A GREEDY PDE ROUTER FOR BLENDING NEURAL OPERATORS AND CLASSICAL METHODS

Anonymous authors
Paper under double-blind review

ABSTRACT

When solving PDEs, classical numerical solvers are often computationally expensive, while machine learning methods can suffer from spectral bias, failing to capture high-frequency components. Designing an optimal hybrid iterative solver–where, at each iteration, a solver is selected from an ensemble of solvers to leverage their complementary strengths–poses a challenging combinatorial problem. While greedy selection is desirable for its constant-factor approximation guarantee to the optimal solution under contraction and sequence-structure assumptions, it requires knowledge of the true error at each step, which is unavailable in practice. We address this by proposing an approximate greedy router that efficiently mimics a greedy approach to solver selection. Empirical results on the Poisson, convection–diffusion, and anisotropic diffusion equations show that our method reduces time-to-solution relative to single-solver baselines and existing hybrid approaches such as HINTS.

Natural phenomena and engineered systems are often governed by ordinary and partial differential equations (PDEs). Solving these equations enables a variety of tasks - predicting the evolution of the system through simulation (e.g., forecasting weather using the Navier-Stokes equations) (Kalnay, 2003; Bauer et al., 2015; Staniforth & Côté, 1991), addressing control problems (e.g., optimizing heat shield design for spacecrafts using heat transfer equations) (Anderson, 1989; Tröltzsch, 2010), and tackling inverse problems based on external measurements (e.g., reconstructing brain activity from EEG data using electrophysiological models) (Baillet et al., 2001; Grech et al., 2008).

Traditionally, PDEs are solved using finite difference methods (Smith, 1985; LeVeque, 2007), which discretize the domain and solve a system of equations over the grid points. Finite element methods (Hughes, 2003; Bathe, 2006; Brenner, 2008) instead construct a system of equations based on local basis functions, but likewise rely on numerical linear algebra techniques–such as the Jacobi and Gauss-Seidel (Saad, 2003)–to compute the solution. These iterative methods, however, lack generalization across different initial conditions, boundary conditions, or forcing functions, as even minor changes to any of these parameters require re-solving the entire system of equations from scratch.

These challenges have motivated neural operators (Kovachki et al., 2023)–a class of machine learning models which learn the solution operator directly, enabling fast inference across a range of parameters. They lift the conventional linear layer to an operator–typically a kernel integral operator–between function spaces. Despite strong approximation guarantees (Chen & Chen, 1995), neural operators exhibit spectral bias (Liu & Cai, 2021; Liu et al., 2024; You et al., 2024; Khodakarami et al., 2025; Xu et al., 2025), similar to traditional neural networks (Rahaman et al., 2019; Xu et al., 2019; Luo et al., 2019). They therefore favor learning low-frequency components of the solution and often struggle to capture high-frequency features that iterative solvers excel at.

Recognizing this complementarity, Zhang et al. (2024) proposed HINTS, a hybrid solver that interleaves a neural operator pass every τ^{th} iteration of a classical solver, with τ fixed a priori. While HINTS reports significant empirical gains in convergence and accuracy over the standalone neural operator and Jacobi solver, this fixed schedule can be detrimental: a poorly timed neural correction may increase the error and undo recent progress.

Although state-dependent solver schedules are desirable, this naturally suggests a reinforcement learning formulation for learning routing policies. However, in our setting, the structure of iterative PDE solvers allows for a simpler approach: each solver induces a known update, and its effect can be evaluated through error-based signals. Leveraging this, we adopt a greedy rule that selects, at

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.

each iteration, the solver with the largest immediate error reduction. Such a rule can be near-optimal when the final error satisfies (weak) supermodularity–so that applying a beneficial solver earlier is at least as valuable as applying it later. To support this approach, we make the following contributions:

- We present a general hybrid PDE solver in which a routing rule chooses, at each iteration, among classical solvers and neural operators. With oracle access to the true error, we show that greedy routing achieves a constant-factor approximation to the optimal strategy for linear PDEs under zero-preserving contraction and a sequence-monotonicity condition, with the former establishing weak supermodularity.
- Given that the true error is not observed at test time, a convex surrogate loss is introduced, which, when minimized, enables the learned model to imitate the greedy router without access to true error information. This alignment is guaranteed through Bayes consistency.
- We empirically demonstrate on Poisson, convection–diffusion, and anisotropic-diffusion equations that the approximate greedy router reduces time-to-solution relative to standalone numerical solvers and HINTS.

1 BACKGROUND

Consider the following linear PDE:

$$\mathcal{L}_x^a(u) = f, x \in D; \quad \mathcal{B}_x(u) = b, x \in \partial D \tag{1}$$

where $\mathcal{L}_x^a$ is a differential operator with respect to the spatial variable $x \in D$ parameterized by the coefficient function a, f is a forcing function, u is the PDE solution, $\mathcal{B}_x$ is the boundary operator, and b is the boundary term. Such PDEs can be solved numerically using various discretization strategies, such as finite differences (Smith, 1985; LeVeque, 2007), finite element, and spectral methods (Boyd, 2001; Canuto et al., 2006). Here, we focus on finite differences, which replace derivatives with difference quotients (e.x., $\partial_x u(x) \approx (u(x+h) - u(x))/h$) on a uniform grid.

Formally, let h be the grid size and $G_h(D)$ be a uniform grid with spacing h over D. For a function $g : D \to \mathbb{R}$, we denote its restriction to $G_h(D)$ by $g_h \in \mathbb{R}^N$, where $N = |G_h(D)|$. If $\mathcal{L}_h^a \in \mathbb{R}^{N \times N}$ is the discretized operator, the PDE reduces to a linear system of equations

$$\mathcal{L}_h^a u_h = f_h, \tag{2}$$

with the boundary conditions incorporated in $\mathcal{L}_h^a$. Unlike much of the neural operator literature, which retains function space formulations for discretization invariance, we follow the conventions of classical numerical analysis and represent functions and operators as vectors and matrices. Discretization invariance is not essential here, since we focus on fixed-grid problems.

1.1 ITERATIVE PDE SOLVERS

Direct methods like Gauss Elimination and Thomas Algorithm can be computationally expensive in high-dimensional domains when solving systems like Equation (2). In contrast, iterative methods like Jacobi and Gauss-Seidel offer computational speedups by iteratively updating the solution, gradually converging to the true solution for the PDE. The general iterative update is

$$u^{(t+1)} = u^{(t)} + C\left(f_h - \mathcal{L}_h^a u^{(t)}\right), \tag{3}$$

where $u^{(t)}$ is the tth iterate of the solution and C is a preconditioning matrix. Jacobi uses $C = D^{-1}$, where D is the diagonal of $\mathcal{L}_h^a$, and Gauss-Seidel uses $C = (D + L)^{-1}$, where L denotes the strict lower triangular part. However, these methods tend to damp low frequency parts of the error slowly.

1.2 NEURAL OPERATORS

Suppose we observe samples $\{(a_i, f_i, u_i)\}_{i=1}^N$ such that $(a_i, f_i) \sim \mathcal{P}$ are i.i.d. samples, and $u_i = \mathcal{G}^*(a_i, f_i)$ be generated by a deterministic solution operator $\mathcal{G}^*$. A neural operator (NO) $\mathcal{G}_\theta$ seeks to approximate $\mathcal{G}^*$ by minimizing the expected squared $L^2(D)$ error:

$$\mathcal{R}_{\text{NO}}(\mathcal{G}_\theta) = \mathbb{E}_{(a,f)\sim\mathcal{P}}\left[\int_D \|u_i(x) - \mathcal{G}_\theta(a_i, f_i)(x)\|_2^2 \, dx\right]$$

Neural operators use discretization-invariant layers. Prominent instantiations include Fourier Neural Operators (FNO), which use spectral transforms (Li et al., 2020a), Graph Neural Operators, which perform graph-based aggregation over sampled points (Li et al., 2020b), and DeepONets, which employ a trunk–branch decomposition to map input functions to output functions (Lu et al., 2021).

1.3 Greedy Optimization

Consider maximizing $g(S)$ over $S \subseteq \Omega$ with $|S| \leq T$ where $g : 2^{\Omega} \to \mathbb{R}$ is a set function defined on subsets of a set Ω. An exhaustive search is infeasible, so an efficient alternative is the greedy algorithm. It builds the solution subset iteratively by adding ω^* that yields the largest marginal gain, i.e. $\omega^* = \arg\max_{\omega \in \Omega \setminus S} g(S \cup \omega)$. When g is non-negative, monotone (adding elements never decreases the value), and submodular (diminishing returns), the greedy algorithm achieves a $(1 - e^{-1})$ approximation to the optimal solution (Nemhauser et al., 1978). Formally, a function g is submodular if $g(A \cup \{\omega\}) - g(A) \geq g(B \cup \{\omega\}) - g(B)$ for all $A \subseteq B \subseteq \Omega$ and $\omega \in \Omega \setminus B$; intuitively, delaying an addition cannot increase its benefit. For set function minimization problems, supermodularity, where the inequality flips, plays an analogous role, and the greedy rule achieves constant-factor approximation guarantees (Liberty & Sviridenko, 2017).

The suboptimality of the greedy algorithm has been extensively studied for both set-function minimization (Bounia & Koriche, 2023) and maximization (Das & Kempe, 2018; Feige et al., 2011; Bian et al., 2017; Harshaw et al., 2019) when standard assumptions–non-negativity, monotonicity, and sub/supermodularity–are weakened. In contrast, results on greedy sequence maximization (Streeter & Golovin, 2008; Alaei et al., 2021; Zhang et al., 2015; Bernardini et al., 2020; Van Over et al., 2024; Tschiatschek et al., 2017)–where the ordering of the elements affects the function–have led to sequential analogues of submodularity and monotonicity. We leverage these tools to analyze the suboptimality of the greedy solution to minimizing the final error.

2 General framework for Hybrid Solvers

To solve Equation (2), consider the following hybrid iterative update:

$$u_h^{(t+1)} = u_h^{(t)} + C_{S_t}\left(f_h - \mathcal{L}_h^a u_h^{(t)}\right) \tag{4}$$

where $\mathcal{C} = \{C_j\}_{j=1}^K$ is a set of preconditioning functions and $S_t \in [K] = \{1, \ldots, K\}$ indexes the function chosen at step t. Here, we use "preconditioning function" broadly to include classical linear updates ($C_j(x) = C_j x$ where C_j is a matrix), and learned models, such as neural operators. This update generalizes the classical iterative update in Equation (3), by allowing C_j to be non-linear and enabling the preconditioning function to be adaptively chosen at every step. $\mathcal{C}$ can also accommodate parameterized solver families (e.g., different Jacobi relaxation weights), enabling adaptive selection of solver parameters. As the number of solvers K grows, the chance of selecting a more effective update increases. Furthermore, HINTS (Zhang et al., 2024) is a special case of this hybrid solver with $K = 2$, where C_1 is a neural operator and C_2 is a classical preconditioner. Its routing rule is given by $S_t = \mathbf{1}_{t \bmod \tau > 0} + 1$, which selects C_1 every τ steps and C_2 otherwise.

Let $e_h^{(t)} = u_h - u_h^{(t)}$ denote the error at step t. Then,

$$e_h^{(t+1)} = u_h - u_h^{(t)} - C_{S_t}\left(\mathcal{L}_h^a u_h - \mathcal{L}_h^a u_h^{(t)}\right) = \left(I - C_{S_t} \circ \mathcal{L}_h^a\right)\left(e_h^{(t)}\right)$$

where I is the identity map and $I - C_j \circ \mathcal{L}_h^a$ is the error propagation function for solver j. Here, "$\circ$" denotes function composition (matrix multiplication for linear updates). The objective is to select a sequence $S = (S_1, \ldots, S_T)$ that minimizes the error norm after T steps or $\|e_h^{(T)}\|_2^2$:

$$\min_{S_t \in [K], |S| \leq T} h(S) := \left\| \left(I - C_{S_{|S|}} \circ \mathcal{L}_h^a\right) \circ \cdots \circ \left(I - C_{S_1} \circ \mathcal{L}_h^a\right)\left(e_h^{(0)}\right) \right\|_2^2 \tag{5}$$

with compositions applied from right to left, so that the S_1 update acts on the initial error. Our main analysis treats each solver application as one action and provides guarantees over a fixed number of routing steps. When solver runtimes differ, these guarantees characterize convergence in terms of actions but do not directly imply faster wall-clock convergence. In our experiments, we address this issue by grouping solver iterations into approximately equal-cost macro-actions, as detailed in Section 6. Appendix C extends the analysis to actions with genuinely heterogeneous costs.

3 GREEDY ALGORITHM

Equation (5) defines a combinatorial optimization problem with a search space exponential in T, making exact optimization intractable for large T or K. As a starting point, we consider an "omniscient" greedy algorithm that assumes access to the true initial error $e_h^{(0)}$. This is unrealistic since knowledge of $e_h^{(0)}$ could directly recover the solution via $u_h = u_h^{(0)} + e_h^{(0)}$, but it provides a useful benchmark. In Section 4, we relax this assumption using a practical learning strategy that is Bayes consistent with this omniscient rule, thereby recovering the guarantees shown below.

As discussed in Section 1.3, when supermodularity and monotonicity hold, the greedy rule–such as the one described in Algorithm 1–enjoys constant-factor approximation guarantees. However, classical results focus on *set* functions, with only recent extensions made to sequences. Building on this line, we introduce a sequence-based notion of weak supermodularity.

Algorithm 1 Greedy algorithm for a hybrid PDE solver

Require: $\{C_j\}_{j=1}^K, T, \mathcal{L}_h^a, e_h^{(0)}$
  $S^0 \leftarrow \emptyset$
  for $t < T$ **do**
    $S^{t+1} \leftarrow S^t \oplus \arg\min_{j\in[K]} \|(I - C_j \circ \mathcal{L}_h^a)(e_h^{(t)})\|_2^2$
    $e_h^{(t+1)} \leftarrow (I - C_{S_{t+1}} \circ \mathcal{L}_h^a)(e_h^{(t)})$
  end for
  return S^T

Let Ω^* denote the space of sequences with elements in Ω. For $S, S' \in \Omega^*$, we denote their concatenation as $S \oplus S'$. S is a prefix of S' or $S \preceq S'$ if $S' = S \oplus L$ for some $L \in \Omega^*$. A function $g : \Omega^* \to \mathbb{R}$ is prefix monotonically non-increasing if $g(S \oplus S') \leq g(S)$ for all $S, S' \in \Omega^*$, and postfix monotonically non-increasing if $g(S' \oplus S) \leq g(S)$ for all $S, S' \in \Omega^*$. A prefix non-increasing function g is sequence supermodular if, for all $S', S \in \Omega^* : S \preceq S'$, it holds that

$$g(S) - g(S \oplus \omega) \geq g(S') - g(S' \oplus \omega), \quad \forall \omega \in \Omega \tag{6}$$

However, $h(S)$ (defined in Equation (5)) may not satisfy this property in general. Therefore, we introduce weak sequence supermodularity. A prefix non-increasing function g is weakly supermodular with respect to $S' \in \Omega^*$ if there exists $\alpha(S') \geq 1$, for every $S \in \Omega^{\leq |S'|}$, such that

$$g(S) - g(S \oplus S') \leq \alpha(S') \sum_{i\in[|S'|]} g(S) - g(S \oplus S'_i) \tag{7}$$

The parameter $\alpha(S')$ or the supermodularity ratio quantifies deviation from exact sequence supermodularity. Expanding the $g(S) - g(S \oplus S')$ as a telescoping sum $\sum_{i=1}^{|S'|} g(S \oplus (S'_1, \ldots, S'_{i-1})) - g(S \oplus (S'_1, \ldots, S'_i))$ shows that the marginal decrease from appending S'_i after its predecessors is controlled by the effect of appending S'_i directly to S. Thus, postponing the inclusion of S'_i cannot yield a significantly larger benefit than adding it earlier. The supermodularity ratio $\alpha(S)$ quantifies the extent to which future gains from delays may exceed immediate gains.

Having introduced these notions, we now characterize the suboptimality of greedy solutions of weakly supermodular and postfix monotonic sequence functions in Theorem 3.1.

Theorem 3.1. *Let $g : \Omega^* \to \mathbb{R}$ be a weakly supermodular function with respect to the optimal solution $O = \arg\min_{S\in\Omega^T} g(S)$ with a supermodularity ratio of $\alpha(O)$ and postfix monotonicity. Let the greedy solution of length T be S^T. If $\phi_T(\alpha) = \left(1 - \frac{1}{\alpha T}\right)^T$, then*

$$g(\emptyset) - g(S^T) \geq (1 - \phi_T(\alpha(O)))\,(g(\emptyset) - g(O))$$

The proof of Theorem 3.1 appears in Appendix A.2. As $T \to \infty$, $1 - \phi_T(\alpha(O))$ decreases to $1 - e^{-1/\alpha(O)}$, so the worst-case performance of the greedy rule saturates rather than degrades with horizon. Larger $\alpha(O)$, indicating higher reward for delayed inclusions, loosens the bound. Theorem 3.1 requires h to be weakly supermodular with respect to the optimum and postfix monotone. Proposition 3.2 establishes weak-supermodularity under zero-preserving contraction assumptions. We defer the proof to Appendix A.3.

Proposition 3.2. *Suppose that for all $j \in [K]$, the error propagation function $I - C_j \circ \mathcal{L}_h^a$ is ρ_j-Lipschitz continuous with $\rho_j < 1$, and that $(I - C_j \circ \mathcal{L}_h^a)(0_N) = 0_N$. Then, the function h is weakly supermodular with respect to the optimal solution O, with*

$$\alpha(O) = \max\left\{\frac{1}{T - \sum_{i=1}^{T} \rho_{O_i}^2}, 1\right\}$$

The conditions of Proposition 3.2 are natural. For classical solvers, the Lipschitz constant is $\|I_N - C_j \mathcal{L}_h^a\|$, which is typically less than 1 for well-posed linear elliptic PDEs (i.e., the update is damping errors). For neural networks, Lipschitz continuity can be enforced via weight regularization (Gouk et al., 2021) and a sufficiently trained model should approximate $(\mathcal{L}_h^a)^{-1}$ well enough to make the Lipschitz constant small. The requirement $(I - C_j \circ \mathcal{L}_h^a)(0_N) = 0_N$ is both natural and desirable: it precludes spurious updates when the residual $f_h - \mathcal{L}_h^a u_h^{(t)}$ is 0. This holds by design for classical schemes, and it can be enforced for any learned model by excluding bias terms.

The form of $\alpha(O)$ in Proposition 3.2 highlights that the suboptimality factor in Theorem 3.1 is governed by the collective contraction factors of the solvers chosen by the optimal solution: as $\sum_{i=1}^{T} \rho_{O_i}^2 \to T$, $\alpha(O)$ grows, resulting in a weaker bound. Zero-preserving contraction guarantees prefix monotonicity, but does not in general guarantee postfix monotonicity. A sufficient condition for postfix monotonicity is pairwise commutativity of the error propagation maps: prepended maps can then be commuted past the remaining sequence and applied as contractions to its output. We do not assume commutativity for general solver ensembles; the approximation guarantee applies when the stated postfix-monotonicity condition holds. We empirically assess zero preservation, contraction, and postfix monotonicity for the solvers used in the experiments in Appendix E.

Theorem 3.1 thus indicates that the approximation guarantee is strongest when the sequence is nearly supermodular ($\alpha(O) \approx 1$). Generally, if all error propagation maps share a common orthonormal eigenbasis, supermodularity is established. This occurs, for example, for linear, constant-coefficient PDEs with periodic boundary conditions where the solver ensemble includes Jacobi and a single-layer linear Fourier Neural Operator. The proof of Proposition 3.3 is deferred to Appendix A.5.

Proposition 3.3. *Let $\|I_N - C_j \mathcal{L}_h^a\| \leq 1$ for all $j \in [K]$ and $(I - C_j \mathcal{L}_h^a) = P \Lambda_j P^{-1}$, where P is an orthogonal matrix. Then, h is supermodular.*

4 Approximate Greedy Router

The results in Section 3 indicate that the error reduction from the greedy solution closely matches that of the optimal sequence, but this is predicated on having access to the initial error, $e_h^{(0)}$. Inaccurate error estimates can result in poor solver decisions, causing errors to amplify. To remedy this, we learn a router r that selects solvers myopically, as in Algorithm 1, without access to true errors.

We adopt the following learning setup. Let $\mathcal{A}, \mathcal{F}, \mathcal{U} \subseteq \mathbb{R}^N$ denote the spaces of coefficient, forcing, and solution functions on the grid $G_h(D)$. We assume an application-specific data distribution $\mathcal{P}_{\mathcal{A}\times\mathcal{F}}$ over $\mathcal{A} \times \mathcal{F}$ that reflects test time conditions. During training, (a_h, f_h) is drawn from $\mathcal{P}_{\mathcal{A}\times\mathcal{F}}$, and a high-accuracy reference solution u_h is computed, providing the true per-step error. The router r is learned offline on this data; at test time, it operates without access to true errors.

At iteration t, router r selects a solver using the coefficient $a_h \in \mathcal{A}$, forcing $f_h \in \mathcal{F}$, and the current iterate $u_h^{(t)} \in \mathcal{U}$. If $r(a_h, f_h, u_h^{(t)}) = j$, the next iterate is computed with solver j, resulting in an error of $\|(I - C_j \circ \mathcal{L}_h^a)(e_h^{(t)})\|_2^2$. Learning such a router requires minimizing the following loss:

$$l_{\text{route}}\left(r, a_h, f_h, u_h^{(t)}, u_h\right) = \sum_{j=1}^{K} \left\|(I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right)\right\|_2^2 \mathbf{1}_{r(a_h, f_h, u_h^{(t)})=j} \tag{8}$$

with risk $\mathcal{R}_{\text{route}}(r) = \mathbb{E}_{a_h, f_h \sim \mathcal{P}_{\mathcal{A}\times\mathcal{F}}}[l_{\text{route}}(r, a_h, f_h, u_h^{(t)}, u_h)]$. For analysis, we fix the iterate generation via teacher-forcing (Williams & Zipser, 1989; Lamb et al., 2016): during training, the iterates fed to the router are produced by an oracle greedy rollout, not by the router's own past choices. Formally, with $u_h^{(0)} = 0_N$, for $t \geq 1$,

$$j_t^* \in \text{argmin}_{j\in[K]} \left\|(I - C_j \circ \mathcal{L}_h^a)\left(e_h^{(t-1)}\right)\right\|_2^2, \quad u_h^{(t)} = u_h^{(t-1)} + C_{j_t^*}\left(f_h - \mathcal{L}_h^a u_h^{(t-1)}\right)$$

Thus, $\{u_h^{(t)}\}$ is a deterministic function of (a_h, f_h). At each t, l_{route} is evaluated on $r(a_h, f_h, u_h^{(t)})$ which takes in the teacher-forced iterate. Finally, the greedy choice j_{t+1}^* is used to advance $u_h^{(t+1)}$. Thus the input distribution seen by r depends on (a_h, f_h), not on the router's predictions. However, full teacher forcing induces a distributional mismatch at test time (exposure bias). In practice, we mitigate this distribution shift using a DAgger-style dataset-aggregation procedure. After training on oracle-generated states, we roll out the learned router with controlled exploration, label the newly visited states using the greedy oracle, and retrain on the aggregated data. See Appendix D for details.

Minimizing $\mathcal{R}_{\text{route}}(r)$ yields the following Bayes-Optimal Router:

$$r^*(a_h, f_h, u_h^{(t)}) \in \text{argmin}_{j\in[K]} \left\| (I - C_j \circ \mathcal{L}_h^a) \left(e_h^{(t)} \right) \right\|_2^2 \tag{9}$$

which matches the update in Algorithm 1. Hence, l_{route} is consistent with learning the greedy rule.

4.1 Surrogate Loss

Since Equation (8) is discontinuous and non-convex, direct minimization is intractable in practice. We instead introduce a surrogate loss that (1) is convex, enabling efficient optimization; (2) upper bounds the original loss; and (3) is Bayes consistent, ensuring the Bayes optimal decision of the original loss is preserved upon minimization. Formally, a surrogate ϕ is considered to be Bayes consistent with respect to the loss l if

$$\lim_{n\to\infty} \mathcal{R}_\phi(f_n) - \mathcal{R}_\phi^* = 0 \implies \lim_{n\to\infty} \mathcal{R}_l(f_n) - \mathcal{R}_l^* = 0$$

where $\mathcal{R}_l(f) = \mathbb{E}[l(f(X), Y)]$ and $\mathcal{R}_l^* = \inf_f \mathcal{R}_l(f)$. In other words, over the class of all measurable functions, if the risk of a sequence of learned hypotheses $\{f_n\}$ converges to the optimal risk under ϕ, it also converges to the optimal risk with respect to the original loss l.

To define a surrogate loss for the routing problem, consider a set of scoring functions $\mathbf{g} = \{g_j\}_{j=1}^K$ with $g_j : \mathcal{A} \times \mathcal{F} \times \mathcal{U} \to \mathbb{R}$ and we define the router as $r(a, f, u^{(t)}) = \text{argmax}_{j\in[K]} g_j(a, f, u^{(t)})$. For example, $\mathbf{g}$ can be a neural network with K outputs. Then, minimizing the following surrogate loss yields the same decision as minimizing Equation (8):

$$\Psi\left(\mathbf{g}, a_h, f_h, u_h^{(t)}, u_h\right) = -\sum_{j=1}^K \sum_{k=1}^K \tilde{c}_k(a_h, u_h^{(t)}, u_h) \mathbf{1}_{k\neq j} \log \left(\frac{\exp\left(g_j(a_h, f_h, u_h^{(t)})\right)}{\sum_{m=1}^K \exp\left(g_m(a_h, f_h, u_h^{(t)})\right)} \right) \tag{10}$$

where $\tilde{c}_j(a_h, u_h^{(t)}, u_h) = \|(I - C_j \circ \mathcal{L}_h^a)(u_h - u_h^{(t)})\|_2^2$. The Ψ-risk is denoted by $\mathcal{R}_\Psi(\mathbf{g}) = \mathbb{E}_{a_h, f_h \sim \mathcal{P}_{\mathcal{A}\times\mathcal{F}}}[\Psi(\mathbf{g}, a_h, f_h, u_h^{(t)}, u_h)]$. The convexity of Ψ with respect to $\mathbf{g}$ follows from the convexity of negative log-softmax function in its inputs. Moreover, Ψ upper bounds l_{route} up to a constant factor, and we refer the reader to Appendix B.2 for the proof. Finally, Theorem 4.1 shows that Ψ achieves Bayes consistency with respect to l_{route}; the proof can be found in Appendix B.3.

Theorem 4.1. *Let $\tilde{c}_j(a_h, u_h^{(t)}, u_h) < \bar{E} < \infty$ for all $j \in [K]$. If there exists $j \in [K]$ such that $\tilde{c}_j(a_h, u_h^{(t)}, u_h) > E_{min} > 0$, then, for any collection of solvers $\{C_j\}_{j=1}^K$ and linear discrete operator $\mathcal{L}_h^a$, Ψ is Bayes consistent surrogate for l_{route}.*

Theorem 4.1 is a cost-sensitive analogue of the classical Bayes-consistency of multiclass cross-entropy for 0-1 loss: in the infinite-sample limit, minimizing cross-entropy recovers the true conditional class probabilities, so the induced decision is Bayes optimal. Our result extends this to cross-entropy with instance-dependent weights $\sum_{k\neq j}^K \tilde{c}_k(a_h, u_h^{(t)}, u_h)$. The uniform upper bound holds when all preconditioning functions are error-damping. It is also reasonable to assume at least one solver cannot annihilate the error in one step, yielding the lower bound. Under these conditions, minimizing Ψ recovers the Bayes-optimal router of Equation (9), i.e., the greedy solution of Equation (5). Following the work of Mao et al. (2024), the proof uses standard conditional-risk calibration: we relate the excess risks of l_{route} and Ψ and take the infinite-sample limit.

5 Related Works

Hybrid PDE Solvers: Early data-driven solvers sought convergence guarantees by predicting components of iterative schemes such as preconditioners, multi-grid smoothers or restriction matrices (Taghibakhshi et al., 2021; Caldana et al., 2024; Kopaničáková & Karniadakis, 2025; Huang et al., 2022; Katrutsa et al., 2020). These works, however, do not leverage the neural surrogates' ability to generalize across coefficients and forcings (Section 1.2) and offer only modest speedups over classical solvers. HINTS (Zhang et al., 2024) introduced hybrid solvers that interleave a classical method with a pre-trained DeepONet on a fixed schedule (e.g., 24 Jacobi steps, then one DeepONet correction), accelerating convergence over the standalone numerical solver. Subsequent works extend HINTS to new geometries (Kahana et al., 2023) and analyze error mode damping, replacing DeepONet with alternatives such as MIONet (Jin et al., 2022) (Hu & Jin, 2025) or FNO (Li et al., 2020a; Cui et al., 2022). However, these hybrid solvers are limited in two ways: they, firstly, pair the trained surrogate with a *single* numerical solver, and secondly rely on a fixed, heuristic schedule. Our method addresses these shortcomings, resulting in significant empirical improvements (Section 6).

Model Routing: Routing (Shnitzer et al., 2023; Hu et al., 2024; Ding et al., 2024; Huang et al., 2025) selects, for each input, a model from a fixed set to optimize a task metric under cost or latency constraints. Simple heuristics–e.g., thresholding a cheap model's uncertainty estimates (Chuang et al., 2024; 2025)–often poorly balance the cost-accuracy tradeoff, motivating learned routers that map inputs to the best model (Hari & Thomson, 2023; Mohammadshahi et al., 2024; Šakota et al., 2024). We instead route among numerical and neural PDE solvers at each iteration. Our work is the first to apply routing to the problem of learning hybrid PDE solvers, in contrast to prior approaches that use fixed schedules. Additionally, by exploiting the algebraic structure of solvers, we derive theoretical guarantees for our routing strategy (Section 3), unlike typical model-routing settings.

Mixture of Experts: Classical mixture-of-experts (MoE) (Jacobs et al., 1991; Jordan & Jacobs, 1994) uses a gating network to combine the outputs of multiple experts, trained jointly with the gate. In modern LLMs, MoE instead performs sparse, token-level routing to a subset of in-layer experts, enabling large capacity without proportional compute, typically with load-balancing mechanisms (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022; Jiang et al., 2024). Our method can be regarded as gating of a different kind: "experts" (solvers) are not jointly trained with the router, and a single router is reused across iterations, unlike MoE which employs layer-specific gates.

6 Experiments

We evaluate the approximate greedy router on the Poisson, Convection-Diffusion (ConvDiff), and Anistropic Diffusion (Aniso) equations posed on the unit domain $D = [0, 1]^2$:

$$\begin{aligned}
\textbf{Poisson:} \quad & -\Delta u(\mathbf{x}) = f(\mathbf{x}), && \mathbf{x} \in D, \\
\textbf{ConvDiff:} \quad & -\Delta u(\mathbf{x}) + 20\frac{\partial u}{\partial x_1}(\mathbf{x}) + 20\frac{\partial u}{\partial x_2}(\mathbf{x}) = f(\mathbf{x}), && \mathbf{x} \in D, \\
\textbf{Aniso:} \quad & -0.01\frac{\partial^2 u}{\partial x_1^2}(\mathbf{x}) - \frac{\partial^2 u}{\partial x_2^2}(\mathbf{x}) = f(\mathbf{x}), && \mathbf{x} \in D.
\end{aligned}$$

We impose periodic boundary conditions and center f to satisfy $\int_D f(\mathbf{x})d\mathbf{x} = 0$; Dirichlet results are deferred to Appendix F.8. The domain D is discretized on a 128×128 uniform grid; results on a finer grid are provided in Appendix F.7. Data is sampled from a zero-mean Hierarchical Gaussian Random Field on the periodic domain with covariance operator $\alpha(-\Delta + \beta I)^{-\gamma}$, where α, β, γ vary across samples (See Appendix D). We run two experimental settings. First, we restrict to solver pairs ($K = 2$) and compare against HINTS and standalone classical solvers. Second, we assess performance as the solver ensemble grows; since HINTS does not support this setting, we simply compare each ensemble router with its strongest constituent pairwise router. Over 64 test samples, we report the mean charged wall-clock time to a relative solution error tolerance, $\|u_h - u_h^{(t)}\|_2/\|u_h\|_2$. Solution error is used only for evaluation, not as a stopping criterion. The paired and growing-ensemble studies use tolerances of 10^{-12} and 10^{-6}, respectively.

Table 1: Selected-action wall-clock replay time in milliseconds (lower is better), reported as mean (standard error) in milliseconds over successful cases. Brackets indicate the number of successes when coverage is incomplete, and a dash indicates no successful cases. Bold marks the fastest deployable method with full coverage in each solver family; True-Greedy is an oracle.

Equation	Poisson	Conv	Aniso
Methods	Time (ms)	Time (ms)	Time (ms)
Jacobi-related Solvers			
Jacobi Only	6428.52 (4.15)	10146.76 (14.21)	–
HINTS-Jacobi ($m = 50/50/50$)	2288.27 (125.46)	3278.90 (161.52)	**1845.77 (126.04)**
Learned Greedy-Jacobi	**1280.80 (64.97)**	**2900.52 (181.73)**	1117.90 (77.21) [62/64]
True-Greedy-Jacobi	*1226.02 (66.29)*	*2120.53 (103.75)*	*1130.38 (75.91)*
GS-related Solvers			
GS Only	10224.51 (6.58)	9921.16 (13.81)	–
HINTS-GS ($m = 5/5/25$)	48.26 (0.31)	45.60 (0.15)	216.46 (1.58)
Learned Greedy-GS	**35.63 (0.71)**	**35.60 (0.75)**	**55.53 (2.28)**
True-Greedy-GS	*34.94 (0.81)*	*35.77 (0.73)*	*53.38 (2.42)*
SymGS-related Solvers			
SymGS Only	9210.31 (6.18)	9770.15 (13.53)	–
HINTS-SymGS ($m = 5/5/5$)	37.57 (0.08)	40.77 (0.36)	236.63 (2.77)
Learned Greedy-SymGS	**35.04 (0.68)**	**36.69 (0.73)**	**69.44 (4.28)**
True-Greedy-SymGS	*34.21 (0.79)*	*36.06 (0.82)*	*55.12 (2.73)*
SOR(1.5)-related Solvers			
SOR(1.5) Only	3408.10 (2.17)	2391.65 (3.32)	–
HINTS-SOR(1.5) ($m = 10/10/10$)	56.19 (0.60)	56.53 (0.16)	121.00 (0.83)
Learned Greedy-SOR(1.5)	**44.60 (0.97)**	**45.97 (1.06)**	**47.98 (0.80)**
True-Greedy-SOR(1.5)	*42.68 (0.95)*	*46.39 (1.04)*	*46.58 (0.81)*
MG-related Solvers			
MG Only	36.19 (0.36)	41.51 (0.39)	871.50 (6.76)
HINTS-MG ($m = 5/5/5$)	32.09 (0.32)	36.46 (0.30)	337.11 (3.57)
Learned Greedy-MG	**29.93 (0.46)**	**33.71 (0.56)**	**82.76 (5.05)**
True-Greedy-MG	*29.86 (0.46)*	*33.35 (0.49)*	*78.04 (5.00)*

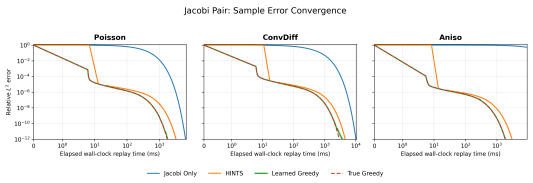

Figure 1: Convergence histories for representative test instances using Jacobi. Columns: Poisson, Convection Diffusion, and anisotropic Diffusion. Greedy yields the fastest monotone decay.

Paired Solver Experiments: We seek to optimally solve the aforementioned PDEs with access to a single numerical scheme and a trained neural operator. In particular, we consider settings coupling the neural solver with Jacobi, Gauss-Seidel (GS), and Symmetric Gauss-Seidel (SymGS) solvers, successive over-relaxation (SOR) with $\omega = 1.5$, and multigrid (MG). For each pair, we compare our learned router against the corresponding single-solver schedules and HINTS variants. The reported HINTS results use the best-performing cadence for each PDE and solver family. To account for heterogeneous solver runtimes, we measure each solver update time c_j and define a classical macro-action as $m_j = \max\{1, \text{round}(c_{NO}/c_j)\}$ consecutive iterations, using one neural correction as the reference action. Macro sizes are calibrated on validation data and frozen before testing. We train a separate MLP router for each pair using the loss in Equation (10). Training details of the neural operators and the routers can be found in Appendix D. To ensure fairness, comparisons are restricted to methods with identical solver access, and are grouped by solver family (e.g., Jacobi-related methods). We additionally evaluate a *true-greedy* oracle that, at each routing round, evaluates every available macro-action and selects the one yielding the largest immediate error reduction. This oracle has access to the ground-truth error and is therefore not implementable in practice, but helps evaluate how well the learned router approximates the greedy policy.

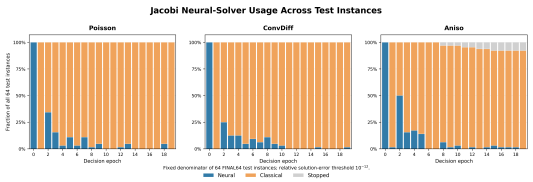

Figure 2: Action usage of the learned Jacobi router across 64 test instances. Bars show the fraction assigned to the neural action, the classical action, or already converged at each of the first 20 routing rounds. Results for other solvers appear in Appendix F.6.

Table 2: Selected-action wall-clock replay time for growing solver ensembles, reported in milliseconds as mean (standard error) over successful cases. The pairwise comparator is selected on validation data. Brackets indicate incomplete coverage, a dash indicates no successes, and one-sided paired-test p-values are reported for full-coverage comparisons.

Actions	**Method**	**Poisson**		**ConvDiff**		**Aniso**	
		Time (ms)	p-value	Time (ms)	p-value	Time (ms)	p-value
$\mathcal{A}_3$	Best pairwise router	9.21 (0.69)	–	18.56 (0.83)	–	16.92 (0.22)	–
	Router($\mathcal{A}_3$)	**4.33 (0.27)**	$< 10^{-3}$	**6.99 (0.30)**	$< 10^{-3}$	**14.45 (0.37)**	$< 10^{-3}$
$\mathcal{A}_4$	Best pairwise router	9.21 (0.69)	–	16.76 (0.76)	–	14.93 (0.32)	–
	Router($\mathcal{A}_4$)	**4.33 (0.27)**	$< 10^{-3}$	**7.05 (0.30)**	$< 10^{-3}$	**14.60 (0.38)**	0.057
$\mathcal{A}_5$	Best pairwise router	9.21 (0.69)	–	14.09 (0.65)	–	14.93 (0.32)	–
	Router($\mathcal{A}_5$)	**4.33 (0.27)**	$< 10^{-3}$	**7.20 (0.31)**	$< 10^{-3}$	**14.56 (0.38)**	0.030
$\mathcal{A}_6$	Best pairwise router	9.21 (0.69)	–	14.09 (0.65)	–	14.93 (0.32)	–
	Router($\mathcal{A}_6$)	**4.40 (0.28)**	$< 10^{-3}$	**7.16 (0.31)**	$< 10^{-3}$	**14.60 (0.38)**	0.058

As shown in Table 1, the greedy router generally reduces time-to-solution relative to its single-solver and HINTS counterparts. Moreover, its performance closely tracks that of the true-greedy oracle, indicating that our surrogate-based training procedure effectively approximates the greedy strategy. The representative convergence histories in Figure 1 illustrate an important distinction between adaptive and fixed routing: a well-timed neural-solver update can substantially accelerate convergence, whereas a poorly timed update can increase the error. The neural-operator usage patterns in Figure 2 further show that the router does not invoke the learned correction at a fixed frequency. Instead, it adapts neural-operator usage to the evolving solver state.

Solver Ensembles Experiments: We consider routing over more than two solvers. For each PDE, we construct nested ensembles $\mathcal{A}_3 \subset \cdots \subset \mathcal{A}_6$ using a PDE-specific action order frozen on validation data. For Poisson and ConvDiff, the available actions include weighted Jacobi ($\omega = 0.67$), GS, SOR(1.5), MG, a Fourier neural operator, and a local neural correction. For anisotropic diffusion, we instead use x_2-line Jacobi and semi-coarsened MG to account for its strong directional coupling, together with SOR and global and directional neural corrections. We compare each ensemble router with its strongest constituent pairwise router, selected on validation data. As shown in Table 2, $\mathcal{A}_3$ substantially reduces time-to-solution relative to the strongest pairwise router across all three PDEs, and the improvement persists as the action set grows. The improvement persists as the Poisson and convection–diffusion ensembles grow, while the larger anisotropic ensembles retain smaller gains. Thus, most of the benefit comes from a small, PDE-appropriate set of complementary actions. Additional experiments are provided in Appendix F.

7 Discussion

We have introduced an adaptive method for selecting solvers that efficiently minimizes the final error of a PDE in iterative methods, opening many directions for future work. Our neural solvers are trained without accounting for their downstream roles in correction term prediction; jointly training the ML solver and router could yield larger gains. Reinforcement learning may provide a practical approach for learning routing policies that directly optimize solution accuracy under compute budgets and extend naturally to continuously parameterized solver families. Finally, framing routing as an online learning problem could enable adaptation to unknown deployment conditions.

AI USE STATEMENT

We used generative AI tools to provide feedback on experimental methodology, assist with the interpretation of experimental results, and support the development and debugging of experimental and evaluation code. We also used generative AI for recommended-disclosure tasks, including drafting and editing portions of the manuscript for clarity, suggesting organizational revisions, and limited brainstorming about experimental presentation. We did not use generative AI to formulate the central theoretical claims, supply ingredients for or write the mathematical proofs, generate research data, conduct literature review, or identify relevant literature. All AI-assisted code, analyses, and text were reviewed and verified by the authors. We take responsibility for the final content of this work, including all claims, experimental results, and artifacts produced with the aid of generative AI.

REFERENCES

Saeed Alaei, Ali Makhdoumi, and Azarakhsh Malekian. Maximizing sequence-submodular functions and its application to online advertising. *Management Science*, 67(10):6030–6054, 2021.

John David Anderson. *Hypersonic and high temperature gas dynamics*. Aiaa, 1989.

Sylvain Baillet, John C Mosher, and Richard M Leahy. Electromagnetic brain mapping. *IEEE Signal processing magazine*, 18(6):14–30, 2001.

Klaus-Jürgen Bathe. *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015.

Sara Bernardini, Fabio Fagnani, and Chiara Piacentini. Through the lens of sequence submodularity. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pp. 38–47, 2020.

Andrew An Bian, Joachim M Buhmann, Andreas Krause, and Sebastian Tschiatschek. Guarantees for greedy maximization of non-submodular functions with applications. In *International conference on machine learning*, pp. 498–507. PMLR, 2017.

Louenas Bounia and Frederic Koriche. Approximating probabilistic explanations via supermodular minimization. In *Uncertainty in Artificial Intelligence*, pp. 216–225. PMLR, 2023.

John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

Susanne C Brenner. *The mathematical theory of finite element methods*. Springer, 2008.

Matteo Caldana, Paola F Antonietti, et al. A deep learning algorithm to accelerate algebraic multigrid methods in finite element solvers of 3d elliptic pdes. *Computers & Mathematics with Applications*, 167:217–231, 2024.

Claudio Canuto, M. Yousuff Hussaini, Alfio Quarteroni, and Thomas A. Zang. *Spectral Methods: Fundamentals in Single Domains*. Springer, 2006. ISBN 9783540307264.

Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4):911–917, 1995.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route llms with confidence tokens. *arXiv preprint arXiv:2410.13284*, 2024.

Yu-Neng Chuang, Leisheng Yu, Guanchu Wang, Lizhe Zhang, Zirui Liu, Xuanting Cai, Yang Sui, Vladimir Braverman, and Xia Hu. Confident or seek stronger: Exploring uncertainty-based on-device llm routing from benchmarking to generalization. *arXiv preprint arXiv:2502.04428*, 2025.

Chen Cui, Kai Jiang, Yun Liu, and Shi Shu. Fourier neural solver for large sparse linear algebraic systems. *Mathematics*, 10(21):4014, 2022.

Abhimanyu Das and David Kempe. Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection. *Journal of Machine Learning Research*, 19 (3):1–34, 2018.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks VS Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*, 2024.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Uriel Feige, Vahab S Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

Roberta Grech, Tracey Cassar, Joseph Muscat, Kenneth P Camilleri, Simon G Fabri, Michalis Zervakis, Petros Xanthopoulos, Vangelis Sakkalis, and Bart Vanrumste. Review on solving the inverse problem in eeg source analysis. *Journal of neuroengineering and rehabilitation*, 5:1–33, 2008.

Surya Narayanan Hari and Matt Thomson. Tryage: Real-time, intelligent routing of user prompts to large language models. *arXiv preprint arXiv:2308.11601*, 2023.

Chris Harshaw, Moran Feldman, Justin Ward, and Amin Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *International Conference on Machine Learning*, pp. 2634–2643. PMLR, 2019.

Jun Hu and Pengzhan Jin. A hybrid iterative method based on mionet for pdes: Theory and numerical examples. *Mathematics of Computation*, 2025.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*, 2024.

Ru Huang, Ruipeng Li, and Yuanzhe Xi. Learning optimal multigrid smoothers via neural networks. *SIAM Journal on Scientific Computing*, 45(3):S199–S225, 2022.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. Routereval: A comprehensive benchmark for routing llms to explore model-level scaling up in llms. *arXiv preprint arXiv:2503.10657*, 2025.

Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2003.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Pengzhan Jin, Shuai Meng, and Lu Lu. Mionet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6):A3490–A3514, 2022.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Adar Kahana, Enrui Zhang, Somdatta Goswami, George Karniadakis, Rishikesh Ranade, and Jay Pathak. On the geometry transferability of the hybrid iterative numerical solver for differential equations. *Computational Mechanics*, 72(3):471–484, 2023.

Eugenia Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.

Alexandr Katrutsa, Talgat Daulbaev, and Ivan Oseledets. Black-box learning of multigrid parameters. *Journal of Computational and Applied Mathematics*, 368:112524, 2020.

Siavash Khodakarami, Vivek Oommen, Aniruddha Bora, and George Em Karniadakis. Mitigating spectral bias in neural operators via high-frequency scaling for physical systems. *arXiv preprint arXiv:2503.13695*, 2025.

Alena Kopaničáková and George Em Karniadakis. Deeponet based preconditioning strategies for solving parametric linear systems of equations. *SIAM Journal on Scientific Computing*, 47(1): C151–C181, 2025.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29, 2016.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM, 2007.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020b.

Edo Liberty and Maxim Sviridenko. Greedy minimization of weakly supermodular set functions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*, pp. 19–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017.

Lizuo Liu and Wei Cai. Multiscale deeponet for nonlinear operators in oscillatory function spaces for building seismic wave responses. *arXiv preprint arXiv:2111.04860*, 2021.

Xinliang Liu, Bo Xu, Shuhao Cao, and Lei Zhang. Mitigating spectral bias for the multiscale operator learning. *Journal of Computational Physics*, 506:112944, 2024.

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.

Tao Luo, Zheng Ma, Zhi-Qin John Xu, and Yaoyu Zhang. Theory of the frequency principle for general deep neural networks. *arXiv preprint arXiv:1906.09235*, 2019.

Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications. In *International conference on Machine learning*, pp. 23803–23828. pmlr, 2023.

Anqi Mao, Mehryar Mohri, and Yutao Zhong. Regression with multi-expert deferral. In *International Conference on Machine Learning*, pp. 34738–34759. PMLR, 2024.

Alireza Mohammadshahi, Arshad Rafiq Shaikh, and Majid Yazdani. Routoo: Learning to route to large language models effectively. *arXiv preprint arXiv:2401.13979*, 2024.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pp. 5301–5310. PMLR, 2019.

Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

Marija Šakota, Maxime Peyrard, and Robert West. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 606–615, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*, 2023.

Gordon D Smith. *Numerical solution of partial differential equations: finite difference methods*. Oxford university press, 1985.

Andrew Staniforth and Jean Côté. Semi-lagrangian integration schemes for atmospheric models—a review. *Monthly weather review*, 119(9):2206–2223, 1991.

Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. *Advances in Neural Information Processing Systems*, 21, 2008.

Ali Taghibakhshi, Scott MacLachlan, Luke Olson, and Matthew West. Optimization-based algebraic multigrid coarsening using reinforcement learning. *Advances in neural information processing systems*, 34:12129–12140, 2021.

Fredi Tröltzsch. *Optimal control of partial differential equations: theory, methods, and applications*, volume 112. American Mathematical Soc., 2010.

Sebastian Tschiatschek, Adish Singla, and Andreas Krause. Selecting sequences of items via submodular maximization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Brandon Van Over, Bowen Li, Edwin KP Chong, and Ali Pezeshki. A performance bound for the greedy algorithm in a generalized class of string optimization problems. *arXiv preprint arXiv:2409.05020*, 2024.

Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019.

Zhi-Qin John Xu, Lulu Zhang, and Wei Cai. On understanding and overcoming spectral biases of deep neural network learning methods for solving pdes. *arXiv preprint arXiv:2501.09987*, 2025.

Zhilin You, Zhenli Xu, and Wei Cai. Mscalefno: Multi-scale fourier neural operator learning for oscillatory function spaces. *arXiv preprint arXiv:2412.20183*, 2024.

Enrui Zhang, Adar Kahana, Alena Kopaničáková, Eli Turkel, Rishikesh Ranade, Jay Pathak, and George Em Karniadakis. Blending neural operators and relaxation methods in pde numerical solvers. *Nature Machine Intelligence*, pp. 1–11, 2024.

Zhenliang Zhang, Edwin KP Chong, Ali Pezeshki, and William Moran. String submodular functions with curvature constraints. *IEEE Transactions on Automatic Control*, 61(3):601–616, 2015.

A PROOFS FOR SECTION 3

A.1 PROOF OF PROPOSITION A.1

Proposition A.1. *Any prefix monotonically non-increasing sequence supermodular function g is weakly supermodular with respect to all sequences $S \in \Omega^*$ with $\alpha(S) = 1$*

Proof. This proof is adapted from Liberty & Sviridenko (2017).

If g is sequence supermodular then,

$$\begin{aligned}
&g(S) - g(S \oplus S') \\
&= \sum_{i=1}^{|S'|} g(S \oplus (S'_1, \ldots, S'_{i-1})) - g(S \oplus (S'_1, \ldots, S'_i)) \\
&\overset{(a)}{\leq} \sum_{i=1}^{|S'|} g(S) - g(S \oplus S'_i) \\
&\leq |S'| \max_{i \in [|S'|]} g(S) - g(S \oplus S'_i)
\end{aligned}$$

(a) by supermodularity

□

A.2 PROOF OF THEOREM 3.1

Theorem 3.1. *Let $g : \Omega^* \to \mathbb{R}$ be a weakly supermodular function with respect to the optimal solution $O = \arg\min_{S \in \Omega^T} g(S)$ with a supermodularity ratio of $\alpha(O)$ and postfix monotonicity. Let the greedy solution of length T be S^T. If $\phi_T(\alpha) = \left(1 - \frac{1}{\alpha T}\right)^T$, then*

$$g(S^T) \leq \left(1 - \phi_T(\alpha(O))\right) g(O) + \phi_T(\alpha(O)) g(\emptyset)$$

Proof. This proof strategy is inspired by Streeter & Golovin (2008) and Liberty & Sviridenko (2017)

$$\begin{aligned}
g(S^t) - g(O) &\overset{(a)}{\leq} g(S^t) - g(S^t \oplus O) \\
&= \sum_{i=1}^{|O|} g(S^t \oplus (o_1, \ldots o_{i-1})) - g(S^t \oplus (o_1, \ldots, o_i)) \\
&\overset{(b)}{\leq} \alpha(O) \sum_{i=1}^{|O|} g(S^t) - g(S^t \oplus o_i) \\
&\leq \alpha(O)|O| \max_{i \in [|O|]} g(S^t) - g(S^t \oplus o_i) \\
&\leq \alpha(O) T g(S^t) - \alpha(O) T \min_{\omega \in \Omega} g(S^t \oplus \omega) \\
&= \alpha(O) T g(S^t) - \alpha(O) T g(S^{t+1})
\end{aligned}$$

(a) by postfix monotonicity, (b) by weak supermodularity.

After rearranging the inequality, we get:

$$\begin{aligned}
g(S^{t+1}) &\leq \frac{1}{\alpha(O)T} \left(g(O) + (\alpha(O)T - 1)\, g(S^t)\right) \\
&= \frac{1}{\alpha(O)T} g(O) + \left(1 - \frac{1}{\alpha(O)T}\right) g(S^t)
\end{aligned}$$

When recursively applying this inequality, we get:

$$
\begin{aligned}
g(S^T) &\leq \frac{1}{\alpha(O)T} g(O) \sum_{i=0}^{T-1} \left(1 - \frac{1}{\alpha(O)T}\right)^i + \left(1 - \frac{1}{\alpha(O)T}\right)^T g(\emptyset) \\
&= \left(1 - \left(1 - \frac{1}{\alpha(O)T}\right)^T\right) g(O) + \left(1 - \frac{1}{\alpha(O)T}\right)^T g(\emptyset)
\end{aligned}
$$

□

A.3 Proof of Proposition 3.2

Proposition 3.2. *Suppose that for all $j \in [K]$, the error propagation function $I - C_j \circ \mathcal{L}_h^a$ is ρ_j-Lipschitz continuous with $\rho_j < 1$, and that $(I - C_j \circ \mathcal{L}_h^a)(0_N) = 0_N$. Then, the function h is weakly supermodular with respect to the optimal solution O, with*

$$
\alpha(O) = \max\left\{\frac{1}{T - \sum_{i=1}^T \rho_{O_i}^2}, 1\right\}
$$

Proof. For brevity, we use the notation $(g_1 \circ \cdots \circ g_T)(x) = \circ_{t=1}^T g_t(x)$, where composition is applied from right to left so that g_T acts first. In this proof, we use a few properties of Lipschitz continuous functions:

- **Property 1:** If g is ρ-Lipschitz continuous and $g(0) = 0$, then $\|g(x)\|_2 = \|g(x) - 0\|_2 = \|g(x) - g(0)\|_2 \leq \rho\|x - 0\|_2 = \rho\|x\|_2$
- **Property 2:** If g_1 and g_2 are Lipschitz continuous functions with Lipschitz constants of ρ_1 and ρ_2 respectively, the Lipschitz constant of $g_1 + g_2$ and $g_1 - g_2$ is $\rho_1 + \rho_2$.
- **Property 3:** If g_1 and g_2 are Lipschitz continuous functions with Lipschitz constants of ρ_1 and ρ_2 respectively, the Lipschitz constant of $g_1 \circ g_2$ is $\rho_1\rho_2$.

In order to prove weakly-α-supermodularity, we must first prove prefix monotonicity.

Prefix monotonicity: Let $S \preceq S'$ where $S' = S \oplus N$.

$$
\begin{aligned}
h(S') &= \left\| \circ_{t=|S'|}^1 \left(I - C_{S'_t} \circ \mathcal{L}_h^a\right) \left(e_h^{(0)}\right) \right\|_2^2 \\
&= \left\| \circ_{t=|S'|}^{|S|+1} \left(I - C_{S'_t} \circ \mathcal{L}_h^a\right) \circ \circ_{t=|S|}^1 \left(I - C_{S'_t} \circ \mathcal{L}_h^a\right) \left(e_h^{(0)}\right) \right\|_2^2 \\
&\overset{(a)}{\leq} \left(\prod_{t=|S|+1}^{|S'|} \rho_{S'_t}^2 \right) \left\| \circ_{t=|S|}^1 \left(I - C_{S_t} \circ \mathcal{L}_h^a\right) \left(e_h^{(0)}\right) \right\|_2^2 \\
&\leq h(S)
\end{aligned}
$$

(a) by Property 1

Weak supermodularity: We will upper bound $\alpha(O)$ by providing an upper bound for $h(S) - h(S \oplus O)$ and a lower bound for $\sum_{i=1}^T h(S) - h(S \oplus O_i)$.

$$
h(S) - h(S \oplus O) \leq h(S)
$$

since h is always non-negative

To lower bound $\sum_{i=1}^{|O|} h(S) - h(S \oplus O_i)$

$$\sum_{i=1}^{|O|} h(S) - h(S \oplus O_i) = \sum_{i=1}^{|O|} \left\| \circ_{t=|S|}^{1} \left(I - C_{S_t} \circ \mathcal{L}_h^a\right) \left(e_h^{(0)}\right) \right\|_2^2 - \left\| \left(I - C_{O_i} \circ \mathcal{L}_h^a\right) \circ \circ_{t=|S|}^{1} \left(I - C_{S_t} \circ \mathcal{L}_h^a\right) \left(e_h^{(0)}\right) \right\|_2^2$$

$$\overset{(a)}{\geq} \sum_{i=1}^{|O|} \left\| \circ_{t=|S|}^{1} \left(I - C_{S_t} \circ \mathcal{L}_h^a\right) \left(e_h^{(0)}\right) \right\|_2^2 - \rho_{O_i}^2 \left\| \circ_{t=|S|}^{1} \left(I - C_{S_t} \circ \mathcal{L}_h^a\right) \left(e_h^{(0)}\right) \right\|_2^2$$

$$= \left\| \circ_{t=|S|}^{1} \left(I_N - C_{S_t} \circ \mathcal{L}_h^a\right) \left(e_h^{(0)}\right) \right\|_2^2 \left(T - \sum_{i=1}^{T} \rho_{O_i}^2\right)$$

(a) by Property 3

Finally,

$$\frac{h(S) - h(S \oplus O)}{T \max_i h(S) - h(S \oplus O_i)} \leq \max \left\{ \frac{\left\| \circ_{t=|S|}^{1} \left(I_N - C_{S_t} \circ \mathcal{L}_h^a\right) \left(e_h^{(0)}\right) \right\|_2^2}{\left\| \circ_{t=|S|}^{1} \left(I_N - C_{S_t} \circ \mathcal{L}_h^a\right) \left(e_h^{(0)}\right) \right\|_2^2 \left(T - \sum_{i=1}^{T} \rho_{O_i}^2\right)}, 1 \right\}$$

$$= \max \left\{ \frac{1}{T - \sum_{i=1}^{T} \rho_{O_i}^2}, 1 \right\}$$

□

A.4 Proof of Lemma A.2

Lemma A.2. *Let* $(I - C_j \mathcal{L}_h^a) = P \Lambda_j P^{-1}$ *where* P *is an orthogonal matrix and* $\Lambda_j = \operatorname{diag}(\lambda_{j1}, \ldots, \lambda_{jN})$. *If* $P^{-1} e_h^{(0)} = z$, *then the following equality holds:*

$$h(S) = \sum_{i=1}^{N} z_i^2 \prod_{j=1}^{K} \lambda_{ji}^{2m_j(S)} \tag{11}$$

where $m_j(S) = \sum_{t=1}^{|S|} \mathbf{1}_{S_t = j}$

Proof.

$$h(S) = \left\| \prod_{t=|S|}^{1} \left(I_N - C_{S_t} \mathcal{L}_h^a\right) e_h^{(0)} \right\|^2$$

$$= \left\| \prod_{t=|S|}^{1} \left(P \Lambda_{S_t} P^{-1}\right) e_h^{(0)} \right\|^2$$

$$= \left\| P \prod_{t=|S|}^{1} \Lambda_{S_t} P^{-1} e_h^{(0)} \right\|^2$$

$$= \left\| \prod_{t=|S|}^{1} \Lambda_{S_t} P^{-1} e_h^{(0)} \right\|^2$$

$$= \sum_{i=1}^{N} z_i^2 \prod_{t=T}^{1} \lambda_{S_t i}^2$$

$$= \sum_{i=1}^{N} z_i^2 \prod_{j=1}^{K} \lambda_{ji}^{2m_j(S)}$$

□

A.5 Proof of Proposition 3.3

Proposition 3.3. *Let $\|I_N - C_j\mathcal{L}_h^a\| \leq 1$ for all $j \in [K]$ and $(I - C_j\mathcal{L}_h^a) = P\Lambda_j P^{-1}$, where P is an orthogonal matrix. Then, h is supermodular.*

Proof. Let $S \preceq S'$ where $S' = S \oplus B$. By Lemma A.2,

$$h(S) = \sum_{i=1}^{N} z_i^2 \prod_{j=1}^{K} \lambda_{ji}^{2m_j(S)}$$

where $m_j(S) = \sum_{t=1}^{|S|} \mathbf{1}_{S_t=j}$ be the number of times a sequence S calls the solver j. Recall that h is considered sequence supermodular if $\forall S', S \in \Omega^*$ such that $S \preceq S'$, it holds that

$$h(S) - h(S \oplus \omega) \geq h(S') - h(S' \oplus \omega)$$

$$\begin{aligned} h(S) - h(S \oplus \omega) &= \sum_{i=1}^{N} z_i^2 \prod_{j=1}^{K} \lambda_{ji}^{2m_j(S)} - \sum_{i=1}^{N} z_i^2 \lambda_{\omega i}^2 \prod_{k=1}^{K} \lambda_{ki}^{2m_k(S)} \\ &= \sum_{i=1}^{N} \left(1 - \lambda_{\omega i}^2\right) z_i^2 \prod_{j=1}^{K} \lambda_{ji}^{2m_j(S)} \end{aligned}$$

Similarly,

$$\begin{aligned} h(S') - h(S' \oplus \omega) &= \sum_{i=1}^{N} \left(1 - \lambda_{\omega i}^2\right) z_i^2 \prod_{j=1}^{K} \lambda_{ji}^{2m_j(S')} \\ &\overset{(a)}{=} \sum_{i=1}^{N} \left(1 - \lambda_{\omega i}^2\right) z_i^2 \prod_{j=1}^{K} \lambda_{ji}^{2(m_j(S)+m_j(B))} \\ &\overset{(b)}{\leq} \sum_{i=1}^{N} \left(1 - \lambda_{\omega i}^2\right) z_i^2 \prod_{j=1}^{K} \lambda_{ji}^{2m_j(S)} \\ &= h(S) - h(S \oplus \omega) \end{aligned}$$

(a) Since $m_j(S') = \sum_{t=1}^{|S'|} \mathbf{1}_{S'_t=j} = \sum_{t=1}^{|S|} \mathbf{1}_{S'_t=j} + \sum_{t=|S|}^{|S'|} \mathbf{1}_{S'_t=j} = \sum_{t=1}^{|S|} \mathbf{1}_{S_t=k} + \sum_{t=1}^{|B|} 1_{B_t=j} = m_j(S) + m_j(B)$, (b) since $|\lambda_{ji}| \leq 1$ and $m_j(B) \geq 0$ for all $j \in [K]$, the additional factors $\lambda_{ji}^{2m_j(B)}$ lie in $[0, 1]$.

□

B Proofs for Section 4

B.1 Proof of Lemma B.1

Lemma B.1. *For any set of preconditioning functions $\mathcal{C}$, any discrete operator $\mathcal{L}_h^a$, any router r, any $a_h, f_h, u_h^{(t)}, u_h \in \mathcal{A} \times \mathcal{F} \times \mathcal{U} \times \mathcal{U}$, the following equality holds true:*

$$\begin{aligned} l_{\textit{route}}\left(r, a_h, f_h, u_h^{(t)}, u_h\right) = &\sum_{j=1}^{K}\sum_{k=1}^{K} \left\|(I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right)\right\|_2^2 \mathbf{1}_{k\neq j}\mathbf{1}_{r(a_h,f_h,u_h^{(t)})\neq j} \\ &- (K-2)\sum_{j=1}^{K} \left\|(I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right)\right\|_2^2 \end{aligned}$$

Proof. Note that $\sum_{j=1}^{K} \mathbf{1}_{r(a_h, f_h, u_h^{(t)}) \neq j} = K - 1$

$$
\begin{aligned}
l_{\text{route}}\left(r, a_h, f_h, u_h^{(t)}, u_h\right) &= \sum_{j=1}^{K} \left\| (I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 \mathbf{1}_{r(a_h, f_h, u_h^{(t)}) = j} \\
&= \sum_{j=1}^{K} \left\| (I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 - \sum_{j=1}^{K} \left\| (I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 \mathbf{1}_{r(a_h, f_h, u_h^{(t)}) \neq j} \\
&= \sum_{j=1}^{K} \left\| (I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 - \sum_{j=1}^{K} \left\| (I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 \mathbf{1}_{r(a_h, f_h, u_h^{(t)}) \neq j} \\
&\quad + (K-1) \sum_{k=1}^{K} \left\| (I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 - (K-1) \sum_{k=1}^{K} \left\| (I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 \\
&= \sum_{j=1}^{K} \left\| (I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 - \sum_{j=1}^{K} \left\| (I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 \mathbf{1}_{r(a_h, f_h, u_h^{(t)}) \neq j} \\
&\quad + \sum_{j=1}^{K} \mathbf{1}_{r(a_h, f_h, u_h^{(t)}) \neq j} \sum_{k=1}^{K} \left\| (I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 \\
&\quad - (K-1) \sum_{k=1}^{K} \left\| (I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 \\
&= \sum_{j=1}^{K} \left(\sum_{k=1}^{K} \left\| (I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 - \left\| (I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 \right) \mathbf{1}_{r(a_h, f_h, u_h^{(t)}) \neq j} \\
&\quad - (K-2) \sum_{k=1}^{K} \left\| (I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 \\
&= \sum_{j=1}^{K} \sum_{k=1}^{K} \left\| (I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2 \mathbf{1}_{k \neq j} \mathbf{1}_{r(a_h, f_h, u_h^{(t)}) \neq j} \\
&\quad - (K-2) \sum_{k=1}^{K} \left\| (I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right) \right\|_2^2
\end{aligned}
$$

□

B.2 Proof of Proposition B.2

Proposition B.2. *For any router r defined by $r(a, f, u^{(t)}) = \text{argmax}_{j \in [K]} g_j(a, f, u^{(t)})$, any $a_h \in \mathcal{A}$, $f_h \in \mathcal{F}$, and $u_h^{(t)}, u_h \in \mathcal{U}$, the routing loss l_{route} satisfies:*

$$
\log(2) l_{route}\left(r, a_h, f_h, u_h^{(t)}, u_h\right) \leq \Psi(\mathbf{g}, a_h, f_h, u_h^{(t)}, u_h)
$$

Proof. By Lemma B.1, we know that

$$
\begin{aligned}
\log(2) l_{\text{route}}\left(r, a_h, f_h, u_h^{(t)}, u_h\right) &= \log(2) \sum_{j=1}^{K}\sum_{k=1}^{K} \left\|(I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right)\right\|_2^2 \mathbf{1}_{k\neq j}\mathbf{1}_{r(a_h,f_h,u_h^{(t)})\neq j} \\
&\quad - \log(2)\,(K-2) \sum_{j=1}^{K} \left\|(I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right)\right\|_2^2 \\
&\leq \log(2) \sum_{j=1}^{K}\sum_{k=1}^{K} \left\|(I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right)\right\|_2^2 \mathbf{1}_{k\neq j}\mathbf{1}_{r(a_h,f_h,u_h^{(t)})\neq j} \\
&\overset{(a)}{\leq} -\sum_{j=1}^{K}\sum_{k=1}^{K} \left\|(I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right)\right\|_2^2 \mathbf{1}_{k\neq j} \log\left(\frac{\exp\left(g_j(a,f,u^{(t)})\right)}{\sum_{k=1}^{K}\exp\left(g_k(a,f,u^{(t)})\right)}\right) \\
&= \Psi(\mathbf{g}, a_h, f_h, u_h^{(t)}, u_h)
\end{aligned}
$$

(a) if $r(a_h, f_h, u_h^{(t)}) \neq j$, $\frac{\exp\left(g_j(a,f,u^{(t)})\right)}{\sum_{k=1}^{K}\exp\left(g_k(a,f,u^{(t)})\right)} < 0.5$ which implies that $-\log\left(\frac{\exp\left(g_j(a,f,u^{(t)})\right)}{\sum_{k=1}^{K}\exp\left(g_k(a,f,u^{(t)})\right)}\right) \geq \log(2)\mathbf{1}_{r(a_h,f_h,u_h^{(t)})\neq j}$

□

B.3 Proof of Theorem 4.1

Theorem 4.1. *Let $\tilde{c}_j(a_h, u_h^{(t)}, u_h) < \bar{E} < \infty$ for all $j \in [K]$. If there exists $j \in [K]$ such that $\tilde{c}_j(a_h, u_h^{(t)}, u_h) > E_{min} > 0$, then, for any collection of solvers $\{C_j\}_{j=1}^K$ and linear discrete operator $\mathcal{L}_h^a$, Ψ is Bayes consistent surrogate for l_{route}.*

Proof. For a given a_h, f_h, let u_h be $\mathcal{G}_h(a_h, f_h)$ where $\mathcal{G}_h$ denotes the solution operator acting on the grid G_h. Furthermore, let's consider routers of the form

$$r(a, f, u^{(t)}) = \text{argmax}_{j\in[K]} g_j(a, f, u^{(t)})$$

For a given $a_h, f_h, u_h^{(t)} \in \mathcal{A} \times \mathcal{F} \times \mathcal{U}$, let the optimal loss under l_{route} be $l^*_{\text{route}}\left(a_h, f_h, u_h^{(t)}\right) = \inf_{\tilde{r}} l_{\text{route}}\left(\tilde{r}, a_h, f_h, u_h^{(t)}, \mathcal{G}_h(a_h, f_h)\right)$. Similarly, let the optimal loss under Ψ be $\Psi^*\left(a_h, f_h, u_h^{(t)}\right) = \inf_{\tilde{\mathbf{g}}} \Psi\left(\tilde{\mathbf{g}}, a_h, f_h, u_h^{(t)}, \mathcal{G}_h(a_h, f_h)\right)$. Let $B_j(a_h, f_h, u_h^{(t)}) = \sum_{k=1}^{K}\left\|(I - C_k \circ \mathcal{L}_h^a)\left(\mathcal{G}_h(a_h, f_h) - u_h^{(t)}\right)\right\|_2^2 \mathbf{1}_{k\neq j}$.

$$
\begin{aligned}
&l_{\text{route}}\left(r, a_h, f_h, u_h^{(t)}, \mathcal{G}_h(a_h, f_h)\right) - l^*_{\text{route}}\left(a_h, f_h, u_h^{(t)}\right) \\
&\overset{(a)}{=} \sum_{j=1}^{K}\sum_{k=1}^{K} \left\|(I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right)\right\|_2^2 \mathbf{1}_{k\neq j}\mathbf{1}_{r(a_h,f_h,u_h^{(t)})\neq j} - (K-2)\sum_{j=1}^{K} \left\|(I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right)\right\|_2^2 \\
&\quad - \inf_{\tilde{r}} \sum_{j=1}^{K}\sum_{k=1}^{K} \left\|(I - C_k \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right)\right\|_2^2 \mathbf{1}_{k\neq j}\mathbf{1}_{\tilde{r}(a_h,f_h,u_h^{(t)})\neq j} + (K-2)\sum_{j=1}^{K} \left\|(I - C_j \circ \mathcal{L}_h^a)\left(u_h - u_h^{(t)}\right)\right\|_2^2 \\
&= \sum_{j=1}^{K} B_j(a_h, f_h, u_h^{(t)})\mathbf{1}_{r(a_h,f_h,u_h^{(t)})\neq j} - \inf_{\tilde{r}} \sum_{j=1}^{K} B_j(a_h, f_h, u_h^{(t)})\mathbf{1}_{\tilde{r}(a_h,f_h,u_h^{(t)})\neq j} \\
&= \sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)}) \left(\sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})}\mathbf{1}_{r(a_h,f_h,u_h^{(t)})\neq j} - \inf_{\tilde{r}} \sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})}\mathbf{1}_{\tilde{r}(a_h,f_h,u_h^{(t)})\neq j}\right)
\end{aligned}
$$

(a) by Lemma B.1

Let $\mathcal{X} = \mathcal{A} \times \mathcal{F} \times \mathcal{U}$ and $\mathcal{Y} = [K]$. Let $\mathcal{P}_{\mathcal{X}}$ denote the degenerate distribution suported at the point $(a_h, f_h, u_h^{(t)})$. We define the conditional distribtion - $P(Y = j \mid X = (a_h, f_h, u_h^{(t)})) = \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})}$ for $j \in [K]$. The risk and optimal risk of $0-1$ loss under this distribution can be written as:

$$\mathcal{R}_{0-1}(r) = \sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})} \mathbf{1}_{r(a_h, f_h, u_h^{(t)}) \neq j}$$

$$\mathcal{R}^*_{0-1} = \inf_r \sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})} \mathbf{1}_{r(a_h, f_h, u_h^{(t)}) \neq j}$$

If $r(a_h, f_h, u_h^{(t)}) = \arg\max_{j \in [k]} g_j(a_h, f_h, u_h^{(t)})$ for all $x \in \mathcal{X}$, then the he risk and optimal risk of cross entropy loss ($l_{ce}(\mathbf{g}, x, y) - \log\left(\frac{\exp(g_y(x))}{\sum_{k=1}^{K} \exp(g_k(x))}\right)$) under this distribution can be written as:

$$\mathcal{R}_{ce}(\mathbf{g}) = -\sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})} \log\left(\frac{\exp\left(g_j(a_h, f_h, u_h^{(t)})\right)}{\sum_{k=1}^{K} \exp\left(g_k(a_h, f_h, u_h^{(t)})\right)}\right)$$

$$\mathcal{R}^*_{ce} = \inf_{\mathbf{g}} -\sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})} \log\left(\frac{\exp\left(g_j(a_h, f_h, u_h^{(t)})\right)}{\sum_{k=1}^{K} \exp\left(g_k(a_h, f_h, u_h^{(t)})\right)}\right)$$

From Theorem 3.1 of (Mao et al., 2023), $\mathcal{R}_{0-1}(r) - \mathcal{R}^*_{0-1} \leq \Gamma^{-1}\left(\mathcal{R}_{ce}(\mathbf{g}) - \mathcal{R}^*_{ce}\right)$ if $r(a_h, f_h, u_h^{(t)}) = \arg\max_{j \in [k]} g_j(a_h, f_h, u_h^{(t)})$ where $\Gamma(z) = \frac{1+z}{2}\log(1+z) + \frac{1-z}{2}\log(1-z)$. Then,

$$
\begin{aligned}
&l_{\text{route}}\left(r, a_h, f_h, u_h^{(t)}, \mathcal{G}_h\left(a_h, f_h\right)\right) - l^*_{\text{route}}\left(a_h, f_h, u_h^{(t)}\right) \\
&= \sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)}) \left(\sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})} \mathbf{1}_{r(a_h, f_h, u_h^{(t)}) \neq j} - \inf_{\tilde{r}} \sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})} \mathbf{1}_{\tilde{r}(a_h, f_h, u_h^{(t)}) \neq j} \right) \\
&\leq \sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)}) \Gamma^{-1} \left(-\sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})} \log \left(\frac{\exp\left(g_j(a_h, f_h, u_h^{(t)})\right)}{\sum_{k=1}^{K} \exp\left(g_k(a_h, f_h, u_h^{(t)})\right)} \right) \right. \\
&\qquad \left. - \inf_{\mathbf{g}} - \sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})} \log \left(\frac{\exp\left(g_j(a_h, f_h, u_h^{(t)})\right)}{\sum_{k=1}^{K} \exp\left(g_k(a_h, f_h, u_h^{(t)})\right)} \right) \right) \\
&\overset{(a)}{\leq} \bar{E} K (K-1) \Gamma^{-1} \left(-\sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})} \log \left(\frac{\exp\left(g_j(a_h, f_h, u_h^{(t)})\right)}{\sum_{k=1}^{K} \exp\left(g_k(a_h, f_h, u_h^{(t)})\right)} \right) \right. \\
&\qquad \left. - \inf_{\mathbf{g}} - \sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{\sum_{k=1}^{K} B_k(a_h, f_h, u_h^{(t)})} \log \left(\frac{\exp\left(g_j(a_h, f_h, u_h^{(t)})\right)}{\sum_{k=1}^{K} \exp\left(g_k(a_h, f_h, u_h^{(t)})\right)} \right) \right) \\
&\overset{(b)}{\leq} \bar{E} K (K-1) \Gamma^{-1} \left(-\sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{(K-1) E_{min}} \log \left(\frac{\exp\left(g_j(a_h, f_h, u_h^{(t)})\right)}{\sum_{k=1}^{K} \exp\left(g_k(a_h, f_h, u_h^{(t)})\right)} \right) \right. \\
&\qquad \left. - \inf_{\mathbf{g}} - \sum_{j=1}^{K} \frac{B_j(a_h, f_h, u_h^{(t)})}{(K-1) E_{min}} \log \left(\frac{\exp\left(g_j(a_h, f_h, u_h^{(t)})\right)}{\sum_{k=1}^{K} \exp\left(g_k(a_h, f_h, u_h^{(t)})\right)} \right) \right) \\
&= \bar{E} K (K-1) \Gamma^{-1} \left(\frac{\Psi\left(\mathbf{g}, a_h, f_h, u_h^{(t)}, \mathcal{G}_h\left(a_h, f_h\right)\right) - \Psi^*\left(a_h, f_h, u_h^{(t)}\right)}{(K-1) E_{min}} \right)
\end{aligned}
$$

(a) since $\left\| (I - C_j \circ \mathcal{L}_h^a)\left(e_h^{(t)}\right) \right\|_2^2 < \bar{E}$ for all $j \in [K]$, (b) since Γ^{-1} is non-decreasing and $\exists j \in [K]$ such that $\left\| (I - C_j \circ \mathcal{L}_h^a)\left(e_h^{(t)}\right) \right\|_2^2 > E_{min}$

Finally,

$$
\begin{aligned}
&\lim_{n \to \infty} \mathcal{R}_{\text{route}}\left(r_n\right) - \mathcal{R}^*_{\text{route}} \\
&\overset{(a)}{=} \lim_{n \to \infty} \mathbb{E}_{a_h, f_h \sim \mathcal{P}_{\mathcal{A} \times \mathcal{F}}} \left[l_{\text{route}}\left(r_n, a_h, f_h, u_h^{(t)}, \mathcal{G}_h\left(a_h, f_h\right)\right) - l^*_{\text{route}}\left(a_h, f_h, u_h^{(t)}\right) \right] \\
&\leq \lim_{n \to \infty} \mathbb{E}_{a_h, f_h \sim \mathcal{P}_{\mathcal{A} \times \mathcal{F}}} \left[\bar{E} K (K-1) \Gamma^{-1} \left(\frac{\Psi\left(\tilde{\mathbf{g}}, a_h, f_h, u_h^{(t)}, \mathcal{G}_h\left(a_h, f_h\right)\right) - \Psi^*\left(a_h, f_h, u_h^{(t)}\right)}{(K-1) E_{min}} \right) \right] \\
&\overset{(b)}{\leq} \lim_{n \to \infty} \bar{E} K (K-1) \Gamma^{-1} \left(\frac{\mathbb{E}_{a_h, f_h \sim \mathcal{P}_{\mathcal{A} \times \mathcal{F}}} \left[\Psi\left(\mathbf{g}_n, a_h, f_h, u_h^{(t)}, \mathcal{G}_h\left(a_h, f_h\right)\right) - \Psi^*\left(a_h, f_h, u_h^{(t)}\right) \right]}{(K-1) E_{min}} \right)
\end{aligned}
$$

$$
\begin{aligned}
&= \lim_{n\to\infty} \bar{E}K\left(K-1\right)\Gamma^{-1}\left(\frac{\mathcal{R}_\Psi\left(\mathbf{g}_n\right)-\mathcal{R}^*_\Psi}{(K-1)E_{min}}\right)\\
&\overset{(c)}{=} \bar{E}K\left(K-1\right)\Gamma^{-1}\left(\frac{\lim_{n\to\infty}\mathcal{R}_\Psi\left(\mathbf{g}_n\right)-\mathcal{R}^*_\Psi}{(K-1)E_{min}}\right)\\
&= \bar{E}K\left(K-1\right)\Gamma^{-1}\left(0\right)\\
&\overset{(d)}{=} 0
\end{aligned}
$$

(a) $\mathcal{R}^*_{\text{route}} = \mathbb{E}_{a_h,f_h\sim\mathcal{P}_{\mathcal{A}\times\mathcal{F}}}\left[l^*_{\text{route}}\left(a_h,f_h,u_h^{(t)}\right)\right]$ since the infimum is taken over all measurable functions, (b) by Jensen's inequality since Γ^{-1} is concave, (c) by continuity of Γ^{-1} at 0, (d) $\Gamma^{-1}(0)=0$

□

C COST-AWARE EXTENSION

The horizon in Theorem 3.1 counts routing decisions. This formulation is appropriate for deriving wall-clock guarantees when all actions have the same cost, because the number of routing decisions are proportional to wall-clock time. When action costs differ–for example, a neural correction is often more expensive than one Jacobi iteration–a routing decision count guarantee does not directly imply a wall-clock guarantee. In experiments, we mitigate this discrepancy by grouping multiple Jacobi iterations into a macro-action whose measured cost approximately matches that of one neural correction. More generally, heterogeneous costs can be handled through a cost-aware formulation whose objective is to identify the best solver sequence with total cost at most B.

Formally, suppose each action $j \le K$ has a deterministic cost $c_j > 0$. In keeping with the sequence convention used throughout the paper, for $S = (S_1, \ldots, S_{|S|}) \in \Omega^*$, we define

$$
c(S) := \sum_{t=1}^{|S|} c_{S_t}, \qquad \mathcal{F}(B) := \{S \in \Omega^* : c(S) \le B\}.
$$

S_1 is still applied first and concatenation $S \oplus j$ appends action j to the current solver trajectory. For a total budget $B > 0$, let

$$
O_B \in \arg\min_{S\in\mathcal{F}(B)} g(S)
$$

be a budget-optimal sequence. We denote $\Delta_j(S) := g(S) - g(S \oplus j)$ for the one-step decrease. The natural cost-aware greedy rule is the marginal-decrease per unit cost rule or

$$
j_t \in \arg\max_{j\le K} \frac{\Delta_j(S^t)}{c_j}, \qquad S^{t+1} = S^t \oplus j_t, \tag{12}
$$

This greedy algorithm continues as long as $c(S^t) < B$. The action selected at such a prefix is completed even if it makes $c(S^{t+1})$ exceed B, after which the algorithm stops. Under prefix monotonicity, the numerator is nonnegative. Notice that minimizing the next error $g(S \oplus j)$ and maximizing Equation (12) are equivalent only when the actions are equivalent in cost. This property is leveraged in experiments.

This extension to heterogeneous costs requires minor changes to the main paper's weak-supermodularity definition. Specifically, we say that g is *budget-range weakly supermodular with respect to* O_B with ratio $\alpha(O_B)$ if Equation (7) holds with $S' = O_B$ for every prefix S satisfying $c(S) < B$. The original length-based condition, which requires the prefixes to satisfy $|S| \le |O_B|$, is insufficient here: within the same budget, a greedy polict may select more inexpensive actions than the number of actions in O_B, producing sequences longer than O_B.

With these details established, we can now characterize the suboptimality of a greedy algorithm for optimizing cost-aware solver selection in Theorem C.1.

Theorem C.1 (Cost-aware greedy with one-action budget slack)**.** *Let $g : \Omega^* \to \mathbb{R}$ be prefix and postfix monotonically non-increasing and budget-range weakly supermodular with respect to O_B,*

with ratio $\alpha(O_B)$. Suppose the action costs are fixed and $c_{\max} := \max_{j\in\Omega} c_j \le B$. Let S_{B+} be the sequence returned by Equation (12) *under the stopping rule above. Then $B \le c(S_{B+}) < B + c_{\max}$ and*

$$g(\emptyset) - g(S_{B+}) \ge \left(1 - e^{-1/\alpha(O_B)}\right)\big(g(\emptyset) - g(O_B)\big). \tag{13}$$

Proof. Let $c_t = c_{j_t}$ denote the cost of the greedy action selected at step t. For every greedy prefix S^t with $c(S^t) < B$, we have

$$\begin{aligned}
g(S^t) - g(O_B) &\overset{(a)}{\le} g(S^t) - g(S^t \oplus O_B) \\
&\overset{(b)}{\le} \alpha(O_B) \sum_{i=1}^{|O_B|} \big(g(S^t) - g(S^t \oplus (O_B)_i)\big) \\
&= \alpha(O_B) \sum_{i=1}^{|O_B|} c_{(O_B)_i} \frac{g(S^t) - g(S^t \oplus (O_B)_i)}{c_{(O_B)_i}} \\
&\le \alpha(O_B) c(O_B) \max_{j\in\Omega} \frac{g(S^t) - g(S^t \oplus j)}{c_j} \\
&\overset{(c)}{\le} \alpha(O_B) B \frac{g(S^t) - g(S^{t+1})}{c_t}.
\end{aligned}$$

(a) follows from postfix monotonicity, (b) from budget-range weak supermodularity, and (c) from $c(O_B) \le B$ and the density-greedy rule in Equation (12).

After rearranging the inequality, we get

$$g(S^{t+1}) - g(O_B) \le \left(1 - \frac{c_t}{\alpha(O_B)B}\right)\big(g(S^t) - g(O_B)\big).$$

Let m be the first index for which $c(S^m) \ge B$. Recursively applying the preceding inequality gives

$$\begin{aligned}
g(S^m) - g(O_B) &\le \prod_{t=0}^{m-1} \left(1 - \frac{c_t}{\alpha(O_B)B}\right)(g(\emptyset) - g(O_B)) \\
&\le \exp\left(-\frac{\sum_{t=0}^{m-1} c_t}{\alpha(O_B)B}\right)(g(\emptyset) - g(O_B)) \\
&\le e^{-1/\alpha(O_B)}\,(g(\emptyset) - g(O_B)),
\end{aligned}$$

where the second inequality uses $1 - x \le e^{-x}$ and the last uses $c(S^m) = \sum_{t=0}^{m-1} c_t \ge B$. Rearranging yields Equation (13). Finally, $c(S^{m-1}) < B$ and the last action costs at most $c_{\max}$, so $B \le c(S^m) < B + c_{\max}$. □

If all actions cost c and $B = Tc$, the analysis above is equivalent to the discrete action-count recurrence of Theorem 3.1, which yields $1 - (1 - 1/\lceil\alpha(O_B)T\rceil)^T$.

D Experimental Setup

D.1 Data and reference solutions

All training, development, and test splits use disjoint random seeds. Test forcings follow the zero-mean hierarchical Gaussian random-field distribution described in Section 6. For a frequency $k \neq 0$, its Fourier coefficient has variance $\alpha(4\pi^2\|k\|_2^2 + \beta)^{-\gamma}$, where $\alpha \sim \mathrm{LogUniform}(10^{-2}, 10^2)$, $\beta \sim \mathrm{LogUniform}(10^{-1}, 10^3)$, and γ is uniform on $\{0.5, 1, 1.5, 2, 2.5, 3, 4\}$. Hermitian symmetry produces a real field and the zero mode is removed. Neural-corrector training uses an equal mixture of four input distributions. "GRF" denotes the forcing distribution above. "Rough" draws a GRF error e and uses $\mathcal{L}_h^a e$ as the residual, which emphasizes the high-frequency content produced

during an iterative solve. "White" draws independent standard-normal grid values before demeaning and spectral restriction, giving an approximately flat in-band spectrum. "Tilted" draws complex Gaussian Fourier coefficients and scales them by

$$\left(a_x^2 k_x^2 + a_x^{-2} k_y^2 + 1\right)^{-q/2}, \qquad q \sim \mathrm{Uniform}(-1, 3), \quad \log a_x \sim \mathrm{Uniform}(-1, 1),$$

thereby randomizing both spectral slope and directional anisotropy. These additional distributions expose the correctors to residuals beyond the forcing law used for final evaluation, allowing the neural correctors to behave more like an iterative solver. Reference solutions are computed from the same discrete PDE operator used in the rollout.

D.2 Neural corrections

The global correction is a linear Fourier-trunk DeepONet. Its trunk is a fixed orthonormal Fourier basis, and its branch is fit by least squares to map a restricted, normalized residual to the coefficients of its exact error correction. At inference time, the predicted correction is multiplied by the input residual norm, making the map scale equivariant and explicitly zero preserving. The global model in the growing-ensemble experiments, denoted $\mathrm{s32}$, has a 32×32 sensor grid and is fit using $16{,}384$ training and $1{,}024$ validation pairs. We also train a stronger 64×64-sensor model using $64{,}000$ training and $2{,}000$ validation pairs; this model is used in the paired-solver experiments and as an external pairwise control for the ensemble study. For anisotropic diffusion, the sensor layout retains full resolution in the weak-diffusion direction. A separate model is fit for each PDE. Because the branch is obtained by a direct least-squares solve rather than by gradient descent, these models have no optimizer, learning rate, training epochs, or SGD batch size. Training pairs are generated in chunks of $1{,}024$ only to control memory use; this chunk size is not an optimization hyperparameter.

The local correction is a learned periodic 3×3 convolution. It is trained on the residual left after the $\mathrm{s32}$ global correction. We generate 512 such residuals from the same four-component mixture (GRF, rough, white, and tilted spectra), solve the discrete PDE for their exact remaining errors, normalize each pair by the residual norm, and fit the kernel by ridge least squares. An independent set of 32 GRF residuals is used for validation. Specifically, the nine kernel coefficients are obtained by a direct ridge least-squares solve with coefficient 10^{-8}, not by gradient descent.

D.3 Router training

The learned routers are two layer MLPs with ReLU activations. The input features for the routers include the log relative residual, its change over the preceding macro-action and an exponential moving average of that change, the decision index, the previous-action indicator, per-action counts, the number of neural calls, and the number of operations since the last neural call.

At each decision point, the training oracle applies every available macro-action to a cloned state and labels the example using the true next-error values in Equation (10). Round zero uses oracle rollouts with 0.1 probability of random exploration. Subsequent DAgger rounds roll out the current router, relabel the states it visits with the oracle, and refit on the accumulated dataset. The paired-solver study uses 128 instances per round and two DAgger rounds. The growing-ensemble study uses 256 instances per round and four DAgger rounds; within that study, every constituent pair, historical control, and growing ensemble receives this same training strength.

D.4 Cost matching for paired and ensemble experiments

We measure the complete cost c_j of every primitive action on a common single-thread CPU, including the update and subsequent residual refresh. In a paired experiment, the reference unit is one call to that pair's global neural correction. A classical decision executes

$$m_j = \max\{1, \mathrm{round}(c_{\mathrm{NO}}/c_j)\}$$

consecutive classical iterations, whereas a neural decision executes one correction. The router therefore compares approximately equal-cost macro-actions rather than one expensive neural call with one inexpensive sweep. Only the classical updates are grouped by this rule. Each neural correction is one indivisible action: in particular, the cheaper local convolution is not repeated to match the $\mathrm{s32}$ cost.

Table 3: Router hyperparameters. A DAgger round denotes an additional router-policy rollout and oracle relabeling pass after the initial exploratory oracle round.

Hyperparameter	Paired study	Growing ensembles
Feature variants	Scalar	Scalar; pooled residual
Hidden width	64	64 scalar; 32 pooled
Instances per round	128	256
DAgger rounds	2	4
Epochs per fit	150	150
Optimizer	Adam	Adam
Initial learning rate	2×10^{-3}	2×10^{-3}
Schedule	Cosine decay	Cosine decay
Batch size	2,048	2,048
Training error cutoff	10^{-9}	10^{-9}
Maximum query epochs	3,000	512

Table 4: Measured complete-action costs for the $N = 128$ experiments. Costs include the update and subsequent residual refresh. For growing-ensemble actions, m is the number of primitive applications in one macro-action and the last column is mc_j. The s32 cost is the matching reference. The full global model is used only by paired controls, so no growing-ensemble macro count is assigned to it.

Equation	Action	Primitive cost (ms)	m	Macro cost (ms)
Poisson	WJacobi(.67)	0.147	6	0.884
	GS	0.443	2	0.886
	SOR(1.5)	0.448	2	0.896
	MG	3.614	1	3.614
	Global s32	0.897	1	0.897
	Local 3×3	0.370	1	0.370
	Global full (paired control)	5.115	–	–
ConvDiff	WJacobi(.67)	0.208	4	0.833
	GS	0.502	2	1.003
	SOR(1.5)	0.502	2	1.003
	MG	3.956	1	3.956
	Global s32	0.915	1	0.915
	Local 3×3	0.426	1	0.426
	Global full (paired control)	5.254	–	–
AnisoDiff	WJacobi(.67)	0.164	5	0.820
	GS	0.464	2	0.928
	SOR(1.5)	0.465	2	0.930
	MG	3.637	1	3.637
	Global s32	0.895	1	0.895
	Local 3×3	0.386	1	0.386
	Global full (paired control)	5.745	–	–

For each growing-ensemble PDE, all action sets share one cost table and one set of frozen macro-action sizes. The reference unit is the cost of the s32 global correction; every neural action is called once, and every classical action executes the rounded number of iterations implied by the shared unit. The same action consequently has identical semantics in every pair, historical control, and ensemble size. Costs and macro sizes are frozen before final cases are evaluated. Selected-path times charge only executed actions and router overhead.

Table 4 reports the resulting 128×128 grid calibration. The primitive and effective macro costs are shown separately. Thus, for example, the Poisson weighted-Jacobi macro costs $6(0.147) = 0.884$ ms, close to the 0.897 ms cost of one s32 call. Multigrid remains substantially more expensive than the reference unit because a V-cycle is indivisible; the cost-adjusted score accounts for this mismatch. The local correction is deliberately cheaper than the unit but is also an indivisible one-call action. The full global model is not a member of the growing ensemble and is reported separately as the neural correction used by the paired controls.

Table 5: Frozen action order for the $N = 128$ growing-ensemble experiments. Each size-K action set is the length-K prefix of the corresponding row.

Equation	a_1	a_2	a_3	a_4	a_5	a_6
Poisson	WJacobi(.67)	Local 3×3	Global s32	GS	SOR(1.5)	MG
ConvDiff	WJacobi(.67)	Local 3×3	Global s32	GS	SOR(1.5)	MG
AnisoDiff	WJacobi(.67)	Local 3×3	GS	SOR(1.5)	Global s32	MG

Table 6: Post-selected HINTS results at relative solution error 10^{-8} on the 64 shared $N = 128$ test cases. Each cell reports the selected period τ and mean selected-action replay time in milliseconds (standard error in parentheses). The period is the best of $\{5, 10, 25, 50\}$ on these same cases, so these are optimistic HINTS results rather than held-out cadence estimates.

Classical solver	Poisson	ConvDiff	AnisoDiff
Jacobi	50; 785.88 (92.58)	50; 1190.68 (125.93)	50; 353.62 (60.96)
WJacobi(.67)	10; 17.34 (0.36)	10; 18.23 (0.41)	25; 23.16 (0.15)
GS	5; 28.88 (0.38)	5; 30.93 (0.37)	25; 69.99 (1.43)
SymGS	5; 30.60 (0.03)	5; 31.64 (0.21)	5; 29.02 (0.59)
SOR(1.5)	10; 36.74 (0.50)	10; 43.53 (0.58)	10; 42.50 (0.62)
MG	5; 23.44 (0.21)	5; 26.37 (0.24)	5; 57.12 (2.06)

D.5 Action ordering for the growing ensembles

The growing-ensemble experiment requires a nested sequence of action sets so that increasing K adds one action without changing the actions already available. We determine this order on 16 independent samples before router training and final evaluation. Each action is first evaluated as a singleton. Their individual strength is ranked by number of failures, mean successful modeled time, and median terminal relative error. We initialize the pairwise set with the weakest numerical action and the weakest neural action, then append the remaining actions from weaker to stronger according to this ranking. This helps answer the question of whether progressively adding stronger and more complementary actions improves the learned policy; it is not an order chosen to make final performance monotone. All prefixes, including negative ones, are retained in the final evaluation.

Table 5 gives the resulting frozen orders. For an order $(a_1, \ldots, a_6)$, the size-K ensemble is $\mathcal{A}_K = \{a_1, \ldots, a_K\}$. Poisson and convection–diffusion yield the same order, whereas anisotropic diffusion promotes GS and SOR ahead of the global correction because their ranking differs on that PDE.

D.6 Selection of the HINTS cadence

HINTS replaces a classical update with a neural correction at a fixed period τ. Its performance is sensitive to this period: small τ can waste expensive neural calls or reintroduce error, whereas large τ can delay a useful neural correction. For each PDE and classical solver in the paired study, we evaluate $\tau \in \{5, 10, 25, 50\}$ using the same neural model, stopping rule, action costs, and test instances as the learned router. Table 1 reports the period with the lowest mean selected-action time to the target tolerance, retaining failures. Table 6 reports the selected cadence and its resulting mean time.

E Assumption Satisfaction in the Experiments

This section evaluates how closely the experimental macro-actions satisfy the sufficient conditions used in Proposition 3.2 and Theorem 3.1. The theorem requires zero-preserving strict contractions in the stated norm for the weak-supermodularity argument, as well as postfix monotonicity for the greedy approximation guarantee. The measurements below characterize the experimental maps but do not replace these uniform assumptions by a proof.

Table 7: Contraction diagnostics for finalized $N = 128$ macro-actions in the mean-zero Euclidean solution-error norm. "Exact" denotes Fourier diagonalization of the deployed linear map, "SV" a matrix-free largest singular-value estimate, and "LB" a sampled lower bound rather than an upper certificate. Values greater than one falsify strict contraction.

PDE	WJ(.67)	GS	SOR(1.5)	MG	Global neural	Local convolution
Poisson	0.997581 (exact)	1.019231 (SV)	1.369303 (SV)	≥ 0.036064 (LB)	≥ 1 (exact witness)	1.064909 (exact)
ConvDiff	0.998390 (exact)	1.022788 (SV)	1.475764 (SV)	≥ 0.024636 (LB)	≥ 1 (exact witness)	1.016401 (exact)
AnisoDiff	0.999936 (exact)	1.050241 (SV)	1.459752 (SV)	≥ 0.853678 (LB)	≥ 1 (exact witness)	35.822662 (exact)

Table 8: Additional contraction diagnostics for actions appearing in the finalized pairwise experiments but not in the growing action pool. Parentheses give the frozen number of primitive iterations per routed macro.

PDE	Undamped Jacobi	SymGS
Poisson	1.000000 (44; exact)	0.985629 (8; SV)
ConvDiff	1.000000 (27; exact)	0.989307 (7; SV)
AnisoDiff	1.000000 (32; exact)	1.048533 (7; SV)

Zero preservation. Every finalized macro-action mapped the zero error to zero in the exact-source $N = 128$ audit. For the classical stationary iterations and local convolution, this also follows algebraically from linearity and the absence of an additive bias. The global neural correction explicitly returns a correction scaled by the restricted-residual norm and therefore also maps a zero residual to zero.

Contraction diagnostics. Table 7 reports macro-action norms in the same Euclidean, mean-zero state space used by the objective. Weighted Jacobi and the local convolution are translation invariant, so their norms are computed exactly from their Fourier multipliers. For the non-normal GS and SOR maps, we report matrix-free largest-singular-value estimates for the exact deployed macros. The values above one are direct numerical witnesses against strict contraction. The MG entries are only lower bounds obtained from structured Fourier and seeded random probes, and therefore do not certify contraction. The global neural map is nonlinear; moreover, because its correction is band-limited, every excluded Fourier error mode is unchanged. Its global Lipschitz constant is consequently at least one, independently of sampling.

The pairwise experiments additionally route to undamped Jacobi and SymGS. Their independently frozen macro sizes and exact-source $N = 128$ diagnostics are shown in Table 8. Undamped periodic Jacobi has exact norm one for all three PDEs, and is therefore nonexpansive but not a strict contraction. SymGS is contractive for the Poisson and ConvDiff pairwise macros, whereas its anisotropic macro has norm greater than one.

These results illustrate why spectral radii or average error ratios are not sufficient conditions for Proposition 3.2. In particular, lexicographic GS and SOR are non-normal: their asymptotic spectral behavior can be stable even though their one-macro Euclidean operator norm exceeds one. Hence the strict contraction premise is not uniformly satisfied by the complete experimental action sets. WJ is individually contractive, although its anisotropic contraction factor is very close to one.

Postfix monotonicity Postfix monotonicity requires

$$h(S' \oplus S) \leq h(S)$$

for every prepended sequence S', suffix S, and initial state. The latter does not follow from zero preservation alone and can fail for noncommuting actions even when the individual maps contract. We tested every ordered pair of one-macro prepended and suffix actions on eight structured Fourier probes and 64 seeded random mean-zero probes. This produced $36 \times 72 = 2592$ comparisons per PDE. At least one sampled postfix violation occurred for 13 of 36 ordered pairs for Poisson, 13 of 36 for ConvDiff, and 14 of 36 for AnisoDiff. These witnesses disprove uniform postfix monotonicity for the corresponding pairs; absence of a sampled violation would not prove the property.

The experiments nevertheless exhibit useful approximate postfix behavior on representative states. To quantify it without conflating it with the uniform theorem assumption, define an empirical (μ, δ)

Table 9: Distributional postfix diagnostics over all ordered one-macro action pairs and the declared $N = 128$ probe set. Quantiles are for the norm ratio r; the slack for the squared-error objective is $\mu = r^2$.

PDE	Median r	95th percentile	99th percentile	Fraction $r \leq 1.05$
Poisson	0.4666	0.9976	1.0649	98.96%
ConvDiff	0.4706	0.9984	1.0081	99.38%
AnisoDiff	0.7496	11.7255	24.4790	83.22%

condition under a declared probe distribution by

$$\Pr[h(S' \oplus S) \leq \mu h(S)] \geq 1 - \delta. \tag{14}$$

If $r = \|E_S E_{S'} e\|_2 / \|E_S e\|_2$, the corresponding squared-objective slack is $\mu = r^2$. Table 9 summarizes all 2592 ratios per PDE.

For ConvDiff, the empirical 99th-percentile squared-objective slack is $\mu \approx 1.0081^2 \approx 1.0163$; for Poisson it is $\mu \approx 1.0649^2 \approx 1.1340$. Thus Poisson and ConvDiff are close to postfix monotone for most tested action–state combinations, despite rare structured counterexamples. Indeed, all comparisons on the 64 random probes were non-increasing for these two PDEs; the violations arose from deliberately included structured Fourier directions. AnisoDiff behaves differently: only 83.22% of all comparisons have $r \leq 1.05$, and its large upper quantiles persist even on random probes. A close-to-one high-probability postfix description is therefore not supported for the complete AnisoDiff action set.

The audit supports two conclusions. First, zero preservation—one of the structural premises of Proposition 3.2—holds for all finalized actions. Second, although strict contraction and uniform postfix monotonicity do not hold globally, their empirical violations are mild for most tested Poisson and convection–diffusion action–state pairs; anisotropic diffusion exhibits more substantial violations.

F Additional Experimental Results

We present additional visualizations and tables that further highlight the strengths of our method.

F.1 Error Comparisons with Paired t-Test Results

For each solver family, we compare the learned greedy router against baseline methods (single-solver only and HINTS) using paired t-tests with a one-sided alternative hypothesis that the learned router achieves lower error or AUC. Results are reported in Table 10. Across all five solver families, the resulting p-values indicate statistically significant improvements of the learned greedy router over both single-solver and HINTS baselines.

F.2 Convergence Histories

See Figure 3 for convergence histories for the five solver families.

Table 10: Selected-action wall-clock replay time to relative solution error 10^{-12}, reported as mean (standard error) in milliseconds over 64 shared test instances. Brackets indicate incomplete coverage. Paired one-sided t-tests compare each baseline with Learned Greedy under the alternative $T_{\text{Learned}} < T_{\text{baseline}}$, using jointly successful instances. A dash indicates that no paired test is available. True-Greedy is an oracle and is included as a reference.

	Poisson		**ConvDiff**		**Anisotropic diffusion**	
Method	Time (ms)	p-value	Time (ms)	p-value	Time (ms)	p-value
Jacobi-related solvers						
Jacobi Only	6428.52 (4.15)	$< 10^{-3}$	10146.76 (14.21)	$< 10^{-3}$	–	–
HINTS-Jacobi	2288.27 (125.46)	$< 10^{-3}$	3278.90 (161.52)	$< 10^{-3}$	1845.77 (126.04)	$< 10^{-3}$
Learned Greedy-Jacobi	1280.80 (64.97)	–	2900.52 (181.73)	–	1117.90 (77.21) [62/64]	–
True-Greedy-Jacobi	1226.02 (66.29)	–	2120.53 (103.75)	–	1130.38 (75.91)	–
Weighted-Jacobi-related solvers ($\omega = 0.67$)						
Weighted Jacobi Only	–	–	–	–	–	–
HINTS-Weighted Jacobi	27.55 (0.85)	$< 10^{-3}$	32.47 (0.96)	$< 10^{-3}$	27.88 (0.09)	$< 10^{-3}$
Learned Greedy-Weighted Jacobi	19.95 (0.37)	–	21.23 (0.19)	–	27.47 (0.14)	–
True-Greedy-Weighted Jacobi	18.96 (0.24)	–	21.10 (0.21)	–	28.32 (0.41)	–
GS-related solvers						
GS Only	10224.51 (6.58)	$< 10^{-3}$	9921.16 (13.81)	$< 10^{-3}$	–	–
HINTS-GS	48.26 (0.31)	$< 10^{-3}$	45.60 (0.15)	$< 10^{-3}$	216.46 (1.58)	$< 10^{-3}$
Learned Greedy-GS	35.63 (0.71)	–	35.60 (0.75)	–	55.53 (2.28)	–
True-Greedy-GS	34.94 (0.81)	–	35.77 (0.73)	–	53.38 (2.42)	–
SymGS-related solvers						
SymGS Only	9210.31 (6.18)	$< 10^{-3}$	9770.15 (13.53)	$< 10^{-3}$	–	–
HINTS-SymGS	37.57 (0.08)	$< 10^{-3}$	40.77 (0.36)	$< 10^{-3}$	236.63 (2.77)	$< 10^{-3}$
Learned Greedy-SymGS	35.04 (0.68)	–	36.69 (0.73)	–	69.44 (4.28)	–
True-Greedy-SymGS	34.21 (0.79)	–	36.06 (0.82)	–	55.12 (2.73)	–
SOR-related solvers ($\omega = 1.5$)						
SOR Only	3408.10 (2.17)	$< 10^{-3}$	2391.65 (3.32)	$< 10^{-3}$	–	–
HINTS-SOR	56.19 (0.60)	$< 10^{-3}$	56.53 (0.16)	$< 10^{-3}$	121.00 (0.83)	$< 10^{-3}$
Learned Greedy-SOR	44.60 (0.97)	–	45.97 (1.06)	–	47.98 (0.80)	–
True-Greedy-SOR	42.68 (0.95)	–	46.39 (1.04)	–	46.58 (0.81)	–
MG-related solvers						
MG Only	36.19 (0.36)	$< 10^{-3}$	41.51 (0.39)	$< 10^{-3}$	871.50 (6.76)	$< 10^{-3}$
HINTS-MG	32.09 (0.32)	$< 10^{-3}$	36.46 (0.30)	$< 10^{-3}$	337.11 (3.57)	$< 10^{-3}$
Learned Greedy-MG	29.93 (0.46)	–	33.71 (0.56)	–	82.76 (5.05)	–
True-Greedy-MG	29.86 (0.46)	–	33.35 (0.49)	–	78.04 (5.00)	–

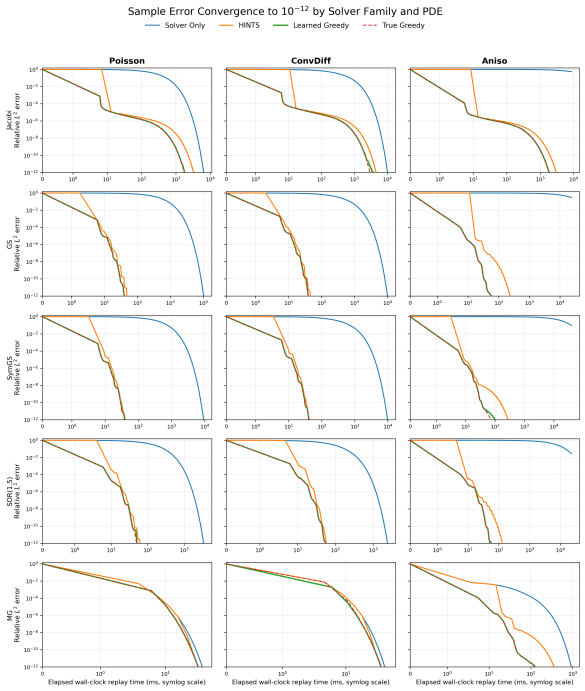

Figure 3: Convergence histories for representative test instances. Rows: Jacobi, Gauss-Seidel, Symmetric Gauss-Seidel, Successive Overrelaxation ($\omega = 1.5$), and Multigrid. Columns: Poisson, Convection Diffusion, and Anisotropic Diffusion. Greedy yields the fastest monotone decay.

F.3 Prediction and Error Visualizations

Figures 4 to 6 provide qualitative visualizations of sample predictions across routing strategies and numerical solvers for all equations for a fixed wall-clock budget of 10 ms. While these prediction may appear identical to the ground truth, Figures 7 to 9 reveals distinct error patterns with varying magnitudes across all predictions. In particular, our learned method consistently achieves the lower-magnitude errors compared to its respective baselines.

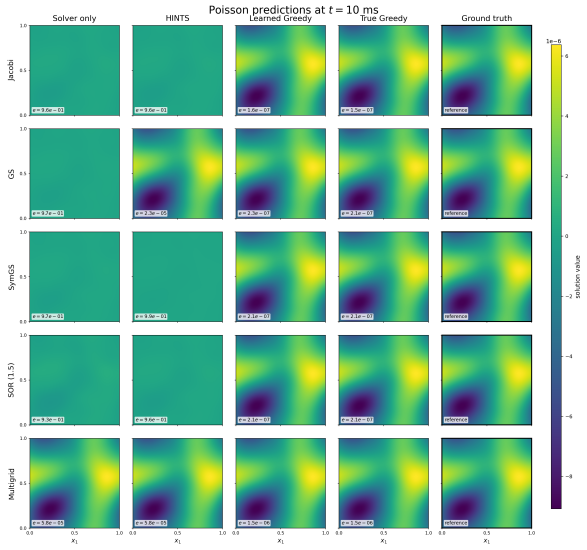

Figure 4: Sample predictions for the Poisson equation across different solver pairings. Each row corresponds to a numerical solver (Jacobi, Gauss–Seidel (GS), Symmetric GS, Damped Jacobi, and Successive Over-relaxation, MG), while columns show predictions from the corresponding solver-only baseline, HINTS, and the learned greedy router. The ground truth solution is shown on the right.

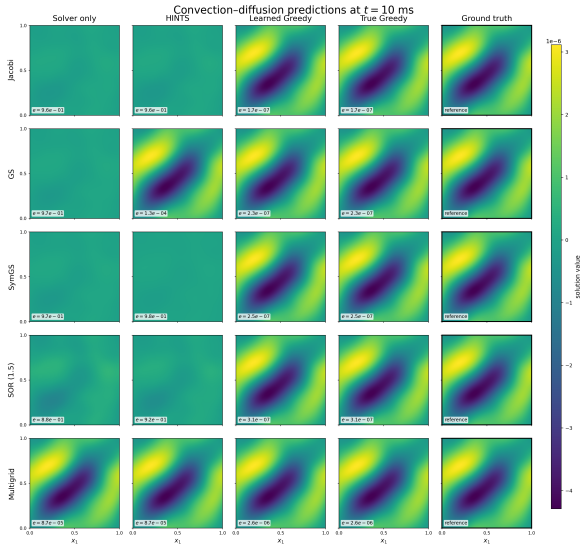

Figure 5: Sample predictions for the convection diffusion equation across different solver pairings. Each row corresponds to a numerical solver (Jacobi, Gauss–Seidel (GS), Symmetric GS, Damped Jacobi, and Successive Over-relaxation, MG), while columns show predictions from the corresponding solver-only baseline, HINTS, and the learned greedy router. The ground truth solution is shown on the right.

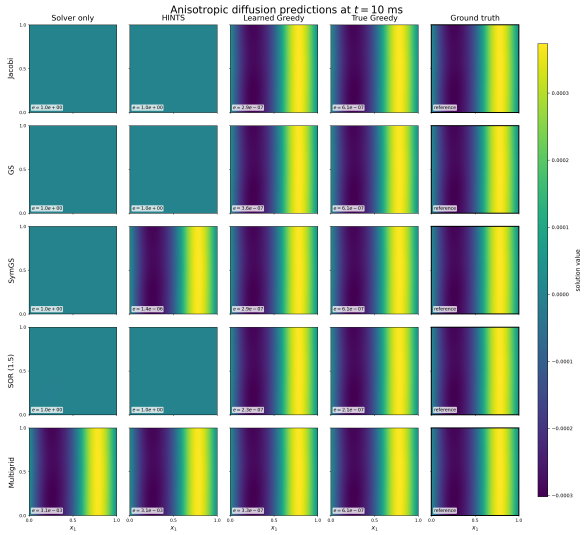

Figure 6: Sample predictions for the anisotropic diffusion equation across different solver pairings. Each row corresponds to a numerical solver (Jacobi, Gauss–Seidel (GS), Symmetric GS, Damped Jacobi, and Successive Over-relaxation, MG), while columns show predictions from the corresponding solver-only baseline, HINTS, and the learned greedy router. The ground truth solution is shown on the right.

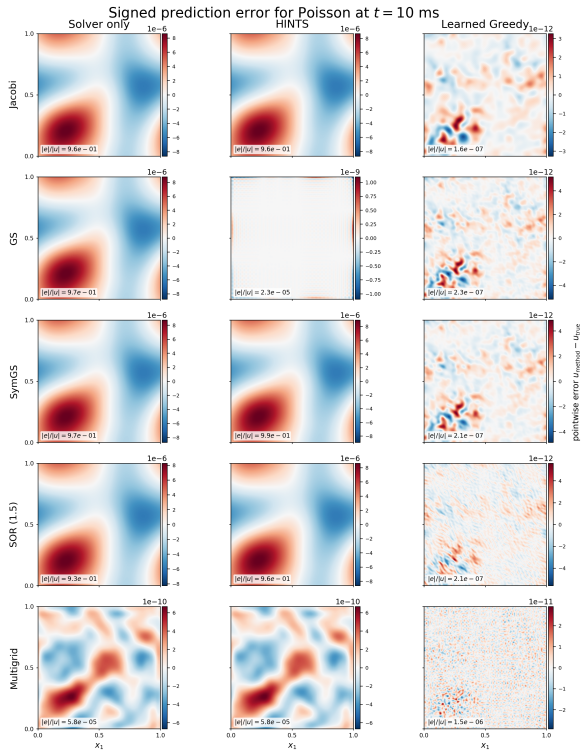

Figure 7: Pointwise error maps for Poisson sample predictions across solver families. Columns correspond to different numerical solvers, while columns show solver-only, HINTS, and learned greedy router. Errors are computed as $u_h - u_h^{(T)} = e_h^{(T)}$. The learned greedy router consistently yields lower-magnitude errors compared to both solver-only and HINTS baselines.

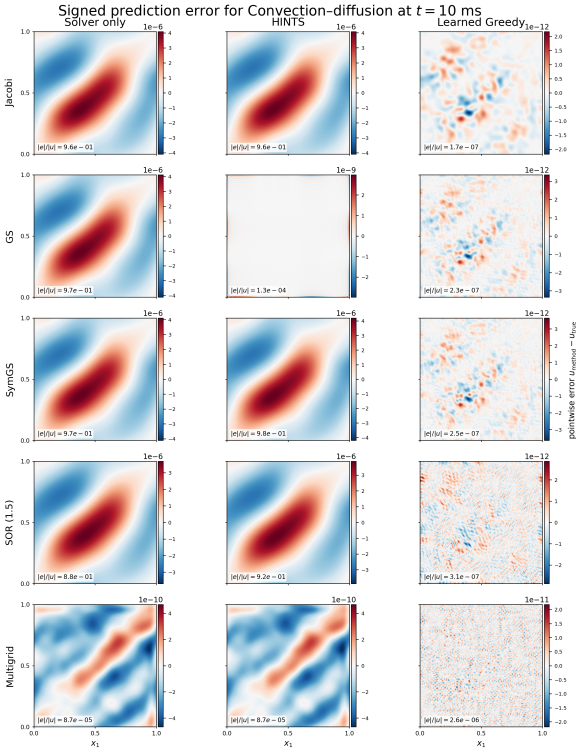

Figure 8: Pointwise error maps for convection diffusion sample predictions across solver families. Columns correspond to different numerical solvers, while columns show solver-only, HINTS, and learned greedy router. Errors are computed as $u_h - u_h^{(T)} = e_h^{(T)}$. The learned greedy router consistently yields lower-magnitude errors compared to both solver-only and HINTS baselines.

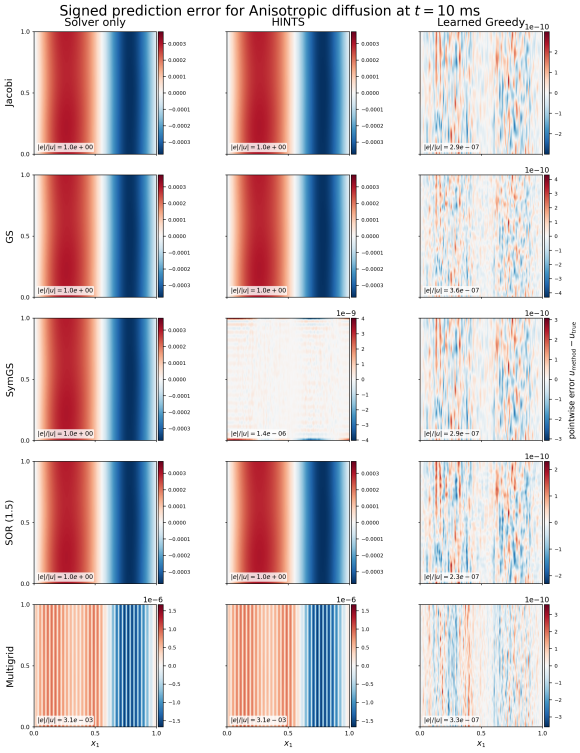

Figure 9: Pointwise error maps for anisotropic diffusion sample predictions across solver families. Columns correspond to different numerical solvers, while columns show solver-only, HINTS, and learned greedy router. Errors are computed as $u_h - u_h^{(T)} = e_h^{(T)}$. The learned greedy router consistently yields lower-magnitude errors compared to both solver-only and HINTS baselines.

F.4 Residual Comparison

Table 11: Relative residual at a fixed 25 ms cost-calibrated selected-action budget and time-averaged residual AUC (lower is better). Values are mean (standard error) over 64 shared test instances. We use $R(t) = \|f_h - \mathcal{L}_h^a u_h(t)\|_2 / \|f_h\|_2$ and $\mathrm{AUC}_{25} = (25\,\mathrm{ms})^{-1} \int_0^{25\,\mathrm{ms}} R(t)\,dt$. The state is held constant between completed actions; complete update and residual-refresh costs, the initial residual, and learned-router inference are charged. A learned routing decision is admitted only when its complete cost-matched macro-action fits in the remaining budget. Each reported p-value is from a paired one-sided t-test of the stated baseline against Learned Greedy, with the alternative that Learned Greedy is lower.

Equation	Poisson			ConvDiff			Anisotropic diffusion			
Method	$R(25\,\mathrm{ms})$	p	AUC_{25}	p	$R(25\,\mathrm{ms})$	p	AUC_{25}	p	$R(25\,\mathrm{ms})$	p	AUC_{25}	p
Jacobi-related solvers												
Jacobi Only	0.606 (0.034)	$< 10^{-3}$	0.697 (0.034)	$< 10^{-3}$	0.659 (0.035)	$< 10^{-3}$	0.737 (0.033)	$< 10^{-3}$	0.649 (0.031)	$< 10^{-3}$	0.737 (0.029)	$< 10^{-3}$
HINTS-Jacobi	0.005 (9.55×10^{-4})	$< 10^{-3}$	0.404 (0.017)	$< 10^{-3}$	0.005 (0.001)	1.000	0.535 (0.022)	$< 10^{-3}$	0.013 (0.003)	1.000	0.452 (0.014)	$< 10^{-3}$
Learned Greedy-Jacobi	0.004 (7.74×10^{-4})	–	0.259 (0.001)	–	0.005 (0.001)	–	0.263 (0.002)	–	0.013 (0.003)	–	0.248 (0.004)	–
GS-related solvers												
GS Only	0.662 (0.035)	$< 10^{-3}$	0.741 (0.033)	$< 10^{-3}$	0.660 (0.035)	$< 10^{-3}$	0.742 (0.033)	$< 10^{-3}$	0.690 (0.032)	$< 10^{-3}$	0.769 (0.029)	$< 10^{-3}$
HINTS-GS	1.99×10^{-6} (5.61×10^{-8})	$< 10^{-3}$	0.271 (0.008)	0.011	3.78×10^{-7} (9.04×10^{-9})	$< 10^{-3}$	0.265 (0.008)	$< 10^{-3}$	3.77×10^{-4} (2.01×10^{-5})	$< 10^{-3}$	0.546 (0.019)	$< 10^{-3}$
Learned Greedy-GS	9.02×10^{-8} (1.52×10^{-8})	–	0.249 (8.81×10^{-4})	–	1.51×10^{-7} (2.55×10^{-8})	–	0.231 (9.68×10^{-4})	–	4.51×10^{-6} (5.74×10^{-7})	–	0.234 (0.001)	–
SymGS-related solvers												
SymGS Only	0.649 (0.035)	$< 10^{-3}$	0.735 (0.033)	$< 10^{-3}$	0.657 (0.035)	$< 10^{-3}$	0.742 (0.033)	$< 10^{-3}$	0.682 (0.032)	$< 10^{-3}$	0.767 (0.030)	$< 10^{-3}$
HINTS-SymGS	7.44×10^{-7} (3.47×10^{-8})	$< 10^{-3}$	0.312 (0.011)	$< 10^{-3}$	7.57×10^{-7} (3.53×10^{-8})	$< 10^{-3}$	0.327 (0.011)	$< 10^{-3}$	7.03×10^{-5} (3.27×10^{-6})	$< 10^{-3}$	0.311 (0.009)	$< 10^{-3}$
Learned Greedy-SymGS	3.21×10^{-8} (5.52×10^{-9})	–	0.253 (0.001)	–	5.90×10^{-8} (1.01×10^{-8})	–	0.259 (0.001)	–	2.63×10^{-6} (3.35×10^{-7})	–	0.238 (0.002)	–
SOR (1.5)-related solvers												
SOR (1.5) Only	0.541 (0.033)	$< 10^{-3}$	0.666 (0.034)	$< 10^{-3}$	0.492 (0.031)	$< 10^{-3}$	0.640 (0.033)	$< 10^{-3}$	0.588 (0.032)	$< 10^{-3}$	0.716 (0.031)	$< 10^{-3}$
HINTS-SOR (1.5)	1.50×10^{-5} (1.14×10^{-6})	0.005	0.327 (0.013)	$< 10^{-3}$	2.91×10^{-5} (1.25×10^{-6})	1.000	0.354 (0.015)	$< 10^{-3}$	1.64×10^{-4} (5.63×10^{-6})	$< 10^{-3}$	0.340 (0.011)	$< 10^{-3}$
Learned Greedy-SOR (1.5)	1.19×10^{-5} (2.10×10^{-6})	–	0.255 (0.002)	–	6.25×10^{-4} (1.11×10^{-4})	–	0.266 (0.002)	–	1.01×10^{-4} (1.47×10^{-5})	–	0.241 (0.002)	–
MG-related solvers												
MG Only	6.60×10^{-8} (5.70×10^{-9})	0.998	0.164 (2.53×10^{-4})	1.000	2.79×10^{-7} (2.25×10^{-8})	$< 10^{-3}$	0.196 (2.12×10^{-4})	1.000	0.205 (0.013)	$< 10^{-3}$	0.458 (0.019)	$< 10^{-3}$
HINTS-MG	1.43×10^{-6} (1.07×10^{-7})	$< 10^{-3}$	0.164 (2.53×10^{-4})	1.000	7.46×10^{-7} (5.29×10^{-8})	$< 10^{-3}$	0.196 (2.12×10^{-4})	1.000	0.001 (6.69×10^{-5})	$< 10^{-3}$	0.426 (0.017)	$< 10^{-3}$
Learned Greedy-MG	1.10×10^{-7} (1.77×10^{-8})	–	0.278 (0.005)	–	3.29×10^{-8} (5.38×10^{-9})	–	0.286 (0.006)	–	2.35×10^{-4} (3.88×10^{-5})	–	0.259 (0.005)	–

Table 12: Relative residual at a fixed 25 ms cost-calibrated selected-action budget and time-averaged residual AUC for growing solver ensembles and their validation-selected best constituent pairwise routers (lower is better). Values are mean (standard error) over 64 shared test instances. We use $R(t) = \|f_h - \mathcal{L}_h^a u_h(t)\|_2 / \|f_h\|_2$ and $\mathrm{AUC}_{25} = (25\,\mathrm{ms})^{-1} \int_0^{25\,\mathrm{ms}} R(t)\,dt$. The state is held constant between completed actions; complete update and residual-refresh costs, the initial residual, and learned-router inference are charged. Each p-value is from a paired one-sided t-test with the alternative that the ensemble router is lower than its best pairwise comparator.

Actions	Method	Poisson				ConvDiff				Anisotropic diffusion			
		$R(25\,\mathrm{ms})$	p	AUC_{25}	p	$R(25\,\mathrm{ms})$	p	AUC_{25}	p	$R(25\,\mathrm{ms})$	p	AUC_{25}	p
$\mathcal{A}_3$	Best pairwise router	1.23×10^{-8} (1.75×10^{-9})	$< 10^{-3}$	0.057 (2.07×10^{-3})	$< 10^{-3}$	2.25×10^{-6} (2.39×10^{-7})	$< 10^{-3}$	0.068 (2.43×10^{-3})	$< 10^{-3}$	4.18×10^{-7} (2.84×10^{-8})	$< 10^{-3}$	0.279 (7.57×10^{-3})	$< 10^{-3}$
	Router($\mathcal{A}_3$)	1.04×10^{-13} (6.06×10^{-15})	–	0.055 (1.67×10^{-3})	–	1.13×10^{-11} (2.30×10^{-12})	–	0.062 (1.77×10^{-3})	–	8.51×10^{-8} (6.80×10^{-8})	–	0.243 (8.12×10^{-3})	–
$\mathcal{A}_4$	Best pairwise router	1.23×10^{-8} (1.75×10^{-9})	$< 10^{-3}$	0.057 (2.07×10^{-3})	$< 10^{-3}$	7.24×10^{-7} (8.62×10^{-8})	$< 10^{-3}$	0.069 (2.50×10^{-3})	$< 10^{-3}$	6.09×10^{-7} (7.81×10^{-8})	$< 10^{-3}$	0.279 (7.70×10^{-3})	$< 10^{-3}$
	Router($\mathcal{A}_4$)	1.23×10^{-13} (4.75×10^{-15})	–	0.055 (1.67×10^{-3})	–	2.35×10^{-13} (2.12×10^{-14})	–	0.062 (1.78×10^{-3})	–	7.00×10^{-8} (1.01×10^{-8})	–	0.243 (8.03×10^{-3})	–
$\mathcal{A}_5$	Best pairwise router	1.23×10^{-8} (1.75×10^{-9})	$< 10^{-3}$	0.057 (2.07×10^{-3})	$< 10^{-3}$	6.95×10^{-8} (9.13×10^{-9})	$< 10^{-3}$	0.070 (2.91×10^{-3})	$< 10^{-3}$	6.09×10^{-7} (7.81×10^{-8})	$< 10^{-3}$	0.279 (7.70×10^{-3})	$< 10^{-3}$
	Router($\mathcal{A}_5$)	1.03×10^{-13} (2.85×10^{-15})	–	0.055 (1.62×10^{-3})	–	3.29×10^{-11} (7.37×10^{-12})	–	0.062 (1.77×10^{-3})	–	6.45×10^{-8} (2.67×10^{-8})	–	0.243 (8.20×10^{-3})	–
$\mathcal{A}_6$	Best pairwise router	1.23×10^{-8} (1.75×10^{-9})	$< 10^{-3}$	0.057 (2.07×10^{-3})	$< 10^{-3}$	6.95×10^{-8} (9.13×10^{-9})	$< 10^{-3}$	0.070 (2.91×10^{-3})	$< 10^{-3}$	6.09×10^{-7} (7.81×10^{-8})	$< 10^{-3}$	0.279 (7.70×10^{-3})	$< 10^{-3}$
	Router($\mathcal{A}_6$)	6.82×10^{-13} (1.10×10^{-13})	–	0.055 (1.67×10^{-3})	–	1.57×10^{-10} (7.03×10^{-11})	–	0.063 (1.94×10^{-3})	–	7.56×10^{-8} (2.99×10^{-8})	–	0.243 (8.00×10^{-3})	–

Table 11 summarizes the performance of single-solver schedules, HINTS, and greedy with respect to the final residuals $r_h^{(T)} = \|f_h - \mathcal{L}_h^a u_h^{(T)}\| / \|f_h\|$ and its residual area under curve $\mathrm{AUC}_{25} = (25\,\mathrm{ms})^{-1} \int_0^{25\,\mathrm{ms}} R(t)\,dt$. Greedy outperforms its HINTS and single-solver counterparts in most equations. We must note that our greedy router is trained to reduce error, not residual. The same error can induce very different residuals depending on the spectrum $\mathcal{L}_h^a$. Table 12 exhibits how residuals are affected by the number of solvers in the solver ensemble. Similar to error, we observe both the final residual and AUC decrease as the number of solvers increase.

F.5 Fourier mode-wise error comparison

We assess frequency-resolved performance by projecting the solution er onto the discrete Fourier basis. Tables 13 to 15reports the endpoint magnitude and time-averaged AUC of modes 1, 5, and 10 at a fixed $25\,\mathrm{ms}$ budget. The learned router generally improves low-frequency convergence relative to solver-only schedules, while classical smoothers can retain an advantage for some higher-frequency modes or AUCs. This behavior is consistent with the complementary strengths motivating hybrid solvers: neural corrections are effective on smooth error components, whereas classical iterations efficiently damp oscillatory components. The router nevertheless optimizes the full L^2 error rather than individual Fourier coefficients. By Parseval's identity,

$$\|e_h^{(t)}\|_2^2 = \sum_m |\widehat{e}_m^{(t)}|^2,$$

reducing its objective decreases aggregate spectral energy, but does not guarantee improvement in every mode separately. Tables 16 to 18 compares each growing-ensemble router with its validation-selected best constituent pairwise router. The ensembles substantially reduce many endpoint mode errors, particularly for convection–diffusion and for the intermediate- and high-frequency modes of the other equations. Their AUC gains are smaller and occasionally mixed, especially for Poisson, indicating that similar spectral trajectories can lead to different endpoint accuracies. Moreover, the results do not improve monotonically from $\mathcal{A}_3$ to $\mathcal{A}_6$; instead, they largely saturate after a small number of complementary actions. Thus, the frequency-resolved results support the benefit of solver diversity without suggesting that every added action improves every Fourier component.

Table 13: Fourier-mode error at a 25 ms wall-clock budget for Poisson on 64 shared test instances. Each entry is mean (standard error) of $|\widehat{e}_{k,k}|$, or its time-averaged wall-clock AUC, using NumPy's unnormalized DFT of the demeaned solution error. Lower is better. Bold marks the lowest deployable method within each solver family and column.

	Mode $(1,1)$		Mode $(5,5)$		Mode $(10,10)$	
Method	Error	AUC	Error	AUC	Error	AUC
Jacobi-related solvers						
Solver Only	2.53×10^{-1} (7.91×10^{-2})	2.82×10^{-1} (8.81×10^{-2})	3.41×10^{-5} (1.39×10^{-5})	$\mathbf{1.39\times10^{-3}}$ $\mathbf{(5.66\times10^{-4})}$	2.20×10^{-13} (9.06×10^{-14})	$\mathbf{4.67\times10^{-5}}$ $\mathbf{(1.92\times10^{-5})}$
HINTS	$\mathbf{1.75\times10^{-8}}$ $\mathbf{(5.54\times10^{-9})}$	1.57×10^{-1} (4.93×10^{-2})	4.62×10^{-9} (1.30×10^{-9})	1.49×10^{-3} (6.06×10^{-4})	3.42×10^{-11} (8.64×10^{-12})	4.71×10^{-5} (1.94×10^{-5})
Learned Greedy	1.91×10^{-8} (6.15×10^{-9})	$\mathbf{7.83\times10^{-2}}$ $\mathbf{(2.45\times10^{-2})}$	$\mathbf{3.03\times10^{-10}}$ $\mathbf{(9.01\times10^{-11})}$	1.81×10^{-3} (7.39×10^{-4})	$\mathbf{1.16\times10^{-15}}$ $\mathbf{(3.42\times10^{-16})}$	2.22×10^{-4} (9.14×10^{-5})
GS-related solvers						
Solver Only	2.74×10^{-1} (8.56×10^{-2})	2.93×10^{-1} (9.16×10^{-2})	3.21×10^{-4} (1.32×10^{-4})	2.28×10^{-3} (9.33×10^{-4})	5.43×10^{-8} (2.25×10^{-8})	$\mathbf{9.42\times10^{-5}}$ $\mathbf{(3.87\times10^{-5})}$
HINTS	$\mathbf{1.98\times10^{-10}}$ $\mathbf{(6.86\times10^{-11})}$	9.76×10^{-2} (3.05×10^{-2})	$\mathbf{4.60\times10^{-10}}$ $\mathbf{(1.87\times10^{-10})}$	1.89×10^{-3} (7.70×10^{-4})	$\mathbf{4.33\times10^{-10}}$ $\mathbf{(1.97\times10^{-10})}$	1.51×10^{-4} (6.22×10^{-5})
Learned Greedy	9.98×10^{-9} (4.43×10^{-9})	$\mathbf{7.63\times10^{-2}}$ $\mathbf{(2.39\times10^{-2})}$	6.15×10^{-9} (2.35×10^{-9})	$\mathbf{1.77\times10^{-3}}$ $\mathbf{(7.20\times10^{-4})}$	5.78×10^{-10} (2.08×10^{-10})	2.17×10^{-4} (8.91×10^{-5})
SymGS-related solvers						
Solver Only	2.69×10^{-1} (8.42×10^{-2})	2.91×10^{-1} (9.10×10^{-2})	2.19×10^{-4} (9.03×10^{-5})	2.14×10^{-3} (8.73×10^{-4})	9.18×10^{-8} (2.66×10^{-8})	$\mathbf{9.45\times10^{-5}}$ $\mathbf{(3.90\times10^{-5})}$
HINTS	6.03×10^{-9} (1.79×10^{-9})	1.15×10^{-1} (3.59×10^{-2})	3.75×10^{-9} (1.44×10^{-9})	1.93×10^{-3} (7.86×10^{-4})	8.60×10^{-10} (3.71×10^{-10})	1.21×10^{-4} (5.01×10^{-5})
Learned Greedy	$\mathbf{3.35\times10^{-9}}$ $\mathbf{(1.43\times10^{-9})}$	$\mathbf{7.68\times10^{-2}}$ $\mathbf{(2.40\times10^{-2})}$	$\mathbf{2.11\times10^{-9}}$ $\mathbf{(7.89\times10^{-10})}$	$\mathbf{1.78\times10^{-3}}$ $\mathbf{(7.25\times10^{-4})}$	$\mathbf{2.52\times10^{-10}}$ $\mathbf{(8.96\times10^{-11})}$	2.18×10^{-4} (8.98×10^{-5})
SOR(1.5)-related solvers						
Solver Only	2.12×10^{-1} (6.62×10^{-2})	2.59×10^{-1} (8.11×10^{-2})	1.05×10^{-5} (4.04×10^{-6})	$\mathbf{1.19\times10^{-3}}$ $\mathbf{(4.86\times10^{-4})}$	1.76×10^{-8} (6.56×10^{-9})	$\mathbf{8.17\times10^{-5}}$ $\mathbf{(3.31\times10^{-5})}$
HINTS	1.77×10^{-7} (5.94×10^{-8})	1.21×10^{-1} (3.77×10^{-2})	2.43×10^{-8} (9.51×10^{-9})	1.40×10^{-3} (5.70×10^{-4})	1.62×10^{-9} (5.47×10^{-10})	9.88×10^{-5} (4.00×10^{-5})
Learned Greedy	$\mathbf{4.96\times10^{-8}}$ $\mathbf{(2.20\times10^{-8})}$	$\mathbf{7.68\times10^{-2}}$ $\mathbf{(2.40\times10^{-2})}$	$\mathbf{1.01\times10^{-8}}$ $\mathbf{(3.75\times10^{-9})}$	1.78×10^{-3} (7.25×10^{-4})	$\mathbf{5.86\times10^{-10}}$ $\mathbf{(2.24\times10^{-10})}$	2.19×10^{-4} (8.98×10^{-5})
MG-related solvers						
Solver Only	3.71×10^{-12} (1.28×10^{-12})	$\mathbf{5.01\times10^{-2}}$ $\mathbf{(1.57\times10^{-2})}$	4.36×10^{-9} (1.75×10^{-9})	$\mathbf{1.26\times10^{-3}}$ $\mathbf{(5.14\times10^{-4})}$	1.47×10^{-10} (5.22×10^{-11})	$\mathbf{1.52\times10^{-4}}$ $\mathbf{(6.23\times10^{-5})}$
HINTS	$\mathbf{8.71\times10^{-14}}$ $\mathbf{(4.41\times10^{-14})}$	5.01×10^{-2} (1.57×10^{-2})	$\mathbf{8.21\times10^{-14}}$ $\mathbf{(3.15\times10^{-14})}$	1.26×10^{-3} (5.14×10^{-4})	$\mathbf{1.23\times10^{-13}}$ $\mathbf{(5.09\times10^{-14})}$	1.52×10^{-4} (6.23×10^{-5})
Learned Greedy	1.06×10^{-11} (3.79×10^{-12})	7.76×10^{-2} (2.43×10^{-2})	5.69×10^{-10} (2.22×10^{-10})	1.80×10^{-3} (7.32×10^{-4})	2.10×10^{-10} (8.08×10^{-11})	2.20×10^{-4} (9.06×10^{-5})

Table 14: Fourier-mode error at a 25 ms wall-clock budget for ConvDiff on 64 shared test instances. Each entry is mean (standard error) of $|\widehat{e}_{k,k}|$, or its time-averaged wall-clock AUC, using NumPy's unnormalized DFT of the demeaned solution error. Lower is better. Bold marks the lowest deployable method within each solver family and column.

	Mode $(1,1)$		Mode $(5,5)$		Mode $(10,10)$	
Method	Error	AUC	Error	AUC	Error	AUC
Jacobi-related solvers						
Solver Only	8.21×10^{-2} (2.57×10^{-2})	8.78×10^{-2} (2.75×10^{-2})	2.19×10^{-4} (8.95×10^{-5})	1.82×10^{-3} (7.43×10^{-4})	9.26×10^{-10} (3.81×10^{-10})	$\mathbf{7.17\times10^{-5}}$ $\mathbf{(2.95\times10^{-5})}$
HINTS	8.12×10^{-9} (2.69×10^{-9})	6.22×10^{-2} (1.95×10^{-2})	3.49×10^{-9} (1.22×10^{-9})	1.81×10^{-3} (7.37×10^{-4})	1.21×10^{-10} (4.03×10^{-11})	7.20×10^{-5} (2.96×10^{-5})
Learned Greedy	$\mathbf{5.98\times10^{-9}}$ $\mathbf{(1.70\times10^{-9})}$	$\mathbf{2.37\times10^{-2}}$ $\mathbf{(7.41\times10^{-3})}$	$\mathbf{6.84\times10^{-10}}$ $\mathbf{(1.99\times10^{-10})}$	$\mathbf{1.55\times10^{-3}}$ $\mathbf{(6.30\times10^{-4})}$	$\mathbf{3.14\times10^{-13}}$ $\mathbf{(1.04\times10^{-13})}$	2.14×10^{-4} (8.81×10^{-5})
GS-related solvers						
Solver Only	8.10×10^{-2} (2.53×10^{-2})	8.73×10^{-2} (2.73×10^{-2})	2.05×10^{-4} (8.48×10^{-5})	1.83×10^{-3} (7.48×10^{-4})	2.79×10^{-8} (1.03×10^{-8})	$\mathbf{9.02\times10^{-5}}$ $\mathbf{(3.72\times10^{-5})}$
HINTS	$\mathbf{1.86\times10^{-10}}$ $\mathbf{(5.36\times10^{-11})}$	2.85×10^{-2} (8.92×10^{-3})	$\mathbf{2.98\times10^{-10}}$ $\mathbf{(1.15\times10^{-10})}$	1.50×10^{-3} (6.13×10^{-4})	$\mathbf{1.99\times10^{-10}}$ $\mathbf{(8.82\times10^{-11})}$	1.30×10^{-4} (5.37×10^{-5})
Learned Greedy	1.49×10^{-8} (6.63×10^{-9})	$\mathbf{2.11\times10^{-2}}$ $\mathbf{(6.61\times10^{-3})}$	8.76×10^{-9} (3.33×10^{-9})	$\mathbf{1.38\times10^{-3}}$ $\mathbf{(5.62\times10^{-4})}$	7.90×10^{-10} (2.85×10^{-10})	1.92×10^{-4} (7.87×10^{-5})
SymGS-related solvers						
Solver Only	8.17×10^{-2} (2.56×10^{-2})	8.77×10^{-2} (2.75×10^{-2})	2.42×10^{-4} (9.51×10^{-5})	1.93×10^{-3} (7.82×10^{-4})	1.70×10^{-7} (5.91×10^{-8})	$\mathbf{9.89\times10^{-5}}$ $\mathbf{(4.09\times10^{-5})}$
HINTS	6.20×10^{-9} (1.86×10^{-9})	3.60×10^{-2} (1.13×10^{-2})	3.60×10^{-9} (1.39×10^{-9})	1.71×10^{-3} (6.94×10^{-4})	7.76×10^{-10} (3.25×10^{-10})	1.23×10^{-4} (5.08×10^{-5})
Learned Greedy	$\mathbf{4.84\times10^{-9}}$ $\mathbf{(2.11\times10^{-9})}$	$\mathbf{2.36\times10^{-2}}$ $\mathbf{(7.37\times10^{-3})}$	$\mathbf{3.29\times10^{-9}}$ $\mathbf{(1.22\times10^{-9})}$	$\mathbf{1.54\times10^{-3}}$ $\mathbf{(6.27\times10^{-4})}$	$\mathbf{5.05\times10^{-10}}$ $\mathbf{(1.80\times10^{-10})}$	2.14×10^{-4} (8.77×10^{-5})
SOR(1.5)-related solvers						
Solver Only	4.83×10^{-2} (1.51×10^{-2})	6.89×10^{-2} (2.16×10^{-2})	8.09×10^{-7} (2.80×10^{-7})	$\mathbf{7.74\times10^{-4}}$ $\mathbf{(3.18\times10^{-4})}$	4.88×10^{-8} (1.79×10^{-8})	$\mathbf{7.21\times10^{-5}}$ $\mathbf{(2.93\times10^{-5})}$
HINTS	3.51×10^{-7} (1.23×10^{-7})	3.80×10^{-2} (1.19×10^{-2})	3.18×10^{-8} (1.27×10^{-8})	9.07×10^{-4} (3.73×10^{-4})	1.54×10^{-9} (5.71×10^{-10})	7.96×10^{-5} (3.23×10^{-5})
Learned Greedy	$\mathbf{1.66\times10^{-12}}$ $\mathbf{(6.19\times10^{-13})}$	$\mathbf{2.40\times10^{-2}}$ $\mathbf{(7.50\times10^{-3})}$	$\mathbf{2.09\times10^{-12}}$ $\mathbf{(9.18\times10^{-13})}$	1.57×10^{-3} (6.38×10^{-4})	$\mathbf{1.24\times10^{-12}}$ $\mathbf{(5.34\times10^{-13})}$	2.18×10^{-4} (8.93×10^{-5})
MG-related solvers						
Solver Only	1.02×10^{-10} (3.91×10^{-11})	$\mathbf{1.79\times10^{-2}}$ $\mathbf{(5.61\times10^{-3})}$	7.59×10^{-9} (3.12×10^{-9})	$\mathbf{1.24\times10^{-3}}$ $\mathbf{(5.06\times10^{-4})}$	9.37×10^{-10} (3.48×10^{-10})	$\mathbf{1.72\times10^{-4}}$ $\mathbf{(7.05\times10^{-5})}$
HINTS	$\mathbf{2.58\times10^{-14}}$ $\mathbf{(1.36\times10^{-14})}$	1.79×10^{-2} (5.61×10^{-3})	$\mathbf{2.72\times10^{-14}}$ $\mathbf{(1.02\times10^{-14})}$	1.24×10^{-3} (5.06×10^{-4})	$\mathbf{2.35\times10^{-14}}$ $\mathbf{(1.08\times10^{-14})}$	1.72×10^{-4} (7.05×10^{-5})
Learned Greedy	1.40×10^{-11} (5.40×10^{-12})	2.35×10^{-2} (7.35×10^{-3})	1.54×10^{-10} (6.13×10^{-11})	1.53×10^{-3} (6.25×10^{-4})	1.17×10^{-10} (4.41×10^{-11})	2.13×10^{-4} (8.74×10^{-5})

Table 15: Fourier-mode error at a 25 ms wall-clock budget for Anisotropic diffusion on 64 shared test instances. Each entry is mean (standard error) of $|\widehat{e}_{k,k}|$, or its time-averaged wall-clock AUC, using NumPy's unnormalized DFT of the demeaned solution error. Lower is better. Bold marks the lowest deployable method within each solver family and column.

	Mode $(1,1)$		Mode $(5,5)$		Mode $(10,10)$	
Method	Error	AUC	Error	AUC	Error	AUC
Jacobi-related solvers						
Solver Only	5.19×10^{-1} (1.63×10^{-1})	5.68×10^{-1} (1.78×10^{-1})	1.74×10^{-4} (7.08×10^{-5})	3.32×10^{-3} (1.35×10^{-3})	2.15×10^{-11} (8.82×10^{-12})	$\mathbf{1.13\times10^{-4}}$ $\mathbf{(4.65\times10^{-5})}$
HINTS	1.30×10^{-6} (4.32×10^{-7})	3.34×10^{-1} (1.05×10^{-1})	1.52×10^{-7} (3.90×10^{-8})	3.34×10^{-3} (1.36×10^{-3})	1.59×10^{-9} (4.52×10^{-10})	1.14×10^{-4} (4.67×10^{-5})
Learned Greedy	$\mathbf{6.83\times10^{-7}}$ $\mathbf{(1.80\times10^{-7})}$	$\mathbf{1.41\times10^{-1}}$ $\mathbf{(4.40\times10^{-2})}$	$\mathbf{3.83\times10^{-8}}$ $\mathbf{(1.01\times10^{-8})}$	$\mathbf{3.25\times10^{-3}}$ $\mathbf{(1.33\times10^{-3})}$	$\mathbf{4.10\times10^{-12}}$ $\mathbf{(1.35\times10^{-12})}$	3.99×10^{-4} (1.64×10^{-4})
GS-related solvers						
Solver Only	5.46×10^{-1} (1.71×10^{-1})	5.82×10^{-1} (1.82×10^{-1})	7.71×10^{-4} (3.21×10^{-4})	4.75×10^{-3} (1.94×10^{-3})	1.40×10^{-5} (6.27×10^{-6})	2.04×10^{-4} (8.33×10^{-5})
HINTS	2.05×10^{-5} (6.51×10^{-6})	4.03×10^{-1} (1.26×10^{-1})	4.15×10^{-6} (1.73×10^{-6})	4.54×10^{-3} (1.85×10^{-3})	2.16×10^{-7} (9.42×10^{-8})	$\mathbf{2.01\times10^{-4}}$ $\mathbf{(8.19\times10^{-5})}$
Learned Greedy	$\mathbf{7.85\times10^{-8}}$ $\mathbf{(2.76\times10^{-8})}$	$\mathbf{1.39\times10^{-1}}$ $\mathbf{(4.35\times10^{-2})}$	$\mathbf{4.87\times10^{-8}}$ $\mathbf{(1.87\times10^{-8})}$	$\mathbf{3.22\times10^{-3}}$ $\mathbf{(1.31\times10^{-3})}$	$\mathbf{9.79\times10^{-9}}$ $\mathbf{(3.86\times10^{-9})}$	3.97×10^{-4} (1.63×10^{-4})
SymGS-related solvers						
Solver Only	5.34×10^{-1} (1.67×10^{-1})	5.77×10^{-1} (1.80×10^{-1})	5.31×10^{-4} (2.01×10^{-4})	4.24×10^{-3} (1.72×10^{-3})	4.81×10^{-5} (2.05×10^{-5})	$\mathbf{2.33\times10^{-4}}$ $\mathbf{(9.39\times10^{-5})}$
HINTS	5.96×10^{-7} (1.95×10^{-7})	2.16×10^{-1} (6.74×10^{-2})	2.42×10^{-7} (9.85×10^{-8})	3.65×10^{-3} (1.49×10^{-3})	6.43×10^{-8} (3.19×10^{-8})	2.42×10^{-4} (9.91×10^{-5})
Learned Greedy	$\mathbf{1.73\times10^{-7}}$ $\mathbf{(6.85\times10^{-8})}$	$\mathbf{1.41\times10^{-1}}$ $\mathbf{(4.40\times10^{-2})}$	$\mathbf{6.89\times10^{-8}}$ $\mathbf{(2.81\times10^{-8})}$	$\mathbf{3.26\times10^{-3}}$ $\mathbf{(1.33\times10^{-3})}$	$\mathbf{1.52\times10^{-8}}$ $\mathbf{(5.54\times10^{-9})}$	4.01×10^{-4} (1.65×10^{-4})
SOR(1.5)-related solvers						
Solver Only	4.22×10^{-1} (1.32×10^{-1})	5.16×10^{-1} (1.62×10^{-1})	9.91×10^{-5} (3.78×10^{-5})	$\mathbf{2.46\times10^{-3}}$ $\mathbf{(9.97\times10^{-4})}$	2.45×10^{-5} (9.88×10^{-6})	$\mathbf{1.88\times10^{-4}}$ $\mathbf{(7.46\times10^{-5})}$
HINTS	1.06×10^{-6} (3.40×10^{-7})	2.30×10^{-1} (7.19×10^{-2})	1.55×10^{-7} (6.40×10^{-8})	2.74×10^{-3} (1.12×10^{-3})	$\mathbf{2.30\times10^{-8}}$ $\mathbf{(8.58\times10^{-9})}$	1.97×10^{-4} (7.82×10^{-5})
Learned Greedy	$\mathbf{4.08\times10^{-7}}$ $\mathbf{(1.56\times10^{-7})}$	$\mathbf{1.39\times10^{-1}}$ $\mathbf{(4.35\times10^{-2})}$	$\mathbf{9.04\times10^{-8}}$ $\mathbf{(3.37\times10^{-8})}$	3.22×10^{-3} (1.31×10^{-3})	3.06×10^{-8} (1.17×10^{-8})	3.98×10^{-4} (1.63×10^{-4})
MG-related solvers						
Solver Only	3.33×10^{-5} (1.05×10^{-5})	9.49×10^{-2} (2.97×10^{-2})	7.51×10^{-6} (3.27×10^{-6})	2.40×10^{-3} (9.76×10^{-4})	3.88×10^{-6} (1.46×10^{-6})	2.92×10^{-4} (1.19×10^{-4})
HINTS	1.18×10^{-7} (2.83×10^{-8})	$\mathbf{9.49\times10^{-2}}$ $\mathbf{(2.97\times10^{-2})}$	9.05×10^{-8} (3.54×10^{-8})	$\mathbf{2.40\times10^{-3}}$ $\mathbf{(9.76\times10^{-4})}$	1.50×10^{-7} (5.93×10^{-8})	$\mathbf{2.92\times10^{-4}}$ $\mathbf{(1.19\times10^{-4})}$
Learned Greedy	$\mathbf{1.24\times10^{-10}}$ $\mathbf{(4.55\times10^{-11})}$	1.39×10^{-1} (4.34×10^{-2})	$\mathbf{1.15\times10^{-10}}$ $\mathbf{(3.71\times10^{-11})}$	3.21×10^{-3} (1.31×10^{-3})	$\mathbf{1.49\times10^{-10}}$ $\mathbf{(4.92\times10^{-11})}$	3.94×10^{-4} (1.62×10^{-4})

Table 16: Fourier-mode error at a 25 ms wall-clock budget for Poisson growing solver ensembles on 64 shared test instances. Each entry is mean (standard error) of $|\widehat{e}_{k,k}|$, or its time-averaged wall-clock AUC, using NumPy's unnormalized DFT of the demeaned solution error. Lower is better. Each ensemble router is compared with its validation-selected best constituent pairwise router; bold marks the lower method within each ensemble size and column.

Actions	Method	Mode $(1,1)$		Mode $(5,5)$		Mode $(10,10)$	
		Error	AUC	Error	AUC	Error	AUC
$\mathcal{A}_3$	Best pairwise router	5.69×10^{-15} (2.58×10^{-15})	0.012 (5.41×10^{-3})	1.05×10^{-15} (4.13×10^{-16})	1.52×10^{-4} (7.32×10^{-5})	1.75×10^{-16} (7.88×10^{-17})	5.19×10^{-5} (2.77×10^{-5})
	Router($\mathcal{A}_3$)	$\mathbf{2.36\times10^{-15}}$ $\mathbf{(1.04\times10^{-15})}$	$\mathbf{0.012}$ $\mathbf{(5.40\times10^{-3})}$	$\mathbf{7.05\times10^{-18}}$ $\mathbf{(2.88\times10^{-18})}$	$\mathbf{1.51\times10^{-4}}$ $\mathbf{(7.29\times10^{-5})}$	$\mathbf{2.94\times10^{-18}}$ $\mathbf{(1.22\times10^{-18})}$	$\mathbf{5.19\times10^{-5}}$ $\mathbf{(2.78\times10^{-5})}$
$\mathcal{A}_4$	Best pairwise router	$\mathbf{5.69\times10^{-15}}$ $\mathbf{(2.58\times10^{-15})}$	$\mathbf{0.012}$ $\mathbf{(5.41\times10^{-3})}$	1.05×10^{-15} (4.13×10^{-16})	1.52×10^{-4} (7.32×10^{-5})	1.75×10^{-16} (7.88×10^{-17})	$\mathbf{5.19\times10^{-5}}$ $\mathbf{(2.77\times10^{-5})}$
	Router($\mathcal{A}_4$)	5.98×10^{-15} (2.94×10^{-15})	0.012 (5.41×10^{-3})	$\mathbf{4.01\times10^{-17}}$ $\mathbf{(1.85\times10^{-17})}$	$\mathbf{1.52\times10^{-4}}$ $\mathbf{(7.31\times10^{-5})}$	$\mathbf{3.16\times10^{-18}}$ $\mathbf{(1.51\times10^{-18})}$	5.21×10^{-5} (2.78×10^{-5})
$\mathcal{A}_5$	Best pairwise router	5.69×10^{-15} (2.58×10^{-15})	$\mathbf{0.012}$ $\mathbf{(5.41\times10^{-3})}$	1.05×10^{-15} (4.13×10^{-16})	$\mathbf{1.52\times10^{-4}}$ $\mathbf{(7.32\times10^{-5})}$	1.75×10^{-16} (7.88×10^{-17})	$\mathbf{5.19\times10^{-5}}$ $\mathbf{(2.77\times10^{-5})}$
	Router($\mathcal{A}_5$)	$\mathbf{2.23\times10^{-15}}$ $\mathbf{(8.87\times10^{-16})}$	0.012 (5.46×10^{-3})	$\mathbf{1.37\times10^{-17}}$ $\mathbf{(7.59\times10^{-18})}$	1.52×10^{-4} (7.38×10^{-5})	$\mathbf{3.70\times10^{-18}}$ $\mathbf{(1.70\times10^{-18})}$	5.24×10^{-5} (2.80×10^{-5})
$\mathcal{A}_6$	Best pairwise router	5.69×10^{-15} (2.58×10^{-15})	$\mathbf{0.012}$ $\mathbf{(5.41\times10^{-3})}$	1.05×10^{-15} (4.13×10^{-16})	$\mathbf{1.52\times10^{-4}}$ $\mathbf{(7.32\times10^{-5})}$	1.75×10^{-16} (7.88×10^{-17})	$\mathbf{5.19\times10^{-5}}$ $\mathbf{(2.77\times10^{-5})}$
	Router($\mathcal{A}_6$)	$\mathbf{2.90\times10^{-15}}$ $\mathbf{(1.37\times10^{-15})}$	0.012 (5.42×10^{-3})	$\mathbf{2.37\times10^{-17}}$ $\mathbf{(1.22\times10^{-17})}$	1.52×10^{-4} (7.33×10^{-5})	$\mathbf{8.21\times10^{-17}}$ $\mathbf{(4.83\times10^{-17})}$	5.21×10^{-5} (2.78×10^{-5})

Table 17: Fourier-mode error at a 25 ms wall-clock budget for convection–diffusion growing solver ensembles on 64 shared test instances. Each entry is mean (standard error) of $|\widehat{e}_{k,k}|$, or its time-averaged wall-clock AUC, using NumPy's unnormalized DFT of the demeaned solution error. Lower is better. Each ensemble router is compared with its validation-selected best constituent pairwise router; bold marks the lower method within each ensemble size and column.

Actions	Method	Mode $(1,1)$		Mode $(5,5)$		Mode $(10,10)$	
		Error	AUC	Error	AUC	Error	AUC
$\mathcal{A}_3$	Best pairwise router	4.47×10^{-9} (1.73×10^{-9})	$\mathbf{1.97\times10^{-3}}$ $(\mathbf{5.03\times10^{-4}})$	1.14×10^{-9} (6.59×10^{-10})	$\mathbf{1.45\times10^{-4}}$ $(\mathbf{7.93\times10^{-5}})$	3.76×10^{-12} (1.66×10^{-12})	$\mathbf{2.71\times10^{-5}}$ $(\mathbf{1.45\times10^{-5}})$
	Router($\mathcal{A}_3$)	$\mathbf{6.04\times10^{-16}}$ $(\mathbf{2.38\times10^{-16}})$	1.98×10^{-3} (5.05×10^{-4})	$\mathbf{2.64\times10^{-17}}$ $(\mathbf{1.28\times10^{-17}})$	1.47×10^{-4} (8.03×10^{-5})	$\mathbf{1.12\times10^{-18}}$ $(\mathbf{5.09\times10^{-19}})$	2.73×10^{-5} (1.47×10^{-5})
$\mathcal{A}_4$	Best pairwise router	1.64×10^{-10} (8.82×10^{-11})	$\mathbf{1.97\times10^{-3}}$ $(\mathbf{5.00\times10^{-4}})$	9.04×10^{-11} (4.16×10^{-11})	1.47×10^{-4} (7.97×10^{-5})	2.49×10^{-11} (1.48×10^{-11})	2.75×10^{-5} (1.46×10^{-5})
	Router($\mathcal{A}_4$)	$\mathbf{6.02\times10^{-16}}$ $(\mathbf{2.25\times10^{-16}})$	1.97×10^{-3} (5.03×10^{-4})	$\mathbf{3.78\times10^{-17}}$ $(\mathbf{1.82\times10^{-17}})$	$\mathbf{1.45\times10^{-4}}$ $(\mathbf{7.89\times10^{-5}})$	$\mathbf{7.55\times10^{-19}}$ $(\mathbf{2.03\times10^{-19}})$	$\mathbf{2.71\times10^{-5}}$ $(\mathbf{1.44\times10^{-5}})$
$\mathcal{A}_5$	Best pairwise router	7.28×10^{-10} (3.57×10^{-10})	1.97×10^{-3} (5.04×10^{-4})	6.28×10^{-11} (3.35×10^{-11})	1.47×10^{-4} (7.99×10^{-5})	5.99×10^{-11} (3.55×10^{-11})	2.79×10^{-5} (1.48×10^{-5})
	Router($\mathcal{A}_5$)	$\mathbf{6.47\times10^{-16}}$ $(\mathbf{2.39\times10^{-16}})$	$\mathbf{1.96\times10^{-3}}$ $(\mathbf{5.01\times10^{-4}})$	$\mathbf{2.74\times10^{-17}}$ $(\mathbf{1.28\times10^{-17}})$	$\mathbf{1.46\times10^{-4}}$ $(\mathbf{7.96\times10^{-5}})$	$\mathbf{1.10\times10^{-18}}$ $(\mathbf{5.04\times10^{-19}})$	$\mathbf{2.71\times10^{-5}}$ $(\mathbf{1.45\times10^{-5}})$
$\mathcal{A}_6$	Best pairwise router	7.28×10^{-10} (3.57×10^{-10})	1.97×10^{-3} (5.04×10^{-4})	6.28×10^{-11} (3.35×10^{-11})	1.47×10^{-4} (7.99×10^{-5})	5.99×10^{-11} (3.55×10^{-11})	2.79×10^{-5} (1.48×10^{-5})
	Router($\mathcal{A}_6$)	$\mathbf{1.23\times10^{-16}}$ $(\mathbf{3.07\times10^{-17}})$	$\mathbf{1.96\times10^{-3}}$ $(\mathbf{5.02\times10^{-4}})$	$\mathbf{8.18\times10^{-18}}$ $(\mathbf{2.86\times10^{-18}})$	$\mathbf{1.45\times10^{-4}}$ $(\mathbf{7.94\times10^{-5}})$	$\mathbf{3.02\times10^{-18}}$ $(\mathbf{1.22\times10^{-18}})$	$\mathbf{2.71\times10^{-5}}$ $(\mathbf{1.45\times10^{-5}})$

Table 18: Fourier-mode error at a 25 ms wall-clock budget for anisotropic diffusion growing solver ensembles on 64 shared test instances. Each entry is mean (standard error) of $|\widehat{e}_{k,k}|$, or its time-averaged wall-clock AUC, using NumPy's unnormalized DFT of the demeaned solution error. Lower is better. Each ensemble router is compared with its validation-selected best constituent pairwise router; bold marks the lower method within each ensemble size and column.

Actions	Method	Mode $(1,1)$		Mode $(5,5)$		Mode $(10,10)$	
		Error	AUC	Error	AUC	Error	AUC
$\mathcal{A}_3$	Best pairwise router	$\mathbf{2.56\times10^{-15}}$ $(\mathbf{6.66\times10^{-16}})$	$\mathbf{0.047}$ $(\mathbf{0.012})$	1.26×10^{-12} (6.90×10^{-13})	2.42×10^{-3} (1.32×10^{-3})	$\mathbf{2.25\times10^{-17}}$ $(\mathbf{6.13\times10^{-18}})$	3.99×10^{-4} (2.13×10^{-4})
	Router($\mathcal{A}_3$)	1.90×10^{-14} (9.20×10^{-15})	0.047 (0.012)	$\mathbf{1.15\times10^{-16}}$ $(\mathbf{5.16\times10^{-17}})$	$\mathbf{2.30\times10^{-3}}$ $(\mathbf{1.32\times10^{-3}})$	2.35×10^{-17} (6.25×10^{-18})	$\mathbf{3.84\times10^{-4}}$ $(\mathbf{2.14\times10^{-4}})$
$\mathcal{A}_4$	Best pairwise router	2.69×10^{-15} (7.08×10^{-16})	$\mathbf{0.047}$ $(\mathbf{0.012})$	1.56×10^{-11} (6.59×10^{-12})	2.42×10^{-3} (1.32×10^{-3})	1.51×10^{-11} (7.02×10^{-12})	4.00×10^{-4} (2.14×10^{-4})
	Router($\mathcal{A}_4$)	$\mathbf{2.48\times10^{-15}}$ $(\mathbf{6.38\times10^{-16}})$	0.047 (0.012)	$\mathbf{1.88\times10^{-12}}$ $(\mathbf{1.09\times10^{-12}})$	$\mathbf{2.30\times10^{-3}}$ $(\mathbf{1.33\times10^{-3}})$	$\mathbf{2.51\times10^{-15}}$ $(\mathbf{9.32\times10^{-16}})$	$\mathbf{3.84\times10^{-4}}$ $(\mathbf{2.14\times10^{-4}})$
$\mathcal{A}_5$	Best pairwise router	2.69×10^{-15} (7.08×10^{-16})	0.047 (0.012)	1.56×10^{-11} (6.59×10^{-12})	2.42×10^{-3} (1.32×10^{-3})	1.51×10^{-11} (7.02×10^{-12})	4.00×10^{-4} (2.14×10^{-4})
	Router($\mathcal{A}_5$)	$\mathbf{2.47\times10^{-15}}$ $(\mathbf{6.38\times10^{-16}})$	$\mathbf{0.047}$ $(\mathbf{0.012})$	$\mathbf{1.83\times10^{-12}}$ $(\mathbf{1.09\times10^{-12}})$	$\mathbf{2.30\times10^{-3}}$ $(\mathbf{1.33\times10^{-3}})$	$\mathbf{9.12\times10^{-16}}$ $(\mathbf{6.13\times10^{-16}})$	$\mathbf{3.84\times10^{-4}}$ $(\mathbf{2.14\times10^{-4}})$
$\mathcal{A}_6$	Best pairwise router	2.69×10^{-15} (7.08×10^{-16})	$\mathbf{0.047}$ $(\mathbf{0.012})$	1.56×10^{-11} (6.59×10^{-12})	2.42×10^{-3} (1.32×10^{-3})	1.51×10^{-11} (7.02×10^{-12})	4.00×10^{-4} (2.14×10^{-4})
	Router($\mathcal{A}_6$)	$\mathbf{2.48\times10^{-15}}$ $(\mathbf{6.38\times10^{-16}})$	0.047 (0.012)	$\mathbf{1.84\times10^{-12}}$ $(\mathbf{1.09\times10^{-12}})$	$\mathbf{2.30\times10^{-3}}$ $(\mathbf{1.32\times10^{-3}})$	$\mathbf{1.05\times10^{-15}}$ $(\mathbf{6.26\times10^{-16}})$	$\mathbf{3.84\times10^{-4}}$ $(\mathbf{2.13\times10^{-4}})$

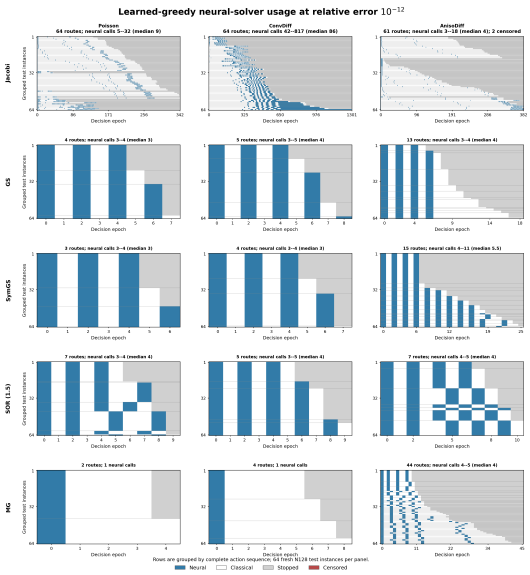

Figure 10: Learned-router solver usage for paired classical–neural solvers across 64 test instances at a relative solution-error tolerance of 10^{-12}. Each panel shows the selected action at each decision epoch, with instances grouped by their complete action sequence. Gray indicates that the tolerance has been reached, while red denotes censored instances that do not reach the tolerance within the evaluation horizon.

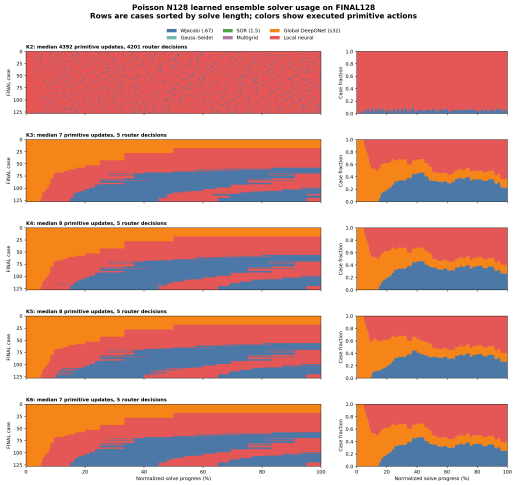

Figure 11: Solver usage by the learned router for growing Poisson solver ensembles on 128 test instances. Rows correspond to instances sorted by solve length, and colors indicate the executed primitive actions over normalized solve progress. The right column reports the fraction of instances using each action at each point in the solve.

F.6 Ensemble Solver Decisions and DeepONet Usage

Figure 10 visualize the routing decisions of the learned greedy policies across all test samples and solver families.

This figure suggests that, in some solver families, the routing strategy is highly sample-dependent, indicating that a fixed routing schedule across all instances would be suboptimal and can slow convergence. This is especially true in cases where the solver requires many decision epochs to achieve the desired error tolerance.

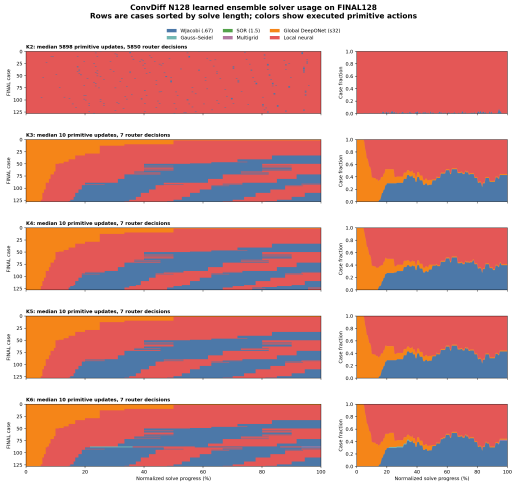

Figure 12: Solver usage by the learned router for growing convection-diffusion solver ensembles on 128 test instances. Rows correspond to instances sorted by solve length, and colors indicate the executed primitive actions over normalized solve progress. The right column reports the fraction of instances using each action at each point in the solve.

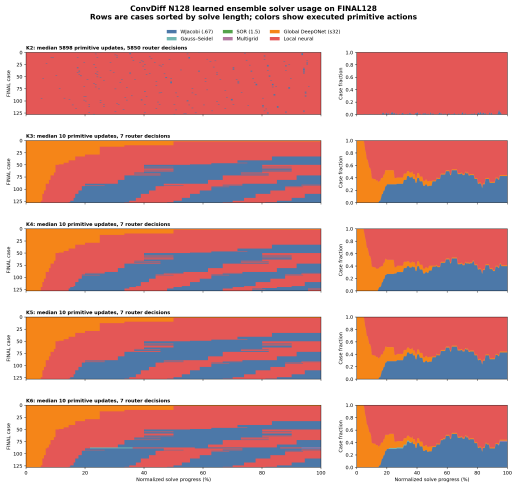

Figure 13: Solver usage by the learned router for growing anisotropic diffusion solver ensembles on 128 test instances. Rows correspond to instances sorted by solve length, and colors indicate the executed primitive actions over normalized solve progress. The right column reports the fraction of instances using each action at each point in the solve.

Figures 11 to 13 report solver selection patterns across different solver ensembles. The non-uniformity across samples is more apparent in these experiments. As expected, there seems to be a heavy reliance on 3 solvers, which explains the diminishing returns observed in wall-clock time with ensemble sizes larger than 3.

F.7 Results on a finer grid

We have replicated the Paired Solver experiments for a grid of 512×512.

In Table 19, we notice that Jacobi is anomalous at 512×512, with incomplete coverage for both HINTS and the learned router, whereas across GS, SymGS, SOR, and MG, the learned router consistently achieves the lowest deployable wall-clock time. Moreover, most standalone iterative solvers generally fail to reach the target accuracy at this resolution, highlighting the increasing advantage of hybrid routing on finer grids

Table 19: Selected-action wall-clock replay time to relative solution error 10^{-6} on the 512×512 grid (lower is better). Values are mean (standard error) in milliseconds over successful cases among 16 shared test instances; entries without brackets have 16/16 successes. Brackets report incomplete coverage, and a dash indicates no successes. The reported HINTS result uses the best observed cadence in $\{5, 10, 25, 50\}$ for each equation and solver family at this tolerance. One-sided paired t-tests compare each full-coverage deployable baseline with the learned router under the alternative that the router is faster; tests are omitted when either method lacks full coverage. Bold marks the fastest deployable method with full coverage, while True-Greedy is an oracle with access to the exact error.

Method	Poisson		ConvDiff	
	Time (ms)	p-value	Time (ms)	p-value
Jacobi-related solvers				
Jacobi Only	–	–	–	–
HINTS-Jacobi ($m = 50/50$)	2999.34 (1822.99) [14/16]	–	948.34 (302.28) [12/16]	–
Learned Greedy-Jacobi	302.78 (71.31) [12/16]	–	878.31 (305.26) [12/16]	–
True-Greedy-Jacobi	*2780.05 (1710.63) [14/16]*	–	*859.51 (293.84) [12/16]*	–
GS-related solvers				
GS Only	–	–	–	–
HINTS-GS ($m = 25/25$)	639.92 (62.71)	0.002	825.77 (64.83)	0.003
Learned Greedy-GS	**524.84 (91.17)**	–	**677.56 (102.12)**	–
True-Greedy-GS	*520.29 (90.21)*	–	*673.75 (102.22)*	–
SymGS-related solvers				
SymGS Only	–	–	–	–
HINTS-SymGS ($m = 10/10$)	529.52 (55.57)	$< 10^{-3}$	649.97 (55.09)	0.001
Learned Greedy-SymGS	**428.67 (73.93)**	–	**534.63 (79.04)**	–
True-Greedy-SymGS	*425.87 (73.70)*	–	*530.39 (78.59)*	–
SOR (1.5)-related solvers				
SOR (1.5) Only	–	–	–	–
HINTS-SOR (1.5) ($m = 10/10$)	333.37 (31.02)	$< 10^{-3}$	407.01 (30.25)	$< 10^{-3}$
Learned Greedy-SOR (1.5)	**259.79 (43.64)**	–	**322.55 (46.52)**	–
True-Greedy-SOR (1.5)	*258.26 (43.36)*	–	*319.34 (46.19)*	–
MG-related solvers				
MG Only	289.14 (10.25)	$< 10^{-3}$	344.36 (9.70)	$< 10^{-3}$
HINTS-MG ($m = 5/5$)	262.52 (5.12)	$< 10^{-3}$	314.69 (2.11)	$< 10^{-3}$
Learned Greedy-MG	**150.68 (22.59)**	–	**199.65 (24.74)**	–
True-Greedy-MG	*149.29 (22.22)*	–	*199.83 (24.62)*	–

F.8 Results with Dirichlet Boundary Conditions

We have replicated the Paired Solver experiments for solutions with zero dirichlet boundary conditions.

In Table 20, we notice that, the learned router generally provides the lowest deployable wall-clock time for Jacobi, GS, SymGS, and most SOR pairings, often closely tracking the true-greedy oracle. The advantage is not universal: selected HINTS is faster for all MG pairings and for ConvDiff–SOR,

indicating that effective fixed schedules can remain preferable when the constituent solver is already strong.

Table 20: Selected-action wall-clock replay time to relative solution error 10^{-6} with zero Dirichlet boundary conditions (lower is better). Values are mean (standard error) in milliseconds over successful cases among 64 shared test instances with 128×128 interior unknowns; entries without brackets have 64/64 successes. Brackets report incomplete coverage, and a dash indicates no successes. The reported HINTS result uses the best tested cadence in $\{5, 10, 25, 50\}$ for each equation and solver family on these test instances and is therefore an optimistic comparator. One-sided paired t-tests compare each full-coverage deployable baseline with the learned router under the alternative that the router is faster; tests are omitted when either method lacks full coverage. Bold marks the fastest deployable method with full coverage, while True-Greedy is an oracle with access to the exact error.

	Poisson		ConvDiff		Aniso	
Method	Time (ms)	p-value	Time (ms)	p-value	Time (ms)	p-value
Jacobi-related solvers						
Jacobi Only	–	–	1116.25 (8.57)	$< 10^{-3}$	–	–
HINTS-Jacobi ($m = 10/10/50$)	**599.86 (177.87)**	–	583.50 (109.39)	$< 10^{-3}$	454.80 (78.08)	$< 10^{-3}$
Learned Greedy-Jacobi	124.03 (37.49) [62/64]	–	**148.24 (23.96)**	–	**314.83 (52.80)**	–
True-Greedy-Jacobi	*148.39 (40.34)*	–	*144.40 (23.41)*	–	*299.83 (49.03)*	–
GS-related solvers						
GS Only	9160.73 (547.70) [6/64]	–	1495.98 (12.02)	$< 10^{-3}$	–	–
HINTS-GS ($m = 10/10/50$)	28.42 (0.52)	$< 10^{-3}$	32.08 (0.16)	$< 10^{-3}$	109.45 (1.53)	$< 10^{-3}$
Learned Greedy-GS	**20.80 (0.45)**	–	**30.55 (0.42)**	–	**90.78 (1.57)**	–
True-Greedy-GS	*20.10 (0.46)*	–	*30.20 (0.42)*	–	*88.62 (1.37)*	–
SymGS-related solvers						
SymGS Only	8879.15 (69.42)	$< 10^{-3}$	1247.99 (9.36)	$< 10^{-3}$	9090.66 (33.18)	$< 10^{-3}$
HINTS-SymGS ($m = 10/10/25$)	27.34 (0.20)	$< 10^{-3}$	28.67 (0.07)	0.023	93.81 (1.37)	$< 10^{-3}$
Learned Greedy-SymGS	**19.48 (0.36)**	–	**27.52 (0.59)**	–	**79.59 (1.18)**	–
True-Greedy-SymGS	*19.40 (0.41)*	–	*27.15 (0.59)*	–	*77.63 (1.12)*	–
SOR (1.5)-related solvers						
SOR (1.5) Only	3658.02 (27.28)	$< 10^{-3}$	455.89 (3.52)	$< 10^{-3}$	3739.55 (13.66)	$< 10^{-3}$
HINTS-SOR (1.5) ($m = 10/10/25$)	30.52 (0.41)	$< 10^{-3}$	**27.74 (0.48)**	0.999	52.93 (0.47)	$< 10^{-3}$
Learned Greedy-SOR (1.5)	**20.37 (0.41)**	–	29.28 (0.27)	–	**43.41 (0.64)**	–
True-Greedy-SOR (1.5)	*19.79 (0.37)*	–	*29.10 (0.30)*	–	*43.33 (0.61)*	–
MG-related solvers						
MG Only	11.14 (0.13)	0.949	14.74 (0.19)	0.987	314.56 (3.94)	$< 10^{-3}$
HINTS-MG ($m = 25/5/25$)	**11.09 (0.09)**	0.993	**14.73 (0.20)**	0.990	**125.08 (2.76)**	0.852
Learned Greedy-MG	11.40 (0.14)	–	15.13 (0.22)	–	126.90 (2.12)	–
True-Greedy-MG	*11.11 (0.09)*	–	*14.63 (0.17)*	–	*124.49 (2.12)*	–

G LIMITATIONS

A primary limitation of our approach is the higher on-time computational cost requires to learn the greedy routing strategy. In contrast, the simpler baselines such as HINTS incur no training overhead. While this cost is amortized during inference time and leads to improved solution quality, it may be prohibitive in settings with limited computational resources or when rapid deployment is required.

Additionally, the effectiveness of the learned router depends on the neural operator exhibiting heterogeneous performance across samples. When the neural operators produces predictions of relatively uniform quality, the benefit of adaptive routing diminishes and it is more practical to set a fixed schedule based on the quality of the neural operator. In practice, achieving such heterogeneity can be sensitive to training choices, and neural operators themselves can be finicky to train.