Derived Fields Increase the Recoverable Information in Budgeted Neural PDE Surrogates

Wenshuo Wang[1], **Fan Zhang**[2,3,4*]
[1] School of Future Technology, South China University of Technology, China
[2] State Key Laboratory of Ocean Sensing,
ZJU-Hangzhou Global Scientific and Technological Innovation Center,
Zhejiang University, Hangzhou, 311215, China
[3] Ocean College, Zhejiang University, Zhoushan, Zhejiang 316000, China
[4] Kavli Institute for Astrophysics and Space Research,
Massachusetts Institute of Technology, Cambridge, MA, 02139, USA
202364870251@mail.scut.edu.cn, f.zhang@zju.edu.cn

Abstract

For neural surrogates of time-dependent partial differential equations (PDEs), storage and memory budgets require training data to be obtained by coarsening and quantizing very-high-resolution numerical simulations. Fine-scale information can thus be lost before it reaches the model. We analyze this effect in periodic incompressible Navier–Stokes: quantization spreads error evenly over the retained modes, so the finest retained band, which holds the least energy, is the most corrupted, and velocity and vorticity are distorted differently under the same operator. Building on this analysis, we propose **Derived-Field Optimization (DerivOpt)**, which chooses which primitive and derived fields to carry from a supplied library, and how many bits each receives, by exactly minimizing a closed-form posterior risk under a calibrated channel model. This optimization problem always has an optimum and requires no knowledge of the governing equation. On PDEBENCH, DerivOpt not only lowers the pooled rollout nRMSE but also substantially reduces the fine-band error. These gains are already present in the predictor-free input state and add to those of a multiscale architecture. Which fields are carried, and at what precision, is thus an optimizable design variable upstream of the predictor that determines how much information a fixed budget leaves recoverable.

1 Introduction

Neural surrogates for time-dependent partial differential equations (PDEs) learn to advance the discretized state of a simulation by one step and are rolled out autoregressively in place of a numerical solver, at a fraction of its cost [1–5]. Existing neural simulators typically commit in advance to one state representation: primitive physical variables, hand-crafted derived variables, or latent coordinates [6, 7]; see Section 2. Whatever the representation, training data and activations must fit storage and GPU memory, so high-precision, high-resolution simulation data are usually coarsened and quantized before training. On a given GPU, the budget for the fields on which a neural PDE surrogate can be trained is fixed. If the state representation is not treated as fixed, this raises a structural question that is largely orthogonal to the choice of backbone: under a fixed storage budget, which physical fields should be carried, and how should the budget be split among them?

*Corresponding author.
Our code is available at https://github.com/VinceWangVW/DerivOpt.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To examine this question analytically and empirically, we use periodic incompressible Navier–Stokes as the clearest testbed and develop the canonical mechanism in Sections 3–5. Section 3 first shows that, under a fixed coarse operator and storage budget, modes above the retained cutoff are removed while quantization adds error evenly over the retained modes; because the finest retained band holds the least energy, its relative error is the largest, so the primitive state can already be detail-poor before it enters the model. Although they hold little energy, fine scales often govern the quantities that matter most in practice, such as peak local loads, front sharpness, mixing efficiency, boundary stress, and transport bottlenecks [8–10]. Section 4 then shows that this fine-band distortion is not merely a generic quantization effect: under the same operator, primitive and derived fields suffer systematically different retained-band distortions, so some carried states preserve fine-scale information more faithfully than others. Building on this observation, Section 5 introduces **Derived-Field Optimization** (`DerivOpt`), a budgeted state-design framework that treats the carried physical fields as optimization variables. Once a family-specific analysis basis, a calibrated channel model, and a task-aligned target are given, it assigns every feasible field subset and bit allocation an explicit closed-form score and finds the optimum exactly. Notably, the problem is PDE-agnostic, and building the candidate library requires no knowledge of the governing equation.

The empirical scope is broader than the canonical Navier–Stokes derivation. The same state-design problem arises across benchmark PDE families with different boundary conditions, field semantics, and rollout regimes, so Section 6 instantiates `DerivOpt` on the full time-dependent forward subset of PDEBENCH: advection, Burgers, diffusion–sorption, diffusion–reaction, radial dam break, incompressible Navier–Stokes, and compressible Navier–Stokes [11]. Across the seven families, four backbones, and nine budget–resolution regimes (252 configurations, five seeds), `DerivOpt` gives the lowest pooled rollout nRMSE in our comparison set, together with lower fine-band error and a longer detail-valid horizon. In incompressible Navier–Stokes, it lowers the predictor-free input fine-band error by 30%, so more of the fine scales remain recoverable at the same budget. The rollout gains are largest at the tightest budget, vanish once the budget suffices for the primitive state, and add to those of a multiscale architecture.

Section 7 returns to this broader perspective and argues that, in tightly budgeted neural simulation, the carried state determines how much information the budget leaves recoverable, whatever the predictor. Carried-state design should therefore be treated as a first-class design axis alongside architecture, loss, and rollout strategy, rather than as a fixed backdrop to predictor design.

2 Related Work

2.1 Neural PDE surrogates across primitive, derived, and latent representations

Existing neural simulators for time-dependent PDE forecasting can be organized not only by architecture, but also by the state representation they advance in time. Broadly, current work advances three kinds of representations: primitive physical fields, manually derived physical fields, and constructed reduced coordinates such as learned latent spaces or modal variables [6, 7].

One major paradigm advances primitive or directly observable variables. In this setting, neural simulators forecast future velocity-, pressure-, density-, or temperature-carrying states directly on grids or meshes [3, 12, 13]. Another paradigm advances manually derived physical variables. Here the state is reparameterized into fields that expose incompressibility, transport, or rotational structure more explicitly, such as vorticity, potential vorticity, or Helmholtz potentials [2, 14–17]. A third paradigm performs rollout in compressed coordinates. In these models, the high-dimensional flow is first encoded into low-dimensional latent or reduced coordinates, and temporal dynamics are learned in that space before decoding back to the physical domain [18–22].

Taken together, the dominant pattern in the literature is to commit to one fixed representation family before learning begins. Under storage constraints, however, this choice is not merely a modeling convenience; it determines how much fine-scale information survives coarsening and quantization. What is largely missing is work that treats the carried physical fields and their budget allocation as optimization variables under a fixed budget. `DerivOpt` addresses this gap directly. In tight-budget regimes, that design choice determines how much fine-scale information the predictor can recover in the first place.

2.2 Long-horizon and fine-scale fidelity in neural PDE surrogates

Beyond the choice of state representation, another central line of work asks how neural simulators can maintain accurate autoregressive rollouts over long horizons without washing out, destabilizing, or misrepresenting fine-scale structure [6, 23, 24].

Within this line, five broad baseline families are especially representative of how current work tries to preserve fine detail under long autoregressive rollouts. The first is direct primitive-state autoregression, which advances only the observable state and serves as the default starting point for most learned simulators [2, 3, 12, 13]. The second is explicit single-derived-state rollout, which asks whether one transparent derived channel such as vorticity, a Helmholtz component, or another family-specific field already repairs the primitive bottleneck [14–17]. The third keeps the carried state fixed but changes the training signal through rollout-aligned multiscale or spectral supervision [5, 25]. The fourth builds multiscale structure directly into the predictor architecture [26–29]. The fifth learns a compressed carried state end to end under an explicit storage budget, typically by coupling an encoder–decoder surrogate to latent or entropy-model compression [18, 20, 22, 30].

Taken together, dominant remedies remain model-side, loss-side, inference-side, or learned-bottleneck-side. They improve how a state is propagated, regularized, stabilized, or compressed, yet an overlooked issue is that, at the coarse resolutions typically used by budgeted simulators, the carried state may already lose much of the fine-scale information before rollout even begins. Once that state is detail-poor, downstream improvements can still reduce plain rollout error, but they generally cannot recover fine-scale information that is no longer present. In this sense, the central bottleneck under tight storage constraints often lies upstream of the predictor. `DerivOpt` therefore targets a more fundamental design axis: instead of assuming a fixed primitive rollout state and then improving propagation, it optimizes how a fixed state budget is distributed across primitive and derived fields so that fine-scale information is retained more faithfully from the outset.

3 Unrecoverable High-Frequency Information of Primitive Fields Under Coarsening and Quantization

We first isolate an error that arises before any learned predictor acts. Sections 3 and 4 use periodic incompressible Navier–Stokes, the most analyzable standard testbed, with the state construction of Appendix B. Let u be the fine-grid velocity in d dimensions, with Fourier coefficients $\hat{u}(k)$ at wavevectors k, N_f and N_c the fine and coarse resolutions per axis, $r = N_f/N_c$ the coarsening ratio, and $k_c = N_c/2$ the coarse cutoff, and define the retained band $\mathcal{B}_{\mathrm{exp}}$ and its top band $\mathcal{B}_{\mathrm{hf}}$ as

$$\mathcal{B}_{\mathrm{exp}} := \{k : |k| < k_c\}, \qquad \mathcal{B}_{\mathrm{hf}} := \{k : \gamma k_c < |k| < k_c\}, \qquad 0 < \gamma < 1. \tag{1}$$

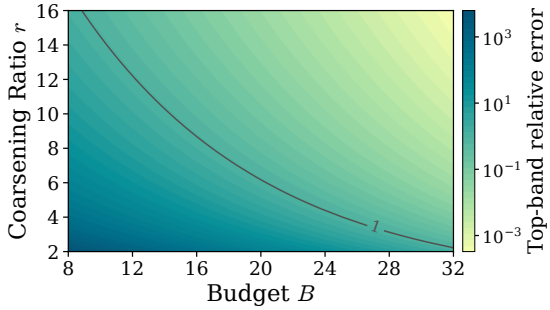
For the decoded primitive input $\tilde{u}$, i.e., at rollout step 0, the input fine-band error is

$$\mathrm{hf}_{L_2}(0)^2 := \frac{\sum_{k\in\mathcal{B}_{\mathrm{hf}}} \|\widehat{\tilde{u}}(k) - \hat{u}(k)\|_2^2}{\sum_{k\in\mathcal{B}_{\mathrm{hf}}} \|\hat{u}(k)\|_2^2}. \tag{2}$$

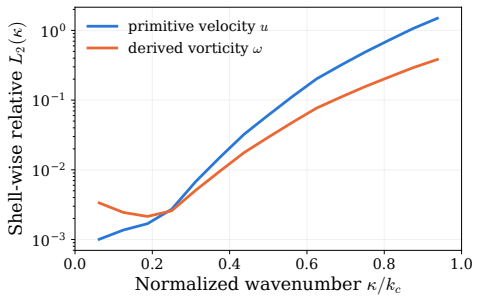
Standard primitive-state construction, which builds low-resolution training data for neural PDE surrogates from high-resolution simulation data, applies the spectral projector P_{exp} onto $\mathcal{B}_{\mathrm{exp}}$, samples the result on the coarse grid, quantizes each stored component j with b_j bits, $\mathbf{b} = (b_1, \ldots, b_d)$, and decodes by coarse trigonometric interpolation. Because the projector removes every mode outside $\mathcal{B}_{\mathrm{exp}}$, no unresolved mode folds back, and every retained coefficient satisfies $\widehat{\tilde{u}}(k) - \hat{u}(k) = \hat{\eta}(k)$, where $\hat{\eta}$ is the quantization noise lifted to the retained band. The numerator of Eq. (2) is thus the noise energy in $\mathcal{B}_{\mathrm{hf}}$, and the denominator is the signal energy there.

Under the standard high-rate closure (noise that is zero-mean, uncorrelated with the signal and across components, and white over the coarse grid), a uniform b_j-bit quantizer clipped at a standard deviations adds noise of variance $\frac{a^2}{3} 2^{-2b_j} \sigma_j^2$ to a component of variance σ_j^2 and spreads it evenly over the N_c^d coarse modes, so

$$\sum_{k\in\mathcal{B}_{\mathrm{hf}}} \mathbb{E}\|\hat{\eta}(k)\|_2^2 = N_c^d \, \vartheta_{\mathrm{hf}} \sum_{j=1}^{d} \frac{a^2}{3} \, 2^{-2b_j} \sigma_j^2, \qquad \vartheta_{\mathrm{hf}} := \frac{|\mathcal{B}_{\mathrm{hf}}|}{N_c^d}, \tag{3}$$



(a) Top-band relative error $D_q^{1/2}$ of the decoded primitive velocity; log color scale, contour at 1. INS setting of Section 6, with the spectrum fitted to its measured direct-decode input error at $B = 16$, $r = 8$ (Appendix A.1).



(b) Shell-wise relative error of the latent coefficient reconstructed from u alone or from ω alone under the same operator (Appendix A.2); $r = 8$, 8 bits per stored component (equal bits per component, not equal storage), setting as in (a).

where ϑ_{hf} is the fraction of coarse modes in $\mathcal{B}_{\mathrm{hf}}$ ($\vartheta_{\mathrm{hf}} \approx \frac{\pi}{4}(1-\gamma^2)$ in 2D). The signal energy in $\mathcal{B}_{\mathrm{hf}}$ is the fraction ρ_{hf} of the retained energy $N_c^d \sum_j \sigma_j^2$; for a shell spectrum $E(\kappa)$ with $\kappa = |k|$,

$$\sum_{k \in \mathcal{B}_{\mathrm{hf}}} \mathbb{E}\|\hat{u}(k)\|_2^2 = \rho_{\mathrm{hf}}(r,\gamma)\, N_c^d \sum_{j=1}^{d} \sigma_j^2, \qquad \rho_{\mathrm{hf}}(r,\gamma) = \frac{\int_{\gamma k_c}^{k_c} E\, d\kappa}{\int_0^{k_c} E\, d\kappa}. \tag{4}$$

For isotropic flow, $\sigma_j = \sigma$, and the ratio of Eqs. (3) and (4) gives

$$\mathbb{E}\big[\mathrm{hf}_{L_2}(0)^2\big] = \frac{\vartheta_{\mathrm{hf}}\, a^2}{3d\, \rho_{\mathrm{hf}}(r,\gamma)} \sum_{j=1}^{d} 2^{-2b_j} \geq D_q(B,r,\gamma) := \frac{\vartheta_{\mathrm{hf}}\, a^2}{3\, \rho_{\mathrm{hf}}(r,\gamma)}\, 2^{-\frac{2B}{d}}, \tag{5}$$

where $B = \sum_j b_j$ is the primitive budget in bits per retained site. By Jensen's inequality, the equal split attains the minimum over splits with the same B, which defines the top-band quantization distortion D_q. Equation (5) states the mechanism: quantization spreads error evenly over all coarse modes, but the top retained band holds only the fraction ρ_{hf} of the energy, so its relative error exceeds that of the whole field by the factor $(\vartheta_{\mathrm{hf}}/\rho_{\mathrm{hf}})^{1/2}$ and reaches order one at tight budgets. Appendix A.1 gives the full derivation.

Figure 1a evaluates $D_q^{1/2}$ from Eq. (5) over budgets B and coarsening ratios r in the INS setting of Section 6, computing ρ_{hf} at each r, with $k_c = N_f/(2r)$, from a shell spectrum fitted to the measured direct-decode input error at $B = 16$, $r = 8$. The error falls by a factor of two for each additional bit per velocity component, i.e., each increase of B by 2, and exceeds one at small budgets, including the 2 KiB INS state ($B = 16$, $r = 8$). It depends on r only through ρ_{hf}: a larger r lowers k_c and moves the top band to more energetic wavenumbers, so ρ_{hf} grows and the error falls; a steeper spectrum, which lowers ρ_{hf}, raises it at every budget. The error is fixed before prediction. Within the budgeted setting, a predictor, loss, or rollout scheme therefore starts from an input whose finest retained scales are corrupted, and it cannot reliably recover sample-specific detail that the state no longer contains.

4 Field-Dependent Information Preservation

In this section, we find that the quantization sensitivity of the finest retained modes depends on which field is stored. In 2D incompressible flow, the divergence-free degree of freedom $z(k)$ satisfies $\hat{u}(k) = e_\perp(k) z(k)$ and $\hat{\omega}(k) = i|k| z(k)$, where $e_\perp(k)$ is the unit vector orthogonal to k. Each channel c thus has a symbol M_c with $\hat{x}_c(k) = M_c(k) z(k)$, namely $M_u(k) = 1$ after projection onto $e_\perp(k)$ and $M_\omega(k) = i|k|$. Let $\hat{z}_c(k)$ be the latent coefficient reconstructed from channel c alone. Under the operator of Section 3, the normalized per-mode distortion is

$$\mathcal{D}_c(k) := \frac{\mathbb{E}|\hat{z}_c(k) - z(k)|^2}{S_z(k)} = \frac{\sigma_c^q(k; b_c)}{|M_c(k)|^2 S_z(k)}, \tag{6}$$

where $S_z(k) := \mathbb{E}|z(k)|^2$ and $\sigma_c^q(k; b_c)$ is the quantization variance of channel c at b_c bits. For velocity and vorticity,

$$\mathcal{D}_u(k) = \frac{\sigma_u^q(k; b_u)}{S_z(k)}, \qquad \mathcal{D}_\omega(k) = \frac{\sigma_\omega^q(k; b_\omega)}{|k|^2 S_z(k)}. \tag{7}$$

Velocity pays the full quantization penalty on every shell; vorticity, mapped back to the latent coefficient, divides it by $|k|^2$ and is therefore better conditioned near the cutoff (Appendix A.2). To visualize this difference, Figure 1b aggregates Eq. (6) over shells at $r = 8$ with 8 bits per stored component: velocity is less distorted on the low shells, vorticity near the cutoff, and the two curves cross at $\kappa \approx 0.24\, k_c$. Different fields thus carry different parts of the spectrum well under the same operator. Which fields to carry, and how to split a fixed number of bits among them, is therefore a design problem.

5 Derived-Field Optimization for Budgeted Neural Simulators

Sections 3 and 4 show, in the canonical periodic incompressible Navier–Stokes setting, that under a fixed budget the simulator state itself should be designed rather than fixed *a priori*. At the same time, the design rule is more general than this canonical specialization.

Let $\{\phi_\ell\}_{\ell \in \mathcal{I}}$ be an analysis basis adapted to the geometry and boundary conditions of a PDE family, $z(\ell)$ the latent coefficients of the state in that basis, and $\mathcal{C}$ a finite library of candidate fields. Each candidate c has m_c stored components per site and a linear relation $\hat{x}_c(\ell) = M_c(\ell) z(\ell)$, which is fitted on training data for candidates evaluated by physical operators (Appendix B.2). The library contains the primitive channels $\mathcal{P}$ and fields obtained from them by fixed operators, such as spectral powers of a boundary-adapted Laplacian, gradients, curl and divergence, or band projections (Appendix B.4). Building it uses the field types, the geometry, and the boundary conditions, but not the governing equation, its coefficients, or the identity of the PDE. Under a shared budgeted coarsen–quantize–decode operator (Appendix B.1), each decoded channel is modeled as

$$\hat{y}_c(\ell) = B_c(\ell) M_c(\ell) z(\ell) + n_c(\ell), \tag{8}$$

where the in-band transfer $B_c(\ell)$ and the covariance $\Sigma_c(\ell; b_c)$ of the residual $n_c(\ell)$ are estimated from training data for every admissible bit width b_c, with full covariance across the stored components.

For a target $\hat{g}(\ell) = L_g(\ell) z(\ell)$ on an index set $\mathcal{I}_\star$, a design is a nonempty subset $S \subseteq \mathcal{C}$ of fields to carry together with their bit widths $\mathbf{b} = (b_c)_{c \in S}$, and $\hat{g}_S(\ell)$ denotes the target decoded from the stored channels of S. **Derived-Field Optimization** (`DerivOpt`) chooses the design that recovers the target best on average within a budget of B bits per retained site:

$$\begin{aligned} (S^\star, \mathbf{b}^\star) &\in \arg\min_{(S,\mathbf{b}) \in \mathcal{D}_B} \sum_{\ell \in \mathcal{I}_\star} w_\ell\, \mathbb{E}\big\|\hat{g}_S(\ell) - \hat{g}(\ell)\big\|_2^2, \\ \mathcal{D}_B &:= \Big\{(S, \mathbf{b}) :\ \emptyset \neq S \subseteq \mathcal{C},\ \ b_c \in \mathbb{Z}_{\geq 1} \text{ for } c \in S,\ \ \textstyle\sum_{c \in S} m_c b_c \leq B\Big\}, \end{aligned} \tag{9}$$

where $w_\ell \geq 0$ are task weights and unused bits are zero padding. Stacking the selected channels gives the linear Gaussian model $\hat{y}_S(\ell) = A_S(\ell) z(\ell) + n_S(\ell)$, with $A_S(\ell) := \mathrm{rowstack}_{c \in S}[B_c(\ell) M_c(\ell)]$ and stacked residual covariance $\Sigma_S(\ell; \mathbf{b})$. Under the prior $z(\ell) \sim \mathcal{CN}(0, S_z(\ell))$, the error-minimizing decoder is the posterior mean $\hat{z}_S(\ell)$, so $\hat{g}_S = L_g \hat{z}_S$; this posterior mean is also the shared decoder of every explicit-state method in Section 6. Its error is set by the posterior covariance $\Sigma_{\mathrm{post},S}$, so the objective of Eq. (9) equals the design risk

$$\begin{aligned} \mathcal{J}(S, \mathbf{b}) &= \sum_{\ell \in \mathcal{I}_\star} w_\ell\, \mathrm{tr}\Big(L_g(\ell)\, \Sigma_{\mathrm{post},S}(\ell)\, L_g(\ell)^*\Big), \\ \Sigma_{\mathrm{post},S}(\ell) &= \Big(S_z(\ell)^{-1} + A_S(\ell)^* \Sigma_S(\ell; \mathbf{b})^{-1} A_S(\ell)\Big)^{-1} \end{aligned} \tag{10}$$

(Appendix A.3). $\mathcal{J}$ is the expected weighted error of recovering the target from the stored bits, so its minimum over $\mathcal{D}_B$ measures how much of the target a budget B leaves recoverable under the calibrated model. It involves only calibrated statistics and no predictor, so once the channels are calibrated every feasible design has a closed-form score, and Eq. (9) reduces to minimizing this score over $\mathcal{D}_B$.

Proposition 1 (Existence, exactness, and primitive fallback)**.** *Let $\mathcal{C}$ be finite and contain the nonempty set $\mathcal{P}$ of primitive channels, let $B \in \mathbb{N}$ with $B \geq \min_{c\in\mathcal{P}} m_c$, and let $S_z(\ell) \succ 0$. Then (i) $\mathcal{D}_B$ is finite and nonempty and $\mathcal{J}$ is finite on it, so a global minimizer $(S^\star, \mathbf{b}^\star)$ exists and finite search finds it exactly; (ii) $\mathcal{J}(S^\star, \mathbf{b}^\star) \leq \min\{\mathcal{J}(S, \mathbf{b}) : (S, \mathbf{b}) \in \mathcal{D}_B,\ S \subseteq \mathcal{P}\}$, with equality if and only if no design that carries a non-primitive channel has lower risk than the best primitive-only design; (iii) for libraries $\mathcal{C} \subseteq \mathcal{C}'$, the optimal risk over $\mathcal{C}'$ is at most that over $\mathcal{C}$.*

Proof sketch. Each b_c lies in $\{1, \ldots, \lfloor B/m_c \rfloor\}$ and $\mathcal{C}$ is finite, so $\mathcal{D}_B$ is finite, and it is nonempty because the primitive channel with the fewest components fits the budget at one bit per component. Since $S_z^{-1} \succ 0$ and $A_S^* \Sigma_S^{-1} A_S \succeq 0$, the matrix inverted in Eq. (10) is positive definite, so $\mathcal{J}$ is finite, and a finite nonempty set attains its minimum. The primitive-only designs form a subset of $\mathcal{D}_B$, and $\mathcal{D}_B(\mathcal{C}) \subseteq \mathcal{D}_B(\mathcal{C}')$; a minimum over a superset cannot be larger. Appendix A.4 gives the details and an extended Navier–Stokes library. □

Proposition 1 makes `DerivOpt` applicable to any PDE family and any finite library for which Eq. (8) can be calibrated. The problem is PDE-agnostic: its formulation, calibration, and solution use only the candidate library and training snapshots, never the governing equation. No winning field has to be specified in advance, and a library without useful candidates returns the best primitive-only design. The proposition concerns the calibrated risk; whether a lower risk yields better rollouts is tested in Section 6. We solve Eq. (9) with an exact branch-and-bound search that prunes only by valid lower bounds on $\mathcal{J}$ and breaks ties toward lower cost, never by outcome metrics (Appendix B.3).

Periodic incompressible Navier–Stokes. Returning to the canonical example of Sections 3 and 4, the design risk can be written in closed form. With velocity as the target, incompressibility leaves $d-1$ latent degrees of freedom per mode and velocity is an isometry of them, so $\mathcal{J} = \sum_{k\in\mathcal{B}_\star} w(k) \operatorname{tr} \Sigma_{\mathrm{post},S}(k)$, where $\mathcal{B}_\star$ is the set of target modes. For $\mathcal{C} = \{u, \omega\}$ with isotropic residuals $\Sigma_c = \sigma_c^{\mathrm{eff}} I$, the channel precisions add, and the score becomes

$$\mathcal{J}_{\mathrm{NS}}(S, \mathbf{b}) = (d-1) \sum_{k\in\mathcal{B}_\star} w(k) \left(S_z(k)^{-1} + \mathbf{1}_{\{u\in S\}} \frac{|B_u(k)|^2}{\sigma_u^{\mathrm{eff}}(k; b_u)} + \mathbf{1}_{\{\omega\in S\}} \frac{|B_\omega(k)|^2 |k|^2}{\sigma_\omega^{\mathrm{eff}}(k; b_\omega)} \right)^{-1}. \quad (11)$$

The bracket is the posterior precision of mode k: the prior precision $S_z(k)^{-1}$ plus the calibrated signal-to-noise ratio contributed by each carried field, and $\mathcal{J}_{\mathrm{NS}}$ sums the resulting posterior variances over the target modes with weights $w(k)$. The vorticity term carries the extra factor $|k|^2$ of Section 4, so its contribution grows toward the cutoff, whereas that of velocity does not. Each contribution costs $m_c b_c$ bits, and the optimizer trades the precision a field adds against these bits. Appendix A.3 derives Eq. (11), and Appendix C gives the instantiations for the other families.

6 Empirical Evaluation

We test two claims in the budgeted setting. The mechanism claim is that the carried state is already detail-poor at input and that derived fields reduce this input error. The outcome claim is that this loss carries into rollouts, where predictor- or codec-side alternatives at the same budget do not recover it, whereas `DerivOpt` reduces it.

6.1 Experimental setup

Benchmark. We use the seven time-dependent forward families of PDEBENCH [11]: 1D advection, Burgers, and diffusion–sorption, and 2D diffusion–reaction, radial dam break, incompressible Navier–Stokes (INS), and compressible Navier–Stokes. They cover periodic and non-periodic boundaries and scalar and multi-field states. Each family uses 1,000 trajectories split 640/104/256 into training, validation, and test, with one test cohort shared by all methods, budgets, and seeds. PDEBENCH INS is wall-bounded; it is the closest benchmark counterpart of Sections 3–5 rather than an instance of the periodic analysis. Grids and time windows are listed in Appendix D.

Budget. A configuration fixes a family, a backbone, a budget ratio $R \in \{0.25, 0.5, 1\}$, and a per-axis retained fraction $f \in \{1/8, 1/4, 1/2\}$, giving $7 \times 4 \times 3 \times 3 = 252$ configurations. The state

passed between autoregressive steps keeps $n_r = (fN)^d$ sites, where N is the reference resolution per axis, and may use $B = 32\, m_p R$ bits per site, where m_p is the number of primitive scalar components; that is, the fraction R of FP32 primitive storage at that resolution. Every serialized bit counts against the cap $n_r B$, including field codes, latent codes, headers, and padding, whereas statistics fixed in the checkpoint cost nothing (Appendix D). The canonical regime $(R, f) = (0.25, 1/8)$ was fixed on a training-split INS pilot before any test evaluation (Appendix D); for INS it gives 32^2 sites at 16 bits per site, i.e., 2,048 B per state.

Methods. All methods share one predictor interface: the network reads and predicts decoded coarse primitive fields, derived fields are never added as network inputs, and the external state passed between steps is the encoded payload. The state-design ladder consists of `Primitive` (primitive fields only, bits split evenly), `BestSingle` (the best single derived field of the library, with all bits), `DerivBase` (the field subset selected by `DerivOpt`, with bits split evenly across its components and the remainder padded), and `DerivOpt`. All four use the shared posterior decoder, so differences within the ladder come from the carried state alone.

We also compare three strong baselines for high-frequency recovery at the same cap, one from each of the other three families of Section 2.2: `ArchMulti`, a multiscale variant of each backbone in the spirit of [26, 27]; `RolloutMulti`, rollout-aligned multiscale supervision in the spirit of [5, 25]; and `LatentAE-Hyper`, a latent autoencoder [18] with a hyperprior entropy model [30] whose realized bitstream is rate-controlled to the cap. `ArchMulti` and `RolloutMulti` carry the `Primitive` state. `DerivOpt` is run once per family and regime on training data (63 selector runs), with the libraries of Appendix B.4, the primitive field as the target, and weights w_ℓ fixed in advance at 0.35 on the low and middle bands and 0.65 on the top band, normalized within each group by training statistics.

Training. The backbones are FNO, U-Net, ConvLSTM, and Transformer, each trained as a one-step autoregressive predictor. For every method and configuration, 12 hyperparameter trials (learning rate, weight decay, and gradient clipping), each of 10,000 updates, are compared on the validation split with seed 11; the selected setting is then trained with four further seeds (29, 47, 71, 103). We report means over the five seeds.

Metrics. In each family's boundary-adapted basis, let P_{exp} project onto the retained band and P_{fine} onto its top band, $(0.75\, k_c, k_c)$. For the primitive target x_h at step h and its decoded or predicted estimate $\hat{x}_h$,

$$\mathrm{hf}_{L_2}(h) = \frac{\|P_{\mathrm{fine}}(\hat{x}_h - x_h)\|_2}{\|P_{\mathrm{fine}} x_h\|_2}, \qquad H = \frac{1}{T} \max\big\{\tau \le T : \text{ steps } 1, \dots, \tau \text{ are all valid}\big\}. \tag{12}$$

The input error $\mathrm{hf}_{L_2}(0)$ is measured on the decoded input with no predictor. Over closed-loop rollouts that run to the end of each stored trajectory, $h = 1, \dots, T$ (Appendix D), nRMSE and mean hf_{L_2} sum squared errors and squared reference norms over the 256 test trajectories and all T steps and take the square root of their ratio, for the full field and for the top band, respectively. The detail-valid horizon $H \in [0, 1]$ is computed per trajectory ($H = 0$ if step 1 is invalid) and averaged over trajectories. A step is valid when the prediction is accurate and spectrally faithful at once: $\mathrm{expr}_{L_2}(h) < 1$ on the retained band, $\mathrm{hf}_{L_2}(h) < 1$, the fine-band energy ratio $Q_{\mathrm{fine}}(h) \in [0.8, 1.25]$, and the out-of-band leakage $e_{\mathrm{out}}(h) < 0.1$ (definitions in Appendix D); one threshold set is used for all families, methods, and backbones. Pooled values average over configurations and then over seeds, and paired 95% intervals use a 12,000-sample hierarchical bootstrap over family, configuration, and paired seed.

6.2 Main results

6.2.1 Input-stage mechanism in the canonical case and across the full suite

The first column of Table 1 removes the predictor. At the 2 KiB INS budget, the primitive state decoded by the shared posterior decoder has $\mathrm{hf}_{L_2}(0) = 0.999$: the error in the top retained band is as large as the signal, so the predictor receives essentially none of the information in this band. Decoding the same bits directly, without fusion, gives 1.394, so the shared decoder already helps and is not the source of the gap. With the decoder, the target, and the inputs fixed, changing only the carried state lowers $\mathrm{hf}_{L_2}(0)$ to 0.863 with vorticity alone (`BestSingle`), to 0.830 with velocity and

Table 1: INS in the canonical regime (lower is better). Carried states: `Primitive`, `ArchMulti`, and `RolloutMulti` u 8+8; `BestSingle` ω 16; `DerivBase` u 5+5, ω 5 (one padding bit); `DerivOpt` u 4+4, ω 8; `LatentAE-Hyper` a learned code padded to the cap. The encoder of `LatentAE-Hyper` is trained with each backbone, so its $\mathrm{hf}_{L_2}(0)$ averages the four backbones within each seed.

	Input	FNO		U-Net		ConvLSTM		Transformer	
Method	$\mathrm{hf}_{L_2}(0)$	nRMSE	mean hf_{L_2}	nRMSE	mean hf_{L_2}	nRMSE	mean hf_{L_2}	nRMSE	mean hf_{L_2}
`Primitive`	0.999	0.0680	0.831	0.0749	0.855	0.1054	0.889	0.0943	0.893
`BestSingle`	0.863	0.0661	0.755	0.0718	0.778	0.0989	0.800	0.0896	0.820
`DerivBase`	0.830	0.0644	0.682	0.0697	0.704	0.0971	0.727	0.0867	0.746
`DerivOpt`	**0.700**	0.0585	**0.584**	**0.0634**	**0.597**	**0.0880**	**0.634**	**0.0785**	**0.630**
`ArchMulti`	0.999	**0.0582**	0.685	**0.0634**	0.700	0.0883	0.720	0.0791	0.734
`RolloutMulti`	0.999	0.0675	0.794	0.0720	0.815	0.1008	0.820	0.0906	0.855
`LatentAE-Hyper`	0.812	0.0633	0.672	0.0692	0.689	0.0962	0.716	0.0868	0.731

Table 2: All 252 configurations (seed standard deviations are at most 1×10^{-4} for nRMSE, 0.0022 for mean hf_{L_2}, and 0.002 for H). (a) Pooled over families, backbones, and regimes. (b) Pooled by budget ratio R; `ArchMulti` carries the `Primitive` input, and at $R = 1$ `DerivOpt` selects the primitive encoding. Lower is better except for H.

(a) Pooled

Method	nRMSE	mean hf_{L_2}	H
`Primitive`	0.0673	0.764	0.384
`BestSingle`	0.0678	0.743	0.375
`DerivBase`	0.0650	0.701	0.451
`DerivOpt`	**0.0614**	**0.653**	**0.524**
`ArchMulti`	0.0646	0.699	0.458
`RolloutMulti`	0.0664	0.742	0.410
`LatentAE-Hyper`	0.0649	0.696	0.463

(b) By budget ratio

	Method	$R = 0.25$	0.5	1
$\mathrm{hf}_{L_2}(0)$	`Primitive`	1.0057	0.2913	0.0205
	`DerivOpt`	**0.7099**	**0.1008**	0.0205
nRMSE	`Primitive`	0.0735	0.0672	0.0611
	`ArchMulti`	0.0672	0.0656	**0.0610**
	`DerivOpt`	**0.0621**	**0.0609**	0.0611
H	`Primitive`	0.127	0.385	0.641
	`ArchMulti`	0.316	0.404	**0.653**
	`DerivOpt`	**0.446**	**0.484**	0.641

vorticity at equal bits (`DerivBase`), and to 0.700 with `DerivOpt`'s allocation, a 29.9% reduction. Field mixing and allocation both contribute, and only with the optimized allocation is the carried state more accurate than the learned code of `LatentAE-Hyper` at the same cap (0.812).

In this wall-bounded flow, vorticity alone is the most accurate in the interior but the least accurate near the walls (0.689 and 1.300, against 1.054 and 0.777 for `Primitive`), whereas `DerivOpt` is the most accurate near the walls (0.605) and overall, though not everywhere (Appendix E); each field is thus the more accurate one where the other is weak, and the mixed state combines them.

Across the suite (Table 2b), `DerivOpt` lowers the pooled $\mathrm{hf}_{L_2}(0)$ from 1.006 to 0.710 at $R = 0.25$ and from 0.291 to 0.101 at $R = 0.5$; at $R = 1$ it selects the primitive encoding, and the input is unchanged. The input loss of the primitive state thus grows as the budget tightens, as Section 3 predicts, and derived fields reduce it at both tighter budgets. Under a fixed budget, the fine-scale information available to the predictor is therefore set by the carried state before any learning, and carrying derived fields with an optimized bit allocation makes more of it recoverable from the same bits, except when the budget already suffices for the primitive state.

6.2.2 Canonical incompressible Navier–Stokes case and full-suite aggregate effectiveness

The input deficit carries into rollouts. In INS (Table 1), `DerivOpt` has the lowest mean hf_{L_2} on all four backbones, 0.584–0.634 against 0.672–0.731 for the next best, `LatentAE-Hyper`, and on every backbone the four ladder states have the same order in mean hf_{L_2} as in $\mathrm{hf}_{L_2}(0)$: for a given backbone, a more accurate input gives a more accurate rollout. `DerivOpt` is also below `ArchMulti` and `RolloutMulti`, which change the predictor or the loss but receive the primitive input; a stronger predictor or a rollout-aligned loss improves on `Primitive` but does not make up for what the

primitive input lacks. Its nRMSE is close to that of `ArchMulti`, lower on three of the four backbones in the five-seed mean and in 12 of the 20 backbone–seed pairs, and its horizon H exceeds that of `ArchMulti` in all 20 pairs: the two are comparable on the full field, but `DerivOpt` stays detail-valid for longer.

Across the 252 configurations (Table 2a), `DerivOpt` has the lowest pooled nRMSE, the lowest pooled mean hf_{L_2} (0.653, against 0.696 for `LatentAE-Hyper`), and the highest H. Its nRMSE is 5.0% below that of `ArchMulti` and 5.4% below that of `LatentAE-Hyper`, the two strongest alternatives, with paired differences of -0.0032 $[-0.0053, -0.0014]$ and -0.0035 $[-0.0055, -0.0016]$ (lower in 162 and 161 of the 252 configurations), and its H is higher by 0.066 $[0.037, 0.096]$ and 0.061 $[0.032, 0.092]$. `DerivOpt` also improves pooled nRMSE and H over `ArchMulti` in each of the seven families. The INS pattern thus holds across the suite, with a moderate gain in full-field error and a larger one in the detail-valid horizon. The ladder shows where the gain comes from. A single derived field lowers the mean hf_{L_2} (0.743 for `BestSingle` against 0.764 for `Primitive`) but not the pooled nRMSE or H, on which `BestSingle` is slightly worse than `Primitive`. Mixing primitive and derived fields at equal bits improves all three metrics (`DerivBase`), and optimizing the allocation improves all three again (`DerivOpt`). Replacing the primitive state by a derived field is thus not enough; the gain comes from carrying both and from allocating the bits between them.

The gains depend on the budget (Table 2b). At $R = 0.25$, `DerivOpt` lowers nRMSE by 15.5% relative to `Primitive` and by 7.5% relative to `ArchMulti`, and it raises H from 0.127 and 0.316 to 0.446. At $R = 0.5$, its nRMSE is 7.2% below that of `ArchMulti` in aggregate (paired difference -0.0047 $[-0.0104, -0.0002]$); the gain comes from diffusion–reaction, INS, and compressible Navier–Stokes, while the other four families select the primitive encoding, and `DerivOpt` has the lower nRMSE in only 36 of the 84 configurations at this budget, so the improvement there is an aggregate one. At $R = 1$, the selected state is the primitive state, the gain from state design is zero, and the small lead of `ArchMulti` reflects its predictor. The rollout gain thus follows the input deficit of Section 6.2.1: it is largest where the primitive state loses most at input and vanishes once the budget suffices for that state. Under a fixed budget, the fine-scale loss of the carried state therefore persists into the rollout, and `DerivOpt`, which reduces this loss at its source, outperforms the predictor-side and codec-side alternatives at the same budget in both fine-band error and detail-valid horizon.

6.2.3 Generality, complementarity, and stability

No equation-level knowledge. We replace the seven expert libraries by one shared library generated by a frozen rule from tensor type, dimension, and boundary type alone: each primitive scalar, four spectral powers of the boundary-adapted Laplacian, and four band projections, plus divergence and curl for vector fields and a DCT-II rotation across the components of multichannel states (120 channels, 108 of them derived; Appendix B.4). In the canonical regime with FNO, `DerivOpt` selects the same effective encodings as with the expert libraries and assigns zero bits to 100 of the 108 derived channels, so all results are unchanged. Averaged over the seven families, `DerivOpt` reaches nRMSE 0.0511 and H 0.486, against 0.0602 and 0.162 for `Primitive`, 0.0576 and 0.256 for `BestSingle`, and 0.0563 and 0.342 for `DerivBase` under the same library. The gains therefore do not depend on hand-picking a winning field per PDE.

Non-uniform grids. On the CylinderFlow meshes of MeshGraphNets [3], an input-stage mesh protocol keeps 25% of the nodes (e.g., 1,888 to 472) at 16 bits per retained node, i.e., 944 B per state. `DerivOpt` selects u 5+5, ω 6 and lowers the input $\mathrm{hf}_{L_2}(0)$ from 0.992 (`Primitive`, shared decoder) to 0.649 ± 0.015 over five runs, so the input-stage gain does not require a uniform grid. We evaluate this case at the input stage only.

Complementarity. Feeding the `DerivOpt` state to `ArchMulti` requires no change to either method. On INS, `DerivOpt+ArchMulti` improves on both single methods on each of the four backbones, with macro-average nRMSE 0.0703 and H 0.522, so state design adds to predictor design rather than replacing it.

Stability and selected designs. `DerivOpt` has the highest H in 130 of the 252 configurations (10 of them ties). It keeps the highest pooled H under stricter validity thresholds ($Q_{\mathrm{fine}} \in [1/1.15, 1.15]$, $e_{\mathrm{out}} < 0.075$), looser ones ($Q_{\mathrm{fine}} \in [1/1.5, 1.5]$, $e_{\mathrm{out}} < 0.15$), and a wider fine band $(2k_c/3, k_c)$ (Appendix E), so its horizon gain does not hinge on these choices. Of the 63 selected designs, 30

carry derived fields: all 21 at $R = 0.25$, 9 at $R = 0.5$, and none at $R = 1$; the tighter the budget, the more often derived fields are worth their bits. Appendix E lists the canonical designs.

7 Discussion

We showed that under a fixed storage budget, a neural PDE surrogate first loses fine-scale information in its carried state, and that how much is lost depends on which fields are carried and how their bits are allocated. Choosing them with `DerivOpt` makes more of this information recoverable from the same bits: the fine-band error falls already at the input and stays lower in rollouts, which reach the longest detail-valid horizons and the lowest pooled full-field error across seven PDE families and four backbones; the gain adds to that of a multiscale architecture, and the primitive state is kept once the budget suffices for it. Because no predictor can reliably recover sample-specific detail that the carried state no longer contains, the representation supplied to a learned simulator is part of the learning problem, and carried-state design should be treated as a first-class design axis alongside architecture, loss, and rollout strategy rather than as a fixed backdrop to predictor design.

Notably, this axis gains weight as surrogates scale. A 500-step trajectory of velocity and pressure on a 1024^3 grid occupies 7.81 TiB in FP32, and a single width-20 FNO feature map at that resolution needs 80 GiB, the entire memory of an 80 GB GPU. Coarsening by eight per axis brings the feature map to 160 MiB, and storing the coarsened state at 8 bits per component brings the trajectory to 3.91 GiB; these are the budget ratio and retained fraction of the canonical regime of Section 6, at which the primitive INS state had already lost its top retained band. At such scales, reducing the state is a precondition for training rather than an option, so the choice of carried fields decides how much fine-scale physics a surrogate can learn at all. Numerical methods have long chosen their prognostic variables for the structure they preserve; for learned simulators under a bit budget, `DerivOpt` makes the same choice by optimization, before any network is trained, for any predictor, and without knowledge of the governing equation.

The main limitation is that `DerivOpt` selects among candidate fields but does not construct them. Its improvement over the best primitive-only design requires the library to contain at least one field that preserves the target band better per bit than the primitive state; otherwise it returns that design, and calibration and selection add cost without gain. What matters is therefore the coverage of the library, whereas redundancy does not hurt the calibrated optimum (Proposition 1) and only enlarges the search. A library generated without equation-level knowledge sufficed in our experiments (Section 6.2.3), but this does not guarantee a useful candidate for every PDE; for less understood dynamics, candidates may have to come from operator libraries built or learned from data, such as PDE-FIND and PDE-Net [31, 32], and learning candidates jointly with the selection remains open.

The selected design also depends on calibrated transfer and residual statistics under a linear-Gaussian channel model: if the calibration data do not represent the deployment regime or the model is misspecified, the selected subset and allocation can change, and a lower calibrated risk need not give better rollouts. The size and spectral distribution of the input loss depend on the storage operator; our experiments cover one family of storage operators, basis truncation followed by uniform scalar quantization, and one learned codec, and other coarseners, quantizers, and codecs require recalibration and were not tested. The claims concern explicit-state autoregressive surrogates under tight budgets and do not imply that all fine-scale error of neural simulators originates in the carried state; outside this setting, model expressiveness can remain the dominant bottleneck. Carrying derived fields also adds per-step encoding and fusion work, the non-uniform-mesh case was evaluated at the input stage only, and 3D benchmarks remain future work.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant No. 62372409), the Self-funding of the State Key Laboratory of Ocean Sensing (No. OSZ-2025PY19) and the Ministry of Science and Technology of the People's Republic of China (Grant No. 2023ZD0120704 of Project No. 2023ZD0120700). The authors declare no competing interests.

References

[1] Brandstetter, J., Worrall, D., & Welling, M. (2022). Message passing neural PDE solvers. In *International Conference on Learning Representations*.

[2] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2021). Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*.

[3] Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., & Battaglia, P. W. (2021). Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*.

[4] Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., & Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), e2101784118.

[5] Lippe, P., et al. (2023). PDE-Refiner: Achieving accurate long rollouts with neural PDE solvers. In *Advances in Neural Information Processing Systems*, 36, 67398–67433.

[6] Lino, M., et al. (2023). Current and emerging deep-learning methods for the simulation of fluid dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 479(2275), 20230058.

[7] Fukagata, K., & Fukami, K. (2025). Compressing fluid flows with nonlinear machine learning: mode decomposition, latent modeling, and flow control. *Fluid Dynamics Research*, 57(4), 041401.

[8] Park, G. I., & Moin, P. (2016). Space-time characteristics of wall-pressure and wall shear-stress fluctuations in wall-modeled large eddy simulation. *Physical Review Fluids*, 1(2), 024404.

[9] Yerragolam, G. S., Verzicco, R., Lohse, D., & Stevens, R. J. A. M. (2022). How small-scale flow structures affect the heat transport in sheared thermal convection. *Journal of Fluid Mechanics*, 944, A1.

[10] Snarski, S. R., & Lueptow, R. M. (1995). Wall pressure and coherent structures in a turbulent boundary layer on a cylinder in axial flow. *Journal of Fluid Mechanics*, 286, 137–171.

[11] Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., & Niepert, M. (2022). PDEBench: An extensive benchmark for scientific machine learning. In *Advances in Neural Information Processing Systems*, 35, 1596–1611.

[12] Lee, S., & You, D. (2019). Data-driven prediction of unsteady flow over a circular cylinder using deep learning. *Journal of Fluid Mechanics*, 879, 217–254.

[13] Qiu, J., et al. (2024). Pi-fusion: Physics-informed diffusion model for learning fluid dynamics. *arXiv preprint arXiv:2406.03711*.

[14] Kim, J., Kim, J., & Lee, C. (2024). Prediction and control of two-dimensional decaying turbulence using generative adversarial networks. *Journal of Fluid Mechanics*, 981, A19.

[15] Lguensat, R., et al. (2019). Learning generalized quasi-geostrophic models using deep neural numerical models. *arXiv preprint arXiv:1911.08856*.

[16] Xing, L., et al. (2024). HelmFluid: Learning Helmholtz dynamics for interpretable fluid prediction. In *Proceedings of the 41st International Conference on Machine Learning*, PMLR 235, 54673–54697.

[17] Asaka, T., Yoshimatsu, K., & Schneider, K. (2024). Machine learning-based vorticity evolution and super-resolution of homogeneous isotropic turbulence using wavelet projection. *Physics of Fluids*, 36(2), 025120.

[18] Wiewel, S., Becher, M., & Thuerey, N. (2019). Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer Graphics Forum*, Vol. 38, No. 2, pp. 71–82.

[19] Eivazi, H., et al. (2020). Deep neural networks for nonlinear model order reduction of unsteady flows. *Physics of Fluids*, 32(10), 105104.

[20] Mohan, A., et al. (2019). Compressed convolutional LSTM: An efficient deep learning framework to model high fidelity 3D turbulence. *arXiv preprint arXiv:1903.00033*.

[21] Nakamura, T., et al. (2021). Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow. *Physics of Fluids*, 33(2), 025116.

[22] Solera-Rico, A., et al. (2024). β-variational autoencoders and transformers for reduced-order modelling of fluid flows. *Nature Communications*, 15(1), 1361.

[23] Vinuesa, R., & Brunton, S. L. (2022). Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6), 358–366.

[24] Wang, H., et al. (2025). FD-Bench: A Modular and Fair Benchmark for Data-driven Fluid Simulation. *arXiv preprint arXiv:2505.20349*.

[25] Worrall, D. E., et al. (2024). Spectral shaping for neural PDE surrogates.

[26] Rahman, M. A., Ross, Z. E., & Azizzadenesheli, K. (2023). U-NO: U-shaped neural operators. *Transactions on Machine Learning Research*.

[27] Gupta, G., Xiao, X., & Bogdan, P. (2021). Multiwavelet-based operator learning for differential equations. In *Advances in Neural Information Processing Systems*, Vol. 34, pp. 24048–24062.

[28] Fortunato, M., et al. (2022). MultiScale MeshGraphNets. *arXiv preprint arXiv:2210.00612*.

[29] Cao, Y., et al. (2023). Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network. In *Proceedings of the 40th International Conference on Machine Learning*, PMLR 202, 3541–3558.

[30] Ballé, J., Minnen, D., Singh, S., Hwang, S. J., & Johnston, N. (2018). Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*.

[31] Rudy, S. H., Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2017). Data-driven discovery of partial differential equations. *Science Advances*, 3(4), e1602614.

[32] Long, Z., Lu, Y., Ma, X., & Dong, B. (2018). PDE-Net: Learning PDEs from data. In *Proceedings of the 35th International Conference on Machine Learning*, PMLR 80, 3208–3216.

Contents of Appendices

A Derivations 14
B State Construction, Calibration, and Selection 16
C Family-Specific Instantiations 17
D Experimental Protocol 18
E Additional Results 19
F Use of Large Language Models 20

A Derivations

A.1 Primitive-state distortion (Section 3)

We expand the steps leading to Eq. (5). Let u_j be the j-th velocity component, Λ_f the fine Fourier lattice, $\Lambda_c := \Lambda_f \cap \mathcal{B}_{\exp}$ the retained lattice, and $P_{\exp}$ the spectral projector onto Λ_c. Primitive coarsening band-limits the field, stores coarse nodal values, and quantizes them componentwise: for $x_n^c := \frac{2\pi n}{N_c}$ and $j = 1, \dots, d$,

$$u_j^{\mathrm{lp}} = P_{\exp} u_j, \qquad u_j^c[n] = u_j^{\mathrm{lp}}(x_n^c), \qquad y_j[n] = Q_{b_j}(u_j^c[n]), \qquad \sum_{j=1}^{d} b_j = B. \tag{13}$$

Because u_j^{lp} is band-limited to Λ_c, coarse trigonometric interpolation reconstructs it exactly before quantization. With $\eta_j[n] := y_j[n] - u_j^c[n]$ and $\eta_j(x)$ its coarse trigonometric interpolant, the decoded field is $\tilde{u}_j = u_j^{\mathrm{lp}} + \eta_j$, so

$$\widehat{\tilde{u}_j}(k) - \hat{u}_j(k) = \hat{\eta}_j(k), \qquad k \in \Lambda_c, \tag{14}$$

because $P_{\exp}$ preserves $\hat{u}_j$ on Λ_c and removes every other mode; in particular, no folded alias term enters $\mathcal{B}_{\mathrm{hf}}$.

Noise and signal energy in the top band. A uniform b-bit quantizer on the clipped standardized range $[-a, a]$ has step $2a\,2^{-b}$ and, in the high-rate regime, per-sample error variance $c_q 2^{-2b}\sigma_j^2$ with $c_q = \frac{a^2}{3}$, where σ_j^2 is the variance of the stored component. Under the high-rate closure the errors are zero-mean and uncorrelated with the signal, across components, and across sites. With the unitary coarse discrete Fourier transform, white noise of per-sample variance v has expected energy v on each of the N_c^d coarse modes, so

$$\sum_{k \in \mathcal{B}_{\mathrm{hf}}} \mathbb{E}|\hat{\eta}_j(k)|^2 = |\mathcal{B}_{\mathrm{hf}}|\, c_q 2^{-2b_j}\sigma_j^2 = N_c^d\, \vartheta_{\mathrm{hf}}\, c_q 2^{-2b_j}\sigma_j^2, \qquad \vartheta_{\mathrm{hf}} := \frac{|\mathcal{B}_{\mathrm{hf}}|}{N_c^d}. \tag{15}$$

Counting lattice points by volume, $\vartheta_{\mathrm{hf}} \approx 1 - \gamma$ in 1D, $\frac{\pi}{4}(1-\gamma^2)$ in 2D, and $\frac{\pi}{6}(1-\gamma^3)$ in 3D. For a zero-mean component, the same transform gives $\sum_{k\in\Lambda_c} \mathbb{E}|\hat{u}_j(k)|^2 = N_c^d \sigma_j^2$. Let

$$\rho_{\mathrm{hf}}(r,\gamma) := \frac{\sum_{k\in\mathcal{B}_{\mathrm{hf}}} \mathbb{E}\|\hat{u}(k)\|_2^2}{\sum_{k\in\mathcal{B}_{\exp}} \mathbb{E}\|\hat{u}(k)\|_2^2} \approx \frac{\int_{\gamma k_c}^{k_c} E(\kappa)\, d\kappa}{\int_0^{k_c} E(\kappa)\, d\kappa}, \qquad k_c = \frac{N_f}{2r}, \tag{16}$$

be the fraction of retained-band energy in $\mathcal{B}_{\mathrm{hf}}$, where $E(\kappa)$ is the shell spectrum. For isotropic flow every component has the same band fraction and the same variance σ^2. Inserting Eqs. (14)–(16) into Eq. (2) gives

$$\mathbb{E}\big[\mathrm{hf}_{L_2}(0)^2\big] = \frac{\sum_j N_c^d \vartheta_{\mathrm{hf}} c_q 2^{-2b_j}\sigma^2}{d\,\rho_{\mathrm{hf}} N_c^d \sigma^2} = \frac{\vartheta_{\mathrm{hf}} c_q}{d\,\rho_{\mathrm{hf}}(r,\gamma)} \sum_{j=1}^{d} 2^{-2b_j}. \tag{17}$$

Because $b \mapsto 2^{-2b}$ is convex and $\sum_j b_j = B$, Jensen's inequality yields $\frac{1}{d}\sum_j 2^{-2b_j} \geq 2^{-\frac{2B}{d}}$, with equality at the equal split $b_1 = \cdots = b_d = \frac{B}{d}$. Therefore

$$\mathbb{E}\big[\mathrm{hf}_{L_2}(0)^2\big] \geq \frac{\vartheta_{\mathrm{hf}}\, a^2}{3\,\rho_{\mathrm{hf}}(r,\gamma)} 2^{-\frac{2B}{d}}, \tag{18}$$

which is Eq. (5). Anti-aliasing removes every unresolved mode, but quantization still injects approximately white error over all coarse modes, while the top retained band holds only the fraction ρ_{hf} of the retained energy. Its squared relative error therefore exceeds that of the field as a whole, $\frac{a^2}{3}2^{-2B/d}$, by the factor $\vartheta_{\mathrm{hf}}/\rho_{\mathrm{hf}}$, which grows as the spectrum steepens; the budget B sets the overall level. Figure 1a evaluates Eq. (18) in the canonical INS setting of Section 6: $N_f = 256$, the top band $(0.75\,k_c, k_c)$ of its metrics ($\gamma = 0.75$), the clip $a = 4$ of Appendix B, and b bits per velocity component, $B = 2b$. The INS spectrum is not available in closed form, so we use $E(\kappa) = 1/(1 + (\kappa/4)^\alpha)$ with $\alpha = 9.13$, the value at which Eq. (18), with $\vartheta_{\mathrm{hf}} \approx \frac{\pi}{4}(1-\gamma^2)$ and ρ_{hf} from the shell integral, equals the measured direct-decode input error, 1.394, at $B = 16$ and $r = 8$ (Section 6). Figure 1b uses the same spectrum at $r = 8$.

A.2 Field-dependent distortion (Section 4)

In 2D incompressible flow, each Fourier mode has a single divergence-free latent scalar $z(k)$, with $\hat{u}(k) = e_\perp(k)z(k)$ and $\hat{\omega}(k) = i|k|z(k)$. For any candidate channel c we write

$$\hat{x}_c(k) = M_c(k)z(k), \tag{19}$$

so that $M_u(k) = 1$ after projection onto the divergence-free direction and $M_\omega(k) = i|k|$. Applying the operator of Section 3 channel-wise gives

$$\widehat{\hat{x}}_c(k) = M_c(k)z(k) + \hat{\eta}_c(k), \qquad |k| < k_c, \tag{20}$$

where $\hat{\eta}_c(k)$ is the quantization noise of channel c on the retained band. Dividing by $M_c(k)$ yields $\hat{z}_c(k) - z(k) = \hat{\eta}_c(k)/M_c(k)$, and under the same closure, with $S_z(k) := \mathbb{E}|z(k)|^2$ and $\sigma_c^q(k; b_c) := \mathbb{E}|\hat{\eta}_c(k)|^2$,

$$\mathcal{D}_c(k) := \frac{\mathbb{E}|\hat{z}_c(k) - z(k)|^2}{S_z(k)} = \frac{\sigma_c^q(k; b_c)}{|M_c(k)|^2 S_z(k)}, \tag{21}$$

which is Eq. (6); for velocity and vorticity it gives Eq. (7). The shell curves of Figure 1b aggregate Eq. (21) over the shells $\mathcal{S}_\kappa := \{k : \kappa - \frac{1}{2} < |k| \le \kappa + \frac{1}{2}\}$:

$$L2_c(\kappa)^2 := \frac{\sum_{k\in\mathcal{S}_\kappa} \mathbb{E}|\hat{z}_c(k) - z(k)|^2}{\sum_{k\in\mathcal{S}_\kappa} S_z(k)} = \frac{\sum_{k\in\mathcal{S}_\kappa} S_z(k)\mathcal{D}_c(k)}{\sum_{k\in\mathcal{S}_\kappa} S_z(k)}. \tag{22}$$

If the shell-wise quantization spectra vary slowly, then $\mathcal{D}_\omega(k)/\mathcal{D}_u(k) \approx \sigma_\omega^q(\sigma_u^q)^{-1}|k|^{-2}$, so the vorticity curve falls relative to the velocity curve as $|k|$ approaches the cutoff, and a crossover appears near $|k|_\times \approx (\sigma_\omega^q/\sigma_u^q)^{1/2}$ whenever this variance ratio is approximately constant.

A.3 Posterior risk and the Navier–Stokes specialization (Section 5)

For each selected channel $c \in S$, the decoded observation is modeled as $\hat{y}_c(k) = B_c(k)M_c(k)z(k) + n_c(k)$, where $z(k) \in \mathbb{C}^{d-1}$ is the divergence-free latent state and $n_c(k)$ has covariance $\Sigma_c(k; b_c)$. Stacking the selected channels gives

$$\hat{y}_S(k) = A_S(k)z(k) + n_S(k), \qquad A_S(k) := \mathrm{rowstack}_{c\in S}[B_c(k)M_c(k)], \tag{23}$$

with covariance $\Sigma_S(k; \mathbf{b})$ for $n_S(k)$, and we use the prior $z(k) \sim \mathcal{CN}(0, S_z(k)I_{d-1})$. The system is linear and Gaussian, so the posterior is Gaussian with

$$\begin{aligned} \Sigma_{\mathrm{post},S}(k) &= \big(S_z(k)^{-1}I_{d-1} + A_S(k)^*\Sigma_S(k; \mathbf{b})^{-1}A_S(k)\big)^{-1}, \\ \hat{z}_S(k) &= \Sigma_{\mathrm{post},S}(k)A_S(k)^*\Sigma_S(k; \mathbf{b})^{-1}\hat{y}_S(k). \end{aligned} \tag{24}$$

The posterior mean $\hat{z}_S$ is the shared decoder. For a target $\hat{g}(k) = L(k)z(k)$, the Bayes risk of the selected state is

$$\mathcal{J}(S, \mathbf{b}) = \sum_{k\in\mathcal{B}_\star} w(k)\,\mathrm{tr}\left(L(k)\Sigma_{\mathrm{post},S}(k)L(k)^*\right), \tag{25}$$

which is Eq. (10) on a Fourier basis. For incompressible Navier–Stokes the target is velocity. In 2D, $\hat{u}(k) = e_\perp(k)z(k)$ with $e_\perp^* e_\perp = 1$; in 3D, $\hat{u}(k) = E(k)z(k)$ with $E^*E = I_2$. The velocity map is thus an isometry of the latent variable, and

$$\mathcal{J}(S, \mathbf{b}) = \sum_{k\in\mathcal{B}_\star} w(k)\ \mathrm{tr}\,\Sigma_{\mathrm{post},S}(k). \tag{26}$$

For $\mathcal{C} = \{u, \omega\}$, $M_u = e_\perp$ and $M_\omega = i|k|$ in 2D, and $M_u = E$ and $M_\omega = i|k|G$ in 3D, where $G(k)$ is the orthonormal curl basis with $G^*G = I_2$. With isotropic residuals $\Sigma_u = \sigma_u^{\mathrm{eff}} I_d$ and $\Sigma_\omega = \sigma_\omega^{\mathrm{eff}} I_d$, the channel precisions add as multiples of I_{d-1},

$$A_S^*\Sigma_S^{-1}A_S = \Big(\mathbf{1}_{\{u\in S\}}\frac{|B_u(k)|^2}{\sigma_u^{\mathrm{eff}}(k; b_u)} + \mathbf{1}_{\{\omega\in S\}}\frac{|B_\omega(k)|^2|k|^2}{\sigma_\omega^{\mathrm{eff}}(k; b_\omega)}\Big)I_{d-1}, \tag{27}$$

and substituting into Eqs. (24) and (26) gives Eq. (11).

A.4 Proof of Proposition 1 and an extended library

(i) Existence and exactness. For $c \in S$, $b_c \in \{1, \dots, \lfloor B/m_c \rfloor\}$. There are at most $2^{|\mathcal{C}|}$ subsets and finitely many allocations per subset, so $|\mathcal{D}_B| < \infty$, and $\mathcal{D}_B \neq \emptyset$ because a primitive channel c with $m_c \leq B$ is feasible at one bit per component. For every $(S, \mathbf{b}) \in \mathcal{D}_B$ and every ℓ, the matrix $M_S(\ell) := S_z(\ell)^{-1} + A_S(\ell)^* \Sigma_S(\ell; \mathbf{b})^{-1} A_S(\ell)$ is Hermitian with $M_S(\ell) \succeq S_z(\ell)^{-1} \succ 0$, hence invertible, and $\mathrm{tr}(L_g M_S^{-1} L_g^*) \in [0, \infty)$. Thus $\mathcal{J}$ is a real-valued function on a finite nonempty set and attains its minimum. Exhaustive enumeration returns a minimizer, and so does any search that discards a design only when a valid lower bound on its risk exceeds the risk of a feasible incumbent (Appendix B.3).

(ii) Primitive fallback. Let $\mathcal{D}_B^{\mathcal{P}} := \{(S, \mathbf{b}) \in \mathcal{D}_B : S \subseteq \mathcal{P}\}$. Since $\mathcal{D}_B^{\mathcal{P}} \subseteq \mathcal{D}_B$, $\min_{\mathcal{D}_B} \mathcal{J} \leq \min_{\mathcal{D}_B^{\mathcal{P}}} \mathcal{J}$. Equality holds exactly when the minimum over $\mathcal{D}_B$ is attained in $\mathcal{D}_B^{\mathcal{P}}$, that is, when no design that carries a non-primitive channel has lower risk.

(iii) Library monotonicity. The calibrated channel of a candidate does not depend on which other candidates are in the library, so $\mathcal{D}_B(\mathcal{C}) \subseteq \mathcal{D}_B(\mathcal{C}')$ with identical risks on the common designs, and $\min_{\mathcal{D}_B(\mathcal{C}')} \mathcal{J} \leq \min_{\mathcal{D}_B(\mathcal{C})} \mathcal{J}$.

Extended library. Any finite set of linear candidates fits the same template. In periodic 2D incompressible flow, consider

$$\mathcal{C}_{\mathrm{ext}} = \{u,\ \omega,\ \psi,\ \nabla u,\ \varepsilon(u),\ \nabla \cdot u,\ \Delta u,\ \Lambda u\}, \qquad \varepsilon(u) = \tfrac{1}{2}\big(\nabla u + (\nabla u)^\top\big), \qquad \Lambda = (-\Delta)^{1/2}, \tag{28}$$

where ψ is the streamfunction. Each candidate has a Fourier symbol $M_c(k)$ acting on the scalar latent $z(k)$, with $\mu_c(k) := M_c(k)^* M_c(k)$ equal to 1, $|k|^2$, $|k|^{-2}$, $|k|^2$, $|k|^2/2$, 0, $|k|^4$, and $|k|^2$, respectively. With isotropic residuals, the precisions add and

$$\mathcal{J}_{\mathrm{ext}}(S, \mathbf{b}) = \sum_{k \in \mathcal{B}_\star} w(k) \Big(S_z(k)^{-1} + \sum_{c \in S} \frac{|B_c(k)|^2 \mu_c(k)}{\sigma_c^{\mathrm{eff}}(k; b_c)} \Big)^{-1}. \tag{29}$$

The divergence of a divergence-free field has $\mu = 0$: it adds cost but no information, so a minimizer with ties broken toward lower cost never carries it. The extended problem has the same structure and the same exact solution procedure as the two-field case; only $\mathcal{D}_B$ is larger.

B State Construction, Calibration, and Selection

B.1 Coarsen–quantize–decode operator

Every explicit-state method uses the same construction:

1. compute each carried field from the primitive state;
2. project it onto the retained band of the family's basis (strict truncation, $|k| < k_c$ in the periodic case) and sample it on the n_r retained sites;
3. standardize each stored component with a mean and scale frozen from training data, clip it to ± 4 standard deviations, and quantize it uniformly with b_c bits;
4. serialize the codes; this payload is the external state passed between autoregressive steps;
5. decode by dequantizing and fusing all stored channels with the posterior mean of Eq. (24), which returns the coarse primitive field given to the predictor.

Direct decoding, which dequantizes and interpolates the primitive channels without fusion, is used only as a diagnostic (Section 6.2).

B.2 Calibration

Calibration uses training trajectories only. Passing training states through the operator, it estimates the prior $S_z(\ell)$ and, for every candidate and admissible width, the in-band transfer $B_c(\ell)$ and residual

covariance $\Sigma_c(\ell; b_c)$ of Eq. (8). Statistics are pooled over shells of similar frequency and keep the full covariance across stored components. Candidates evaluated by physical operators rather than spectral symbols (finite-difference gradients on non-periodic grids and the Burgers flux) enter through training-fitted effective responses. Widths are capped at 26 bits per component, an empirical precision plateau of our float32 pipeline, and the 4-standard-deviation clip is likewise an empirical choice; at $R = 1$ the selected designs use 26 bits per primitive component and pad the remaining bits.

B.3 Exact selection

For each candidate the admissible options are "absent" and every calibrated width that fits the budget; an option whose information matrix equals that of a cheaper option is discarded. The search seeds incumbents with all single-candidate designs and with greedy designs, orders candidates by their best single-candidate risk, and visits the tree of options depth first. A node is pruned only if one of three valid lower bounds on $\mathcal{J}$ over all of its completions exceeds the incumbent risk: a positive-semidefinite envelope of the remaining options, a multiple-choice knapsack bound on the diagonal of the information matrix, and a supporting-plane bound from the convexity of $\mathcal{J}$ in the information matrix. The returned design is therefore a global minimizer of Eq. (9). Ties are broken by lower cost and then by the lexicographically smaller allocation; no validation or test metric is consulted. One selection per family and regime (63 runs) serves all four backbones. `BestSingle` is the single derived channel with the lowest calibrated risk at the full budget, and `DerivBase` keeps the subset selected by `DerivOpt` with an equal split.

B.4 Candidate libraries

Table 3 lists the expert libraries. $\Lambda = (-L)^{1/2}$ for the family's boundary-adapted Laplacian L (periodic, Robin/Cauchy-adapted L_R, or Neumann-adapted Δ_N). The shared library of Section 6.2.3 is generated by one frozen rule that uses only the role of each primitive component (scalar or vector), the spatial dimension, and the boundary type. Each primitive scalar x contributes nine channels: x; $(-L)^s x$ for $s \in \{\frac{1}{2}, 1, \frac{3}{2}, 2\}$; and the projections of x onto the radial bands $[0, 0.25)$, $[0.25, 0.5)$, $[0.5, 0.75)$, and $[0.75, 1]$ of the retained cutoff. Each vector field also contributes its divergence and curl, and each multichannel state a DCT-II rotation across its physical components. Every generated channel is RMS-normalized on the training split. The rule never uses the identity, equation, or coefficients of the PDE or any validation or test result, and no expert field is inserted by name.

Table 3: Expert candidate libraries and library sizes in scalar channels (expert / shared).

Family	Primitive	Expert library	Channels
Advection	u	$u,\ \Lambda u,\ -\Delta u,\ u_x,\ u_{xx}$	5 / 9
Burgers	u	$u,\ \Lambda u,\ -\Delta u,\ \frac{u^2}{2},\ \partial_x \frac{u^2}{2}$	5 / 9
Diffusion–sorption	u	$u,\ \Lambda_R u,\ -L_R u,\ \partial_x u$	4 / 9
Diffusion–reaction	$q = (u, v)$	$q,\ \Lambda_N q,\ -\Delta_N q,\ d = \frac{u-v}{\sqrt{2}},\ s = \frac{u+v}{\sqrt{2}}$	8 / 20
Radial dam break	h	$h,\ \Lambda_N h,\ -\Delta_N h,\ \partial_r h$	4 / 9
Incompressible NS	u	$u,\ \omega,\ \psi,\ \Lambda u$	6 / 22
Compressible NS	$q = (\rho, v, p)$	$q,\ \chi = \nabla \cdot v,\ \omega,\ \Lambda p,\ \Lambda \rho,\ -\Delta p$	9 / 42
Total			41 / 120

C Family-Specific Instantiations

Each family instantiates Eqs. (8)–(10) with a basis $\{\phi_\ell\}$ that satisfies its boundary conditions, the eigenvalues $\lambda_\ell \geq 0$ of the corresponding negative Laplacian, $-L\phi_\ell = \lambda_\ell \phi_\ell$, and the primitive field as target.

Scalar families. Advection and Burgers use the periodic Fourier basis, diffusion–sorption a Robin/Cauchy-adapted basis, and radial dam break a Neumann-adapted basis; the latent coefficient is $z(\ell) = \langle x, \phi_\ell \rangle$ for the primitive scalar $x \in \{u, h\}$. A spectral candidate $x_c = (-L)^{t_c} x$ has symbol $M_c(\ell) = \lambda_\ell^{t_c}$, with $t_c = 0, \frac{1}{2}, 1$ for x, Λx, and $-Lx$. With scalar residual variances the score

is

$$\mathcal{J}_{\mathrm{scalar}}(S,\mathbf{b}) = \sum_{\ell\in\mathcal{I}_\star} w_\ell \Big(S_x(\ell)^{-1} + \sum_{c\in S} \frac{|B_c(\ell)|^2 \lambda_\ell^{2t_c}}{\sigma_c^2(\ell;b_c)} \Big)^{-1}. \tag{30}$$

Physical candidates (finite-difference gradients and the Burgers flux) enter through their fitted responses $B_c(\ell)M_c(\ell)$.

Diffusion–reaction. The primitive state is $q = (u,v)^\top$. The common and imbalance modes $s = \frac{u+v}{\sqrt{2}}$ and $d = \frac{u-v}{\sqrt{2}}$ are expanded in a Neumann-adapted basis, $z(\ell) = [\hat{s}(\ell), \hat{d}(\ell)]^\top$, and R denotes the fixed rotation between (u,v) and (s,d). The posterior precision is

$$\Sigma_{\mathrm{post}}^{-1}(\ell) = S_z(\ell)^{-1} + \mathbf{1}_{\{q\in S\}} R^* \frac{|B_q(\ell)|^2}{\sigma_q^2(\ell;b_q)} R + \mathbf{1}_{\{\Lambda_N q\in S\}} R^* \frac{|B_\Lambda(\ell)|^2\lambda_\ell}{\sigma_\Lambda^2(\ell;b_\Lambda)} R + \mathbf{1}_{\{\Delta_N q\in S\}} R^* \frac{|B_\Delta(\ell)|^2\lambda_\ell^2}{\sigma_\Delta^2(\ell;b_\Delta)} R + \mathbf{1}_{\{d\in S\}} e_2 \frac{|B_d(\ell)|^2}{\sigma_d^2(\ell;b_d)} e_2^\top, \tag{31}$$

with the analogous term for s, and the risk is $\sum_\ell w_\ell \operatorname{tr}(R\,\Sigma_{\mathrm{post}}(\ell) R^\top)$.

Incompressible Navier–Stokes. The periodic specialization is Eq. (11). PDEBENCH INS is wall-bounded, so its basis is boundary-adapted and the curl and streamfunction candidates are evaluated by physical operators with fitted responses; the score is the general form Eq. (10).

Compressible Navier–Stokes. The primitive state is $q = (\rho, v, p)$, with a Helmholtz-adapted latent decomposition $z = (\rho, \phi, \psi, p)$, $v = \nabla\phi + \nabla^\perp\psi$. With $L_q(\ell)$ mapping the latent variables to the primitive target and $A_S(\ell)$ stacking the selected calibrated channels,

$$\begin{aligned} \Sigma_{\mathrm{post}}(\ell) &= \Big(S_z(\ell)^{-1} + A_S(\ell)^* \Sigma_S(\ell;\mathbf{b})^{-1} A_S(\ell) \Big)^{-1}, \\ \mathcal{J}_{\mathrm{cNS}}(S,\mathbf{b}) &= \sum_{\ell\in\mathcal{I}_\star} w_\ell \operatorname{tr}\big(L_q(\ell) \Sigma_{\mathrm{post}}(\ell) L_q(\ell)^* \big). \end{aligned} \tag{32}$$

D Experimental Protocol

Data. Reference grids have 1,024 points for the three 1D families, 128^2 for diffusion–reaction and radial dam break, and 256^2 for INS; compressible Navier–Stokes is projected from 512^2 to 128^2 by a periodic Galerkin projection. INS rollouts start at frame 40 ($t = 0.20$) and all other families at frame 0; every rollout then predicts all remaining stored frames, $h = 1, \ldots, T$. Each family uses 1,000 trajectories split 640/104/256, and the test cohort is shared by all methods, budgets, and seeds. The PDEBENCH data are released under CC BY 4.0 [11]. CylinderFlow is obtained with the download script of the MeshGraphNets repository (Apache License 2.0) [3] and is evaluated at the input stage only, keeping 25% of the mesh nodes at 16 bits per node (944 B per state).

Budget ledger. With $n_r = (fN)^d$ retained sites and m_p primitive scalar components, the cap is $\lfloor 32\, m_p R n_r/8 \rfloor$ bytes. Every serialized component counts against it: field codes $n_r \sum_{c\in S} m_c b_c$ bits, latent primary and hyperprior codes, headers, and zero padding up to the cap. Field identities, widths, scales, and normalization statistics are fixed in the checkpoint and transmit no bits per state. In the INS canonical regime, $n_r = 32^2$, $B = 16$, and every explicit-state record is exactly 2,048 B: `Primitive`, `ArchMulti`, and `RolloutMulti` use u 8+8, `BestSingle` ω 16, `DerivBase` u 5+5 and ω 5 plus one padding bit per site, and `DerivOpt` u 4+4 and ω 8. `LatentAE-Hyper` is accounted by its realized bitstream with per-state rate control; on INS its unpadded payload averages 1,747.9 B (maximum 2,046 B) and is zero-padded to exactly 2,048 B. Architecture-internal tensors, such as the ConvLSTM hidden and cell states, are identical across methods and are not part of the payload; the cap equalizes external-state bits, not model size or computation.

Training. Each method and configuration is tuned over learning rate $\{3\times10^{-4}, 10^{-3}, 3\times10^{-3}\}$, weight decay $\{0, 10^{-4}\}$, and gradient clipping $\{0.5, 1\}$ (12 trials of 10,000 updates) on the validation split with seed 11. The selected trial is reused as the seed-11 run, and four further seeds (29, 47, 71, 103) are trained with the same setting. All experiments ran on 8 NVIDIA A100 GPUs.

Metrics. For the primitive target x_h at step h and its prediction $\hat{x}_h$, with the basis-adapted projectors P_{exp} and P_{fine} of Section 6.1,

$$\begin{aligned} \mathrm{expr}_{L_2}(h) &= \frac{\|P_{\mathrm{exp}}(\hat{x}_h - x_h)\|_2}{\|P_{\mathrm{exp}} x_h\|_2}, & \mathrm{hf}_{L_2}(h) &= \frac{\|P_{\mathrm{fine}}(\hat{x}_h - x_h)\|_2}{\|P_{\mathrm{fine}} x_h\|_2}, \\ Q_{\mathrm{fine}}(h) &= \frac{\|P_{\mathrm{fine}}\hat{x}_h\|_2^2 / \|P_{\mathrm{exp}}\hat{x}_h\|_2^2}{\|P_{\mathrm{fine}} x_h\|_2^2 / \|P_{\mathrm{exp}} x_h\|_2^2}, & e_{\mathrm{out}}(h) &= \frac{\|(I - P_{\mathrm{exp}})\hat{x}_h\|_2^2}{\|P_{\mathrm{exp}} x_h\|_2^2}. \end{aligned} \tag{33}$$

A step is valid when $\mathrm{expr}_{L_2}(h) < 1$, $\mathrm{hf}_{L_2}(h) < 1$, $Q_{\mathrm{fine}}(h) \in [\tau_Q^{-1}, \tau_Q]$, and $e_{\mathrm{out}}(h) < \tau_{\mathrm{out}}$, with $(\tau_Q, \tau_{\mathrm{out}}) = (1.25, 0.10)$. For each trajectory, $H = \frac{1}{T}\max\{\tau \le T : \text{steps } 1, \dots, \tau \text{ are valid}\}$, with $H = 0$ if step 1 is invalid; the input $h = 0$ is not part of the horizon, and trajectories are not filtered by their input. $\mathrm{hf}_{L_2}(0)$, nRMSE, and mean hf_{L_2} pool squared norms over the test trajectories (and steps) before taking ratios. Under strict truncation $e_{\mathrm{out}}(0)$ is close to zero, and we use it only as a protocol check.

Canonical regime. The canonical regime $(R, f) = (0.25, 1/8)$ is shared by all families and was selected once, before any test evaluation, on a pilot of 100 INS training trajectories rolled out with all four backbones, i.e., 400 equally weighted rollouts per regime. A non-finite rollout counts as final-step nRMSE 1 and $H = 0$. A regime is eligible if at least 95% of the `Primitive` rollouts are finite, the mean final-step nRMSE of `Primitive` is at most 0.15 while its mean H is at most 0.10, and `DerivBase` or `ArchMulti` reaches a mean final-step nRMSE of at most 0.15. Among the eligible regimes, we take the one with the smallest payload ratio Rf^2 relative to the full-resolution FP32 state, breaking ties by smaller R and then smaller f. The rule thus picks the smallest payload at which the primitive state already has a short detail-valid horizon while full-field errors remain moderate. No metric of `DerivOpt`, no horizon of the controls, and no test result enters the selection.

Statistics. For each seed, a pooled metric averages the configurations it covers; tables report the mean and sample standard deviation over the five seeds. Paired differences (`DerivOpt` minus comparator) are resampled 12,000 times hierarchically over families, configurations within families, and paired seeds to obtain 95% intervals.

E Additional Results

Table 4: Designs selected by `DerivOpt` for the seven PDEBENCH families in the canonical regime $(R, f) = (0.25, 1/8)$, in bits per scalar component; none uses padding. CylinderFlow does not use the grid retained fraction f; it follows a separate input-stage mesh protocol that keeps 25% of the nodes (e.g., 1,888 to 472) at 16 bits per node, i.e., 944 B per state.

Family	Bits/site	Selected design
Advection	8	$u\,3$, $\Lambda u\,5$
Burgers	8	$u\,3$, $\Lambda u\,5$
Diffusion–sorption	8	$u\,3$, $\Lambda_R u\,5$
Diffusion–reaction	16	$u\,4$, $v\,4$, $d\,8$
Radial dam break	8	$h\,3$, $\Lambda_N h\,5$
Incompressible NS	16	u 4+4, $\omega\,8$
Compressible NS	32	$\rho\,4$, v 4+4, $p\,4$, $\chi\,8$, $\omega\,8$
CylinderFlow (input only)	16 per node	u 5+5, $\omega\,6$

Selected designs. Table 4 lists the canonical designs. Over all 63 family–regime cells, 30 designs carry derived fields: all 21 at $R = 0.25$, the 9 cells of diffusion–reaction, INS, and compressible Navier–Stokes at $R = 0.5$, and none at $R = 1$. The other four families at $R = 0.5$ select the 16-bit primitive encoding, and all families at $R = 1$ select 26 bits per primitive component with padding, so `Primitive`, `DerivBase`, and `DerivOpt` coincide in those 33 cells. Bit widths do not simply scale with the budget. The designs of all 63 cells are listed in the code repository.

Table 5: INS input stage, fine-band error by region (canonical regime, same test inputs). The near-wall region comprises the outermost 13 grid layers along all four walls on the 256×256 reference grid (zero-based indices with $\min(i, j, 255 - i, 255 - j) < 13$), leaving a 230×230 interior; it contains 22.3% of the true fine-band energy.

State	Global	Near-wall	Interior	Near-wall share of error energy
`Primitive` (u 8+8)	0.999	0.777	1.054	13.5%
`BestSingle` (ω 16)	0.863	1.300	**0.689**	50.5%
`DerivBase` (u 5+5, ω 5)	0.830	0.870	0.819	24.4%
`DerivOpt` (u 4+4, ω 8)	**0.700**	**0.605**	0.725	16.6%

Table 6: Pooled horizon H over the 252 configurations under four evaluation protocols. The first three rows rescore the same predictions with different thresholds; the last rescores them with a wider fine band.

Protocol	$(\tau_Q, \tau_{\text{out}})$	Fine band	`DerivOpt`	`ArchMulti`	`LatentAE-Hyper`
Baseline	(1.25, 0.10)	$(0.75, 1]$	**0.524**	0.458	0.463
Stricter	(1.15, 0.075)	$(0.75, 1]$	**0.329**	0.287	0.290
Looser	(1.50, 0.15)	$(0.75, 1]$	**0.844**	0.742	0.750
Wider band	(1.25, 0.10)	$(2/3, 1]$	**0.832**	0.730	0.741

Wall partition and evaluation protocols. Table 5 splits the INS input error of Section 6.2 by region, and Table 6 rescores the pooled horizon under stricter, looser, and wider-band protocols; `DerivOpt` has the highest H in every row.

F Use of Large Language Models

LLMs were used only to assist with writing and presentation: checking technical terminology, correcting grammar, translating from the authors' native language, and improving sentence structure while preserving the scientific content. They were not used for research ideation, experimental design, data analysis, or any content that constitutes an intellectual contribution. All scientific insights, methodological decisions, and conclusions are the authors' own.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state the scoped claims (budgeted setting, the stated comparison set), which Sections 3–6 support.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 discusses the candidate library, calibration and storage-operator dependence, and the scope of the claims.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Assumptions are stated with Eq. (5) and Proposition 1; Appendix A gives the derivations and proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 6.1 and Appendices B and D give the operator, budgets, libraries, selection, training, and metrics; the code is released.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is at https://github.com/VinceWangVW/DerivOpt with run instructions; all data are public (Appendix D).

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Section 6.1 and Appendix D give splits, tuning grid, training length, seeds, and metrics.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report five-seed means with seed standard deviations and paired 95% hierarchical-bootstrap intervals (Section 6, Appendix D).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix D states the hardware (8 NVIDIA A100 GPUs) and the training budget of every run.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses public simulation data and involves no human subjects or sensitive data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is foundational methodology for PDE surrogates with no direct path to negative societal applications.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We release no model or dataset with a high risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: PDEBench (CC BY 4.0) and MeshGraphNets CylinderFlow (Apache 2.0) are cited, with licenses in Appendix D.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released code is documented in its repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work involves no crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work involves no human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used only for writing and editing (Appendix F), not as part of the method.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.