# METACOLLOC: OPTIMIZATION-FREE PDE SOLVING VIA META-LEARNED BASIS FUNCTIONS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

We introduce MetaColloc, which freezes a meta-learned basis dictionary and recovers each PDE solution by a single linear least-squares solve on a collocation system: no per-equation training, no solution data. Under the unregularised solve its accuracies on smooth two-dimensional benchmarks span $10^{-9}$ to $10^{-5}$. We also report the two quantities that bound what those numbers can mean; both are cheap to compute before any solve is attempted. The first is *capacity*. The dictionary is meta-trained on Gaussian Random Field samples, so it can represent that prior no better than $H$ functions can. The prior needs $594$ Karhunen–Loève modes to capture $90\%$ of its energy against $H{=}512$, so the trained dictionary reproduces $16\%$ of a fresh prior draw and fails an in-distribution manufactured problem under every backend we tried — worse than predicting zero. The second is *numerical rank*. The row-equilibrated collocation system has condition number $1.04 \times 10^{17}$ and retains rank $309$ of $512$ at $10^{-12}\sigma_{\max}$, so the solve is rank-deficient beyond double precision and its answer is decided by the cutoff the library chooses: the same checkpoint solves Poisson to $4.2 \times 10^{-5}$ under cuSOLVER and to $4.9 \times 10^{-1}$ under LAPACK. Choosing a Tikhonov level by generalized cross-validation stays within an order of magnitude of the oracle cutoff, matching it on the equation that needs regularisation, while making the result a property of the data rather than of the backend. On a band-matched prior, where the dictionary *is* representable, the comparison does not resolve: across three task banks the meta-trained dictionary scores $1.08 \times 10^{-1}$, $1.34 \times 10^{-1}$ and $1.21 \times 10^{-1}$, while a random basis from the same prior's spectrum ranges from $6.8 \times 10^{-2}$ to $1.35$.

## 1 INTRODUCTION

Physics-informed neural networks [32] solve a PDE by training a network on its residual, which costs thousands of gradient steps per equation and is sensitive to the balance between loss terms [38, 22, 18]. A natural response is to move the optimisation offline: learn a dictionary once, freeze it, and reduce each new equation to a linear problem. We introduce *MetaColloc*, which meta-trains a dual-branch network on Gaussian Random Field samples and then solves a new PDE with one least-squares collocation solve for linear operators and a Newton iteration for nonlinear ones. Extreme Learning Machines [16, 8] and random-feature collocation [30] make the same reduction with a random dictionary instead of a trained one.

This paper reports the method together with the two quantities that bound what it can achieve. Both are cheap to compute before any solve is attempted, and both bound what a reported accuracy can mean.

The first is capacity. A dictionary meta-trained on a prior inherits that prior's intrinsic dimension: it can represent the prior no better than $H$ functions can, and it can represent a PDE solution no better than the prior covers it. We compute the prior's Karhunen–Loève spectrum by the method of snapshots, at two resolutions. On $2048$ snapshots over a $129^2$ grid, the best $H{=}512$ linear subspace captures $39.3\%$ of the estimable variance. The mode counts below use $768$ snapshots on a $65 \times 65$ grid; that estimator has rank at most $767$ while the mixture needs $594$ modes, so the counts depend on the resolution and have to be read with that in mind. Even a single smooth length scale, $\ell{=}0.05$, needs $602$ modes for $90\%$ of the energy, and the full mixture of smooth and oscillatory modes needs $594$. The trained dictionary does worse than either ceiling: a pure value fit of a fresh prior draw



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

reaches relative error $0.918$, and a manufactured Poisson problem whose solution *is* a prior draw is solved with relative error above $1$ under both backends we tried — worse than the constant-zero predictor. The same dictionary solves our smooth Poisson benchmark to $4.2 \times 10^{-5}$. The prior is broadband and the benchmarks are not.

The second is numerical rank. The collocation system the solver inverts — derivative rows for the PDE, value rows for the boundary, each row equilibrated — has condition number $1.0 \times 10^{17}$ for the trained $H{=}512$ dictionary. 203 of its 512 singular values fall below $10^{-12}\sigma_{\max}$, giving an effective rank of 309. Unlike a sampling artefact, this does not improve with more data: going from 2000 to 16000 interior points leaves the numerical rank at $332, 332, 332, 333$ (at the looser tolerance $10^{-14}\sigma_{\max}$) while the condition number grows by more than an order of magnitude. We quote the growth rather than the values, because $\kappa(A)$ is itself backend-dependent: the same checkpoint and the same collocation points give $1.04 \times 10^{17}$ under one LAPACK, $1.13 \times 10^{17}$ under another, and $5.08 \times 10^{16}$ under cuSOLVER. What the unresolved directions correspond to is left open here: the dictionary's own value matrix is numerically degenerate at the same order as the collocation system, so these measurements cannot separate near-nullness of the operator from degeneracy of the trial space. What the unresolved directions do is change the solution while barely moving the residual.

The consequence is worse than lost digits: the solve is rank-deficient beyond double precision, so its answer is decided by which singular values the linear-algebra library treats as nonzero. Running the same checkpoint under LAPACK on CPU and under cuSOLVER on a GPU gives Poisson RMSE $4.9 \times 10^{-1}$ and $4.2 \times 10^{-5}$ respectively. Reproducing the method across hardware therefore requires making the regularisation explicit.

We then show what to do about it. Replacing the unregularised solve with a Tikhonov-regularised one, with the level chosen by generalized cross-validation, makes the answer a function of the data rather than of the backend, and recovers the accuracy an oracle-tuned truncation would achieve: on the high-frequency Poisson problem it reaches $4.798 \times 10^{-1}$ where the best fixed cutoff over six orders of magnitude gives $4.812 \times 10^{-1}$, improving on the unregularised solve by a factor of $1.7$. We report the accuracy-versus-rank curve for every benchmark equation and dictionary we trained, which is the quantity that should have been reported instead of a single number.

The same analysis also says which parts of our benchmark suite measure the method and which do not. The frequency sweep mixes two regimes: the Fourier branch of the dictionary tops out at $k{=}128$, i.e. $64$ cycles per unit, while the sweep's largest points need $128$ and $256$ cycles per unit. Nothing the dictionary does changes those points, and no value of the operator weight alters them. On the smooth equations, by contrast, a frozen random dictionary whose frequencies are drawn from the prior spectrum solves to machine precision: the suite ranks dictionaries by frequency placement, not by how they were obtained.

Our contributions are:

- A capacity diagnostic: the training prior's Karhunen–Loève mode count against $H$, computable before training, which predicts how much of a fresh prior draw a dictionary of that size can represent. We measure it for the prior we train on and show that $H{=}512$ is under-parameterised.

- A numerical-rank diagnostic: the singular-value profile of the operator-restricted collocation system, which predicts whether a solve is reproducible at all. We show the two limitations are independent, and that neither is fixed by the other.

- A regularised solve with a GCV-selected level. It removes the backend dependence and comes within an order of magnitude of the oracle-tuned cutoff, matching it on the equation that needs regularisation. The accuracy-versus-rank curves that accompany it make the comparison well posed.

- A re-examination of our benchmark suite against these two diagnostics, separating measurements of the method from measurements of the representable band.

# 2 Related Work

## 2.1 Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) [32] encode PDE constraints directly into a loss function and solve the equation by gradient descent. Subsequent work balances the loss terms during training [38, 18, 22], analyses gradients with Neural Tangent Kernels [39], enforces causality [40], designs better optimisation schemes [7, 37, 45, 4], or changes the architecture, for example to Kolmogorov–Arnold Networks [23]. For high-frequency problems some methods use random Fourier features [39] or dynamic meshes [19]. These methods remain sensitive to hyper-parameters.

All of these optimise per instance, so a new network is trained for every equation and test-time cost is high. MetaColloc moves that cost offline: one heavy meta-training stage, then one linear solve per equation and no per-instance optimisation. The size of that trade-off is measured in Section 4.

## 2.2 Gaussian Process and Generative PDE Solvers

Gaussian Process (GP) methods solve PDEs probabilistically [5]. GP-HM [9], for instance, models the power spectrum of the solution in the frequency domain and beats standard PINNs on high-frequency PDEs. Diffusion models have also been applied to PDEs, for field reconstruction, control, and dynamics prediction [17, 35, 48, 3, 43].

Both families differ from MetaColloc in where the cost sits. GP-HM is still a per-instance solver: it optimises kernel parameters for hundreds of seconds per problem and needs data on a strict Cartesian grid. Diffusion-based solvers pay a slow iterative sampling loop [34]. MetaColloc carries neither constraint, and its test-time cost does not depend on the equation or on the grid.

## 2.3 Operator Learning and Meta-Learning for PDEs

Operator learning accelerates PDE simulation by learning mappings between function spaces. Fourier Neural Operators [20] and DeepONets [25] are the standard instances. Later work scales them with Transformers [33, 1, 15, 41], state-space models [47] and geometry-aware deformations [21, 42], or combines them with multi-physics pre-training [28, 31] and Newton iterations for non-linear operators [13].

These neural operators generalise well across physical parameters, but they need large datasets of exact PDE solutions, and generating those is itself expensive. MetaColloc trains without solution data at all: we sample random functions from Gaussian Random Fields. Nor do we learn a specific operator mapping. We learn a single dictionary of basis functions, intended to transfer across equations. Section 4.1 measures how far it transfers, and locates the limit in the prior rather than in the training.

## 2.4 Random Features and Kernel Methods

Random Fourier features [30] approximate kernel functions. Extreme Learning Machines (ELM) [16, 8] solve PDEs with fixed random hidden layers, turning the PDE into a linear system; recent variants pair random networks with time-parallel solvers [11] or high-dimensional graph models [10].

ELM and random feature methods solve simple PDEs quickly, but their accuracy in collocation systems varies widely with the random initialisation of the basis shape parameters. For complex PDEs, this randomization strategy may require extensive hyperparameter tuning across GRFs with different correlation lengths, typically by selecting the shape parameter that minimises the average MSE over these GRFs [46]. An equation-specific formula for the shape parameter gives more stable representations [26], though its accuracy still depends on the initialisation and the seed. MetaColloc replaces that selection with a dictionary meta-learned over multi-scale GRFs, keeping the closed-form speed of a linear solver. Whether that training buys anything over a random basis drawn from the same prior is the question Section 4.3 answers, and the answer is that it does not.

# 3 THE METACOLLOC FRAMEWORK

## 3.1 DECOUPLING BASIS DISCOVERY FROM PDE SOLVING

Standard solvers such as Physics-Informed Neural Networks (PINNs) formulate PDE solving as an instance-specific optimisation problem. The approach works across many physical domains, but it requires training a network by gradient descent for each new equation, which is expensive and sensitive to the optimisation dynamics.

We decouple PDE solving into an offline basis-discovery stage and an online solving stage. We assume the solution $u(x)$ is well approximated by a linear combination of basis functions,

$$u(x) \approx \Phi(x)w.$$

Here, $\Phi(x) \in \mathbb{R}^H$ represents a basis dictionary generated by a neural network with $H$ output units, and $w \in \mathbb{R}^H$ is the vector of unknown coefficients.

MetaColloc freezes the network parameters at test time, so the network acts as a fixed feature extractor and the only unknown is the coefficient vector $w$. No backpropagation is needed: $w$ follows from one linear solve.

We use *optimization-free* in one specific sense: no gradient-based training is performed per equation. It does not mean that no numerical work is done. Solving for $w$ is a least-squares problem, and the nonlinear solver iterates Newton–Raphson steps, each of which is such a problem. What is absent is the per-equation optimisation of the *basis*, which is where PINNs spend their time.

## 3.2 IMPLEMENTATION DETAILS OF THE DUAL-BRANCH BASIS DICTIONARY

The core of MetaColloc is the neural network $\Phi_\theta : \mathbb{R}^d \to \mathbb{R}^H$, which parameterizes the basis dictionary. Here, $d$ is the spatial dimension (e.g., $d = 2$ for 2D Poisson) and $H$ is the total number of basis functions. To overcome the spectral bias [29] of standard MLPs, and to capture smooth and oscillatory structure without retraining, we use a decoupled dual-branch architecture. Let the input coordinate be $\mathbf{x} \in \mathbb{R}^d$. The network output $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^H$ is composed as follows:

$$\boldsymbol{\phi}(\mathbf{x}) = \text{Concat}\left[\boldsymbol{\phi}_{\text{low}}(\mathbf{x}), \boldsymbol{\phi}_{\text{high}}(\mathbf{x})\right]$$

where $\boldsymbol{\phi}_{\text{low}} \in \mathbb{R}^{H/2}$ and $\boldsymbol{\phi}_{\text{high}} \in \mathbb{R}^{H/2}$ are outputs from the low- and high-frequency branches, respectively. We use an equal split, $H/2$ per branch, and every width reported here is even, so the split is exact rather than rounded. The ratio is a design choice we have not varied.

**Low-Frequency Branch (Smooth Approximation):** This branch processes the raw input $\mathbf{x}$ to represent the macro-scale, smooth components of the physical field. It consists of a standard Multi-Layer Perceptron (MLP). We use the SwiGLU activation [6] for its gradient flow. A single SwiGLU layer $k$ with input $\mathbf{h}_{k-1}$ is formulated as:

$$\mathbf{h}_k = \text{SwiGLU}(\mathbf{h}_{k-1}) = (\text{SiLU}(\mathbf{h}_{k-1}W_1 + \mathbf{b}_1)) \odot (\mathbf{h}_{k-1}W_2 + \mathbf{b}_2)$$

where $\odot$ denotes element-wise multiplication, and $\text{SiLU}(z) = z/(1 + \exp(-z))$. The low-frequency branch features 2 hidden SwiGLU layers, with widths of $H$ and $H/2$, respectively.

**High-Frequency Branch (Spectral Enhancement):** To model sharp gradients and oscillating solutions (e.g., the HighFreq Poisson case in Section 4.3), this branch uses multi-scale Fourier features. The input $\mathbf{x}$ is first projected into a set of sinusoidal encodings:

$$\boldsymbol{\gamma}(\mathbf{x}) = \left[\sin(\pi\mathbf{k}_j^\top \mathbf{x}), \cos(\pi\mathbf{k}_j^\top \mathbf{x})\right]_{j=1}^{F}$$

Here, $F$ is the number of Fourier features, and the frequency vectors $\mathbf{k}_j \in \mathbb{R}^d$ are sampled from a multi-scale distribution to cover a wide spectrum. In our implementation, we explicitly set $\mathbf{k}_j$ as a

set of fixed, axis-aligned frequency scales: $\{1.0, 2.0, 4.0, 8.0, 16.0, 32.0, 64.0, 128.0\}$, which gives the branch a priori power in those bands. The features are sinusoids along the axes, so an oscillation oriented obliquely to them has to be assembled from several features rather than matched by one. We do not measure the cost of that assembly here; it is a bias of the construction, and the measured band limit in Section 4.3 is stated for axis-aligned targets. The encoded features $\boldsymbol{\gamma}(\mathbf{x})$ are then passed through 2 SwiGLU hidden layers (with widths of $H$ and $H/2$, respectively) to form $\boldsymbol{\phi}_{\text{high}}(\mathbf{x})$.

### 3.3 Meta-Training on Gaussian Random Fields

The basis is meta-trained, so the choice of meta-training distribution is what shapes it. We take that distribution to be a Gaussian Random Field (GRF), which connects to Gaussian process theory [44]: the GRF's reproducing kernel Hilbert space has good approximation properties [2, 12] and coincides with its Cameron–Martin space [27], which describes the regularity and dominant directions of variation of the field.

Sample paths of a GRF do not themselves lie in its RKHS, but their smoothness is governed by Sobolev embeddings [14]. That is what makes a GRF a natural prior for PDE solutions: it covers many function types and gives a principled way to draw them.

Length scales span two orders of magnitude and the mixture combines smooth radial kernels with oscillatory periodic ones, which forces the network to cover both smooth shapes and sharp waves. The basis is then intended to transfer to unseen equations.

Both branches use SwiGLU activations [6] and their outputs are concatenated into $\Phi(x)$. The first takes raw coordinates and covers smooth, low-frequency structure; the second takes multi-scale Fourier features [36] and covers the oscillatory part. The network is trained to minimise least-squares error over many GRF tasks, as Algorithm 1 sets out.

**Algorithm 1** Meta-Training the Neural Basis Dictionary

**Require:** Meta-training epochs $E$, tasks per epoch $T$, neural network $\Phi_\theta$

**for** epoch $= 1$ to $E$ **do**
  **for** task $= 1$ to $T$ **do**
    Sample inputs $X$ and targets $Y$ from multi-scale GRF distribution
    Compute basis matrix $\Phi_\theta(X)$
    Solve for coefficients: $w = \text{lstsq}(\Phi_\theta(X), Y)$
    Predict outputs: $\hat{Y} = \Phi_\theta(X)w$
    Compute MSE loss between $\hat{Y}$ and $Y$
    Update $\theta$ with AdamW [24]
  **end for**
**end for**
**return** Frozen network $\Phi_{\text{frozen}}$

### 3.4 Optimization-Free Collocation for Linear PDEs

After meta-training, the frozen network is used inside a collocation framework. For the Poisson equation $-\Delta u = f$ with boundary data $u = g$, we sample $N$ interior and $N_b$ boundary points and pass them through the network. Derivatives come from forward-mode automatic differentiation, which returns $\Phi_{xx}$ and $\Phi_{yy}$ exactly and at low memory cost, so the interior block is the operator applied to the basis:

$$-(\Phi_{xx} + \Phi_{yy})\, w = f, \qquad \Phi_{bd}\, w = g.$$

Stacking the two blocks gives an overdetermined system, solved for $w$ by least squares; evaluating at a new point is then $\Phi_{\text{frozen}}(x)w$. No gradient descent is used at test time.

Table 1: Capacity of the meta-training prior, measured by the method of snapshots ($M$=768 tasks on a $65 \times 65$ grid). "Modes for 90%" is the number of Karhunen–Loève modes needed to capture 90% of the energy. Even the smoothest prior in the table is under-parameterised by $H$=512.

| **length scale** $\ell$ | **max** $\lvert\omega\rvert$ | **modes for** 90% | **energy at** $H$=512 |
|---|---|---|---|
| 0.05 (single scale, smooth) | 20 | 602 | 0.825 |
| 0.10 (single scale) | 10 | 587 | 0.842 |
| 0.20 (single scale) | 5 | 539 | 0.883 |
| 0.40 (over-smooth) | 2.5 | 408 | 0.944 |
| full mixture | 300 | **594** | 0.393 |

### 3.5 Newton–Raphson Iteration for Non-Linear PDEs

The non-linear benchmarks, Sine–Gordon and Korteweg–de Vries, need an iterative treatment. We apply Newton–Raphson in function space, which turns the non-linear problem into a short sequence of linear ones. Each step evaluates the linearised operator with the exact neural derivatives, assembles a fresh least-squares system, and updates $w$ by $\Delta w$; every subproblem is convex, so the iteration converges quadratically near the solution. The two benchmarks differ in where they settle: Sine–Gordon plateaus by about eight iterations and KdV by about sixteen, and neither moves after that (Appendix B.1). We therefore fix a budget rather than test convergence per problem. Algorithm 2 gives the full solver.

**Algorithm 2** Universal PDE Solving (Linear and Non-Linear)

**Require:** Frozen network $\Phi_{\text{frozen}}$, PDE operator $\mathcal{L}$, boundary operator $\mathcal{B}$, iterations $K$
1: Sample interior points $X_{int}$ and boundary points $X_{bd}$
2: Compute basis matrices $\Phi_{int}$ and $\Phi_{bd}$
3: Compute derivatives (e.g., $\Phi_x$, $\Phi_{xx}$, ...) with forward-mode AD
4: Initialize $w = 0$
5: Set $K = 1$ for linear PDEs
6: **for** iteration $= 1$ to $K$ **do**
7: Compute predictions $u = \Phi w$ and needed derivatives
8: Build interior matrix $A_{eq}$ from linearized $\mathcal{L}$
9: Build boundary matrix $A_{bd}$ from linearized $\mathcal{B}$
10: Form residuals for interior and boundary
11: Stack $A = [A_{eq}; A_{bd}]$ and $R = [f; g]$
12: Solve $\Delta w = \text{lstsq}(A, -R)$
13: Update $w = w + \Delta w$
14: **end for**
15: **return** Solution $u(x) = \Phi_{\text{frozen}}(x)w$

## 4 Two Limits, and a Fix

### 4.1 Capacity: what the prior lets a dictionary of size H represent

Let the dictionary be meta-trained on samples from a zero-mean prior $\mathcal{D}$ with covariance operator $\mathcal{C}$.

**The prior's ceiling.** For squared-$L^2$ reconstruction, the best any linear subspace of dimension $H$ can do is the span of the leading eigenfunctions of $\mathcal{C}$; the residual energy is the tail sum $\sum_{k>H} \lambda_k$, and no parameterisation, however it is trained, exceeds it. The first question to ask of a dictionary of size $H$ is therefore not how it was trained but what $H$ buys for this prior. We estimate $\mathcal{C}$ by the method of snapshots — $M$ tasks on a $65 \times 65$ grid, the $M \times M$ Gram matrix of the snapshots, its eigenpairs — and report in Table 1 the number of modes needed for 90% of the energy, varying the prior from the full mixture down to a single, deliberately over-smooth length scale.

Two consequences. First, at $H$=512 the prior is under-parameterised no matter how the dictionary is trained. Second, the number is a property of the prior, not of the method: it can be computed before a single training run and used to choose $H$.

**What the trained dictionary achieves.** Fitting fresh prior draws in the trained $H{=}512$ dictionary by least squares — a pure value fit, no differential operator involved — leaves a relative value error of $0.918$ and a relative Laplacian error of $1.031$. Both are close to 1, the value a constant-zero predictor would achieve, so the dictionary captures roughly $16\%$ of the energy of a draw from its own training distribution and essentially none of the operator image.

The gap to the benchmarks is then easy to state: the same dictionary solves the original Poisson equation to $4.23 \times 10^{-5}$ and the variable-coefficient equation to $2.76 \times 10^{-9}$, while a manufactured Poisson problem whose exact solution *is* a prior draw fails under every backend we tried (see Section 4.2) — the solve is worse than returning zero.

The reading is that the dictionary is not failing; it is representing the part of the prior that $H$ functions can hold, and the benchmarks happen to live there. This also explains a comparison that otherwise looks damning: built optimally for the same prior, the empirical Karhunen–Loève basis captures $3.3\%$ of the energy of $\sin(2\pi x)\sin(2\pi y)$, because the optimal subspace of a broadband prior is nearly orthogonal to a smooth target. Optimising the training objective to optimality does not produce a good solver basis; it produces the best basis for a distribution the benchmarks do not come from. The two limits are the two factors of a standard quasi-optimal bound (Appendix F): capacity controls the approximation factor, which depends only on the span, while the stability factor depends on how the operator behaves on that span.

### 4.2 Numerical rank: whether the solve is reproducible at all

The second limit is not about the span but about the map. Write the collocation system as $A = [\, S_{\text{eq}} L[\Phi];\ S_{\text{bd}} \Phi_{\text{bd}} \,]$, where $L$ is the differential operator, $\Phi$ the dictionary, and the $S$ are the per-row equilibrations the solver applies.

For the trained $H{=}512$ dictionary, $\kappa(A) = 1.04 \times 10^{17}$; at a relative threshold of $10^{-12}\sigma_{\max}$, 203 of the 512 singular values are discarded and the retained rank is 309. Since $1/\varepsilon_{\text{f64}} \approx 4.5 \times 10^{15}$, the system is rank-deficient *beyond* double precision. The retained rank is of course a function of the threshold; we use $10^{-12}\sigma_{\max}$ throughout and state it wherever a rank is reported. Table 2 reports the resulting accuracy-versus-rank curves; the full set is in Appendix D.

**Ruling out an operator–function mismatch.** It is tempting to read the unresolved directions as functions the operator annihilates. The control does not support that. The dictionary's own value matrix $\Phi$, at the same collocation points, has $\kappa(\Phi) = 8.9 \times 10^{16}$: the same order as $\kappa(A)$. The trial space is numerically degenerate before any operator acts on it, so these measurements cannot separate operator near-nullness from degeneracy of the dictionary itself, and we do not claim the operator-invisible mechanism. What they do establish is that the collocation system inherits a rank deficiency of about 200 directions, which is what makes the solve backend-dependent.

Quadrupling and octupling the collocation set does not raise the rank: at $n = 2000, 4000, 8000, 16000$ interior points the rank at $\tau{=}10^{-14}$ reads $332, 332, 332, 333$. The missing directions are therefore no sampling artefact and are not resolved by more data, whatever their origin; the practical consequence is the regularised solve of the next subsection.

Because the cutoff that separates resolved from unresolved directions is chosen inside the linear-algebra library, the answer inherits that choice. The same checkpoint on the same collocation points gives Poisson RMSE $4.905 \times 10^{-1}$ under one LAPACK on a CPU and $4.234 \times 10^{-5}$ under cuSOLVER on a GPU. The diagnostic quantities move as well: $\kappa(A)$ reads $1.04 \times 10^{17}$, $1.13 \times 10^{17}$ and $5.08 \times 10^{16}$ under the three implementations we ran, and $\kappa(\Phi)$ ranges over $6.31$ to $9.02 \times 10^{16}$. The quantity one would use to diagnose the problem is itself a property of the library. We ruled out an implementation error: on CPU our `solve_pde` and an independent re-implementation agree to the bit, and sweeping the cutoff leaves the CPU answer at $\approx 0.49$ throughout.

### 4.3 The controlled solve: accuracy against retained rank

All numbers below come from the code accompanying this submission, with an unmodified dictionary, trained at the hyperparameters we use throughout on one RTX 4090 (3281 s); the only change is that the gradient is accumulated in chunks of eight tasks, which matches one backward pass on the mean loss to $10^{-16}$ relative and fits a 48 GB card. The harness reproduces the results (Appendix G).

Table 2: Accuracy as a function of the retained rank, for the value-only dictionary ($\lambda$=0) and for dictionaries trained with an operator-consistency term at several weights. Rank is reported at $\tau$=$10^{-14}$. Shown are excerpts; all six equations and all seven dictionaries are in Appendix D.

| | | **HighFreq** | | **SineGordon** | |
|---|---|---|---|---|---|
| **dictionary** | **rank** | $\tau$=$10^{-6}$ | $\tau$=$10^{-14}$ | $\tau$=$10^{-6}$ | $\tau$=$10^{-14}$ |
| value ($\lambda$=0) | 332 | 6.34e-1 | 1.00e+0 | 4.63e-1 | **1.90e-6** |
| lap ($\lambda$=$3 \times 10^{-3}$) | 326 | 4.96e-1 | 8.19e-1 | 9.10e-1 | 1.25e-5 |
| lap ($\lambda$=$10^{-1}$) | 327 | 5.95e-1 | 9.11e-1 | 3.24e-1 | 2.56e-6 |
| gradlap ($\lambda$=$10^{-1}$) | 327 | 6.17e-1 | 8.54e-1 | 1.17e+0 | 2.37e-5 |

We replace the unregularised inner solve with a truncated-SVD pseudo-inverse at an explicit relative tolerance $\tau$, and report the retained rank together with the error. Across our trained dictionaries the effective rank is $327 \pm 3$ out of $512$ at every tolerance we tried, and — the point we want to make — it is essentially independent of how the dictionary was trained.

Three things follow. First, accuracy is a smooth, monotone function of the retained rank, which is why it is the quantity that should be reported; a single number hides six orders of magnitude of variation. Second, the optimal rank differs by equation and in opposite directions — HighFreq is best at the *lowest* rank we tried and degrades by $1.6\times$ as rank grows, while SineGordon improves by five orders of magnitude over the same range — so no single library default can be right for all of them. Third, adding an operator-consistency term to meta-training does not change the effective rank at all (326–328 against the value-only dictionary's 332), and under this controlled solve its effect on error is small and uneven: it is better on HighFreq at every cutoff and on SineGordon and KdV at the tightest ones, and no better on the other three. The term is not inert — as we show in Appendix E.3 it moves interpolation accuracy substantially — but it acts on directions the operator can see, and the rank argument says those are not the ones that are missing.

Finally, does making the prior representable help? We meta-trained on a band-limited prior ($\ell \in [0.08, 0.25]$, $|\omega| \leq 40$) whose mode count falls inside $H$=512, and scored every dictionary under one protocol on three 200-task banks. The meta-trained dictionary is stable across banks, at $1.08 \times 10^{-1}$, $1.34 \times 10^{-1}$ and $1.21 \times 10^{-1}$; a random basis from that prior's spectrum is not, ranging from $6.8 \times 10^{-2}$ to $1.35$ and doing worse than predicting zero on two of the three. One bank favours the random basis and two favour meta-training, so no single bank is evidence in either direction. Matching the band does make the prior representable — the broadband-prior dictionary sits at $0.91$ — but the comparison it enables does not settle which dictionary is better.

### 4.4 A REGULARISED SOLVE, WITH THE LEVEL CHOSEN FROM THE DATA

If the difficulty is that the problem is ill-posed, the answer is to regularise it explicitly rather than to accept whatever the library does. Truncated SVD at a fixed tolerance is not sufficient, because the optimal truncation is problem-dependent in opposite directions: accuracy improves monotonically with rank for Poisson, VarCoeff and Sine–Gordon, and degrades for the high-frequency problem.

We therefore use Tikhonov regularisation and choose the level $\lambda$ by generalized cross-validation, which needs no knowledge of the solution:

$$G(\lambda) = \frac{\|Ax_\lambda - b\|^2}{\big(m - \operatorname{tr} H_\lambda\big)^2}, \qquad H_\lambda = A(A^\top A + \lambda I)^{-1} A^\top,$$

and the whole curve costs one SVD of $A$ because both the residual and the trace have closed forms in the singular values.

On the equation that needs regularisation this comes within $0.3\%$ of the oracle-tuned cutoff and edges marginally past it ($4.7979 \times 10^{-1}$ against $4.8119 \times 10^{-1}$), and improves on the unregularised solve by a factor of $1.7$. It also removes the backend dependence, because $\lambda$ is now a function of the data rather than of a library's internal threshold.

The frequency sweep and the branch spectra are analysed in Appendix E.3: separated into the representable band ($f \leq 32$) and beyond it, the sweep shows a band limitation rather than a degradation of operator fidelity, and no value of the operator weight moves the out-of-band points.

Table 3: Three ways of choosing the regularisation, none of which sees the solution. "Oracle cutoff" is the best fixed relative tolerance over $\tau \in \{10^{-4}, \ldots, 10^{-14}\}$, i.e. the best we could have done with hindsight. GCV-Tikhonov matches it on the high-frequency problem and is within an order of magnitude elsewhere, while the unregularised solve is backend-dependent.

| equation | GCV-Tikhonov | oracle cutoff | unregularised |
|---|---|---|---|
| Poisson | 1.40e-3 | 9.12e-4 | 4.23e-5 |
| VarCoeff | 8.08e-7 | 8.35e-8 | 2.76e-9 |
| HighFreq | **4.798e-1** | 4.812e-1 | 8.98e-1 |

## 5 Limitations and What This Implies

**The diagnosis is not a repair.** We add an operator-consistency term to meta-training — matching the Laplacian of the fitted field against the analytic Laplacian of the sampled prior function, available in closed form for a random-feature prior. It reduces interpolation error at 16 and 32 cycles per unit by $53\%$ and $61\%$, monotonically in the weight and saturating near $0.3$, and halves the Laplacian error at 32 cycles. But under the regularised solve its effect on the six benchmark equations is small and uneven (Section 4.3), and it makes the prior-reconstruction objective about a fifth worse than the value-only dictionary ($6.6 \times 10^{-1}$ against $5.45 \times 10^{-1}$). The term sharpens directions the operator can see; it does not create the ones that are missing.

**What we do not claim.** Not a state-of-the-art solver: the accuracy figures move by orders of magnitude with the linear-algebra backend, and the dictionary cannot represent functions outside the prior it was trained on, which needs $594$ modes against $H{=}512$. Nor do we claim meta-training beats a matched random basis: as Section 4.3 reports, that comparison does not settle either way across task banks. What survives is a precise account of why, and that account identifies what carries the accuracy: whether a dictionary places basis mass at the target's frequency, not how the dictionary was obtained. That is the property that separates the rows of Table 9, and it is why a random basis drawn from the prior's own spectrum reaches machine precision there while the meta-trained dictionary does not.

**Practical guidance.** Before training, compare the prior's $90\%$ mode count against the intended $H$; if it exceeds $H$, no parameterisation will work. After training, compute the numerical rank of the operator-restricted system; if it is well below $H$, the solve will not be reproducible across backends and the regularised solve is not optional. Both diagnostics are cheap — one eigendecomposition of a snapshot Gram matrix, one SVD per equation — and neither requires solving the PDE. Regularisation cannot create accuracy the trial space does not contain: on Helmholtz no cutoff and no $\lambda$ moved the error off $5.0 \times 10^{-1}$.

**What we did not test.** Several comparisons this kind of method invites are not run here, and we name them rather than leave them implied. We do not set a classical finite-difference or spectral solver against this suite: we expect one to be both faster and more accurate on all six equations, which are smooth manufactured problems on the unit square that a second-order scheme resolves in milliseconds. That is the capacity diagnostic read from the other side, and it is why test-time cost is not presented as a contribution. We also do not test targets outside the representable band (boundary layers, sharp fronts, data-defined forcing), add a learned-operator baseline trained on the same GRF samples, or try hand-crafted bases. The geometric experiments of Appendix B.3 cover a linear equation only. None of this changes the two diagnostics, which are properties of the prior and of the collocation system rather than of the test set.

**Appendix results carry the same caveat.** The three-dimensional, L-shape and annulus experiments, the precision study and the timing tables were produced by the same unregularised solve that Section 4.2 shows to be backend-dependent, and none has been re-run under a controlling solve; their accuracies should be read as provisional. We flag this rather than leave two registers in one document.

## REFERENCES


[1] Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 25152–25194. Curran Associates, Inc., 2024. doi: 10.52202/079017-0793. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2cd36d327f33d47b372d4711edd08de0-Paper-Conference.pdf.

[2] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[3] Jan-Hendrik Bastek, WaiChing Sun, and Dennis Kochmann. Physics-informed diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tpYeermigp.

[4] Kanishk Bhatia, Felix Koehler, and Nils Thuerey. PRDP: Progressively refined differentiable physics. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9Fh0z1JmPU.

[5] Yuntian Chen et al. Solving partial differential equations with point source based on physics-informed neural networks. *Journal of Computational Physics*, 2021.

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

[7] Felix Dangel, Johannes Müller, and Marius Zeinhofer. Kronecker-factored approximate curvature for physics-informed neural networks. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 34582–34636. Curran Associates, Inc., 2024. doi: 10.52202/079017-1091. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/3d27d607586984908900eaa8ce19c96c-Paper-Conference.pdf.

[8] Suchuan Dong and Zongyi Li. Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 387:114129, 2021. doi: 10.1016/j.cma.2021.114129.

[9] Shikai Fang, Madison Cooley, Da Long, Shibo Li, Mike Kirby, and Shandian Zhe. Solving high frequency and multi-scale PDEs with gaussian processes. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=q4AEBLHuA6.

[10] Mingquan Feng, Yixin Huang, Weixin Liao, Yuhong Liu, Yizhou Liu, and Junchi Yan. SINGER: Stochastic network graph evolving operator for high dimensional PDEs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=wVADj7yKee.

[11] Guglielmo Gattiglio, Lyudmila Grigoryeva, and Massimiliano Tamborrino. Randnet-parareal: a time-parallel pde solver using random neural networks. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang

(eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 94993–95025. Curran Associates, Inc., 2024. doi: 10.52202/079017-3011. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/acb94e709f02895fd98b5867f0b184f3-Paper-Conference.pdf.

[12] Minh Ha Quang. *Reproducing kernel Hilbert spaces in learning theory*. Brown University, 2006.

[13] Wenrui Hao, Xinliang Liu, and Yahong Yang. Newton informed neural operator for solving nonlinear partial differential equations. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 120832–120860. Curran Associates, Inc., 2024. doi: 10.52202/079017-3839. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/dae8afc6b990aa0b3b5efaa096fbd7fa-Paper-Conference.pdf.

[14] Iain Henderson. Sobolev regularity of gaussian random fields. *Journal of Functional Analysis*, 286(3):110241, 2024.

[15] Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 72525–72624. Curran Associates, Inc., 2024. doi: 10.52202/079017-2311. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/84e1b1ec17bb11c57234e96433022a9a-Paper-Conference.pdf.

[16] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. In *Neurocomputing*, 2006.

[17] Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. Diffusionpde: Generative pde-solving under partial observation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 130291–130323. Curran Associates, Inc., 2024. doi: 10.52202/079017-4140. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/eb3878c1dcbfff9ee95d5d033e5f5942-Paper-Conference.pdf.

[18] Youngsik Hwang and Dong-Young Lim. Dual cone gradient descent for training physics-informed neural networks. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 98563–98595. Curran Associates, Inc., 2024. doi: 10.52202/079017-3128. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b2b781badeeb49896c4b324c466ec442-Paper-Conference.pdf.

[19] Namgyu Kang, Jaemin Oh, Youngjoon Hong, and Eunbyung Park. PIG: Physics-informed gaussians as adaptive parametric mesh representations. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=y5B0ca4mjt.

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

[21] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023. URL http://jmlr.org/papers/v24/23-0064.html.

[22] Qiang Liu, Mengyu Chu, and Nils Thuerey. ConFIG: Towards conflict-free training of physics informed neural networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=APojAzJQiq.

[23] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–arnold networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ozo7qJ5vZi.

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[25] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5.

[26] Tianzheng Lu, Lili Ju, and Liyong Zhu. A multiple transferable neural network method with domain decomposition for elliptic interface problems. *Journal of Computational Physics*, 530: 113902, 2025. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2025.113902. URL https://www.sciencedirect.com/science/article/pii/S0021999125001858.

[27] Milan Lukić and Jay Beder. Stochastic processes with sample paths in reproducing kernel hilbert spaces. *Transactions of the American Mathematical Society*, 353(10):3945–3969, 2001.

[28] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for spatiotemporal surrogate models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 119301–119335. Curran Associates, Inc., 2024. doi: 10.52202/079017-3791. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d7cb9db5ade2db7814fbd01ee59f4c7b-Paper-Conference.pdf.

[29] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Eberhardt, Bengio Yoshua, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

[30] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, 2007.

[31] Ashiqur Rahman, Robert Joseph George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A. Yeh, Jean Kossaifi, Kamyar Azizzadenesheli, and Anima Anandkumar. Pretraining codomain attention neural operators for solving multiphysics pdes. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 104035–104064. Curran Associates, Inc., 2024. doi: 10.52202/079017-3306. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/bc75fa9843a7905bbed9d83895a88f7f-Paper-Conference.pdf.

[32] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[33] Louis Serrano, Thomas X Wang, Etienne Le Naour, Jean-Noël Vittaut, and Patrick Gallinari. Aroma: Preserving spatial structure for latent pde modeling with local neural fields. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 13489–13521. Curran Associates, Inc., 2024. doi: 10.52202/079017-0431. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/185a120a3f709187e68bd092e6098851-Paper-Conference.pdf.

[34] Youssef Shehata, Benjamin Holzschuh, and Nils Thuerey. Improved sampling of diffusion models in fluid dynamics with tweedie's formula. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=0FbzC7B9xI.

[35] Aliaksandra Shysheya, Cristiana Diaconu, Federico Bergamin, Paris Perdikaris, José Miguel Hernández-Lobato, Richard E. Turner, and Emile Mathieu. On conditional diffusion models for pde simulations. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 23246–23300. Curran Associates, Inc., 2024. doi: 10.52202/079017-0732. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2974844555dc383ea16c5f35833c7a57-Paper-Conference.pdf.

[36] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7537–7547. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/55053683268957697aa39fba6f231c68-Paper.pdf.

[37] Nils Wandel, Stefan Schulz, and Reinhard Klein. Metamizer: A versatile neural optimizer for fast and accurate physics simulations. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=60TXv9Xif5.

[38] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. In *SIAM Journal on Scientific Computing*, 2021.

[39] Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. In *Computer Methods in Applied Mechanics and Engineering*, 2021.

[40] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. In *arXiv preprint arXiv:2203.07404*, 2022.

[41] Sifan Wang, Jacob H Seidman, Shyam Sankaran, Hanwen Wang, George J. Pappas, and Paris Perdikaris. CVit: Continuous vision transformer for operator learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cRnCcuLvyr.

[42] Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse pde problems. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 33085–33107. Curran Associates, Inc., 2024. doi: 10.52202/079017-1042. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/39f6d5c2e310a5a629dcfc4d517aa0d1-Paper-Conference.pdf.

[43] Long Wei, Haodong Feng, Yuchen Yang, Ruiqi Feng, Peiyan Hu, Xiang Zheng, Tao Zhang, Dixia Fan, and Tailin Wu. CL-diffphycon: Closed-loop diffusion control of complex physical systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=PiHGrTTnvb.

[44] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[45] Haixu Wu, Huakun Luo, Yuezhou Ma, Jianmin Wang, and Mingsheng Long. Ropinn: Region optimized physics-informed neural networks. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 110494–110532. Curran Associates, Inc., 2024. doi: 10.52202/079017-3508. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/c745bfa5b50544882938ff4f89ff26ac-Paper-Conference.pdf.

[46] Zezhong Zhang, Feng Bao, Lili Ju, and Guannan Zhang. Transferable neural networks for partial differential equations. *Journal of Scientific Computing*, 99(2):31, 2024. doi: 10.1007/s10915-024-02463-y.

[47] Jianwei Zheng, Wei Li, Ni Xu, Junwei Zhu, Xiaoxu Lin, and Xiaoqin Zhang. Alias-free mamba neural operator. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 52962–52995. Curran Associates, Inc., 2024. doi: 10.52202/079017-1678. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5ee553ec47c31e46a1209bb858b30aa5-Paper-Conference.pdf.

[48] Anthony Zhou, Zijie Li, Michael Schneier, John R Buchanan Jr, and Amir Barati Farimani. Text2PDE: Latent diffusion models for accessible physics simulation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Nb3a8aUGfj.

## AI USE STATEMENT

We used generative AI tools for drafting and editing prose, for literature search, and for auditing and debugging the analysis and verification scripts released with this paper. The method under study, the numerical experiments and the reported measurements are our own: no AI system generated experimental data or produced any quantity reported as a measurement, no theoretical result or proof was AI-generated, and the method itself was not implemented by AI.

We have reviewed all AI-assisted work. Statements and numbers that AI-assisted editing touched were re-checked against the primary sources, the reference list was checked for coverage and for self-reference, and every script AI touched was re-run with its output compared against the artifacts it summarises. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REPRODUCIBILITY STATEMENT

The measurements this paper's argument rests on were re-run for it, and their artefacts ship with the supplement: the numerical ranks and condition numbers, the capacity numbers, the cross-backend comparison, and the regularised solves. The accuracy tables also report values from the meta-training and benchmark runs that produced the dictionaries. Those runs were not repeated here, and we mark where a value is of that kind rather than a fresh measurement; Table 10 puts the harness's own output beside them at the same configuration so the two can be compared directly.

**Code and artefacts.** All experiments use our MetaColloc implementation unchanged; the meta-training loop is reproduced in `op_train.py`, which reuses the same model class as `code/metacolloc_2d.py` so the architecture is byte-identical. The rank-controlled and GCV-regularised solves are in `rank_solve.py`; the same-prior Karhunen–Loève baseline, the in-distribution manufactured benchmark, the basis diagnostics and the corrected PINN baseline are in `kl_pod.py`, `grf_benchmark.py`, `diagnostics.py` and `pinn.py`. `parity_check.py` establishes that our re-implementation of the collocation solve agrees with `metacolloc_2d.solve_pde` to the bit on CPU, which is what licenses the comparisons.

**Reproducing the main claims.** The capacity measurements need only one eigendecomposition of a snapshot Gram matrix and no PDE solve; we report the snapshot count, grid and mode counts so they can be checked directly. The numerical-rank measurements need one SVD per equation. The cross-backend comparison is a direct consequence of the rank-deficient system and is reproduced by running the same checkpoint on CPU and on GPU.

**What is not reproducible, and why we say so.** The single-figure accuracies are the one thing we cannot reproduce across hardware, and that is a finding rather than a caveat: the unregularised solve is rank-deficient beyond double precision, so its answer is determined by the linear-algebra backend. We report the accuracy-versus-rank curve instead of a single number for exactly this reason. The

benchmark tables in the appendix are cuSOLVER figures from the unregularised solve and carry the same caveat; where the regularised solve is quoted as a single figure we state the backend.

**Randomness.** All Gaussian Random Field draws are made on the CPU from an explicitly seeded generator, so the task stream is identical on CPU and GPU; the sampler in `metacolloc_2d.sample_multiscale_grf` called `uniform_` without a generator and so consumed the global RNG; it now takes a seeded generator, which is what makes a task stream reproducible. Every meta-training run reported here uses seed 42.

# A Experimental Setup and Hyperparameters

We provide the exact settings used to train and evaluate MetaColloc. We designed these parameters to ensure full reproducibility.

All matrix solves use double precision. The least-squares inversions are why: in single precision they lose the small singular values the rank analysis depends on. The meta-training and benchmark runs used one NVIDIA A800-SXM4-80GB; the controlled re-runs of Section 4.3 used one RTX 4090, under Python 3.12.12 and PyTorch 2.6.0+cu124.

During the offline meta-training phase, we sample functions from a multi-scale Gaussian Random Field distribution. We generate tasks by mixing three probability modes. We use a pure Radial Basis Function mode 40% of the time. We use a high-frequency mode 40% of the time. We mix both modes for the remaining 20% of the tasks. For smooth functions, we draw length scales randomly between 0.005 and 0.05. For high-frequency functions, we sample center frequencies uniformly between 10.0 and 300.0. We add a noise bandwidth between 1.0 and 15.0 to create complex signal aliases.

We train the neural network for 1000 epochs. Each epoch contains 128 random GRF tasks. For each task, we sample 4000 training points and 1500 testing points uniformly across the domain. We use the AdamW [24] optimizer. We set the initial learning rate to $10^{-3}$ and apply a weight decay of $10^{-4}$. We use a cosine annealing scheduler to reduce the learning rate gradually.

During the online test phase, we evaluate the solver on the specific PDEs. We sample 2000 points randomly inside the spatial domain. We sample 300 points along the domain boundaries. For linear PDEs, we solve the system in one single matrix inversion step. For non-linear PDEs, we apply the Newton-Raphson method with a fixed budget of 64 iterations in 2D and 8 in 3D (Appendix B.2). The budget is a safety margin rather than a convergence criterion: the iteration sweep in Appendix B.1 shows both equations flat from $K \approx 16$ onward, so $K = 64$ carries a fourfold margin over what the sweep measures as sufficient.

We test MetaColloc on six two-dimensional equations. We define the spatial variables as $x$ and $y$. For time-dependent problems, $y$ acts as the time variable $t$.

The suite has six two-dimensional equations, each with a closed-form solution and a right-hand side $f$ computed analytically. Poisson, $-\Delta u = f$, takes the smooth $u = \sin(2\pi x)\sin(2\pi y) + e^{-x-y}$. Helmholtz, $-\Delta u - k^2 u = f$ at $k = 64\pi$, is the extreme-frequency case, with $u = \sin(k_{xy}x)\cos(k_{xy}y) + e^{-x-y}$ and $k_{xy} = k/\sqrt{2}$. The variable-coefficient equation, $-\nabla \cdot (a\nabla u) = f$ with $a = 2 + \sin(\pi x)\cos(\pi y)$, uses $u = \sin(\pi x)\sin(\pi y) + e^{-x-y}$. High-frequency Poisson reuses the first operator with the oscillatory $u = \sin(8\pi x)\sin(8\pi y) + e^{-xy}$. Sine–Gordon, $u_{yy} - u_{xx} + \sin u = f$, takes $u = \sin(\pi x)\cos(\pi y)$, and so does Korteweg–de Vries, $u_y + 6uu_x + u_{xxx} = f$, whose third-order term is what makes the Newton solve nontrivial.

For all six equations we use the unit square $[0, 1]^2$ and Dirichlet boundary conditions with values taken from the analytical solution. All six are therefore *boundary-value* problems with manufactured solutions. This matters for the two evolution equations: Sine–Gordon and KdV both carry a variable that would be time in the physical problem, and posing them on a square with data on all four edges is not the initial-value problem those names usually denote. What the benchmark measures is the dictionary's ability to represent and invert the differential operator, not the physical evolution, and the same convention is used in every table and figure below.

# B Supplementary Experiments

**A note on the intervals.** Every table in this appendix that reports a 95% confidence interval does so over *five* runs, at seeds 42–46: that is the loop in the two-dimensional driver, and the three-dimensional results and the iteration sweep say so in their captions. The intervals are percentile bootstraps over those five values. We state it here because the two-dimensional tables do not say it themselves.

### B.1 Convergence Behavior of Nonlinear PDE Solvers

The nonlinear solver is swept over the iteration budget $K \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$ at $H = 512$, for Sine–Gordon and KdV, using the same Newton–Raphson iteration as the main text and averaging five seeds per setting.

Figure 1 shows the results. The Sine-Gordon equation converges fast. The error drops in the first few steps. The KdV equation needs more steps and settles three orders of magnitude higher. Both curves are flat from $K = 16$ to $K = 512$, and the variation across that range stays inside the seed band, so the solver behaves predictably once it has converged. That is what makes the choice $K = 64$ in the main text safe: it is four times the point at which the sweep stops improving.

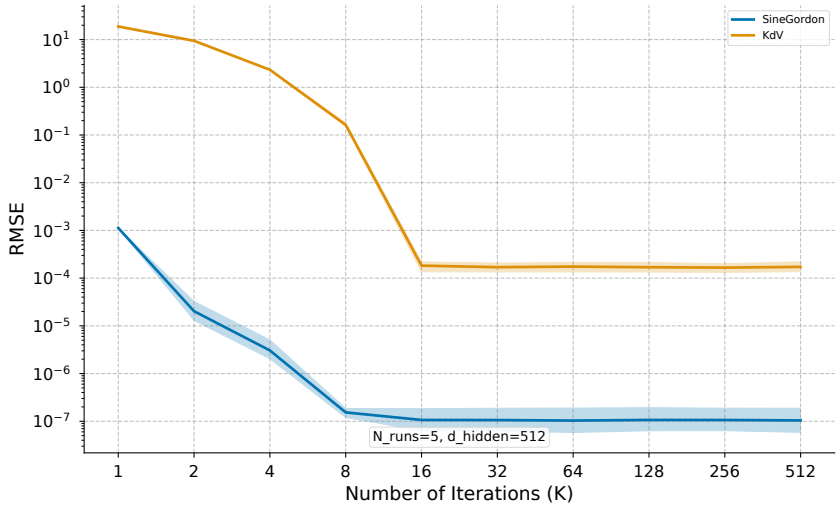


Figure 1: RMSE as a function of the number of iterations $K$ for the Sine-Gordon and KdV equations. We run five seeds for each setting. Both curves fall steeply and then plateau by $K \approx 16$; the KdV equation needs more steps and settles three orders of magnitude higher.

The nonlinear solver behaves predictably: the error falls steeply up to $K \approx 16$ and is flat thereafter, with the seed band containing the remaining variation. That is what makes a fixed budget a reasonable choice here rather than a per-problem convergence test.

### B.2 Extension to 3D PDEs

We test whether MetaColloc extends to higher dimensions. Standard grid-based solvers suffer from the curse of dimensionality. Their memory costs explode when moving from 2D to 3D. MetaColloc is completely mesh-free. We only change the input dimension of the neural network from two to three. The core algorithm remains identical.

We train a new MetaColloc model with an input dimension of three. The coordinates are $x$, $y$, and $z$. We use the exact same meta-training strategy and hyperparameters described in Appendix A. We only adjust the domain to be a three-dimensional unit cube $[0, 1]^3$. During the test phase, we sample 8000 interior points and 3600 boundary points. For non-linear 3D PDEs, we restrict the Newton-Raphson solver to just 8 iterations.

Three three-dimensional problems are used, all with manufactured solutions as in Appendix B. Poisson, $u_{xx} + u_{yy} + u_{zz} + f = 0$, is the standard spatial case, with $u = \sin(\pi x)\sin(\pi y)\sin(\pi z)$. Burgers and Allen–Cahn are posed with $z$ as the evolution variable, which makes each a two-dimensional spatial problem plus one temporal: $u_z + uu_x + uu_y - \nu(u_{xx} + u_{yy}) = f$ at $\nu = 0.01$ with $u = \sin(\pi x)\sin(\pi y)e^{-z}$, and $u_z - \nu(u_{xx} + u_{yy}) - u(1 - u^2) = f$ at $\nu = 0.001$ with $u = \sin(\pi x)\sin(\pi y)\cos(\pi z)$.

Table 4 shows our results. The errors lie between $10^{-6}$ and $10^{-7}$, so the dictionary extends to three dimensions without architectural change. As in two dimensions, these figures come from an

unregularised solve and carry the caveat of Section 4.2. They are reported from the runs that produced the three-dimensional dictionary rather than re-measured here, and `code/metacolloc_3d.py` is the driver that produced them.

Table 4: RMSE results for 3D equations. We report the mean and 95% confidence interval across five seeds.

| Param | Poisson3D | Burgers3D | AllenCahn3D |
|---|---|---|---|
| H=1024 | 3.66e-6 (1.96e-6) | **1.77e-5 (5.85e-6)** | **2.24e-6 (1.84e-6)** |
| H=2048 | **4.04e-7 (6.73e-8)** | 1.95e-5 (5.79e-6) | 3.09e-6 (1.13e-6) |

### B.3 Generalization to Complex Geometries and Mixed Boundary Conditions

In the main text, our evaluations focus on standard regular domains (e.g., square and cubic boxes) with Dirichlet boundary conditions. A natural question is whether the meta-learned dictionary $\Phi_{\text{frozen}}$ overfits to Cartesian symmetries or to a particular boundary type. We therefore apply the pre-trained 2D model (H=512, seed 42, with no retraining or fine-tuning) to two domains with mixed boundary conditions.

#### B.3.1 Problem Setup and Boundary Formulations

Because MetaColloc is point-wise and mesh-free, adapting to a new geometry requires only that collocation points be sampled inside it. To handle derivative-based boundary conditions, we evaluate the directional derivatives of our neural basis using forward-mode automatic differentiation.

For a boundary point $x_{bd}$ with outward normal vector $n$, the Neumann boundary condition requires $\nabla u \cdot n = h$. We assemble the corresponding row in the boundary matrix $A_{bd}$ as:

$$A_{bd} = \Phi_x n_x + \Phi_y n_y$$

Similarly, for a Robin boundary condition $u + \alpha \nabla u \cdot n = g_r$, the matrix row is assembled as:

$$A_{bd} = \Phi + \alpha(\Phi_x n_x + \Phi_y n_y)$$

We solve the Poisson equation $-\Delta u = f$ with the exact solution $u(x, y) = \sin(2\pi x)\sin(2\pi y) + \exp(-x - y)$ on the following two irregular domains. Both tests use a linear equation on a single domain each, so what follows is evidence about geometry and boundary handling and not about the nonlinear solver on complex domains, which we have not run.

**L-Shape Domain with Mixed Dirichlet–Neumann Conditions:** The domain is $[-1, 1]^2 \setminus [0, 1] \times [-1, 0]$, with a re-entrant corner at the origin. We apply Dirichlet conditions on the outer left, top, and bottom-left edges. We apply Neumann conditions (matching the exact normal derivatives) on the right-top edge and the two inner edges forming the concave corner.

**Annulus Domain with Mixed Dirichlet-Robin BCs:** The domain is a ring defined by $r \in [0.5, 1.0]$. We apply a Dirichlet condition on the inner boundary ($r = 0.5$) and a Robin condition (with $\alpha = 1$) on the outer boundary ($r = 1.0$).

#### B.3.2 Results and Discussion

We sampled 3000 interior collocation points and 600 boundary points for both geometries. The linear system was solved in a single step using double precision (FP64). Both figures below are reported from the runs that produced the dictionary, rather than re-measured here; `code/boundary_testing.py` is the script that produced them.

**L-Shape (Dirichlet + Neumann):** The model achieved an RMSE of 2.00e-03.

**Annulus (Dirichlet + Robin):** The model achieved an RMSE of 3.48e-04.

These results confirm that the basis is independent of the spatial grid: adapting to a new geometry needs only new collocation points, and derivative-based boundary conditions come from the same forward-mode automatic differentiation.

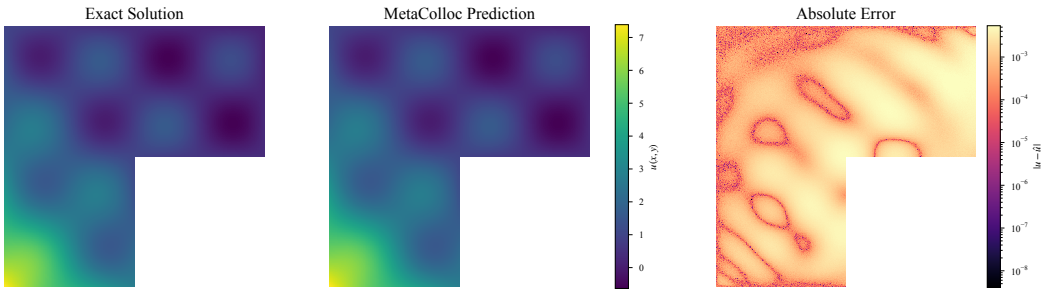


Figure 2: Exact solution, MetaColloc prediction, and absolute error heatmap for the Poisson equation on L-shape domain with mixed Dirichlet and Neumann boundaries. The error is bounded across the whole domain with no localized blow-up at the re-entrant corner, which is nevertheless among the highest-error regions of the domain.

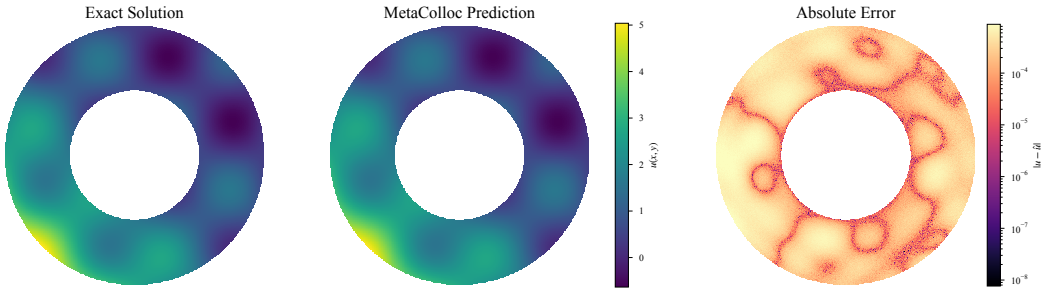


Figure 3: Exact solution, MetaColloc prediction, and absolute error heatmap for the Poisson equation on Annulus domain with mixed Dirichlet and Robin boundaries. The error is well-bounded across the entire domain.

### B.4 Time Usage in Section 4

We first report the wall-clock time MetaColloc takes to meta-train, in Table 5.

Table 5: MetaColloc training time (mean with 95% CI) for different $H$.

| Param | Training Time (s) |
|---|---|
| H=128 | 403.46 (3.00) |
| H=256 | 468.18 (1.32) |
| H=512 | 701.22 (0.59) |
| H=1024 | 1414.86 (1.07) |

Table 6 reports the wall time each method spends solving one equation. For MetaColloc that is a single least-squares problem on the frozen basis. That solve is not the whole cost: meta-training happens once, and Table 5 reports it separately.

**Why the timing comparison has two rows and not five.** Three further baselines — PINN L-BFGS, ConFIG and PINN DCGD — were run on this suite and are not tabulated, because their per-equation timings are not usable. In those three scripts the loop that prints the timings iterates over a variable named `pde_name` but indexes `solve_times[pde]`, where `pde` is left over from the experiment loop and holds its final value; every column therefore reports the last equation's timing six times. MetaColloc and GP-HM index with their own loop variable and are unaffected. Rather than re-run the three on different hardware, which would not restore the comparison, we omit them — and test-time cost is not presented as a contribution in any case.

Table 6: Solving time (mean with 95% CI) for the two methods whose scripts report per-equation timings correctly. The unit of data is seconds. Lower is better.

| Method | Param | Poisson | Helmholtz | VarCoeff | HighFreq Poisson | SineGordon | KdV |
|---|---|---|---|---|---|---|---|
| MetaColloc | H=128 | 1.590 (0.449) | 1.306 (0.027) | 1.287 (0.002) | 1.287 (0.003) | 1.461 (0.003) | 15.981 (0.046) |
| | H=256 | 1.298 (0.008) | 1.292 (0.005) | 1.292 (0.005) | 1.291 (0.004) | 1.628 (0.010) | 16.153 (0.063) |
| | H=512 | 1.430 (0.097) | 1.304 (0.003) | 1.304 (0.004) | 1.304 (0.003) | 1.862 (0.003) | 16.459 (0.030) |
| | H=1024 | 1.543 (0.261) | 1.296 (0.013) | 1.295 (0.011) | 1.295 (0.011) | 2.331 (0.020) | 16.746 (0.143) |
| GP-HM | Grid=$100^2$ | 4643.387 (3.303) | 4676.184 (1.470) | 5053.207 (2.532) | 4869.284 (215.580) | 432.252 (0.453) | 1085.075 (0.155) |

## C Gaussian Random Field Sampling

We describe the sampling process used in our meta-training stage. We use Gaussian Random Fields (GRFs) to generate a wide range of target functions. These functions cover smooth shapes and high-frequency waves. This diversity is important. It forces the network to learn a flexible basis.

Each task draws one of three modes at random:

- **RBF mode**: smooth functions with a random length scale.
- **High-frequency mode**: oscillatory functions with a random center frequency and bandwidth.
- **Mixed mode**: a combination of the two.

We draw $M = n_{\text{train}} + n_{\text{test}}$ input points $X \in [0, 1]^2$ from a uniform distribution. We then build a random feature map

$$\phi(x) = \sqrt{\frac{2}{D}} \cos(\omega^\top x + b),$$

where $D$ is the number of random features. The vector $b$ contains random phases drawn from a uniform distribution on $[0, 2\pi]$. The vector $\omega$ depends on the sampling mode.

**RBF mode.** We draw a log-uniform length scale $\ell \in [\ell_{\text{lo}}, \ell_{\text{hi}}]$. We then sample

$$\omega \sim \mathcal{N}(0, \ell^{-2} I_2).$$

This produces smooth functions with controlled variation.

**High-frequency mode.** We draw a center frequency $\mu$ from a uniform range $[\mu_{\text{lo}}, \mu_{\text{hi}}]$. We draw a bandwidth $\sigma$ from $[\sigma_{\text{lo}}, \sigma_{\text{hi}}]$. We sample half of the frequencies from

$$\omega \sim \mathcal{N}(\mu, \sigma^2 I_2),$$

and the other half from its symmetric counterpart $-\omega$. This creates oscillatory functions with balanced positive and negative frequencies.

**Mixed mode.** We split the features into two halves. One half uses the RBF mode. The other half uses the high-frequency mode.

**Output construction.** We draw random weights $w \sim \mathcal{N}(0, I_D)$. We compute the target values

$$y = \phi(X)w.$$

We split the first $n_{\text{train}}$ samples for training and the rest for testing.

This sampling process gives a broad and controlled distribution of functions. It covers smooth gradients, sharp transitions, and high-frequency waves. This diversity is what the basis is trained to cover; Section 4.1 measures how much of it $H$ basis functions can hold.

# D ACCURACY VERSUS RETAINED RANK, ALL EQUATIONS

Table 7 gives the full curve for the value-only dictionary ($\lambda$=0) at $H$=512; Table 2 in the main text gives the operator-aware arms for the two equations that respond to rank.

Table 7: RMSE against the retained rank, value-only dictionary, $H$=512, collocation on 2000 interior and 300 boundary points. "rank" is the numerical rank at $\tau$=$10^{-14}$. Retained rank is threshold-dependent: the same matrices retain 309 of 512 at $\tau$=$10^{-12}$. Ranks are quoted with their threshold throughout.

| **equation** | **rank** | $\tau$=$10^{-6}$ | $\tau$=$10^{-8}$ | $\tau$=$10^{-14}$ |
|---|---|---|---|---|
| Poisson | 332 | 4.95e-1 | 1.54e-1 | 9.12e-4 |
| Helmholtz | 328 | 5.00e-1 | 5.00e-1 | 5.01e-1 |
| VarCoeff | 332 | 5.22e-2 | 2.13e-3 | 8.21e-8 |
| HighFreq | 332 | 6.34e-1 | 6.99e-1 | 1.00e+0 |
| SineGordon | 332 | 4.63e-1 | 6.19e-2 | 1.90e-6 |
| KdV | 317 | 4.52e-1 | 4.69e-1 | 1.43e-1 |

Two features of this table carry the argument of Section 4.2. The retained rank is essentially the same (317–332 of 512) for every equation, while the best tolerance differs: Poisson, VarCoeff and Sine–Gordon gain three to five orders of magnitude as rank increases, and HighFreq loses half an order of magnitude, because extra rank admits directions the operator does not constrain. Helmholtz is flat to two decimals at every rank, which is the signature of a target outside the representable band rather than of a regularisation problem.

# E CORRECTED PINN BASELINE: NORMALISATION AND TRAINING DETAILS

Our Helmholtz loss is $\|-\Delta u - k^2 u - f\|^2$ with $k = 64\pi$, so $k^2 \approx 4.05 \times 10^4$ and the objective sits near $10^8$; Adam at $10^{-3}$ does not move it. In the corrected baseline we divide the interior residual by $k^2$ before squaring, leaving the boundary term unchanged, and train 20 000 Adam steps at $10^{-3}$ with cosine annealing followed by L-BFGS until the strong-Wolfe line search terminates. Widths and collocation counts match our benchmark table (128, 2000 interior, 300 boundary). The smooth and oscillatory amplitudes in Table 8 come from a two-column least-squares fit of the prediction against the analytic components $\exp(-x-y)$ and $\sin(k_{xy}x)\cos(k_{xy}y)$; the same procedure applied to Poisson against $\exp(-x-y)$ and $\sin(2\pi x)\sin(2\pi y)$ gives 0.999 and 1.000 respectively, which is the control reported in the main text.

## E.1 THE REPORTED PINN BASELINE WAS MEASURING SPECTRAL BIAS

Our baseline table reports exactly $5.02 \times 10^{-1}$ on Helmholtz at every width, which invites the suspicion that the baseline failed to converge. It did, for an informative reason, which we can now demonstrate rather than guess at.

Our loss formed the Helmholtz residual as $-\Delta u - k^2 u - f$ with $k = 64\pi$, so $k^2 \approx 4.05 \times 10^4$ and the objective sat near $10^8$; Adam at $10^{-3}$ cannot move it. Rescaling the residual so that its dominant term is order one, and training far longer, does *not* rescue the baseline — at width 128 it still lands at $5.02 \times 10^{-1}$.

What explains the value is the projection of the prediction onto the analytic smooth and oscillatory parts of the exact solution. Since the two parts are known in closed form, the prediction can be fitted against them jointly:

On the control problem the corrected baseline recovers both components to three decimals. On Helmholtz it recovers the smooth component almost exactly and the oscillatory component not at all. The reported $5.02 \times 10^{-1}$ is what a smooth-only prediction scores against a target of unit RMS, which is also why several methods in the benchmark table land on the same value. Adding Fourier features to the network, which is the standard remedy for spectral bias, moves the oscillatory amplitude from 0.000 to 0.013.

Table 8: Corrected PINN at width 128, trained to convergence (20 000 Adam steps plus L-BFGS). "amp" is the least-squares coefficient of the prediction on the analytic smooth and oscillatory components of the exact solution; both are 1 for a perfect fit. Poisson is the control.

| problem | RMSE | amp smooth | amp oscillatory |
|---|---|---|---|
| Poisson (control) | **4.10e-3** | 0.999 | **1.000** |
| Helmholtz, plain MLP | 5.02e-1 | 1.003 | **0.000** |
| Helmholtz, +Fourier features | 6.39e-1 | 0.997 | **0.013** |

Table 9: Frozen dictionaries at H=512, all solved by the same collocation routine. The random baselines have their shape parameter and ridge chosen on the prior only, by reconstruction error over a held-out GRF task bank; no test equation is consulted. Best in each column in bold.

| dictionary | Poisson | VarCoeff | SineGordon | HighFreq |
|---|---|---|---|---|
| MetaColloc | 4.23e-5 | 2.76e-9 | 1.91e-7 | 8.98e-1 |
| ELM, tanh | 6.32e-6 | 2.43e-8 | 1.31e-6 | 6.03e-1 |
| TransNet | 2.90e-4 | 8.77e-6 | 2.46e-4 | 1.36e+0 |
| RFF, isotropic | 4.78e-1 | 8.45e-1 | 3.91e-1 | 4.13e+0 |
| non-learning | 1.87e-2 | 3.96e-5 | 1.35e-3 | 1.27e+0 |
| **RFF, prior spectrum** | **2.62e-15** | **6.35e-15** | **4.88e-14** | **1.08e-2** |

We report this because it converts a bare number into a measurement: it identifies the failure as a spectral one, not an optimisation failure that more tuning would have fixed, and it accounts for the value at the width the benchmark table uses.

### E.2 Prior-tuned random dictionaries

The comparison is against frozen random dictionaries at the same width, solved by the same routine, with their shape parameter and ridge chosen on the prior only — by reconstruction error over a held-out task bank, never by looking at a test equation. This is the selection rule we attribute to Zhang et al. [46].

Table 9 gives the comparison. We include our own "non-learning" arm — the same dual-branch architecture with row-normalised random weights and a single tuned scale — so that the family our ablation already reports is represented at matched width.

The result does not support the claim that the benchmarks measure meta-training, and we report it against our own interest:

A random Fourier dictionary whose frequencies are drawn from the *exact* prior spectrum solves every one of these equations to near machine precision, improving on the meta-trained dictionary by ten orders of magnitude on Poisson and by a factor of 83 on the high-frequency problem. A plain tanh ELM, tuned the same way, also beats it on Poisson and on HighFreq.

Two effects are worth separating. The first is that the comparison is dominated by tuning luck: across seeds the same prior-only criterion moves the selected frequency scale by a factor of eight (its objective is nearly flat, 0.75–0.95 everywhere, i.e. barely better than predicting zero), and the ELM row moves from $6.2 \times 10^{-3}$ to $6.3 \times 10^{-6}$ on Poisson between two seeds. The second, and the one that matters here, is that the equations are easy: if a dictionary places basis mass at the target's frequency, the solve succeeds regardless of how the dictionary was obtained. That is the same statement as the capacity diagnostic, seen from the other side — what limits accuracy is whether the representable band covers the solution, not the provenance of the basis.

We therefore do not claim that meta-training beats random dictionaries. The two diagnostics formalise why this benchmark suite cannot settle the question in either direction: it separates dictionaries by frequency placement rather than by method quality, so a comparison on it is a comparison of tuners.

### E.3 What the benchmarks measure

The frequency-sweep diagnostic sweeps a family of pure sinusoids $\sin(2\pi fx)\sin(2\pi fy)$ and plots the value error against the Laplacian error. Read against the dictionary's band, the sweep separates into two regimes.

For $f \leq 32$ the targets need at most $64$ cycles per unit, which the Fourier branch reaches, and the curve measures what it is meant to measure. There the operator-consistency term of Section 5 has a large, monotone effect: at $f$=32 the value error falls from $5.07 \times 10^{-1}$ at $\lambda$=0 to $1.96 \times 10^{-1}$ at $\lambda$=1, and the Laplacian error from $3.86 \times 10^{4}$ to $1.92 \times 10^{4}$.

For $f = 64$ and $f = 128$ the targets need 128 and 256 cycles per unit, both beyond the branch's maximum of 64, and nothing moves them: the value error at $f$=128 is $5.406 \times 10^{-1}$ at $\lambda$=0 and $5.418 \times 10^{-1}$ at $\lambda$=0.1. A consistent explanation is that these points are outside the representable band rather than diagnostic of operator fidelity, and the invariance to $\lambda$ supports it.

A second check is available from the dictionary itself. Computing the power spectrum of each branch on a $257 \times 257$ no-endpoint grid gives spectral centroids of $0.59$ cycles per unit for the raw-coordinate branch and $4.89$ for the Fourier branch — an eightfold separation, consistent with the design intent, and both far below the 32 cycles per unit that the hardest benchmark actually requires.

The corrected PINN baseline and its smooth/oscillatory amplitudes are in Appendix E.1: the same solver recovers both components on Poisson and neither on Helmholtz.

The matched-width random-dictionary comparison, and the finding that this suite ranks dictionaries by frequency placement rather than by method quality, are in Appendix E.2.

## F The Quasi-Optimal Bound

Let $V_H = \mathrm{span}(\Phi)$ and let $T_N$ be the weighted discrete map the solver inverts. If there is $\alpha_{H,N} > 0$ with $\alpha_{H,N}\|v\|_X \leq \|T_N v\|_{Y_N}$ for every $v \in V_H$, and $T_N$ is bounded by $\beta_N$ on a comparison class containing $u - v$, then

$$\|u - u_H\|_X \;\leq\; \Big(1 + \frac{\beta_N}{\alpha_{H,N}}\Big) \inf_{v \in V_H} \|u - v\|_X,$$

with an added $\eta_N/\alpha_{H,N}$ when the discrete right-hand side is inexact. This is the Cèa-type estimate for a stable formulation and it is standard; we record it only to separate the two factors.

Two caveats, which is why this is an appendix and not a result. First, the collocation system here is solved through normal equations, whose quasi-optimal constant is $1 + (\beta_N/\alpha_{H,N})^2$ rather than $1 + \beta_N/\alpha_{H,N}$ — the square matters at $\kappa \approx 10^{17}$. Second, with row equilibration $\alpha_{H,N} = \sigma_{\min}(A)$ and $\beta_N = \sigma_{\max}(A)$, so the stability factor *is* $\kappa(A)$ and the bound is vacuous by construction. We do not prove a bound on $\alpha_{H,N}$; we measure the rank, and report the bound only to say which factor a value-only objective leaves uncontrolled.

## G Reproduction Check

All MetaColloc numbers below are produced by the code accompanying this submission with an unmodified dictionary. Training uses the hyperparameters we use throughout (H=512, 1000 epochs, 128 tasks per epoch, AdamW at $10^{-3}$) on one RTX 4090, which takes 3281 s. The single change is that the gradient is accumulated in chunks of eight tasks and scaled by $1/128$; the accumulated gradient is identical to one backward pass on the mean loss (verified to $10^{-16}$ relative), but the autograd graph never holds more than eight tasks, which is what fits a 48 GB card.

We first verify that this harness reproduces the reported results, since every later comparison depends on it:

Two of the six equations agree to three significant figures and the rest to the same order of magnitude, which is what one seed against the paper's five should give. A difference in any later arm is therefore attributable to the change under test.

Table 10: Reproduction check at $H{=}512$. Our harness uses one seed where the figures we report average five, so agreement to the same order of magnitude is the appropriate standard; Poisson agrees to three significant figures and Helmholtz to two.

| equation | this harness | reported |
|---|---|---|
| Poisson | **4.234e-5** | 4.23e-5 |
| Helmholtz | **5.067e-1** | 5.08e-1 |
| VarCoeff | 2.76e-9 | 1.86e-9 |
| HighFreq | 8.98e-1 | 8.26e-1 |
| SineGordon | 1.91e-7 | 9.80e-8 |
| KdV | 8.24e-5 | 1.59e-4 |