MESHOP: OPERATOR LEARNING FOR ADAPTIVE MESH REFINEMENT IN PARAMETRIC PDES

Anonymous authors
Paper under double-blind review

ABSTRACT

Adaptive finite-element methods (AFEM) construct problem-dependent meshes through repeated solve–estimate–mark–refine cycles, but this sequential mesh-discovery process must be repeated across PDE instances with different input functions or physical parameters. We propose MeshOp, an operator-learning framework for adaptive mesh refinement across parametric PDE families. MeshOp predicts AFEM-derived refinement fields directly from PDE inputs, thereby reducing the cost of sequential mesh discovery. The predicted field is converted into a conforming hierarchical mesh by a dedicated score-to-mesh realization procedure, while residual-based AFEM can continue from this learned initialization when higher accuracy is required. Across the three benchmarks, the mean exact-level disagreement between quantized predictions and AFEM targets ranges from 11.4% to 27.4%, yet the realized meshes attain finite-element errors comparable to those of fixed-cycle AFEM. This empirical downstream tolerance suggests that effective discretization does not require exact pointwise recovery of the AFEM refinement field. MeshOp reduces online time by $2.34\times$ and $2.10\times$ relative to reference-error-matched AFEM on Burgers and disk convection–diffusion, respectively, and by $3.06\times$ relative to comparable-accuracy fixed-cycle AFEM on three-dimensional reaction–diffusion.

1 INTRODUCTION

High-fidelity simulation of partial differential equations (PDEs) is often limited not only by the solver, but also by the cost of constructing an effective discretization. Uniform meshes allocate resolution independently of the solution structure and may therefore waste degrees of freedom in smooth regions while underresolving localized layers, fronts, or shocks.

Adaptive finite-element methods (AFEM) address this mismatch through repeated solve–estimate–mark–refine cycles (Dörfler, 1996; Morin et al., 2002). Classical analyses establish convergence and optimal or quasi-optimal convergence rates for adaptive finite-element methods (Binev et al., 2004; Cascon et al., 2008). In this process, a suitable problem-dependent mesh is discovered sequentially: each refinement decision depends on the discrete solution and error indicators computed on the current mesh. Consequently, PDE instances with different initial conditions, forcing terms, boundary data, or physical parameters require repeated solution, estimation, marking, and refinement before the desired discretization is reached. This repeated mesh-discovery process can become a substantial part of the computational cost when solving families of related PDEs.

We propose MeshOp, a mesh-operator learning framework that learns from offline AFEM trajectories to predict adaptive refinement for new PDE instances, reducing the need for repeated solve–estimate–mark–refine cycles. Specifically, MeshOp learns an operator that maps PDE inputs directly to an AFEM-derived refinement-demand field. A dedicated score-to-mesh procedure then converts the predicted field into a conforming hierarchical mesh, on which a conventional finite-element solver computes the solution. When higher accuracy is required, standard residual-based AFEM can continue directly from the learned mesh.

Unlike solution prediction, refinement prediction is evaluated by the induced mesh and downstream finite-element error rather than exact pointwise recovery of the AFEM target. Our results reveal an empirical downstream tolerance: imperfect score fields can still induce effective meshes, consistent

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.

with the aggregation, quantization, and conformity closure in mesh realization. Appendix B.1 provides a comparison with direct solution prediction, while Appendix B shows that this utility persists under a Burgers distribution shift without retraining.

The main contributions are threefold:

- We formulate fixed-budget AFEM refinement as a function-to-field operator mapping functional PDE inputs to AFEM-derived refinement-generation fields on a common query grid.
- We develop a dedicated score-to-mesh procedure that converts predicted refinement fields into conforming hierarchical meshes through recursive refinement and conformity closure, enabling conventional FEM solution and continued residual-based AFEM refinement when needed.
- Across three complementary PDE families, we demonstrate an empirical downstream tolerance: despite non-negligible pointwise refinement-score errors, MeshOp produces meshes with finite-element errors comparable to those of fixed-cycle AFEM, while achieving $2.10\times$–$3.06\times$ online speedups over error-matched or comparable-accuracy AFEM baselines.

2 Related Work

Neural operators. Neural operators learn mappings between function spaces and provide a natural framework for PDE families with varying functional inputs (Kovachki et al., 2023). Representative architectures include DeepONet and its POD-based extension, the Fourier neural operator (FNO), the convolutional neural operator (CNO), and graph- and transformer-based neural operators (Lu et al., 2021; 2022; Li et al., 2021; Raonic et al., 2023; Li et al., 2020; 2023b). In this work, neural operators are used to learn mappings from functional PDE inputs to AFEM-derived refinement fields rather than directly to PDE solutions.

Learned numerical components. Rather than predicting PDE solutions directly, another line of work integrates learning into conventional numerical solvers by learning problem-dependent components, including multigrid prolongation operators, preconditioners, and problem-dependent subspaces (Greenfeld et al., 2019; Li et al., 2023a; 2025). In this context, MeshOp retains the conventional finite-element solver while learning a problem-dependent adaptive mesh.

Learning-based mesh adaptation. Learning-based mesh adaptation spans several directions, including reinforcement-learning methods that jointly learn surrogate dynamics and policies for mesh refinement and coarsening (Wu et al., 2023), swarm-based refinement strategies (Freymuth et al., 2023), supervised methods that derive element-size fields from structured geometry and boundary-value-problem descriptors through learned pointwise convergence models (Legeland et al., 2025), iterative adaptive mesh-generation methods that reconstruct expert meshing strategies by repeatedly predicting sizing fields on evolving mesh graphs (Freymuth et al., 2025), approaches that infer adaptive sizing information from coarse or approximate numerical solutions (Zhang et al., 2025), and r-adaptive methods that relocate existing nodes (Hu et al., 2024). A closely related supervised approach is that of Chan et al. (2022), who formulate fixed-resolution mesh-density prediction as an operator-learning problem and generate meshes through Gmsh-based remeshing. More recently, MeshONet (Xiao et al., 2026) formulates structured mesh generation as an operator-learning problem, enabling generalization across varying geometries without retraining. In contrast, MeshOp learns the refinement-generation field produced by a fixed-budget residual-based AFEM procedure and introduces a threshold-controlled NVB realization procedure to recover conforming hierarchical meshes from the predicted scores.

3 Problem Formulation

We consider families of PDEs on a fixed domain Ω with a fixed initial mesh $\mathcal{T}_0$, parameterized by inputs $\xi \in \mathcal{X}$, which may include functional inputs such as initial conditions, forcing terms, and boundary data, as well as finite-dimensional physical parameters, and seek efficient adaptive finite-element discretizations for each instance. We denote the corresponding solution by $u(\xi)$.

3.1 ADAPTIVE FINITE-ELEMENT REFINEMENT

Given an initial mesh $\mathcal{T}_0$, AFEM constructs a sequence of meshes by repeatedly solving the discrete problem, computing local residual-based indicators, marking important elements, and refining them. Let $\mathcal{T}_k$ denote the mesh at refinement step k. A typical AFEM cycle is

$$u_k = \mathcal{S}(\xi, \mathcal{T}_k), \qquad \{\eta_K\}_{K\in\mathcal{T}_k} = \mathcal{E}(u_k, \mathcal{T}_k), \tag{1}$$

where $\mathcal{S}$ is the finite-element solver and η_K is the local residual-based indicator associated with element $K \in \mathcal{T}_k$. The marked elements are locally refined to produce

$$\mathcal{T}_{k+1} = \mathrm{Adapt}(\mathcal{T}_k, \{\eta_K\}_{K\in\mathcal{T}_k}), \tag{2}$$

where Adapt comprises marking, local refinement, and the additional refinements required to maintain mesh conformity.

Although AFEM provides reliable error control under suitable assumptions, its sequential structure requires multiple PDE solves for every new instance ξ. The online cost therefore includes not only the final finite-element solve, but also the repeated discovery of where additional resolution is needed.

3.2 LEARNING ADAPTIVE REFINEMENT AS AN OPERATOR PROBLEM

Instead of reproducing the entire adaptive iteration, we aim to predict the refinement field accumulated by an offline AFEM procedure. Let $\mathcal{T}_\xi^{\mathrm{AFEM}}$ denote the final mesh produced by the prescribed AFEM procedure for the PDE instance ξ. For each final leaf element $K \in \mathcal{T}_\xi^{\mathrm{AFEM}}$, we define its target refinement generation as

$$g_K^\star = \min\{\mathrm{gen}(K), G\}, \tag{3}$$

where $\mathrm{gen}(K) \in \mathbb{N}_0$ denotes the raw bisection generation of K: elements in the initial mesh have generation 0, and each successive bisection increases the generation by one. The cap G defines the maximum target refinement level used for operator learning.

The elementwise generations define the piecewise-constant target field

$$s^\star(\xi; \boldsymbol{x}) = g_K^\star, \qquad \boldsymbol{x} \in K, \quad K \in \mathcal{T}_\xi^{\mathrm{AFEM}}. \tag{4}$$

For a fixed deterministic AFEM configuration, each input ξ uniquely determines the target field $s^\star(\xi, \cdot)$. Since $0 \le s^\star(\xi; \boldsymbol{x}) \le G$ and Ω is bounded,

$$s^\star(\xi, \cdot) \in L^\infty(\Omega) \subset L^2(\Omega), \qquad \|s^\star(\xi, \cdot)\|_{L^2(\Omega)} \le G|\Omega|^{1/2}. \tag{5}$$

Thus, the AFEM target defines an $L^2(\Omega)$-valued refinement operator

$$\mathcal{A}_G : \mathcal{X} \to L^2(\Omega), \qquad \mathcal{A}_G(\xi) = s^\star(\xi, \cdot). \tag{6}$$

We sample this field on a common query grid and use the sampled values as supervision. We approximate the refinement operator using a neural operator

$$\mathcal{G}_\theta : \mathcal{X} \to \mathcal{Y}, \qquad \widehat{s}_\theta(\xi; \boldsymbol{x}) = \big[\mathcal{G}_\theta(\xi)\big](\boldsymbol{x}), \tag{7}$$

where $\mathcal{Y}$ denotes the fixed-grid representation of the refinement field and $\widehat{s}_\theta(\xi, \cdot)$ denotes the predicted refinement-score field for the PDE input ξ. The refinement score is an intermediate representation, and distinct pointwise score fields may induce similar meshes after aggregation, quantization, and conformity closure.

4 MESH OPERATOR LEARNING FRAMEWORK

Our framework consists of offline target generation and training, followed by online mesh construction and finite-element solution. Figure 1 provides an overview of the complete offline and online workflow. The complete pipeline is

$$\widehat{s}_\theta = \mathcal{G}_\theta(\xi), \qquad \widehat{\mathcal{T}} = \mathcal{R}(\mathcal{T}_0, \widehat{s}_\theta), \qquad u_{\widehat{\mathcal{T}}} = \mathcal{S}(\xi, \widehat{\mathcal{T}}), \tag{8}$$

where $\mathcal{R}$ converts the predicted score field into a conforming hierarchical mesh.

4.1 TARGET GENERATION AND OPERATOR TRAINING

For each training instance ξ_i, we run AFEM from the common initial mesh $\mathcal{T}_0$ and construct the target field $s^\star(\xi_i;\cdot)$ using equation 3–equation 4. Evaluating this field on a fixed query grid $\{\boldsymbol{x}_j\}_{j=1}^{N_q}$ produces fixed-size training labels for adaptive meshes with different element arrangements.

We parameterize $\mathcal{G}_\theta$ using a neural-operator architecture and train it to predict the target refinement field. For Burgers and three-dimensional reaction–diffusion, we use the standard grid-point MSE. For disk convection–diffusion, we use a physical-area-weighted MSE to account for the nonuniform physical cell areas represented by the polar query grid:

$$\mathcal{L}_{\mathrm{CD}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \frac{\sum_{k,\ell} a_{k\ell}\left(\widehat{s}_{i,k\ell} - s^\star_{i,k\ell}\right)^2}{\sum_{k,\ell} a_{k\ell}}, \qquad a_{k\ell} = \frac{1}{2}\left(r_{k+1/2}^2 - r_{k-1/2}^2\right)\Delta\theta_\ell. \tag{9}$$

Here $a_{k\ell}$ denotes the physical area of the polar cell associated with (r_k, θ_ℓ), accounting for the radial Jacobian and nonuniform radial spacing.

4.2 HIERARCHICAL SCORE-TO-MESH REALIZATION

For a new PDE instance ξ, the operator first predicts $\widehat{s}_\theta(\xi,\cdot)$ on the common query grid. We clip the predicted grid values to obtain

$$\widehat{s}^{\,c}_\theta(\xi, \boldsymbol{x}_j) = \mathrm{clip}(\widehat{s}_\theta(\xi, \boldsymbol{x}_j), 0, G), \qquad j = 1, \ldots, N_q. \tag{10}$$

The clipped values are linearly interpolated and used throughout mesh realization.

Let $\widehat{\mathcal{T}}^{(0)} = \mathcal{T}_0$. At realization step r, for each leaf element $K \in \widehat{\mathcal{T}}^{(r)}$, let $\mathcal{Q}(K)$ denote a small set of query points in K. Its aggregated score and predicted generation are

$$\overline{s}_K = \max_{\boldsymbol{x}\in\mathcal{Q}(K)} \widehat{s}^{\,c}_\theta(\xi, \boldsymbol{x}), \qquad \widehat{g}_K = \lfloor \overline{s}_K + 1 - \tau \rfloor, \tag{11}$$

where $\tau = 0.5$ in the principal experiments unless otherwise stated.

The elements requiring refinement at step r are

$$\mathcal{M}^{(r)} = \left\{ K \in \widehat{\mathcal{T}}^{(r)} : \mathrm{gen}(K) < \widehat{g}_K \right\}. \tag{12}$$

The marked elements are refined by newest-vertex bisection (NVB) (Maubach, 1995; Stevenson, 2008), together with any additional bisections required for conformity, producing $\widehat{\mathcal{T}}^{(r+1)}$. The same interpolated score field is then evaluated on the newly created leaf elements. The procedure terminates when $\mathcal{M}^{(r)} = \emptyset$, and the resulting conforming hierarchical mesh is denoted by $\widehat{\mathcal{T}}$. A detailed description of the online mesh-realization procedure is provided in Appendix C.6.

4.3 FINITE-ELEMENT SOLUTION AND AFEM CORRECTION

The operator-generated mesh is passed to the conventional finite-element solver:

$$u_{\widehat{\mathcal{T}}} = \mathcal{S}(\xi, \widehat{\mathcal{T}}). \tag{13}$$

When additional accuracy is needed, standard solve–estimate–mark–refine cycles are continued from $\widehat{\mathcal{T}}$. The learned operator therefore provides an adaptive initial mesh, while subsequent AFEM cycles can correct under-refined regions and further improve the solution accuracy.

5 NUMERICAL EXPERIMENTS

5.1 EXPERIMENTAL PROTOCOL

We evaluate MeshOp on three PDE families with complementary adaptive structures: a space–time Burgers equation, a two-dimensional convection–diffusion problem on the unit disk, and a three-dimensional reaction–diffusion problem. Unless otherwise stated, FNO is used as the representative

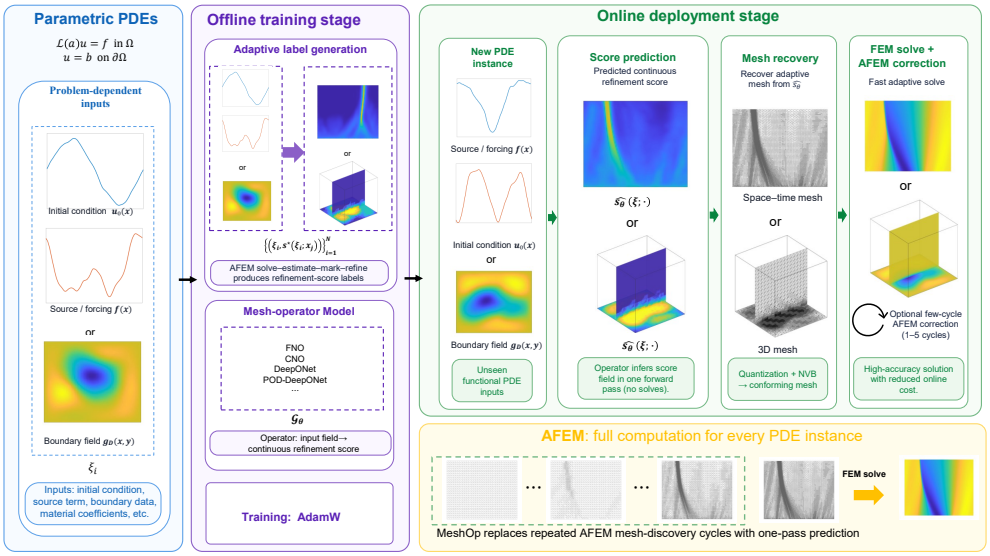


Figure 1: Overview of the proposed mesh-operator learning framework.

refinement operator, trained on 3000 instances and evaluated on 100 independent instances. Predicted scores are clipped to $[0, G]$, converted to integer generations using equation 11 with $\tau = 0.5$, and realized by conforming NVB. The target field is processed identically as an oracle diagnostic reference. Online time includes operator inference, mesh construction, finite-element assembly and solution, and any explicitly reported AFEM correction; offline training, target generation, reference construction, error evaluation, and plotting are excluded. FNO inference runs on an NVIDIA RTX A6000 GPU, while mesh construction and finite-element computations run on an Intel Xeon Gold 6240 CPU. Comparisons with alternative neural-operator backbones and mesh-generation baselines, a Burgers OOD test, and additional implementation details are reported in the appendix.

5.2 Benchmark Problems

Space–time Burgers. We first consider a forced viscous Burgers equation on the space–time domain $(-1, 1) \times (0, 1]$,

$$\begin{cases} u_t + u\,u_x - \nu u_{xx} = f(x), & (x,t) \in (-1,1) \times (0,1], \\ u(-1,t) = u(1,t), \quad u_x(-1,t) = u_x(1,t), & t \in (0,1], \\ u(x,0) = u_0(x), & x \in [-1,1]. \end{cases} \tag{14}$$

with $\nu = 5 \times 10^{-3}$ and $u_0(x) = -a\sin(\pi(x+b)) + g(x)$, where $a \sim \mathcal{U}(0.5, 1.5)$ and $b \sim \mathcal{U}(-0.85, 0.85)$ control the amplitude and spatial shift. The independent mean-zero, period-two Gaussian fields satisfy $g \sim \mathcal{N}(0, 625(-\Delta + 25I)^{-2.5})$ and $f \sim \mathcal{N}(0, 800(-\Delta + 25I)^{-3})$, with their real Fourier expansions truncated to wavenumbers $k = 1, \ldots, 64$. These random inputs generate moving localized steep-gradient structures whose refinement trajectories vary across instances, providing a space–time test of the learned refinement operator.

Disk convection–diffusion. We next consider steady convection–diffusion on the unit disk $\Omega = \{(x, y) : x^2 + y^2 < 1\}$,

$$\begin{cases} -\varepsilon \Delta u + \boldsymbol{\beta} \cdot \nabla u = f, & \boldsymbol{x} \in \Omega, \\ u = 0, & \boldsymbol{x} \in \partial\Omega. \end{cases} \tag{15}$$

with $\varepsilon = 10^{-3}$ and $\boldsymbol{\beta} = (\cos\theta, \sin\theta)$, where $\theta \sim \mathcal{U}(0, 2\pi)$. The forcing is

$$f(x,y) = \exp\left[-\frac{(q - q_c)^2}{2\sigma_q^2}\right] g_{\mathrm{per}}(x,y), \qquad q = -x\sin\theta + y\cos\theta, \tag{16}$$

Table 1: Benchmark and AFEM target-generation configurations.

Problem	Initial mesh	Target generation	Query grid
Burgers	32×32 Cartesian partition, P1 triangles	12 fixed AFEM cycles, Dörfler $\theta = 0.8$, periodic NVB	101×101
Disk CD	Delaunay disk mesh, 1016 P1 triangles	12 fixed AFEM cycles, Dörfler $\theta = 0.8$, conforming NVB	64×128 polar
3D RD	16^3 cubes, six P1 tetrahedra per cube	10 fixed AFEM cycles, Dörfler $\theta = 0.8$, conforming NVB	65^3, $z_i = (i/64)^2$

where $q_c \sim \mathcal{U}(-0.5, 0.5)$ and $\sigma_q \sim \mathcal{U}(0.05, 0.15)$ control the transverse location and width, while $g_{\mathrm{per}} \sim \mathcal{N}(0, 625(-\Delta + 25I)^{-2})$ is a mean-zero, period-two Fourier GRF on $[-1, 1]^2$, truncated to nonzero modes $|k_1|, |k_2| \leq 16$ (one per opposite-mode pair) and restricted to the disk.

Three-dimensional reaction–diffusion. Finally, we consider the three-dimensional steady reaction–diffusion problem

$$\begin{cases} -\varepsilon^2 \Delta u + u = 0, & \boldsymbol{x} \in \Omega = (0,1)^3, \\ u(x, y, 0) = g_D(x, y), & (x, y) \in (0,1)^2, \\ u = 0, & \text{on } \partial\Omega \setminus \{z = 0\}. \end{cases} \tag{17}$$

with $\varepsilon = 2 \times 10^{-2}$. The nonhomogeneous Dirichlet data on the bottom face are defined by

$$g_D(x, y) = 30 \cdot 16x(1 - x)y(1 - y)\, g_{\mathrm{per}}(x, y), \tag{18}$$

where

$$g_{\mathrm{per}} \sim \mathcal{N}\big(0, 625(-\Delta + 25I)^{-4}\big) \tag{19}$$

is a mean-zero periodic Gaussian random field truncated to the nonzero Fourier modes satisfying $|k_1|, |k_2| \leq 16$ (one per opposite-mode pair). The polynomial envelope ensures compatibility with the homogeneous conditions along the edges of the bottom face. The small diffusion scale produces a thin layer near $z = 0$, while the random boundary field determines its horizontal refinement structure, testing whether the same refinement-operator formulation extends to three-dimensional tetrahedral meshes. For all three problems, the covariance notation follows the unnormalized real-Fourier convention used in data generation.

AFEM targets. Burgers and disk convection–diffusion use equation 3 with $G = 12$, while three-dimensional reaction–diffusion uses $G = 10$. For Burgers and reaction–diffusion, the uniform initial meshes allow the raw refinement generation to be recovered from the element measure as $\mathrm{round}(\log_2(A_0/|K|))$ and $\mathrm{round}(\log_2(V_0/|K|))$, respectively. For the disk problem, whose initial triangulation contains elements of different areas, the refinement generation is tracked directly during NVB refinement. The benchmark-specific AFEM target-generation protocols and main discretization settings are summarized in Table 1.

5.3 Refinement-field prediction

Table 2 summarizes FNO refinement-field prediction across the three PDE families. We report MAE, RMSE, exact-level accuracy after quantization with $\tau = 0.5$, within-one-score accuracy defined by $|\widehat{s}_{ij} - s^\star_{ij}| \leq 1$, and the mean relative score error. For Disk CD, the generic query-point index is $j = (k, \ell)$ and $\omega_j = a_{k\ell} / \sum_{p,q} a_{pq}$, while the other problems use $\omega_j = 1/N_q$. The relative score error is

$$E_{\mathrm{score}} = \frac{1}{N_{\mathrm{test}}} \sum_{i=1}^{N_{\mathrm{test}}} \left(\frac{\sum_j \omega_j (\widehat{s}_{ij} - s^\star_{ij})^2}{\sum_j \omega_j (s^\star_{ij})^2} \right)^{1/2}. \tag{20}$$

5.4 Finite-element performance

Performance at the target-generation level. Table 3 compares the finite-element performance of the FNO-generated mesh, the target-field mesh (an oracle reference obtained from the AFEM

Table 2: FNO refinement-field prediction over 100 independent test instances.

Problem	MAE	RMSE	Exact-level acc	Within 1	Rel. score L^2
Burgers	0.336	0.518	0.726	0.927	0.217
Disk CD	0.327	0.616	0.746	0.886	0.350
3D reaction–diffusion	0.157	0.320	0.886	0.979	0.088

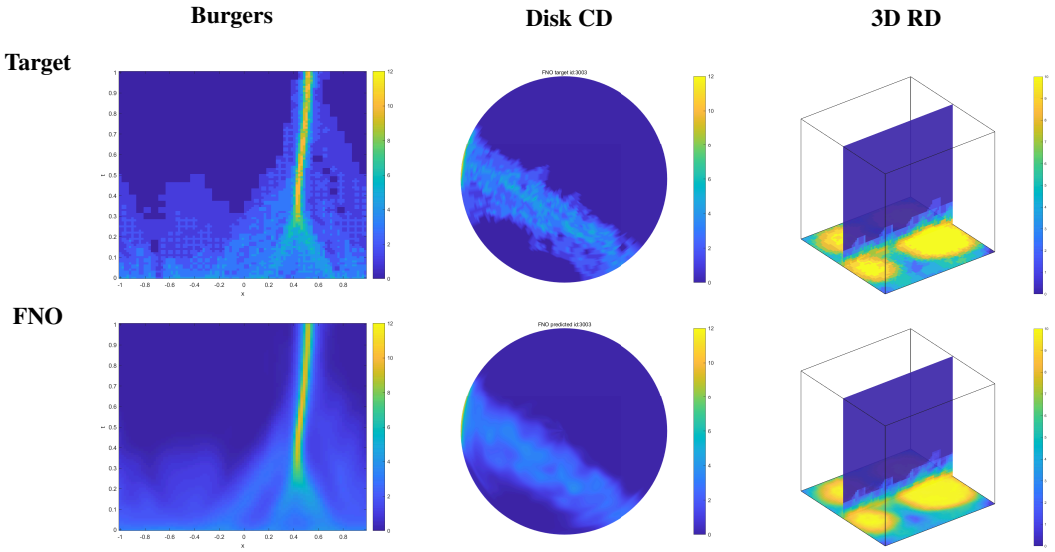


Figure 2: Representative AFEM-derived target refinement fields and FNO predictions for Burgers (test 4), disk convection–diffusion (test 3), and three-dimensional reaction–diffusion (test 12).

target field), standard AFEM baselines, and fixed-mesh FEM. For reference-error-matched AFEM, we select post hoc, for each test instance, the closer of the two consecutive standard-AFEM iterates whose reference-solution errors bracket the corresponding FNO error, and average the resulting quantities over the selected iterates. This matching is used only for retrospective cost–accuracy comparison. For three-dimensional reaction–diffusion, the reference-error-matched and fixed-cycle AFEM baselines are identical for all 100 test instances, using 10 cycles in the base comparison and 13 cycles in the corrected comparison.

Although the mean relative refinement-score error reaches 0.350 and the mean exact-level accuracy falls to 72.6%, the FNO-generated meshes still achieve finite-element accuracy competitive with fixed-cycle AFEM, revealing a substantial empirical decoupling between pointwise score accuracy and downstream solution accuracy. At comparable accuracy, FNO is $2.34\times$ and $2.10\times$ faster than reference-error-matched AFEM for Burgers and disk convection–diffusion, respectively; relative to fixed-cycle AFEM, the speedups are $2.93\times$, $2.52\times$, and $3.06\times$ across the three problems, with slightly lower 3D reaction–diffusion error but about 50% more DOFs.

AFEM correction. The learned mesh can also be used as an adaptive initialization rather than a terminal discretization. Table 4 reports higher-accuracy results obtained by continuing residual-based AFEM from the learned and target-field meshes, together with the corresponding standard-AFEM baselines.

With AFEM correction, MeshOp retains a clear advantage. For Burgers, two correction cycles achieve an error of 5.172×10^{-4} with a $1.46\times$ speedup over reference-error-matched AFEM. For disk convection–diffusion, three cycles attain comparable accuracy (1.700×10^{-2} versus 1.696×10^{-2}) with a $1.54\times$ speedup. For reaction–diffusion, three correction cycles yield a mean relative L^2 error of 8.582×10^{-3}, compared with 9.284×10^{-3} after 13 standard AFEM refinements, while retaining a $1.76\times$ speedup. Thus, the learned mesh is effective both directly and as an initialization for subsequent AFEM correction.

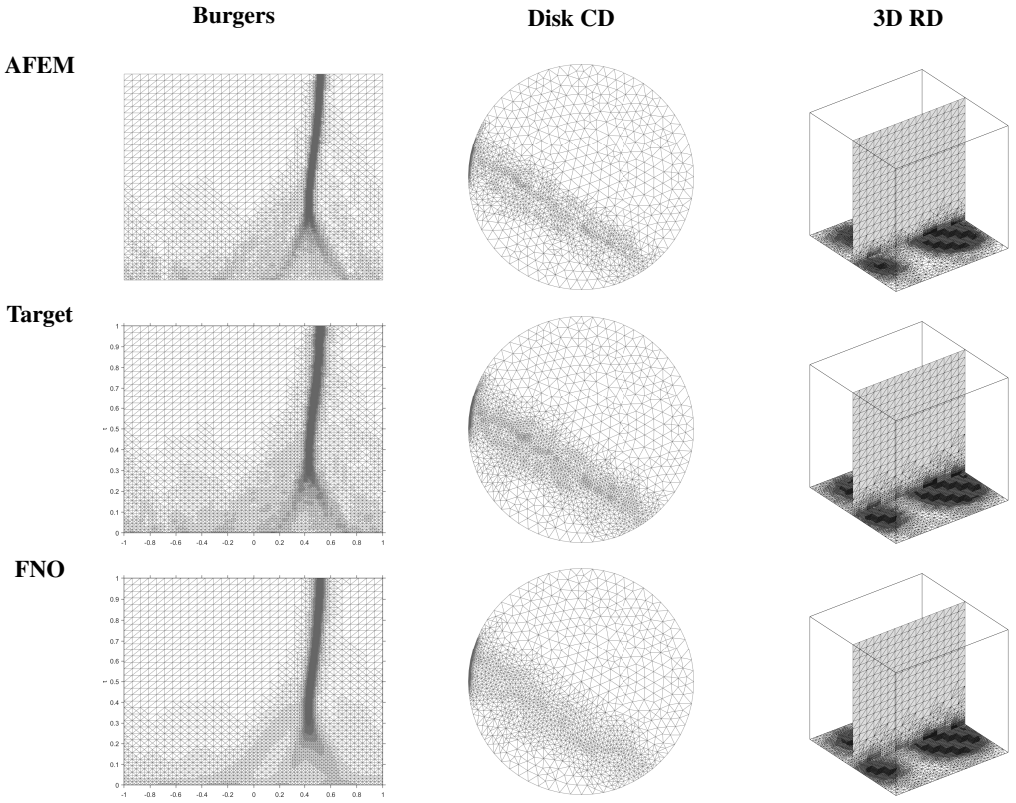


Figure 3: Conforming meshes generated by standard AFEM, target fields, and FNO-predicted fields for Burgers (test 4), disk convection–diffusion (test 3), and 3D reaction–diffusion (test 12).

Table 3: Finite-element performance at the target-generation level, averaged over 100 test instances; "Cycles" denotes the mean number of AFEM refinement cycles over the test instances.

Problem	Method	Rel. L^2	DOF	Cycles	Time (s)
Burgers	FNO	1.243e-3	14869	0	0.458
	Target-field	9.810e-4	17928	0	0.552
	Fixed-12-cycle AFEM	9.330e-4	16377	12	1.341
	Reference-error-matched AFEM	1.235e-3	11364	11.290	1.073
	Uniform FEM	1.588e-3	422500	0	15.094
Disk CD	FNO	2.944e-2	2914	0	0.167
	Target-field	2.149e-2	3949	0	0.210
	Fixed-12-cycle AFEM	2.307e-2	3004	12	0.420
	Reference-error-matched AFEM	2.899e-2	2136	10.850	0.351
	Quasi-uniform FEM	4.130e-2	52614	0	1.068
3D RD	FNO	3.071e-2	71765	0	3.027
	Target-field	2.954e-2	82258	0	3.502
	Fixed-10-cycle AFEM	3.345e-2	47875	10	9.259
	Uniform FEM	3.832e-2	300763	0	5.253

5.5 Online timing analysis

To identify the source of the online speedup, Table 5 decomposes the mean wall-clock time into mesh construction, finite-element assembly, system solution, and other adaptive overhead. Detailed timing conventions are provided in Appendix D.2.

Across all three problems, the speedup primarily comes from avoiding repeated adaptive mesh-discovery cycles rather than accelerating the algebraic solver. For Burgers and disk convection–

Table 4: Finite-element performance after AFEM correction; "Cycles" denotes the mean number of AFEM refinement cycles over the test instances.

Problem	Method	Rel. L^2	DOF	Cycles	Time (s)
Burgers	FNO + 2 AFEM	5.172e-4	33880	2	1.559
	Target + 2 AFEM	4.837e-4	41191	2	1.931
	Fixed-14-cycle AFEM	4.263e-4	45321	14	2.949
	Reference-error-matched AFEM	5.075e-4	31788	13.410	2.281
Disk CD	FNO + 3 AFEM	1.700e-2	5284	3	0.424
	Target + 3 AFEM	1.474e-2	7625	3	0.558
	Fixed-15-cycle AFEM	1.417e-2	9027	15	0.851
	Reference-error-matched AFEM	1.696e-2	5170	13.610	0.653
3D RD	FNO + 3 AFEM	8.582e-3	323226	3	35.909
	Target + 3 AFEM	8.309e-3	343769	3	37.776
	Fixed-13-cycle AFEM	9.284e-3	293661	13	63.243
	Uniform FEM	9.319e-3	2571353	0	54.482

Table 5: Online timing breakdown over 100 test instances (seconds). Individual timing components and totals are rounded independently.

Method	Mesh	Assembly	Solve	Other	Total
Space–time Burgers					
FNO	0.113	0.157	0.165	0.024	0.458
Target-field	0.140	0.187	0.199	0.027	0.552
Fixed-12-cycle AFEM	0.190	0.431	0.295	0.425	1.341
Reference-error-matched AFEM	0.126	0.336	0.263	0.348	1.073
Uniform FEM	0.158	5.152	9.487	0.296	15.094
Disk convection–diffusion					
FNO	0.139	0.017	0.010	2.8e-4	0.167
Target-field	0.176	0.021	0.013	0.000	0.210
Fixed-12-cycle AFEM	0.143	0.113	0.053	0.111	0.420
Reference-error-matched AFEM	0.128	0.092	0.040	0.090	0.351
Quasi-uniform FEM	0.499	0.208	0.361	0.000	1.068
3D reaction–diffusion					
FNO	1.202	0.621	0.334	0.869	3.027
Target-field	1.401	0.725	0.391	0.984	3.502
Fixed-10-cycle AFEM	6.091	1.141	0.340	1.686	9.259
Uniform FEM	0.650	3.793	0.601	0.209	5.253

diffusion, FNO replaces repeated assembly, estimation, marking, and refinement with direct score-based mesh realization. In the disk problem, the sparse solve itself is inexpensive, so the remaining standard-AFEM cost mainly arises from repeated assembly and adaptive bookkeeping. The effect is most pronounced in three dimensions, where fixed-10-cycle AFEM spends $6.091\,\mathrm{s}$ of its $9.259\,\mathrm{s}$ total time in mesh-related operations, compared with $1.202\,\mathrm{s}$ for FNO-based mesh realization and $3.027\,\mathrm{s}$ in total. Including offline costs, break-even requires about 2.5×10^3–4.2×10^4 queries; see Appendix D.3.

5.6 Discussion and limitations

Across the three benchmarks, MeshOp reduces sequential mesh-discovery cost while remaining compatible with conventional FEM and residual-based AFEM correction. Non-negligible pointwise score errors need not translate proportionally into finite-element errors, suggesting that recovering the dominant refinement regions matters more than exact target recovery.

The observed downstream tolerance is empirical and does not constitute a stability guarantee. Because each model is trained separately for one PDE family on a fixed geometry, cross-PDE, cross-geometry, and broader distribution-shift generalization beyond the Burgers OOD test remain future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code development and debugging, provide feedback on research methodology, experimental evaluation, and presentation, and improve the language, clarity, and readability of the manuscript. We did not use generative AI tools to formulate mathematical claims, assist with mathematical proofs, propose or refine hypotheses, or generate numerical results; all other required disclosure tasks were not applicable.

All AI-assisted code was manually inspected and tested by the authors. Numerical results and reference solutions were independently validated, literature citations were checked against their original sources, and all scientific claims were reviewed by the authors. All figures and schematics were prepared by the authors using conventional scientific plotting and drawing software. We take responsibility for the final content of this work, including text, claims, code, and other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies numerical methods using synthetic PDE benchmarks and does not involve human subjects, personal data, or safety-critical applications. We are not aware of any direct ethical risks arising from this study.

REPRODUCIBILITY STATEMENT

The main paper and appendices describe the PDE definitions, AFEM target-generation protocols, neural-operator training settings, mesh-realization procedure, solver configurations, and evaluation metrics, with the corresponding datasets made available for download. Additional implementation details, including reference-solution construction, elementwise score aggregation, the online mesh-realization algorithm, solver initialization, and timing conventions, are provided in the appendix. Source code, experimental configurations, and supporting data are available at `https://anonymous.4open.science/r/MeshOp/` to facilitate reproduction of the reported experiments.

REFERENCES

Peter Binev, Wolfgang Dahmen, and Ron DeVore. Adaptive Finite Element Methods with convergence rates. *Numerische Mathematik*, 97(2):219–268, 2004.

J Manuel Cascon, Christian Kreuzer, Ricardo H Nochetto, and Kunibert G Siebert. Quasi-Optimal Convergence Rate for an Adaptive Finite Element Method. *SIAM Journal on Numerical Analysis*, 46(5):2524–2550, 2008.

Chiu Ling Chan, Felix Scholz, and Thomas Takacs. Locally refined quad meshing for linear elasticity problems based on convolutional neural networks. *Engineering with Computers*, 38(5): 4631–4652, 2022.

Willy Dörfler. A Convergent Adaptive Algorithm for Poisson's Equation. *SIAM Journal on Numerical Analysis*, 33(3):1106–1124, 1996.

Niklas Freymuth, Philipp Dahlinger, Tobias Würth, Simon Reisch, Luise Kärger, and Gerhard Neumann. Swarm reinforcement learning for adaptive mesh refinement. In *Advances in Neural Information Processing Systems*, volume 36, pp. 73312–73347, 2023.

Niklas Freymuth, Tobias Würth, Nicolas Schreiber, Balázs Gyenes, Andreas Boltres, Johannes Mitsch, Aleksandar Taranovic, Tai Hoang, Philipp Dahlinger, Philipp Becker, Luise Kärger, and Gerhard Neumann. AMBER: Adaptive Mesh Generation by Iterative Mesh Resolution Prediction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Daniel Greenfeld, Meirav Galun, Ronen Basri, Irad Yavneh, and Ron Kimmel. Learning to optimize multigrid PDE solvers. In *International Conference on Machine Learning*, volume 97, pp. 2415–2423. PMLR, 2019.

Peiyan Hu, Yue Wang, and Zhi-Ming Ma. Better Neural PDE Solvers Through Data-Free Mesh Movers. In *International Conference on Learning Representations*, 2024.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Martin Legeland, Kevin Linka, Roland C Aydin, and Christian J Cyron. Non-iterative generation of optimized meshes for finite element simulations with deep learning. *Machine Learning for Computational Science and Engineering*, 1(1):12, 2025.

Haoqian Li, Jiguang Sun, and Zhiwen Zhang. Deep Eigenspace Network for Parametric Non-self-adjoint Eigenvalue Problems. *arXiv preprint arXiv:2512.20058*, 2025.

Yichen Li, Peter Yichen Chen, Tao Du, and Wojciech Matusik. Learning preconditioners for conjugate gradient PDE solvers. In *International Conference on Machine Learning*, volume 202, pp. 19425–19439. PMLR, 2023a.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023b.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*, 2021.

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.

Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

Joseph M Maubach. Local bisection refinement for n-simplicial grids generated by reflection. *SIAM Journal on Scientific Computing*, 16(1):210–227, 1995.

Pedro Morin, Ricardo H. Nochetto, and Kunibert G. Siebert. Convergence of Adaptive Finite Element Methods. *SIAM Review*, 44(4):631–658, 2002.

Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional Neural Operators for robust and accurate learning of PDEs. In *Advances in Neural Information Processing Systems*, volume 36, pp. 77187–77200, 2023.

Rob Stevenson. The completion of locally refined simplicial partitions created by bisection. *Mathematics of Computation*, 77(261):227–241, 2008.

Tailin Wu, Takashi Maruyama, Qingqing Zhao, Gordon Wetzstein, and Jure Leskovec. Learning Controllable Adaptive Simulation for Multi-resolution Physics. In *International Conference on Learning Representations*, 2023.

Jing Xiao, Xinhai Chen, Jiaming Peng, Qinglin Wang, and Jie Liu. MeshONet: a generalizable and efficient operator learning method for structured mesh generation. *Neural Networks*, pp. 108746, 2026.

Zhiyuan Zhang, Amir Vaxman, Stefanos-Aldo Papanicolopulos, and Kartic Subr. Learned Adaptive Mesh Generation. *arXiv preprint arXiv:2505.20457*, 2025.

Table 6: Sensitivity of FNO refinement-field prediction to the number of training samples. All results are evaluated on the same 100 independent test instances for each problem.

Train samples	MAE	RMSE	Exact-level acc	Within 1 score	Rel. score L^2
Space–time Burgers					
1000	0.366	0.567	0.704	0.910	0.231
2000	0.345	0.533	0.717	0.922	0.222
3000	0.336	0.518	0.726	0.927	0.217
Disk convection–diffusion					
1000	0.382	0.726	0.723	0.858	0.407
2000	0.335	0.641	0.744	0.883	0.358
3000	0.327	0.616	0.746	0.886	0.350
Three-dimensional steady reaction–diffusion					
1000	0.165	0.340	0.877	0.975	0.093
2000	0.161	0.328	0.881	0.977	0.090
3000	0.157	0.320	0.886	0.979	0.088

Appendix Overview

The appendix is organized as follows:

- Appendix A presents sensitivity analyses for the training-set size and score-to-generation threshold, together with neural-operator backbone comparisons and random-seed robustness.
- Appendix B compares MeshOp with direct solution prediction and U-Net-based mesh-size prediction, and reports out-of-distribution tests for Burgers.
- Appendix C provides implementation details on reference solutions, AFEM target generation and marking, model configurations, solvers, score aggregation, and the online algorithm.
- Appendix D presents the online complexity analysis and detailed timing conventions.
- Appendix E provides additional experimental results for all three benchmark problems, including neural-operator comparisons, mesh realizations, AFEM correction, and solution-error visualizations.

A Sensitivity Analyses

A.1 Sensitivity to training-set size

We further examine the sensitivity of FNO refinement-field prediction to the number of training instances. The model architecture and evaluation protocol are kept fixed, while the training-set size is varied from 1000 to 3000 samples.

Table 6 shows consistent improvements with increasing training-set size across all three problems. Burgers and disk convection–diffusion exhibit gradual gains over the full range from 1000 to 3000 samples, while reaction–diffusion shows relatively little improvement beyond 2000 samples.

A.2 Neural-operator backbone comparison and random-seed robustness

We further examine the dependence of refinement-field prediction on the neural-operator backbone and stochastic training variability. FNO, CNO, DeepONet, and POD-DeepONet are compared on the same Burgers dataset using 3000 training instances and 100 independent test instances. The FNO experiment is additionally repeated with five random seeds (40–44), while all other settings are kept fixed. Table 7 reports the five-seed mean and standard deviation for FNO together with the principal seed-42 results for the other backbones.

Table 7: Neural-operator backbone comparison and random-seed robustness for the Burgers refinement field. FNO results are reported as mean ± standard deviation over five random seeds (40–44); the other backbones use the principal seed 42. All models use the same 3000 training instances and 100 independent test instances.

Model	**MAE**	**RMSE**	**Exact-level acc.**	**Within 1**	**Rel. score L^2**
FNO (5 seeds)	0.336 ± 0.004	0.519 ± 0.006	0.725 ± 0.002	0.927 ± 0.002	0.217 ± 0.001
CNO	0.345	0.545	0.725	0.924	0.223
DeepONet	0.573	0.925	0.607	0.819	0.367
POD-DeepONet	0.529	0.818	0.623	0.838	0.340

Table 8: Sensitivity of the FNO-generated Burgers meshes to the score-to-generation threshold τ. Values are means over the same 100 test instances, with no additional AFEM correction.

τ	**Space–time rel. L^2**	**Final-time rel. L^2**	**Free DOF**	**Time (s)**
0.1	9.783e-4	1.485e-3	19291	0.549
0.3	1.096e-3	1.646e-3	16905	0.475
0.5	1.243e-3	1.858e-3	14869	0.458
0.7	1.409e-3	2.085e-3	13108	0.396
0.9	1.601e-3	2.383e-3	11588	0.348

FNO and CNO provide the strongest refinement-field prediction among the considered backbones, with FNO achieving slightly lower MAE, RMSE, and relative score error. The small standard deviations across the five FNO runs also indicate limited sensitivity to stochastic initialization and training variability.

A.3 Sensitivity to the score-to-generation threshold

We further examine the sensitivity of the Burgers experiment to the score-to-generation threshold τ in equation 11. The trained FNO and all other settings are kept fixed, while only τ is varied. A smaller threshold converts the predicted score into refinement generations more aggressively, whereas a larger threshold produces a relatively coarser mesh.

Table 8 shows that increasing τ produces coarser FNO-generated meshes and reduces both the number of free degrees of freedom and the online computational time, while the space–time and final-time errors increase gradually. The threshold therefore provides a direct accuracy–cost control. We use $\tau = 0.5$ in the main experiments as an intermediate choice between refinement cost and solution accuracy.

B Alternative Baselines and Additional Generalization Tests

B.1 Direct Solution Prediction versus Refinement-Field Prediction

A central design choice in MeshOp is to learn the AFEM-derived refinement field rather than directly approximate the PDE solution. To examine this choice, we train neural operators to predict the corresponding reference solutions directly using the same 3000 training instances and 100 test instances. For Burgers, we consider FNO, CNO, DeepONet, and POD-DeepONet, while FNO is used for the disk convection–diffusion and three-dimensional reaction–diffusion benchmarks.

Direct solution prediction has substantially lower inference cost, since it requires only a neural-network forward pass. However, under the present controlled setting, its solution errors are consistently larger than those obtained by solving the PDE on MeshOp-generated meshes, as shown in Table 9. For Burgers, this trend is observed across all four neural-operator backbones, with direct-solution errors approximately 24–55 times larger. FNO shows the same qualitative behavior on disk convection–diffusion and three-dimensional reaction–diffusion, with error ratios of $8.79\times$ and $1.85\times$, respectively.

Table 9: Comparison between direct neural-operator solution prediction and refinement-field prediction followed by conventional FEM. Errors are mean relative L^2 errors over the same 100 test instances. Direct inference time denotes neural-operator inference only, whereas MeshOp online time includes operator inference, mesh realization, finite-element assembly, and solution.

Problem	**Backbone**	**Direct rel.** L^2	**Infer. (s)**	**MeshOp rel.** L^2	**Online (s)**
Burgers	FNO	2.999e-2	1.400e-2	1.243e-3	0.458
Burgers	CNO	4.808e-2	5.998e-3	1.145e-3	0.506
Burgers	DeepONet	1.479e-1	4.617e-4	3.418e-3	0.348
Burgers	POD-DeepONet	1.542e-1	2.879e-4	2.792e-3	0.373
Disk CD	FNO	2.588e-1	2.866e-3	2.944e-2	0.167
3D RD	FNO	5.691e-2	5.910e-2	3.071e-2	3.027

Table 10: Burgers comparison between MeshOp-FNO and the adapted U-Net/Gmsh mesh-size baseline. Results are averaged over the same 100 test instances.

Method	τ	**ST rel.** L^2	**FT rel.** L^2	**Free DOF**	**Time (s)**
MeshOp-FNO	0.1	9.783e-4	1.485e-3	19291	0.549
MeshOp-FNO	0.3	1.096e-3	1.646e-3	16905	0.475
MeshOp-FNO	0.5	1.243e-3	1.858e-3	14869	0.458
U-Net/Gmsh	–	8.764e-4	1.269e-3	16123	1.107

B.2 Comparison with U-Net-Based Mesh-Size Prediction

Among the learning-based mesh-adaptation approaches discussed in Section 2, Chan et al. (2022) provides the closest controlled baseline for our setting because it likewise uses one-shot prediction of a fixed-resolution mesh-control field, conventional mesh realization, and a downstream finite-element solve. Following the continuous mesh-size prediction strategy of Chan et al. (2022), we construct an adapted Burgers baseline in which a parameter-matched U-Net predicts a continuous mesh-size field and Gmsh realizes the corresponding space–time mesh. The original method targets static two-dimensional elasticity with quadrilateral meshes; our adaptation retains its mesh-size-prediction and Gmsh-generation principle, while using the same Burgers problem, periodic P1 discretization, training/test split, reference solutions, and error evaluator as MeshOp. Gmsh operates in normalized coordinates $(\xi, \tau) = ((x+1)/2, t)$, consistent with MeshOp's logical 32×32 initial partition. The U-Net targets are derived from the same fixed-12-cycle AFEM meshes used for MeshOp target generation by mapping the sampled leaf-generation field $s^\star$ to $h = (1/32)2^{-s^\star/2}$. The U-Net and FNO contain 48,064 and 48,190 real trainable parameters, respectively, and their online times follow the same timing convention.

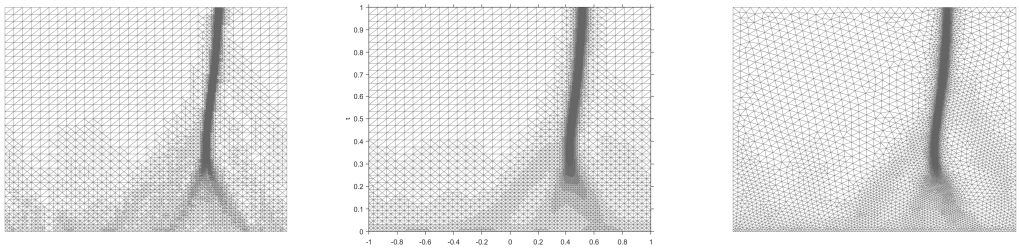
At the closest tested space–time accuracy, Table 10 shows that MeshOp-FNO with $\tau = 0.1$ is $2.02\times$ faster than U-Net/Gmsh (0.549 versus 1.107 s), whereas U-Net/Gmsh uses fewer DOFs and attains slightly lower space–time and final-time errors. Its online time comprises 0.671 s for the Gmsh-based mesh-construction pipeline, 0.182 s for assembly, 0.225 s for linear solves, and 0.029 s for inference and other overhead. Mesh construction accounts for 60.7% of its online cost using the unrounded measurements because Gmsh generates a new global unstructured mesh from the predicted continuous size field for each instance, rather than locally refining a reusable hierarchical mesh. Thus, continuous mesh-size prediction is accurate and DOF-efficient, whereas MeshOp provides lower online latency through refinement-field prediction and hierarchical NVB construction.

B.3 Out-of-Distribution Tests for Burgers

To assess generalization beyond the training distribution, we evaluate the trained Burgers MeshOp model on 100 out-of-distribution (OOD) instances without retraining. The OOD initial conditions augment the in-distribution shifted first-harmonic component with an independent second harmonic,

$$u_0(x) = -a_1 \sin(\pi(x + b_1)) - a_2 \sin(2\pi(x + b_2)) + g(x),$$

where $a_1 \sim \mathcal{U}(0.5, 1.5)$, $a_2 \sim \mathcal{U}(0.5, 1.0)$, $b_1, b_2 \sim \mathcal{U}(-0.85, 0.85)$, and g follows the same Gaussian random field distribution as in training. The PDE, forcing distribution, AFEM target-generation procedure, and evaluation protocol are otherwise kept unchanged.



Standard AFEM MeshOp-FNO U-Net/Gmsh

Figure 4: Meshes for Burgers Test 004: fixed-12-cycle standard AFEM, MeshOp-FNO without additional AFEM correction, and the U-Net/Gmsh mesh-size baseline.

Table 11: Finite-element performance on 100 out-of-distribution Burgers instances. MeshOp-FNO is evaluated without retraining; "Cycles" denotes the mean number of AFEM refinement cycles over the test instances.

Method	**ST rel. L^2**	**FT rel. L^2**	**DOF**	**Cycles**	**Time (s)**
Direct mesh realization					
MeshOp-FNO	3.928e-3	8.418e-3	11553	0	0.456
Reference-error-matched AFEM	3.781e-3	7.199e-3	8280	10.170	0.998
Higher-accuracy comparison					
MeshOp-FNO + 2 AFEM	1.601e-3	3.015e-3	19728	2	1.163
Reference-error-matched AFEM	1.539e-3	2.779e-3	20006	12.020	1.748

As shown in Table 11 and Figure 5, MeshOp-FNO retains useful adaptive-refinement information under this distribution shift without retraining. Although the direct prediction exhibits some degradation, it achieves accuracy comparable to reference-error-matched AFEM while reducing the mean online time from 0.998 s to 0.456 s, corresponding to a $2.19\times$ speedup. After two additional AFEM correction cycles, the finite-element accuracy is further improved while retaining a $1.50\times$ speedup over the corresponding reference-error-matched AFEM baseline.

C Additional Implementation Details

C.1 Reference solution generation

For the space–time Burgers problem, the reference solution is computed using a periodic Fourier pseudo-spectral method with ETDRK4 time integration and time step 2.5×10^{-4}. For disk convection–diffusion, we use a Fourier–Chebyshev spectral discretization with angular cutoff $M = 128$ and 255 radial intervals; a coarser $M = 64$, 127-interval solve is used to verify spectral resolution. For reaction–diffusion, the reference is obtained from the exact modal representation of the linear operator, using a 256×256 horizontal sine-spectral discretization and the exact vertical dependence $\sinh(\mu(1-z))/\sinh(\mu)$ on a 161-point grid clustered near $z = 0$. In all cases, the random input fields are reconstructed from the stored Fourier/GRF coefficients.

C.2 AFEM discretization and marking

Initial meshes. All experiments start from a fixed, sample-independent initial mesh. For Burgers, $\mathcal{T}_0$ is a 32×32 Cartesian partition of the space–time domain into P1 triangles. For the disk convection–diffusion problem, $\mathcal{T}_0$ is a Delaunay triangulation of the unit disk generated by MATLAB's PDE Toolbox `initmesh` with $H_{\max} = 0.1$ and a fixed random seed, yielding 541 vertices and 1016 triangles. For reaction–diffusion, $\mathcal{T}_0$ is a 16^3 Cartesian partition of the unit cube, with each cube divided into six Kuhn tetrahedra. The same initial mesh is used in data generation and in all downstream FEM evaluations.

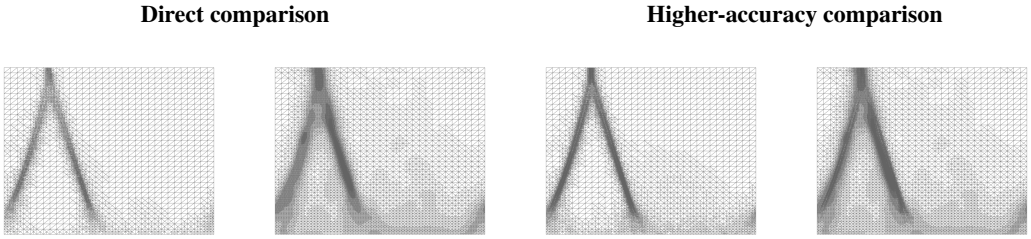


Figure 5: Meshes for Burgers OOD Test 001. The left pair shows the direct comparison, while the right pair compares MeshOp-FNO after two AFEM correction cycles with the corresponding reference-error-matched AFEM baseline. MeshOp-FNO is evaluated without retraining.

Residual indicators and Dörfler marking. The element indicators used for AFEM marking are

$$\eta_{K,\mathrm{B}}^2 = h_K^2 \|f - u_{h,t} - u_h u_{h,x}\|_K^2 + \frac{1}{2} \sum_{E \in \mathcal{E}_K^{\mathrm{int,per}}} h_E \|[\nu u_{h,x} n_x]\|_E^2 + \sum_{E \subset \partial K \cap \{t=0\}} \|u_0 - I_h u_0\|_E^2,$$

$$\eta_{K,\mathrm{CD}}^2 = h_K^2 \|f - \boldsymbol{\beta} \cdot \nabla u_h\|_K^2 + \frac{1}{2} \sum_{E \in \mathcal{E}_K^{\mathrm{int}}} h_E \|[\varepsilon \nabla u_h \cdot n_E]\|_E^2,$$

$$\eta_{K,\mathrm{RD}}^2 = \alpha_K^2 \|u_h\|_K^2 + \frac{1}{2} \sum_{F \in \mathcal{F}_K^{\mathrm{int}}} \frac{\alpha_F}{\varepsilon} \|[\varepsilon^2 \nabla u_h \cdot n_F]\|_F^2 + \sum_{F \subset \partial K \cap \{z=0\}} \frac{\varepsilon^2}{h_F} \|g_D - I_h g_D\|_F^2, \tag{21}$$

where $\|\cdot\|_D$ is the $L^2(D)$ norm, $\alpha_K = \min\{h_K/\varepsilon, 1\}$ and $\alpha_F = \min\{h_F/\varepsilon, 1\}$. At each cycle, the smallest set $\mathcal{M}_k$, after sorting the indicators in descending order, is selected such that

$$\sum_{K \in \mathcal{M}_k} \eta_K^2 \geq \theta \sum_{K \in \mathcal{T}_k} \eta_K^2, \qquad \theta = 0.8. \tag{22}$$

Periodic seam edges are included for Burgers, and conformity closure is applied after marking.

C.3 Neural-operator architectures and training

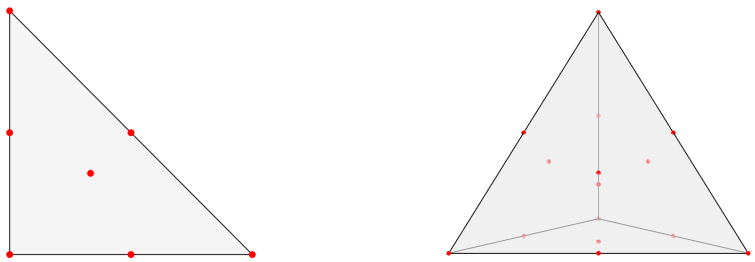
The neural-operator configurations used in the principal experiments and Burgers backbone comparison are summarized in Table 12. For the Burgers comparison, the four neural operators are configured with comparable parameter counts (approximately 4.5×10^4–5.5×10^4 trainable parameters) to reduce differences arising solely from model capacity. The Burgers and three-dimensional reaction–diffusion models are trained using the standard grid-point MSE, whereas disk convection–diffusion uses the physical-area-weighted MSE in equation 9. For reaction–diffusion, the standard MSE on the stretched grid $z_i = (i/64)^2$ emphasizes refinement near the random Dirichlet boundary at $z = 0$ and is distinct from the physical relative L^2 solution error. All models use AdamW with an initial learning rate of 10^{-3}, weight decay 10^{-6}, and a cosine-annealing learning-rate schedule with a minimum learning rate of 10^{-5}. Unless otherwise stated, all single-run results use seed 42 and final-epoch checkpoints.

C.4 Linear solvers and initialization

The algebraic solver is chosen according to each PDE. For Burgers, we use damped Newton iterations with MATLAB's sparse direct solver for the Jacobian systems. For disk convection–diffusion, the linear SUPG system is solved directly using MATLAB's sparse solver. For reaction–diffusion, the symmetric positive-definite system is solved by PCG with diagonal preconditioning and an incomplete-Cholesky fallback. For the Burgers and reaction–diffusion adaptive runs, the solution on the previous mesh is interpolated to the refined mesh and used as the initial iterate, while the learned- and target-mesh solves are initialized from the common coarse-grid solution. For disk convection–diffusion, the direct sparse solve requires neither an initial iterate nor coarse-grid initialization.

Table 12: Learned-model architectures and training configurations used in the experiments. Parameter counts denote equivalent real-valued trainable parameters, with each complex-valued parameter counted as two real scalars.

Problem	Model	Architecture	Params.	Epochs	Batch
Burgers	FNO	Width 9; 4 Fourier layers; 6×6 Fourier modes	48190	1000	16
Burgers	CNO	Width 10; 2 resolution levels; 2 bottleneck and 2 intermediate residual blocks	53181	1000	16
Burgers	DeepONet	Latent dim. 64; hidden dim. 56; branch/trunk depth $4/4$	48961	10000	16
Burgers	POD-DeepONet	Hidden dim. 56; depth 4; POD rank ≤ 196	46284	10000	8
Burgers	U-Net	Channel widths 3–6–12–24–27; continuous mesh-size prediction	48064	1000	16
Disk convection–diffusion	FNO	Width 12; 4 Fourier layers; 8×6 Fourier modes	112281	1000	16
Reaction–diffusion	FNO	Width 11; 4 Fourier layers; $5 \times 5 \times 5$ Fourier modes	485504	1000	8



Triangle: 7 query points Tetrahedron: 15 query points

Figure 6: Elementwise query points used to aggregate the predicted refinement score during mesh realization.

C.5 Elementwise query-point aggregation

During score-to-mesh realization, the predicted refinement field is evaluated at a small set of representative points $\mathcal{Q}(K)$ in each current leaf element K. For the triangular elements in the Burgers and disk convection–diffusion problems, we use seven query points consisting of the three vertices, three edge midpoints, and the element centroid. For the tetrahedral elements in the reaction–diffusion problem, we use fifteen query points consisting of the four vertices, six edge midpoints, four face centroids, and the element centroid. Figure 6 illustrates these configurations.

The element score is defined by

$$\overline{s}_K = \max_{\boldsymbol{x} \in \mathcal{Q}(K)} \widehat{s}^{\,c}_\theta(\xi, \boldsymbol{x}), \tag{23}$$

before conversion to an integer refinement generation. This maximum-based aggregation is deliberately conservative: a locally large predicted refinement demand is sufficient to trigger refinement, reducing the risk of missing narrow or localized high-refinement regions.

Algorithm 1 Online mesh realization and finite-element solution

Require: PDE input ξ, trained operator $\mathcal{G}_\theta$, initial mesh $\mathcal{T}_0$, maximum generation G, threshold τ, query sets $\mathcal{Q}(K)$

Ensure: Conforming mesh $\widehat{\mathcal{T}}$ and finite-element solution $u_{\widehat{\mathcal{T}}}$

Predict $\widehat{s}_\theta = \mathcal{G}_\theta(\xi)$, clip it to $[0, G]$, and construct its interpolant $\widehat{s}^c_\theta$.
Initialize $\widehat{\mathcal{T}} \leftarrow \mathcal{T}_0$.
repeat
  $\mathcal{M} \leftarrow \emptyset$.
  for each leaf element $K \in \widehat{\mathcal{T}}$ **do**
    Set $\overline{s}_K = \max_{\boldsymbol{x} \in \mathcal{Q}(K)} \widehat{s}^c_\theta(\xi, \boldsymbol{x})$ and $\widehat{g}_K = \lfloor \overline{s}_K + 1 - \tau \rfloor$.
    if $\mathrm{gen}(K) < \widehat{g}_K$ **then**
      $\mathcal{M} \leftarrow \mathcal{M} \cup \{K\}$.
    end if
  end for
  if $\mathcal{M} \neq \emptyset$ **then**
    Refine $\mathcal{M}$ by NVB and enforce mesh conformity.
  end if
until $\mathcal{M} = \emptyset$
Assemble and solve the finite-element system on $\widehat{\mathcal{T}}$ to obtain $u_{\widehat{\mathcal{T}}}$.
if additional accuracy is required **then**
  Continue standard residual-based solve–estimate–mark–refine cycles from $\widehat{\mathcal{T}}$, updating $\widehat{\mathcal{T}}$ and $u_{\widehat{\mathcal{T}}}$.
end if
return $\widehat{\mathcal{T}}$, $u_{\widehat{\mathcal{T}}}$.

C.6 Detailed online algorithm

Algorithm 1 summarizes the complete online procedure, including neural-operator inference, recursive mesh realization, finite-element solution, and optional residual-based AFEM correction.

D Computational Cost Analysis

D.1 Online complexity

Let M_r denote the number of leaf elements at mesh-realization step r, M the number of leaf elements in the final realized mesh, and $q = |\mathcal{Q}(K)|$ the fixed number of query points per element. If mesh realization terminates after $R \leq G$ refinement steps, evaluating the predicted score on the current leaf elements requires

$$\mathcal{O}\left(q \sum_{r=0}^{R} M_r\right) = \mathcal{O}(M), \tag{24}$$

since q and the maximum generation G are fixed and $M_r \leq M$ for all $r \leq R \leq G$. The remaining mesh-construction cost consists of hierarchical NVB refinement and conformity closure, denoted by $C_{\mathrm{ref}}(M)$.

Ignoring the comparatively small neural-operator inference, interpolation, and initialization overheads, the online cost before optional AFEM correction can be expressed as

$$C_{\mathrm{MeshOp}} = \mathcal{O}(M) + C_{\mathrm{ref}}(M) + C_{\mathrm{asm}}(n_M) + C_{\mathrm{solve}}(n_M), \tag{25}$$

where $C_{\mathrm{asm}}(n_M)$ and $C_{\mathrm{solve}}(n_M)$ denote finite-element assembly and algebraic solution on the realized mesh with n_M free degrees of freedom.

For an AFEM trajectory $\{\mathcal{T}_k\}_{k=0}^{K}$, let $N_k := |\mathcal{T}_k|$ be the number of active elements and n_k the number of free finite-element degrees of freedom on $\mathcal{T}_k$.

$$C_{\mathrm{AFEM}} = \sum_{k=0}^{K} \left[C_{\mathrm{asm}}(n_k) + C_{\mathrm{solve}}(n_k)\right] + \sum_{k=0}^{K-1} \left[C_{\mathrm{est}}(N_k) + C_{\mathrm{mark}}(N_k) + C_{\mathrm{ref}}(N_k)\right], \tag{26}$$

Table 13: Offline costs and break-even (BE) numbers of online PDE instances.

Problem	Target gen. (h)	Training (h)	Base-stage BE	Corrected/strict BE
Burgers	0.80	0.91	1.00×10^4	8.52×10^3
Disk CD	0.52	1.61	4.16×10^4	3.36×10^4
3D RD	5.80	13.25	1.10×10^4	2.51×10^3

because finite-element assembly and solution, error estimation, marking, and refinement are repeated along the adaptive trajectory. MeshOp replaces most of this sequential mesh-discovery process by one operator evaluation and one hierarchical mesh realization, followed by a single finite-element assembly and solve.

D.2 Timing conventions

The online wall-clock time reported in Table 5 is decomposed into mesh construction, finite-element assembly, system solution, and other online costs. Mesh construction includes score-based hierarchical mesh realization and conformity closure for the FNO and target-field methods, as well as mesh-management operations during standard AFEM. Assembly and system solution record the costs of the main finite-element assembly and the corresponding linear or nonlinear solves, including all adaptive iterations for standard AFEM and the final-mesh solve for the FNO and target-field methods. When coarse-grid initialization is used, its assembly and linear-solution costs are included in the corresponding components. Other online costs include residual estimation and Dörfler marking where applicable, neural-operator inference, and remaining adaptive bookkeeping.

Offline training, target generation, reference construction, error evaluation, and plotting are excluded. Common initial-mesh templates are not charged per test instance. The fixed quasi-uniform disk mesh is constructed once, and its construction cost is amortized over the 100 test instances. For reaction–diffusion, the timing breakdown corresponds to the fixed-10-cycle target-generation level. For disk convection–diffusion, the FNO and target-field rows correspond to the base stage with no additional AFEM correction, while the quasi-uniform baseline uses a fixed $H_{\max} = 0.01$ disk mesh.

D.3 Offline-cost amortization

To quantify the amortization of offline costs, we define the break-even number of online queries as

$$N_{\mathrm{BE}} = \frac{T_{\mathrm{target}} + T_{\mathrm{train}}}{T_{\mathrm{AFEM}} - T_{\mathrm{MeshOp}}},$$

where T_{target} and T_{train} denote the one-time costs of AFEM target generation and operator training, respectively, while $T_{\mathrm{AFEM}} - T_{\mathrm{MeshOp}}$ is the measured per-instance online saving relative to the corresponding AFEM baseline.

Table 13 reports the measured offline costs and the resulting break-even query counts. Target-generation times correspond to generating all 3000 AFEM training targets, while operator-training times correspond to the FNO models used in the main experiments. The AFEM targets are generated independently across PDE instances using a parallel worker pool. The base-stage break-even counts use the online comparisons in Table 3, while the corrected/strict counts use those in Table 4; for reaction–diffusion, these correspond to comparisons against 10 and 13 standard AFEM refinements, respectively. The reported offline costs include AFEM target generation and operator training, while auxiliary reference construction and evaluation costs are reported separately from the deployment cost. The results show that the one-time offline cost is recovered after approximately 2.5×10^3–4.2×10^4 subsequent PDE queries, depending on the problem and accuracy requirement.

E Additional Experimental Results

This section provides additional results for all three benchmark problems, including comparisons between neural-operator backbones, mesh realizations, AFEM correction, and solution-error visualizations.

Table 14: Finite-element performance of different neural-operator backbones for the Burgers problem, before and after two residual-based AFEM correction cycles. Values are means over the same 100 independent test instances.

Method	**Space–time rel. L^2**	**Final-time rel. L^2**	**Free DOF**	**Time (s)**
Direct mesh realization				
FNO	1.243e-3	1.858e-3	14869	0.458
CNO	1.145e-3	1.762e-3	15911	0.506
DeepONet	3.418e-3	4.894e-3	11015	0.348
POD-DeepONet	2.792e-3	4.431e-3	12218	0.373
After two AFEM correction cycles				
FNO + 2 AFEM	5.172e-4	6.894e-4	33880	1.559
CNO + 2 AFEM	5.188e-4	7.010e-4	34295	1.701
DeepONet + 2 AFEM	1.479e-3	2.147e-3	15439	0.856
POD-DeepONet + 2 AFEM	1.246e-3	1.914e-3	18186	0.966

E.1 Burgers operator comparison and additional results

To examine whether the downstream finite-element performance depends strongly on the neural-operator backbone, we compare FNO, CNO, DeepONet, and POD-DeepONet on the same Burgers test set. Each predicted refinement field is converted into a conforming mesh using the same score-to-mesh procedure with $\tau = 0.5$. Table 14 reports the resulting finite-element performance both before and after two additional residual-based AFEM correction cycles.

CNO gives the lowest errors before AFEM correction, while FNO achieves comparable accuracy with fewer degrees of freedom and lower online time. After two AFEM correction cycles, FNO and CNO reach nearly identical accuracy, whereas DeepONet and POD-DeepONet also improve substantially but remain less accurate. These results indicate that MeshOp is compatible with multiple neural-operator backbones, while the quality of the predicted refinement field influences both the direct-mesh performance and the effectiveness of subsequent AFEM correction.

Figures 7, 8, and 9 provide additional qualitative Burgers results. The learned meshes recover the dominant localized refinement regions identified by fixed-cycle AFEM and the target field, while two residual-based AFEM correction cycles further refine locally under-resolved regions.

E.2 Additional disk convection–diffusion results

Figures 10 and 11 provide additional qualitative results for disk convection–diffusion. The FNO-generated mesh reproduces the dominant localized refinement structure identified by fixed-cycle AFEM and the target field while using substantially fewer degrees of freedom than the fixed quasi-uniform baseline.

E.3 Additional reaction–diffusion results

Figures 12 and 13 provide additional qualitative comparisons for reaction–diffusion. Figure 12 compares cross-sectional tetrahedral meshes on the $z = 0$ boundary plane and the $y = 0.5$ interior plane for standard AFEM, the mesh generated from the target refinement field, and FNO. Figure 13 compares the corresponding finite-element solutions and pointwise absolute errors for standard AFEM and FNO on the same two slices.

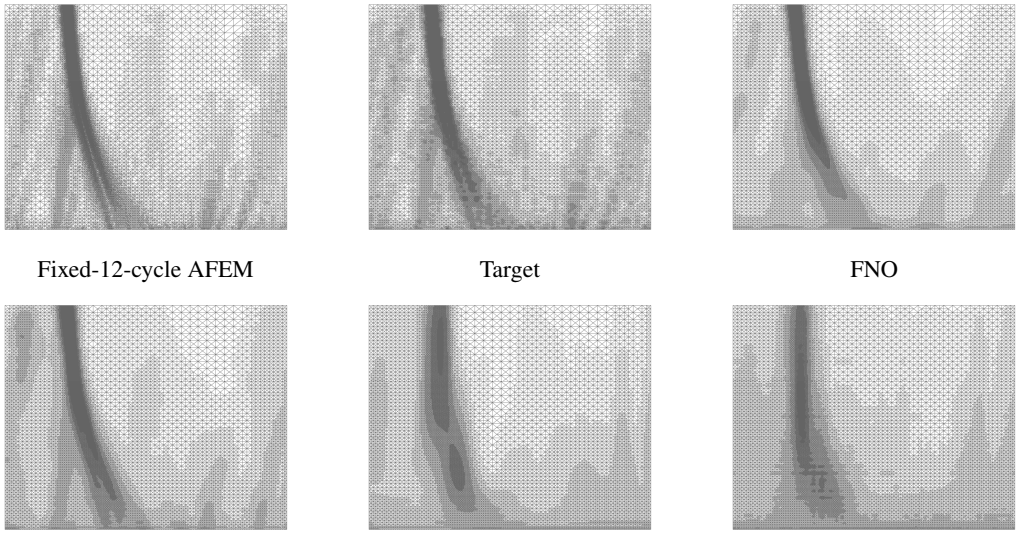


Figure 7: Conforming periodic meshes for Burgers Test 069 constructed by fixed-12-cycle AFEM, target refinement field, and the refinement fields predicted by different neural operators. No additional AFEM correction is applied to the target-field or learned-score meshes.

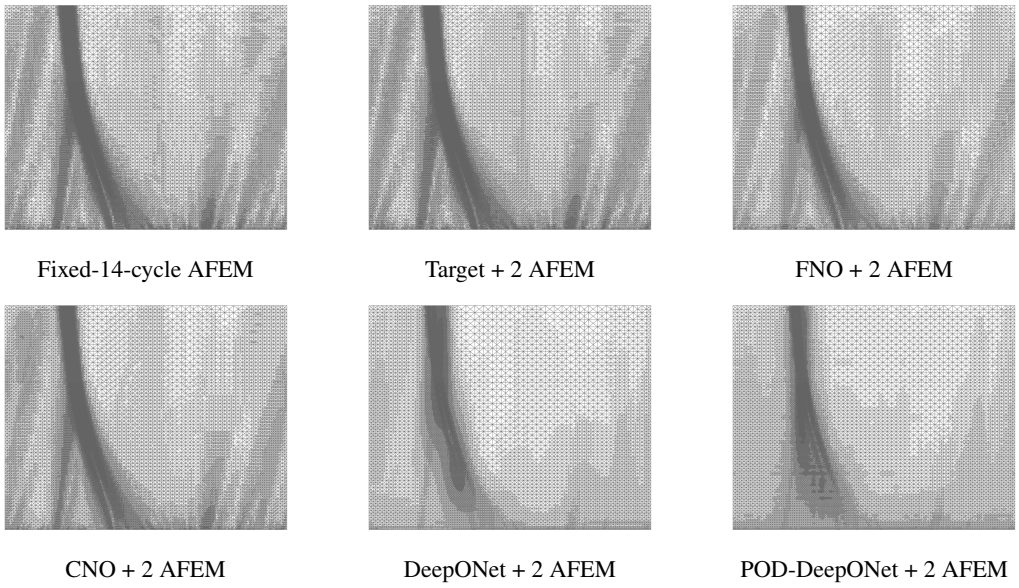


Figure 8: Conforming periodic meshes for Burgers Test 069 after two residual-based AFEM correction cycles applied to the target-field and learned-score initializations.

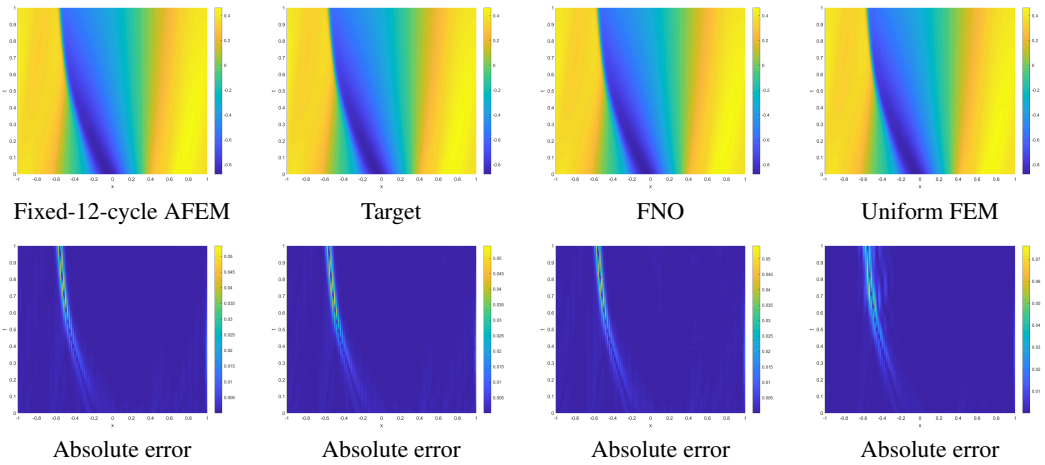


Figure 9: Finite-element solutions and pointwise absolute errors for Burgers Test 069. Columns correspond to fixed-12-cycle AFEM, the target-field mesh, the FNO-generated mesh, and fixed-size uniform FEM, respectively. The top row shows the numerical solutions, and the bottom row shows the corresponding pointwise absolute errors with respect to the reference solution.

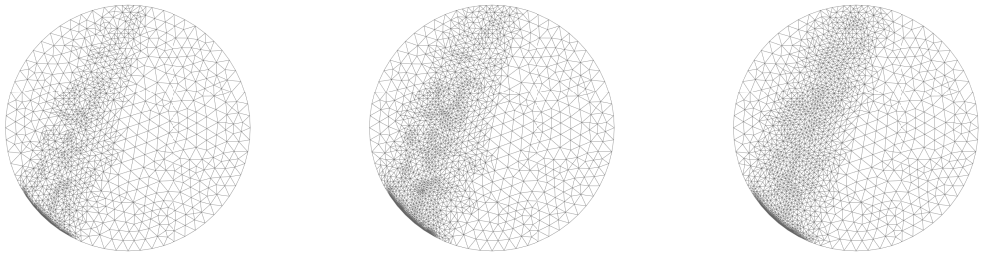


Figure 10: Representative conforming meshes for disk convection–diffusion Test 005 obtained by fixed-12-cycle AFEM, target refinement field, and the FNO-predicted refinement field.

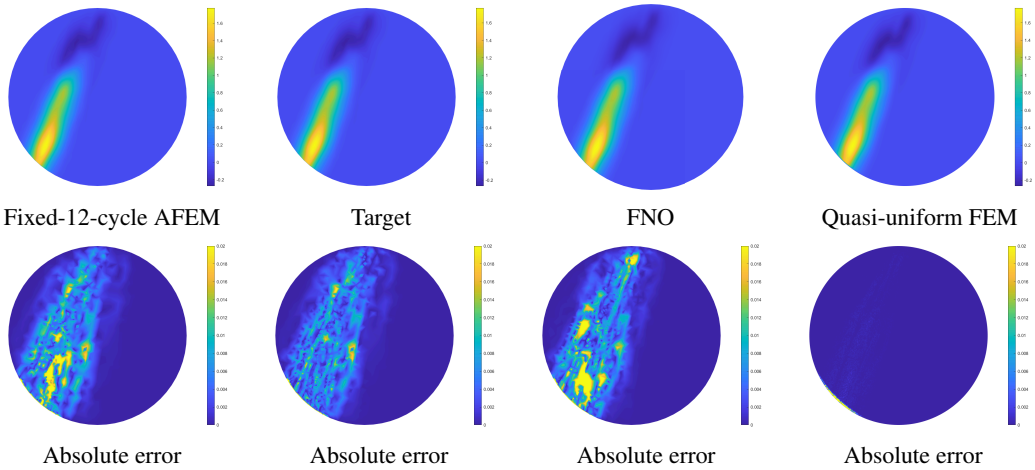
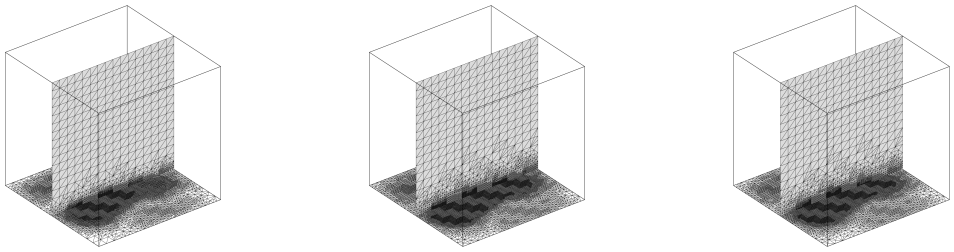
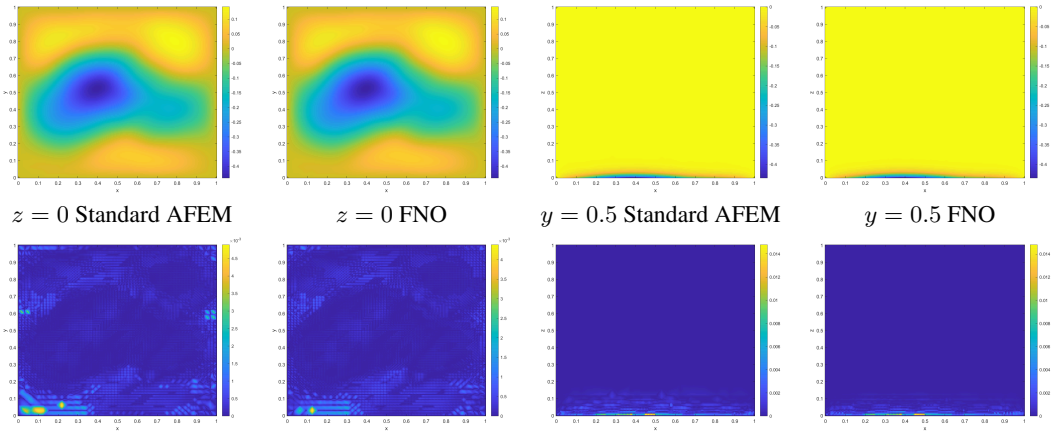


Figure 11: Finite-element solutions and pointwise absolute errors for disk convection–diffusion Test 005. Columns correspond to fixed-12-cycle AFEM, the target-field mesh, the FNO-generated mesh, and fixed quasi-uniform FEM, respectively. The top row shows the numerical solutions, and the bottom row shows the corresponding pointwise absolute errors with respect to the reference solution.



Standard AFEM Target FNO

Figure 12: Tetrahedral meshes for reaction–diffusion Test 065 constructed by the standard AFEM, target refinement field, and FNO-predicted refinement field. Each panel shows the $z = 0$ boundary plane and the $y = 0.5$ interior cross-section in a 3-D view.



$z = 0$ Standard AFEM $z = 0$ FNO $y = 0.5$ Standard AFEM $y = 0.5$ FNO

Absolute error Absolute error Absolute error Absolute error

Figure 13: Finite-element solutions and pointwise absolute errors for reaction–diffusion Test 065. The first two columns show the $z = 0$ boundary slice for standard AFEM and FNO-predicted refinement field, while the last two columns show the corresponding $y = 0.5$ interior slice. The top row shows the numerical solutions and the bottom row shows the absolute errors with respect to the reference solution.