity alignment of the pretrained model.

4 EXPERIMENTS

Benchmarks and backbone. We evaluate both methods across three PDE benchmarks spanning distinct mathematical regimes: 1D viscous Burgers (nonlinear advection and shock steepening), 2D Darcy flow (steady elliptic transport through heterogeneous log-permeability fields), and 2D compressible Navier-Stokes (hyperbolic-parabolic gas dynamics). Every model shares one Cartesian DeepONet backbone with Swish activations, Glorot-normal initialization, and latent dimension $K = 100$. Architectures, batch sizes, and dataset scales are detailed in Appendix B.

Training and evaluation protocol. All models on each benchmark are trained for 200 000 iterations using Adam in single precision with an exponentially decayed step schedule (factor 0.7 every $50\,000$ steps). Variables are z-score normalized during training; all reported metrics are in physical units. Nominal coverage is set to 90% ($\alpha = 0.10$). We report MSE, empirical prediction interval coverage probability (PICP), mean prediction interval width (MPIW), Gaussian proxy negative log-likelihood (NLL), and Spearman rank correlations against absolute error (ρ_{Err}) and physical reference sensitivity (ρ_{Sens}).

Baselines. Under the same-backbone protocol (Subsection 5.1), we compare five configurations sharing identical architecture, data, and optimizer: (i) a deterministic operator, (ii) a constant-width pinball band, (iii) a spatial quantile head conditioned on coordinates $\mathbf{x}$, (iv) a conditional quantile head conditioned on $(u, \mathbf{x})$, and (v) Jacobian-DeepONet. To ensure fair sharpness comparisons, all models are evaluated after split-conformal calibration on a held-out half of the test split, with metrics reported over three random seeds.

5 RESULTS

We evaluate the sensitivity hypothesis across three PDE regimes: 1D viscous Burgers, 2D Darcy flow, and 2D compressible Navier-Stokes. Table 7 summarizes cross-benchmark metrics, Subsection 5.1 presents the same-backbone comparison, and Subsection 5.2 evaluates post-hoc calibration on frozen operators (Method 2). Per-benchmark diagnostics, error-decile decompositions, Monte Carlo comparisons, and ablations are detailed in Appendices E, F, and H.

5.1 SAME-BACKBONE COMPARISON AGAINST QUANTILE AND CONFORMAL BASELINES

A deterministic operator isolates the cost of adding intervals and leaves open whether the Jacobian is the right prior for their spatial shape, while a constant-width band holds its width fixed across space by construction. Testing the prior itself requires a baseline that learns the width from data on the same backbone and budget. Table 1 reports that comparison against the conditional quantile head; the constant-width and spatial-quantile baselines are in Table 8. Every model shares the backbone architecture, latent dimension K, dataset, optimizer, schedule, and iteration count. All widths are compared after split-conformal calibration on a held-out test half to ensure matched coverage.

Three primary findings emerge from the three-seed evaluation. First, at matched conformal coverage, the Jacobian prior achieves competitive sharpness: it is 5.7% sharper than the conditional quantile head on Burgers ($(3.98 \pm 0.02) \times 10^{-2}$ vs $(4.22 \pm 0.11) \times 10^{-2}$) and 3.5% sharper on Navier-Stokes ($(0.891 \pm 0.006) \times 10^{-1}$ vs $(0.923 \pm 0.012) \times 10^{-1}$), but 6.0% wider on Darcy ($(3.37 \pm 0.09) \times 10^{-3}$ vs $(3.18 \pm 0.18) \times 10^{-3}$). Second, the methods differ markedly in the accuracy penalty imposed on the surrogate. Relative to a deterministic baseline, Jacobian-DeepONet degrades Burgers MSE by 14%, whereas the conditional quantile head incurs a 21% penalty; on Darcy and Navier-Stokes neither pays a penalty: both stay within 2% of the deterministic baseline, except the quantile head on Darcy, which is 9% more accurate. This degradation reflects representational partitioning: the conditional quantile

Table 1: Same-backbone comparison, three seeds. PICP is as trained; MPIW follows a split-conformal multiplier fitted on one half of the test set and is measured on the other, so widths are compared at equal coverage. Darcy's ρ_{Sens} comes from a separate 500-sample sensitivity set. Proxy-NLL, cost and the remaining two baselines are in Table 8.

Model	**MSE**	**PICP (%)**	**MPIW@90%**	ρ_{Err}	ρ_{Sens}
Burgers 1D (MSE $\times 10^{-3}$*, MPIW* $\times 10^{-2}$*)*					
Deterministic	1.537 ± 0.014	N/A	N/A	N/A	N/A
Conditional quantile	1.865 ± 0.013	88.7 ± 0.9	4.22 ± 0.11	0.806 ± 0.004	0.477 ± 0.001
Jacobian-DeepONet	1.757 ± 0.044	93.4 ± 0.1	3.98 ± 0.02	0.746 ± 0.005	0.628 ± 0.001
Darcy 2D (MSE $\times 10^{-6}$*, MPIW* $\times 10^{-3}$*)*					
Deterministic	3.01 ± 0.11	N/A	N/A	N/A	N/A
Conditional quantile	2.73 ± 0.11	81.5 ± 0.3	3.18 ± 0.18	0.661 ± 0.003	0.573 ± 0.009
Jacobian-DeepONet	3.04 ± 0.44	83.2 ± 0.8	3.37 ± 0.09	0.614 ± 0.010	0.612 ± 0.006
Navier–Stokes 2D (MSE $\times 10^{-3}$*, MPIW* $\times 10^{-1}$*)*					
Deterministic	1.094 ± 0.009	N/A	N/A	N/A	N/A
Conditional quantile	1.081 ± 0.011	87.5 ± 0.1	0.923 ± 0.012	0.514 ± 0.002	0.479 ± 0.002
Jacobian-DeepONet	1.076 ± 0.015	88.1 ± 0.2	0.891 ± 0.006	0.519 ± 0.005	0.525 ± 0.001

head expands the latent dimension to $3K$ (adding 2.0×10^4 parameters), while Jacobian-DeepONet adds only two scalar multipliers to the existing latent space. On Burgers, Jacobian-DeepONet yields 5.7% sharper bands surrounding a prediction that is 5.8% more accurate. Third, the Jacobian width follows the reference physical sensitivity more faithfully than any baseline across all benchmarks: $\rho_{\text{Sens}} = 0.628 \pm 0.001$ vs 0.477 ± 0.001 on Burgers, 0.612 ± 0.006 vs 0.573 ± 0.009 on Darcy, and 0.525 ± 0.001 vs 0.479 ± 0.002 on Navier-Stokes. The constant-width band merely replicates the training variance floor $(0.177, 0.485, 0.008)$, confirming that the Jacobian provides sensitivity structure of its own. The quantile head maintains a higher ρ_{Err} because it optimizes empirical quantiles directly, reflecting a natural division of labor between error-fitted and sensitivity-informed objectives.

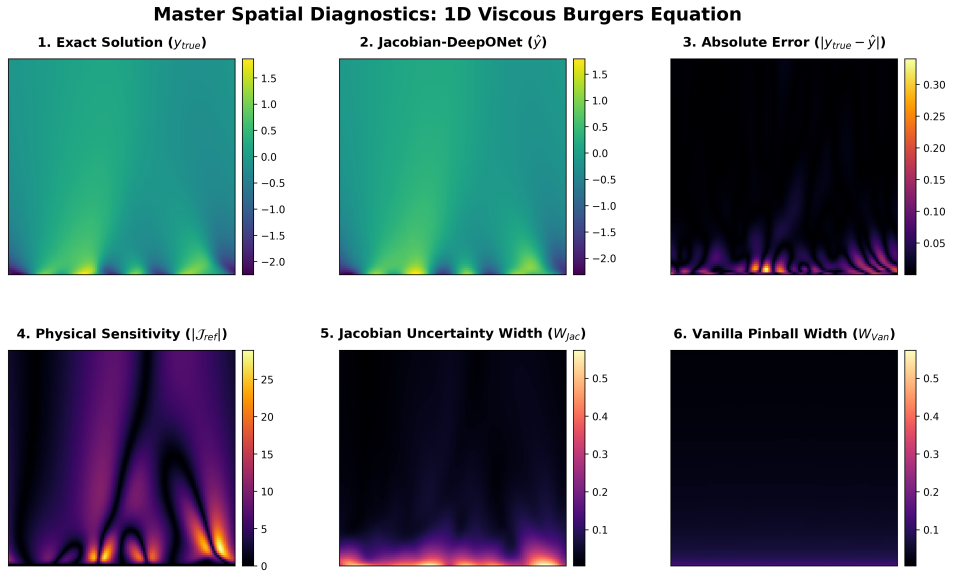


Figure 1: Spatial diagnostics for 1D Burgers over $(x, t) \in [0, 1]^2$ (Jacobian-DeepONet vs. constant-width at $\lambda = 4$, seed 0, 75th percentile test MSE). Top: exact solution, prediction, absolute error. Bottom: reference sensitivity $|\partial u/\partial \nu|$, Jacobian width W_{Jac}, and constant-width W_{CW} on a shared scale $[0.004, 0.574]$. The Jacobian prior dynamically captures the shock geometry, whereas the constant-width band is spatially uniform.

Coverage as trained. No parameterization achieves nominal 90% coverage on Darcy (81.0% to 83.2%) or Navier-Stokes (87.5% to 88.1%) prior to calibration, whereas on Burgers coverage reaches 87.6% to 93.4% (where Jacobian-DeepONet slightly overshoots). This reflects a characteristic of the pinball loss on these problems rather than the Jacobian prior, which attains the closest-to-nominal coverage on Darcy and Navier-Stokes. Split-conformal calibration reliably restores nominal coverage across all models.

Sensitivity to the quantile weight. A sweep over $\lambda \in \{1, 4, 8, 20\}$ on all three benchmarks (Table 6, Appendix C) shows that Jacobian-DeepONet maintains a lower interval score than constant-width baselines across all weights (15% to 43% improvement). While λ scales width and coverage (7% width expansion on Burgers), rank correlations remain invariant within ± 0.02, confirming that sensitivity geometry is structurally determined rather than an artifact of hyperparameter tuning.

Computational cost. Jacobian-DeepONet adds two scalars versus 2.0×10^4 parameters for the split-branch quantile head. Per-sample inference latency on Burgers is 0.284 ms versus 0.087 ms for the quantile head, while uniquely supplying the analytical sensitivity field. Training on an A100 GPU requires 58 minutes versus 17 minutes for the quantile head (Appendix C).

5.2 Bands Without Training: Evaluating JCB (Method 2)

Method 2 investigates whether the intrinsic geometry of a frozen deterministic operator provides calibrated uncertainty bands without retraining. Table 2 reports conformalized intervals on a frozen operator across different shape templates. On Burgers and Navier-Stokes, the Jacobian template narrows interval width by 5% and 7% relative to the best constant-shape conformal band, reduces interval score by 25% and 14%, and elevates sensitivity correlation from 0.187 to 0.647 on Burgers and from 0.006 to 0.499 on Navier-Stokes. This demonstrates that the frozen operator's own Jacobian captures most of the sensitivity alignment of a fully trained Jacobian-DeepONet (0.499 vs 0.525 on Navier-Stokes) with zero gradient updates and zero added parameters. On Darcy, the static training variance field captures dominant spatial variations ($\rho_{\text{Sens}} = 0.485$), yet the frozen Jacobian still achieves the highest physical sensitivity alignment (0.754 vs 0.485), although it is the widest of the three bands there. Thus, post-hoc conformalization offers an immediate, parameter-free route to sensitivity-informed UQ for existing neural operators.

Table 2: Conformal bands on a frozen deterministic operator trained under MSE alone. Within a block all rows share the weights and the split-conformal calibration and differ only in the shape of the half-width, so rows compare shape alone. Row 2 is the per-node training standard deviation; on Navier-Stokes it is almost uniform and coincides with row 1. Three seeds; Darcy's ρ_{Sens} from the sensitivity set.

Band shape	**PICP (%)**	**MPIW@90%**	**Interval score**	ρ_{Err}	ρ_{Sens}
Burgers 1D (MPIW and interval score $\times 10^{-2}$)					
Constant width	90.2 ± 0.0	5.43 ± 0.10	16.87 ± 0.13	N/A	N/A
Constant $\times\ \sigma_{\text{train}}$	90.4 ± 0.2	4.44 ± 0.10	14.31 ± 0.10	0.497 ± 0.009	0.187 ± 0.000
Jacobian	90.2 ± 0.2	$\mathbf{4.21 \pm 0.08}$	$\mathbf{10.70 \pm 0.17}$	$\mathbf{0.654 \pm 0.008}$	$\mathbf{0.647 \pm 0.008}$
Darcy 2D (MPIW and interval score $\times 10^{-3}$)					
Constant width	91.1 ± 0.3	4.30 ± 0.04	7.54 ± 0.08	N/A	N/A
Constant $\times\ \sigma_{\text{train}}$	91.2 ± 0.2	$\mathbf{3.89 \pm 0.04}$	$\mathbf{6.95 \pm 0.08}$	$\mathbf{0.425 \pm 0.002}$	0.485 ± 0.000
Jacobian	91.4 ± 0.4	4.36 ± 0.08	7.18 ± 0.05	0.405 ± 0.005	$\mathbf{0.754 \pm 0.009}$
Navier–Stokes 2D (MPIW and interval score $\times 10^{-1}$)					
Constant width	88.3 ± 0.2	0.946 ± 0.003	1.636 ± 0.010	N/A	N/A
Constant $\times\ \sigma_{\text{train}}$	88.3 ± 0.2	0.946 ± 0.003	1.636 ± 0.010	0.006 ± 0.004	0.006 ± 0.000
Jacobian	88.8 ± 0.1	$\mathbf{0.883 \pm 0.004}$	$\mathbf{1.408 \pm 0.010}$	$\mathbf{0.420 \pm 0.005}$	$\mathbf{0.499 \pm 0.009}$

5.3 Robustness to Input Uncertainty

The comparison so far assumes the input is known exactly. It rarely is: the coefficient field is measured, the initial condition estimated. Table 3 calibrates every band once on clean data, freezes

the multiplier, feeds the model a perturbed input and asks whether the band still contains the true solution. Nothing is recalibrated, deliberately: conformal rescaling brings any shape to the nominal level, so recalibrating would hide the very thing being measured, which is where a band stops being valid. The learned-width head is the one that fails. On Burgers it falls from 90.2% to 73.1% under isotropic noise and to 59.7% under structured noise, an interval sold as 90% that contains the truth three times in five: its width is fitted to where this model errs on clean inputs, and that map stops applying once the input moves. Jacobian-DeepONet degrades in the same direction but less, 89.7% to 76.3%, and JCB loses 3.5 points. The ordering repeats on Darcy and Navier–Stokes at smaller amplitude, where only the structured noise separates the methods.

Two readings of this would be wrong. The first is that the unconditioned bands are the safe choice, since constant width retains 87.7% on Burgers. That is inertia rather than accuracy: a width that ignores the input cannot become misaligned with it, and the price appears as soon as coverage is read conditionally: on Burgers it covers 40.9% of the decile of samples on which the operator errs most, against 55.1% for JCB. That column is not uniform: on Darcy the constant-width band is the better of the two, 63.1% against 54.2%, so the ordering favours the sensitivity prior only on Burgers and Navier–Stokes. The second is that this settles the comparison in favour of the sensitivity prior. It does not: the conditional quantile head is the best of the five on that hardest decile, 81.8% against 55.1%, because it is the only one ever told where the model errs. What the table establishes is narrower and more useful. The two alignments fail in different places, a width learned from residuals being accurate about model error and fragile to input error and a width read from the Jacobian the reverse, so which one a deployment should prefer is a question about its inputs rather than about the method.

Table 3: Coverage when the input is measured with error. Each band is calibrated once on clean data and its multiplier frozen; the model then receives $u + \varepsilon$. *iso*: $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ on the standardized branch input, the covariance the latent Gram assumes; *smooth*: the same noise passed through a Gaussian filter over the input field. Hardest decile: the isotropic case on the decile of samples with the largest error. σ as in training; three seeds, 16 draws each. Bold: best retention among bands whose width depends on the input. Darcy uses the sensitivity set.

Band	**PICP clean (%)**	**PICP iso (%)**	**PICP smooth (%)**	**Hardest decile (%)**	**MPIW**
Burgers 1D (MPIW $\times 10^{-2}$, $\sigma = 0.025$)					
Constant-width pinball	90.63 ± 0.19	89.30 ± 0.25	87.66 ± 0.28	40.86 ± 1.81	3.886
Spatial quantile	90.54 ± 0.43	88.46 ± 0.42	86.01 ± 0.54	38.10 ± 1.56	5.464
Conditional quantile	90.24 ± 0.41	73.14 ± 0.74	59.70 ± 0.78	81.75 ± 1.03	4.218
Jacobian-DeepONet	89.68 ± 0.44	85.80 ± 0.40	76.30 ± 0.43	61.70 ± 1.33	3.984
JCB (frozen operator)	90.13 ± 0.15	89.29 ± 0.18	$\mathbf{86.60 \pm 0.19}$	55.08 ± 0.66	4.207
Darcy 2D (MPIW $\times 10^{-3}$, $\sigma = 0.025$)					
Constant-width pinball	90.36 ± 0.04	90.26 ± 0.02	89.10 ± 0.08	63.09 ± 0.88	4.084
Spatial quantile	90.06 ± 0.27	89.97 ± 0.26	88.77 ± 0.25	63.03 ± 0.94	4.120
Conditional quantile	90.38 ± 0.67	89.68 ± 0.68	84.26 ± 0.93	78.39 ± 2.18	3.240
Jacobian-DeepONet	90.41 ± 0.32	90.04 ± 0.41	87.66 ± 0.80	74.17 ± 0.76	3.454
JCB (frozen operator)	90.02 ± 0.39	89.94 ± 0.39	$\mathbf{89.31 \pm 0.44}$	54.20 ± 1.44	4.296
Navier–Stokes 2D (MPIW $\times 10^{-1}$, $\sigma = 0.05$)					
Constant-width pinball	88.40 ± 0.22	88.26 ± 0.22	87.28 ± 0.25	58.38 ± 0.92	0.969
Spatial quantile	88.48 ± 0.00	88.33 ± 0.01	87.41 ± 0.05	58.52 ± 0.29	0.973
Conditional quantile	90.19 ± 0.37	89.86 ± 0.41	85.80 ± 0.44	89.51 ± 0.89	0.923
Jacobian-DeepONet	89.83 ± 0.16	89.48 ± 0.32	85.53 ± 0.25	85.16 ± 0.36	0.891
JCB (frozen operator)	88.77 ± 0.10	88.60 ± 0.11	$\mathbf{86.54 \pm 0.17}$	67.44 ± 0.89	0.884

5.4 Limitations

The benefits of the sensitivity prior depend on the PDE regime. Joint training requires choosing the loss weight λ and can affect central prediction accuracy: MSE remains comparable to the deterministic baseline on Darcy and Navier–Stokes but increases by 14% on Burgers. At matched coverage, intervals are 5.7% and 3.5% narrower than the conditional-quantile baseline on Burgers and Navier–Stokes, respectively, but 6.0% wider on Darcy. The first-order variance approximation may also become inaccurate under strongly nonlinear input perturbations. Finally, pinball training

alone does not attain nominal coverage on Darcy or Navier–Stokes, and calibration on in-distribution data does not ensure coverage under distribution shift (Appendix G).

6 Conclusions and Future Directions

We proposed the analytical sensitivity of a learned operator as an inductive prior for the spatial shape of predictive uncertainty, computable in K reverse-mode vector-Jacobian products by exploiting the Cartesian factorization of DeepONets, and realized it two ways: end-to-end in Jacobian-DeepONet, under an asymmetric pinball loss, and post-hoc in Jacobian Conformal Bands, which need no retraining and add no parameters. Across three benchmarks the prior tracks the reference physical sensitivity more closely than any baseline at comparable width and accuracy. The practical result is that the two alignments fail in different places: a width learned from residuals is accurate about model error and fragile when the input is uncertain, and a Jacobian-derived width the reverse.

Ethics Statement

This work trains surrogate models on synthetic data produced by numerical solvers for canonical PDE benchmarks. No human subjects, personal data or sensitive information are involved, and no new data collection was performed.

Reproducibility Statement

The code for the same-backbone comparison, the evaluation protocol, the table generation and the cluster job scripts is released with the paper. Every model in Table 8 shares one backbone, dataset, optimizer, schedule, loss weight λ, uncertainty scale σ and iteration count, and differs only in how the interval is parameterized; the exact launchers are provided per benchmark. Data generation is deterministic given the seed offsets recorded in the generation scripts, and training is seeded explicitly. Per-run metrics are released in machine-readable form alongside the command that renders Table 8 from them. Results are reported as mean and standard deviation over three seeds; the loss-weight sweep of Table 6 uses one seed per cell.

The whole pipeline runs from five commands, `data`, `verify`, `train`, `tables` and `appendix`, driven by a single script that submits scheduler array jobs where one is available and otherwise executes the same stages sequentially, so the pipeline can be exercised on one machine. Every metric behind every table is released as machine-readable per-run output, and the table generators take those files as arguments, so each number can be traced back to the run that produced it.

Data availability. The datasets are deposited in an open repository under a persistent identifier, and the code repository is linked from the deposit; both identifiers are withheld here to preserve anonymity and appear in the camera-ready version. The deposit follows the FAIR principles. Each file is a single HDF5 container exposing the same keys across benchmarks (`branch_inputs`, `trunk_inputs`, `targets`, and `jacobian_reference` where a reference sensitivity is available), so no reader beyond `h5py` is required. Provenance is explicit at file granularity: the generator, its arguments and its random seed offset are recorded for every file, the split verification is released alongside the data, and a companion document states what was measured about each file rather than what was intended. Regeneration from the recorded seed offsets is deterministic and reproduces the deposited files, so the deposit and the generators are interchangeable.

Use of Large Language Models

Gemini was used for general execution of research ideas: turning experimental designs proposed by the authors into runnable code, and iterating on implementations of the model variants and their training loops.

Claude Code was used for a reproducibility audit: it re-derived the reported tables from the stored checkpoints and metrics, re-ran the experiments to check that the results replicate, and flagged discrepancies between the numbers in the manuscript and the numbers the pipeline produces.

All experimental results, numerical values and claims in this paper were verified by the authors against the stored outputs of the pipeline. The authors take full responsibility for the content.

REFERENCES

Abdolmehdi Behroozi, Chaopeng Shen, and Daniel Kifer. Sensitivity-constrained Fourier neural operators for forward and inverse problems in parametric differential equations. In *International Conference on Learning Representations (ICLR)*, 2025.

Abdolmehdi Behroozi, Chaopeng Shen, Daniel Kifer, and Kathryn Lawson. Sensitivity-constrained neural operators for data-efficient forward and inverse modeling of partial differential equation systems. *arXiv preprint arXiv:2608.29888*, 2026.

Patricio Clark Di Leoni, Lu Lu, Charles Meneveau, George Em Karniadakis, and Tamer A. Zaki. Neural operator prediction of linear instability waves in high-speed boundary layers. *Journal of Computational Physics*, 474:111793, 2022. doi: 10.1016/j.jcp.2022.111793.

Wojciech Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. 06 2017. doi: 10.48550/arXiv.1706.04859.

Paul Escapil-Inchauspé, Eduardo Álvarez, Adolfo Parra, Vicente Iligaray, Vicente Opazo, and Danilo Aballay. Fair scientific machine learning, 2024. URL https://github.com/pescap/fair-sciml.

Shailesh Garg and Souvik Chakraborty. Vb-deeponet: A bayesian operator learning framework for uncertainty quantification. *Engineering Applications of Artificial Intelligence*, 118:105685, 2023. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2022.105685. URL https://www.sciencedirect.com/science/article/pii/S0952197622006753.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/2650d6089a6d640c5e85b2b88265dc2b-Paper.pdf.

Roger Koenker. *Quantile Regression*. Cambridge University Press, Cambridge, 2005. ISBN 9780521608275.

Jing Lei and Larry Wasserman. Distribution-free prediction bands for non-parametric regression. *Journal of the Royal Statistical Society: Series B*, 76(1):71–96, 2014. doi: 10.1111/rssb.12021.

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 *Advances in Neural Information Processing Systems*, volume 33, pp. 9522–9533, 2020. URL https://proceedings.neurips.cc/paper/2020/file/4b21ce5d0b6cf42d4d0c3b23f4c91e3d-Paper.pdf.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5.

Yingbo Ma, Vaibhav Dixit, Michael Innes, Xingjian Guo, and Chris Rackauckas. A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions. pp. 1–9, 09 2021. doi: 10.1109/HPEC49654.2021.9622796.

Emilia Magnani, Marvin Pförtner, Tobias Weber, and Philipp Hennig. Linearization turns neural operators into function-valued gaussian processes, 2024.

Christian Moya, Amirhossein Mollaali, Zecheng Zhang, Lu Lu, and Guang Lin. Conformalized-deeponet: A distribution-free framework for uncertainty quantification in deep operator networks. *Physica D: Nonlinear Phenomena*, 471:134418, 2025. ISSN 0167-2789. doi: https://doi.org/10.1016/j.physd.2024.134418. URL https://www.sciencedirect.com/science/article/pii/S0167278924003683.

Gary W. Oehlert. A note on the delta method. *The American Statistician*, 46(1):27–29, 1992. doi: 10.1080/00031305.1992.10475842.

Thomas O'Leary-Roseberry, Peng Chen, Umberto Villa, and Omar Ghattas. Derivative-informed neural operator: An efficient framework for high-dimensional parametric derivative learning. *Journal of Computational Physics*, 496:112555, 2024. doi: 10.1016/j.jcp.2023.112555.

Oscar Hernan Madrid Padilla, Wesley Tansey, and Yanzhen Chen. Quantile regression with deep relu networks: Estimators and minimax rates, 2020. URL https://arxiv.org/abs/2010.08236.

Harris Papadopoulos, Alex Gammerman, and Volodya Vovk. Normalized nonconformity measures for regression conformal prediction. In *IASTED International Conference on Artificial Intelligence and Applications*, pp. 64–69, 2008.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020. URL https://arxiv.org/abs/2001.04385.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017. URL https://arxiv.org/abs/1710.05941.

Yaniv Romano, Evan Patterson, and Emmanuel J. Candès. Conformalized quantile regression, 2019. URL https://arxiv.org/abs/1905.03222.

Tom Stent and Nicolas Boullé. Conformal uncertainty quantification guarantees for neural operators, 2026.

Natasa Tagasovska and David Lopez-Paz. Single-model uncertainties for deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/73c03186765e199c116224b68adc5fa0-Paper.pdf.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning, 2024. URL https://arxiv.org/abs/2210.07182.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2 edition, 2004. doi: 10.1007/978-3-031-06509-1.

Jiaxin Zhang. Modern monte carlo methods for efficient uncertainty quantification and propagation: A survey. *arXiv preprint arXiv:2011.00680*, 2020. URL https://doi.org/10.48550/arXiv.2011.00680.

A Implicit Regularization, the Constant-Width Baseline, and the Algorithm

This appendix derives the second-order gradients introduced by backpropagating through the Jacobian norm, defines the unconditioned constant-width baseline used in the ablation, and provides pseudocode for training and inference.

A.1 Latent Jacobian Extraction

While the Delta Method establishes a theoretical link between parametric sensitivity and predictive variance, materializing the full analytical Jacobian $\mathcal{J}_u \in \mathbb{R}^{N_{\text{nodes}} \times N_{\text{in}}}$ during every training iteration suffers from high memory allocation. For high-dimensional input fields and dense spatial grids, standard reverse-mode automatic differentiation exhausts GPU memory.

To overcome this fundamental limitation, we exploit the structural factorization inherent to the Cartesian DeepONet. Recall that the operator prediction evaluates as $\mathcal{G}_\theta(u)(X) = B(u)T(X)^T + b$. The Trunk Net $T(X)$ processes only spatial coordinates and maintains no functional dependence on the input parameters u. By applying the chain rule, the full operator Jacobian $\mathcal{J}_u \in \mathbb{R}^{N_{\text{nodes}} \times N_{\text{in}}}$ factors as a product of the Trunk basis and the latent Branch Jacobian:

$$\mathcal{J}_u = \frac{\partial \mathcal{G}}{\partial u} = T(X)\, \mathcal{J}_{\text{branch}} \in \mathbb{R}^{N_{\text{nodes}} \times N_{\text{in}}} \tag{8}$$

where $\mathcal{J}_{\text{branch}} = \partial B / \partial u \in \mathbb{R}^{K \times N_{\text{in}}}$ is the latent Jacobian of the Branch network. Instead of materializing the massive $\mathcal{J}_u$ tensor, we directly reformulate the pointwise variance utilizing a Gram matrix duality in the latent space. From the Delta Method, the full spatial covariance matrix is given by $\Sigma_{\text{out}} \approx \sigma^2 \mathcal{J}_u \mathcal{J}_u^T$. Substituting the factorized Jacobian $\mathcal{J}_u = T\, \mathcal{J}_{\text{branch}}$ yields:

$$\Sigma_{\text{out}} \approx \sigma^2 \left(T(X)\, \mathcal{J}_{\text{branch}}\right) \left(T(X)\, \mathcal{J}_{\text{branch}}\right)^T = \sigma^2\, T(X) \left(\mathcal{J}_{\text{branch}} \mathcal{J}_{\text{branch}}^T\right) T(X)^T \tag{9}$$

We define the latent Gram Matrix $\mathbf{G} \in \mathbb{R}^{K \times K}$ as the unscaled covariance of the latent Jacobian:

$$\mathbf{G} = \mathcal{J}_{\text{branch}} \mathcal{J}_{\text{branch}}^T \tag{10}$$

Since our uncertainty quantification framework only requires the pointwise variance at each discrete spatial node (i.e., the main diagonal of the spatial covariance matrix Σ_{out}), the exact pointwise variance vector $V_{\text{pred}} \in \mathbb{R}^{N_{\text{nodes}}}$, whose square root we denote $\sigma_{\text{pred}} = \sqrt{V_{\text{pred}}}$, is computed as:

$$V_{\text{pred}} = \sigma^2\, \text{diag}\left(T(X) \mathbf{G} T(X)^T\right) \tag{11}$$

The critical observation is that the full operator Jacobian $\mathcal{J}_u \in \mathbb{R}^{N_{\text{nodes}} \times N_{\text{in}}}$ need never be materialized. By isolating differentiation to the Branch Net via the factorization $\mathcal{J}_u = T(X)\, \mathcal{J}_{\text{branch}}$, the automatic differentiation engine constructs a computational graph over K latent output dimensions rather than N_{nodes} spatial nodes. This reduces the memory footprint of the Jacobian extraction from $\mathcal{O}(N_b \cdot N_{\text{nodes}} \cdot N_{\text{in}})$ to $\mathcal{O}(N_b \cdot K \cdot N_{\text{in}})$, which is the primary reason GPU memory does not collapse: for the Darcy Flow benchmark ($N_{\text{in}} = 1024$, $N_{\text{nodes}} = 4096$, $K = 100$), this corresponds to a $41\times$ reduction factor (N_{nodes}/K). The remaining matrix-vector operation $\text{diag}(T\mathbf{G}T^T)$ is computed efficiently in $\mathcal{O}(N_{\text{nodes}} \cdot K^2)$ without materializing the full $N_{\text{nodes}} \times N_{\text{nodes}}$ covariance matrix. Bounding the latent dimension K therefore bounds the cost of the extraction itself. The trunk evaluation, the output field and the $\text{diag}(T\mathbf{G}T^T)$ contraction still scale linearly in N_{nodes}, so denser grids remain more expensive; what the factorization removes is the $N_{\text{nodes}} \times N_{\text{in}}$ term, which is the one that would otherwise dominate.

Extracting $\mathcal{J}_{\text{branch}} \in \mathbb{R}^{K \times N_{\text{in}}}$ requires K reverse-mode backward passes through the Branch network. This can be done via a Python loop over K calls to `torch.autograd.grad`, or via `vmap(jacrev)` from PyTorch's functional API (Paszke et al., 2019). The former dispatches K sequential GPU commands from Python, incurring significant overhead. The latter evaluates the K products as one vectorized call, which is substantially faster in wall-clock time. Critically, under `torch.func`, `jacrev` automatically constructs the second-order computational

graph when its output participates in a subsequent `.backward()` call, without requiring explicit `create_graph=True`. This implicit graph construction lets the pinball gradient flow through $\mathcal{J}_{\text{branch}}$ back to the branch parameters, as discussed in the next subsection.

Mathematically, `jacrev` computes vector-Jacobian products (VJP). For a latent projection vector $\mathbf{v} \in \mathbb{R}^K$, the VJP evaluates the row-wise sensitivity:

$$\text{VJP}(u, \mathbf{v}) = \mathbf{v}^T \mathcal{J}_{\text{branch}} = \sum_{k=1}^{K} v_k \frac{\partial B_k(u)}{\partial u} \in \mathbb{R}^{1 \times N_{\text{in}}} \tag{12}$$

By vectorizing over the K standard basis vectors $\mathbf{e}_k \in \mathbb{R}^K$ along the latent output dimension via `vmap` (while processing input samples independently across the batch size N_b), `vmap(jacrev)` evaluates the K reverse-mode VJPs as a single vectorized call rather than a Python loop; how that call is scheduled on the device is left to the backend. This constructs the full latent Jacobian $\mathcal{J}_{\text{branch}} \in \mathbb{R}^{N_b \times K \times N_{\text{in}}}$, from which we form the unscaled latent covariance matrix $\mathbf{G} = \mathcal{J}_{\text{branch}} \mathcal{J}_{\text{branch}}^T \in \mathbb{R}^{N_b \times K \times K}$ efficiently.

A.2 Implicit Jacobian Regularization via Pinball Loss

A distinctive property of the proposed formulation is that backpropagating the pinball loss through the Jacobian norm induces a second-order regularization on the Branch Network's input-output derivatives, without requiring any explicit derivative supervision. We formalize this mechanism below.

Consider the total loss $\mathcal{L}_{\text{total}} = \mathcal{L}_{MSE} + \lambda \mathcal{L}_{pin}$, where the prediction bounds depend on the Branch parameters θ_B through two distinct computational pathways: the central prediction $\hat{y} = B_\theta(u) \cdot T_\theta(X)^T + b$, and the uncertainty shape $s(\mathbf{x}) = s_{\text{J}}(\mathbf{x}) \propto \|T(\mathbf{x})\, \mathcal{J}_{\text{branch}}\|$, where $[\mathcal{J}_{\text{branch}}]_{ki} = \partial B_k / \partial u_i$. The total gradient with respect to θ_B decomposes as:

$$\frac{\partial \mathcal{L}_{\text{total}}}{\partial \theta_B} = \underbrace{\frac{\partial \mathcal{L}_{MSE}}{\partial \theta_B}}_{\textit{(I) regression}} + \underbrace{\lambda \frac{\partial \mathcal{L}_{pin}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_B}}_{\textit{(II) coverage through } \hat{y}} + \underbrace{\lambda \frac{\partial \mathcal{L}_{pin}}{\partial s} \cdot \frac{\partial s}{\partial \theta_B}}_{\textit{(III) implicit Jacobian reg.}} \tag{13}$$

Terms (I) and (II) involve standard first-order derivatives of the network output. Term (III), however, propagates the pinball gradient through the Jacobian norm $\|T(\mathbf{x})\, \mathcal{J}_{\text{branch}}\|$. Since $[\mathcal{J}_{\text{branch}}]_{ki} = \partial B_k / \partial u_i$ already involves first-order derivatives of the Branch output with respect to the input, differentiating with respect to θ_B yields implicit Hessian-vector products:

$$\frac{\partial s}{\partial \theta_B} \propto \frac{\partial \|T(\mathbf{x})\, \mathcal{J}_{\text{branch}}\|}{\partial \theta_B} = \sum_{k,i} \frac{\partial \|T(\mathbf{x})\, \mathcal{J}_{\text{branch}}\|}{\partial [\mathcal{J}_{\text{branch}}]_{ki}} \cdot \frac{\partial^2 B_k(u)}{\partial u_i\, \partial \theta_B} \tag{14}$$

This second-order term regularizes the curvature of the Branch Network's input-output mapping without materializing the full Hessian matrix, as the products $\partial^2 B_k / (\partial u_i\, \partial \theta_B)$ are computed efficiently via reverse-mode automatic differentiation.

The *direction* of this regularization is determined by the pinball loss, not by external labels. At spatial locations where the prediction error exceeds the current band width ($|e(x_j)| > W(x_j)$), the gradient $\partial \mathcal{L}_{pin} / \partial s(x_j)$ drives the Jacobian norm to increase, expanding the bands locally. Where the error is well covered ($|e(x_j)| < W(x_j)$), the gradient penalizes excessive band width, contracting the Jacobian norm. Since the learnable scalars ($w_{\text{low}}, w_{\text{up}}$) control only the global scale of the intervals while the spatial shape is governed exclusively by $s(\mathbf{x}) \propto \|T(\mathbf{x})\, \mathcal{J}_{\text{branch}}\|$, the pinball gradient pushes the spatial profile of s towards that of the residual, $s(\mathbf{x}) \sim |e(\mathbf{x})|$; the loss alone does not guarantee that it gets there, and how far it does is what ρ_{Err} measures.

In nonlinear PDE regimes, the error topology concentrates along high-sensitivity physical structures (shock fronts, permeability interfaces, boundary layers), which is the route by which the network's internal Jacobian can come to align with the operator's physical sensitivity. This mechanism is related to, but fundamentally distinct from, classical Sobolev training (Czarnecki et al., 2017): whereas Sobolev training minimizes $\|\nabla f_\theta - \nabla f_{\text{true}}\|$ using supervised derivative labels, the implicit Jacobian regularization described here is driven entirely by the spatial error distribution under the pinball loss, achieving an analogous derivative-smoothing effect without any ground-truth derivative annotations.

A.3 Constant-Width Pinball DeepONet Baseline

To assess the inductive benefit of the analytical Jacobian prior $s_{\mathrm{J}}(\mathbf{x})$, we construct a direct non-parametric baseline termed **Constant-Width Pinball DeepONet**. To isolate the effect of the spatial prior, this baseline is a deliberate simplification rather than a general conditional Deep Quantile Regression (Tagasovska & Lopez-Paz, 2019) framework.

Formally, the Constant-Width Pinball DeepONet shares the identical central operator backbone $\mathcal{G}_\theta(u)(X) = B_\theta(u)T_\theta(X)^T + b$, but parameterizes its lower and upper quantile bounds via unconstrained scalar offsets without the spatial Jacobian norm:

$$y_{\text{low}}^{\text{vanilla}}(\mathbf{x}) = \mathcal{G}_\theta(u)(\mathbf{x}) - \text{Softplus}(w_{\text{low}}^{\text{vanilla}}) \tag{15}$$

$$y_{\text{up}}^{\text{vanilla}}(\mathbf{x}) = \mathcal{G}_\theta(u)(\mathbf{x}) + \text{Softplus}(w_{\text{up}}^{\text{vanilla}}) \tag{16}$$

The model is trained end-to-end to minimize the composite objective:

$$\mathcal{L}_{\text{vanilla}} = \text{MSE}(\mathcal{G}_\theta(u), y_{\text{true}}) + \lambda \left[\mathcal{L}_{\text{pinball}}\left(y_{\text{low}}^{\text{vanilla}}, y_{\text{true}}, \frac{\alpha}{2}\right) + \mathcal{L}_{\text{pinball}}\left(y_{\text{up}}^{\text{vanilla}}, y_{\text{true}}, 1 - \frac{\alpha}{2}\right)\right] \tag{17}$$

Because the empirical loss aggregates uniformly over the spatial domain, $\mathcal{L}_{\text{vanilla}}$ provides no spatial incentive to concentrate uncertainty in steep gradient regions or parameter interfaces. As demonstrated in Subsection 5.1, this absence of physical structure leads to a spatially uniform interval width, where global coverage is achieved trivially through over-coverage in simple regimes and severe under-coverage in high-error regimes.

A.4 Algorithmic Formulation: Method 1, Jacobian-DeepONet

The complete end-to-end training procedure for Jacobian-DeepONet is summarized in Algorithm 1.

Algorithm 1 End-to-End Training Step of Jacobian-DeepONet

Require: Mini-batch inputs $u \in \mathbb{R}^{N_b \times N_{\text{in}}}$, spatial grid $X \in \mathbb{R}^{N_{\text{nodes}} \times d}$, targets $y_{\text{true}} \in \mathbb{R}^{N_b \times N_{\text{nodes}}}$, loss weight λ, target quantile level α, noise scale σ.

1: **Forward Pass**: Compute Branch representations $B(u) \in \mathbb{R}^{N_b \times K}$ and Trunk basis $T(X) \in \mathbb{R}^{N_{\text{nodes}} \times K}$.
2: Evaluate central operator prediction: $\hat{y} \leftarrow B(u)T(X)^T + b$.
3: **Latent VJP Extraction**: Evaluate K reverse-mode VJPs as one vectorized call:
4: $\quad \mathcal{J}_{\text{branch}} \leftarrow \texttt{vmap(jacrev)}(B)(u) \in \mathbb{R}^{N_b \times K \times N_{\text{in}}}$.
5: **Gram Matrix Factorization**: Compute unscaled latent covariance:
6: $\quad \mathbf{G} \leftarrow \mathcal{J}_{\text{branch}}\mathcal{J}_{\text{branch}}^T \in \mathbb{R}^{N_b \times K \times K}$.
7: **Pointwise Predictive Variance**: Extract spatial diagonal:
8: $\quad V_{\text{pred}} \leftarrow \sigma^2 \,\text{diag}\left(T(X)\mathbf{G}T(X)^T\right) \in \mathbb{R}^{N_b \times N_{\text{nodes}}}, \quad \sigma_{\text{pred}} \leftarrow \sqrt{V_{\text{pred}}}$.
9: **Jacobian Shape Normalization**: Isolate spatial topology:
10: $\quad \overline{\sigma}_{\text{EMA}} \leftarrow \beta_{\text{EMA}}\overline{\sigma}_{\text{EMA}} + (1 - \beta_{\text{EMA}})\,\text{mean}(\sigma_{\text{pred}}).\,\text{detach}()$ (frozen after training).
11: $\quad s_{\mathrm{J}} \leftarrow \sigma_{\text{pred}}/\overline{\sigma}_{\text{EMA}}$.
12: **Quantile Bounds**: Parameterize asymmetric prediction intervals:
13: $\quad y_{\text{low}} \leftarrow \hat{y} - \text{Softplus}(w_{\text{low}}) \cdot s_{\mathrm{J}}, \quad y_{\text{up}} \leftarrow \hat{y} + \text{Softplus}(w_{\text{up}}) \cdot s_{\mathrm{J}}$.
14: **Composite Loss**: Evaluate total objective:
15: $\quad \mathcal{L}_{\text{total}} \leftarrow \text{MSE}(\hat{y}, y_{\text{true}}) + \lambda\,[\mathcal{L}_{pinball}(y_{\text{low}}, y_{\text{true}}, \alpha/2) + \mathcal{L}_{pinball}(y_{\text{up}}, y_{\text{true}}, 1 - \alpha/2)]$.
16: **Second-Order Backpropagation**: Update network parameters $\theta = \{\theta_B, \theta_T, b\}$ and calibration weights $\{w_{\text{low}}, w_{\text{up}}\}$ via Adam.

A.5 Algorithmic Formulation: Method 2, Jacobian Conformal Bands

Method 2 shares no training procedure with Method 1: the operator is given, frozen, and never updated. Algorithm 2 states the two phases separately, because only the first touches labelled data and only the second runs per query.

Two points on what the reported experiments do. First, we run the procedure once per candidate shape and report each one, rather than selecting between them inside the loop; the comparison in Section 5

is therefore a measurement of how much the shape matters, and the selection step it motivates is a deployment recipe we state but do not need in order to report these numbers. Second, the two scale constants, the floor reference $\mathrm{median}(\sigma_{\mathrm{pred}})$ and the compression scale $\overline{\sigma}_{\mathrm{pred}}$, are pooled over the whole evaluation corpus rather than over the calibration split alone. They are two scalars and $\hat{q}$ absorbs any global factor, so this does not affect the comparison between shapes, but a deployment that freezes them on the calibration split alone is the stricter choice: recomputing either per batch at query time would make one input's interval depend on the inputs it is batched with, which is the failure mode discussed in Subsection 3.2.

Algorithm 2 Post-hoc Jacobian Conformal Band on a Frozen Operator

Require: Frozen operator $\mathcal{G}_\theta$ (trained under any objective), labelled set $\mathcal{D}$, spatial grid X, nominal level α, floor β, shape s drawn from $\mathcal{S} = \{\,\mathrm{const},\ \sigma_{\mathrm{pred}},\ \log(1+\sigma_{\mathrm{pred}}/\overline{\sigma}_{\mathrm{pred}})\,\}$.

Phase 1: calibration (once, offline)

Partition $\mathcal{D}$ by sample, with a fixed seed independent of the model, into a calibration half $\mathcal{C}$ and an evaluation half $\mathcal{E}$.

Compute the trunk basis once: $T(X) \in \mathbb{R}^{N_{\mathrm{nodes}} \times K}$.

for each u in $\mathcal{D}$ **do**

  $\mathcal{J}_{\mathrm{branch}}(u) \leftarrow \texttt{vmap}\,(\texttt{jacrev})\,(B_\theta)(u)$ ▷ K VJPs, no gradient to θ

  $\mathbf{G}(u) \leftarrow \mathcal{J}_{\mathrm{branch}}(u)\,\mathcal{J}_{\mathrm{branch}}(u)^T$

  $\sigma_{\mathrm{pred}}(u,\cdot) \leftarrow \sqrt{\mathrm{diag}\,(T(X)\,\mathbf{G}(u)\,T(X)^T)}$

end for

$m \leftarrow \mathrm{median}(\sigma_{\mathrm{pred}}), \quad \overline{\sigma}_{\mathrm{pred}} \leftarrow \mathrm{mean}(\sigma_{\mathrm{pred}})$ ▷ pooled over $\mathcal{D}$; see remark

$h(u,\mathbf{x}) \leftarrow s\,(\sigma_{\mathrm{pred}}(u,\mathbf{x})) + \beta m$

$\hat{q} \leftarrow \mathrm{Quantile}_{1-\alpha}\,(\{|y_i - \mathcal{G}_\theta(u)_i| \,/\, h(u,\mathbf{x}_i)\}_{\mathcal{C}})$ ▷ calibration half only

return $(\hat{q}, m, \overline{\sigma}_{\mathrm{pred}})$; all reported metrics are computed on $\mathcal{E}$, which took no part in fitting $\hat{q}$.

Phase 2: inference (per query u)

$\hat{y} \leftarrow B_\theta(u)T(X)^T + b$ ▷ one forward pass

$\sigma_{\mathrm{pred}} \leftarrow \sqrt{\mathrm{diag}\,(T(X)\,\mathcal{J}_{\mathrm{branch}}(u)\mathcal{J}_{\mathrm{branch}}(u)^T\,T(X)^T)}$ ▷ one fused VJP pass

$h \leftarrow s(\sigma_{\mathrm{pred}}) + \beta m$ ▷ $m, \overline{\sigma}_{\mathrm{pred}}$ from Phase 1

return $y_{\mathrm{low}} \leftarrow \hat{y} - \hat{q}\,h, \quad y_{\mathrm{up}} \leftarrow \hat{y} + \hat{q}\,h.$

B Experimental Setup

To train and evaluate the performance of the Jacobian-DeepONet, we benchmark the model on high-fidelity simulated data across varying linear and non-linear partial differential equations. The datasets were generated using numerical solvers (finite volume and finite difference methods) and formatted into dense Cartesian tensors. The ground-truth solution fields and the corresponding analytical sensitivities were produced with two established scientific machine learning repositories, PDEBench (Takamoto et al., 2024) and FAIR-SciML (Escapil-Inchauspé et al., 2024), extended where a reference sensitivity was required.

In the neural operator learning formulation with Cartesian product evaluation, each individual sample corresponds to an entire continuous function trajectory or parametric field instance evaluated across a dense spatial or spatiotemporal discretization mesh of N_{nodes} points. Consequently, the total volume of supervised pointwise evaluations scales as the tensor product $N_{samples} \times N_{\mathrm{nodes}}$. For instance, the 1D viscous Burgers' benchmark comprises 15 000 training trajectories evaluated over 12 800 spatiotemporal nodes ($N_x \times N_t = 128 \times 100$), yielding 1.92×10^8 (192 million) supervised pointwise data points during training and 6.4×10^6 in testing. Similarly, 2D Darcy Flow trains on 10 000 function fields and 2D Navier–Stokes on 20 000, both over 4096 spatial grid nodes. Table 4 details both the functional trajectory counts and the resulting tensorial pointwise dataset sizes across all benchmarks.

Table 4: Discretization, Functional Trajectories, and Tensorial Dataset Volume for each PDE benchmark. The sensor count is what the branch reads: on Darcy and Navier–Stokes the field is subsampled on a stride-2 lattice of the 64×64 solver grid, while the target stays at full resolution.

Equation	**Sensors** (N_{in})	**Grid Nodes** (N_{nodes})	**Train Traj.** (N_{train})	**Test Traj.** (N_{test})	**Train Points** ($N_{train} \times N_{\text{nodes}}$)	**Test Points** ($N_{test} \times N_{\text{nodes}}$)
Burgers	129	128×100 (12 800)	15 000	500	1.92×10^8 (192M)	6.40×10^6 (6.40M)
Darcy Flow	1024	64×64 (4096)	10 000	1000	4.10×10^7 (41.0M)	4.10×10^6 (4.10M)
Navier–Stokes	1026	64×64 (4096)	20 000	500	8.19×10^7 (81.9M)	2.05×10^6 (2.05M)

B.1 DeepONet Architecture

To ensure consistent evaluation across all physical benchmarks, we standardized the core DeepONet design, adapting only the Branch input dimension to problem-specific requirements.

Both subnetworks have 5 hidden layers of 100 neurons, with $K = 100$. On Darcy and Navier–Stokes the branch input is subsampled on a stride-2 lattice of the 64×64 grid, so the operator reads $32 \times 32 = 1024$ sensors while the target field remains at the full $N_{\text{nodes}} = 4096$ resolution; Burgers has only 129 sensors and is left untouched. The models have 9.4×10^4 parameters on Burgers and 1.84×10^5 on Darcy and Navier–Stokes, and uncertainty extraction costs $K = 100$ vector–Jacobian products.

Table 5: DeepONet architecture and optimization hyperparameters per benchmark.

Equation	**Activation Function**	**Branch / Trunk Layers**	**Latent Dim** (K)	**Batch Size** (N_b)	**Loss Weight** (λ)	**Learning Rate** (η_0)
Burgers	Swish	$5 \times [100]$	100	128	4.0	1×10^{-3}
Navier–Stokes	Swish	$5 \times [100]$	100	128	4.0	1×10^{-3}
Darcy Flow	Swish	$5 \times [100]$	100	128	4.0	1×10^{-3}

B.2 Training Procedure and Implementation

Inputs, coordinates, and targets were standardized using $z = (x - \mu)/(\sigma + 10^{-8})$. Input and target statistics were computed component-wise from the training data and retained for evaluation in physical units. Dataset partitioning and sensor sampling used seed 42. Both subnetworks used Glorot-normal initialization (Glorot & Bengio, 2010) and Swish activations (Ramachandran et al., 2017), whose smoothness supports differentiation through the Jacobian.

All models were trained with Adam in `float32` for 200 000 iterations with batch size 128. The learning rate started at 10^{-3} and was multiplied by 0.7 every 50 000 steps. Checkpoints were selected on a separate validation split using interval score for interval models and MSE for the deterministic baseline.

Interval models used the joint MSE and pinball objective with quantile levels 0.05 and 0.95. We evaluated $\lambda \in \{1, 4, 8, 20\}$ and used $\lambda = 4$ across benchmarks (Table 5). For Jacobian-DeepONet, the sensitivity normalization scale was frozen at evaluation to ensure batch-independent intervals; Appendix H compares this with per-batch normalization.

Experiments ran on a SLURM-managed cluster, with one NVIDIA A100 GPU, four CPU cores, and 32 GB RAM per job. Jacobian extraction used PyTorch 2.x `vmap(jacrev)` to vectorize the reverse-mode VJPs.

B.3 Evaluation Metrics and Monte Carlo Benchmark

We evaluate central predictions using MSE and relative L_2 error on held-out test data. Interval quality is assessed at a nominal coverage of 90% using empirical coverage (PICP), mean width (MPIW), and interval score. Writing $y_{ij} = y_{\text{true}}(u_i)(x_j)$, $\ell_{ij} = y_{\text{low}}(u_i)(x_j)$, and $r_{ij} = y_{\text{up}}(u_i)(x_j)$, we

compute

$$\mathrm{PICP} = \frac{100\%}{N_{\mathrm{test}} N_{\mathrm{nodes}}} \sum_{i,j} \mathbf{1}\{\ell_{ij} \le y_{ij} \le r_{ij}\}, \tag{18}$$

$$\mathrm{MPIW} = \frac{1}{N_{\mathrm{test}} N_{\mathrm{nodes}}} \sum_{i,j} (r_{ij} - \ell_{ij}). \tag{19}$$

PICP measures average pointwise coverage, not simultaneous coverage of the full output field. Widths are compared after calibration toward the same nominal level.

The interval score jointly measures width and penalties for observations outside the bounds:

$$\mathrm{IS}_\alpha = \frac{1}{N_{\mathrm{test}} N_{\mathrm{nodes}}} \sum_{i,j} \left[r_{ij} - \ell_{ij} + \frac{2}{\alpha}(\ell_{ij} - y_{ij})_+ + \frac{2}{\alpha}(y_{ij} - r_{ij})_+ \right], \tag{20}$$

where $(a)_+ = \max(a, 0)$ and $\alpha = 0.10$. Lower scores indicate better interval quality.

We also report a Gaussian-proxy NLL:

$$\mathrm{NLL} = \frac{1}{N_{\mathrm{test}} N_{\mathrm{nodes}}} \sum_{i,j} \left[\frac{1}{2} \log(2\pi\sigma_{ij}^2) + \frac{(y_{ij} - \hat{y}_{ij})^2}{2\sigma_{ij}^2} \right], \qquad \sigma_{ij} = \frac{r_{ij} - \ell_{ij}}{2 z_{0.95}}, \tag{21}$$

with $z_{0.95} \approx 1.645$. This symmetric Gaussian proxy discards interval asymmetry and is used only as a comparative diagnostic, not as a likelihood supplied by the model.

For the Monte Carlo comparison, we draw $M \in \{16, 64, 256\}$ independent perturbations $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ per test input. We compare the sampled output standard deviation with its first-order Jacobian approximation, then evaluate it as an interval shape under the same calibration protocol (Appendix F).

B.4 1D Viscous Burgers Equation

To evaluate the model's robustness under highly non-linear dynamics and near-singular gradients, we utilize the 1D viscous Burgers' equation:

$$\partial_t u + u \partial_x u = \nu \partial_{xx} u, \quad x \in [0, 1],\ t \in [0, 1] \tag{22}$$

where $u(x, t)$ represents the fluid velocity and ν is the kinematic viscosity coefficient. The functional input to the Branch network is parameterized as the initial velocity field $u_0(x) = u(x, 0)$, generated via a truncated Fourier series with 5 random harmonic modes:

$$u_0(x) = \sum_{k=1}^{5} A_k \sin\left(\frac{2\pi k x}{L} + \phi_k \right), \quad A_k \sim \mathcal{U}(-1, 1),\ \phi_k \sim \mathcal{U}(0, 2\pi) \tag{23}$$

discretized over $N_x = 128$ uniform spatial coordinates and concatenated with the sampled viscosity coefficient $\nu \sim \mathcal{U}(0.001, 0.1)$, yielding a total input dimension of $N_{\mathrm{in}} = 129$. The branch is not subsampled here, since 129 sensors are already fewer than the latent basis would justify reducing. The operator is otherwise the same backbone as the other two benchmarks, branch $[129, 100, 100, 100, 100, 100]$ and trunk $[2, 100, 100, 100, 100, 100]$ with Swish activations and $K = 100$, for 9.4×10^4 parameters. This is the smallest model in the paper and it carries the largest output field, $N_{\mathrm{nodes}} = 12\,800$, which is precisely the regime where the $\mathcal{O}(K)$ extraction cost pays off.

Physically, the nonlinear advective term $(u\partial_x u)$ causes wave steepening into sharp shock fronts where parametric sensitivity concentrates. Reference solution trajectories were simulated using a Finite Volume Method (FVM) with a Van Leer flux limiter and an explicit 2nd-order Runge-Kutta (RK2) time integrator across $N_t = 100$ temporal snapshots ($N_{\mathrm{nodes}} = 128 \times 100 = 12\,800$).

To validate the model's learned uncertainty against true physical sensitivities without encountering non-differentiable limiter singularities in automatic differentiation, the reference ground-truth sensitivity field was computed via central finite differences directly through the high-fidelity numerical solver:

$$\mathcal{J}_{\mathrm{ref}}(x, t) = \frac{\partial u(x, t)}{\partial \nu} \approx \frac{u(x, t; \nu + \delta\nu) - u(x, t; \nu - \delta\nu)}{2\delta\nu}, \quad \delta\nu = 10^{-4} \tag{24}$$

This high-fidelity numerical reference benchmark provides the physical ground truth for evaluating the directional sensitivity rank correlation $\rho_{\text{Sens}} = 0.628 \pm 0.001$ over three seeds, reported in Table 7. As on Navier–Stokes, this reference travels with the test split, so every metric in the Burgers row is measured on the same 500 trajectories.

B.5 2D Darcy Flow Equation

To assess the model's capacity to propagate parametric uncertainty through highly heterogeneous, spatially continuous input fields, we utilize the 2D steady-state Darcy Flow equation:

$$-\nabla \cdot (\kappa(\mathbf{x}) \nabla u(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x} \in [0,1]^2 \tag{25}$$

subject to homogeneous Dirichlet boundary conditions $u(\mathbf{x}) = 0$ on $\partial\Omega$ under a constant unit forcing $f(\mathbf{x}) = 1$. The functional input to the Branch network is the log-permeability field $a(\mathbf{x})$, where the physical permeability is defined as $\kappa(\mathbf{x}) = \exp(a(\mathbf{x}))$. The field $a(\mathbf{x})$ is generated as a random surface via a 2D truncated Fourier series:

$$a(\mathbf{x}) = \sum_{i=1}^{3} \sum_{j=1}^{3} A_{ij} \sin\left(2\pi i x + 2\pi j y + \phi_{ij}\right), \quad A_{ij} \sim \mathcal{N}(0, 0.5^2),\ \phi_{ij} \sim \mathcal{U}(0, 2\pi) \tag{26}$$

discretized over a dense 64×64 Cartesian grid. The branch reads this field on a stride-2 lattice, $N_{\text{in}} = 32 \times 32 = 1024$ sensors, while the solver, the target field and the reference sensitivity all remain at the full 64×64 resolution, $N_{\text{nodes}} = 4096$. The exponential mapping ensures strict ellipticity ($\kappa(\mathbf{x}) > 0$) while producing localized permeability channels with high spatial gradients.

The operator for this benchmark is the backbone of Table 5: branch $[1024, 100, 100, 100, 100, 100]$ and trunk $[2, 100, 100, 100, 100, 100]$, both with Swish activations, contracted on a latent dimension $K = 100$, for 1.84×10^5 parameters in total. Uncertainty extraction therefore costs $K = 100$ vector–Jacobian products through the branch, independently of the 4096 output nodes.

The reference solution fields were generated using a second-order 5-point Finite Difference Method (FDM) solved via Conjugate Gradients (CG) in JAX. Because the forward elliptic problem is linear with respect to the solution state, exact physical sensitivities with respect to the permeability field are available by automatic differentiation through the implicit solver.

Materializing the full matrix $\partial u / \partial \kappa \in \mathbb{R}^{4096 \times 4096}$ is possible but not practical to store: it costs 67 MB per sample, roughly 33 GB for a set of 500. We therefore record the *reduced* sensitivity, the row sum of that Jacobian,

$$\mathcal{J}_{\text{ref}}(\mathbf{x}_i) = \sum_{j=1}^{4096} \frac{\partial u(\mathbf{x}_i)}{\partial \kappa_j} = (\mathcal{J}\mathbf{1})_i \tag{27}$$

which is one forward-mode JVP in the direction of ones rather than 4096 reverse-mode passes, and 16 KB per sample instead of 67 MB. It answers the question the hypothesis actually poses, namely how strongly the solution at $\mathbf{x}_i$ responds to a perturbation of the permeability field as a whole.

One asymmetry is worth stating, and it is specific to this benchmark. The sum above runs over all 4096 permeability nodes of the solver grid, whereas the branch reads the 1024 of them that survive the stride-2 subsampling. The reference therefore perturbs directions the operator never observes, unlike Burgers and Navier–Stokes, where the reference differentiates with respect to a scalar parameter (ν, γ) that the branch receives as an input of its own. We keep the full-field reference because it is the physical quantity the hypothesis is about, and because restricting it to the observed nodes would make the target depend on a preprocessing choice; the consequence is that ρ_{Sens} on Darcy is measured against a strictly richer perturbation than the operator has access to, which makes it a conservative estimate rather than a favourable one.

This reference is evaluated on a dedicated set of 500 instances generated from a disjoint seed range and verified to share no sample with either the training or the test split. The resulting correlation is $\rho_{\text{Sens}} = 0.612 \pm 0.006$ over three seeds, reported in Table 7. The test split itself carries no sensitivity labels for this benchmark, which is why ρ_{Sens} and the remaining metrics of that row are measured on different sets.

B.6 2D Compressible Navier–Stokes Equation

To test the network on a field produced by coupled gas dynamics, we trained it on the 2D Compressible Navier–Stokes equations. The solver evolves all four conservative variables; the operator we learn maps the initial density field to the final density field, so the target is a single scalar field and the coupling enters through the dynamics that generate it:

$$\partial_t U + \partial_x F(U) + \partial_y G(U) = \mu \Delta U_{vel}, \quad (x, y, t) \in [0, 1]^2 \times [0, T] \tag{28}$$

where $U = [\rho, \rho u, \rho v, E]^T$ is the conservative state vector representing density, momentum in both spatial dimensions, and total energy; $F(U)$ and $G(U)$ denote the inviscid Euler fluxes, and μ is the dynamic viscosity. The functional input to the Branch network consists of the initial density distribution $\rho_0(x, y) = \rho(x, y, 0)$, generated via 4×4 randomized spatial Fourier harmonics on a 64×64 grid, concatenated with the adiabatic exponent $\gamma \sim \mathcal{U}(1.2, 2.0)$ and dynamic viscosity $\mu \sim \mathcal{U}(10^{-3}, 10^{-2})$. As on Darcy, the branch reads the density field on a stride-2 lattice and the two scalar parameters are appended unchanged, yielding $N_{\text{in}} = 32 \times 32 + 2 = 1026$. The target is the density field at the final time on the full 64×64 grid, $N_{\text{nodes}} = 4096$, evaluated through a two-dimensional trunk. The backbone is the same configuration used on Darcy, branch $[1026, 100, 100, 100, 100, 100]$ and trunk $[2, 100, 100, 100, 100, 100]$ with Swish activations and $K = 100$, for 1.84×10^5 parameters; the two scalar parameters enter the branch as additional sensors, so the Jacobian extraction covers the sensitivity to γ and μ alongside that to the initial density field.

The reference solution fields at final time $T = 0.5$ were computed using the MacCormack predictor-corrector finite difference scheme over 1000 internal integration steps. The reference sensitivity of the final density state $\rho(\mathbf{x}, T)$ with respect to the thermodynamic gas parameter γ was extracted via high-fidelity central finite differences:

$$\mathcal{J}_{\text{ref}}(\mathbf{x}) = \frac{\partial \rho(\mathbf{x}, T)}{\partial \gamma} \approx \frac{\rho(\mathbf{x}, T; \gamma + \delta\gamma) - \rho(\mathbf{x}, T; \gamma - \delta\gamma)}{2\delta\gamma}, \quad \delta\gamma = 10^{-4} \tag{29}$$

yielding the high-fidelity numerical ground truth for evaluating the directional sensitivity correlation $\rho_{\text{Sens}} = 0.525 \pm 0.001$ over three seeds. Unlike Darcy, this reference travels with the test split itself, so every metric in the Navier–Stokes row is measured on the same 500 instances.

C Loss Weight and Cost

Table 6 is the sweep summarized in Subsection 5.1, reported for all three benchmarks.

Table 6: Loss-weight sweep, one seed per cell, on all three benchmarks. Everything except λ is held fixed: architecture, data, schedule and iteration count. Metrics are as trained, with no conformal rescaling, since the point is what each parameterization does on its own. † marks the λ selected by validation interval score, which never sees the test set. The sweep is what licenses reading the main comparison as a property of the parameterization rather than of a tuned weight: the Jacobian interval score stays below the constant-width one at every λ on all three benchmarks, with no crossing, and the rank correlations are nearly flat in λ while the gap between the two parameterizations is not. The constant-width ρ_{Sens} is identical to three decimals across the sweep on Darcy and on Navier–Stokes, at 0.485 and 0.008, because that band is the training standard deviation field up to a scalar and λ cannot reshape it. Navier–Stokes is where that field carries nothing at all: the constant-width band sits at $\rho_{\text{Sens}} = 0.008$ and $\rho_{\text{Err}} \leq 0.009$ at every λ, against 0.520 and 0.495 for the Jacobian parameterization, so on that benchmark the entire correlation is supplied by the prior.

Model	λ	**MSE**	**PICP (%)**	**MPIW**	**Interval score**	ρ_{Err}	ρ_{Sens}
Burgers 1D (MSE $\times 10^{-3}$, MPIW and interval score $\times 10^{-2}$)							
Jacobian-DeepONet	1	1.635	93.00	4.696	8.089	0.733	0.619
Jacobian-DeepONet	4	1.776	93.50	4.844	7.702	0.750	0.628
Jacobian-DeepONet	8†	1.865	93.40	4.932	7.631	0.734	0.633
Jacobian-DeepONet	20	1.892	94.02	5.035	**7.595**	0.749	**0.635**
Constant-width pinball	1	1.457	87.76	3.329	13.521	0.496	0.176
Constant-width pinball	4	1.478	87.91	3.255	13.447	0.492	0.177
Constant-width pinball	8	1.436	87.79	3.232	13.282	0.494	0.177
Constant-width pinball	20†	1.447	87.45	3.181	13.304	0.485	0.177
Darcy 2D (MSE $\times 10^{-6}$, MPIW and interval score $\times 10^{-3}$)							
Jacobian-DeepONet	1	2.973	82.72	2.677	6.536	0.603	0.609
Jacobian-DeepONet	4†	2.782	84.08	2.893	6.157	0.603	0.617
Jacobian-DeepONet	8	3.731	83.87	3.005	6.432	0.618	0.614
Jacobian-DeepONet	20	3.178	85.80	3.275	**6.152**	0.598	**0.643**
Constant-width pinball	1	2.856	82.46	2.913	7.726	0.428	0.485
Constant-width pinball	4†	2.619	81.19	2.743	7.582	0.428	0.485
Constant-width pinball	8	3.020	80.97	2.735	8.023	0.434	0.485
Constant-width pinball	20	3.458	83.28	3.315	8.245	0.434	0.485
Navier–Stokes 2D (MSE $\times 10^{-3}$, MPIW and interval score $\times 10^{-1}$)							
Jacobian-DeepONet	1	1.072	88.14	0.830	1.141	0.495	**0.530**
Jacobian-DeepONet	4	1.071	88.47	0.827	1.111	0.518	0.523
Jacobian-DeepONet	8†	1.074	88.27	0.827	1.105	0.523	0.523
Jacobian-DeepONet	20	1.084	88.47	0.833	**1.099**	0.529	0.520
Constant-width pinball	1†	1.112	87.85	0.901	1.603	0.005	0.008
Constant-width pinball	4	1.150	87.62	0.908	1.626	0.009	0.008
Constant-width pinball	8	1.184	87.66	0.923	1.646	0.003	0.008
Constant-width pinball	20	1.189	87.53	0.922	1.644	0.003	0.008

Cost. The Jacobian parameterization adds two scalars to the backbone against 2.0×10^4 for the split-branch quantile head, but it needs a VJP pass at prediction time: $3.3\times$ the quantile head on Burgers, $2.7\times$ on Navier–Stokes and $2.6\times$ on Darcy. That is not the comparison a practitioner faces, though, because the quantile head returns no sensitivity field. Across every route to a spatially resolved uncertainty on Burgers, timed identically on one A100 at the evaluation batch size and including host–device transfer, the ordering is 0.100 ms per sample for the deterministic operator, 0.087 for the quantile head, 0.284 for the Jacobian model and 0.766 for the post-hoc Jacobian band of Subsection 3.3, against 2.82 ms for a Monte Carlo ensemble of $M = 64$ forward passes and 7.92 ms for the $M = 256$ ensemble that the band actually converges to (Appendix F): an order of magnitude cheaper than the only alternative returning the same quantity, and still well under a millisecond. Training cost is paid once: on Burgers at 200k iterations on one A100, 13 minutes for the deterministic model and 17 for the quantile heads against 58 minutes for the Jacobian model, the K VJPs dominating the step. The post-hoc band is the most expensive at inference because our implementation evaluates two scale variants in double precision as a diagnostic; a deployment computing one variant in single precision costs about half. All other timings are strict single precision (FP32), with TensorFloat-32 (TF32) disabled.

D FULL RESULT TABLES

Table 7 gives the cross-benchmark summary and Table 8 the five-model same-backbone comparison whose three-model extract appears as Table 1 in the body.

Table 7: Cross-benchmark summary on held-out test sets, mean $\pm$ standard deviation over three seeds. Spearman rank correlations align the predicted width with the reference physical sensitivity (ρ_{Sens}) and with the realized absolute error (ρ_{Err}). Three points on how to read it. PICP and MPIW are *as trained*, with no conformal rescaling, so the widths here are not the equal-coverage widths of Table 1 and should not be compared across rows. The benchmark column is a deterministic operator trained under mean-squared error alone on the identical backbone, which is why the Jacobian parameterization can match or beat it. On Darcy the spread of the Jacobian column is dominated by one seed of three, and ρ_{Sens} is measured on the separate 500-instance sensitivity set of Appendix B rather than on the test split.

Equation	Physical Regime	λ	MSE (Benchmark)	MSE (Jacobian)	PICP	MPIW	Proxy-NLL	ρ_{Sens}	ρ_{Err}
Burgers 1D	Nonlinear Shock Dynamics	4	$(1.537 \pm 0.014) \times 10^{-3}$	$(1.757 \pm 0.044) \times 10^{-3}$	$93.42 \pm 0.09\%$	$(4.793 \pm 0.060) \times 10^{-2}$	-3.76 ± 0.01	0.628 ± 0.001	0.746 ± 0.005
Darcy Flow 2D	Heterogeneous Porous Media	4	$(3.013 \pm 0.108) \times 10^{-6}$	$(3.037 \pm 0.442) \times 10^{-6}$	$83.22 \pm 0.81\%$	$(2.731 \pm 0.140) \times 10^{-3}$	-5.46 ± 0.11	0.612 ± 0.006	0.614 ± 0.010
Navier–Stokes 2D	Coupled Compressible Gas	4	$(1.094 \pm 0.009) \times 10^{-3}$	$(1.076 \pm 0.015) \times 10^{-3}$	$88.12 \pm 0.21\%$	$(8.231 \pm 0.033) \times 10^{-2}$	-2.38 ± 0.00	0.525 ± 0.001	0.519 ± 0.005

Table 8: Same-backbone comparison at matched architecture, data, schedule, λ and number of iterations (200k). Bands are compared at equal empirical coverage via a split-conformal multiplier calibrated on held-out in-distribution data; PICP is reported as trained. $\pm$ is the standard deviation over seeds. On Darcy alone, ρ_{Sens} is measured on the separate 500-instance sensitivity set rather than on the test split. Latency is per sample on one A100 at the evaluation batch size, averaged over seeds.

Model	MSE	PICP (%)	MPIW @90%	Proxy-NLL	ρ_{Err}	ρ_{Sens}	Params	ms/sample
Burgers 1D (3 seeds)								
Deterministic (n=3)	$(1.537 \pm 0.014) \times 10^{-3}$	–	–	–	–	–	94k	0.100
Constant-width pinball (n=3)	$(1.429 \pm 0.043) \times 10^{-3}$	87.6 ± 0.3	$(3.89 \pm 0.06) \times 10^{-2}$	-0.68 ± 0.12	0.497 ± 0.006	0.177 ± 0.000	94k	0.101
Spatial quantile (n=3)	$(1.752 \pm 0.020) \times 10^{-3}$	87.7 ± 0.5	$(5.46 \pm 0.24) \times 10^{-2}$	0.66 ± 0.33	0.553 ± 0.005	0.176 ± 0.000	94k	0.077
Conditional quantile (n=3)	$(1.865 \pm 0.013) \times 10^{-3}$	88.7 ± 0.9	$(4.22 \pm 0.11) \times 10^{-2}$	-4.36 ± 0.00	0.806 ± 0.004	0.477 ± 0.001	114k	0.087
Jacobian-DeepONet, EMA scale (n=3)	$(1.757 \pm 0.044) \times 10^{-3}$	93.4 ± 0.1	$(3.98 \pm 0.02) \times 10^{-2}$	-3.81 ± 0.01	0.746 ± 0.005	0.628 ± 0.001	94k	0.284
Darcy 2D (3 seeds)								
Deterministic (n=3)	$(3.013 \pm 0.108) \times 10^{-6}$	–	–	–	–	–	184k	0.035
Constant-width pinball (n=3)	$(2.826 \pm 0.180) \times 10^{-6}$	81.0 ± 0.2	$(4.08 \pm 0.02) \times 10^{-3}$	-5.21 ± 0.04	0.429 ± 0.002	0.485 ± 0.000	184k	0.035
Spatial quantile (n=3)	$(2.960 \pm 0.284) \times 10^{-6}$	81.6 ± 0.4	$(4.12 \pm 0.07) \times 10^{-3}$	-5.18 ± 0.06	0.437 ± 0.000	0.580 ± 0.012	184k	0.032
Conditional quantile (n=3)	$(2.735 \pm 0.105) \times 10^{-6}$	81.5 ± 0.3	$(3.18 \pm 0.18) \times 10^{-3}$	-5.77 ± 0.05	0.661 ± 0.003	0.573 ± 0.009	204k	0.035
Jacobian-DeepONet, EMA scale (n=3)	$(3.037 \pm 0.442) \times 10^{-6}$	83.2 ± 0.8	$(3.37 \pm 0.09) \times 10^{-3}$	-5.61 ± 0.06	0.614 ± 0.010	0.612 ± 0.006	184k	0.092
Navier–Stokes 2D (3 seeds)								
Deterministic (n=3)	$(1.094 \pm 0.009) \times 10^{-3}$	–	–	–	–	–	184k	0.033
Constant-width pinball (n=3)	$(1.145 \pm 0.015) \times 10^{-3}$	87.8 ± 0.1	$(0.969 \pm 0.011) \times 10^{-1}$	-1.91 ± 0.00	0.001 ± 0.004	0.008 ± 0.000	184k	0.035
Spatial quantile (n=3)	$(1.140 \pm 0.006) \times 10^{-3}$	87.8 ± 0.1	$(0.973 \pm 0.004) \times 10^{-1}$	-1.92 ± 0.00	0.004 ± 0.002	0.003 ± 0.002	184k	0.033
Conditional quantile (n=3)	$(1.081 \pm 0.011) \times 10^{-3}$	87.5 ± 0.1	$(0.923 \pm 0.012) \times 10^{-1}$	-2.35 ± 0.00	0.514 ± 0.002	0.479 ± 0.002	204k	0.034
Jacobian-DeepONet, EMA scale (n=3)	$(1.076 \pm 0.015) \times 10^{-3}$	88.1 ± 0.2	$(0.891 \pm 0.006) \times 10^{-1}$	-2.35 ± 0.00	0.519 ± 0.005	0.525 ± 0.001	184k	0.092

Table 9 is the extended form of the training-free comparison of Subsection 5.2. Two things it adds are worth reading directly. The calibration multiplier $\hat{q}$ is not a nuisance parameter: on Navier–Stokes it has to grow from 0.475 for the constant shape to 0.621 for the Jacobian one, which is the sense in which the Jacobian band is not simply a rescaling of the constant band, since a pure rescaling would leave every other column unchanged. And the compressed shape, which damps the dynamic range of σ_{pred}, trades interval score for width on Burgers and Navier–Stokes, where it is the narrowest row and the worse interval score of the two Jacobian shapes; on Darcy it improves on the uncompressed shape on both counts. We report both because the choice between them is exactly the kind of decision the calibration split can make, and because the uncompressed shape is the one that corresponds to the delta-method estimate without further processing.

Table 9: Training-free conformal bands in full: the extended form of Table 2, adding the calibration multiplier, the proxy-NLL, the compressed Jacobian shape and the per-sample cost. Every row of a block uses one frozen deterministic operator, trained under mean-squared error alone, and one split-conformal calibration on a held-out half of the test set; only the shape of the half-width h changes. Three seeds, mean $\pm$ standard deviation. The parameter count is the backbone's and is identical across rows, because none of these constructions adds a parameter: 9.4×10^4 on Burgers and 1.84×10^5 on Darcy and Navier–Stokes. Latency is per sample on one A100; the Jacobian rows pay one fused VJP pass on top of the forward, and our implementation evaluates two scale variants in double precision, so a deployment computing one in single precision costs about half. Darcy's ρ_{Sens} comes from the separate 500-instance sensitivity set.

Band shape	$\hat{q}$	PICP (%)	MPIW@90%	Interval score	Proxy-NLL	ρ_{Err}	ρ_{Sens}
Burgers 1D (MPIW and interval score $\times 10^{-2}$; 0.100 ms per sample, 0.766 for the Jacobian rows)							
Constant width	0.0259 ± 0.0005	90.22 ± 0.03	5.432 ± 0.103	16.872 ± 0.134	-0.26 ± 0.06	–	–
Constant $\times\ \sigma_{\text{train}}$	0.0932 ± 0.0020	90.41 ± 0.16	4.440 ± 0.096	14.313 ± 0.104	-1.11 ± 0.09	0.497 ± 0.009	0.187 ± 0.000
Jacobian	0.1206 ± 0.0043	90.19 ± 0.15	4.213 ± 0.081	$\mathbf{10.696 \pm 0.165}$	$\mathbf{-3.21 \pm 0.02}$	0.654 ± 0.008	$\mathbf{0.647 \pm 0.008}$
Jacobian, compressed	0.1299 ± 0.0020	90.29 ± 0.07	$\mathbf{3.850 \pm 0.056}$	11.916 ± 0.149	-2.96 ± 0.03	$\mathbf{0.662 \pm 0.007}$	0.607 ± 0.008
Darcy 2D (MPIW and interval score $\times 10^{-3}$; 0.035 ms per sample, 0.209 for the Jacobian rows)							
Constant width	0.0020 ± 0.0000	91.10 ± 0.27	4.300 ± 0.035	7.543 ± 0.076	-4.94 ± 0.03	–	–
Constant $\times\ \sigma_{\text{train}}$	0.2489 ± 0.0022	91.19 ± 0.23	$\mathbf{3.893 \pm 0.035}$	$\mathbf{6.948 \pm 0.077}$	-5.17 ± 0.05	$\mathbf{0.425 \pm 0.002}$	0.485 ± 0.000
Jacobian	0.3344 ± 0.0041	91.40 ± 0.39	4.357 ± 0.079	7.181 ± 0.047	-5.17 ± 0.04	0.405 ± 0.005	$\mathbf{0.754 \pm 0.009}$
Jacobian, compressed	0.3704 ± 0.0062	91.40 ± 0.35	4.156 ± 0.066	7.051 ± 0.058	$\mathbf{-5.21 \pm 0.04}$	0.413 ± 0.004	0.724 ± 0.009
Navier–Stokes 2D (MPIW and interval score $\times 10^{-1}$; 0.033 ms per sample, 0.216 for the Jacobian rows)							
Constant width	0.0451 ± 0.0001	88.25 ± 0.17	0.946 ± 0.003	1.636 ± 0.010	-1.93 ± 0.01	–	–
Constant $\times\ \sigma_{\text{train}}$	0.4753 ± 0.0013	88.26 ± 0.17	0.946 ± 0.003	1.636 ± 0.010	-1.93 ± 0.01	0.006 ± 0.004	0.006 ± 0.000
Jacobian	0.6210 ± 0.0116	$\mathbf{88.75 \pm 0.10}$	0.883 ± 0.004	$\mathbf{1.408 \pm 0.010}$	$\mathbf{-2.15 \pm 0.01}$	$\mathbf{0.420 \pm 0.005}$	$\mathbf{0.499 \pm 0.009}$
Jacobian, compressed	0.6440 ± 0.0020	88.54 ± 0.12	$\mathbf{0.875 \pm 0.004}$	1.456 ± 0.009	-2.11 ± 0.01	$\mathbf{0.420 \pm 0.005}$	$\mathbf{0.499 \pm 0.009}$

E Per-Benchmark Results

This appendix reports each benchmark separately: predictive accuracy, the alignment of the Jacobian width with the physical sensitivity and with the realized error, and the error-decile decomposition of coverage and width. The cross-benchmark synthesis is in Subsection 5.1.

E.1 Quartile Spatial Decomposition

Figures 2 and 3 decompose the bands across error quartiles (the 25th, 50th, 75th and 95th percentiles of per-sample error) for the two constructions of Section 5: the trained Jacobian-DeepONet and the post-hoc band on a frozen deterministic operator. Both are drawn from the same runs and the same calibration as the tables, at $\lambda = 4$ and seed 0, so the pictures and the numbers describe the same objects.

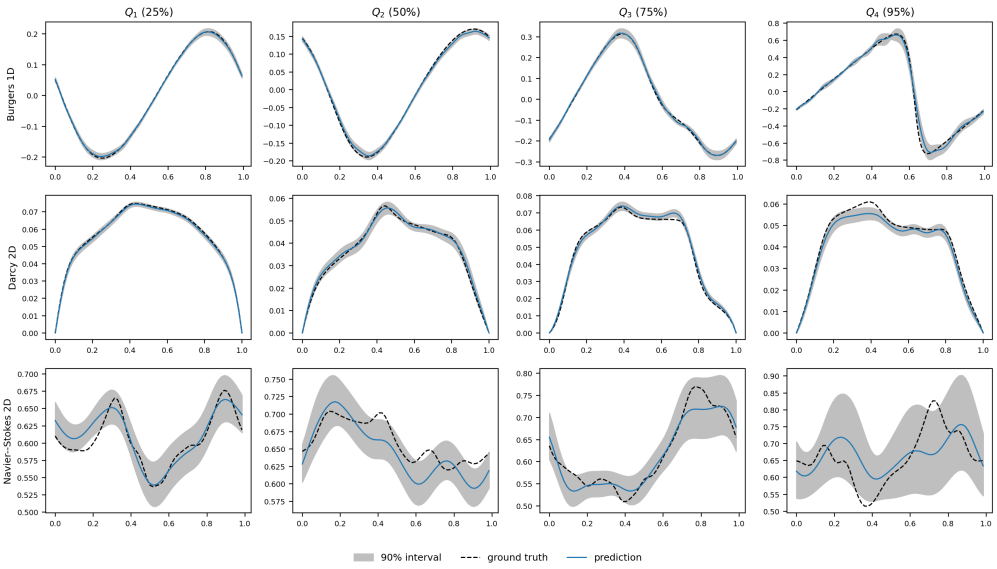


Figure 2: Quartile decomposition of the trained Jacobian-DeepONet bands. One row per benchmark, one column per difficulty quartile: test samples are ordered by their own mean squared error and the ones at the 25th, 50th, 75th and 95th percentiles are shown. Each panel is a one-dimensional slice at the midpoint of the second trunk coordinate, an intermediate time on Burgers and the central row of the grid on Darcy and Navier–Stokes. Gray bands are the 90% prediction interval as trained, dashed black the ground truth, solid blue the central prediction.

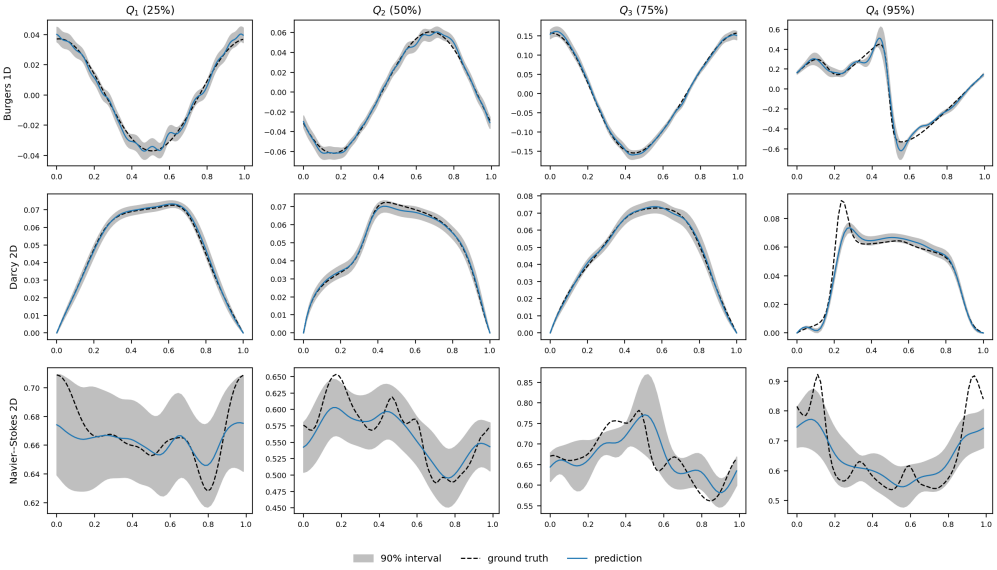


Figure 3: The same decomposition for the post-hoc Jacobian conformal band of Subsection 3.3, on a deterministic operator trained under mean-squared error alone. The centre is that operator's prediction, and the band is its own Jacobian shape scaled by the conformal multiplier of Table 2; no weight differs from the deterministic baseline. Panels, slices and quartiles are constructed exactly as in Figure 2, but the samples are ordered by this model's error, so the columns need not contain the same test instances.

Read together, the two figures show the same qualitative behaviour reached by different routes. On Burgers the width tracks the shock in both: it is a thin envelope where the profile is smooth and opens

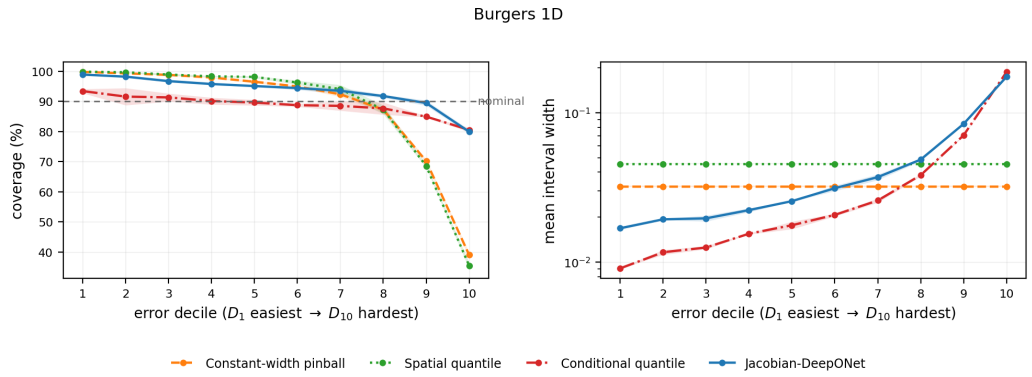


Figure 4: Burgers, four parameterizations at $\lambda = 4$, binned by per-sample error decile over three seeds; the shaded band is the seed spread. Left: coverage as trained, with the nominal level dashed. The two unconditioned baselines start above nominal and collapse to 39.1% and 35.5% in the hardest decile, while the conditioned ones hold 80.0% and 80.6%. Right: the mean width behind that behaviour, flat by construction for the unconditioned baselines and rising by an order of magnitude across deciles for the conditioned ones.

at the discontinuity in Q_4. On Darcy both follow the head profile closely and both miss the same feature in Q_4, a sharp interior peak the operator does not resolve, which is the visual counterpart of the coverage shortfall that benchmark shows in every table. Navier–Stokes is where the two separate: the trained band is tighter around the prediction, while the post-hoc band is wider and more uneven, which is the cost of reading the shape off a model that was never trained to place it. In no panel does either band inflate uniformly, which is the property the constant-width baseline would show and neither of these does.

E.2 1D Viscous Burgers' Equation: Nonlinear Advection and Shock Singularities

The 1D viscous Burgers' equation tests the operator's capacity to resolve steep gradient steepening and near-singular shock fronts. The reference sensitivity with respect to the viscosity parameter ν concentrates sharply along the spatio-temporal shock discontinuity.

As reported in Table 7 (loss weight $\lambda = 4$, three seeds), Jacobian-DeepONet reaches a rank correlation of $\rho_{\text{Sens}} = 0.628 \pm 0.001$ with the high-fidelity finite-difference sensitivity field, against 0.477 ± 0.001 for the conditional quantile head and 0.177 ± 0.000 for a constant-width band, and an error correlation of $\rho_{\text{Err}} = 0.746 \pm 0.005$. The model attains an MSE of $(1.757 \pm 0.044) \times 10^{-3}$, 14% above the deterministic benchmark $(1.537 \pm 0.014) \times 10^{-3}$, where the conditional quantile head pays 21%; this is the one benchmark on which training an interval costs central accuracy. Coverage as trained is PICP $= 93.42 \pm 0.09\%$, above the nominal level rather than below it as on the other two, so the split-conformal multiplier here contracts the band rather than expanding it, $\hat{q} = 0.853 \pm 0.011$, whose spatio-temporal response is mapped in Figure 1. At equal coverage the Jacobian band is sharper than the conditional quantile head, $(3.98 \pm 0.02) \times 10^{-2}$ against $(4.22 \pm 0.11) \times 10^{-2}$; the constant-width band is narrower still, $(3.89 \pm 0.06) \times 10^{-2}$, but reaches NLL $= -0.68 \pm 0.12$ against -3.81 ± 0.01 here, the gap reflecting how much of the width is placed where the residuals actually are.

Figure 1 highlights the physical dynamics of shock formation: as time progresses along the vertical axis, the non-linear advective term $u\partial_x u$ sharpens the initial velocity field into localized front discontinuities. Physical sensitivity (Panel 4) and absolute prediction errors (Panel 3) strictly align along these fronts. Jacobian-DeepONet (Panel 5) dynamically allocates high uncertainty along the shock trajectories (up to $W_{Jac} \approx 0.57$), whereas Constant-Width Pinball (Panel 6) renders a uniform, low-magnitude band across the entire domain.

Figure 4 details the mechanism. Because the constant-width band cannot adjust locally, its coverage falls from 99.8% in the easiest decile to 39.1% in D_{10}, and the spatial quantile head, whose width depends on position but not on the instance, behaves the same way, 99.9% to 35.5%. Both spend

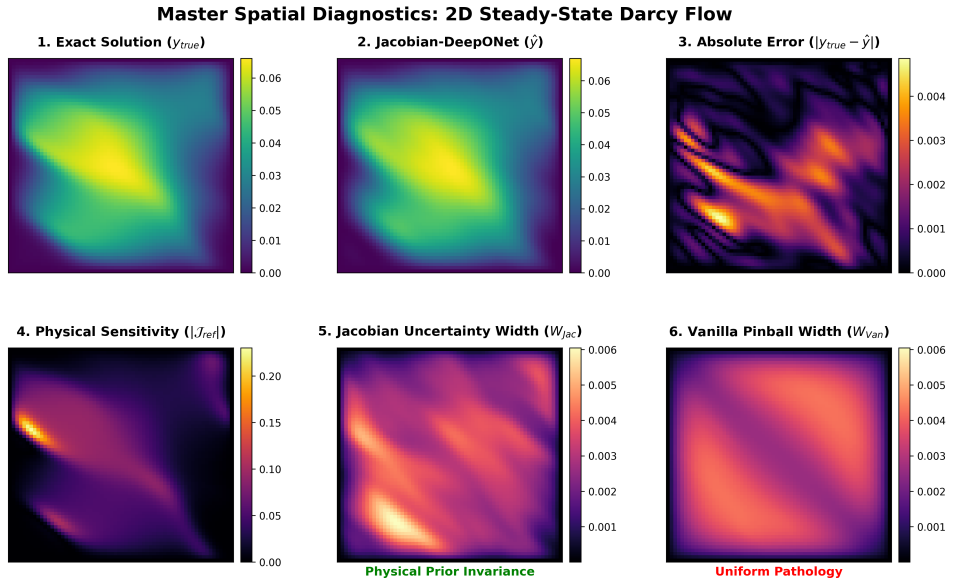


Figure 5: Master spatial diagnostics (2×3 grid) for 2D Steady-State Darcy Flow over the 64×64 Cartesian domain, Jacobian-DeepONet and constant-width pinball at $\lambda = 4$, seed 0. Top row: exact solution y_{true} (Panel 1), prediction $\hat{y}$ (Panel 2), absolute error $|y_{\text{true}} - \hat{y}|$ (Panel 3). Bottom row: reference sensitivity $|\mathcal{J}_{\text{ref}}|$ (Panel 4), Jacobian width W_{Jac} (Panel 5), constant-width W_{CW} (Panel 6), the last two on the shared scale $[0, 6.1 \times 10^{-3}]$, on the instance at the 75th percentile of MSE. Darcy is the one benchmark whose certified test split does not store $\partial u/\partial \kappa$, so all six panels come from the auxiliary set on which its ρ_{Sens} is measured: same generator and same distribution, with its own seed range. The stored quantity is $\partial u/\partial \kappa$ summed over the permeability field, that is the response to a uniform perturbation of κ.

their budget where it is not needed. The conditioned parameterizations widen instead: the Jacobian band by a factor of 10.3 from the smoothest decile to the sharpest and the conditional quantile head by 20.7, and both hold about 80% in D_{10}, twice the coverage of the unconditioned pair. Burgers is the benchmark where the reallocation goes furthest; on Darcy and Navier–Stokes it narrows the gap without closing it.

E.3 2D Steady-State Darcy Flow: Heterogeneous Porous Media

In 2D Darcy Flow, the operator maps random log-permeability fields $a(\mathbf{x})$, with permeability $\kappa(\mathbf{x}) = \exp(a(\mathbf{x}))$, to hydraulic head fields. High permeability channels produce localized preferential flow paths with steep spatial head gradients.

As reported in Table 7 (loss weight $\lambda = 4$, three seeds, 200 000 iterations), the Jacobian parameterization matches the deterministic operator in central predictive accuracy on the same backbone, $(3.037 \pm 0.442) \times 10^{-6}$ against $(3.013 \pm 0.108) \times 10^{-6}$: training an interval alongside the operator costs nothing here, but neither does it help, and the spread of the Jacobian column is dominated by one seed of the three. The intervals are sharp, MPIW $= (2.731 \pm 0.140) \times 10^{-3}$ with proxy-NLL -5.46 ± 0.11, and their width tracks both the physical sensitivity and the realized error, $\rho_{\text{Sens}} = 0.612 \pm 0.006$ and $\rho_{\text{Err}} = 0.614 \pm 0.010$. Coverage as trained is $83.22 \pm 0.81\%$, below the nominal level; as discussed in Subsection 5.1 this affects every parameterization on this benchmark and is removed by split-conformal rescaling, with spatial channel allocations shown in Figure 5.

In Figure 5 the reference sensitivity concentrates along the preferential permeability channels of this instance, a factor of six above its own mean (Panel 4), and the Jacobian band carves the same diagonals (Panel 5), where the realized error also sits (Panel 3). The constant-width band (Panel 6) carries a single smooth bubble instead. Two qualifications apply. That bubble is not noise: in physical units the constant-width band is the training standard deviation field up to a scalar, which peaks in the interior and vanishes on the Dirichlet boundary, and it is why constant width already reaches

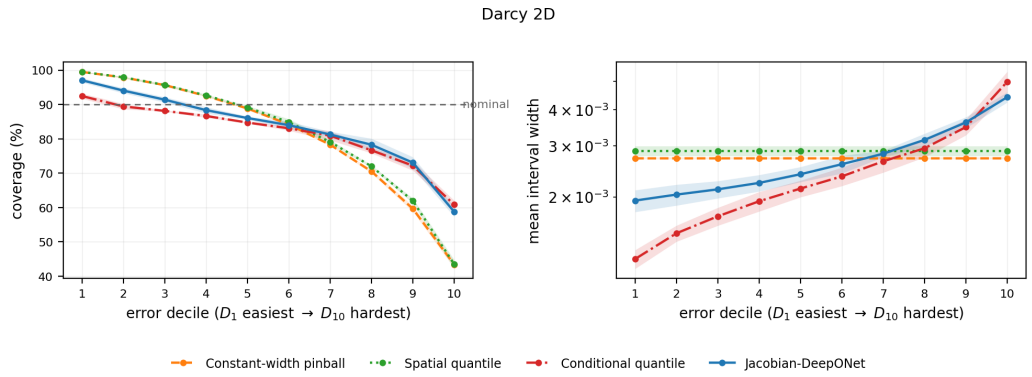


Figure 6: Darcy, same construction as Figure 4. The ordering is the same and the width still adapts, but on this benchmark no parameterization holds the nominal level as the error grows: coverage falls to 58.8% for the Jacobian band and 43.4% for the constant-width baseline in D_{10}, and the two adaptive curves cross the unconditioned ones at around D_6.

$\rho_{\text{Sens}} = 0.485$ on Darcy against 0.177 on Burgers. And the margin is benchmark dependent: on the easiest instances of this test set the reference sensitivity is itself a smooth interior profile, and the two parameterizations are then hard to tell apart by eye.

Figure 6 shows the mechanism, and also its limit on this benchmark. Coverage falls in every parameterization as the error grows, from 99.5% to 43.4% for the constant-width band and from 97.0% to 58.8% for the Jacobian one. The analytic width does widen where the operator is sensitive, by a factor of 2.3 against the flat profile of the unconditioned baselines, which is why the decline is some fifteen points shallower. But on Darcy that reallocation slows the loss of coverage rather than holding it flat as on Burgers, and the crossing point around D_6 shows what it costs: to cover the hard instances better the conditioned bands give up the over-coverage the flat ones enjoy on the easy ones.

E.4 2D Compressible Navier–Stokes: Coupled Gas Dynamics

The 2D Compressible Navier–Stokes equations model the nonlinear coupling of four conservative state variables $(\rho, \rho u, \rho v, E)$ under perturbations of the adiabatic exponent γ. Shock waves, contact discontinuities, and thermal expansion fans interact simultaneously across the spatial domain.

As reported in Table 7 (loss weight $\lambda = 4$, three seeds, 200 000 iterations on 20 000 training trajectories), Jacobian-DeepONet captures these coupled cross-sensitivities through its latent representation, attaining a central accuracy of MSE $= (1.076 \pm 0.015) \times 10^{-3}$ against $(1.094 \pm 0.009) \times 10^{-3}$ for a deterministic operator trained under mean-squared error alone on the identical backbone. The interval is therefore obtained at no cost in accuracy, although the margin is about one standard deviation and we do not read it as an improvement. Coverage as trained is PICP $= 88.12 \pm 0.21\%$ at an average width of MPIW $= (8.231 \pm 0.033) \times 10^{-2}$ and proxy-NLL -2.38, so the width comparison of Subsection 5.1 is made after split-conformal recalibration, as it is for every model here. The rank correlations are $\rho_{\text{Sens}} = 0.525 \pm 0.001$ and $\rho_{\text{Err}} = 0.519 \pm 0.005$, measured against the finite-difference sensitivity carried by the test split itself, and the corresponding spatial structure is visualized in Figure 7.

Navier–Stokes uses 20 000 training trajectories, twice the Darcy set. With 10 000 its held-out error rose from the earliest checkpoints under this schedule; with 20 000 it trains the full 200 000 iterations of the other benchmarks with no divergence in any seed of any parameterization.

The spatial distribution in Figure 7 highlights the multi-scale advantage of the proposed framework: the coupled thermodynamic sensitivity (Panel 4) reveals complex acoustic wave fronts across the gas flow. Jacobian-DeepONet (Panel 5) dynamically inflates interval widths specifically over high-gradient advective shock fronts, reaching $W_{Jac} = 0.194$ where the error concentrates and falling to 0.050 where it does not. This instance makes the comparison as clean as it gets: the two bands spend almost the same average width, 0.092 against 0.091, so what separates them is not how much they

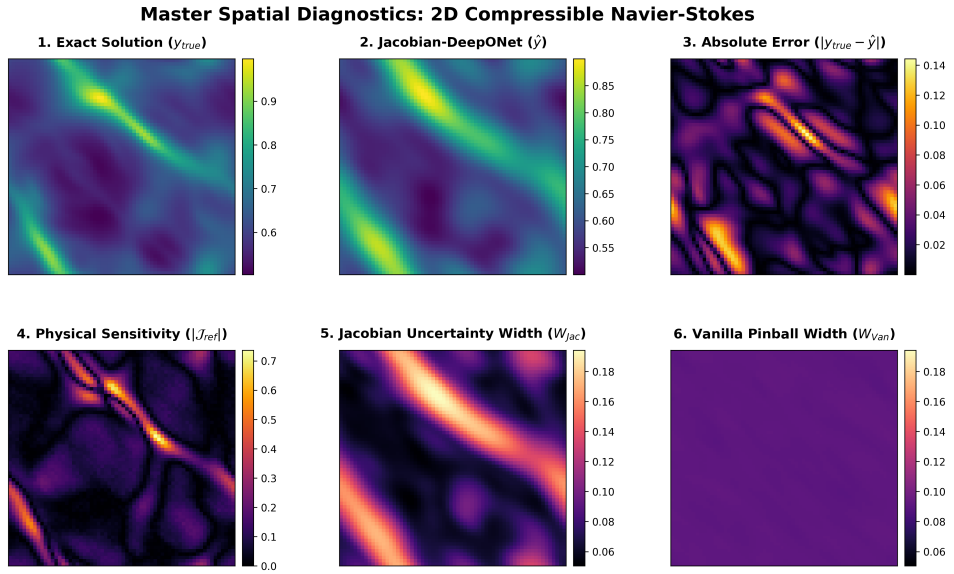


Figure 7: Master spatial diagnostics (2×3 grid) for the 2D Compressible Navier–Stokes system evaluated on a 64×64 Cartesian grid. Top row: Exact solution y_{true} (Panel 1), Jacobian-DeepONet prediction $\hat{y}$ (Panel 2), and absolute error $|y_{\text{true}} - \hat{y}|$ (Panel 3). Bottom row: Reference physical sensitivity $|\mathcal{J}_{\text{ref}}|$ (Panel 4), Jacobian-DeepONet uncertainty width W_{Jac} (Panel 5), and Constant-Width Pinball width W_{CW} (Panel 6). Panels 5 and 6 share the identical scale ($[0.050, 0.194]$), revealing how W_{Jac} conforms dynamically to the acoustic and shock structure while W_{CW} collapses into a flat sheet. Both at $\lambda = 4$, seed 0, on the instance at the 75th percentile of test MSE.

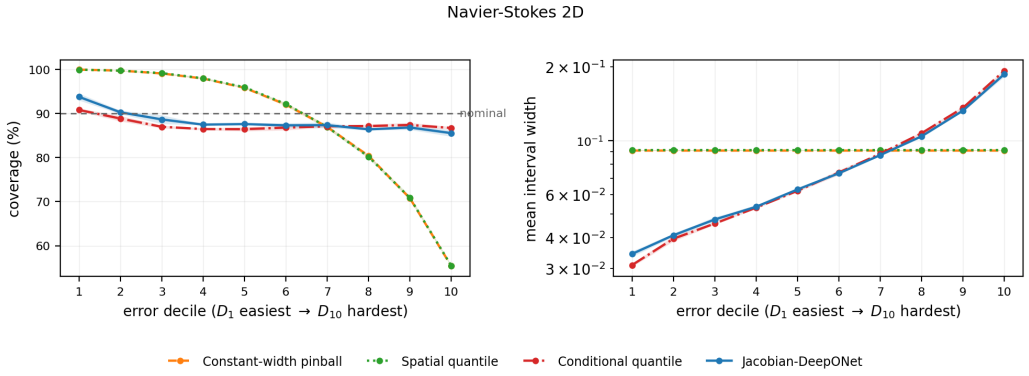


Figure 8: Navier–Stokes, same construction as Figure 4. This is the benchmark where conditioning holds up best: the Jacobian band runs from 93.8% to 85.6% and the conditional quantile head from 90.8% to 86.7% across the ten deciles, while the unconditioned pair starts at saturation and falls to about 55%. The widths on the right explain it, flat against a factor of five.

spend but where. Constant-Width Pinball (Panel 6) varies between 0.089 and 0.093 across the entire domain.

As depicted in Figure 8, the Jacobian band runs from 93.8% in D_1 to 85.6% in D_{10} while its width grows by a factor of 5.4 across the same range. The constant-width band saturates at 100% on D_1 and reaches only 55.4% on D_{10} at a flat width. This is the benchmark where conditioning comes closest to holding the nominal level throughout: the drop across the whole difficulty range is eight points for the Jacobian band and four for the conditional quantile head, against forty-five for the unconditioned pair.

E.5 Decile Behaviour Across Benchmarks

Uniform Bands vs. Sensitivity-Informed Structure: The comparative analysis across Figures 4–8 reveals the structural advantage of conditioning uncertainty on sensitivity information. Without a physical sensitivity prior, optimizing an unconditioned Pinball loss typically causes the network to learn a spatially uniform interval width. This can yield artificial over-coverage ($> 98\%$) on easy instances and under-coverage ($< 50 - 60\%$) on difficult instances. In contrast, by parameterizing bounds via the normalized Jacobian norm $s_{\mathrm{J}}(\mathbf{x}) = \sigma_{\mathrm{pred}}(\mathbf{x})/\overline{\sigma}_{\mathrm{EMA}}$, Jacobian-DeepONet dynamically expands intervals in high-sensitivity regions, providing evidence that sensitivity-informed uncertainty can better allocate uncertainty to spatial regions associated with larger surrogate errors.

Dynamic Interval Scaling Across Error Deciles: The differences in coverage are quantitatively revealed by decomposing the Mean Prediction Interval Width (MPIW) across error deciles ($D_1 \to D_{10}$), as presented in the right panels of Figures 4–8. In the constant-width baseline, the absence of spatial or instance conditioning forces the scalar quantile weights ($w_{\mathrm{low}}, w_{\mathrm{up}}$) to learn a static global offset, exhibiting a flat, horizontal MPIW profile across all deciles. Consequently, it allocates an identical interval width to low-error instances ($D_1 - D_3$) and high-error instances ($D_8 - D_{10}$) alike.

In contrast, by parameterizing bounds via the normalized Jacobian norm prior $s_{\mathrm{J}}(\mathbf{x}) \propto \|T(\mathbf{x})\,\mathcal{J}_{\mathrm{branch}}\|$, Jacobian-DeepONet establishes a direct physical coupling between local operator sensitivity and interval width. As demonstrated across all three benchmarks in the right panels of Figures 4–8, Jacobian-DeepONet exhibits an adaptive, monotonically ascending MPIW trajectory across error deciles:

- **1D Burgers**: the Jacobian width expands by a factor of 10.3 from D_1 to D_{10} at $\lambda = 4$, enveloping the propagating front. The constant-width band holds one width across all deciles and falls from 99.8% in D_1 to 39.1% in D_{10} (and spatial quantile to 35.5%), where the conditioned bands hold approximately 80%.
- **2D Darcy Flow**: the analytic width expands on high-contrast permeability realizations, by a factor of 2.3 across deciles, but both parameterizations lose coverage in the hardest deciles: 58.8% in D_{10} for the Jacobian bands against 43.4% for a static width. The ordering is preserved and the collapse is milder, yet neither is calibrated where the operator errs most.
- **2D Navier–Stokes**: the static width covers 100% of the easiest decile and only 55.4% of the hardest, spending most of its budget where the operator is already accurate. The Jacobian bands redistribute it, holding from 93.8% in D_1 to 85.6% in D_{10} while widening by a factor of 5.4, which is the most stable decile preservation among the benchmarks. At matched coverage, aggregate width is 3.5% sharper.

All decile figures in this subsection average the three seeds of each parameterization, and the shaded band is their sample spread rather than a confidence interval. The pattern is a reallocation of width towards the instances that need it, rather than a uniform inflation that would raise coverage everywhere at once.

F Comparison Against Calibrated Monte Carlo

F.1 Brute-Force Monte Carlo Sampling

This is the estimator deferred from Subsection 3.1. The gold-standard empirical approach to that pushforward variance draws M independent realizations $\varepsilon^{(1)}, \ldots, \varepsilon^{(M)} \sim \mathcal{N}(0, \sigma^2 I)$ and executes M sequential forward evaluations of the neural operator,

$$\mathrm{Var}_{\mathrm{MC}}(x_j) = \frac{1}{M-1} \sum_{m=1}^{M} \Big(\mathcal{G}_\theta(u + \varepsilon^{(m)})(x_j) - \bar{\mathcal{G}}(x_j) \Big)^2, \quad \bar{\mathcal{G}}(x_j) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{G}_\theta(u + \varepsilon^{(m)})(x_j) \tag{30}$$

which is consistent as $M \to \infty$ but requires M complete forward passes per input query. By the law of large numbers the empirical variance converges in probability to the true pushforward variance, and by Taylor's theorem that variance matches the analytic estimate of Subsection 3.1 up to $\mathcal{O}(\sigma^4)$,

$$\mathrm{Var}_{\mathrm{MC}}(x_j) \xrightarrow{p} \mathrm{Var}\left(\mathcal{G}_\theta(u + \varepsilon)(x_j)\right) = \mathrm{Var}_{\Delta}(x_j) + \mathcal{O}(\sigma^4). \tag{31}$$

The two estimate the same physical quantity, which is what licenses using Monte Carlo as empirical ground truth for the spatial uncertainty topology the single-pass extraction produces. The rest of this appendix reports that comparison.

F.2 Comparison with Monte Carlo-Based Intervals

We compare Monte Carlo and Jacobian-based interval shapes using the same frozen deterministic operator and identical calibration and evaluation splits. For each shape h, we calibrate a scalar multiplier q using the normalized residual score $s = |y - \hat{y}|/h$. This holds the central prediction and calibration protocol fixed, isolating the effect of the interval shape.

We draw perturbations $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ on the standardized branch input, using the same σ as in the Jacobian model. This matches the input covariance used in the latent Gram formulation: Monte Carlo estimates the nonlinear output variance, while the Jacobian provides its first-order approximation.

We evaluate the sampled standard deviation σ_{MC} in two ways. The last column of Table 10 reports uncalibrated intervals $\hat{y} \pm z_{90}\sigma_{\mathrm{MC}}$, where $z_{90} \approx 1.645$ is the Gaussian multiplier for a central 90% interval. The remaining Monte Carlo results use $h = \sigma_{\mathrm{MC}}$ as the interval shape and fit a multiplier q on the calibration set using normalized residuals, as in Table 11.

The distinction is that σ_{MC} measures output variability induced by input perturbations, rather than the surrogate's approximation error. At $M = 256$, the calibrated multipliers are 4.83 on Burgers, 13.39 on Darcy, and 12.45 on Navier–Stokes, substantially larger than z_{90}. These values indicate that the perturbation scale alone yields intervals too narrow to cover the observed residuals at the target level. Both Monte Carlo and the Jacobian therefore supply a spatial template whose overall scale is calibrated against prediction errors.

Table 10: Empirical verification of the Delta Method equivalence of Subsection 3.1. Spearman correlation between the Monte Carlo standard deviation and the analytic band over all sample-node pairs of the evaluation half, and the median of their ratio divided by σ. The analytic band is stated per unit perturbation, so that ratio equals 1 exactly when the first-order estimate is exact and departs from 1 by the $\mathcal{O}(\sigma^2)$ remainder. The last column is the coverage obtained by trusting the sampling scale directly, $\hat{y} \pm z_{90}\sigma_{\mathrm{MC}}$, with no conformal rescaling; its calibrated counterpart is the PICP column of Table 11, where the same σ_{MC} is used as a shape and a fitted multiplier supplies the scale. Three seeds, mean and sample standard deviation.

Benchmark	M	Spearman	Ratio / σ	PICP uncalibrated (%)
Burgers 1D	16	0.9827 ± 0.0004	0.9835 ± 0.0028	54.05 ± 1.10
	64	0.9959 ± 0.0000	0.9968 ± 0.0016	54.30 ± 0.94
	256	$\mathbf{0.9989 \pm 0.0001}$	$\mathbf{1.0004 \pm 0.0005}$	54.44 ± 0.92
Darcy 2D	16	0.9305 ± 0.0021	0.9793 ± 0.0029	23.23 ± 0.69
	64	0.9809 ± 0.0006	0.9953 ± 0.0007	23.60 ± 0.65
	256	$\mathbf{0.9950 \pm 0.0002}$	$\mathbf{0.9988 \pm 0.0006}$	23.69 ± 0.66
Navier–Stokes 2D	16	0.8600 ± 0.0064	0.9782 ± 0.0022	18.31 ± 0.34
	64	0.9621 ± 0.0023	0.9956 ± 0.0007	18.54 ± 0.30
	256	$\mathbf{0.9902 \pm 0.0003}$	$\mathbf{0.9986 \pm 0.0006}$	18.55 ± 0.29

Table 10 is the verification the Delta Method argument asks for. Agreement between the sampled standard deviation and the analytic band rises with the budget, reaching a Spearman correlation of 0.999 on Burgers, 0.995 on Darcy and 0.990 on Navier–Stokes at $M = 256$, while the median of their ratio, stated per unit perturbation, converges to within 0.002 of 1 on all three. At the tested perturbation scale, the Monte Carlo estimates closely agree with the first-order approximation. Increasing the sample count reduces sampling error, while the linearization error depends on the perturbation amplitude and the operator's nonlinearity. The same table shows that the sampling scale cannot be trusted on its own, reaching 54%, 24% and 19% coverage against a nominal 90%. That deficit is structural rather than statistical: it barely moves with the budget, from 54.05% to 54.44% on Burgers between $M = 16$ and $M = 256$, so additional draws estimate the same wrong scale more precisely rather than correcting it. Conformal rescaling is what closes the gap, which is why every

row of Table 11 carries it and why the two tables must be read as the uncalibrated and calibrated halves of one comparison.

Table 11: Monte Carlo perturbation against the analytic bands, on the same frozen deterministic operator and under the identical split-conformal procedure: same centre, same calibration and evaluation halves, same score $s = |y - \hat{y}|/h$ and a single multiplier q. Only the shape h differs, so the widths are directly comparable and every row sits at the nominal level by construction. Three seeds, mean and sample standard deviation. The perturbation is $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ on the standardized branch input, the same covariance the latent Gram assumes, with σ the value the Jacobian model trains with. Sampling converges to the analytic band rather than competing with it: at $M = 256$ the two agree to within 0.2% in width and 1% in both rank correlations, on all three benchmarks, at roughly eleven times the cost. JCB is the band of Subsection 3.3, $h = \sigma_{\text{pred}} + \beta\,\text{median}(\sigma_{\text{pred}})$; the last row of each block replaces σ_{pred} by $\log(1 + \sigma_{\text{pred}}/\overline{\sigma}_{\text{pred}})$ inside the same procedure, and is reported only as a variant. Latency is per sample on one NVIDIA A100, measured identically for every row.

Band	PICP (%)	MPIW	Interval score	ρ_{Err}	ρ_{Sens}	ms/sample
Burgers 1D (MPIW and interval score $\times 10^{-2}$)						
Constant width	90.22 ± 0.03	5.432 ± 0.103	16.872 ± 0.134	–	–	0.079
Constant width, Y_{std} profile	90.41 ± 0.16	4.440 ± 0.096	14.313 ± 0.104	0.497 ± 0.009	0.187 ± 0.000	0.079
Monte Carlo, $M = 16$	90.47 ± 0.08	4.466 ± 0.051	10.708 ± 0.153	0.643 ± 0.007	0.642 ± 0.005	1.546
Monte Carlo, $M = 64$	90.16 ± 0.10	4.255 ± 0.085	10.715 ± 0.175	0.651 ± 0.008	0.645 ± 0.008	2.824
Monte Carlo, $M = 256$	90.15 ± 0.12	4.212 ± 0.079	10.721 ± 0.170	0.653 ± 0.008	0.647 ± 0.007	7.917
Jacobian conformal band (JCB)	90.19 ± 0.15	4.213 ± 0.081	10.696 ± 0.165	0.654 ± 0.008	0.647 ± 0.008	0.742
JCB, log-compressed shape	90.29 ± 0.07	$\mathbf{3.850 \pm 0.056}$	11.916 ± 0.149	0.662 ± 0.007	0.607 ± 0.008	0.742
Darcy 2D (MPIW and interval score $\times 10^{-3}$)						
Constant width	91.10 ± 0.27	4.300 ± 0.035	7.543 ± 0.076	–	–	0.039
Constant width, Y_{std} profile	91.19 ± 0.23	$\mathbf{3.893 \pm 0.035}$	6.948 ± 0.077	0.425 ± 0.002	–	0.039
Monte Carlo, $M = 16$	91.29 ± 0.50	4.490 ± 0.107	7.310 ± 0.037	0.396 ± 0.005	–	0.984
Monte Carlo, $M = 64$	91.37 ± 0.45	4.379 ± 0.091	7.205 ± 0.049	0.403 ± 0.005	–	1.661
Monte Carlo, $M = 256$	91.35 ± 0.40	4.356 ± 0.083	7.186 ± 0.038	0.404 ± 0.005	–	4.359
Jacobian conformal band (JCB)	91.40 ± 0.39	4.357 ± 0.079	7.181 ± 0.047	0.405 ± 0.005	–	0.383
JCB, log-compressed shape	91.40 ± 0.35	4.156 ± 0.066	7.051 ± 0.058	0.413 ± 0.004	–	0.383
Navier–Stokes 2D (MPIW and interval score $\times 10^{-1}$)						
Constant width	88.25 ± 0.17	0.946 ± 0.003	1.636 ± 0.010	–	–	0.039
Constant width, Y_{std} profile	88.26 ± 0.17	0.946 ± 0.003	1.636 ± 0.010	0.006 ± 0.004	0.006 ± 0.000	0.039
Monte Carlo, $M = 16$	88.71 ± 0.21	0.919 ± 0.007	1.440 ± 0.008	0.361 ± 0.002	0.433 ± 0.006	0.996
Monte Carlo, $M = 64$	88.74 ± 0.19	0.891 ± 0.006	1.415 ± 0.009	0.405 ± 0.002	0.482 ± 0.006	1.668
Monte Carlo, $M = 256$	88.67 ± 0.13	0.882 ± 0.005	1.411 ± 0.010	0.417 ± 0.004	0.495 ± 0.009	4.377
Jacobian conformal band (JCB)	88.75 ± 0.10	0.883 ± 0.004	1.408 ± 0.010	0.420 ± 0.005	0.499 ± 0.009	0.388
JCB, log-compressed shape	88.54 ± 0.12	$\mathbf{0.875 \pm 0.004}$	1.456 ± 0.009	0.420 ± 0.005	0.499 ± 0.009	0.388

Inference Throughput and Computational Speedup: Table 11 is therefore not a contest the analytic band wins. At $M = 256$ the ensemble and the analytic band agree to within 0.2% in width and within 1% in interval score and both rank correlations, on all three benchmarks: 4.212 against 4.213 on Burgers, 4.356 against 4.357 on Darcy, 0.882 against 0.883 on Navier–Stokes. They are the same band. What separates them is the bill. One Jacobian extraction costs 0.742 ms per sample on Burgers and 0.39 ms on the two 2D benchmarks, against 7.92 and 4.4 ms for the converged ensemble, a factor of 10.7 to 11.4 paid once instead of M times. Truncating the budget does not close that gap, it degrades the estimate: at $M = 16$ the ensemble is both wider and less well ordered than the analytic band it is approximating, $\rho_{\text{Sens}} = 0.433$ against 0.499 on Navier–Stokes.

Two rows of the table cut against the construction and we report them as they stand. On Darcy the constant Y_{std} profile is the sharpest band of all, 3.893 against 4.357 for the Jacobian shape, with the better interval score and the better ρ_{Err}: on that benchmark the training variance field describes where the error lives better than the operator sensitivity does, and sampling does not recover the difference either, which is the point of running both. And compressing the shape logarithmically, $\log(1 + \sigma_{\text{pred}}/\overline{\sigma}_{\text{pred}})$ in place of σ_{pred}, buys width at the cost of score. That variant is not JCB, which Subsection 3.3 defines without the compression; we report it because it is the shape the trained head of Method 1 induces. It is the sharpest of the sensitivity-based shapes on every benchmark, but its interval score is worse than JCB on Burgers, 11.92 against 10.70, and on Navier–Stokes, 1.456 against 1.408: it concentrates the band further than the coverage it attains justifies. Darcy is the exception, where compression helps on both counts.

G Behaviour Under Distribution Shift

Every coverage figure in the body is in-distribution. This appendix asks what happens when the test distribution moves. Three shifts are generated by the same solvers at disjoint seed offsets, on the same backbone and with three seeds each: Burgers with the viscosity drawn from $\nu \sim \mathcal{U}(0.0005, 0.001)$, below the training minimum, which sharpens the shocks; Darcy with the log-permeability field amplitude raised from 0.5 to 0.75; and Navier–Stokes with the viscosity drawn below the training range, $\mu \in [0.0005, 0.001]$. They are severe: measured on the deterministic operator, they multiply the mean squared error by 15, 54 and 5.5 respectively.

The Burgers shift has to go downwards: the explicit scheme loses stability near $\nu = 0.12$ and the training range $\mathcal{U}(0.001, 0.1)$ already reaches 0.1, so there is no room above it.

Table 12: Behaviour under distribution shift, one shift per benchmark, three seeds, on the backbone of Table 5. Each shift is generated by the same solver at a disjoint seed offset. Burgers and Navier–Stokes receive the same experiment, the viscosity drawn below the training minimum, which sharpens the solution; Darcy has no scalar input parameter, its input is the whole coefficient field, so it receives the closest analogue, the field amplitude raised from 0.5 to 0.75. Measured on the deterministic operator the three multiply the mean squared error by 15, 54 and 5.5. PICP is as trained. The next three columns follow a split-conformal multiplier q fitted on one labelled half of the shifted set and evaluated on the other, so they describe the regime in which some labelled data from the new distribution is available. q is the most direct summary of a shift: the factor by which the band must be rescaled to recover the nominal level.

Model	MSE	PICP (%)	$\text{PICP}_{\text{conf}}$ (%)	q	$\text{MPIW}_{\text{conf}}$	ρ_{Err}	ρ_{Sens}
Burgers, viscosity below the training range (MSE $\times 10^{-3}$, MPIW $\times 10^{-2}$)							
Deterministic	23.075 ± 0.102	–	–	–	–	–	–
Constant-width pinball	22.106 ± 0.093	22.03 ± 0.28	89.53 ± 0.09	12.830 ± 0.143	41.193 ± 0.076	0.398 ± 0.001	0.331 ± 0.000
Spatial quantile	23.714 ± 0.111	20.24 ± 0.28	89.87 ± 0.07	15.929 ± 0.352	72.235 ± 1.347	0.418 ± 0.002	0.330 ± 0.000
Conditional quantile	24.804 ± 0.058	78.87 ± 1.06	90.28 ± 0.23	1.384 ± 0.044	40.975 ± 0.828	0.703 ± 0.005	0.463 ± 0.005
Jacobian-DeepONet	27.089 ± 0.761	72.11 ± 0.72	89.64 ± 0.20	1.683 ± 0.034	41.067 ± 1.077	0.632 ± 0.002	0.433 ± 0.006
Darcy, coefficient amplitude $\times 1.5$ (MSE $\times 10^{-4}$, MPIW $\times 10^{-2}$)							
Deterministic	1.627 ± 0.081	–	–	–	–	–	–
Constant-width pinball	1.279 ± 0.094	46.43 ± 0.63	88.92 ± 0.32	6.418 ± 0.071	1.745 ± 0.029	0.383 ± 0.002	0.389 ± 0.000
Spatial quantile	1.406 ± 0.247	45.98 ± 0.71	88.71 ± 0.32	6.089 ± 0.155	1.750 ± 0.017	0.394 ± 0.002	0.463 ± 0.008
Conditional quantile	1.391 ± 0.140	64.52 ± 1.46	89.38 ± 0.52	2.273 ± 0.101	1.785 ± 0.109	0.724 ± 0.012	0.473 ± 0.008
Jacobian-DeepONet	1.817 ± 0.708	56.63 ± 1.36	87.52 ± 0.66	3.696 ± 0.375	1.580 ± 0.109	0.613 ± 0.006	0.541 ± 0.019
Navier–Stokes, viscosity below the training range (MSE $\times 10^{-3}$, MPIW $\times 10^{-1}$)							
Deterministic	6.038 ± 0.015	–	–	–	–	–	–
Constant-width pinball	6.097 ± 0.041	54.28 ± 0.16	90.04 ± 0.10	2.656 ± 0.010	2.417 ± 0.001	-0.001 ± 0.001	-0.002 ± 0.000
Spatial quantile	6.072 ± 0.014	54.41 ± 0.09	90.03 ± 0.07	2.643 ± 0.006	2.413 ± 0.003	0.005 ± 0.002	0.000 ± 0.001
Conditional quantile	6.121 ± 0.013	85.95 ± 0.08	89.88 ± 0.08	1.106 ± 0.001	2.198 ± 0.003	0.437 ± 0.002	0.575 ± 0.005
Jacobian-DeepONet	6.118 ± 0.055	83.34 ± 0.54	89.85 ± 0.16	1.174 ± 0.011	2.183 ± 0.013	0.451 ± 0.002	0.585 ± 0.006

Three things follow from Table 12. The first is that a shift costs every parameterization its calibration, but not equally, and the split is exactly the conditioned against the unconditioned. Under the Burgers viscosity shift the constant-width band retains 22.0% coverage and the spatial quantile head 20.2%, against 72.1% for the Jacobian band and 78.9% for the conditional quantile head. The multiplier tells the same story in one number: the unconditioned bands need to be widened by 12.8 and 15.9 times to recover the nominal level, the conditioned ones by 1.68 and 1.38. A band whose width cannot depend on the input has no way to notice that the input has changed.

The second is that the error-tracking correlation keeps the conditioned bands apart from the unconditioned ones under every shift, though it is not immune to the shift itself. The Jacobian band reads $\rho_{\text{Err}} = 0.632$ on shifted Burgers against 0.746 in distribution, 0.613 on shifted Darcy against 0.614, and 0.451 on shifted Navier–Stokes against 0.519; the constant-width band falls to 0.398, 0.383 and -0.001. This is the distribution shift counterpart of the result in Appendix F: the shape of the band survives better than the scale does.

The third qualifies the width comparison. At matched coverage the Jacobian band is the narrowest of the four on shifted Darcy, 1.580×10^{-2} against 1.745 for constant width and 1.785 for the conditional quantile head, and on shifted Navier–Stokes, 2.183×10^{-1} against 2.198 and 2.417. On the Burgers shift the constant-width band, the conditional quantile head and the Jacobian band are tied, 41.19, 40.98 and 41.07×10^{-2}. What is uniform is the failure of the purely spatial head on Burgers, 76%

wider than the rest at 72.2×10^{-2}: a width profile fitted to the training geometry is the worst thing to carry into a new distribution.

H Ablation: Per-Batch Normalization of the Jacobian Scale

The Jacobian scale s_{J} of Subsection 3.2 is normalized by a frozen EMA. The simpler alternative divides by the current mini-batch mean; we report the comparison here because the difference is not one of accuracy.

Table 13 evaluates the same trained weights under both denominators. The mean squared error is identical in every cell, to the precision we report and beyond it, because the denominator scales the interval and the predicted centre does not depend on it. On the interval metrics the frozen statistic is slightly ahead as trained, with a better interval score on all three benchmarks, and after conformal rescaling the two are within 1% on Darcy and Navier–Stokes and 6.8% apart on Burgers. Read as a table of aggregate quality, the ablation is close to a tie.

That reading misses the point. The per-batch statistic is recomputed at every forward pass, evaluation included, so the interval a given input receives depends on which other inputs happen to share its batch. It is not a function of the input, and no aggregate metric computed at one fixed batch size can reveal that. The last column of the table measures it directly: we vary the evaluation batch size over $\{7, 13, 50, 128\}$ with the data held fixed and record the largest relative change in the prediction. Under the frozen denominator it is 5.4×10^{-6} to 6.6×10^{-5}, which is floating-point reduction-order noise. Under the per-batch mean it is 0.16 to 0.40: the same input receives an interval that differs by sixteen to forty percent depending on its batch, a factor of six thousand to fifty-seven thousand larger. The drift also falls as the batch grows, from 0.40 at batch 7 to 0.14 at batch 128 on Burgers, which is the signature of a sample mean converging rather than of a numerical artifact.

The protocol reaches the same verdict on its own. Our evaluation scripts refuse to record metrics for a model whose prediction is not invariant to batch size, and under this denominator they abort on all three benchmarks. The rank correlations are therefore absent from the table for the per-batch variant, and that absence is the finding rather than a gap: a correlation between interval width and error is only meaningful if the width is a property of the input. The frozen denominator removes the dependence at no parameter cost and is what we report everywhere else.

Table 13: Ablation of the Jacobian scale denominator, three seeds. Both variants are the same trained weights evaluated two ways: the frozen exponential moving average, fixed once training ends, and the per-batch mean, recomputed at every forward pass. The mean squared error is identical in every cell, and necessarily so: the denominator scales the interval and the centre does not depend on it. The first four columns are as trained and the next two follow the split-conformal rescaling of Table 1. The last column is the decisive one. It is the largest relative change in the prediction when the evaluation batch size is varied over $\{7, 13, 50, 128\}$ with the data held fixed, measured on the centre against its own amplitude and on the width against its own: for a well-defined predictor it is floating-point noise.

Scale denominator	MSE	PICP (%)	MPIW	Interval score	$\mathrm{PICP}_{\mathrm{conf}}$ (%)	$\mathrm{MPIW}_{\mathrm{conf}}$	Batch drift
Burgers 1D (MSE $\times 10^{-3}$, widths and interval score $\times 10^{-2}$)							
Frozen EMA (ours)	1.757 ± 0.044	93.42 ± 0.09	4.793 ± 0.060	7.673 ± 0.120	89.68 ± 0.44	3.984 ± 0.015	6.6×10^{-5}
Per-batch mean	1.757 ± 0.044	91.40 ± 0.15	4.765 ± 0.040	8.225 ± 0.140	89.46 ± 0.32	4.256 ± 0.036	$\mathbf{4.0 \times 10^{-1}}$
Darcy 2D (MSE $\times 10^{-6}$, widths and interval score $\times 10^{-3}$)							
Frozen EMA (ours)	3.037 ± 0.440	83.22 ± 0.81	2.731 ± 0.140	6.207 ± 0.250	90.34 ± 0.30	3.372 ± 0.086	1.3×10^{-5}
Per-batch mean	3.037 ± 0.440	82.98 ± 0.82	2.722 ± 0.140	6.258 ± 0.230	90.38 ± 0.25	3.397 ± 0.088	$\mathbf{1.6 \times 10^{-1}}$
Navier–Stokes 2D (MSE $\times 10^{-3}$, widths and interval score $\times 10^{-1}$)							
Frozen EMA (ours)	1.076 ± 0.015	88.12 ± 0.21	0.823 ± 0.003	1.115 ± 0.004	89.83 ± 0.16	0.891 ± 0.006	5.4×10^{-6}
Per-batch mean	1.076 ± 0.015	87.80 ± 0.52	0.821 ± 0.003	1.129 ± 0.007	89.66 ± 0.17	0.893 ± 0.007	$\mathbf{3.0 \times 10^{-1}}$