ONCE: OPTIMIZED NEURAL DOMAIN DECOMPOSITION FOR SCALABLE PDE SOLVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural surrogates enable efficient PDE solving for scientific discovery and engineering design, yet generalizing to larger domains and unseen geometries remains challenging due to the limited domain and geometric distributions observed during training. Domain decomposition methods (DDMs) provide a promising route to overcome this bottleneck by iteratively coupling the locally trained solver across subdomains. However, the local solver is typically trained under reference states and repeatedly evaluated on states generated by its own predictions. This training-inference mismatch can compromise global accuracy even when the iteration converges. To address these challenges, we present *ONCE*, a novel neural DDM framework for scalable PDE solving that unites optimized Schwarz methods (OSMs) with Trajectory-aligned Tuning (TAT). Specifically, TAT collects intermediate interface states from the pre-trained model rollouts on the original training cases and fine-tunes the local solver against the corresponding reference subdomain solutions, while retaining reference-state supervision to preserve consistency. This procedure requires neither additional numerical PDE solves nor backpropagation through the iterations. Our theoretical analysis characterizes fixed-point convergence of neural DDMs and establishes interface error bounds that account for trajectory supervision and distribution shift. Experimentally, ONCE consistently outperforms existing methods across benchmarks spanning varying domain scales and unseen geometries, demonstrating its superior accuracy and scalability.

1 INTRODUCTION

Partial differential equations (PDEs) provide the fundamental language for describing physical systems in fluid mechanics, solid mechanics, and many other scientific and engineering applications. While most PDEs lack analytical solutions, repeatedly solving them with numerical solvers can also be prohibitively expensive. To this end, neural operators have emerged as promising alternatives by learning the mapping from input to output using a parameterized deep neural model, which enables fast inference to get the solution once trained (Azizzadenesheli et al., 2024). Recent advances have improved their flexibility and accuracy to accommodate complex geometries (Wu et al., 2024; Li et al., 2023) or handle meshes at industrial-level simulations (Luo et al., 2025; Zhou et al., 2026). However, scaling neural operators to expansive, geometrically complex domains still remains fundamentally challenging. A primary bottleneck is that these models are inherently constrained by the spatial domain and geometric distributions observed during training, leaving them prone to failure when evaluated on unseen domain scales or arbitrary geometries.

To overcome this generalization issue, a natural idea is to adopt the divide-and-conquer principle from domain decomposition methods (DDMs) (Mathew, 2008). DDMs partition a global PDE problem into smaller subdomains and recover the global solution by iteratively exchanging information across artificial interfaces among subdomains. This local-to-global paradigm provides a route to spatial scalability, as the same local solver can be reused to solve substantially larger domains. Recent studies have therefore explored replacing conventional numerical solvers at each subdomain with neural operators, combining the efficiency of learned surrogates with the scalability of classical DDMs (Mao et al., 2024; Huang et al., 2026; Wu et al., 2026). Such methods, collectively known as neural DDMs, have demonstrated potential in scaling neural surrogates to large, complex domains.

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.

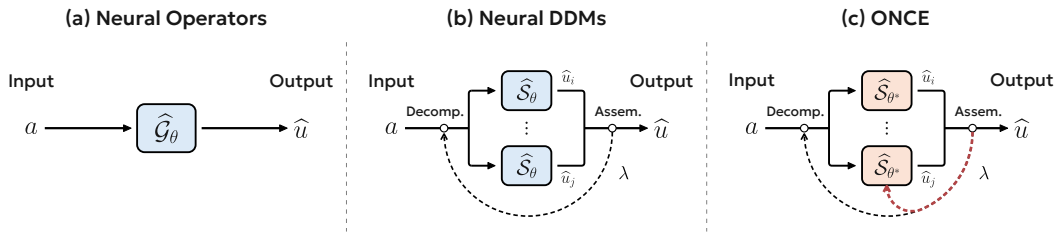


Figure 1: Comparison of neural operators, neural DDMs, and ONCE. (a) Neural operator maps the PDE input a directly to the global solution. (b) Neural DDMs reuse a local neural operator for parallel subdomain solves, exchanging interface states λ through the dashed loops. (c) ONCE involves trajectory awareness learned through TAT on intermediate interface states. The tuned parameters remain fixed during inference.

However, existing approaches typically train the learned local solver only on precomputed boundary-solution pairs either generated by solving synthetic local problems (Huang et al., 2026; Wu et al., 2026) or extracted from converged global solutions (Mao et al., 2024). During inference, the same solver is repeatedly evaluated at intermediate interface states generated from its previous predictions. The solver is therefore trained and evaluated under different input distributions. Consequently, low prediction error at reference interface states does not guarantee convergence to an accurate global solution during the inference stage of DDMs. Instead, the learned iteration may converge to a self-consistent but biased fixed point.

Motivated by this observation, we introduce *ONCE*, an **o**ptimized **n**eural domain de**c**omposition framework for scalab**l**e solving of PDEs. ONCE combines optimized Schwarz methods (OSMs) with a learned local solver pre-trained on subdomain input-solution pairs extracted from existing global training cases, avoiding the need to generate separate local training problems. Also, the proposed framework supports both structured grids and unstructured meshes. The key design within ONCE is *Trajectory-aligned Tuning (TAT)*. As shown in Fig. 1(c), starting from a pre-trained local solver, TAT runs the DDM iteration on the original training cases with the solver's parameters fixed. Then, the intermediate interface states are collected and paired with the reference subdomain solution. The learned local solver is subsequently fine-tuned on these data. In this way, ONCE exposes the learned local solver to the interface states encountered at inference without requiring additional numerical PDE solving.

We summarize our contributions below:

- We introduce TAT to mitigate the mismatch between the reference interface states used for training the local solver and the intermediate interface states encountered during inference.
- We present ONCE, a flexible and scalable framework combining TAT with OSM to improve both accuracy and convergence in large-scale PDE solving.
- We theoretically analyze the fixed-point convergence of neural DDMs and empirically validate the efficacy of ONCE across 2D and 3D elliptic PDEs as well as complex physical simulations.

2 Preliminaries

2.1 Problem Formulation

We consider the systems of steady-state boundary value problems (BVPs) defined on bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$. Let $a \in \mathcal{A}$ denote the problem-specific input parameters, such as PDE coefficients, geometry, etc. Our aim is to find the solution $u \in \mathcal{U}(\Omega)$ that satisfies the following PDE and boundary conditions:

$$\mathcal{L}(u; a) = f, \quad \text{in } \Omega, \qquad \mathcal{B}(u; a) = g, \quad \text{on } \partial\Omega, \tag{1}$$

where $\mathcal{L}$ denotes the differential operator for the unknown function u, and $\mathcal{B}$ denotes the boundary operator imposing the physical boundary conditions (Dirichlet, Neumann, or Robin depending on the problem) on the respective parts of $\partial\Omega$, f is a source term, and g is the imposed boundary value.

Learning the Global Solution Operator. We assume that Eq. (1) is well posed for every input $a \in \mathcal{A}$ and thus have a unique solution $u^*(a)$. This defines the global solution operator $\mathcal{G}$ where:

$$\mathcal{G} : \mathcal{A} \to \mathcal{U}(\Omega), \qquad a \longmapsto u^*(a) = \mathcal{G}(a). \tag{2}$$

Note that $\mathcal{G}$ takes the complete specification a as input and outputs the solution on the entire domain Ω. Under this formulation, conventional methods (Li et al., 2021; Wu et al., 2024) aim to learn a parameterized approximation $\widehat{\mathcal{G}}_\theta$ of $\mathcal{G}$ from input-output data pairs $\{(a, u^*(a))\}$ generated by numerical solvers such that $\widehat{\mathcal{G}}_\theta(a) \approx \mathcal{G}(a)$ for all $a \in \mathcal{A}$, where θ denotes the trainable model parameters. However, in practice, it is challenging to obtain sufficient data with respect to domain size, geometry, and boundary conditions. Thus, the scalability and accuracy of $\widehat{\mathcal{G}}_\theta$ beyond the training distribution is not guaranteed.

2.2 Neural DDMs

To overcome the limitation of training $\widehat{\mathcal{G}}_\theta$, the principle of DDMs can be incorporated into the operator learning framework. Below, we first describe the general problem formulation of neural DDMs and rewrite it as a fixed-point iteration to further analyze it's convergence. Then, we state how the local operator $\widehat{\mathcal{S}}_\theta$ is trained and used for inference to recover the global solution.

Local Subdomain Problems. Instead of approximating $\mathcal{G}$ directly, we first formulate the problem as learning a local solution operator $\widehat{\mathcal{S}}_\theta$ for each subdomain. The global domain Ω is decomposed into N smaller subdomains $\Omega_1, \Omega_2, \ldots, \Omega_N$, such that $\Omega = \bigcup_{i=1}^N \Omega_i$. For each subdomain Ω_i, define its artificial and physical boundaries as $\Gamma_i^{\text{art}} = \partial\Omega_i \setminus \partial\Omega$, $\Gamma_i^{\text{phy}} = \partial\Omega_i \cap \partial\Omega$. We retain the original physical boundary conditions on Γ_i^{phy} and impose a transmission condition on Γ_i^{art}. The resulting local problem is:

$$\mathcal{L}(u_i; a_i) = f, \text{in } \Omega_i, \qquad \mathcal{B}(u_i; a_i) = g, \text{on } \Gamma_i^{\text{phy}}, \qquad \mathcal{B}_i u_i = \lambda_i, \text{on } \Gamma_i^{\text{art}}, \tag{3}$$

where $\mathcal{B}_i$ specifies the transmission condition on the artificial boundary Γ_i^{art}, and λ_i denotes the interface state updated during iterations. Here, a_i denotes the local problem specification restricted to Ω_i, including local geometry, PDE parameters $a|_{\Omega_i}$, and physical boundary data on Γ_i^{phy}. For example, we can use Dirichlet transmission condition that prescribes $u_i = \lambda_i$ on Γ_i^{art}. For a global problem with fixed input a, we assume that each local BVP is uniquely solvable and define the exact local solution operator $\mathcal{S}$ such that $u_i = \mathcal{S}_i(\lambda_i)$ for all $i = 1, 2, \ldots, N$. Under DDMs, the global solution will be assembled by exchanging information across subdomain interfaces and iteratively updating the local solutions until global convergence. In this work, we learn to approximate $\mathcal{S}$ by a local operator $\widehat{\mathcal{S}}_\theta$.

Schwarz Methods as Fixed-Point Iterations. In classical Schwarz methods, interface conditions on Γ_i^{art} are iteratively updated using neighboring solutions until convergence (Gander, 2006). Let Λ denote the space of interface states $\lambda = (\lambda_1, \ldots, \lambda_N)$ equipped with norm $\|\cdot\|_\Lambda$. The transmission operator $\mathcal{C}$ maps local solutions $(u_1, \ldots, u_N)$ to interface states for the subsequent step. Thus, the Schwarz iteration can be formulated as a fixed-point map on Λ:

$$\lambda^{(k+1)} = \mathcal{T}(\lambda^{(k)}), \qquad \mathcal{T} \triangleq \mathcal{C} \circ \mathcal{S}. \tag{4}$$

When local problems are well posed, the reference solution $u^*(a)$ induces a unique interface state λ^* satisfying $\mathcal{T}(\lambda^*) = \lambda^*$, from which $\mathcal{S}(\lambda^*)$ recovers the global solution. The iteration convergence is determined by the contraction factor ρ of $\mathcal{T}$, where $\|\mathcal{T}(\lambda) - \mathcal{T}(\mu)\|_\Lambda \le \rho\|\lambda - \mu\|_\Lambda$ (Mathew, 2008).

Training and Inference. Training $\widehat{\mathcal{S}}_\theta$ requires boundary–solution pairs on subdomains. Instead of independently solving local BVPs with synthetic boundary conditions that often suffers from ill-posedness and numerical divergence in engineering applications, we directly restrict global reference solutions $u^*(a)$ to subdomains, extracting $u_i^* = u^*|_{\Omega_i}$ and $\lambda_i^* = \mathcal{B}_i u_i^*$ on Γ_i^{art}. With samples $(a_i, \lambda_i^*, u_i^*)$, we train a shared local operator $\widehat{\mathcal{S}}_\theta$ by minimizing the reference-state loss:

$$\mathcal{L}_{\text{ref}}(\theta) = \sum_{i=1}^N \left\| u_i^* - \widehat{\mathcal{S}}_\theta(\lambda_i^*, a_i) \right\|^2. \tag{5}$$

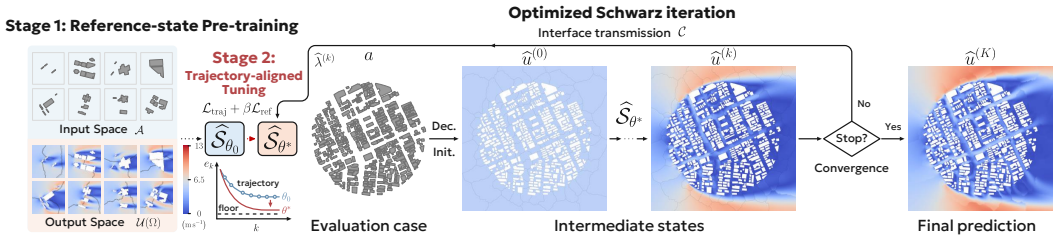


Figure 2: Overview of ONCE. **Stage 1:** a local operator is pre-trained on reference-state subdomain pairs extracted from training cases. **Stage 2:** with the pre-trained operator fixed, interface trajectories are collected within same training sets. TAT then fine-tunes the operator on these intermediate states using the corresponding reference subdomain solutions as targets. At inference, the tuned operator is reused within optimized Schwarz iteration from $\widehat{u}^{(0)}$ to $\widehat{u}^{(K)}$, enabling scalable solving of PDEs.

At inference time, starting from an initial $\widehat{\lambda}^{(0)}$, the fixed operator $\widehat{\mathcal{S}}_\theta$ is iteratively evaluated with transmission conditions:

$$\begin{aligned} \widehat{u}_i^{(k+1)} &= \widehat{\mathcal{S}}_\theta\big(\widehat{\lambda}_i^{(k)}, a_i\big), \\ \widehat{\lambda}^{(k+1)} &= \mathcal{C}\left((\widehat{u}_i^{(k+1)})_{i=1}^N\right) = \widehat{\mathcal{T}}_\theta\big(\widehat{\lambda}^{(k)}\big), \qquad \widehat{\mathcal{T}} = \mathcal{C} \circ \widehat{\mathcal{S}}_\theta, \end{aligned} \tag{6}$$

for $k = 0, \ldots, K-1$. However, this iterative inference produces out-of-distribution interface states $\widehat{\lambda}^{(k)}$ unseen during reference-state training (λ_i^*), thus degrading multi-step prediction accuracy.

3 ONCE: Optimized Neural Domain Decomposition

As illustrated in Fig. 2, ONCE couples optimized Schwarz methods with a two-stage training scheme: the local operator is first pre-trained on reference-state subdomain samples and then tuned via trajectory-aligned tuning on intermediate interface states. At inference, the local operator is iteratively applied to recover global solutions. Below, we describe the Robin transmission conditions and TAT within ONCE, followed by an analysis of the fixed-point convergence of neural DDMs and its connection to local supervision.

3.1 Optimized Schwarz Methods

In this work, we adopt OSMs with Robin transmission conditions (Gander, 2006), which combine solution values with normal-derivative information at subdomain interfaces. Specifically, for each subdomain Ω_i, we construct $\mathcal{B}_i v = \frac{\partial v}{\partial n_i} + \gamma_i v$ ($\gamma_i > 0$) on Γ_i^{art}, where $\mathbf{n}_i$ is the outward unit normal of Ω_i, and γ_i is a transmission parameter balancing the contribution of values and gradients. To describe the classical OSM update, let $\Gamma_{ij} \subseteq \Gamma_i^{\text{art}} \cap \Omega_j$ denote an artificial boundary portion receiving information from subdomain Ω_j. The local problems are coupled by:

$$\frac{\partial u_i^{(k+1)}}{\partial n_i} + \gamma_i u_i^{(k+1)} = \frac{\partial u_j^{(k)}}{\partial n_i} + \gamma_i u_j^{(k)}, \qquad \text{on } \Gamma_{ij}. \tag{7}$$

Here, normal derivatives are evaluated with respect to $\mathbf{n}_i$ across the interface (Fig. 3). Concretely, we employ a restricted additive Schwarz (RAS) iteration (Cai & Sarkis, 1999) with Robin transmission conditions. At each iteration, the local solver is evaluated in parallel on overlapping subdomains. The transmission operator $\mathcal{C}$ first assembles the global iterate by retaining local predictions on non-overlapping cores, and then extracts Robin interface data from this assembled field for the next iteration.

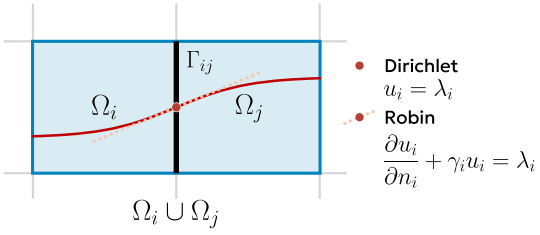


Figure 3: Dirichlet and Robin transmissions. The red dot represents the function value, and the dashed line indicates the normal derivative.

3.2 TRAJECTORY-ALIGNED TUNING

As discussed at the end of Section 2.2, the local solver is typically trained using reference interface states as inputs. During the iterative inference in neural DDMs, however, each local prediction is conditioned on an intermediate interface state generated by the preceding neural predictions. ONCE addresses this discrepancy through TAT, which extends supervision to the interface states encountered by the learned iteration. We present the details of TAT below.

Collecting Interface Trajectories. Let θ_0 denote the parameters of the pre-trained local solver. Holding θ_0 fixed, we unroll the learned Schwarz iteration on the original training problems for K steps: $\widehat{\lambda}_{\theta_0}^{(k+1)} = \widehat{\mathcal{T}}_{\theta_0}(\widehat{\lambda}_{\theta_0}^{(k)})$ for $k = 0, \ldots, K-1$. We uniformly sample n_s distinct interface states from each recorded trajectory. For each subdomain Ω_i, the intermediate states $\widehat{\lambda}_{\theta_0,i}^{(k)}$ are collected and paired with the reference solution u_i^*, yielding samples of the form $\left(a_i,\, \widehat{\lambda}_{\theta_0,i}^{(k)},\, u_i^*\right)$. This requires neither additional numerical PDE solves nor larger-scale training data.

Trajectory Supervision. For a given problem, we represent the sampling strategy by probabilities $p_k \geq 0$ over the recorded interface states at iteration k during tuning, with $\sum_{k=0}^{K-1} p_k = 1$ and $p_k = 0$ for states not retained in the tuning set. We define the trajectory loss as:

$$\mathcal{L}_{\mathrm{traj}}(\theta; \theta_0) = \sum_{k=0}^{K-1} p_k \sum_{i=1}^{N} \left\| u_i^* - \widehat{\mathcal{S}}_\theta\big(\widehat{\lambda}_{\theta_0,i}^{(k)}, a_i\big) \right\|_i^2, \tag{8}$$

where θ denotes the parameters being optimized, while θ_0 identifies the model that generated these intermediate interface inputs. Intuitively, this objective makes the learned local solver aware of the convergence trajectory, directly training it to correct intermediate interface errors and steer iterates toward the reference solution. We theoretically formalize how this local trajectory supervision controls subsequent interface errors in Section 3.3.

Consistency Regularization. Trajectory supervision expands the set of interface inputs seen during training to include intermediate states encountered during iterative inference. However, we observe that fine-tuning exclusively on these intermediate states degrades the consistency of the local solver's prediction at the reference interface states. To preserve the prediction accuracy while learning from iterative trajectories, we retain the reference-state loss as a consistency regularizer in the TAT loss:

$$\mathcal{L}_{\mathrm{TAT}}(\theta; \theta_0) = \mathcal{L}_{\mathrm{traj}}(\theta; \theta_0) + \beta \mathcal{L}_{\mathrm{ref}}(\theta), \quad \beta > 0, \tag{9}$$

where β is the consistency weight that controls the relative strength of reference-state supervision. Starting from θ_0, we obtain the tuned parameters via:

$$\theta^* = \arg\min_\theta \mathbb{E}_a \left[\mathcal{L}_{\mathrm{TAT}}(\theta; \theta_0) \right]. \tag{10}$$

The intermediate interface states $\widehat{\lambda}_{\theta_0,i}^{(k)}$ are treated as fixed training inputs, without backpropagation through trajectory generation. After tuning, the local operator $\widehat{\mathcal{S}}_{\theta^*}$ can be used in the original DDM inference, with the same decomposition and transmission. The training and inference procedure of ONCE is summarized in Algorithm 1 in Appendix D.

3.3 THEORETICAL ANALYSIS

We now analyze the fixed-point convergence of neural DDMs and connect the iteration error bounds to the proposed TAT. For a given PDE problem with domain decomposition, we track the interface error $e_k \triangleq \|\widehat{\lambda}_\theta^{(k)} - \lambda^*\|_\Lambda$ and the reference consistency error $\varepsilon_{\mathrm{ref}}(\theta) \triangleq \|\widehat{\mathcal{T}}_\theta(\lambda^*) - \lambda^*\|_\Lambda$, which quantifies the violation of the fixed-point condition $\widehat{\mathcal{T}}_\theta(\lambda^*) = \lambda^*$.

Theorem 1 (Convergence with fixed-point bias)**.** *Let Λ be a Banach space and $D \subseteq \Lambda$ be a closed set containing $\lambda^*, \widehat{\lambda}_\theta^{(0)}$ with $\widehat{\mathcal{T}}_\theta(D) \subseteq D$. If $\widehat{\mathcal{T}}_\theta$ is contractive on D with factor $\rho_\theta \in [0, 1)$, i.e., $\|\widehat{\mathcal{T}}_\theta(\lambda) - \widehat{\mathcal{T}}_\theta(\lambda')\|_\Lambda \leq \rho_\theta \|\lambda - \lambda'\|_\Lambda$ for $\lambda, \lambda' \in D$, then the iteration converges to a unique fixed*

point $\widehat{\lambda}_\theta^* \in D$. Furthermore, for any $K \geq 1$, the finite-iteration error and asymptotic bias satisfy:

$$e_K \leq \rho_\theta^K e_0 + \frac{1-\rho_\theta^K}{1-\rho_\theta}\,\varepsilon_{\mathrm{ref}}(\theta), \qquad \left\|\widehat{\lambda}_\theta^* - \lambda^*\right\|_\Lambda \leq \frac{\varepsilon_{\mathrm{ref}}(\theta)}{1-\rho_\theta}. \tag{11}$$

Theorem 1 distinguishes two distinct mechanisms governing neural DDMs: contractivity ($\rho_\theta < 1$) guarantees iterative stability and geometric convergence to an internal fixed point $\widehat{\lambda}_\theta^*$, whereas the consistency error $\varepsilon_{\mathrm{ref}}(\theta)$ dictates whether this fixed point aligns with the true solution. A strictly contractive neural DDM may converge reliably, yet converge to a biased solution if $\varepsilon_{\mathrm{ref}}(\theta) > 0$. Here, ρ_θ represents a contraction bound for the learned interface map on D, distinct from the classical numerical contraction factor of exact Schwarz iterations.

Connecting Convergence to Local Supervision. To ground the fixed-point bias in empirical loss functions, we require the transmission operator $\mathcal{C}$ to preserve the reference interface state and stably propagate local field errors:

Assumption 1 (Consistent and stable interface transmission)**.** *The reference subdomain solutions satisfy $\mathcal{C}((u_i^*)_{i=1}^N) = \lambda^*$. Furthermore, there exists a stability constant C_{tr} such that the transmission operator is Lipschitz continuous:*

$$\left\|\mathcal{C}\big((v_i)_{i=1}^N\big) - \mathcal{C}\big((w_i)_{i=1}^N\big)\right\|_\Lambda \leq C_{\mathrm{tr}} \left(\sum_{i=1}^N \|v_i - w_i\|_i^2\right)^{1/2}. \tag{12}$$

The local discrete norms $\|\cdot\|_i$ correspond to those in the training objectives, where C_{tr} may depends on the spatial discretization, domain partitioning, and transmission parameters.

Proposition 1 (Local prediction errors control interface errors)**.** *Under Assumption 1, the updated interface state $\widehat{\mathcal{T}}_\theta(\lambda)$ for any input λ satisfies:*

$$\left\|\widehat{\mathcal{T}}_\theta(\lambda) - \lambda^*\right\|_\Lambda^2 \leq C_{\mathrm{tr}}^2 \sum_{i=1}^N \left\|u_i^* - \widehat{\mathcal{S}}_\theta(\lambda_i, a_i)\right\|_i^2. \tag{13}$$

In particular, evaluating at the reference interface state yields:$\varepsilon_{\mathrm{ref}}(\theta) \leq C_{\mathrm{tr}}\sqrt{\mathcal{L}_{\mathrm{ref}}(\theta)}$.

Combining Proposition 1 with Theorem 1 gives

$$\left\|\widehat{\lambda}_\theta^* - \lambda^*\right\|_\Lambda \leq \frac{C_{\mathrm{tr}}\sqrt{\mathcal{L}_{\mathrm{ref}}(\theta)}}{1-\rho_\theta}. \tag{14}$$

Under the stated assumptions, Eq. (14) bounds the asymptotic fixed-point bias in terms of the reference-state loss and the contraction factor. While $\mathcal{L}_{\mathrm{ref}}(\theta)$ supervises predictions only at reference interface states, TAT extends this supervision to intermediate interface states collected from the learned iteration.

Trajectory Supervision in ONCE. We now apply Proposition 1 to the objective of TAT. Let θ_0 denote the parameters used to collect interface trajectories and θ denote the fine-tuned parameters. Under Assumption 1, using the weights p_k defined in Section 3.2, for any $\beta \geq 0$,

$$\sum_{k=0}^{K-1} p_k \left\|\widehat{\mathcal{T}}_\theta\big(\widehat{\lambda}_{\theta_0}^{(k)}\big) - \lambda^*\right\|_\Lambda^2 + \beta\,\varepsilon_{\mathrm{ref}}(\theta)^2 \leq C_{\mathrm{tr}}^2 \mathcal{L}_{\mathrm{TAT}}(\theta;\theta_0). \tag{15}$$

This bound highlights the complementary design of $\mathcal{L}_{\mathrm{TAT}}$: trajectory supervision constrains intermediate inference steps along the collected rollout, while reference-state regularization penalizes consistency drift at the target interface. However, updating $\theta_0 \to \theta$ inherently shifts the induced inference trajectory away from the intermediate states. We characterize this transfer discrepancy below:

Proposition 2 (Error bound under trajectory distribution shift)**.** *Suppose Assumption 1 holds. Consider the recorded trajectory $\widehat{\lambda}_{\theta_0}^{(k)}$ and the fine-tuned trajectory $\widehat{\lambda}_\theta^{(k)}$ over K iterations, initialized from the same state $\widehat{\lambda}^{(0)}$. We can define the trajectory discrepancy:*

$$B_\theta = \left(\sum_{k=0}^{K-1} p_k \left\|\widehat{\mathcal{T}}_\theta\big(\widehat{\lambda}_\theta^{(k)}\big) - \widehat{\mathcal{T}}_\theta\big(\widehat{\lambda}_{\theta_0}^{(k)}\big)\right\|_\Lambda^2\right)^{1/2}. \tag{16}$$

Then the expected iteration error of the fine-tuned model is bounded by:

$$\left(\sum_{k=0}^{K-1} p_k e_{k+1}(\theta)^2\right)^{1/2} \leq C_{\mathrm{tr}}\sqrt{\mathcal{L}_{\mathrm{traj}}(\theta;\theta_0)} + B_\theta. \tag{17}$$

This bound provides theoretical motivation for the design of TAT. By initializing tuning from the pre-trained weight θ_0 with moderate updates, the trajectory discrepancy B_θ remains controlled, while the consistency penalty $\beta\mathcal{L}_{\mathrm{ref}}$ prevents accuracy degradation on reference interface states, allowing $\mathcal{L}_{\mathrm{traj}}$ to effectively suppress trajectory-level error propagation. Importantly, while Theorem 1 establishes asymptotic stability under contractivity, Propositions 1 and 2 hold purely via transmission stability and the triangle inequality, providing non-asymptotic error bounds across finite iterations without requiring contractivity. Full proofs are provided in Appendix C.

4 Experiments

We evaluate ONCE across 2D/3D elliptic PDEs and complex physical simulations on both structured grids and unstructured meshes, focusing on accuracy, scalability to larger domains, and generalization to unseen geometries. Comprehensive experimental setups and extended results are deferred to Appendices F and E.

4.1 Setup

Selection of Local Operator. ONCE places no restriction on the architecture of $\widehat{\mathcal{S}}_\theta$: any operator architecture that maps the local specification a_i and the interface state λ_i to the subdomain field can be used. In this study, we adopt Transolver (Wu et al., 2024) as local solver due to its linear complexity and flexibility with accuracy across irregular geometries.

Benchmarks. We evaluate ONCE on four benchmarks. For 2D *Poisson* and *Darcy*, we follow BENO (Wang et al., 2024) and test on in-scale domains and larger domains with twice (medium) and four times (large) the in-scale length. We introduce *HeatSink-3D* for steady-state heat conduction, with in-scale and OOD tests varying size and geometry, and *UrbanRANS-2D* for steady-state urban wind flow on in-scale, medium, and large domains. Dataset examples and details are provided in Appendix G (Fig. 14).

Baselines. We compare ONCE with both global neural operators and neural DDMs. Global baselines include BENO (Wang et al., 2024), Geo-FNO (Li et al., 2023), GNOT (Hao et al., 2023), and Transolver (Wu et al., 2024). For neural DDMs, we adopt SNI (Huang et al., 2026) as it can also handle various geometries. Specifically, we retain its local solver GNOT and additive Schwarz iteration for global coupling. Instead of independently generating local simulations as in the original SNI framework, we train its local solver on subdomain samples extracted from the same global reference solutions used by ONCE.

Implementation Details. Following the original Transolver design (Wu et al., 2024), we set the number of layers as 8, the hidden width as 128 or 256, and the number of slices as 32 for all the benchmarks. For a fair comparison, we report all main results without using data augmentations. All the experiments are conducted on two NVIDIA A100 GPUs with 80 GB memory.

4.2 Main Results

Overall Results. Table 1 presents the relative L^2 errors across four benchmarks. For in-scale evaluations, neural DDMs generally outperform the neural operators on *Poisson*, *Darcy*, and *HeatSink-3D*. ONCE further reduces the errors by approximately 47%, 55%, and 57%, respectively, compared with SNI. On *UrbanRANS-2D* that contains more complex geometries, Transolver demonstrates strong in-scale performance, and ONCE achieves a comparable error. On the OOD evaluations, ONCE consistently achieves the lowest errors across all benchmarks, demonstrating its superior generalization across domain scales and unseen geometries.

Table 1: Relative L^2 errors across four benchmarks (lower is better). In-scale denotes evaluation on test domains at scales represented in training. The remaining columns evaluate generalization to larger domains and/or unseen geometries. The best and second-best results are highlighted in **bold** and underlined, respectively.

	Poisson			Darcy			HeatSink-3D		UrbanRANS-2D		
Method	In-scale	Medium	Large	In-scale	Medium	Large	In-scale	OOD	In-scale	Medium	Large
Neural operators											
Geo-FNO (Li et al., 2023)	0.029	1.066	6.314	0.152	1.902	12.707	0.196	0.421	0.110	66.011	82.301
GNOT (Hao et al., 2023)	0.079	0.793	1.025	0.184	0.801	1.023	0.326	0.921	0.152	2.208	4.649
BENO (Wang et al., 2024)	0.192	1.031	1.054	0.341	1.028	1.088	0.556	0.733	0.107	0.351	0.524
Transolver (Wu et al., 2024)	0.048	0.874	0.991	0.144	0.847	1.008	0.235	0.647	0.079	0.279	0.492
Neural DDMs											
SNI (Huang et al., 2026)	0.017	0.049	0.333	0.047	0.142	0.450	0.084	0.107	0.234	0.320	0.466
ONCE (ours)	**0.009**	**0.024**	**0.262**	**0.021**	**0.132**	**0.409**	**0.036**	**0.047**	**0.078**	**0.251**	**0.396**

Scalability and Geometry Generalization. Prediction errors generally increase with domain size as shown in Fig. 4. For neural operators, the errors grow rapidly because these models are constrained by the domain distributions observed during training. Geo-FNO degenerates catastrophically on larger domains as its learned coordinate deformation fails to extrapolate to unseen domains. Neural DDMs alleviate this limitation, with ONCE achieving the lowest errors across all larger-scale and OOD evaluations. Especially for *HeatSink-3D*, ONCE maintains almost invariant accuracy (see Appendix E.7).

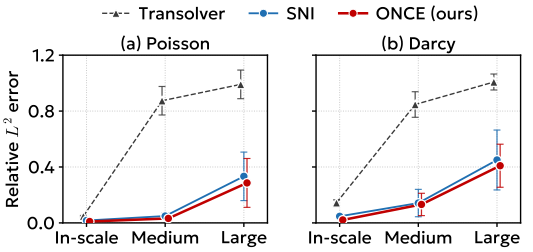


Figure 4: Relative L^2 errors versus domain size on (a) *Poisson* and (b) *Darcy*.

Visualization. Fig. 5 visualizes predictions across benchmarks. On *Poisson* and *Darcy*, Transolver exhibits severe boundary and interior errors on larger domains, whereas neural DDMs respect local boundary conditions, with ONCE yielding the lowest interior error. On the heat-conduction and urban wind flow benchmarks, ONCE demonstrates superior spatial fidelity, accurately capturing fin temperature distributions and intricate building-wake structures (see Appendix E.7 for more visualizations).

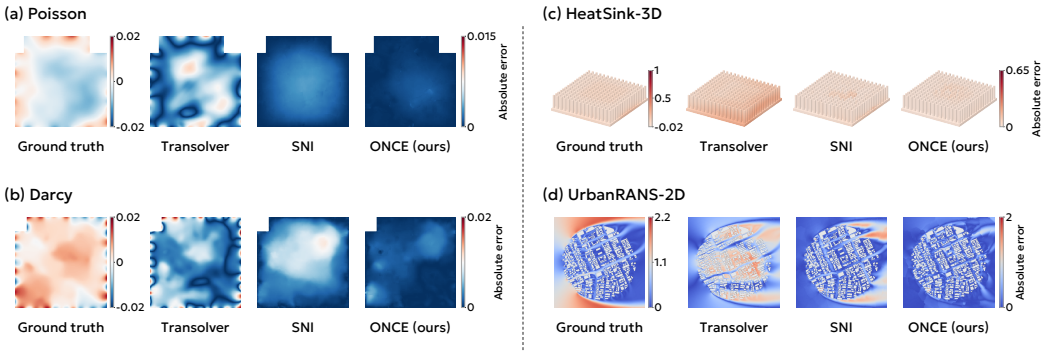


Figure 5: Visualization on four benchmarks. For each benchmark, the first column shows the ground-truth field; the remaining columns show pointwise absolute errors of Transolver, SNI, and ONCE.

4.3 Analysis and Ablations

We conduct extensive analysis and ablation studies to investigate the key properties and effectiveness of ONCE, with additional results deferred to Appendix E.

Effects of TAT and Transmission Conditions. As presented in Table 2, replacing classical Dirichlet transmission with Robin conditions consistently improves scalability on both *Poisson* and *UrbanRANS-2D*, while maintaining comparable in-scale accuracy. Building upon Robin transmission, TAT further reduces iterative prediction errors across all domain scales, achieving up to 19.7% and 15.9% error reductions on large *Poisson* and *UrbanRANS-2D* domains, respectively. This confirms the complementary benefits of Robin transmission and TAT in improving iterative neural DDMs.

Table 2: Ablation study on transmission conditions and TAT across domain scales. Relative L^2 errors are reported.

		Poisson			**UrbanRANS-2D**		
Transmission condition	TAT	In-scale	Medium	Large	In-scale	Medium	Large
Dirichlet	×	0.0100	0.0382	0.3348	0.0978	0.3199	0.4720
Robin	×	0.0108	0.0262	0.3262	0.0833	0.2947	0.4713
Robin	✓	**0.0090**	**0.0238**	**0.2619**	**0.0782**	**0.2511**	**0.3964**

Tuning Strategies. Under identical partitions and iteration budgets for training setup, we compare TAT against reference-only tuning and noise-injected fine-tuning inspired by (Sanchez-Gonzalez et al., 2020). Here, reference-only tuning continues training θ_0 exclusively on ground-truth reference pairs under the same optimization setups, isolating the benefit of trajectory states from mere additional training. As shown in Fig. 6a, even when the interface update drops close to zero, the pre-trained models without fine-tuning remain stuck at larger solution errors, which consistent with the biased fixed-point behavior described in Theorem 1. Among all the tuning strategies, TAT achieves the lowest terminal error. (Extended comparisons are detailed in Appendix E.2). Furthermore, Fig. 6(b) demonstrates that TAT aligns the latent feature distributions between reference and inference states. Specifically, a linear probe classifier distinguishing the two distributions on held-out layouts sees its AUC drop from 0.90 to 0.73, directly verifying the mitigation of inference distribution shift.

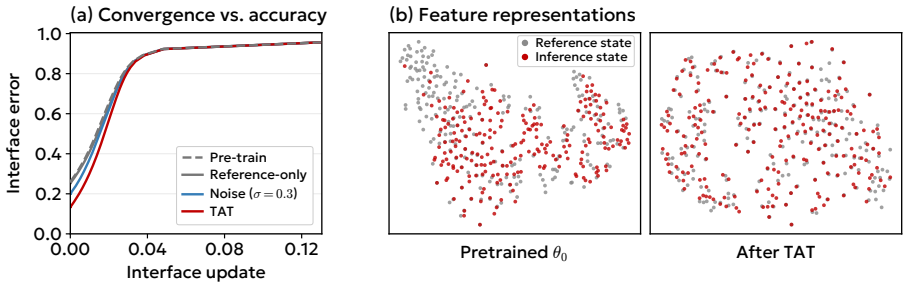


Figure 6: Comparison of tuning strategies. **(a)** Normalized interface error versus update on a large *Poisson* case. **(b)** t-SNE embeddings of node-averaged final-layer features on *UrbanRANS-2D* cases.

5 Conclusion and Discussion

This paper presents *ONCE*, a neural domain decomposition framework that unites optimized Schwarz methods with the inference efficiency of neural operators. Through Trajectory-aligned Tuning (TAT), ONCE mitigates the training-inference distribution shift without requiring additional numerical PDE solves. Our theoretical analysis formalizes the interplay between iterative convergence and fixed-point bias, while experiments across benchmarks demonstrate improved accuracy and generalization to larger domains and unseen geometries.

Despite these gains, the cumulative runtime remains a practical limitation for real-time inference (see Appendix E.1). In the future, we will focus on accelerating iterative convergence to reduce cumulative latency, and extending ONCE to more complex and larger 3D physical applications such as city-scale wind simulations.

AI USE STATEMENT

In this work, we used generative AI tools for manuscript editing and presentation, implementing methods (code development and debugging), assisting in the writing of proofs. We have not used generative AI tools to generate synthetic data sets, interpret results, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist with translation, design or provide feedback on research methodology or experiments, propose or refine hypotheses, clean and reformat dataset or supporting qualitative and thematic data analysis. All AI-assisted content was reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we provide comprehensive theoretical, algorithmic, and experimental details across the manuscript and supplementary materials:

- **Theoretical Guarantees:** Complete statements of assumptions and proofs for the convergence of neural DDMs are provided in Appendix C.
- **Algorithmic Details:** The complete two-stage training workflow and iterative inference procedure are summarized in Algorithm 1 (Appendix D).
- **Benchmarks:** All datasets and benchmark creation are documented in Section 4 and Appendix G.
- **Hyperparameters:** Detailed configurations are summarized in Table 6 and Appendix F.
- **Code & Supplementary Materials:** The source code, including training and evaluation pipelines, is provided in the supplementary materials and will be made publicly available upon publication.

REFERENCES

Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024. doi: 10.1038/s42254-024-00712-5.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28, pp. 1171–1179, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/e995f98d56967d946471af29d7bf99f1-Abstract.html.

Johannes Brandstetter, Max Welling, and Daniel E. Worrall. Lie point symmetry data augmentation for neural PDE solvers. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2241–2256. PMLR, 2022a. URL https://proceedings.mlr.press/v162/brandstetter22a.html.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=vSix3HPYKSU.

Xiao-Chuan Cai and Marcus Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing*, 21(2):792–797, 1999. doi: 10.1137/S106482759732678X.

Martin J. Gander. Optimized Schwarz methods. *SIAM Journal on Numerical Analysis*, 44(2):699–731, 2006. ISSN 0036-1429, 1095-7170. doi: 10.1137/S0036142903425409. URL http://epubs.siam.org/doi/10.1137/S0036142903425409.

Suresh V. Garimella, Amy S. Fleischer, Jayathi Y. Murthy, Ali Keshavarzi, Ravi Prasher, Chandrakant Patel, Sushil H. Bhavnani, Rama Venkatasubramanian, Ravi Mahajan, Yogendra Joshi, Bahgat Sammakia, Bruce A. Myers, Len Chorosinski, Martine Baelmans, Prabhu Sathyamurthy, and Peter E. Raad. Thermal challenges in next-generation electronic systems. *IEEE Transactions on Components and Packaging Technologies*, 31(4):801–815, 2008. doi: 10.1109/TCAPT.2008.2001197.

Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009. doi: 10.1002/nme.2579.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. GNOT: A General Neural Operator Transformer for Operator Learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 12556–12569. PMLR, 2023. URL https://proceedings.mlr.press/v202/hao23c.html.

Jianing Huang, Kaixuan Zhang, Youjia Wu, and Ze Cheng. Operator learning with domain decomposition for geometry generalization in PDE solving. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=IxAnL4PRsg.

Ameya D. Jagtap and George Em Karniadakis. Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020. doi: 10.4208/cicp.OA-2020-0164.

Ameya D. Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020. doi: 10.1016/j.cma.2020.113028.

George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998. doi: 10.1137/S1064827595287997.

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

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

Bjoern List, Li-Wei Chen, Kartik Bali, and Nils Thuerey. Differentiability in unrolled training of neural physics simulators on transient dynamics. *Computer Methods in Applied Mechanics and Engineering*, 433:117441, 2025. doi: 10.1016/j.cma.2024.117441. URL https://www.sciencedirect.com/science/article/pii/S0045782524006960.

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

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41432–41449. PMLR, 2025. URL https://proceedings.mlr.press/v267/luo25o.html.

Chenkai Mao, Robert Lupoiu, Tianxiang Dai, Mingkun Chen, and Jonathan Fan. Towards general neural surrogate solvers with specialized neural accelerators. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 34693–34711. PMLR, 2024. URL https://proceedings.mlr.press/v235/mao24b.html.

Tarek Mathew. *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*, volume 61 of *Lecture Notes in Computational Science and Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-77205-7 978-3-540-77209-5. doi: 10.1007/978-3-540-77209-5.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8459–8468. PMLR, 2020. URL https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html.

The OpenFOAM Foundation. OpenFOAM 7 released, 2019. URL https://openfoam.org/release/7/.

Yoshihide Tominaga, Akashi Mochida, Ryuichiro Yoshie, Hiroto Kataoka, Tsuyoshi Nozu, Masaru Yoshikawa, and Taichi Shirasawa. AIJ guidelines for practical applications of CFD to pedestrian wind environment around buildings. *Journal of Wind Engineering and Industrial Aerodynamics*, 96(10-11):1749–1761, 2008. doi: 10.1016/j.jweia.2008.02.058.

Haixin Wang, Jiaxin Li, Anubhav Dwivedi, Kentaro Hara, and Tailin Wu. BENO: Boundary-embedded neural operators for elliptic PDEs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ZZTkLDRmkg.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53681–53705. PMLR, 2024. URL https://proceedings.mlr.press/v235/wu24r.html.

Rui Wu, Nikola Kovachki, and Burigede Liu. A learning-based domain decomposition method. *Computer Methods in Applied Mechanics and Engineering*, 453:118799, May 2026. ISSN 00457825. doi: 10.1016/j.cma.2026.118799.

Ryuichiro Yoshie, Akashi Mochida, Yoshihide Tominaga, Hiroto Kataoka, Kazuyoshi Harimoto, Tsuyoshi Nozu, and Taichi Shirasawa. Cooperative project for CFD prediction of pedestrian wind environment in the Architectural Institute of Japan. *Journal of Wind Engineering and Industrial Aerodynamics*, 95(9-11):1551–1578, 2007. doi: 10.1016/j.jweia.2007.02.023.

Hang Zhou, Haixu Wu, Haonan Shangguan, Yuezhou Ma, Huikun Weng, Jianmin Wang, and Mingsheng Long. Transolver-3: Scaling up transformer solvers to industrial-scale geometries. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.04940.

APPENDIX

This appendix provides supplementary material for ONCE. It includes related work, proofs of the theoretical results, the training and inference algorithm, followed by extended experimental results and visualizations, implementation details, and benchmark descriptions.

APPENDIX CONTENTS

A Related Work **13**

B Limitations and Future Work **14**

C Proofs of the Theoretical Results **15**
C.1 Proof of Theorem 1 15
C.2 Proof of Proposition 1 15
C.3 Proof of Proposition 2 16

D Algorithm of ONCE **17**

E Extended Results **17**
E.1 Time Costs 17
E.2 Comparison with Noise-injected Pre-training and Tuning 18
E.3 Trajectory Sampling and Consistency Regularization 18
E.4 Effect of Data Augmentation 18
E.5 Effect of Local Operator 19
E.6 Effect of DDM Configurations 20
E.7 Additional Visualizations 21

F Experimental Setups **22**
F.1 Setup of DDMs 22
F.2 TAT Protocol 23
F.3 Model Configurations 23
F.4 Metrics 24

G Benchmarks **24**
G.1 2D Elliptic Equation 24
G.2 3D Elliptic Equation: HeatSink-3D 25
G.3 General Simulations: UrbanRANS-2D 26

A RELATED WORK

Neural Operators. Neural operators learn mappings between function spaces to approximate PDE solution operators. Pioneering architectures include DeepONet (Lu et al., 2021) and the Fourier neural operator (FNO) (Li et al., 2021). To handle irregular geometries in practical engineering, subsequent architectures extended neural operators to unstructured meshes (Li et al., 2023) and complex or inhomogeneous boundaries (Wang et al., 2024). More recently, Transolver (Wu et al., 2024) introduced Physics-Attention to represent discretized domains using learned physical-state tokens, enabling geometry-flexible PDE modeling with computational complexity linear in the number of mesh points and strong performance across standard PDE benchmarks and industrial aerodynamic simulations. Transolver++ (Luo et al., 2025) further scales neural PDE solving to million-point meshes. Nevertheless, these methods primarily learn global solution operators over the training distribution from the paired input-output, making extrapolation to substantially larger domains and more complex geometries challenging.

DDMs with Learned Solvers. DDMs are classical numerical methods to solve PDE problems that are too large or too complex in geometry for a single solver, by partitioning them into smaller subdomain problems that can be solved in parallel and reassembled (Mathew, 2008). They have

initially been explored to combine with physics-informed neural networks (PINNs) to effectively handle large-scale problems (Jagtap et al., 2020; Jagtap & Karniadakis, 2020). This allows different subdomain networks to capture different local physics and reduces the complexity required of each network (Jagtap & Karniadakis, 2020). However, these approaches still require instance-specific optimization of the subdomain networks, severely impeding the computational efficiency of iterative domain decomposition. Recent studies have instead explored pre-trained neural operators as reusable local solvers, avoiding repeated optimization during inference (Mao et al., 2024; Huang et al., 2026; Wu et al., 2026). Specifically, SNAP-DDM incorporates specialized FNOs on regular patches into DDM iterations (Mao et al., 2024), enabling large-scale electromagnetic simulations. Schwarz Neural Inference (SNI) handles variable geometries by training a local operator on solutions over simple canonical shapes (Huang et al., 2026), and combines its predictions through additive Schwarz iterations, enabling generalization to unseen geometries on unstructured meshes. Despite demonstrating geometric flexibility on simple benchmarks, its data generation procedure assumes that meaningful local boundary-value problems can be independently generated and solved under synthetically sampled boundary conditions, making it difficult to extend to complex practical PDE simulations, such as simulating urban wind flow.

Distribution Shift in Learned Iterative Methods. A fundamental challenge in sequential prediction tasks is the mismatch between training and inference (Bengio et al., 2015). While models are typically trained using ground-truth states under one-step supervision, inference requires them to condition autoregressively on their own previous outputs. Therefore, even small errors can shift the visited state distribution away from the training data and degrade long-horizon performance (Ross et al., 2011). Prior studies have mitigated this mismatch by perturbing training states with noise (Sanchez-Gonzalez et al., 2020), training on model-generated inputs through pushforward or unrolled objectives (Brandstetter et al., 2022b; List et al., 2025), or collecting and relabeling states visited by the learned model (Ross et al., 2011). Neural DDMs exhibit the same self-conditioned framework structure: each local prediction determines the interface states used as model input in subsequent iterations. Existing neural DDMs, however, train local solvers exclusively on reference interfaces, leaving them vulnerable to out-of-distribution errors and compounding drift during iterative inference. ONCE addresses this training-inference mismatch by exposing the local solver to states along the actual convergence trajectory.

B Limitations and Future Work

Computational Cost. ONCE recovers the global solution through repeated local solver evaluations and interface exchanges rather than a single forward pass. Although subdomain predictions are computed in parallel within each iteration, the sequential Schwarz iterations and associated DDM overhead lead to substantially higher inference latency than global neural operators, as reported in Appendix E.1. This cumulative runtime poses a practical bottleneck for larger domains or real-time simulation and rapid design-action loops. Future work will focus on accelerating iterative convergence and reducing the overhead of interface transmission and domain assembly.

Scope of Evaluation. While our experiments cover different domain scales and geometries, the evaluated problems still remain limited in physical complexity and application scale. Extending and systematically evaluating the framework in real-world applications is an important direction for future work, for example, 3D city-scale wind simulations.

Self-Evolving via Test-Time Adaptation. During inference, ONCE operates with frozen model parameters, relying purely on feedforward Schwarz iterations. A promising future direction that we have not yet explored is to enable the local operator to update its weights on-the-fly, allowing the neural solver to progressively accumulate physical priors and self-improve across diverse tasks without retraining from scratch.

C Proofs of the Theoretical Results

C.1 Proof of Theorem 1

Since D is a closed subset of the Banach space Λ and $\widehat{\mathcal{T}}_\theta$ is a strict contraction on D, the Banach fixed-point theorem directly guarantees the existence, uniqueness, and convergence to the fixed point $\widehat{\lambda}^*_\theta \in D$.

Next, by the triangle inequality and the contractivity of $\widehat{\mathcal{T}}_\theta$, the one-step interface error satisfies:

$$\begin{aligned} e_{k+1} &= \left\| \widehat{\mathcal{T}}_\theta\big(\widehat{\lambda}^{(k)}_\theta\big) - \lambda^* \right\|_\Lambda \\ &\le \left\| \widehat{\mathcal{T}}_\theta\big(\widehat{\lambda}^{(k)}_\theta\big) - \widehat{\mathcal{T}}_\theta(\lambda^*) \right\|_\Lambda + \left\| \widehat{\mathcal{T}}_\theta(\lambda^*) - \lambda^* \right\|_\Lambda \\ &\le \rho_\theta e_k + \varepsilon_{\mathrm{ref}}(\theta). \end{aligned} \tag{18}$$

Iterating Eq. (18) from $k = 0$ to $K - 1$ gives:

$$\begin{aligned} e_K &\le \rho_\theta^K e_0 + \varepsilon_{\mathrm{ref}}(\theta) \sum_{m=0}^{K-1} \rho_\theta^m \\ &= \rho_\theta^K e_0 + \frac{1-\rho_\theta^K}{1-\rho_\theta}\, \varepsilon_{\mathrm{ref}}(\theta), \end{aligned} \tag{19}$$

which proves the first inequality in Eq. (11). Then letting $K \to \infty$ in Eq. (19), we get the fixed-point bias:

$$\left\| \widehat{\lambda}^*_\theta - \lambda^* \right\|_\Lambda \le \frac{\varepsilon_{\mathrm{ref}}(\theta)}{1-\rho_\theta}, \tag{20}$$

which proves the second inequality in Eq. (11). If $\varepsilon_{\mathrm{ref}}(\theta) = 0$, the iteration converges to the reference interface state λ^* with zero bias. □

C.2 Proof of Proposition 1

By the definition of $\widehat{\mathcal{T}}_\theta$ and Assumption 1, we have:

$$\begin{aligned} \left\| \widehat{\mathcal{T}}_\theta(\lambda) - \lambda^* \right\|_\Lambda^2 &= \left\| \mathcal{C}\left(\big(\widehat{\mathcal{S}}_\theta(\lambda_i, a_i)\big)_{i=1}^N \right) - \mathcal{C}\left((u_i^*)_{i=1}^N \right) \right\|_\Lambda^2 \\ &\le C_{\mathrm{tr}}^2 \sum_{i=1}^N \left\| u_i^* - \widehat{\mathcal{S}}_\theta(\lambda_i, a_i) \right\|_i^2 . \end{aligned} \tag{21}$$

Setting $\lambda = \lambda^*$ in Eq. (21), we obtain

$$\begin{aligned} \varepsilon_{\mathrm{ref}}(\theta)^2 &\le C_{\mathrm{tr}}^2 \sum_{i=1}^N \left\| \widehat{\mathcal{S}}_\theta(\lambda_i^*, a_i) - u_i^* \right\|_i^2 \\ &= C_{\mathrm{tr}}^2 \mathcal{L}_{\mathrm{ref}}(\theta). \end{aligned} \tag{22}$$

Taking square roots in Eq. (22) gives the reference consistency bound in Proposition 1. Under the assumptions of Theorem 1, the second inequality in Eq. (11) further yields:

$$\left\| \widehat{\lambda}^*_\theta - \lambda^* \right\|_\Lambda \le \frac{\varepsilon_{\mathrm{ref}}(\theta)}{1-\rho_\theta} \le \frac{C_{\mathrm{tr}} \sqrt{\mathcal{L}_{\mathrm{ref}}(\theta)}}{1-\rho_\theta}, \tag{23}$$

which proves Eq. (14).

Next, applying Eq. (13) at the collected states and summing with weights $p_k \ge 0$ gives

$$\begin{aligned} &\sum_{k=0}^{K-1} p_k \left\| \widehat{\mathcal{T}}_\theta\big(\widehat{\lambda}^{(k)}_{\theta_0}\big) - \lambda^* \right\|_\Lambda^2 \\ &\quad \le C_{\mathrm{tr}}^2 \sum_{k=0}^{K-1} p_k \sum_{i=1}^N \left\| \widehat{\mathcal{S}}_\theta\big(\widehat{\lambda}^{(k)}_{\theta_0,i}, a_i\big) - u_i^* \right\|_i^2 \\ &\quad = C_{\mathrm{tr}}^2 \mathcal{L}_{\mathrm{traj}}(\theta; \theta_0). \end{aligned} \tag{24}$$

Adding $\beta \geq 0$ times Eq. (22) to Eq. (24) yields

$$\begin{aligned}\sum_{k=0}^{K-1} p_k \left\| \widehat{\mathcal{T}}_\theta\big(\widehat{\lambda}_{\theta_0}^{(k)}\big) - \lambda^* \right\|_\Lambda^2 + \beta\, \varepsilon_{\mathrm{ref}}(\theta)^2 \\ \leq C_{\mathrm{tr}}^2 \left[\mathcal{L}_{\mathrm{traj}}(\theta;\theta_0) + \beta \mathcal{L}_{\mathrm{ref}}(\theta)\right] \\ = C_{\mathrm{tr}}^2 \mathcal{L}_{\mathrm{TAT}}(\theta;\theta_0),\end{aligned} \tag{25}$$

which is Eq. (15). □

C.3 Proof of Proposition 2

For each k, we decompose the interface error as follows:

$$\begin{aligned}\widehat{\lambda}_\theta^{(k+1)} - \lambda^* = \widehat{\mathcal{T}}_\theta\big(\widehat{\lambda}_\theta^{(k)}\big) - \widehat{\mathcal{T}}_\theta\big(\widehat{\lambda}_{\theta_0}^{(k)}\big) \\ + \widehat{\mathcal{T}}_\theta\big(\widehat{\lambda}_{\theta_0}^{(k)}\big) - \lambda^*.\end{aligned} \tag{26}$$

Applying the weighted Minkowski inequality to Eq. (26) and using the definition of B_θ in Eq. (16) gives

$$\begin{aligned}\left(\sum_{k=0}^{K-1} p_k e_{k+1}(\theta)^2\right)^{1/2} &\leq B_\theta + \left(\sum_{k=0}^{K-1} p_k \left\| \widehat{\mathcal{T}}_\theta\big(\widehat{\lambda}_{\theta_0}^{(k)}\big) - \lambda^* \right\|_\Lambda^2\right)^{1/2} \\ &\leq B_\theta + C_{\mathrm{tr}} \sqrt{\mathcal{L}_{\mathrm{traj}}(\theta;\theta_0)},\end{aligned} \tag{27}$$

where the last inequality follows from Eq. (24). This proves Eq. (17). □

D ALGORITHM OF ONCE

Algorithm 1 summarizes the two-stage training procedure and iterative inference of ONCE. We distinguish the trajectory collection length K_{col} from the iteration limit K_{eval}. Inference stops when the absolute interface update falls below tol or the iteration limit is reached, whichever occurs first (Appendix F.4).

Algorithm 1 Training and inference of ONCE

Input: Global training cases; new input a; collection length K_{col}; iteration limit K_{eval}; tolerance tol; sampling settings; consistency weight β.
Output: Global solution approximation $\widehat{u}$.

Training Stage 1: Pre-training
Extract $(a_i, \lambda_i^*, u_i^*)$ from global reference solutions.
Train the local operator with $\mathcal{L}_{\text{ref}}$ to obtain θ_0.
Training Stage 2: Trajectory-aligned Tuning
Freeze θ_0.
for each selected training case **do**
  Initialize $\widehat{\lambda}_{\theta_0}^{(0)}$.
  for $k = 0, \ldots, K_{\text{col}} - 1$ **do**
    If step k is selected, retain $\big(a_i, \widehat{\lambda}_{\theta_0,i}^{(k)}, u_i^*\big)$ for all subdomains.
    $\widehat{\lambda}_{\theta_0}^{(k+1)} \leftarrow \widehat{\mathcal{T}}_{\theta_0}\big(\widehat{\lambda}_{\theta_0}^{(k)}\big)$.
  end for
end for
Fine-tune from θ_0 with $\mathcal{L}_{\text{traj}} + \beta\mathcal{L}_{\text{ref}}$ to obtain θ^*
Inference Stage:
Decompose the domain of a, initialize $\widehat{\lambda}^{(0)}$.
for $k = 0, \ldots, K_{\text{eval}} - 1$ **do**
  $\widehat{u}_i^{(k+1)} \leftarrow \widehat{\mathcal{S}}_{\theta^*}\big(\widehat{\lambda}_i^{(k)}, a_i\big)$, for all subdomains in parallel.
  $\widehat{\lambda}^{(k+1)} \leftarrow \mathcal{C}\big((\widehat{u}_i^{(k+1)})_{i=1}^N\big)$.
  if $\big\|\widehat{\lambda}^{(k+1)} - \widehat{\lambda}^{(k)}\big\|_2 < \text{tol}$ **then**
    break ▷ convergence criterion satisfied
  end if
end for
Combine the latest local predictions to obtain $\widehat{u}$.

E EXTENDED RESULTS

E.1 TIME COSTS

Unlike global neural operators (NOs) that require only a single forward pass, ONCE relies on K Schwarz iterations. Within each iteration, all N independent subdomains (each of bounded size s) are evaluated in parallel via a batched forward pass. Assuming the local operator scales linearly with input size, each iteration costs $\mathcal{O}(Ns) = \mathcal{O}(n)$ for a global domain of n cells, yielding an overall inference complexity of $\mathcal{O}(Kn)$. On a single A100 GPU, one iteration takes 10.6, 22.3, and 75.7 ms for $N = 4, 16$, and 61 subdomains, compared to 16.4, 24.0, and 75.4 ms for the global NO. In practice, the in-scale domain satisfies the convergence tolerance within 26 iterations on average, whereas the medium and large domains requires more iterations because interface updates stagnate at an error floor that grows with N. Here we set a conservative upper limit of

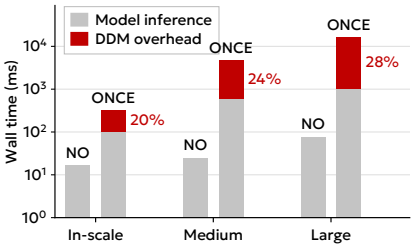


Figure 7: Comparison of wall-clock time on *Poisson*. (DDM overhead includes time for partitioning, interface extraction, transfers, and merging.)

$K_{\text{eval}} = 200$ iterations to ensure sufficient convergence across all cases. As shown in Fig. 7, the resulting wall-clock time is $19\times$, $193\times$, and $210\times$ that of the global NO, with DDM overhead accounting for 20%, 24%, and 28% of the runtime.

E.2 Comparison with Noise-injected Pre-training and Tuning

To mitigate error accumulation during long rollouts, Sanchez-Gonzalez et al. (2020) inject random-walk noise into input velocities during training, aiming to improve the simulator's robustness to its own imperfect predictions at inference. Motivated by this, we compare TAT against noise-injected training. Specifically, we perturb the reference Robin interface $\widetilde{\lambda}_i^* = u_i^* + \gamma^{-1}\partial u_i^*/\partial n_i$ with noise $\sigma s \xi_i$ while keeping the target solution u_i^* exact. Here s is the standard deviation of the reference solutions, and $\xi_i \sim \mathcal{N}(0, I)$ is white Gaussian noise drawn independently at each interface node. We evaluate $\sigma \in \{0.01, 0.03, 0.1, 0.3\}$ both during pre-training from scratch and during fine-tuning from θ_0.

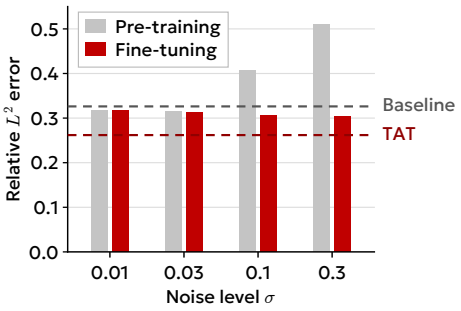


Figure 8: Large-domain relative L^2 errors with noise-injected pre-training and tuning. Dashed lines denote the pre-trained baseline θ_0 and TAT.

Fig. 8 shows that noise-injected pre-training reduces the large-domain error by at most 3.1% relative to θ_0, but stronger perturbations increase it by up to 56.1%. This is expected, as excessive noise corrupts the clean boundary-solution correspondence, and thus compromises the model's reference-state consistency. While applying noise during fine-tuning avoids this degradation and consistently lowers the error by 2.7–7.2%, its gain remains limited compared to TAT, which achieves a 19.7% error reduction relative to θ_0 and outperforms the best noise setting by 13.5%.

E.3 Trajectory Sampling and Consistency Regularization

Table 3 examines the effects of the consistency regularization weight β, n_s, and the number of sample cases used for trajectory collection in *Poisson*. Increasing β from 0.25 to 1 improves accuracy on both scales, whereas $\beta = 5$ offers only a marginal improvement on large-domain. Using 15 states per case performs comparably to 20, suggesting limited benefit from denser sampling. Increasing sampled cases from 50 to 200 improves both scales, but 400 cases improve medium-domain accuracy while worsening large-domain accuracy. These results validate the default configuration as in Appendix F, demonstrating that a moderate trajectory budget with balanced consistency regularization effectively mitigates inference drift without imposing heavy tuning overhead.

Table 3: Relative L^2 errors for TAT design on *Poisson*. Shaded rows indicate the default settings.

(a) Consistency weight β.

β	Medium	Large
0.25	0.0382	0.3156
0.5	0.0283	0.2783
1	**0.0238**	0.2619
5	0.0252	**0.2604**

(b) States per case n_s.

n_s	Medium	Large
5	0.0246	0.2743
10	0.0251	0.2792
15	**0.0238**	**0.2619**
20	0.0240	0.2621

(c) Sampled cases.

No.	Medium	Large
50	0.0258	0.2798
100	0.0246	0.2645
200	0.0238	**0.2619**
400	**0.0221**	0.2642

E.4 Effect of Data Augmentation

To ensure the fair comparison with global neural operator baselines, we did not use data augmentation for neural DDMs. However, as stated in Huang et al. (2026), data augmentation can significantly improve the performance of neural DDMs. Following Brandstetter et al. (2022a), we apply random transformations from the dihedral group D_4 to each training subdomain. This group encompasses the 8 discrete symmetries of a square domain (rotations by multiples of 90° and axis reflections), which inherently preserves the underlying Cartesian grid without interpolation artifacts. As summarized in Fig. 9, incorporating D_4 augmentation substantially improves solution accuracy, reducing

the relative L^2 error by 17% and 16% on the in-scale and large domains of *Poisson*, and by 61% and 40% on *Darcy*, respectively.

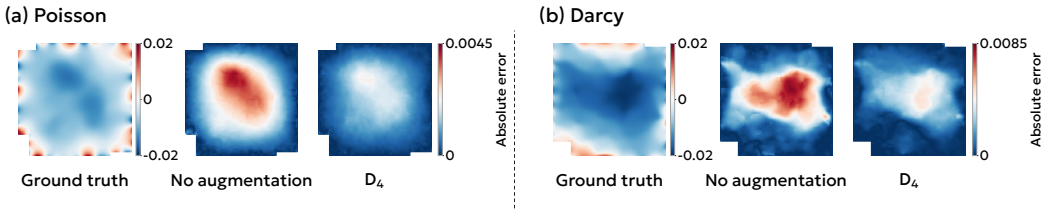


Figure 9: D_4 augmentation reduces the large-domain error of ONCE on (a) *Poisson* and (b) *Darcy*.

However, we found that the same augmentation technique brings no benefit to *UrbanRANS-2D* due to its directional inflow along the $+x$ axis and the convection-dominated nature of the problem. Because the dataset already varies building orientations relative to this inflow, subdomain rotation merely introduces out-of-distribution inflow angles never encountered at inference. This underscore that the design of data augmentation for neural DDMs must strictly adhere to the underlying physical symmetries of each application; we leave such problem-specific designs to future work.

E.5 Effect of Local Operator

Architecture. We first replace Transolver with Geo-FNO and GNOT as the local operator of ONCE, while keeping the domain decomposition, Robin transmission, and pre-training and TAT protocols fixed. As shown in Table 4, Transolver achieves the lowest in-scale and medium-domain errors with fewer parameters, while GNOT yields a slightly lower large-domain error. These results illustrate the flexibility of ONCE in accommodating different local operator architectures.

Table 4: Relative L^2 errors on *Poisson* with different local operators. Best errors are in **bold**; the shaded row denotes the default architecture.

Local operator	Params (M)	In-scale	Medium	Large
Geo-FNO (Li et al., 2023)	3.61	0.0154	0.0453	0.3113
GNOT (Hao et al., 2023)	2.35	0.0123	0.0265	**0.2570**
Transolver (Wu et al., 2024)	1.50	**0.0090**	**0.0238**	0.2619

Model Size. We further assess the effect of Transolver size by varying its depth over $\{2, 6, 8, 12\}$ layers. As shown in Fig. 10, errors decrease with depth on the in-scale and medium domains, while the large-domain error increases from 2 to 6 layers before decreasing at greater depths. Overall, the 12-layer model achieves the lowest errors across all three domain scales, demonstrating that increasing model capacity benefits both in-scale accuracy and OOD generalization.

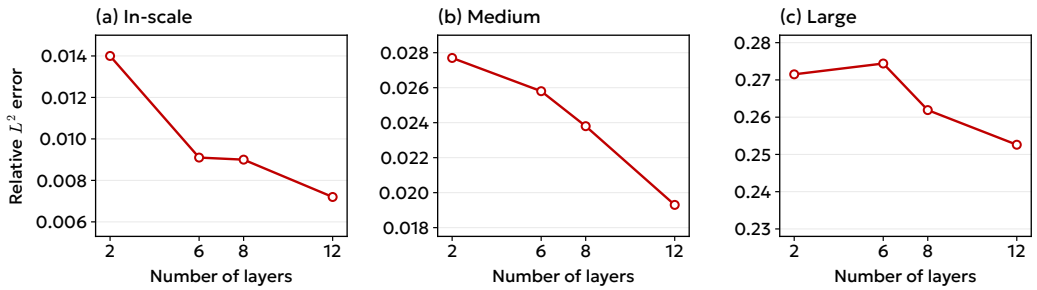


Figure 10: Effect of model size on *Poisson*. Relative L^2 errors are shown across three domain scales.

E.6 Effect of DDM Configurations

In optimized Schwarz methods, the convergence behavior is fundamentally governed by the choice of transmission parameters, domain overlap, and subdomain assembly protocols (Gander, 2006; Cai & Sarkis, 1999). We therefore systematically evaluate how these DDM configurations influence the solution accuracy and convergence rate of ONCE. Specifically, we investigate the sensitivity to the overlap width and the Robin parameter γ, and further compare our default restricted additive Schwarz (RAS) assembly against the classical additive Schwarz (AS) scheme adopted in SNI (Huang et al., 2026). The results are presented in Fig. 11 and Table 5.

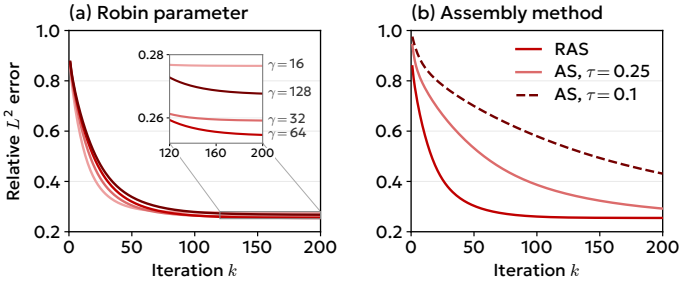


Figure 11: Convergence on *Poisson* large domain: mean relative L^2 errors for (a) Robin parameters and (b) assembly methods. (In (b), all three variants eventually settle to a common plateau error of approximately 0.26.)

Overlap Width. As shown in Table 5(a), increasing the overlap width accelerates convergence and reduces the settling time on large domains by nearly three times. However, a wider overlap slightly increases the error from 0.2136 to 0.2838 for large domain may due to the interfaces placed deeper inside neighboring domains are more prone to accumulating prediction errors. An overlap of two or three layers provides balance between convergence speed and asymptotic accuracy.

Robin Parameters. The parameter γ balances Dirichlet solution values against Neumann normal fluxes in the transmission condition. As illustrated in Fig. 11(a), a smaller γ of 16 emphasizes normal gradients to accelerate early communication across interfaces, achieving the fastest settling time of 5.20 s. Conversely, moderate values such as 32 and 64 rely more on stable function values to achieve lower asymptotic errors. Increasing γ further to 128 shifts the condition closer to standard Dirichlet transmission, which would slow down convergence.

Assembly Methods. Fig. 11(b) shows that RAS outperforms classical AS in convergence rate while reaching a comparable final error. AS averages predictions across overlapping regions, which mixes local errors and requires damping to maintain stability. In contrast, RAS updates the global field strictly on non-overlapping partition cores. This strategy prevents error contamination in shared regions, substantially accelerating iterative convergence in this case.

Table 5: Prediction accuracy and convergence cost for DDM configurations on *Poisson*. Shaded rows indicate the default settings. The best and second-best values in each column are **bold** and underlined, respectively.

	Relative L^2 Error		Iterations to settle		TTS (s)	
Setting	Medium	Large	Medium	Large	Medium	Large
(a) Overlap width.						
1	0.0326	**0.2136**	104.0	180.5	2.36	15.39
2	**0.0177**	0.2312	63.5	103.0	1.57	9.72
3	0.0238	0.2619	44.5	68.0	1.21	7.02
4	0.0242	0.2838	**34.0**	**50.5**	**0.99**	**5.71**
(b) Robin parameter γ.						
16	0.0240	0.2765	**34.0**	**48.5**	**0.96**	**5.20**
32	0.0238	0.2619	44.5	68.0	1.21	7.02
64	**0.0190**	**0.2546**	52.0	82.5	1.39	8.38
128	0.0204	0.2677	57.0	88.0	1.50	8.89

Estimated time to settle (TTS) measures the total wall-clock time (including partitioning setup and iterations on an A100 GPU) for the prediction error to stabilize within $\pm 5\%$ of its final converged value.

E.7 Additional Visualizations

Unseen Heat-Source Configurations in HeatSink-3D. In practical electronic packaging and multi-chip modules (MCMs), heat generation is rarely uniform or singly connected (Garimella et al., 2008). While the training heat sources only have rectangular footprints, Fig. 12 evaluates ONCE and Transolver on a non-rectangular footprint and two spatially separated heat sources. ONCE yields lower relative L^2 errors in both cases, illustrating its ability to handle more varied heat-source configurations.

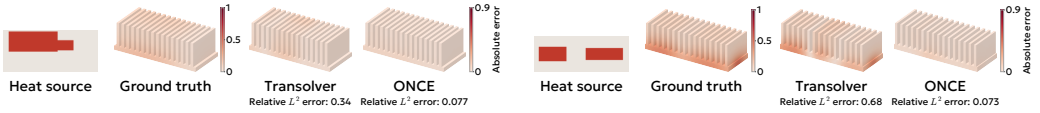


Figure 12: Example results from *HeatSink-3D* for non-rectangular (left) and spatially separated (right) heat sources. Each figure shows the placement of heat source, ground-truth temperature, and absolute-error fields for Transolver and ONCE.

Convergence. Fig. 13 visualizes the inference process of ONCE across representative cases from each benchmark. The iterations illustrate how inter-subdomain communication progressively reconstructs the global solution field. The accompanying curves track both the absolute interface update and the relative L^2 error against the ground truth, decoupling iterative stabilization from prediction accuracy. Across all benchmarks, the interface updates generally decrease over the rollout, although transient fluctuations are observed for UrbanRANS-2D, while solution errors approach plateaus.

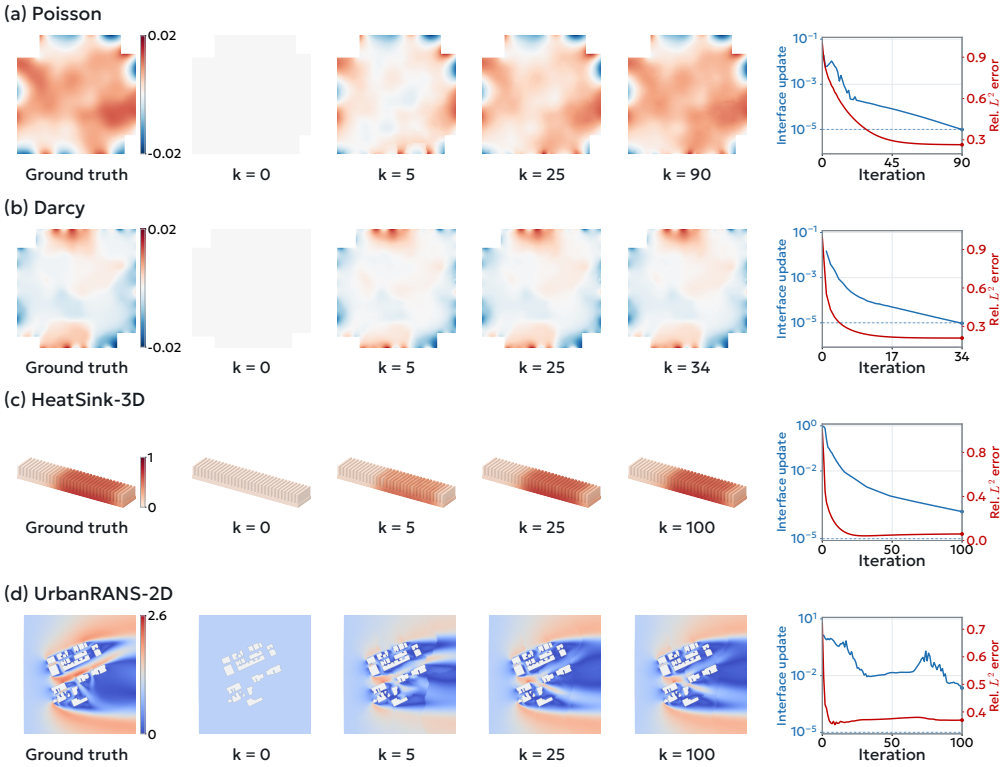


Figure 13: ONCE convergence for one case per benchmark, using selected irregular Large test cases for *Poisson* and *Darcy*. Each row shows ground truth, four selected iterates including initialization, and the absolute interface update and relative L^2 error.

F Experimental Setups

Table 6 summarizes the DDM, pre-training (PT), and TAT configurations of ONCE across the four benchmarks. The following subsections describe the domain decomposition setup, TAT protocol, model configurations, and evaluation metrics in detail.

F.1 Setup of DDMs

Domain Partitions. Following the implementation of Huang et al. (2026), we construct overlapping subdomains by first partitioning a geometry-aware adjacency graph into non-overlapping cores using METIS (Karypis & Kumar, 1998) and then expanding each core outward. For *Poisson* and *Darcy*, graph nodes represent active grid cells connected through their four face neighbors. For *HeatSink-3D*, solid voxels are connected through their six face neighbors. For *UrbanRANS-2D*, which involves complex fluid domains around buildings, graph nodes are fluid-cell centers connected via Delaunay triangulation. To respect solid wall boundaries during overlap creation, subdomains are expanded along graph-reachable paths within a prescribed physical cutoff distance. This procedure naturally yields subdomains with flexible and irregular shapes.

Note that Table 6 provides the partition parameters only at the inference and tuning stage. In contrast, during the pre-training stage, we extract subdomain samples by applying randomized partitioning to the training domains—varying both the number of METIS parts and the overlap layers across multiple draws (detailed in Appendix G). Because inference across diverse domain sizes and geometries inherently produces irregular local patches, this pre-training randomization exposes the local operator to a rich distribution of subdomain geometries, scales, and interface configurations, enhancing its generalization.

Table 6: ONCE configurations across the four benchmarks. PT denotes pre-training.

	Hyperparameter	Poisson	Darcy	HeatSink-3D	UrbanRANS-2D
DDMs	Partition size s	256	256	600	1,650
	Overlap width	3 layers	3 layers	2 layers	32 m
	Initialization	Zero	Zero	Zero	Inlet velocity
	Robin parameter γ	32	32	1	8
	Assembly method	RAS	RAS	RAS	RAS
	Iteration limit K_{eval}	200	200	200	500
	Stopping tolerance $\Delta^{(k)}$	10^{-5}	10^{-5}	10^{-5}	5×10^{-4}
PT	Training epochs	500	500	500	200
	Batch size	64	64	64	48
	Training loss	Relative L_2	Relative L_2	Relative L_2	RMSE
	Learning rate	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
	Optimizer	AdamW	AdamW	AdamW	AdamW
TAT	Training epochs	40	40	40	60
	Sampled cases	200	200	100	400
	Collect iterations K_{col}	50	50	100	200
	States per case n_s	15	15	15	10
	Sampling strategy	Uniform	Uniform	Uniform	Uniform
	Consistency weight β	1	1	1	1
	Batch size	768	768	64	96
	Learning rate	4×10^{-4}	4×10^{-4}	1×10^{-3}	2×10^{-4}

Configurations. For a fair comparison, SNI and ONCE use the same partition size and overlap width on every benchmark, as listed in Table 6. For *Poisson*, *Darcy*, and *HeatSink-3D*, the partition size s is the target number of cells per core used at inference and during TAT trajectory collection: a domain with n cells is divided by METIS into $\text{round}(n/s)$ parts, and each connected component of a part forms one core, so that every core contains approximately s cells or fewer before the overlap is added. For *UrbanRANS-2D*, the partition size is the median number of graph nodes per subdomain after the overlap is added. Likewise, overlap width is measured by graph-adjacency layers for the first three benchmarks and by physical distance in meters for *UrbanRANS-2D*. We set the Robin parameter to $\gamma = 32$ for *Poisson* and *Darcy*, $\gamma = 1$ for *HeatSink-3D*, and $\gamma = 8$ for *UrbanRANS-2D*. During inference, the Schwarz iteration is initialized with a zero field for *Poisson*, *Darcy*, and *HeatSink-3D*, and with a uniform field matching the inlet velocity $\mathbf{v} = (5, 0)\text{ms}^{-1}$ for *UrbanRANS-2D*. Iterations stop according to the pre-defined stopping criteria detailed in Appendix F.4.

F.2 TAT Protocol

Starting from the pre-trained parameters θ_0, we run the learned Schwarz iteration on selected training cases using the same domain partitioning and initialization scheme as in inference. From the recorded iteration steps $k = 0, \ldots, K_{\text{col}} - 1$, we uniformly sample n_s distinct states without replacement, where each state provides N local subdomain inputs paired with their reference solutions u_i^*. During fine-tuning, the operator minimizes the composite objective $\mathcal{L}_{\text{traj}} + \beta\mathcal{L}_{\text{ref}}$. Parameter updates are computed using separate batches for trajectory and reference samples, with each loss term averaged over its respective batch. All trajectory samples carry equal weight and are processed once per epoch, with benchmark-specific settings summarized in Table 6.

F.3 Model Configurations

Neural Operators. Our implementations of Geo-FNO, GNOT, and Transolver are based on Neural-Solver-Library[1], while BENO is based on its origin implementation[2]. To ensure a fair com-

[1] https://github.com/thuml/Neural-Solver-Library

[2] https://github.com/AI4Science-WestlakeU/beno

parison, baseline hyperparameters were tuned on the validation set within the following candidate search spaces:

- **Geo-FNO** (Li et al., 2023): 4 Fourier layers with hidden width $\{32, 64\}$ and Fourier modes $\{32, 64\}$.
- **GNOT** (Hao et al., 2023): $\{1, 2, 3\}$ experts, $\{4, 8\}$ layers, 8 attention heads, and hidden width $\{96, 128\}$.
- **BENO** (Wang et al., 2024): 5 message-passing blocks, 1 Transformer layer with 2 heads, and 128 boundary embedding channels following the original implementation.
- **Transolver** (Wu et al., 2024): 8 layers, hidden width $\{128, 256\}$, 8 attention heads, and 32 physical state slices across all benchmarks.

Neural DDMs. For ONCE, we employ Transolver as the local operator, which is first pre-trained on reference subdomain solutions and then fine-tuned via the TAT protocol (Appendix F.2). For the adapted SNI baseline (Huang et al., 2026), we implement GNOT coupled with additive Schwarz methods for global coupling, trained on the exact same subdomain partitions as ONCE. In both cases, the local operators share the exact same model architectures and hyperparameters as the global setup specified above.

F.4 Metrics

Evaluation Metrics. For all the benchmarks, we evaluate prediction accuracy of each method using the relative L^2 error, which measures the difference between the predicted and reference solutions. Specifically, for a reference solution u^* with model prediction $\widehat{u}$, the relative L^2 error is defined as

$$\text{Relative } L^2 \text{ Error} = \frac{\|\widehat{u} - u^*\|_2}{\|u^*\|_2}, \tag{28}$$

where the norms are computed over the discretization points inside the physical domain. We report the mean error across all the test cases.

Stopping Criterion. During iterative inference, the convergence of ONCE is monitored by the absolute interface update between consecutive iterations:

$$\Delta^{(k)} = \left\|\widehat{\lambda}^{(k)} - \widehat{\lambda}^{(k-1)}\right\|_2, \qquad k \geq 1, \tag{29}$$

where $\widehat{\lambda}^{(k)}$ denotes the concatenated interface state across all artificial interfaces Γ^{art}, as defined in Section 2.2. Iteration stops when $\Delta^{(k)} < \text{tol}$ or when reaching the iteration limit K_{eval}, whichever occurs first. The benchmark-specific tolerances tol and iteration limits K_{eval} are detailed in Table 6.

G Benchmarks

To comprehensively evaluate the accuracy, spatial scalability, and geometric generalization of ONCE, we conduct experiments spanning both scientific benchmarks and engineering applications, as illustrated in Fig. 14. In the following subsections, we provide the full specifications for each benchmark, including governing equations, boundary conditions, data generation, and the dataset setups for both in-scale and out-of-distribution (OOD) evaluations.

G.1 2D Elliptic Equation

Problem setup. We consider Poisson and Darcy equations with inhomogeneous Dirichlet boundary conditions as in Wang et al. (2024). The governing equations are:

$$\begin{aligned} \nabla \cdot \big(\kappa(\mathbf{x}) \nabla u(\mathbf{x})\big) &= f(\mathbf{x}) \quad \text{in } \Omega, \\ u(\mathbf{x}) &= g(\mathbf{x}) \quad \text{on } \partial\Omega. \end{aligned} \tag{30}$$

Here u is the solution field, f is the forcing, and g specifies the boundary values. The Poisson equation uses $\kappa \equiv 1$, while the Darcy equation uses a spatially varying positive coefficient. The task is to predict u from the geometry, forcing, boundary values, and coefficient field.

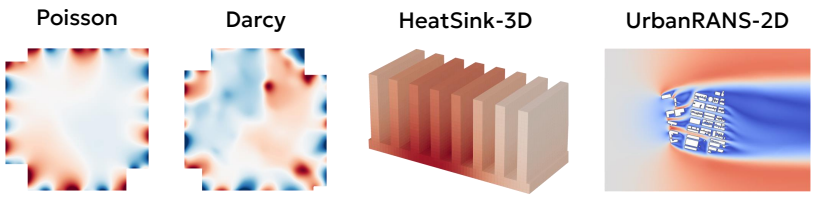


Figure 14: Examples from the *Poisson*, *Darcy*, *HeatSink-3D*, and *UrbanRANS-2D* benchmarks.

Data Generation. BENO consider complex and inhomogenous BVPs solving. In our study, we adopt same geometry setup and generate additional in-scale and out-of-distribution solutions to comprehensively evaluate the performance of ONCE. The domain is a unit square with zero to four rectangular corner cuts, forming the 0c–4c geometry families. Each case is represented on a 32×32 cell-centered grid with an active-domain mask and 128 boundary points. For *Poisson*, we adopt the original BENO solutions. For *Darcy*, we reuse BENO's geometries, forcing, and boundary values and generate new solutions using piecewise-linear finite elements. Positive coefficients are obtained by exponentiating a smooth random trigonometric field whose log-standard deviation is normalized to 0.7.

Fig. 15 illustrates the geometry variations and spatial scales used in these benchmarks. Spatial scaling is evaluated separately on synthetic domains of side lengths 1, 2, and 4, discretized into 32×32, 64×64, and 128×128 cells, respectively, at a fixed spacing of $h = 1/32$. We train on the first 900 cases of 4c, assessing both same-family interpolation (4c) and zero-shot transfer to unseen boundary geometries (0c–3c). Each case is partitioned twice, the number of METIS parts is drawn uniformly from $\{3, 4, 5, 6\}$, and the overlap from $\{2, 3, 4\}$ layers. In total, the evaluation comprises 400 in-scale, 100 medium, and 100 large test cases.

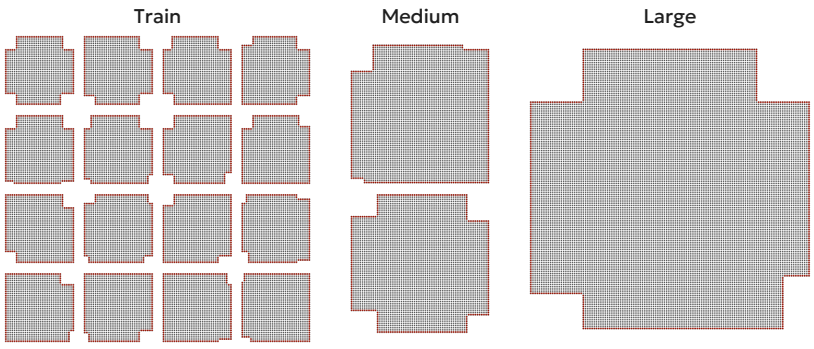


Figure 15: Representative geometries for *Poisson* and *Darcy* benchmarks. In-scale examples (left), medium-scale examples (middle), and large-scale example (right) are shown at a common spatial scale. Medium and large domains have twice and four times the training side length, respectively. Gray and red points denote interior and boundary points.

G.2 3D Elliptic Equation: HeatSink-3D

Problem setup. *HeatSink-3D* models steady heat conduction in three-dimensional heat sinks with varying fin geometries. Let Ω denote the solid, Γ_{chip} the heated footprint on the base underside, and Γ_{conv} the remaining exposed surface. The temperature satisfies:

$$\begin{aligned} -\nabla \cdot (\kappa \nabla T) &= 0 && \text{in } \Omega, \\ T &= T_{\text{chip}} && \text{on } \Gamma_{\text{chip}}, \\ -\kappa\, \partial_n T &= h_{\text{f}}(T - T_\infty) && \text{on } \Gamma_{\text{conv}}. \end{aligned} \tag{31}$$

Here h_{f} is the heat-transfer coefficient and n is the outward normal. In dimensionless units, we set cell width and conductivity to 1, $T_{\mathrm{chip}} = 1$, and $T_{\infty} = 0$, and sample h_{f} log-uniformly from $[0.003, 0.03]$. Heat exchange with the environment is represented by the convection boundary condition. The task is to predict the temperature field from the solid geometry, chip footprint, and film coefficient.

Data Generation. Each part consists of a rectangular base with plate fins or square pin fins on a Cartesian voxel grid. Training geometries vary fin thickness over 2–3 cells, base thickness over 3–5 cells, gaps over 2–4 cells, and fin height over 14–25 cells; chip size and position also vary. Reference solutions are generated by a cell-centered finite-volume method with a sparse direct solver.

Fig. 16 shows an example case and representative training and evaluation geometries. The dataset contains 944 cases in total. The 700 in-scale cases comprise plate heat sinks with 4–6 fins and inline pin arrays of 3×3 or 4×4, with 564 cases used for training, 36 for validation, and 100 for in-scale evaluation. The remaining 244 cases serve as the OOD test set, including larger fin counts, staggered and cross-cut arrangements, nonuniform spacing and heights, and two sampled chip footprints. Cell counts range from 1,080–24,720 for in-scale cases to 5,400–126,464 for OOD cases. Each case is partitioned once, with the number of METIS parts set to $\max\{2, \mathrm{round}(n/600)\}$, where n is the number of voxels, and with the overlap drawn uniformly from $\{2, 3\}$ layers.

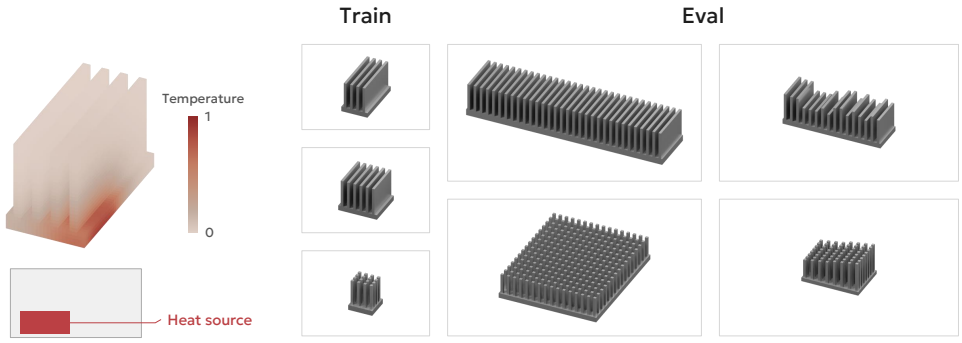


Figure 16: *HeatSink-3D* problem setup and representative geometries. Left: a reference temperature field on the fixed dimensionless scale $0 \le T \le 1$, with the prescribed-temperature heat-source patch shown on the base underside below. Right: training plate-fin and pin-fin geometries, alongside evaluation cases with larger fin counts or nonuniform fin heights.

G.3 General Simulations: UrbanRANS-2D

Problem setup. *UrbanRANS-2D* simulates two-dimensional steady-state turbulent wind flow through urban layouts using the incompressible Reynolds-averaged Navier–Stokes (RANS) equations with the standard k–ϵ turbulence closure. The fluid domain is a rectangular region with building footprints removed. The governing equations are

$$\begin{aligned} \nabla \cdot \mathbf{v} &= 0, \\ (\mathbf{v} \cdot \nabla)\mathbf{v} &= -\nabla p + \nabla \cdot \left[(\nu + \nu_t)\big(\nabla \mathbf{v} + (\nabla \mathbf{v})^{\top}\big) \right], \end{aligned} \tag{32}$$

where $\mathbf{v} = (v_x, v_y)$ is the mean velocity, p is the solver's kinematic pressure with the isotropic turbulent stress absorbed, and $\nu = 10^{-5}\,\mathrm{m^2\,s^{-1}}$. The turbulence closure determines the eddy viscosity ν_t. The inlet prescribes $\mathbf{v} = (5, 0)\,\mathrm{m\,s^{-1}}$, $k = 0.375\,\mathrm{m^2\,s^{-2}}$, and $\epsilon = 0.054\,\mathrm{m^2\,s^{-3}}$. We impose zero pressure at the outlet, symmetry on the lateral boundaries, and no-slip conditions with wall functions on buildings. The task is to predict the velocity field from the urban geometry at given boundary conditions.

Data Generation. Building layouts are originally sampled from the Kowloon City footprint collection and rotated relative to the inflow direction. The surrounding domain includes upstream, downstream, and lateral margins, with a cross-stream blockage ratio of at most 40%. We additionally includes the test layout of the Architectural Institute of Japan (AIJ) benchmark (Case E) as

an external benchmark (Tominaga et al., 2008; Yoshie et al., 2007), which includes more densely packed buildings. All simulations are conducted by OpenFOAM-v7 (The OpenFOAM Foundation, 2019) `simpleFoam` on quadrilateral-dominated meshes generated by `gmsh` (Geuzaine & Remacle, 2009), with target cell sizes of 1.5 m near walls and 10 m in the far field. The simulations were considered converged when the residuals of all variables fell below 10^{-5}. We train on 8,983 in-scale layouts with 67,423 subdomains extracted for pre-training. Evaluation is conducted on three test sets of 200 layouts each across in-scale, medium, and large layouts. Each training layout is partitioned once, with the number of METIS parts cycled through $\{4, 6, 8, 12\}$ across layouts and a fixed overlap of 32 m.

Fig. 17 illustrates the computational domain configurations and representative urban layouts across different scales. To evaluate spatial scalability and geometric generalization, layouts are categorized into three tiers based on the sampled building cluster window: in-scale layouts with window widths of 70-110 m, medium-scale layouts of 200 m, and large-scale layouts spanning 250-500 m. For each layout, the full computational domain extends beyond the building cluster by including upstream, downstream, and lateral far-field buffer margins.

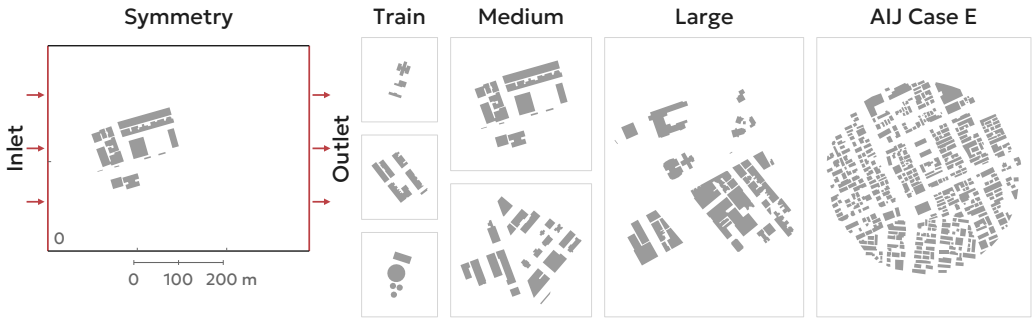


Figure 17: Computational domain and representative urban geometries. Left: computational domain and boundary conditions. Right: training, medium- and large-scale evaluation layouts, and the external AIJ Case E layout.

Unlike previous benchmarks for scalar field prediction, *UrbanRANS-2D* involves coupled velocity and pressure fields subject to heterogeneous boundary conditions. In our experiments, we found that jointly learning the velocity-pressure coupling and explicitly enforcing the different inlet, outlet, and symmetry conditions would introduce additional complexity into the local solver and the coupling system. Therefore, to simplify the learning formulation, we primarily focus on predicting the steady-state velocity field, and use semantic encoding for imposing boundary conditions. Specifically, boundary points are assigned scalar identifiers $\{0, 1, 2, 3\}$ corresponding to artificial interfaces, inlets, outlets, and lateral symmetry boundaries, respectively, enabling the local solver to distinguish physical conditions from interfaces. The model inputs thus comprise cell coordinates, wall distances, prescribed boundary values, and their semantic type labels, while the steady-state velocity components $\mathbf{v} = (v_x, v_y)$ serve as targets. This benchmark assesses whether a local solver trained exclusively on small urban blocks can successfully generalize and assemble complex building-wake interactions across larger-scale layouts.