PACE: A PHYSICS-AWARE MESH COARSENING FRAMEWORK BASED ON ERROR EQUIDISTRIBUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Accurate mesh-based simulation relies on sufficient mesh resolution. However, increasing the global mesh resolution directly raises computational cost, making the balance between physical fidelity and efficiency a fundamental bottleneck in numerical simulation. While careful artificial mesh layout or adaptive mesh coarsening can mitigate this burden by eliminating redundant degrees of freedom, existing strategies often rely on costly iterative optimization. We introduce PACE, a physics-aware mesh coarsening framework based on error equidistribution. PACE designs a graph neural network for efficient edge-collapse selection that couples a coarse physical field with mesh geometric features to predict an edge-wise coarsening score map. PACE then realizes the predicted score map through topology-constrained parallel edge collapse to produce the final coarse mesh. Without assuming a uniquely defined target coarse mesh, PACE formulates error equidistribution as a self-supervised objective for learning how limited mesh resolution should be allocated according to local physical importance. We evaluate on multiple cases covering steady and transient regimes, and observe consistent gains in accuracy and efficiency compared to traditional mesh coarsening baselines.

1 INTRODUCTION

Solving partial differential equations (PDEs) is fundamental for modeling physical phenomena, spanning fluid dynamics, climate science, materials engineering, and beyond Garabedian (2023). Modern PDE solving critically relies on meshes, which serve as the foundational discretization framework for numerical methods Zienkiewicz et al. (2013). In industrial manufacturing, accurately solving these PDEs often involves complex geometries and requires dense mesh discretizations to capture localized physical effects, which significantly increases computational cost Solanki et al. (2003). As a consequence, high-fidelity industrial simulations can take between several days and weeks, limiting their usability in time-critical engineering workflows.

To alleviate this issue, mesh adaptation methods concentrate computational resources (mesh points, polynomial degrees) in regions where a posteriori error estimates indicate insufficient resolution. Among them, the mesh coarsening method adjusts resolution via local element coarsening, altering node counts while retaining fixed positions. These methods follow the principle of error equidistribution, which dictates that optimal allocation equalizes per-element error contributions across the domain Verfürth (2013). However, traditional adaptive coarsening methods are typically reactive: reliable coarsening metrics are obtained only after solving the PDE on the current dense mesh, leading to repeated solve-estimate-adapt cycles and considerable overhead.

Recent advances in learning-based mesh processing provide a promising alternative to traditional adaptive mesh coarsening Teixeira et al. (2026); Hanocka et al. (2019); Cao et al. (2023). Graph neural networks (GNN) can operate directly on unstructured meshes, and recent approaches learn policies for adaptive mesh refinement Freymuth et al. (2026); Jenkins (2026), coarsening Bazzi et al. (2026); Liu et al. (2025b), and relocation Zhang et al. (2024); Wang et al. (2025b). These studies demonstrate that mesh adaptation decisions can be learned from data and transferred across problem instances, thereby amortizing the repeated cost of determining where additional computational resolution is required.

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.

However, some learning-based methods for mesh adaptation are often tightly coupled with learned surrogate simulators, where mesh adaptation is optimized jointly with the prediction model Jenkins (2026); Bazzi et al. (2026). In contrast, methods that directly reduce the degrees of freedom through explicit coarsening operators remain relatively underexplored. Existing learning-based coarsening methods rely on supervised training with expert labels, which can increase dependence on the training distribution and potentially limit generalization to unseen PDEs and geometries Liu et al. (2025b); Wang et al. (2025a).

In this paper, we propose PACE, a physics-aware mesh coarsening framework based on error equidistribution. PACE formulates error equidistribution as a self-supervised objective to learn adaptive mesh coarsening from physical observations. PACE operates in two stages at inference time: (1) an initial forward pass through a neural network produces an edge-wise coarsening score map; (2) iterative coarsening steps are allocated nonuniformly across the domain, reducing the computational resources needed in the low-error region.

The main contributions of this work are:

- We propose a practical architecture for adaptive mesh coarsening, consisting of a physics-aware GNN model and a topology-constrained parallel edge-collapse strategy.
- We design a loss function based on error equidistribution, which guides the model to learn an edge-wise coarsening score map that reflects the local numerical importance of the mesh.
- We demonstrate that PACE achieves higher numerical accuracy while accelerating mesh coarsening by an average of 31.5$\times$ over traditional mesh coarsening methods. On complex PDE cases, PACE further reduces the total runtime by up to 18.8$\times$ and generalizes across initial conditions, geometries, and mesh resolutions.

2 Related work

Mesh neural networks. Recent studies have developed neural architectures specifically for irregular mesh structures Lahav & Tal (2020); Lei et al. (2021); Milano et al. (2020). MeshCNN Hanocka et al. (2019) introduces a mesh-specific pooling operation based on edge collapse, improving the ability of convolutional neural networks (CNNs) to extract features from irregular mesh domains. MeshGraphNet Pfaff et al. (2021) represents computational meshes as graphs and learns physical dynamics through message passing over mesh connectivity. BS-MSGNN Cao et al. (2023) further improves the representation of multi-scale mesh features through a bi-stride graph neural network. These methods primarily improve mesh representation and feature extraction for specific downstream tasks, rather than directly constructing adaptively coarsened meshes for conventional numerical solvers.

Edge collapse methods. Edge collapse is a classical local operation for explicit mesh coarsening, in which candidate edges are typically ranked according to a predefined collapse cost Kulkarni & Narayanan (2025). Quadric error metrics (QEM) Garland & Heckbert (1997) is one of the most widely used cost functions. It evaluates the geometric distortion introduced by vertex pair contraction through a compact quadratic form, enabling efficient greedy mesh coarsening. Subsequent extensions Hoppe (1999) incorporate additional attributes, including color, texture, and surface normals, into the error metric. Following this paradigm, many studies Liu et al. (2025a); Knodt (2025); Bhosikar et al. (2026) retain cost-based edge ranking while redesigning the collapse criterion for different structural constraints and application scenarios. However, these methods determine edge priority primarily from geometric or related mesh features and do not directly account for the contribution of individual edges to the accuracy of the underlying PDE solution. In this work, we extend the cost-guided edge collapse paradigm to PDE solving and develop a parallel edge collapse strategy to accelerate mesh coarsening.

Learning-based mesh coarsening. With the development of deep learning, learning-based methods further expand the design space of mesh coarsening Teixeira et al. (2026); Ryabov et al. (2026). Some methods Potamias et al. (2022) formulate mesh reduction as a fully learnable one-pass process, thereby avoiding the sequential execution of conventional edge collapse algorithms. However, direct prediction of mesh connectivity can introduce holes or invalid elements, which limits applicability to numerical simulation where mesh validity is essential. Other methods Jenkins (2026); Bazzi et al. (2026) jointly optimize mesh coarsening and neural surrogate models to improve simulation

efficiency. More recent studies treat mesh coarsening as an independent learning task. HMSimNet Wang et al. (2025a) and TetSimNet Liu et al. (2025b) construct edge collapse probability labels and learn coarsening decisions for volume meshes with the objective of preserving numerical analysis accuracy. In contrast, we introduce a self-supervised graph neural network framework that learns edge numerical importance without manually constructed labels and uses the predicted scores to guide explicit mesh coarsening.

3 METHOD

3.1 PROBLEM FORMULATION

Given a dense mesh $M = (V_M, E_M)$ and a coarse physical field $\mathbf{p}^c$, we construct an edge-based graph $G = (V_G, E_G)$, where each graph node $v_g \in V_G$ corresponds to an edge e of the original mesh. A connection $e_g \in E_G$ is established between two graph nodes when their corresponding mesh edges share an endpoint. Our goal is to learn a physics-aware allocation of mesh resolution from coarse physical observations without assuming a uniquely defined optimal coarse mesh.

We characterize the local coarsening strategy by an edge-wise coarsening score map

$$S = \{s_e\}_{e \in E_M}, \tag{1}$$

where s_e controls the allowable coarsening scale around edge e. For each edge, let $\bar{l}_e$ denote the local reference length computed from its one-ring neighborhood. The corresponding target edge length is defined as

$$\hat{l}_e = s_e \bar{l}_e, \tag{2}$$

where smaller s_e preserves denser local resolution, while larger s_e permits more aggressive coarsening.

The goal is to find a score map S^* that provides a physics-aware distribution of mesh resolution while enabling effective mesh reduction:

$$\theta^* = \arg\min_{\theta} L(\phi_\theta(G)), \qquad S^* = \phi_{\theta^*}(G), \qquad \text{s.t.} \quad M_c \in M_{valid}, \tag{3}$$

where θ represents the parameters of the model, L denotes the physics-aware coarsening objective introduced in Sec.3.4, M_c is the resulting coarse mesh, and M_{valid} denotes the set of meshes satisfying the required validity constraints.

3.2 ARCHITECTURE OVERVIEW

PACE operates in two stages at inference time, shown in Fig.1:

Stage 1: Edge-wise coarsening score map prediction. A learnable GNN predictor ϕ takes as input an edge-based graph G constructed from the dense mesh and its coarse physical observations, and outputs the score map

$$S = \phi(G) = \{s_e\}_{e \in E_M}. \tag{4}$$

The predicted score map estimates the local coarsening tendency of each mesh edge and is converted into the spatially varying target size field defined in Eq.2.

Stage 2: Topology-constrained parallel edge collapse. The target size field is used to guide a parallel edge-collapse operator C:

$$M_c = C(M, S). \tag{5}$$

At each iteration, candidate edges are selected according to the predicted size field and filtered by topology constraints as shown in Fig.1. The detailed implementation of these topology constraints is provided in Appendix C. Mutually independent candidates are then collapsed in parallel, followed by dynamic topology updates. The process is repeated until no further valid collapses remain or the target coarsening ratio is reached, producing the final coarsened mesh M_c.

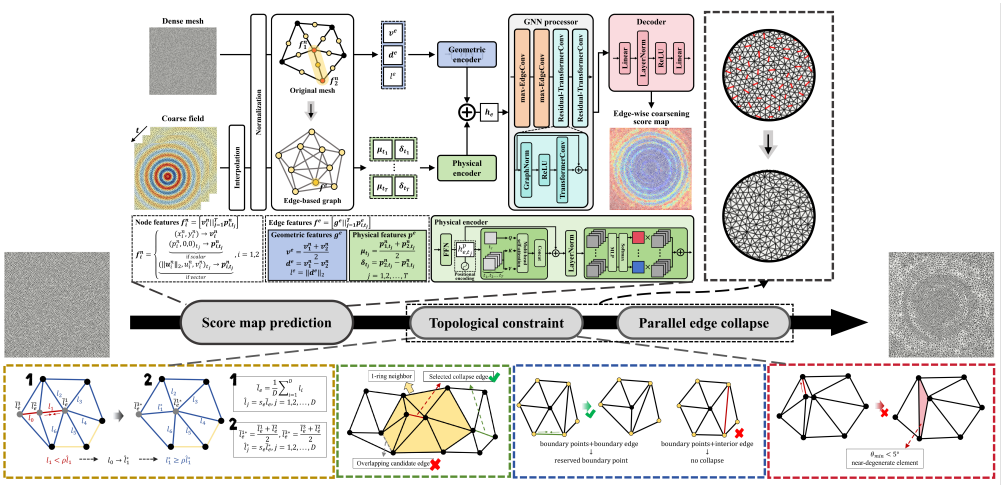


Figure 1: Overview of the PACE framework. Stage 1 produces an edge-wise coarsening score map prediction via a GNN model. Stage 2 allocates coarsening steps nonuniformly based on the score map and applies topology-constrained parallel edge collapse. Among them, from left to right, the topological constraints below are dynamic topology update constraint, topology preservation constraint, boundary constraint, and mesh quality constraints.

3.3 Model Architecture

We propose an edge-based graph neural network with an encoder-processor-decoder architecture. Given a dense mesh and its corresponding coarse physical fields, the model first converts the mesh and its associated features into an edge-based graph and then predicts an edge-wise coarsening score map. The architecture of the proposed edge-based graph network is shown at the top of Fig.1.

Data preprocessing. Let $\mathbf{p}^c$ denote the input coarse physical field. For the coarse field $\mathbf{p}^c_{t_j}$ at each time step, the physical field is mapped from the coarse mesh to the dense mesh by barycentric interpolation to obtain $\mathbf{p}^n_{t_j}$. Each node of the dense mesh is then associated with a spatial coordinate $\mathbf{v}^n$ and a physical quantity $\mathbf{p}^n$. The normalized node features $\mathbf{f}^n$ are subsequently converted into the node features $\mathbf{f}^e$ of the edge-based graph. As shown in the lower-left part of Fig.1, $\mathbf{f}^e$ consists of geometric features $\mathbf{g}^e$ and physical features $\mathbf{p}^e$.

Encoder. The encoder models the geometric and physical features separately. The geometric encoder employs a feed-forward network (FFN) to map the original geometric features to a hidden space:

$$\mathbf{h}_g = \phi_{\text{FFN}}(\mathbf{g}_e) \tag{6}$$

Because the physical features exhibit strong temporal correlations, the physical encoder uses temporal self-attention to capture dependencies among valid temporal states. This module contains only a single multi-head attention layer. We first describe the computation of a single attention head. At time t_i, the physical feature $\mathbf{p}^e_{t_i}$ is used as the query vector to compute the attention weight of the k-th attention head over all valid temporal states $(\mathbf{p}^e_{t_1}, \mathbf{p}^e_{t_2}, \ldots, \mathbf{p}^e_{t_T})$:

$$\mathbf{a}^{(k)}_{t_i} = \text{softmax}\left(\frac{\left(\mathbf{W}^{(k)}_1 \mathbf{p}^e_{t_i}\right)^\top \mathbf{W}^{(k)}_2 \left[\mathbf{p}^e_{t_1}, \mathbf{p}^e_{t_2}, \ldots, \mathbf{p}^e_{t_T}\right]}{\sqrt{d'}} \right). \tag{7}$$

Here, $\mathbf{W}^{(k)}_1$ and $\mathbf{W}^{(k)}_2$ are learnable parameters, and d' denotes the feature dimension after mapping $\mathbf{p}^e_{t_i}$. The encoded vector of a single attention head is then computed using the attention weight:

$$\mathbf{h}^{(k)}_{t_i} = \mathbf{W}^{(k)}_3 \left[\mathbf{p}^e_{t_1}, \mathbf{p}^e_{t_2}, \ldots, \mathbf{p}^e_{t_T}\right] \mathbf{a}^{(k)}_{t_i}, \tag{8}$$

where $\mathbf{W}^{(k)}_3$ is a learnable parameter. Multi-head attention constructs K attention heads with independent parameters in parallel, producing K output vectors $(\mathbf{h}^{(1)}_{t_i}, \ldots, \mathbf{h}^{(K)}_{t_i})$. The outputs of all

attention heads are concatenated and compressed, and the resulting temporal representations are further aggregated into a physical embedding:

$$\mathbf{H}_{t_i} = \left[\mathbf{h}_{t_i}^{(1)} \| \cdots \| \mathbf{h}_{t_i}^{(K)}\right]. \tag{9}$$

$$\mathbf{h}_p = \sum_{j=1}^{T} \alpha_{t_j} \mathbf{H}_{t_j}. \tag{10}$$

The temporal weight α_{t_j} is defined as

$$\alpha_{t_j} = \frac{m_{t_j} \exp\left(\mathrm{MLP}\left(\mathbf{H}_{t_j}\right)\right)}{\sum_{q=1}^{T} m_{t_q} \exp\left(\mathrm{MLP}\left(\mathbf{H}_{t_q}\right)\right)}. \tag{11}$$

Here, m_{t_j} denotes the validity mask associated with the temporal state t_j and MLP denotes multi-layer perceptron. The geometric embedding $\mathbf{h}_g$ and physical embedding $\mathbf{h}_p$ are concatenated and projected to initialize the processor:

$$\mathbf{h}_e^{(0)} = \phi_{proj}\left(\mathbf{h}_g \| \mathbf{h}_p\right). \tag{12}$$

Processor. The processor contains two max-EdgeConv layers and two Residual-TransformerConv layers. This module further refines the graph-node representations through neighborhood feature aggregation. max-EdgeConv explicitly compares each central graph node with its neighboring nodes and retains the strongest neighborhood response:

$$\mathbf{h}_e^{(l+1)} = \max_{f \in N(e)} \phi_{edge}^{(l)} \left[\mathbf{h}_e^{(l)} \| \mathbf{h}_f^{(l)} - \mathbf{h}_e^{(l)}\right]. \tag{13}$$

Here, $\mathbf{h}_e^{(l)}$ and $\mathbf{h}_f^{(l)}$ denote the hidden features of the central edge e and its neighboring edge f, respectively, $N(e)$ denotes the neighborhood of e, and $\phi_{edge}^{(l)}$ denotes the learnable transformation of the l-th max-EdgeConv layer. The residual-TransformerConv then assigns different weights to messages according to neighborhood features and relative positions, allowing the model to focus on physical structures with consistent spatial organization:

$$\mathbf{h}_{trans}^{(l)} = \mathbf{W}_{skip}^{(l)} \overline{\mathbf{h}}_e^{(l)} + \sum_{f \in N(e)} \beta_{fe}^{(l)} \mathbf{h}_{fe}^{(l)}. \tag{14}$$

$$\mathbf{h}_e^{(l+1)} = \mathbf{h}_e^{(l)} + \mathbf{h}_{trans}^{(l)}. \tag{15}$$

Here, $\overline{\mathbf{h}}_e^{(l)}$ denotes the central-node feature after GraphNorm normalization, $\mathbf{W}_{skip}^{(l)}$ is the learnable linear transformation applied to the central-node feature at layer l, $\mathbf{h}_{trans}^{(l)}$ denotes the neighborhood update produced by TransformerConv at layer l, $\beta_{fe}^{(l)}$ denotes the attention weight associated with the message passed from neighboring node f to central node e, and $\mathbf{h}_{fe}^{(l)}$ denotes the message value vector that combines the neighboring-node feature with its relative positional information with respect to the central node.

Decoder. The decoder employs a node-wise multilayer perceptron followed by a sigmoid function to transform the final hidden representation into the edge-wise coarsening score map S:

$$s_e = \phi_{dec}\left(\mathbf{h}_e^{(L)}\right), \quad S = \{s_e\}_{e \in E_M}. \tag{16}$$

3.4 Loss function

The absence of a uniquely defined optimal coarse mesh makes direct supervision with a single reference discretization inherently ambiguous. Accordingly, we follow the error-equidistribution principle to learn a principled allocation strategy for a finite mesh-resolution budget.

Error-equidistribution loss L_{eq}. Let $\mathbf{p}_{t_j}^n$ denote the physical state obtained from the numerical solver at time t_j. For an edge $e = (\mathbf{v}_1^n, \mathbf{v}_2^n)$, its first-order physical variation at time t_j is defined as

$$\nabla \mathbf{p}_{t_j}^e = \frac{\left\| \mathbf{p}_{1,t_j}^n - \mathbf{p}_{2,t_j}^n \right\|_2}{\left\| \mathbf{v}_1^n - \mathbf{v}_2^n \right\|_2}. \tag{17}$$

To prevent a small number of nearly stationary frames from obscuring critical physical states, we average the three largest responses over all valid time frames:

$$\nabla \mathbf{p}^e = \mathrm{TopKMean}_{t_j,\, K=3} \left(\nabla \mathbf{p}^e_{t_j} \right). \tag{18}$$

To further capture local extrema and oscillatory structures, we use a node-wise graph Laplacian to characterize local second-order variations:

$$\nabla^2 \mathbf{p}^n_{i,t_j} = \mathbf{p}^n_{i,t_j} - \frac{1}{|N(i)|} \sum_{q \in N(i)} \mathbf{p}^n_{q,t_j}. \tag{19}$$

The second-order variation associated with edge e is then defined as

$$\nabla^2 \mathbf{p}^e = \max_{t_j} \left(\left\| \nabla^2 \mathbf{p}^n_{1,t_j} \right\|_2^2 + \left\| \nabla^2 \mathbf{p}^n_{2,t_j} \right\|_2^2 \right). \tag{20}$$

We then normalize $\nabla \mathbf{p}^e$ and $\nabla^2 \mathbf{p}^e$ in each graph to get $\nabla \hat{\mathbf{p}}^e$ and $\nabla^2 \hat{\mathbf{p}}^e$. The physical importance of edge e is defined as

$$\eta_e = \rho \nabla \hat{\mathbf{p}}^e + (1 - \rho) \nabla^2 \hat{\mathbf{p}}^e, \tag{21}$$

where ρ is the weight of $\nabla \hat{\mathbf{p}}^e$. Finally, the error-equidistribution loss is formulated as

$$L_{eq} = \mathrm{Var}_{e \in E_{int}} \left[\log \left(\eta_e s_e^{\gamma} \right) \right]. \tag{22}$$

Here, E_{int} denotes the set of mesh edges involved in the loss computation, s_e is the edge-wise coarsening score predicted by the model, and γ controls the influence of s_e on the relative error contribution.

Teacher loss $L_{teacher}$. To further enhance the capability of the model to capture important physical variations, we use the high-fidelity solution during training to provide additional physical guidance. This encourages the network to learn a more accurate resolution allocation strategy under complex physical conditions. This loss has the same form as the error-equidistribution loss, except that the physical importance is computed using the high-fidelity solution.

Coarsening encouragement loss L_{coarse}. When only the error-equidistribution loss is optimized, uniformly small coarsening scores can still produce a relatively balanced error distribution while yielding insufficient mesh reduction. To avoid this degenerate solution, we introduce a coarsening encouragement loss that promotes larger coarsening scores in physically smooth regions:

$$L_{coarse} = -\frac{1}{|E_{int}|} \sum_{e \in E_{int}} \log(s_e). \tag{23}$$

The overall training objective is

$$L = L_{eq} + \lambda_f L_{teacher} + \lambda_c L_{coarse}, \tag{24}$$

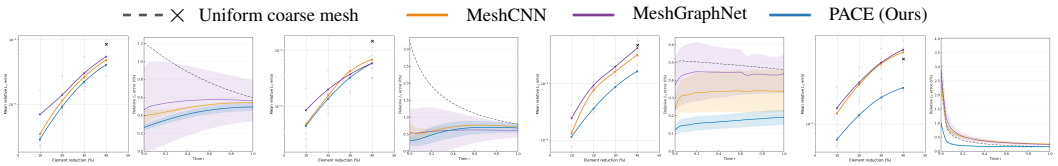
where λ_f and λ_c weight the teacher and coarsening encouragement terms, respectively.

4 Experiments

4.1 Experiment setup

Training. The training set contains 60 transient physical samples generated on different mesh discretizations of the same computational domain. These samples are split into 36 training samples, 12 validation samples, and 12 evaluation samples. All models are trained for at most 200 epochs using the Adam optimizer. The loss-related hyperparameters are set to $\rho = 0.2$, $\lambda_f = 0.35$, and $\lambda_c = 0.02$ for all experiments, with their selection discussed in Sec.4.3. All baseline models follow the same overall training protocol, while model-specific hyperparameters are tuned on the validation set. All experiments are conducted on an RTX 4090 GPU. To reduce result variance and ensure fair comparisons, each experimental configuration is repeated with 3 different random seeds, and the mean performance is reported.

Datasets. We evaluate the proposed method on three sets of fluid mechanics data: (1) meshes on regular computational domains, (2) meshes on polygonal computational domains, and (3) meshes



(a) Pressure Diff. (1) (b) Pressure Diff. (2) (c) Vector React.-Diff. (1) (d) Vector React.-Diff. (2)

Figure 2: Comparison of numerical error on four test cases. For each case, the left panel reports the mean relative L_2 error under different coarsening ratios, while the right panel shows the temporal evolution of the relative L_2 error at a 40% coarsening ratio. Shaded regions indicate variation across models trained with different random seeds.

Table 1: Comparison of numerical error and computational efficiency at a 40% coarsening ratio. Results are reported as mean ± standard deviation over three random seeds.

Equation	Method	Rollout RMSE ↓ ($\times 10^{-4}$)	Mean Rel. L_2 ↓ ($\times 10^{-4}$)	Max Rel. L_2 ↓ ($\times 10^{-3}$)	Solving speedup ↑
Pressure Diff. (1)	MeshCNN	12.20 ± 0.44	48.60 ± 1.64	5.40 ± 0.09	2.98
	MeshGraphNet	13.99 ± 9.29	54.93 ± 35.86	6.48 ± 3.73	**3.03**
	PACE (Ours)	$\mathbf{10.35 \pm 1.05}$	$\mathbf{41.23 \pm 4.22}$	$\mathbf{4.92 \pm 0.44}$	3.01
Pressure Diff. (2)	MeshCNN	12.96 ± 1.12	68.52 ± 5.78	7.74 ± 0.18	3.02
	MeshGraphNet	11.78 ± 9.61	59.74 ± 47.35	7.88 ± 5.26	3.08
	PACE (Ours)	$\mathbf{11.12 \pm 3.58}$	$\mathbf{59.07 \pm 19.11}$	$\mathbf{7.14 \pm 1.84}$	**3.13**
Vector React.-Diff. (1)	MeshCNN	7.01 ± 2.41	32.75 ± 11.11	3.42 ± 1.18	1.64
	MeshGraphNet	9.37 ± 2.95	43.36 ± 13.36	4.66 ± 1.49	**1.65**
	PACE (Ours)	$\mathbf{3.68 \pm 0.86}$	$\mathbf{16.69 \pm 4.03}$	$\mathbf{1.90 \pm 0.42}$	1.64
Vector React.-Diff. (2)	MeshCNN	45.73 ± 6.97	52.23 ± 7.02	25.41 ± 5.09	1.66
	MeshGraphNet	51.38 ± 15.13	55.43 ± 13.51	29.38 ± 9.00	**1.68**
	PACE (Ours)	$\mathbf{19.39 \pm 1.70}$	$\mathbf{23.06 \pm 2.48}$	$\mathbf{9.98 \pm 0.67}$	1.65

from real-world fluid dynamics problems. The first two sets contain both steady-state and transient physical problems, while the third set consists of the CylinderFlow and Airfoil datasets. Together, these three sets are used to evaluate the effectiveness and generalization ability of the proposed model. See Appendix D for more dataset details.

Baselines. We compare the proposed method with mesh coarsening models that use MeshCNN Hanocka et al. (2019) and MeshGraphNet Pfaff et al. (2021) as their respective backbones. For steady-state physical problems, we additionally compare against classical geometric-aware QEM Hoppe (1999) and two physics-aware QEM variants built upon it, namely gradient-error QEM and error equidistribution QEM. Details of each baseline is provided in the Appendix E.2.

Metrics. We evaluate mesh coarsening from three complementary aspects. The coarsening ratio measures the extent of mesh reduction relative to the original dense mesh. Numerical accuracy is evaluated using the relative L_2 error and rollout root mean square error (RMSE), which characterize solution error at individual time steps and over the complete trajectory, respectively. Computational efficiency is assessed using the runtime of different stages and the corresponding speedups. Detailed mathematical definitions of all metrics are provided in Appendix E.3.

4.2 Experiment Results

Effect of coarsening ratio. Fig.2 evaluates the effect of different coarsening ratios on numerical accuracy. As the coarsening ratio increases, the numerical error grows for all methods because fewer elements are available to represent the solution field. PACE exhibits a slower increase in error across all test cases. At a 40% coarsening ratio, PACE consistently achieves lower errors than the uniform coarse mesh and the evaluated baselines.

Accuracy-efficiency trade-off. Fig.2 further shows the temporal evolution of numerical errors after coarsening. PACE produces more stable temporal error profiles with lower error accumulation over time. Table 1 further supports this observation. At a 40% coarsening ratio, PACE achieves the lowest rollout RMSE across all four transient cases. PACE also achieves the largest solving speedup in two transient cases, reaching up to $3.13\times$ on Pressure Diff. (2). These results demonstrate that PACE maintains high numerical accuracy under substantial mesh reduction while improving computational efficiency.

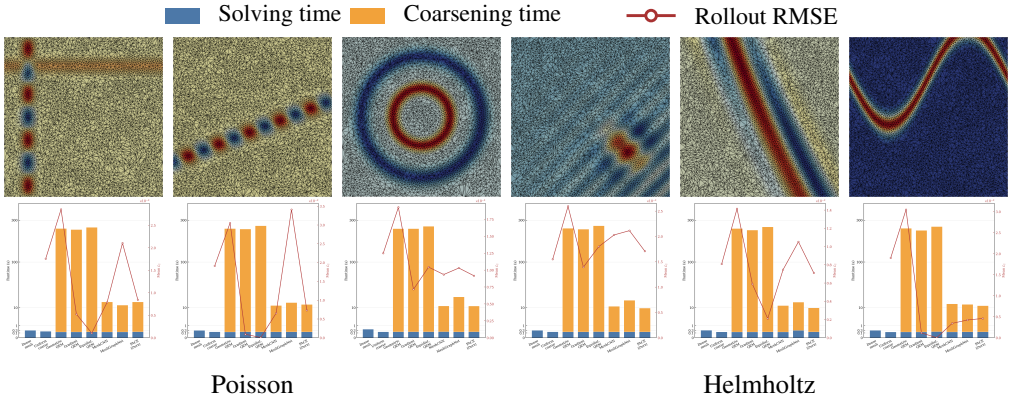


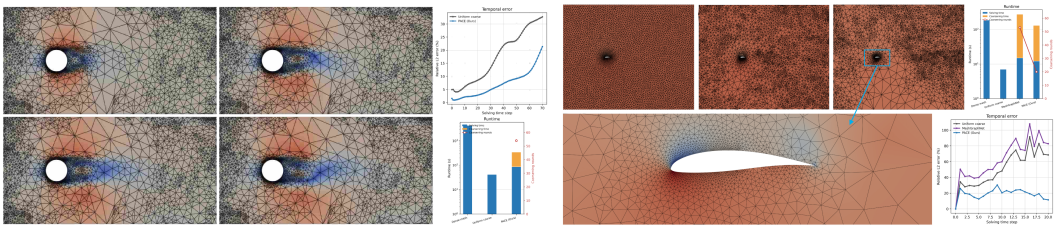
Figure 3: Qualitative and quantitative comparison of the Poisson and Helmholtz equations.

Computational efficiency under different mesh scales and temporal horizons. Table 2 shows that the computational advantage of PACE becomes more pronounced as the mesh size increases. When reducing a 100k-level mesh to approximately 1k elements, PACE achieves a $50.60\times$ solving speedup and an $18.75\times$ total speedup. The improvement is smaller for 10k-level meshes, indicating that mesh coarsening provides greater computational benefits at larger scales where more redundant degrees of freedom can be removed. The steady-state experiments in Fig.3 further evaluate the efficiency of the coarsening procedure itself. PACE requires substantially less coarsening time than conventional QEM-based methods, with an average speedup of $31.5\times$. For long-horizon simulations, the computational benefit of mesh reduction accumulates over repeated solution steps. As shown in Fig.4, PACE achieves greater overall acceleration on large-scale computational fluid dynamics (CFD) cases. The efficiency improvement of PACE arises from two complementary components. First, the GNN predicts edge importance in a single forward pass, avoiding the repeated online evaluation and ranking of candidate edges required by conventional collapse-based methods. Second, the parallel edge-collapse strategy reduces the overhead associated with topology updates during mesh coarsening.

Table 2: Computational performance of PACE across different mesh scales.

Original scale	Coarse scale	Coarsening time (s)	Dense solve time (s)	Coarse solve time (s)	Solving speedup	Total speedup
10k-level	1k-level	11.56	27.68	14.26	$1.94\times$	$1.07\times$
10k-level	10k-level	18.15	55.51	28.63	$1.94\times$	$1.19\times$
100k-level	10k-level	102.83	3,250.28	147.52	$22.03\times$	$12.98\times$
100k-level	1k-level	142.40	4,241.92	83.84	$50.60\times$	$18.75\times$

Generalization across PDEs and geometries. Although PACE is trained on a limited number of transient PDE cases, it generalizes across different PDEs and computational domains. Fig.3 and Appendix F.1 show that PACE consistently preserves higher mesh resolution in regions with strong physical variations while removing more elements from smoother regions. Its generalization capability is further demonstrated on unseen geometries. Fig.8 in Appendix F.1 shows that PACE adapts to irregular computational domains while maintaining appropriate resolution around important physical structures. In the 100k-scale CFD experiments shown in Fig.4, including cylinder flow and airfoil simulations, PACE retains higher mesh resolution near boundaries, wakes, and regions



(a) Cylinder flow velocity field (b) FX63137 airfoil pressure field

Figure 4: 100k-scale experiments on cylinder flow and the FX63137 airfoil. Left compares PACE with uniform coarsening, as MeshGraphNet misses the target ratio. Right shows, from left to right, uniform, MeshGraphNet, and PACE coarse meshes.

with strong flow variations while removing redundant elements from smooth regions. Additional airfoil experiments in Fig.9 of Appendix F.1 further demonstrate consistent coarsening behavior across different airfoil geometries and angles of attack.

4.3 Ablation Study

Fig.5 evaluates the contributions of individual components and the sensitivity to key hyperparameters. Removing either the physical input or the teacher loss increases the relative L_2 error, particularly at higher coarsening ratios, highlighting the importance of physical input and training objectives. The sensitivity analysis shows that $\rho = 0.2$ yields the lowest rollout RMSE, whereas larger values lead to substantial error fluctuations due to excessive emphasis on the first-order physical variation. For the teacher loss weight, $\lambda_f = 0.35$ and $\lambda_f = 0.75$ achieve comparable numerical accuracy and coarsening ratios. We choose $\lambda_f = 0.35$ to reduce the dependence on the high-fidelity solution while maintaining prediction accuracy. For the coarsening weight, $\lambda_c = 0$ achieves similar accuracy but provides no explicit guidance for promoting mesh reduction. A small positive value, $\lambda_c = 0.02$, improves the coarsening behavior without degrading numerical accuracy, whereas larger values occasionally result in abnormally high errors. Based on these observations, we use $\rho = 0.2$, $\lambda_f = 0.35$, and $\lambda_c = 0.02$ in all experiments.

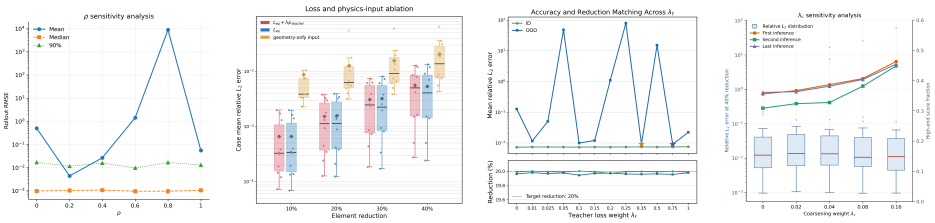


Figure 5: Ablation studies. (a) Sensitivity to the first-order physical variation weight ρ. (b) Effects of $L_{teacher}$ and physics input under different coarsening ratios. (c) Sensitivity to the teacher weight λ_f. In-distribution (ID) denotes held-out cases from the training distribution, while out-of-distribution (OOD) represents cases from the same PDE with unseen geometries or initial conditions. (d) Sensitivity to the coarsening encouragement weight λ_c, showing numerical error and inference behavior.

5 Conclusion

In this paper, we present PACE, a physics-aware mesh coarsening framework based on error equidistribution. A lightweight GNN model identifies regions of low prediction error, and parallel coarsening is concentrated there, achieving 18.8× total runtime speedup on large-scale PDE simulations. The framework is grounded in classical mesh adaptation theory via an error equidistribution principle, guiding the model to learn physics-aware resolution allocation. Experimental results show that it captures the relationship between physical fields and local mesh resolution requirements, maintains low numerical error across different coarsening ratios, and generalizes effectively to unseen physical conditions and complex geometries.

AI USE STATEMENT

In this work, we used generative AI tools for [no tasks with required disclosure]. We have not used generative AI tools for [other tasks with required disclosure], and [the rest of the required disclosure tasks] are not applicable to this work. Additionally, we used generative AI tools for [draft parts of a research paper, summarize or analyse existing literature, edit a research paper to improve readability, identify relevant literature]. We have reviewed all AI-assisted work. [We checked LLM-generated research ideas for potential plagiarism through a manual literature survey. LLM-generated text was verified and tested for correctness by 2 authors.]. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Amir Bazzi, Ramy Nemer, Alves José, and Elie Hachem. Physics-informed coarsening for multigrid graph neural surrogates. In *Forty-third International Conference on Machine Learning*, 2026.

Kunal Bhosikar, Preet Savalia, Lokender Tiwari, and Brojeshwar Bhowmick. Fast and robust mesh simplification for generated and real-world 3d assets. In *CVPR 2026 Workshop on 3D Geometry Generation for Scientific Computing*, 2026.

Yadi Cao, Menglei Chai, Minchen Li, and Chenfanfu Jiang. Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network. In *International conference on machine learning*, pp. 3541–3558, 2023.

Niklas Freymuth, Philipp Dahlinger, Tobias Würth, Simon Reisch, Luise Kärger, and Gerhard Neumann. Adaptive swarm mesh refinement using deep reinforcement learning with local rewards. *Machine Learning*, 115:169, 2026.

Paul R Garabedian. *Partial differential equations*, volume 325. American Mathematical Society, 2023.

Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, 1997.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: A network with an edge. *ACM Transactions on Graphics (TOG)*, 38:1–12, 2019.

Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg. Graph laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 8:1325–1368, 2007.

Hugues Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *Proceedings Visualization '99*, 1999.

Weizhang Huang. Mathematical principles of anisotropic mesh adaptation. *Communications in Computational Physics*, 1:276–310, 2006.

Ben Jenkins. Adaptive test-time compute allocation for neural PDE solvers. In *AI&PDE: ICLR 2026 Workshop on AI and Partial Differential Equations*, 2026.

Julian Knodt. Single edge collapse quad-dominant mesh reduction. *ACM Transactions on Graphics (TOG)*, 44:1–23, 2025.

Purva Kulkarni and Aravind Shankara Narayanan. A comprehensive guide to mesh simplification using edge collapse. *arXiv preprint arXiv:2512.19959*, 2025.

Alon Lahav and Ayellet Tal. Meshwalker: Deep mesh understanding by random walks. *ACM Transactions on Graphics (TOG)*, 39:1–13, 2020.

Huan Lei, Naveed Akhtar, and Ajmal Mian. Picasso: A cuda-based library for deep learning over 3d meshes. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13849–13859, 2021.

Hsueh-Ti Derek Liu, Xiaoting Zhang, and Cem Yuksel. Simplifying textured triangle meshes in the wild. *ACM Transactions on Graphics (TOG)*, 44:1–16, 2025a.

Songyuan Liu, Rui Wang, Hongfei Zhan, and Junhe Yu. Tetsimnet: a tetrahedral mesh simplification network model for preserving analysis accuracy. *Engineering with Computers*, 41:4345–4363, 2025b.

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*. Springer, 2003.

Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. Primal-dual mesh convolutional neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

Ricardo H Nochetto and Andreas Veeser. Primer of adaptive finite element methods. In *Multiscale and Adaptivity: Modeling, Numerics and Applications: CIME Summer School, Cetraro, Italy 2009, Editors: Giovanni Naldi, Giovanni Russo*. Springer, 2011.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

Rolandos Alexandros Potamias, Stylianos Ploumpis, and Stefanos Zafeiriou. Neural mesh simplification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Alexander Ryabov, Viacheslav Naumov, Sergei Shumilin, Nikolay Yavich, Sayan Ranu, N. M. Anoop Krishnan, Evgeny Burnaev, and Vladimir Vanovskiy. Learnable stability-aware computational grid coarsening for accelerating physics simulations. *Machine Learning: Science and Technology*, 7:015019, 2026.

Kiran Solanki, SR Daniewicz, and JC Newman Jr. Finite element modeling of plasticity-induced crack closure with emphasis on geometry and mesh refinement effects. *Engineering Fracture Mechanics*, 70:1475–1489, 2003.

L. Teixeira, V. Teixeira, P. Alves, L. Conceição, G. Marreiros, and D. Martinho. A survey of learning-based mesh processing techniques for mesh simplification. In *Recent Trends and Challenges in Information Systems and Technologies*. Springer Nature Switzerland, 2026.

Rüdiger Verfürth. *A posteriori error estimation techniques for finite element methods*. OUP Oxford, 2013.

Rui Wang, Songyuan Liu, Junhe Yu, and Hongfei Zhan. Hmsimnet: A hexahedral mesh simplification network model for preserving analysis accuracy. *Advances in Engineering Software*, 202: 103874, 2025a.

Zhichao Wang, Xinhai Chen, Qinglin Wang, Xiang Gao, Qingyang Zhang, Menghan Jia, Xiang Zhang, and Jie Liu. UGM2n: An unsupervised and generalizable mesh movement network via m-uniform loss. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Mingrui Zhang, Chunyang Wang, Stephan C. Kramer, Joseph Gregory Wallwork, Siyi Li, Jiancheng Liu, Xiang Chen, and Matthew D Piggott. Towards universal mesh movement networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, and Jian Z Zhu. *The finite element method: its basis and fundamentals*. Elsevier, 2013.

APPENDIX

A ERROR-EQUIDISTRIBUTION PRINCIPLE

Error equidistribution allocates mesh resolution according to the local difficulty of approximating the physical solution Huang (2006). Let Ω_c and Ω_p denote the computational and physical domains.

A symmetric positive-definite monitor matrix $M(x)$ specifies the desired mesh density at $x \in \Omega_p$. Define

$$\varrho(x) = \sqrt{\det M(x)}, \tag{25}$$

where $\varrho(x)$ is the local metric density. The multidimensional equidistribution condition is

$$\varrho(x)J = \frac{\sigma}{|\Omega_c|}, \qquad \sigma = \int_{\Omega_p} \varrho(x)\, dx, \tag{26}$$

where J is the determinant of the Jacobian of the coordinate mapping, $|\Omega_c|$ is the measure of the computational domain, and σ is the total metric density. Since J characterizes the local element-volume scale, a larger $\varrho(x)$ requires a smaller local element. Mesh resolution is therefore concentrated in regions with larger approximation difficulty.

The monitor matrix can be constructed from the physical solution. Gradient information emphasizes rapid first-order variation, while Hessian information captures curvature and directional second-order variation. When the monitor is derived from interpolation error, $\varrho(x)$ can be interpreted as a local error density. Equidistribution then requires each mesh element to carry a comparable amount of this density. In discrete form,

$$|K|\sqrt{\det M_K} \approx C, \tag{27}$$

where K is a mesh element, $|K|$ is its area or volume, M_K is the local monitor matrix, and $C > 0$ is approximately constant across the mesh. Thus, equidistribution does not seek uniform element sizes. It seeks a spatially varying resolution that balances local approximation difficulty.

B Analysis of error-equidistribution loss L_{eq}

B.1 Graph-Laplacian Approximation of Hessian

PACE uses the graph-Laplacian term in Eq.19 to capture second-order physical variation. The connection follows from the continuous identity

$$\Delta u = \operatorname{tr}\left(D^2 u\right), \tag{28}$$

where u is a scalar field, D^2u is its Hessian matrix, Δu is the Laplacian, and $\operatorname{tr}(\cdot)$ denotes the matrix trace. The Laplacian is therefore a scalar summary of the Hessian response.

Graph Laplacians provide a discrete counterpart of this operator on irregular point sets and meshes Meyer et al. (2003); Hein et al. (2007). For a smooth field, the value at a neighboring node can be expanded around the center node by a second-order Taylor expansion. Over a locally balanced neighborhood, the first-order terms largely cancel. The remaining response is dominated by the quadratic term associated with the Hessian. Neighbor aggregation therefore provides a local estimate of second-order variation. This connection underlies the use of graph Laplacians as approximations of continuous Laplace-Beltrami operators.

Eq.19 follows the same idea. It compares the physical state at a node with the mean state of its one-ring neighbors. A locally linear field produces a small response. Curvature, local extrema, or rapidly changing gradients produce a larger response. The resulting $\nabla^2 p^e$ therefore serves as a scalar proxy for the Hessian-related second-order variation. It complements the first-order term ∇p^e in Eq.21, allowing η_e to describe both direct physical variation and changes in that variation.

B.2 Edge-Wise Error Equidistribution

The equidistribution principle in Eq.26 requires local approximation difficulty and local mesh scale to compensate for each other. PACE applies this idea to individual mesh edges. The physical variation η_e in Eq.21 summarizes the first- and second-order variation around edge e. The predicted edge coarsening score s_e controls its target scale through $\hat{l}_e = s_e \bar{l}_e$, where $\bar{l}_e$ is the local reference length and $\hat{l}_e$ is the target edge length.

Classical interpolation estimates have the general structure

$$E_K \lesssim A_K h_K^r, \tag{29}$$

where E_K is the local approximation error of element K, A_K represents local solution regularity or derivative information, h_K is the local mesh size, and $r > 0$ is the corresponding error-order exponent Nochetto & Veeser (2011). Thus, local error combines a physical-difficulty term with a positive power of mesh scale.

PACE retains this structure at the edge level:

$$\widetilde{E}_e = \eta_e s_e^{\gamma}, \tag{30}$$

where $\widetilde{E}_e$ is the relative post-coarsening error proxy and $\gamma > 0$ is an effective scale-sensitivity exponent. Here, η_e approximates the local difficulty term A_K, while s_e^{γ} models the change induced by local coarsening.

Error equidistribution then requires

$$\eta_e s_e^{\gamma} \approx C, \tag{31}$$

where $C > 0$ is a common error level over the interior edges. This immediately gives

$$s_e \propto \eta_e^{-1/\gamma}. \tag{32}$$

Hence, regions with stronger physical variation receive smaller coarsening scores, while smoother regions allow larger scores. Eq.22 implements Eq.31 by minimizing the variance of $\log(\eta_e s_e^{\gamma})$. Equal logarithmic values correspond to equal error proxies. The loss therefore turns the classical equidistribution principle into a differentiable objective for learning the edge-wise coarsening score field.

C THE PROPOSED MESH COARSENING METHOD

The inference procedure of PACE is summarized in Alg.1. Given a dense mesh and the corresponding coarse physical field, PACE first predicts the edge-wise coarsening score map and then iteratively coarsens the mesh through topology-constrained parallel edge collapse. In this algorithm, candidate selection, legality checking, and topology updates are local operations and can be performed in batches, ensuring high efficiency in the mesh coarsening process.

The legality checks in Alg.1 are implemented through four constraints. These constraints are applied at each coarsening round to ensure that the predicted resolution allocation can be realized on a valid mesh.

Dynamic topology update constraint. After each parallel collapse, the mesh connectivity and edge lengths are updated before constructing the next candidate set. The original predicted edge-wise score field remains fixed during coarsening. Using Eq.2, we first compute the original edge-wise target length

$$\hat{l}_e = s_e \bar{l}_e. \tag{33}$$

To propagate this frozen target-size field to the evolving mesh, we define a frozen nodal target size at each vertex v_i by averaging the target lengths of its incident edges:

$$\hat{l}_i = \frac{1}{|\mathcal{E}(v_i)|} \sum_{e' \in \mathcal{E}(v_i)} \hat{l}_{e'}, \tag{34}$$

where $\mathcal{E}(v_i)$ denotes the set of edges incident to v_i. For a current edge $e = (v_i, v_j)$, the updated target length is then obtained by

$$\hat{l}_e^{\mathrm{cur}} = \frac{1}{2}\left(\hat{l}_i + \hat{l}_j\right). \tag{35}$$

An edge is retained in the current candidate set $\mathcal{C}$ only if

$$l_e < \tau \hat{l}_e^{\mathrm{cur}}, \tag{36}$$

where l_e is the current length of edge e and $\tau = 0.8$ is the collapse tolerance. This update makes each collapse round depend on the current mesh geometry while preserving the originally predicted resolution allocation.

Algorithm 1 PACE inference flow

Require: Dense mesh M, coarse physical field $\mathbf{p}^c$, model ϕ, target coarsening ratio R_{tar}
Ensure: Coarsened mesh M^c

Score Map Prediction:
Construct the edge-based graph G from M
$\boldsymbol{S} \leftarrow \phi(G) = \{s_e\}_{e \in E_M}$
Candidate Construction:
Compute the local reference length $\bar{l}_e$ for each $e \in E_M$
$\hat{l}_e \leftarrow s_e \bar{l}_e$
Construct the candidate set $\mathcal{C} \leftarrow \{e \in E_M : l_e < \hat{l}_e\}$
while $\mathcal{C} \neq \varnothing$ **or** R_{tar} **is not reached do**
    Rank candidate edges according to $\hat{l}_e$
    Select a one-ring disjoint legal subset $\mathcal{B} \subseteq \mathcal{C}$
    Parallel Edge Collapse:
    for all $e \in \mathcal{B}$ **do**
        Determine the collapse position
        Apply constraints
        if e violates any constraint **then**
            Remove e from $\mathcal{B}$
        end if
    end for
    Collapse all valid edges in $\mathcal{B}$ in parallel
    Update mesh information
    Update $\mathcal{C}$
end while
$M^c \leftarrow M$

Topology preservation constraint. Once an edge is selected, its two endpoints and their one-ring neighbors are marked as reserved. Here, $\mathcal{R}(e)$ denotes the protected one-ring region of edge e, including its endpoints and their one-ring neighboring vertices. The selected batch satisfies

$$\mathcal{R}(e) \cap \mathcal{R}(e') = \varnothing, \qquad \forall\, e \neq e' \in \mathcal{B}. \tag{37}$$

Thus, no two collapsed edges in the same batch modify overlapping local neighborhoods. The mesh connectivity is rebuilt after each batch to avoid conflicting local topology updates.

Boundary constraint. Boundary nodes are identified from the current mesh connectivity. Their collapse positions are constrained according to Alg.2 to preserve the prescribed domain boundary.

Algorithm 2 Boundary constraint in parallel edge collapse

Require: Candidate edge $e = (v_i, v_j)$, endpoint coordinates $\mathbf{x}_i, \mathbf{x}_j$, boundary indicators $b_i, b_j \in \{0, 1\}$
Ensure: Collapse decision and, if valid, new vertex position $\mathbf{x}_e^{\text{new}}$

if $b_i = 1$ **and** $b_j = 1$ **then**
    Reject e
else if $b_i = 1$ **then**
    $\mathbf{x}_e^{\text{new}} \leftarrow \mathbf{x}_i$
else if $b_j = 1$ **then**
    $\mathbf{x}_e^{\text{new}} \leftarrow \mathbf{x}_j$
else
    $\mathbf{x}_e^{\text{new}} \leftarrow (\mathbf{x}_i + \mathbf{x}_j)/2$
end if

Mesh quality constraint. The candidate collapse is accepted only if every affected triangle preserves its orientation, remains non-degenerate, and satisfies the aspect-ratio bound:

$$J_f^{\text{old}} J_f^{\text{new}} > 0, \qquad |J_f^{\text{new}}| > \varepsilon_J, \qquad q_{\text{AR},f}^{\text{new}} < q_{\max}, \qquad \forall f \in \mathcal{F}(e), \tag{38}$$

where $\mathcal{F}(e)$ denotes the set of triangles affected by collapsing edge e, J_f and $q_{\mathrm{AR},f}$ follow Eq.58 and Eq.57, respectively, $\varepsilon_J = 10^{-14}$ is the minimum Jacobian threshold, and $q_{\max} = 5$ is the prescribed upper bound of the aspect ratio. Candidates violating any condition are rejected.

D DATASET

D.1 TRAIN DATA SETUPS

We use 60 transient physical samples to train PACE. The dataset contains four governing equations:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} - \nu\nabla^2\mathbf{u} = 0, \qquad \text{Burgers,} \tag{39}$$

$$\frac{\partial p}{\partial t} - \kappa\nabla^2 p = 0, \qquad \text{Pressure diffusion,} \tag{40}$$

$$\frac{\partial u}{\partial t} + \mathbf{a}\cdot\nabla u - \kappa\nabla^2 u = 0, \qquad \text{Scalar advection diffusion,} \tag{41}$$

$$\frac{\partial^2 \mathbf{U}}{\partial t^2} - c^2\nabla^2\mathbf{U} = 0, \qquad \text{Vector wave.} \tag{42}$$

The initial conditions are constructed from several analytic function families. They include sinusoidal modes, polynomial-modulated modes, localized Gaussian profiles, multi-Gaussian profiles, and vortex-like vector fields. All training data are generated on a rectangular computational domain using three triangular meshing strategies: the Delaunay triangulation, the advancing-front method, and the orthogonal advancing-front method. For each case, the PDE is solved on paired coarse and dense meshes. At each time step, barycentric interpolation maps the coarse physical field to the dense mesh nodes. The interpolated field is used as the model input. The high-fidelity solution provides the teacher signal during training.

Table 3: Dense and coarse mesh sizes used for training.

Mesh	Dense mesh	Coarse mesh
Delaunay triangulation	14,516	3,766
Advancing-front method	11,192	2,766
Orthogonal advancing-front	9,522	2,312

D.2 TEST DATA SETUPS

The test set contains both canonical PDE benchmarks and real CFD cases.

PDE cases. The PDE cases cover seven governing equations. In addition to the four types of transient equations introduced in Appendix D.1, we further include the Poisson equation, Helmholtz equation, and vector reaction-diffusion equation for testing.

Poisson equation. We solve

$$\nabla^2 u = f, \qquad u = u^\star \text{ on } \partial\Omega, \tag{43}$$

where $u^\star$ is a manufactured solution and $f = \nabla^2 u^\star$.

Helmholtz equation. We solve

$$-\nabla^2 u + u = f, \qquad u = u^\star \text{ on } \partial\Omega, \tag{44}$$

where $f = -\nabla^2 u^\star + u^\star$.

Vector reaction diffusion equation. We solve

$$\frac{\partial \mathbf{U}}{\partial t} - D\nabla^2\mathbf{U} = \alpha\mathbf{U} - \beta\|\mathbf{U}\|_2^2\mathbf{U}, \tag{45}$$

where $D = 10^{-3}$, $\alpha = 1$, and $\beta = 0.1$.

The test initial conditions and manufactured solutions are sampled from a separate analytic function library. We also evaluate the model on irregular computational domains. This setting tests cross-geometry generalization. The corresponding results are reported in Appendix F.

Real CFD cases. We further evaluate the method on airfoil and cylinder flow benchmarks. The flow conditions and mesh resolutions are summarized in Table 4. The low-speed airfoil cases include CLARKY, FX63137, N0012, and N63-215. They use incompressible Reynolds-averaged Navier–Stokes (RANS) with the Spalart-Allmaras turbulence model. The freestream velocity is $30\,\mathrm{m/s}$, except for N0012, where $U_\infty = 52.08\,\mathrm{m/s}$. The simulations use 400 steady pseudo-time iterations.

The transonic RAE2822 case uses compressible RANS with the k-ω shear stress transport (SST) turbulence model. The freestream temperature is $300\,\mathrm{K}$. The simulations use 5000 steady pseudo-time iterations.

The cylinder case considers 2D incompressible laminar flow at $Re = 1000$. We set $U_\infty = 5\,\mathrm{m/s}$ and $\nu = 0.01\,\mathrm{m}^2/\mathrm{s}$. The time step is $0.001\,\mathrm{s}$ and the simulation time is $30\,\mathrm{s}$.

Table 4: Flow conditions and mesh resolutions of the airfoil and cylinder flow test cases. The uniform coarse mesh size is used as the target size for learned mesh coarsening methods.

Case	α (°)	U_∞ (m/s)	Dense mesh	Coarse mesh
CLARKY	0	30.00	162,810	10,102
FX63137	15	30.00	187,436	10,086
N0012	0, 10	52.08	184,144	9,738
N63-215	0, 5	30.00	173,322	10,084
RAE2822	2.31	253.12	179,104	10,038
Cylinder	-	5.00	248,924	5,302

E Experiment setup

E.1 Training and testing details

PACE, MeshCNN, and MeshGraphNet use exactly the same training, validation, and test splits to ensure a fair comparison. All trainable methods use a learning rate of 3×10^{-5}, zero weight decay, and a batch size of 1. Early stopping is applied with a patience of 60 epochs, and the checkpoint with the lowest validation loss is retained for evaluation. The temporal encoder uses four attention heads with zero dropout. Physical fields are normalized using sequence-wise robust normalization, where the normalization statistics are computed over the complete valid temporal sequence of each sample.

E.2 Baseline details

We compare PACE with two learning-based baselines and three QEM-based baselines.

MeshCNN. MeshCNN Hanocka et al. (2019) was originally developed for 3D mesh classification and segmentation, with edge convolution and task-driven mesh pooling as its main components. We retain the edge-convolution backbone and use the pooling and unpooling branches only for multiscale feature extraction. The original geometric encoder is replaced by the geometric and physical encoders. The decoder predicts a size factor s_e for each edge in the original mesh.

MeshGraphNet. MeshGraphNet Pfaff et al. (2021) was originally designed to predict the temporal evolution of physical states on computational meshes. We retain its Encode-Process-Decode message-passing backbone and adapt the graph representation to edge-wise mesh coarsening. Each graph node receives geometric features together with the temporal encoding of $\boldsymbol{p}^c$. The decoder predicts the edge-wise size factor s_e instead of a physical-state update.

Geometric QEM. We use the classical QEM Hoppe (1999) as a geometry-only baseline. For an edge $e_{ij} = (v_i, v_j)$, the collapse cost is defined as

$$C_{\mathrm{QEM}}(e_{ij}) = \min_{\tilde{\boldsymbol{x}}} \tilde{\boldsymbol{x}}^{\mathrm{T}} (\boldsymbol{Q}_i + \boldsymbol{Q}_j) \tilde{\boldsymbol{x}}. \tag{46}$$

Here, $\boldsymbol{Q}_i$ and $\boldsymbol{Q}_j$ are the quadric error matrices associated with vertices v_i and v_j, respectively, and $\tilde{\boldsymbol{x}} = [x, y, z, 1]^{\mathrm{T}}$ denotes the homogeneous coordinate of the merged vertex. A smaller C_{QEM} indicates a smaller geometric distortion and therefore a higher collapse priority.

Gradient-QEM. Gradient-QEM augments the geometric QEM cost with a physics-based penalty derived from the coarse physical field $\boldsymbol{p}^c$. The penalty increases with the magnitude of the local physical gradient, and the collapse cost is defined as

$$C_{\mathrm{grad}}(e_{ij}) = C_{\mathrm{QEM}}(e_{ij}) + \frac{\bar{g}_i + \bar{g}_j}{2}. \tag{47}$$

Here, $\bar{g}_i$ and $\bar{g}_j$ denote the normalized physical-gradient magnitudes at vertices v_i and v_j, respectively.

Equidistribution-QEM. Equidistribution-QEM determines the initial collapse priority from the physical importance of each edge. The physical variation η_e is computed from Eq.21. Based on this variation, the analytical target size factor and the corresponding collapse cost are defined as

$$s_e^{\mathrm{ana}} = \left(\frac{C}{\eta_e}\right)^{5/4}, \qquad C_{\mathrm{eq}}(e) = \frac{1}{s_e^{\mathrm{ana}}}. \tag{48}$$

Here, $C = \mathrm{median}_e\, \eta_e$ denotes the median physical importance over all candidate edges, and s_e^{ana} is the analytical target size factor. A smaller η_e leads to a larger target mesh scale and therefore a lower collapse cost.

E.3 Metric details

Element reduction ratio. We measure the mesh coarsening level by the relative decrease in the number of triangular elements:

$$R_e = 1 - \frac{N_e^s}{N_e^d}. \tag{49}$$

Here, N_e^d and N_e^s denote the numbers of elements in the original dense mesh and the coarse mesh, respectively. A larger R_e indicates a stronger mesh coarsening.

Mean relative L_2 error. For each time step t, we compute the finite element mass-weighted relative L_2 error as

$$r_t = \frac{\left(\int_\Omega \left|u_s^{(t)} - u_d^{(t)}\right|^2 d\Omega\right)^{1/2}}{\left(\int_\Omega \left|u_d^{(t)}\right|^2 d\Omega\right)^{1/2} + \epsilon}. \tag{50}$$

The temporal mean is

$$r_{mean} = \frac{1}{T}\sum_{t=1}^{T} r_t. \tag{51}$$

Here, $u_s^{(t)}$ is the coarse mesh solution interpolated onto the dense mesh at time step t, $u_d^{(t)}$ is the corresponding dense-mesh reference solution, Ω is the computational domain, T is the number of sampled time steps, and ϵ is a small constant for numerical stability. Lower values indicate better agreement with the reference solution.

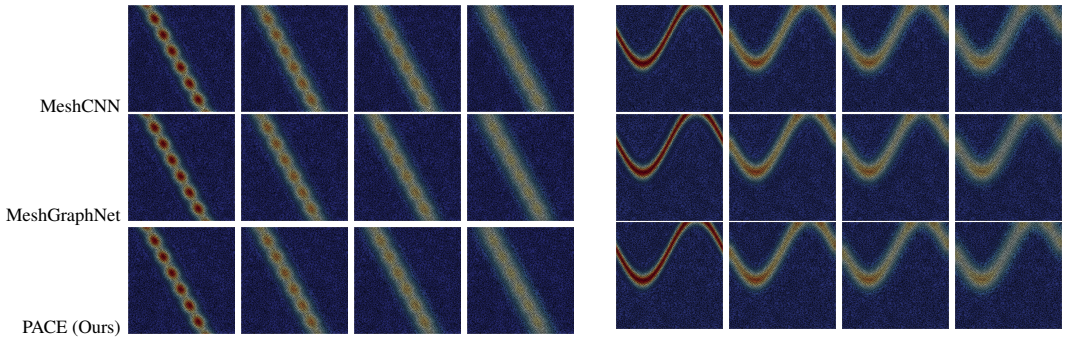
Maximum relative L_2 error. To measure the worst temporal error during a rollout, we report

$$r_{\max} = \max_{1 \le t \le T} r_t. \tag{52}$$

Here, r_t is the relative L_2 error at time step t defined above.

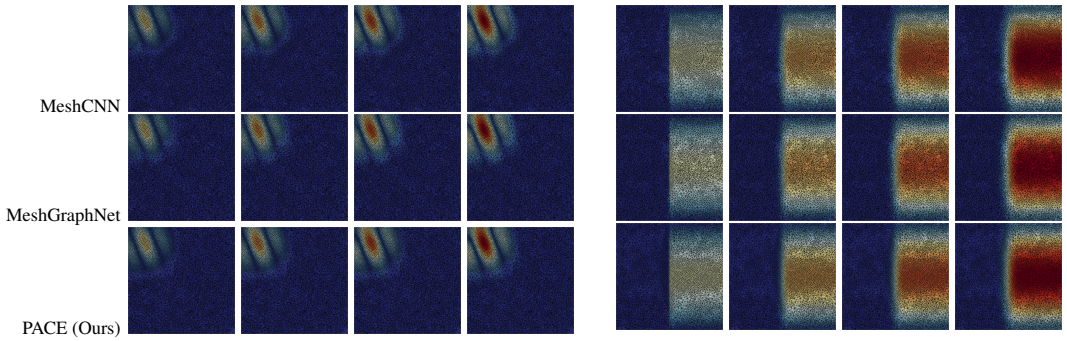
Rollout RMSE. We evaluate the overall trajectory error by accumulating the spatial squared error over all sampled time steps:

$$RMSE_{rollout} = \sqrt{\frac{\sum_{t=1}^{T}\int_\Omega \left|u_s^{(t)} - u_d^{(t)}\right|^2 d\Omega}{\sum_{t=1}^{T}\int_\Omega 1\, d\Omega}}. \tag{53}$$



(a) Pressure diff (1).

(b) Pressure diff (2).



(c) Vector react.-diff (1).

(d) Vector react.-diff (2).

Figure 6: Qualitative comparison of temporal solution fields produced using meshes coarsened by MeshCNN, MeshGraphNet, and PACE.

For vector-valued fields, the squared errors of the two vector components are accumulated jointly.

Solving speedup. The reduction in PDE solving cost is measured by

$$S_{solve} = \frac{T^d_{solve}}{T^s_{solve}}. \tag{54}$$

Here, T^d_{solve} and T^s_{solve} are the solving times on the dense and coarse meshes, respectively.

Total speedup. To account for the additional mesh coarsening cost, we define the total runtime as

$$T_{total} = T_{coarse} + T^s_{solve}, \tag{55}$$

and the total speedup as

$$S_{total} = \frac{T^d_{solve}}{T_{coarse} + T^s_{solve}}. \tag{56}$$

Here, T_{coarse} is the mesh coarsening time, while T^d_{solve} and T^s_{solve} are the dense- and coarse mesh solving times.

Aspect ratio. For a triangular element f with edge lengths l_0, l_1, and l_2, we define

$$q_{AR} = \frac{l_{\max}(l_0 + l_1 + l_2)}{4\sqrt{3}A}, \qquad l_{\max} = \max(l_0, l_1, l_2), \tag{57}$$

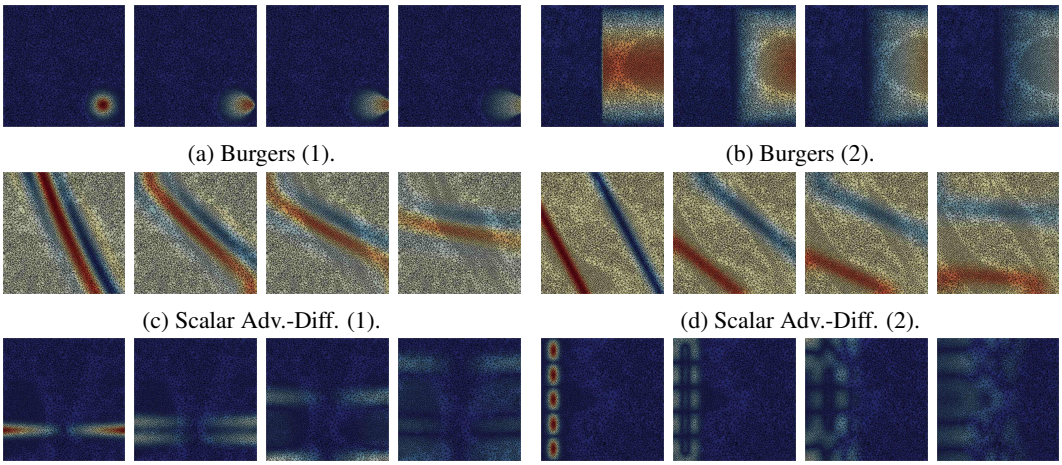
where A is the triangle area. An equilateral triangle satisfies $q_{AR} = 1$, while larger values indicate increasingly elongated or degenerate elements.

Minimum Jacobian. For a two-dimensional triangular element f, the signed Jacobian determinant is

$$J_f = 2A_{s,f}. \tag{58}$$

Here, $A_{s,f}$ is the signed area of triangle f. The mesh-level metric is

$$J_{\min} = \min_f J_f. \tag{59}$$



(a) Burgers (1). (b) Burgers (2).

(c) Scalar Adv.-Diff. (1). (d) Scalar Adv.-Diff. (2).

(e) Vector wave (1). (f) Vector wave (2).

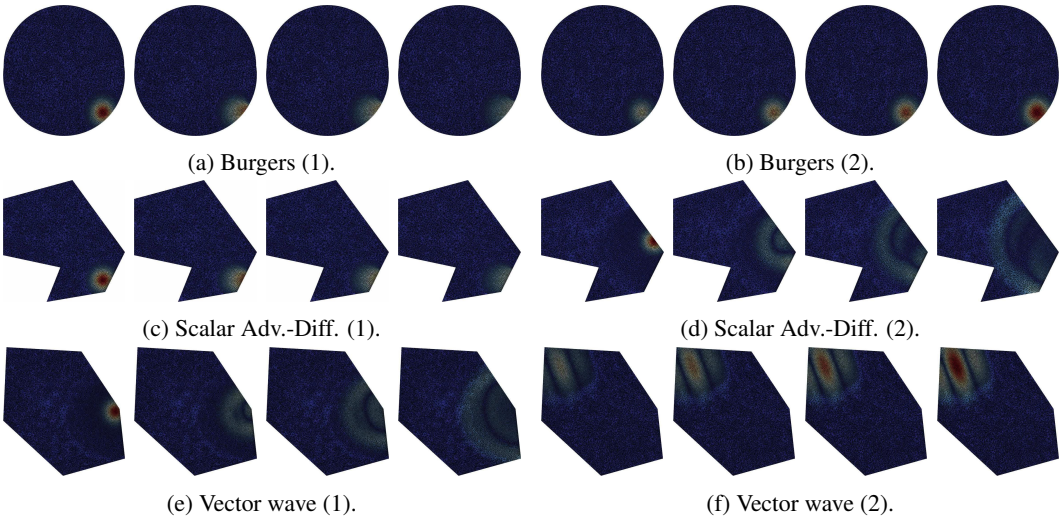
Figure 7: Temporal visualization of PACE coarsening results on six transient PDE cases.

A positive value indicates consistent element orientation, a value close to zero indicates near-degenerate elements, and a negative value indicates at least one flipped element.

F Experiment results

F.1 Visualization results

The additional qualitative results further illustrate how PACE dynamically allocates mesh resolution across different transient physical systems. As shown in Fig.6, the differences among coarsening strategies are most pronounced around localized peaks and narrow transition regions. PACE consistently preserves these critical structures over time while reducing redundant elements in relatively smooth areas. Fig.7 extends this observation to the Burgers, scalar advection-diffusion, and vector wave equations. Despite their distinct temporal dynamics, the retained high-resolution regions consistently align with the evolving physical structures. These results indicate that the learned coarsening strategy adapts to the underlying physical states and effectively captures spatially varying resolution requirements across different systems.



(a) Burgers (1). (b) Burgers (2).

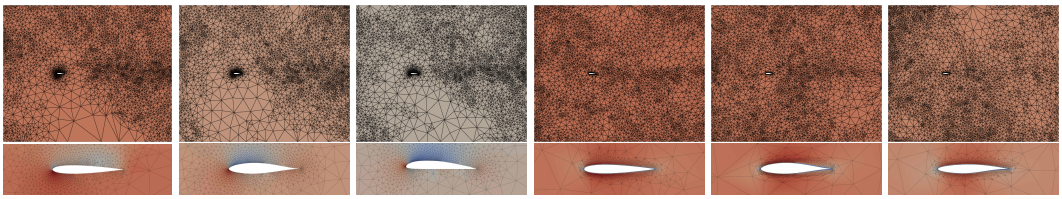
(c) Scalar Adv.-Diff. (1). (d) Scalar Adv.-Diff. (2).

(e) Vector wave (1). (f) Vector wave (2).

Figure 8: Temporal visualization of PACE on unseen geometries.

We further examine the behavior of PACE on irregular computational domains in Fig.8. The circular and polygonal geometries introduce substantially different boundary shapes and mesh connectivity from the regular-domain examples. Nevertheless, PACE consistently preserves higher mesh resolution around dominant physical structures while applying more aggressive coarsening in less important regions as these structures evolve across the domain.

Finally, Fig.9 provides additional results on real-world airfoil configurations. Across different airfoil geometries and operating conditions, PACE produces clearly non-uniform meshes and adjusts the near-field resolution to the corresponding flow pattern.



N0012 $\alpha = 10^\circ$ | N63-215 $\alpha = 5^\circ$ | CLARKY $\alpha = 0^\circ$ | N0012 $\alpha = 0^\circ$ | N63-215 $\alpha = 0^\circ$ | RAE2822 $\alpha = 2.31^\circ$

Figure 9: Visualization of PACE coarsening results on different airfoils. The first three cases correspond to pressure fields, while the last three cases correspond to velocity fields.

F.2 Ablation Study

We conduct ablation and comparative experiments on the topology-constrained parallel edge-collapse strategy to verify its effectiveness. Table 5 shows that the full constraint set produces valid meshes in all test cases, with no duplicate triangles, non-manifold edges, or orientation mismatches. Removing the validity constraints leads to severe mesh defects, while boundary protection alone is insufficient. With the full constraints enabled, Fig.10 compares serial and parallel execution under identical edge priorities. Parallel collapse achieves average speedups of $2.68\times$ and $1.65\times$ at 20% and 40% reduction, respectively. The smaller speedup at 40% reduction is mainly due to the larger number of coarsening rounds, which increases the relative cost of candidate filtering and validity checks. These results show that the constrained parallel strategy improves coarsening efficiency while preserving mesh validity.

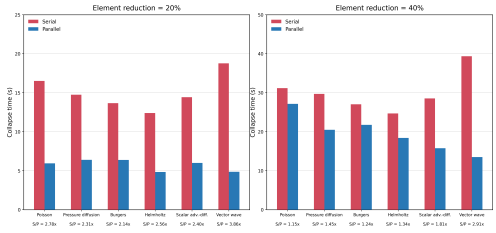


Figure 10: Comparison of serial and parallel collapse efficiency under different coarsening ratios.

Table 5: Ablation of topology constraints in the parallel edge collapse. Count-based metrics report the average number of defects per output mesh. Among them, **Mean AR**, **Deg. Tri.**, **Dup. Tri.**, **Non-man. Edge**, **Orient. Mismatch**, and **Area Err.** denote mean aspect ratio, degenerate triangles, duplicate triangles, non-manifold edges, triangle orientation mismatches, and relative domain-area error, respectively.

Constraint setting	Valid Mesh (%) ↑	Mean AR ↓	Deg. Tri. ↓	Dup. Tri. ↓	Non-man. Edge ↓	Orient. Mismatch ↓	Area Err. (%) ↓	Max Degree ↓
Full constraints	**100.0**	**1.43**	**0.00**	**0.00**	**0.00**	**0.00**	**4.20e−15**	**10.38**
w/o validity	0.0	7.11e9	3.21	24.29	25.46	60.58	0.67	20.96
Boundary only	4.17	7.14e9	2.63	34.75	35.79	65.54	0.98	17.88
Unconstrained	4.17	4.96e7	0.08	35.83	36.83	64.92	0.72	17.58