AMS-GNO: AN AGGREGATION-BASED MULTISCALE GRAPH NEURAL OPERATOR FOR LARGE-SCALE PDES

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural operators provide efficient surrogates for repeated numerical simulation by learning mappings from PDE parameters, geometries, and boundary conditions to solution fields. Scaling neural operators to large unstructured meshes and point clouds, however, remains challenging: purely local operators require many hops to capture long-range dependencies, whereas global attention or dense soft-assignment mechanisms introduce non-negligible computational and memory overhead by coupling every node to every latent token. Motivated by the inherently local structure of PDEs, we propose AMS-GNO, an Aggregation-based Multiscale Graph Neural Operator that performs long-range interaction on a coarsened graph while preserving sparse, topology-aware communication across scales. AMS-GNO first partitions the original mesh or point cloud into balanced local aggregates using geometric and topological priors, preserving physical connectivity while controlling aggregation size. Local smoothing then expands these hard assignments into a sparse overlapping cross-scale graph, allowing nodes near aggregate boundaries to interact with multiple neighboring coarse nodes. Restricted to this sparse support, attention-based operators separately learn downsampling and upsampling, adaptively determining which fine-scale information should be retained at coarse scales and how coarse representations should be reconstructed. The resulting architecture integrates fine-scale local processing, overlapping cross-scale transfer, and coarse-scale long-range interaction within a unified multiscale framework. Across multiple PDE benchmarks involving complex geometries, AMS-GNO achieves competitive or superior performance relative to state-of-the-art baselines while retaining favorable computational efficiency. Scalability experiments further show approximately linear growth in training time and memory consumption with the number of nodes, and demonstrate applicability to industrial-scale datasets containing millions of nodes. These results suggest that AMS-GNO provides an effective route toward scalable neural operators for high-resolution simulation on complex unstructured domains.

1 INTRODUCTION

Partial differential equations (PDEs) are fundamental tools for describing complex physical systems in fluid dynamics, solid mechanics, materials science, electromagnetics, climate science, and many other domains. Solving them efficiently is essential to both scientific discovery and engineering design. Conventional numerical methods discretize a continuous computational domain and obtain approximate solutions using finite-difference, finite-volume, finite-element, or spectral methods. For high-resolution, high-dimensional, or multiscale problems, even a single solve can demand substantial computational and memory resources. This cost accumulates rapidly in applications that require repeated PDE solutions, such as parameter sweeps, inverse problems, real-time control, and design optimization. Neural surrogate models learn mappings from problem inputs to solution fields using precomputed simulation data. Once trained, they can rapidly approximate solutions for new instances within the learned problem family, amortizing the offline cost of data generation and training over repeated queries (Azizzadenesheli et al., 2024).

Neural operators extend surrogate modeling from finite-dimensional vector mappings to mappings between function spaces, learning operators from PDE parameters, coefficients, source terms, or boundary conditions to solution fields (Kovachki et al., 2023). DeepONet provides an influential

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.

realization through branch and trunk networks that encode the input function and output coordinates, respectively (Lu et al., 2021). Other widely used architectures organize spatial communication through spectral kernels, graph neighborhoods, or attention. These mechanisms offer complementary ways to combine local physical structure with the nonlocal dependence of PDE solutions.

Frequency-based methods such as the Fourier Neural Operator (FNO) provide efficient global interaction by parameterizing integral kernels in the Fourier domain (Li et al., 2020a). Their standard FFT-based implementation uses a regular grid, so irregular geometries generally require an additional geometric representation or transfer mechanism. Geo-FNO learns coordinate deformations to a latent domain amenable to Fourier computation (Li et al., 2023a), whereas GINO uses graph neural operators to map between an irregular discretization and a regular latent grid on which an FNO is applied (Li et al., 2023b). Graph-based methods instead operate directly on irregular domains through neighborhood-based kernel integration or message passing (Li et al., 2020b; Pfaff et al., 2020). When supplied with simulation-mesh connectivity, mesh GNNs retain a strong local geometric inductive bias together with information about valid adjacency, boundaries, and thin structures. Their communication range, however, is controlled by graph hops: a fixed-depth local message-passing network has a limited receptive field. Moreover, mesh refinement changes the physical distance represented by a fixed k-hop neighborhood, making efficient long-range communication increasingly important (Fortunato et al., 2022).

Transformer-based solvers offer another route to long-range communication, but standard self-attention over N spatial points requires $\mathcal{O}(N^2)$ pairwise interactions. A common strategy compresses fine-scale points into a smaller latent set before applying global attention. EAGLE uses graph pooling before global attention, while Transolver constructs learned physical-state slices and performs attention over the resulting latent tokens (Janny et al., 2023; Wu et al., 2024). Universal Physics Transformers similarly process compact latent representations and decode them at queried locations (Alkin et al., 2024). A complementary strategy restricts attention to spatial or graph neighborhoods, as in HAMLET's graph-transformer encoder (Bryutkin et al., 2024). Such local attention models nearby interactions without forming a dense point-to-point matrix, but domain-wide communication still requires additional depth or a complementary global mechanism. Importantly, latent compression and sparse neighborhoods address different costs: a dense assignment from N points to S tokens still involves $\mathcal{O}(NS)$ interactions, even when S is fixed and the resulting input-size scaling is linear (Wu et al., 2024).

Multiscale graph architectures provide a natural way to organize these trade-offs. Fine levels preserve local geometric detail, whereas coarse representations enlarge the effective receptive field and shorten communication paths between distant parts of the domain. This separation has a classical counterpart in algebraic multigrid, including smoothed-aggregation methods that construct coarse spaces from local aggregates (Vaněk et al., 1996). In operator learning, the Multipole Graph Neural Operator organizes interactions across levels to capture short- and long-range effects with linear evaluation complexity (Li et al., 2020c). MultiScale MeshGraphNets and BSMS-GNN introduce coarse graph levels for more efficient mesh-based simulation (Fortunato et al., 2022; Cao et al., 2023), while RIGNO passes information between physical point clouds through a downsampled regional mesh (Mousavi et al., 2026). These methods already establish the value of multiscale processing and learned inter-level communication. The design question is therefore not whether to learn cross-scale transfer, but how to combine its adaptivity with sparse, balanced neighborhoods that respect the available input connectivity.

Two considerations motivate our approach. First, *topology fidelity and resolution flexibility* are related but distinct. Native mesh connectivity supplies physical adjacency, whereas coordinate-defined neighborhoods can be rebuilt on new point sets. GINO and GAOT use the latter construction; GAOT combines multiradius attentional graph encoders and decoders with a Transformer processor (Li et al., 2023b; Wen et al., 2026). However, coordinate proximity alone need not reflect connectivity around thin structures or separated boundaries. Conversely, accepting different node counts does not by itself ensure consistent predictions under refinement: cross-resolution accuracy also depends on the discretized operators and their stability (Bartolucci et al., 2023). A useful hierarchy should therefore be reconstructible on each input graph without discarding available connectivity, while its resolution generalization remains subject to analysis and empirical evaluation.

Second, *adaptive transfer and scalable execution* must be considered together. Dense point-to-token assignment permits flexible aggregation, but does not explicitly restrict transfer to local, topology-

aware support. Sparse hierarchies offer such a restriction, making the choice of aggregation neighborhoods and their learned weights central design decisions. To make inter-level message passing efficient while retaining adaptivity, we introduce an edge-aware Transformer that operates only on sparse cross-scale edges. Large-scale execution is already addressed by complementary approaches: X-MeshGraphNet combines multiscale graphs with partition-based execution (Nabian et al., 2024), and Transolver++ extends Transformer solvers to industrial-scale geometries (Luo et al., 2025). These advances motivate a more specific objective: integrating connectivity-aware compression, adaptive bidirectional transfer, and coarse-scale global interaction without dense fine-to-coarse assignments.

To this end, we propose **AMS-GNO** (Aggregation-based Multiscale Graph Neural Operator), a framework built on a topology-aware sparse hierarchy and learnable cross-scale operators. AMS-GNO reconstructs its hierarchy from each input graph, using native mesh connectivity when available or a neighborhood graph for point clouds. Balanced-Lloyd aggregation (Zaman et al., 2024) forms approximately balanced local aggregates, and local smoothing expands their hard assignments into sparse overlapping cross-scale neighborhoods. Inspired by smoothed aggregation (Vaněk et al., 1996), this construction determines where transfer is permitted; independently parameterized attention operators learn how information is aggregated and reconstructed in the two directions. Domain-wide communication is performed only after compression, using Performer-style linear attention at the coarsest level (Choromanski et al., 2020). These properties are designed to combine topology fidelity, resolution flexibility, long-range communication, and scalable computation within a single multiscale framework.

Our main contributions are summarized as follows:

- We introduce a topology-aware multiscale graph neural operator whose sparse hierarchy is reconstructed from each input graph. Balanced local aggregation and sparse overlap retain the available connectivity as a prior while allowing the learned operators to be applied to discretizations with different node counts and resolutions.
- We develop independently parameterized, state-dependent bidirectional cross-scale operators for fine-to-coarse aggregation and coarse-to-fine reconstruction, allowing inter-level transfer to adapt to the current physical features on a shared sparse support.
- We place domain-wide communication on compressed coarse representations, separating local transfer from global interaction and avoiding quadratic self-attention over the original large mesh. Sparse transfer and linear coarse attention yield linear model-execution complexity under bounded-degree and fixed-architecture assumptions; hierarchy preprocessing is treated separately.
- Across the evaluated PDE benchmarks, AMS-GNO is competitive with or improves upon the reported baselines, while the cross-resolution and large-mesh experiments evaluate its resolution flexibility and computational scalability.

Table 1 summarizes the evaluated baselines in terms of native topology, resolution flexibility, local and long-range modeling, attention-based transfer, and interaction scaling.

2 METHOD

Problem setup. Let $\Omega \subset \mathbb{R}^d$ be a bounded domain and $u : \Omega \to \mathbb{R}^{d_u}$ the solution field. We consider a parametric family of PDEs

$$\mathcal{F}(u; a) = 0 \quad \text{in } \Omega, \qquad \mathcal{B}(u; a) = 0 \quad \text{on } \partial\Omega, \tag{1}$$

where $a \in \mathcal{A}$ denotes the collection of coefficients, source terms, boundary conditions, and geometry specifying a PDE instance. For each a, Equation (1) determines a solution operator $\mathcal{G}^\dagger : a \mapsto u$. On a discretization $G = (V, E)$, where V is the set of N nodes and E denotes the mesh connectivity, with node coordinates X, the problem input and solution are represented by the nodewise fields A and U, respectively. Accordingly, we learn

$$\widehat{U} = \mathcal{G}_\theta(A, X, E) \approx U, \tag{2}$$

where the number of nodes, connectivity, and geometry may vary across samples.

Table 1: Architectural properties of the evaluated baselines. ✓: explicitly supported; –: not an explicit design component. Native topology denotes use of supplied mesh connectivity, and local structure denotes explicit geometric or graph neighborhoods. Cross-resolution flexibility denotes applicability to varying discretizations. Attention-based transfer denotes attention-weighted cross-scale or point-to-token exchange. Subquadratic scaling is with respect to input node count, assuming bounded graph degrees and fixed feature widths.

Model	Native topology	Cross-res. flexibility	Local structure	Long-range interaction	Attention transfer	Subquadratic interaction
MeshGraphNet	✓	✓	✓	–	–	✓
GINO	–	✓	✓	✓	–	✓
Geo-FNO	–	✓	–	✓	–	✓
Transolver	–	✓	–	✓	✓	✓
GAOT	–	✓	✓	✓	✓	✓
AMS-GNO	✓	✓	✓	✓	✓	✓

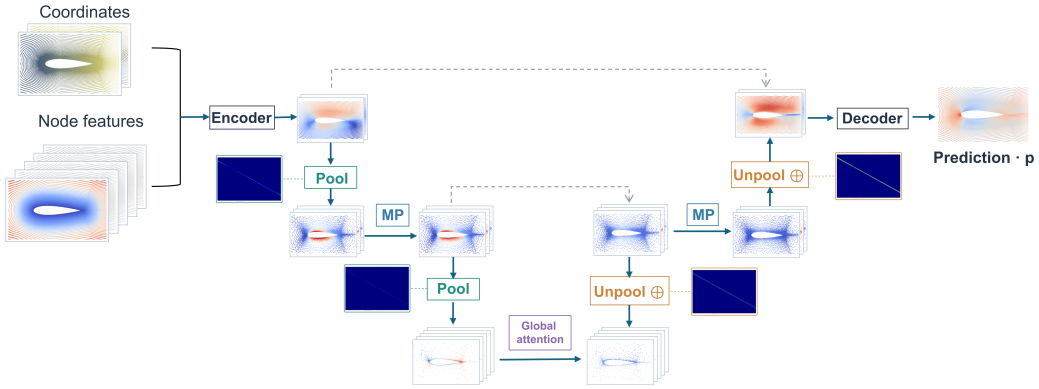

Figure 1: Overall architecture of AMS-GNO. The model follows an encoder–processor–decoder paradigm, with a multiscale V-cycle processor built from learnable Pool and Unpool operators, graph message passing, and coarse-level global communication.

2.1 Overall Model Architecture

As shown in Figure 1, AMS-GNO follows an encoder–processor–decoder architecture. Given a graph $G = (V, E)$, node coordinates $X \in \mathbb{R}^{N\times d}$, and nodewise inputs $A \in \mathbb{R}^{N\times d_a}$, a pointwise encoder lifts the concatenated inputs into a latent representation,

$$H^0 = \mathrm{Enc}_\phi([X, A]) \in \mathbb{R}^{N\times C_0}. \tag{3}$$

The processor then propagates these features through a multiscale graph hierarchy. Specifically, the downward path aggregates fine-scale features into coarser representations, with message passing at intermediate levels and Performer-style linear global attention (Choromanski et al., 2020) at the coarsest level. The upward path subsequently reconstructs fine-scale features through learnable Unpool operators and skip connections. Finally, a pointwise decoder maps the reconstructed representation to the target field:

$$H^P = \mathrm{Proc}_\psi(H^0; \mathcal{H}_G), \qquad \widehat{U} = \mathrm{Dec}_\omega(H^P), \tag{4}$$

where $\mathcal{H}_G$ denotes the multiscale hierarchy and $\theta = (\phi, \psi, \omega)$.

2.2 Graph Construction

To preserve local topology and sparsity while avoiding highly imbalanced aggregates, we adopt Balanced-Lloyd clustering followed by smoothing to construct the multiscale hierarchy. Starting from the original mesh graph $G_0 = (V_0, E_0)$, we recursively apply this procedure across levels

to obtain sparse cross-scale connectivity. Appendix C.1 illustrates the complete hierarchies, and Figure 6 compares local cross-scale connections with standard Lloyd clustering.

Multiscale hierarchy construction. At each level, Balanced-Lloyd partitions the current graph according to shortest path distance, producing approximately balanced clusters while preserving the underlying graph topology. An actual node is selected as the representative of each cluster, yielding the hard fine-to-coarse assignment

$$(H_\ell)_{ic} = \mathbf{1}[\pi_\ell(i) = c], \qquad H_\ell \in \{0,1\}^{N_\ell \times N_{\ell+1}}, \qquad \sum_c (H_\ell)_{ic} = 1, \tag{5}$$

To introduce limited overlap between neighboring clusters, we smooth the hard assignment over the local graph and sparsify the resulting weights:

$$P_\ell = \mathcal{J}_{\omega,m}(H_\ell; A_\ell), \qquad (B_\ell)_{ic} = \mathbf{1}\left[|(P_\ell)_{ic}| > \tau \text{ or } (H_\ell)_{ic} = 1\right]. \tag{6}$$

Here A_ℓ is the sparse adjacency at level ℓ, $\mathcal{J}_{\omega,m}$ denotes m steps of damped Jacobi smoothing, and τ controls sparsification. Smoothing introduces a small overlap between adjacent aggregates, while the union with H_ℓ ensures that every node retains its original parent.

Construction result. Appendix Figure 7 shows that Balanced-Lloyd with smoothing produces concentrated cross-scale in-degree distributions at consecutive hierarchy transitions, thereby avoiding highly imbalanced aggregation neighborhoods.

2.3 Main Components

Downsampling and upsampling. The processor follows a V-cycle over the hierarchy constructed in Section 2.2. Transformer-style Pool and Unpool perform learnable transfer between adjacent scales. For a directed cross-scale edge (s,t), we use the geometric attribute $e_{st} = [x_s - x_t, \|x_s - x_t\|_2]$ and compute local edge-aware attention as

$$\begin{aligned} \alpha_{st} &= \underset{s \in \mathcal{N}(t)}{\mathrm{softmax}} \left(\frac{(W_Q h_t)^\top (W_K h_s + W_E e_{st})}{\sqrt{d_h}} \right), \\ m_t &= \sum_{s \in \mathcal{N}(t)} \alpha_{st} \left(W_V h_s + W_E e_{st} \right), \end{aligned} \tag{7}$$

where the softmax is restricted to the sparse neighborhood of t. In practice, Equation (7) is evaluated with multiple heads. Since attention is computed only over the sparse edges in B_ℓ, it costs $O(|B_\ell|)$, which becomes linear in the number of nodes when the cross-scale degree is bounded.

On the downward path, Pool treats coarse nodes as destinations and aggregates features from their fine-scale neighbors along B_ℓ. On the upward path, Unpool reverses the same sparse graph and reconstructs each fine node from its adjacent coarse representations. The two operators share the cross-scale support but use independent attention parameters, so Unpool also learns a feature-conditioned reconstruction. A transition through level $\ell + 1$ can be summarized as

$$h^{\ell+1} = \mathrm{MP}_{\ell+1}\big(\mathrm{Pool}_\ell(h^\ell; B_\ell)\big), \qquad \widetilde{h}^\ell = \mathrm{LN}\Big(h^{\ell,\mathrm{skip}} + \mathrm{Unpool}_\ell(\widetilde{h}^{\ell+1}; B_\ell^\top)\Big), \tag{8}$$

Same-level message passing. At intermediate levels, MP_ℓ propagates information over the corresponding coarse graph. At the coarsest level, we use Performer-style linear attention rather than standard quadratic self-attention. This provides domain-wide interaction among the coarse representations while retaining linear complexity in the number of coarse nodes. Same-scale message passing at the finest level is omitted by default; when needed, it is applied separately on a local radius graph. Together, the downward path expands the receptive field, while the upward path restores fine-scale features through learned cross-scale reconstruction and skip connections.

2.4 Memory-Efficient Large-Scale Execution

Destination-complete tiling. To bound the activation memory of a single sparse operator, we partition every edge set by destination nodes. Let $\mathcal{T}_t$ be the destination set of tile t. For fine-to-coarse Pool,

$$B_\ell^{(t)} = \{(i,c) \in B_\ell : c \in \mathcal{T}_t\}, \qquad B_\ell = \dot{\bigcup_t} B_\ell^{(t)}. \tag{9}$$

Table 2: Benchmark summary, ordered by representative mesh size. Mesh size denotes the number of spatial nodes or sampled surface elements per instance.

Dataset	Discretization	Mesh size	Predicted field
ShapeNet-Car	3D volume/surface	$\sim$3.7k surface	Velocity and surface pressure
SIMSHIFT Motor	2D motor mesh	$\sim$4.8k	Stress, strain, displacement
NACA0012	2D airfoil points	8,000	Euler flow field
Poisson-C-Sines	2D circular points	16,431	Poisson solution
DeepJEB	3D tetrahedral mesh	$O(10^4)$	von Mises stress
Ahmed Body	3D surface mesh	$\sim$69k surface	Pressure and wall shear stress
AirfRANS	2D volume/surface	$\sim$180k	Velocity, pressure, viscosity
NASA-CRM	3D aircraft surface	454,404	Pressure coefficient C_p
DrivAerML	3D surface samples	$\sim$8–9M full	Pressure and wall shear stress

All incoming edges of a destination node are kept in the same tile. Consequently, the softmax in Equation (7) is normalized over the complete neighborhood, and tiled attention is numerically equivalent to evaluating the full sparse edge set at once. The same destination-complete construction is used for reversed Unpool edges and coarse same-scale message passing.

Streamed fine-level execution. For million-node meshes, AMS-GNO additionally supports a streamed finest-level boundary. The first Encoder–Pool operation is fused and evaluated tile by tile, after which the assembled coarse representation is processed by the unchanged V-cycle from level one onward. During the upward pass, the finest-level encoder is recomputed for each Unpool tile, the saved raw inputs provide the local skip query, and Unpool is fused with the decoder. This execution avoids materializing a full finest-level hidden-state tensor while preserving complete attention neighborhoods and the original model parameters. Raw inputs, outputs, coarse features, and optimizer states remain resident; the streamed path specifically targets the finest-level activation bottleneck.

3 Experiments

3.1 Experiment Setup

Benchmarks. Table 2 summarizes benchmarks spanning two- and three-dimensional PDEs, point clouds and unstructured meshes, and sizes from a few thousand points to millions of surface cells. GAOT introduced the NACA0012 and Poisson-C-Sines tasks (Wen et al., 2026). SIMSHIFT provides the Motor structural benchmark (Setinek et al., 2025), and AirfRANS provides the airfoil RANS simulations (Bonnet et al., 2022). ShapeNet-Car traces to the simulated vehicle-flow dataset of Umetani & Bickel (2018); the Ahmed Body surface data used here are distributed by NVIDIA PhysicsNeMo (NVIDIA, 2025), while GINO established an earlier Ahmed Body neural-operator benchmark (Li et al., 2023b). DeepJEB originates from the jet-engine-bracket dataset of Hong et al. (2025); NASA-CRM uses the AASM Case-4 benchmark (Bekemeyer et al., 2025); and DrivAerML follows Ashton et al. (2024).

Baselines. The evaluated baselines cover complementary mechanisms for local aggregation, latent representation, and global communication. MeshGraphNet (Pfaff et al., 2020) provides a representative local message-passing baseline. GINO (Li et al., 2023b) and Geo-FNO (Li et al., 2023a) mediate irregular geometries through a regular latent representation or learned deformation, respectively. Transolver (Wu et al., 2024) uses learned physical-state slices for global Transformer interaction. GAOT (Wen et al., 2026) is a more recent multiscale graph–Transformer hybrid, coupling multiradius attentional graph encoders and decoders with a Transformer processor.

Implementation details. Unless a benchmark requires otherwise, AMS-GNO uses a width of 64, a three-level hierarchy, one V-cycle, and Performer-style linear global attention at the coarsest level. Balanced-Lloyd retains 10–20% of the nodes between adjacent levels and is followed by one Jacobi smoothing step; same-level processing at the finest resolution is disabled except when an explicit local operator is required. Most experiments train for 500 epochs with an initial learning rate of 10^{-3}, Adam or AdamW, and a cosine or one-cycle schedule; Poisson-C-Sines uses 1,000 epochs

Table 3: Comparison on PDE benchmarks (lower is better). "–" denotes the model cannot complete the task.

		Transformer-based	Graph-based		Frequency-based	
Dataset	AMS-GNO	Transolver	MeshGraphNet	GAOT	GINO	Geo-FNO
ShapeNet-Car	0.0510	**0.0487**	0.0568	0.0695	0.0599	0.2025
Motor Source	**0.0245**	0.0408	0.0283	0.0556	0.0806	0.8010
Motor Target	**0.0087**	0.0102	0.0146	0.0362	0.0790	0.9206
NACA0012	0.0138	**0.0124**	0.0173	0.0318	0.0451	0.1466
Poisson-C-Sines	**0.0246**	0.0923	0.3880	0.0310	0.0247	0.9990
DeepJEB	**0.2529**	0.2921	0.5088	0.3370	0.3834	–
Ahmed Body	**0.0564**	0.0748	0.1487	0.1031	0.1209	0.1360
AirfRANS	**0.0109**	0.0173	0.0601	0.0495	0.0799	0.0662
NASA-CRM	**0.0932**	0.0961	0.1480	0.3038	0.1239	–

and DeepJEB 300. Dataset-specific splits, losses, architecture, and optimization settings are detailed in Appendix B.

3.2 Main Results

Overall comparison. Table 3 compares performance across the benchmark settings. AMS-GNO has the lowest numerical error on Ahmed Body, both SIMSHIFT Motor settings, AirfRANS, NASA-CRM, Poisson-C-Sines, and DeepJEB, and ranks second on ShapeNet-Car and NACA0012.

Overall, AMS-GNO achieves the lowest error on seven of nine benchmark settings. Relative to the best competing result in each row, its error is lower by 13.4% on Motor Source, 14.7% on Motor Target, 0.4% on Poisson-C-Sines, 13.4% on DeepJEB, 24.6% on Ahmed Body, 37.0% on AirfRANS, and 3.0% on NASA-CRM. Its errors are 4.7% and 11.3% higher than Transolver's on ShapeNet-Car and NACA0012, respectively. These results are consistent with the intended role of AMS-GNO: retaining local structure at fine scales while moving long-range interaction to compressed coarse representations. Some cases are shown in Appendix C.

Ablation studies. Table 4 evaluates the learnable cross-scale operators and hierarchy construction. Replacing both Transformer operators with MLPs increases the error from 0.0246 to 0.1781. Moreover, retaining Transformer Unpool is substantially more effective than retaining Transformer Pool (0.0683 vs. 0.3165), showing that learnable coarse-to-fine reconstruction is particularly important in this setting.

For graph construction, Balanced-Lloyd consistently outperforms both standard Lloyd and random aggregation on volume and surface fields. Balanced-Lloyd gives the lowest error in both AirfRANS regions: 0.008156 on volume nodes and 0.002725 on surface nodes. Relative to standard Lloyd, these errors are lower by 14.6% and 27.5%, respectively; relative to random aggregation, they are lower by 34.4% and 18.5%. Figure 7 shows that Balanced-Lloyd also yields more concentrated cross-scale neighborhood sizes, reducing aggregation imbalance across coarse nodes. The matched Lloyd comparison is given in Appendix C.2.

Out-of-distribution generalization. On SIMSHIFT, we train on the source domain without target labels and evaluate on both source and target geometries. AMS-GNO obtains losses of 0.024456 and 0.008719, respectively, and ranks first among the compared methods in both settings. AMS-GNO achieves the lowest loss on both source and unseen target geometries without target-domain labels or adaptation, demonstrating strong generalization under geometry distribution shift.

Zero-shot spatial-resolution generalization. We train AMS-GNO on AirfRANS meshes retaining 12.5%, 50%, 75%, or 100% of the original nodes, then evaluate each model on the same full-resolution test meshes without fine-tuning. As shown in Figure 2, the model trained on 22,724 nodes (12.5%) can be evaluated on 181,794-node meshes, with a normalized total MSE of 0.063625. At 50% training resolution, the test MSE falls to 0.028397. Notably, the model trained on 136,345 nodes (75%) achieves a low test MSE of 0.012899 when evaluated directly on meshes with 181,794

Table 4: Ablation studies of cross-scale transfer and graph construction. Panel (a) reports relative L_2. Panel (b) reports the AirfRANS volume and surface errors; these regional values are not the main-comparison relative-L_2 score.

(a) Pool–Unpool operators on Poisson-C-Sines

Pool	Unpool	rel-L_2
Transformer	Transformer	**0.024573**
MLP	Transformer	0.068268
MLP	MLP	0.178123
Transformer	MLP	0.316488

(b) Aggregation strategies on AirfRANS

Aggregation	Volume error	Surface error
Balanced-Lloyd	**0.008156**	**0.002725**
Lloyd	0.009553	0.003758
Random	0.012430	0.003344

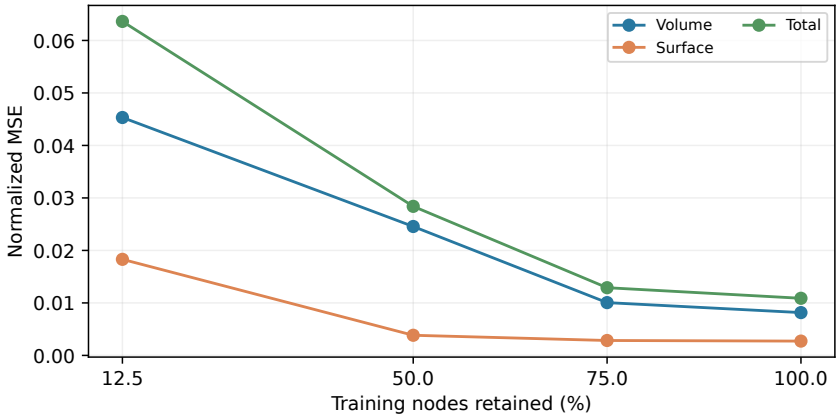

Figure 2: AirfRANS cross-resolution evaluation on full-resolution test meshes. Exact values are in Table 7.

nodes. These results demonstrate zero-shot transfer across spatial resolutions without changing the learned parameters. Further analysis is provided in Appendix A.1.

Scalability. Under bounded graph degree, cross-scale sparsity, and fixed model width, model execution scales linearly with the number of fine nodes. The DrivAerML measurements show median training time rising from 220 to 4,080 seconds per epoch and peak allocated memory from 764 to 13,605 MiB across the 5%–100% sampling range, as shown in Figure 3. The complexity assumptions and exact values are given in Appendix A.2 and Table 8, respectively.

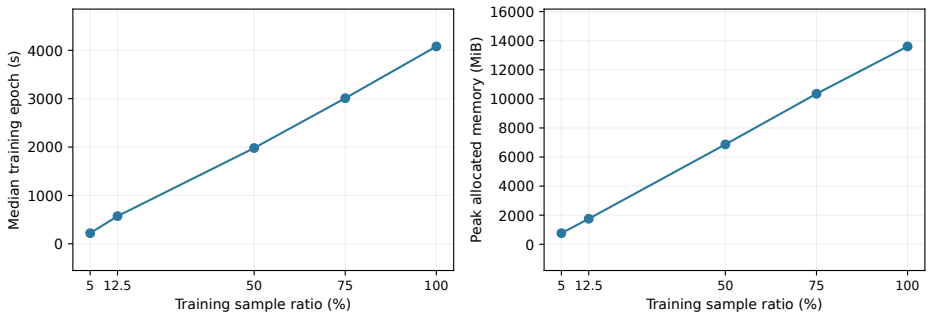

Figure 3: Measured DrivAerML median training time per epoch and peak allocated GPU memory versus retained surface-cell fraction. Exact values are in Table 8.

4 CONCLUSION

AMS-GNO combines local mesh structure with global interaction on a coarse graph. Balanced-Lloyd builds approximately balanced, topology-aware aggregates; Jacobi smoothing adds sparse overlap; learned Transformer Pool and Unpool transfer information across scales; and linear attention processes the coarsest representation. This construction keeps fine-scale communication sparse while adapting to irregular input meshes.

AMS-GNO achieves competitive accuracy across the PDE benchmarks, with its strongest matched comparisons on Ahmed Body, SIMSHIFT Motor, NASA-CRM, DeepJEB, and AirfRANS. The ablations support the cross-scale operators and balanced hierarchy. On AirfRANS, models trained with fewer nodes remain predictive on full-resolution meshes without fine-tuning. The DrivAerML measurements show near-linear growth in training time and memory up to 8,812,462 sampled surface cells. Together, these results indicate that topology-aware aggregation with learned bidirectional transfer can combine local fidelity, global communication, and practical scaling.

AI USE STATEMENT

Generative AI (OpenAI Codex) was used to assist in developing the conditional cross-resolution analysis in Appendix A.1. It was also used to draft and revise portions of the manuscript, assist with writing research code, search for relevant literature and submission requirements, and assist with LaTeX formatting. The authors are responsible for checking the proofs, citations, empirical claims, and AI-assisted text and code, and for the final content of this paper.

REPRODUCIBILITY STATEMENT

To facilitate reproduction, Sections 2.2, 2.3, and 2.4 describe the graph construction, model components, and large-graph execution procedure. Appendix B specifies the benchmark data and splits, preprocessing, evaluation metrics, model configurations, optimization settings, baseline protocols, ablations, and generalization and scalability protocols, together with the hardware used. Appendices A.1 and A.2 state the assumptions and provide the proofs and complexity analysis for the theoretical claims.

REFERENCES

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. *Advances in Neural Information Processing Systems*, 37:25152–25194, 2024.

Neil Ashton, Charles Mockett, Marian Fuchs, Louis Fliessbach, Hendrik Hetmann, Thilo Knacke, Norbert Schonwald, Vangelis Skaperdas, Grigoris Fotiadis, Astrid Walle, et al. Drivaerml: High-fidelity computational fluid dynamics dataset for road-car external aerodynamics. *arXiv preprint arXiv:2408.11969*, 2024.

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

Francesca Bartolucci, Emmanuel De Bézenac, Bogdan Raonic, Roberto Molinaro, Siddhartha Mishra, and Rima Alaifari. Representation equivalent neural operators: a framework for alias-free operator learning. *Advances in Neural Information Processing Systems*, 36:69661–69672, 2023.

Philipp Bekemeyer, Nathan Hariharan, Andrew M Wissink, and Jason Cornelius. Introduction of applied aerodynamics surrogate modeling benchmark cases. In *AIAA Scitech 2025 Forum*, pp. 0036, 2025.

Florent Bonnet, Jocelyn Mazari, Paola Cinnella, and Patrick Gallinari. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35:23463–23478, 2022.

Andrey Bryutkin, Jiahao Huang, Zhongying Deng, Guang Yang, Carola-Bibiane Schönlieb, and Angelica Aviles-Rivero. Hamlet: Graph transformer neural operator for partial differential equations. *arXiv preprint arXiv:2402.03541*, 2024.

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. PMLR, 2023.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. Multiscale meshgraphnets. *arXiv preprint arXiv:2210.00612*, 2022.

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pp. 1225–1234. PMLR, 2016.

Seongjun Hong, Yongmin Kwon, Dongju Shin, Jangseop Park, and Namwoo Kang. Deepjeb: 3d deep learning-based synthetic jet engine bracket dataset. *Journal of Mechanical Design*, 147(4): 041703, 2025.

Steeven Janny, Aurélien Beneteau, Madiha Nadri, Julie Digne, Nicolas Thome, and Christian Wolf. Eagle: Large-scale learning of turbulent fluid dynamics with mesh transformers. *arXiv preprint arXiv:2302.10803*, 2023.

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.

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

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

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in neural information processing systems*, 33:6755–6766, 2020c.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023a.

Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36:35836–35854, 2023b.

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.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for pdes on million-scale geometries. *arXiv preprint arXiv:2502.02414*, 2025.

Sepehr Mousavi, Shizheng Wen, Levi Lingsch, Maximilian Herde, Bogdan Raonic, and Siddhartha Mishra. Rigno: A graph-based framework for robust and accurate operator learning for pdes on arbitrary domains. *Advances in Neural Information Processing Systems*, 38:150039–150101, 2026.

Mohammad Amin Nabian, Chang Liu, Rishikesh Ranade, and Sanjay Choudhry. X-meshgraphnet: Scalable multi-scale graph neural networks for physics simulation. *arXiv preprint arXiv:2411.17164*, 2024.

NVIDIA. Nvidia physicsnemo ahmed body dataset, 2025. URL `https://catalog.ngc.nvidia.com/orgs/nvidia/physicsnemo/resources/physicsnemo_ahmed_body_dataset`. Version 1, accessed September 2026.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Paul Setinek, Gianluca Galletti, Thomas Gross, Dominik Schnürer, Johannes Brandstetter, and Werner Zellinger. Simshift: a benchmark for adapting neural surrogates to distribution shifts. *arXiv preprint arXiv:2506.12007*, 2025.

Nobuyuki Umetani and Bernd Bickel. Learning three-dimensional flow for interactive aerodynamic design. *ACM Transactions on Graphics (TOG)*, 37(4):1–10, 2018.

Petr Vaněk, Jan Mandel, and Marian Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996.

Shizheng Wen, Arsh Kumbhat, Levi Lingsch, Sepehr Mousavi, Yizhou Zhao, Praveen Chandrashekar, and Siddhartha Mishra. Geometry aware operator transformer as an efficient and accurate neural surrogate for pdes on arbitrary domains. *Advances in Neural Information Processing Systems*, 38:155423–155501, 2026.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

Tareq Zaman, Nicolas Nytko, Ali Taghibakhshi, Scott MacLachlan, Luke Olson, and Matthew West. Generalizing lloyd's algorithm for graph clustering. *SIAM Journal on Scientific Computing*, 46 (5):A2819–A2847, 2024.

Hang Zhou, Haixu Wu, Haonan Shangguan, Yuezhou Ma, Huikun Weng, Jianmin Wang, and Mingsheng Long. Transolver-3: Scaling up transformer solvers to industrial-scale geometries. *arXiv preprint arXiv:2602.04940*, 2026.

A Cross-Resolution Analysis and Complexity

A.1 Sparse Training and Cross-Resolution Evaluation

We compare two separately trained models: one trained with $m = rN$ spatial nodes per instance and one trained with all N nodes. The number of PDE training instances is unchanged. The questions are whether their predictions on the same N-node evaluation mesh are close, and whether the sparse-trained model gives similar predictions when evaluated on m and N nodes. Theorem A.1 gives a general multiscale-graph bound; Theorem A.2 shows how omitting finest-level mesh message passing and controlling aggregate balance tighten that certificate under matched sensitivity and hierarchy-perturbation bounds.

Comparison protocol and norms. For one physical input a, let $X_N = \{x_i\}_{i=1}^N$ and let $S_m \subseteq \{1, \dots, N\}$ be a fixed, parameter-independent subset of size $m \geq 1$. Write $r = m/N$ and $F_{\theta,N}(a)_i$ for the prediction at x_i; $F_{\theta,m}(a)_i$ is defined for $i \in S_m$. Both graphs use the same architecture, normalization statistics, deterministic graph/hierarchy construction rule, and fixed random features. At $m = N$ the constructions must agree. Weight sharing over nodes and edges makes the parameter space independent of m; nonempty aggregates and retained hard-parent edges make Pool/Unpool well-defined.

Let $\|v\|_N^2 = N^{-1} \sum_i \|v_i\|_2^2$. To compare arrays of different lengths, use nearest-retained-node extension E_m: $(E_m v)_i = v_{p(i)}$, where $p(i) \in S_m$ minimizes $\|x_i - x_{p(i)}\|_2$, and $p(i) = i$ on S_m. Ties are fixed independently of θ. This extension is used only to compare outputs, not to run the N-node model. Define

$$\begin{aligned} D_{\text{train}}(r; a) &= \|F_{\theta_m,N}(a) - F_{\theta_N,N}(a)\|_N, \\ D_{\text{infer}}(r; a) &= \|E_m F_{\theta_m,m}(a) - F_{\theta_m,N}(a)\|_N. \end{aligned} \tag{10}$$

Here θ_m and θ_N are the outputs of the two training runs. All constants below are uniform over the stated training and evaluation instances and admissible subsets. For varying instance sizes, replace $1 - r$ by the largest omitted fraction; no independence of N is claimed unless the assumptions hold uniformly in N.

Sufficient conditions. Fix a retention range $r \in [r_0, 1]$, a finite number T of training updates, and a convex parameter region Θ containing both trajectories. Use the following conditions.

1. At every $\theta \in \Theta$, the predictions at retained nodes and their parameter Jacobians satisfy, for a nonnegative, nonincreasing function q with $q(1) = 0$,

$$\begin{aligned} \max_{i \in S_m} \|F_{\theta,m}(a)_i - F_{\theta,N}(a)_i\|_2 &\leq C_0 q(r), \\ \max_{i \in S_m} \|\partial_\theta F_{\theta,m}(a)_i - \partial_\theta F_{\theta,N}(a)_i\|_{\text{op}} &\leq C_1 q(r). \end{aligned} \tag{11}$$

   The first bound holds on training and evaluation instances; the second is needed on training instances. These are graph-perturbation conditions at fixed parameters, not assumptions that two trained models are close. A sufficient module-level certificate is given below.

2. On training instances, nodal residuals have norm at most B and nodal parameter Jacobians have operator norm at most J, for both resolutions throughout Θ. The full-resolution objective has L-Lipschitz gradient there. On evaluation instances, the full-resolution nodal parameter Jacobians are bounded by J_{ev}. All required derivatives exist on this region.

3. On evaluation instances, the full-resolution predictions obey the discrete spatial regularity bound

$$\|F_{\theta,N}(a)_i - F_{\theta,N}(a)_j\|_2 \leq H_\nu \|x_i - x_j\|_2^\nu, \qquad 0 < \nu \leq 1, \tag{12}$$

   uniformly on Θ, and the point sets have diameter at most D_Ω.

4. Initially, the two training runs have identical parameters. They use the same training instances and update schedule. We first prove the result for gradient descent; an explicit extension to other stable update rules is stated below.

Lemma (a module-level stability certificate). Represent the network as a fixed finite composition of blocks, carrying skip features as part of each block state. For a state h and its total parameter derivative Z, lift a differentiable block $\Phi^j_{\theta,Q}$ to

$$\widehat{\Phi}^j_{\theta,Q}(h,Z) = \big(\Phi^j_{\theta,Q}(h),\, D_h\Phi^j_{\theta,Q}(h)Z + D_\theta\Phi^j_{\theta,Q}(h)\big), \qquad Q \in \{m, N\}. \tag{13}$$

Use the maximum nodal norm for features and the maximum nodal operator norm for Jacobians, with their sum for the lifted state. Suppose parameter-independent linear alignment maps A_j between corresponding full and reduced states are nonexpansive in these norms, A_0 exactly restricts the raw input, and A_H restricts the output to S_m. They act on both components of a lifted state. Suppose the reduced lifted block is Λ_j-Lipschitz and, on the full states and aligned states involved,

$$\|\widehat{\Phi}^j_{\theta,m}(A_{j-1}s) - A_j\widehat{\Phi}^j_{\theta,N}(s)\| \le c_j q(r). \tag{14}$$

Then Equation (11) holds with both C_0, C_1 at most

$$C_A = \sum_{j=1}^{H} c_j \prod_{k=j+1}^{H} \Lambda_k. \tag{15}$$

Proof. The input lifted states align exactly, including their zero parameter derivatives. If e_j is the difference between the reduced lifted state and the aligned full lifted state after block j, the triangle inequality gives $e_j \le \Lambda_j e_{j-1} + c_j q(r)$, with $e_0 = 0$. Iteration proves $e_H \le C_A q(r)$; each output component is bounded by this sum norm. □

More generally, replace $c_j q(r)$ in Equation (14) by an arbitrary nonnegative local defect $\delta_j(r)$. The same proof gives a joint output/Jacobian certificate

$$\mathfrak{E}(r) = \sum_{j=1}^{H} w_j \delta_j(r), \qquad w_j = \prod_{k=j+1}^{H} \Lambda_k. \tag{16}$$

Both retained-node bounds are at most $\mathfrak{E}(r)$. This form separates finest-level connectivity, cross-scale transfer, and all remaining modules.

This certificate requires stability of both values and derivatives. Small forward perturbations alone do not bound the change in training gradients. For intuition about the forward part, consider one normalized attention neighborhood. If deleted sources carry total original attention mass p, retained logits and values stay fixed, and all values have norm at most V, renormalizing over the retained sources changes the output by at most $2Vp$. Indeed, the original output is $(1-p)\bar{v}_{\mathrm{keep}} + p\bar{v}_{\mathrm{delete}}$, while the reduced output is $\bar{v}_{\mathrm{keep}}$. If logits lie in $[-A, A]$ and a fraction $f < 1$ of sources is removed, $p \le e^{2A} f$. The tangent part is also controlled for such a deletion if the retained functions are unchanged and $\|D_\theta v\|_{\mathrm{op}} \le V_1$, $\|D_\theta \ell\|_2 \le A_1$ for values v and logits ℓ. In fact, writing $\mathbb{E}_\pi$ for the attention-weighted average, the output derivative is

$$D_\theta o = \mathbb{E}_\pi[D_\theta v] + \mathbb{E}_\pi[v(D_\theta \ell)^\top] - \mathbb{E}_\pi[v]\,\mathbb{E}_\pi[D_\theta \ell]^\top. \tag{17}$$

Deletion changes the first two terms by at most $2V_1 p$ and $2VA_1 p$. Splitting the difference of the last product bounds it by $4VA_1 p$, hence the Jacobian perturbation is at most $(2V_1 + 6VA_1)p$. These estimates control one fixed-support deletion. Changes of representatives, new edges, and rebuilt Balanced-Lloyd hierarchies still require the aligned-state certificate in Equation (14); bounded degrees alone do not establish it.

Training objective and gradient perturbation. For n fixed PDE training instances with targets $u_i^{(b)}$, take

$$\begin{aligned} \mathcal{L}_N(\theta) &= \frac{1}{n}\sum_{b=1}^{n}\frac{1}{2N}\sum_{i=1}^{N}\|F_{\theta,N}(a^{(b)})_i - u_i^{(b)}\|_2^2, \\ \mathcal{L}_m(\theta) &= \frac{1}{n}\sum_{b=1}^{n}\frac{1}{2m}\sum_{i\in S_m^{(b)}}\|F_{\theta,m}(a^{(b)})_i - u_i^{(b)}\|_2^2. \end{aligned} \tag{18}$$

The notation uses equal N, m for clarity. Set

$$\Gamma(r) = (JC_0 + BC_1)q(r) + 2JB(1-r). \tag{19}$$

Then $\|\nabla\mathcal{L}_m(\theta) - \nabla\mathcal{L}_N(\theta)\|_2 \le \Gamma(r)$. To prove this, write a nodal gradient as $g_Q = J_Q^\top e_Q$, where e_Q is the residual. At a retained node,

$$\|J_m^\top e_m - J_N^\top e_N\|_2 \le J\|e_m - e_N\|_2 + B\|J_m - J_N\|_{\mathrm{op}} \le (JC_0 + BC_1)q(r). \tag{20}$$

Next, let $\bar{g}_S, \bar{g}_D$ be the averages of the full-graph nodal gradients on retained and deleted nodes. Their full average is $r\bar{g}_S + (1-r)\bar{g}_D$, so

$$\|\bar{g}_S - [r\bar{g}_S + (1-r)\bar{g}_D]\|_2 \le 2JB(1-r). \tag{21}$$

Averaging over instances proves the claim; at $r = 1$ the deletion term is zero and no deleted-node average is needed. Thus the two contributions are graph-induced changes at retained nodes and reweighting after node deletion. No random-sampling assumption is used.

Theorem A.1 (general multiscale-graph training and inference bound). Under the stated conditions, run $\theta_Q^{t+1} = \theta_Q^t - \eta_t\nabla\mathcal{L}_Q(\theta_Q^t)$ for $Q \in \{m, N\}$, $t = 0, \dots, T-1$, with $\eta_t \ge 0$. Define

$$S_T = \sum_{t=0}^{T-1} \eta_t \prod_{k=t+1}^{T-1} (1 + \eta_k L), \qquad h_m = \max_i \min_{j \in S_m} \|x_i - x_j\|_2. \tag{22}$$

For every evaluation instance, with $\theta_Q = \theta_Q^T$,

$$\begin{aligned} D_{\mathrm{train}}(r; a) &\le J_{\mathrm{ev}} S_T \Gamma(r), \\ D_{\mathrm{infer}}(r; a) &\le C_0 q(r) + H_\nu h_m^\nu \sqrt{1-r}, \\ \|E_m F_{\theta_m, m}(a) - F_{\theta_N, N}(a)\|_N &\le J_{\mathrm{ev}} S_T \Gamma(r) + C_0 q(r) + H_\nu h_m^\nu \sqrt{1-r}. \end{aligned} \tag{23}$$

Proof. Set $d_t = \|\theta_m^t - \theta_N^t\|_2$. Add and subtract $\nabla\mathcal{L}_N(\theta_m^t)$ in the update difference. Smoothness and the gradient perturbation estimate give

$$d_{t+1} \le (1 + \eta_t L) d_t + \eta_t \Gamma(r), \qquad d_0 = 0. \tag{24}$$

Unrolling yields $d_T \le S_T\Gamma(r)$. Integration of the full-resolution parameter Jacobian along the segment between the two final parameters gives $D_{\mathrm{train}} \le J_{\mathrm{ev}} d_T$; convexity of Θ keeps this segment in the region where the bound holds. This argument does not require convexity of the training loss.

For the inference comparison, for each full node i add and subtract $F_{\theta_m, N}(a)_{p(i)}$. The retained-node difference is at most $C_0 q(r)$, while the spatial difference is zero on S_m and at most $H_\nu h_m^\nu$ on its complement. The normalized Euclidean norm and the triangle inequality therefore give $D_{\mathrm{infer}} \le C_0 q(r) + H_\nu h_m^\nu \sqrt{(N-m)/N}$. A final triangle inequality through $F_{\theta_m, N}(a)$ proves the third bound. □

In particular, a multiscale graph model satisfying the module certificate (16) obeys the following bounds, without requiring a common power-law rate for its modules:

$$\begin{aligned} \mathcal{U}_{\mathrm{train}}(\mathfrak{E}; r) &= J_{\mathrm{ev}} S_T [(J + B)\mathfrak{E}(r) + 2JB(1-r)], \\ \mathcal{U}_{\mathrm{infer}}(\mathfrak{E}; r) &= \mathfrak{E}(r) + H_\nu h_m^\nu \sqrt{1-r}, \\ D_{\mathrm{train}} \le \mathcal{U}_{\mathrm{train}}, &\qquad D_{\mathrm{infer}} \le \mathcal{U}_{\mathrm{infer}}, \qquad D_{\mathrm{total}} \le \mathcal{U} := \mathcal{U}_{\mathrm{train}} + \mathcal{U}_{\mathrm{infer}}. \end{aligned} \tag{25}$$

Here D_{total} denotes the third difference in Equation (23). Replace the two retained-node bounds $C_0 q, C_1 q$ by $\mathfrak{E}$ in the preceding proof.

Theorem A.2 (a tighter certificate from the two architectural choices). Compare a reference multiscale model G with finest-level mesh message passing and an AMS configuration A without it. Each model is compared with its own full-node training run as in Theorem A.1; we do not compare the two architectures' predictions directly. Assume common upper bounds for $B, J, J_{\mathrm{ev}}, L, H_\nu$ and the same training schedule, input subsets, and extension. Pad the block sequences with identities as needed, and use common downstream amplification bounds w_j. For the Pool blocks $j \in \mathcal{P}$, impose the following verifiable conditions.

1. At the reference resolution, block j has n_j source nodes and K_j hard aggregates. Their minimum sizes are at least $a_j n_j / K_j$ for G and $b_j n_j / K_j$ for A, where $0 < a_j \le b_j \le 1$. Each Pool support contains its hard aggregate, as in Equation (6).

2. For each aligned target, an intermediate operator is obtained from its full Pool neighborhood by deleting at most $d_j(r)$ sources, keeping the remaining value and logit functions unchanged, and renormalizing. Set $\zeta_j(r) = d_j(r)/n_j$ and require $K_j\zeta_j(r) < a_j$. The actual reduced lifted operator differs from this intermediate operator by at most $\rho_j(r)$. This term includes changed representatives, new support edges, reassignments not represented by deletion, and alignment errors. Both models must satisfy the same upper envelope ρ_j; it is not supplied by balance alone.
3. Uniformly at these blocks, logits are in $[-\mathcal{A}_j, \mathcal{A}_j]$, values have norm at most V_j, and their total parameter derivatives and those of the logits are bounded by V_{1j} and A_{1j}. For multiple heads and subsequent projections, include the corresponding norm factors in the constants below. The remaining lifted blocks satisfy the module certificate.

Define $p_j(t;r) = \min\{1, e^{2\mathcal{A}_j}K_j\zeta_j(r)/t\}$ and $c_j = 2V_j + 2V_{1j} + 6V_jA_{1j}$. Let $\mathcal{F}$ contain the finest-level mesh message-passing blocks of G, with local lifted-defect bounds $\delta_j^{\mathrm{mesh}}(r)$, and put

$$\begin{aligned}
\mathfrak{F}(r) &= \sum_{j\in\mathcal{F}} w_j\delta_j^{\mathrm{mesh}}(r),\\
\mathfrak{R}(r) &= \sum_{j\notin\mathcal{P}\cup\mathcal{F}} w_j\delta_j(r) + \sum_{j\in\mathcal{P}} w_j\rho_j(r),\\
\mathfrak{E}_G(r) &= \mathfrak{R}(r) + \mathfrak{F}(r) + \sum_{j\in\mathcal{P}} w_jc_jp_j(a_j;r),\\
\mathfrak{E}_A(r) &= \mathfrak{R}(r) + \sum_{j\in\mathcal{P}} w_jc_jp_j(b_j;r).
\end{aligned} \tag{26}$$

The remaining-block defects in $\mathfrak{R}$ are common upper envelopes, including Unpool, coarse message passing, and global attention. Then these are valid certificates for G and A, respectively, and

$$\begin{aligned}
\Delta(r) &:= \mathfrak{E}_G(r) - \mathfrak{E}_A(r)\\
&= \mathfrak{F}(r) + \sum_{j\in\mathcal{P}} w_jc_j[p_j(a_j;r) - p_j(b_j;r)] \geq 0,\\
\mathcal{U}(\mathfrak{E}_G;r) - \mathcal{U}(\mathfrak{E}_A;r) &= [J_{\mathrm{ev}}S_T(J+B)+1]\Delta(r).
\end{aligned} \tag{27}$$

The corresponding improvements in the training and inference certificates are $J_{\mathrm{ev}}S_T(J+B)\Delta(r)$ and $\Delta(r)$. The total certificate is strictly tighter when $\Delta(r) > 0$.

Proof. A pointwise finest-level map ψ_θ, including a pointwise encoder or decoder, commutes with restriction R_m:

$$\psi_\theta(R_mh) = R_m\psi_\theta(h), \qquad \widehat{\psi}_\theta(R_m(h,Z)) = R_m\widehat{\psi}_\theta(h,Z). \tag{28}$$

Thus its local lifted defect is zero; omitting mesh message passing removes the direct contribution $\mathfrak{F}$. Pointwise maps may still amplify incoming errors, which is accounted for by w_j. This argument requires identical nodal inputs and pointwise or fixed normalization: connectivity-derived input features or resolution-dependent normalization require additional defects.

For Pool, a reference support contains at least tn_j/K_j sources, with $t = a_j$ or b_j. The deleted fraction is therefore at most $K_j\zeta_j/t < 1$. Bounded logits give deleted attention mass at most $p_j(t;r)$. The forward and tangent deletion estimates preceding Theorem A.1 give a summed lifted defect at most $c_jp_j(t;r)$. Adding ρ_j bounds the actual operator. Substitution in Equation (16) proves both certificates. Since $b_j \geq a_j$, $p_j(b_j;r) \leq p_j(a_j;r)$. Subtraction and Equation (25) prove the result. □

Corollary A.3 (a difficulty-dependent retention threshold). Suppose the certificate holds with $q(r) = (1-r)^\alpha$, $0 < \alpha \leq 1$, throughout $[r_0, 1]$. Put

$$\begin{aligned}
\beta &= \min\{\alpha, 1/2\},\\
K_{\mathrm{train}} &= J_{\mathrm{ev}}S_T(JC_0 + BC_1 + 2JB),\\
K_{\mathrm{infer}} &= C_0 + H_\nu D_\Omega^\nu, \qquad K = K_{\mathrm{train}} + K_{\mathrm{infer}}.
\end{aligned} \tag{29}$$

Then $D_{\text{train}} \le K_{\text{train}}(1-r)^{\alpha}$, $D_{\text{infer}} \le K_{\text{infer}}(1-r)^{\beta}$, and the third difference in Equation (23) is at most $K(1-r)^{\beta}$. Consequently, for $K > 0$ and any $\varepsilon > 0$, every admissible $r \ge r_{\varepsilon}$, where

$$r_{\varepsilon} = \max\left\{r_0,\, 1 - \left(\frac{\varepsilon}{2K}\right)^{1/\beta}\right\}, \tag{30}$$

makes all three prediction differences strictly less than ε. If $K = 0$, they vanish for every admissible r.

Proof. Write $s = 1 - r \in [0, 1]$. Since $s \le s^{\alpha}$, the training bound follows from Equation (19). Since $h_m \le D_{\Omega}$ and $s^{\alpha}, s^{1/2} \le s^{\beta}$, the other bounds follow from Theorem A.1. The threshold ensures $Ks^{\beta} \le \varepsilon/2 < \varepsilon$. □

The constants describe difficulty through spatial variation H_{ν}, graph/hierarchy sensitivity C_0, C_1, α, parameter sensitivity J, J_{ev}, residual scale B, and training amplification S_T. Larger constants require retention closer to one. The exponent $1/2$ comes from extending predictions to omitted nodes in a normalized L_2 norm; comparisons on retained nodes alone omit this extension term. With fixed N, ratios are discrete: a certified sub-full choice exists only if some $m < N$ satisfies $m/N \ge r_{\varepsilon}$. The threshold is sufficient, not an optimal or empirically estimated cutoff, and it does not imply that the actual error is monotone in r. Under Theorem A.2, if $\mathfrak{E}_X(r) \le M_X(1-r)^{\alpha}$ for $X \in \{A, G\}$ with $M_A \le M_G$ on a common admissible range, take $C_0 = C_1 = M_X$. Then $K_X = [J_{\text{ev}} S_T(J + B) + 1]M_X + 2J_{\text{ev}} S_T JB + H_{\nu} D_{\Omega}^{\nu}$. Consequently $K_A \le K_G$ and the sufficient thresholds satisfy $r_{\varepsilon,A} \le r_{\varepsilon,G}$, with the zero-constant case interpreted as above. This is a conditional improvement of the required retention certificate.

Other optimizers, regional losses, and residual discrepancies. The recurrence follows the coupled-trajectory stability approach used in optimization analysis (Hardt et al., 2016); the spatial-graph perturbation is established above. For an optimizer state z containing parameters and any moments, let $U_{Q,t}$ be its update map. If its norm controls the parameter norm and, on the states involved,

$$\begin{aligned} \|U_{N,t}(z) - U_{N,t}(z')\| &\le \kappa_t \|z - z'\|, \\ \|U_{m,t}(z) - U_{N,t}(z)\| &\le \lambda_t \Gamma(r), \end{aligned} \tag{31}$$

the proof is unchanged with $S_T = \sum_{t<T} \lambda_t \prod_{k>t} \kappa_k$. This covers matched minibatch sequences if the gradient bound holds for every batch. Adam/AdamW require bounds for their full state updates; the gradient-descent constants cannot be substituted without verification. A common initialization, matched randomness, update count, and checkpoint rule are needed. If initial states differ by d_0 or unmatched updates add defects b_t, the parameter bound instead has the additional term

$$R_T = d_0 \prod_{k<T} \kappa_k + \sum_{t<T} b_t \prod_{k>t} \kappa_k. \tag{32}$$

Prediction bounds then include $J_{\text{ev}} R_T$, which need not vanish as $r \to 1$.

For a weighted sum of regionwise losses, such as AirfRANS volume and surface MSE, apply the gradient argument separately to each region and sum with its loss weight (including any channel/half-square scaling). Each retained region must be nonempty. Use its own omitted fraction $1 - r_c$; a bound in the overall r alone additionally requires $1 - r_c \le C_c(1 - r)$. The inference argument likewise applies within each region, with region-preserving extension for regionwise scores.

A.2 Time and Space Complexity

Let $N_{\ell} = |V_{\ell}|$, $M_{\ell} = |E_{\ell}|$, and $S_{\ell} = |B_{\ell}|$ for $\ell = 0, \ldots, L-1$, with B_{ℓ} defined for $\ell < L - 1$. Let C_{ℓ} be the channel width, d_e the edge-attribute dimension, R the fixed Performer feature dimension, K the number of V-cycles, and P the total number of trainable parameters. The implementation uses $C_{\ell} = 2^{\ell} C_0$; the bounds below retain C_{ℓ} explicitly. We separate one-time graph construction from model execution because the measured epoch times use cached hierarchies.

Graph construction. For Balanced-Lloyd clustering, Theorem 3.6 of Zaman et al. (2024) gives $T_{\text{BL},\ell} = O(N_{\ell})$ when graph degree is bounded, $\Theta(N_{\ell})$ initial centers yield uniformly bounded cluster sizes, and both iteration limits are independent of N_{ℓ}. With a fixed number of Jacobi steps,

bounded nonzeros per assignment row, and linear-time sparse accumulation, smoothing and coarse-graph assembly also cost $O(N_\ell)$ on bounded-degree graphs. Thus, under these conditions and geometric coarsening, hierarchy construction costs $T_{\text{preprocess}} = O(\sum_{\ell=0}^{L-2} N_\ell) = O(N_0)$. The cached hierarchy occupies

$$D_{\text{graph}} = O\left(\sum_{\ell=0}^{L-1}(N_\ell + M_\ell) + \sum_{\ell=0}^{L-2} S_\ell\right), \tag{33}$$

excluding temporary construction workspace.

Forward and backward work. For one Pool or Unpool at transition ℓ, nodewise query/key/value projections cost $O((N_\ell + N_{\ell+1})C_\ell^2)$, while sparse edge projections, logits, softmax, and weighted sums cost $O(S_\ell C_\ell(d_e+1))$. The two directions only change the constant. A fixed-depth edge-MLP message-passing block costs $O((N_\ell + M_\ell)C_\ell^2)$ as a conservative dense-channel bound. Performer on the coarsest level costs $O(N_{L-1}(C_{L-1}^2 + RC_{L-1}))$ for projections and linear attention, rather than $O(N_{L-1}^2)$. Pointwise channel changes and the encoder/decoder add $O(\sum_\ell N_\ell C_\ell^2 + N_0 C_0(d + d_a + d_u))$, where d is the spatial dimension. Hence a forward pass has the bound

$$\begin{aligned} T_{\text{fwd}} = O\Bigg(& K\sum_{\ell=0}^{L-2}\left[(N_\ell + N_{\ell+1})C_\ell^2 + S_\ell C_\ell(d_e+1)\right] + K\sum_{\ell=1}^{L-2}(N_\ell + M_\ell)C_\ell^2 \\ & + KN_{L-1}(C_{L-1}^2 + RC_{L-1}) + N_0 C_0(d + d_a + d_u)\Bigg). \end{aligned} \tag{34}$$

Backpropagation has the same asymptotic order, with a larger constant; checkpointed tiles recompute parts of the forward pass but do not change that order. If $N_{\ell+1} \le rN_\ell$ for fixed $r < 1$, $M_\ell \le d_g N_\ell$, and $S_\ell \le qN_\ell$ with fixed graph-degree bound d_g and sparsity bound q, then for fixed L, K, C_ℓ, R, d_e,

$$T_{\text{train step}} = O(N_0), \qquad T_{\text{total}} = T_{\text{preprocess}} + J\,T_{\text{train step}} \tag{35}$$

for J optimization steps. This is a linear *model-execution* result under sparse-degree assumptions, not a bound on uncached graph construction.

Stored states and activation memory. In an ordinary materialized forward pass, node states require $O(\sum_\ell N_\ell C_\ell)$ memory. Training also retains per-edge intermediates for reverse-mode differentiation, giving the conservative bound

$$\begin{aligned} S_{\text{ordinary}} = O\Big(& P + D_{\text{graph}} + N_0(d + d_a + d_u) + RC_{L-1} \\ & + K\sum_{\ell=0}^{L-1} N_\ell C_\ell + K\sum_{\ell=1}^{L-2} M_\ell C_\ell + K\sum_{\ell=0}^{L-2} S_\ell C_\ell\Big). \end{aligned} \tag{36}$$

The $O(P)$ term includes gradients and optimizer state up to constant factors. Under bounded degrees and fixed widths this is $O(N_0 + P)$, but its edge-activation constant can be substantial.

For destination-complete tiles of at most T destination nodes, let e_t be the number of incoming edges and u_t the number of distinct source nodes in tile t. Local attention needs $O((T + u_t + e_t)C_\ell)$ temporary feature storage, while every destination sees all its incoming edges, so tiling leaves the operator unchanged. Tiling alone does not eliminate a materialized $N_0 C_0$ fine-level state. With the implemented single-V-cycle, checkpointed streamed fine boundary, the encoder and first Pool are fused per tile, and fine-level features are recomputed during Unpool. The fine-level hidden state is then replaced by raw inputs/outputs and a tile-sized working set; an indicative peak bound is

$$\begin{aligned} S_{\text{stream}} = O\Big(& P + D_{\text{coarse}} + N_0(d + d_a + d_u) + RC_{L-1} \\ & + \sum_{\ell=1}^{L-1} N_\ell C_\ell + \max_t (T + u_t + e_t)C_0\Big), \end{aligned} \tag{37}$$

where D_{coarse} covers resident coarse graphs and only the current fine tile is held on the GPU; the full tile plan and hierarchy may remain cached on the CPU. The peak still grows with raw inputs, outputs, and coarse states. In particular, tiling reduces temporary edge and fine-hidden activations, *not* total memory to a constant independent of N_0.

B EXPERIMENTAL DETAILS

This appendix details the data splits, preprocessing, optimization settings, and evaluation protocols. Parts of our implementation are adapted from Neural-Solver-Library (Wu et al., 2024). Unless stated otherwise, normalization statistics are computed from the corresponding training split. NASA-CRM baseline values follow the protocols of the cited papers.

Hardware. All experiments we conducted were run on a single NVIDIA RTX 5090 GPU with 32 GB of memory.

B.1 BENCHMARKS AND DATA SPLITS

NACA0012. Introduced in GAOT (Wen et al., 2026), this benchmark contains steady two-dimensional flow fields around the NACA0012 airfoil on 8,000-point samples. Each sample is represented by two spatial coordinates and three input channels, with one scalar solution field as the target. We use 5,000 training and 200 test samples with split seed 42.

Poisson-C-Sines. Also introduced in GAOT (Wen et al., 2026), this dataset consists of solutions of a two-dimensional Poisson equation with oscillatory forcing on 16,431-point samples. The split contains 2,048 training, 128 validation, and 256 test samples.

SIMSHIFT Motor. SIMSHIFT (Setinek et al., 2025) is a finite-element benchmark for structural response under geometric distribution shift. We use its medium-difficulty motor task, which predicts stress, strain, and displacement fields from the initial mesh, design parameters, and material. The split contains 1,131 training, 377 validation, and 206 test examples. Source-domain training data are used for optimization; source and target test geometries are evaluated separately, without target-domain adaptation.

AirfRANS. We use the full AirfRANS task (Bonnet et al., 2022) to predict two velocity components, pressure, and turbulent viscosity on two-dimensional volume and surface meshes. We use 720 training, 80 validation, and 200 test geometries. Following the setup of Transolver, tensor-based models are trained on up to 32,000 sampled points per instance due to GPU memory constraints, while graph-based models process the full mesh.

ShapeNet-Car. The ShapeNet-Car task traces to the vehicle-flow dataset of Umetani & Bickel (2018). It predicts a three-dimensional velocity field in the fluid volume and pressure on the vehicle surface. Node coordinates, the signed-distance field, and surface normals form the input. We use 700 training, 89 validation, and 100 test geometries. A typical surface has approximately 3,700 nodes.

Ahmed Body. The Ahmed Body data used here are distributed by NVIDIA PhysicsNeMo (NVIDIA, 2025) as three-dimensional vehicle surfaces represented by polygonal VTP meshes. Surface coordinates, normals, and operating conditions are mapped to pressure and three wall-shear components. We use the official 408/50/50 train/validation/test split. Non-coordinate inputs and targets are standardized using training-set statistics; coordinates remain in their physical representation.

DeepJEB. DeepJEB (Hong et al., 2025) contains irregular three-dimensional structural meshes subjected to a vertical load. The target is the von Mises stress field on volume nodes. We use 1,924 training and 214 test cases, split with seed 42. Normalization is enabled.

NASA-CRM. We use the AASM Case-4 surface dataset (Bekemeyer et al., 2025), whose meshes contain approximately 454,000 nodes. Coordinates and surface normals, together with six operating-condition variables, are used to predict the pressure coefficient. The split contains 105 training and 44 test cases. Coordinates are mapped to a unit box and all remaining statistics are estimated from the training split.

DrivAerML. DrivAerML (Ashton et al., 2024) is used to evaluate large-mesh scalability. Surface-cell centers and normals are mapped to pressure and three wall-shear components. We follow the official scarce-data split of 67 training, 34 validation, and 50 test cases. The full surface contains approximately 8–9 million cells per case. For the scaling runs, nested 5%, 12.5%, 50%, 75%, and 100% subsets are generated from a fixed seed-0 permutation for each case.

B.2 Metrics and Losses

Table 5 lists the training objectives and evaluation metrics. The main AirfRANS comparison and cross-resolution study use different metrics.

Table 5: Training objectives and reported metrics. "Standardized" means that targets are transformed using training-split statistics. The main AirfRANS comparison and cross-resolution study use different metrics.

Task	Training objective	Reported metric
NACA0012; Poisson-C-Sines	Relative L_2	Relative L_2
SIMSHIFT Motor	Field MSE + deformation MSE	Same composite loss (source/target)
AirfRANS, main comparison	Relative L_2	Relative L_2
AirfRANS, cross-resolution	Volume MSE + surface MSE	Normalized total MSE
ShapeNet-Car	Velocity MSE $+ 0.5$ surface-pressure MSE	Mean of velocity and pressure relative L_2
Ahmed Body	Relative L_2	Relative L_2
DeepJEB	Standardized stress MSE	Decoded stress relative L_2
NASA-CRM	Normalized C_p MSE	Physical-scale C_p relative L_2
DrivAerML	Standardized four-channel MSE	Epoch time and peak memory

Relative L_2. For a predicted field $\widehat{U}$ and target U on one case, we use

$$\operatorname{rel} L_2(\widehat{U}, U) = \frac{\|\widehat{U} - U\|_2}{\|U\|_2}. \tag{38}$$

Scores are averaged over test cases unless a benchmark specifies otherwise. For DeepJEB, predictions are decoded before this calculation; for NASA-CRM, the reported C_p error is likewise measured on the original target scale. The NASA-CRM literature entries in Table 3 are converted from the results in Transolver-3 (Zhou et al., 2026) to fractions.

Mean squared error and composite losses. For n spatial locations and d_u output channels, $\mathrm{MSE}(\widehat{U}, U) = (nd_u)^{-1} \sum_{i,c} (\widehat{U}_{i,c} - U_{i,c})^2$. "Standardized MSE" applies this formula after target normalization. SIMSHIFT adds the field and nodal-deformation MSE terms. ShapeNet-Car adds full-volume velocity MSE to one-half of surface-pressure MSE during training, whereas its reported score averages the corresponding two relative-L_2 errors; the two should not be equated.

AirfRANS regional MSE. The cross-resolution study evaluates standardized volume and surface errors separately, preventing the much larger volume-node set from overwhelming the surface contribution. For standardized residuals $e_{i,c} = \widehat{U}_{i,c} - U_{i,c}$, volume-node set $\mathcal{V}$, and surface-node set $\mathcal{S}$, its objective is

$$\mathcal{L}_{\mathrm{Air}} = \frac{1}{d_u} \sum_{c=1}^{d_u} \left(\frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} e_{i,c}^2 + \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} e_{i,c}^2 \right). \tag{39}$$

The normalized total MSE in Table 7 is this sum, averaged over cases in the evaluation set indicated in the caption. This metric is not comparable numerically with the relative-L_2 main AirfRANS row. DrivAerML uses standardized four-channel MSE for training, but its reported scaling quantities are training time and peak allocated memory rather than a prediction-error score.

B.3 AMS-GNO IMPLEMENTATION

AMS-GNO uses `level=1`, `unet_depth=3`, Balanced-Lloyd aggregation, and Transformer Pool and Unpool with eight heads. Table 6 gives the dataset-specific settings. The `n_layers` column denotes the implementation parameter rather than the number of hierarchy levels. "Off" means `lms_use_gno=false`.

Table 6: AMS-GNO configurations by benchmark.

Dataset	Finest graph	Width	n_{layers}	Ratio	Finest GNO	Batch
NACA0012	8-NN	64	1	0.20	on, $r = 0.05$	10
Poisson-C-Sines	8-NN	64	1	0.20	off	10
SIMSHIFT	FEM edges	64	1	0.20	off	1
ShapeNet-Car	VTK mesh edges	128	1	0.10	on, $r = 0.05$	1
AirfRANS	VTU mesh edges	64	1	0.10	off	1
Ahmed Body	VTP surface edges	256	1	0.10	off	1
DeepJEB	volume mesh	64	0	0.10	off	1
NASA-CRM	8-NN	64	0	0.10	off	1
DrivAerML	8-NN	64	0	0.10	off	1

The DrivAerML scaling experiments use CUDA Balanced-Lloyd with coarsening ratio 0.1. Streaming is performed with destination-complete tiles of size 2,048, and activation checkpointing is enabled. Hierarchies are constructed and cached before optimization; consequently, graph preprocessing time is not included in the reported per-epoch measurements.

B.4 TRAINING AND OPTIMIZATION SETTINGS

The initial learning rate is 10^{-3} throughout. NACA0012, Poisson-C-Sines, and SIMSHIFT use AdamW with weight decay 10^{-5} and cosine decay for 500, 1,000, and 500 epochs, respectively; the SIMSHIFT AMS-GNO setting also specifies dropout 0.1. AirfRANS and Ahmed Body use Adam without weight decay and a OneCycle schedule for 500 epochs. ShapeNet-Car uses AdamW with weight decay 10^{-5} and a OneCycle schedule for its AMS-GNO run. DeepJEB uses AdamW with weight decay 10^{-2} and cosine decay for *300* epochs. NASA-CRM uses AdamW with weight decay 10^{-2}, batch size 1, and cosine decay for 500 epochs. Its AMS-GNO configuration uses global attention at the coarsest level and zero dropout; training minimizes normalized C_p MSE. DrivAerML uses up to 500 epochs with AdamW, weight decay 10^{-2}, cosine decay, and gradient clipping at 1. All AMS-GNO experiments use FP32 training.

B.5 BASELINE IMPLEMENTATIONS

The baselines represent complementary operator families: Transolver uses learned physical-state tokens, Geo-FNO applies Fourier layers on a geometric representation, GINO maps irregular points through a latent operator grid, MeshGraphNet performs local message passing on a graph, and GAOT uses geometry-aware attention. Their settings are detailed below.

ShapeNet-Car. Transolver uses width 256, eight layers, and eight heads. GINO uses width 32 and four layers. GAOT uses a $32 \times 16 \times 16$ latent grid with radii $(0.2, 0.4)$. The baselines use OneCycle and batch size 1.

SIMSHIFT Motor. Transolver uses width 128 and four layers. We additionally tested width 256, but found that increasing the width degraded its performance; therefore, we report the better-performing width-128 configuration. MeshGraphNet uses width 128 and eight layers on the FEM edges. GINO uses width 256 and four layers. Geo-FNO uses width 256 and three layers. GAOT uses a 64×64 latent grid with radii $(0.05, 0.1)$.

NACA0012. Transolver uses width 256, eight layers, eight heads, and batch size 10; normalization is disabled for this model. MeshGraphNet uses width 128, eight layers, and batch size 10. GINO

uses width 256, four layers, and batch size 10. Geo-FNO uses width 256, three layers, and batch size 10. GAOT uses a 64×64 latent grid, radii $(0.05, 0.1)$, and batch size 10.

Poisson-C-Sines. Transolver uses width 256, eight layers, eight heads, and batch size 10. MeshGraphNet uses width 128, eight layers, and batch size 10. GINO uses width 256, four layers, and batch size 10. Geo-FNO uses width 256, three layers, and batch size 10. GAOT uses a 64×64 latent grid with radii $(0.022, 0.033, 0.044)$, Transformer width 256, five layers, and batch size 10.

DeepJEB. Our Transolver configuration uses width 64, eight layers, and eight heads.

Ahmed Body. Transolver uses width 256 and eight layers. MeshGraphNet uses width 128 and eight layers. GINO uses width 64 and four layers. Geo-FNO uses width 64 and three layers. GAOT uses a $32 \times 16 \times 16$ latent grid with radii $(0.2, 0.4)$. The comparisons use batch size 1.

AirfRANS. We use the 720/80/200 split. Transolver uses width 256, eight layers, and eight heads. MeshGraphNet uses width 128 and eight layers on the VTU edges. GINO uses width 256 and four layers. GAOT uses a 64×64 latent grid with radii $(0.05, 0.1)$.

NASA-CRM. The GINO, Transolver, MeshGraphNet, and GAOT results in Table 3 come from Transolver-3 (Zhou et al., 2026).

B.6 Ablation Protocols

Pool and Unpool. The four cross-scale transfer variants in Table 4 are evaluated on the same Poisson-C-Sines split, objective, three-level hierarchy, and Balanced-Lloyd ratio of 0.2. Each variant is trained for 1,000 epochs.

Graph construction. The aggregation comparison is performed on AirfRANS while holding the physical VTU graph, model width, three-level architecture, optimizer, and evaluation protocol fixed. Only the aggregation method changes among Random, standard Lloyd, and Balanced-Lloyd; each uses a coarsening ratio of 0.1. For Random, at each coarsening step we sample $\max(1, \lfloor 0.1N \rfloor)$ centers without replacement from the current level's N nodes. Each node is then assigned to its nearest sampled center in Euclidean distance, forming Voronoi aggregates; sampled centers remain in their own aggregates. The three-level hierarchy applies this procedure twice, sampling the second set of centers from the first coarse level. A fixed seed of 0 is used to construct each hierarchy, which is cached and not resampled every epoch. Table 4 reports volume and surface errors separately for each variant. This comparison isolates the effect of the aggregation method from the remainder of the model.

B.7 Generalization and Scalability Protocols

Geometry distribution shift. For SIMSHIFT, optimization uses only the labeled source-domain training set. Model selection uses the source validation split, after which the same checkpoint is evaluated on both source and target test geometries. There is no target-domain fine-tuning or unsupervised domain adaptation.

Spatial-resolution generalization. AirfRANS training meshes retain 12.5%, 50%, 75%, or 100% of their original nodes using a fixed seed-0 nested permutation; the validation and test meshes retain 100% resolution. Each discretization receives a newly constructed AMS-GNO hierarchy; the other model and training settings remain fixed across resolutions. Training statistics are computed from the original full training split and reused at every resolution.

Table 7: AirfRANS evaluation on full-resolution test meshes without fine-tuning. Volume and surface MSEs sum to the normalized total MSE.

Train res.	Train nodes	Volume MSE	Surface MSE	Total MSE
12.5%	22,724	0.045322	0.018303	0.063625
50%	90,897	0.024560	0.003837	0.028397
75%	136,345	0.010057	0.002842	0.012899
100%	181,794	0.008157	0.002726	0.010883

Large-mesh scaling. The DrivAerML runs use the nested 5%, 12.5%, 50%, 75%, and 100% surface-cell subsets described in Section B.1 for training, validation, and testing. Figure 3 visualizes the measured scaling; exact values are in Table 8. The rise in time and memory is approximately proportional to the retained fraction. The representative training graphs contain 440,623, 1,101,557, 4,406,231, 6,609,346, and 8,812,462 surface cells at the five fractions, respectively. Input and target statistics are computed from the training subset at each resolution. Per-epoch time is the median over the training window, excluding validation, and memory is the maximum allocated GPU memory over the same window. These measurements evaluate the training step with cached hierarchies, destination-complete tiling, and activation checkpointing.

Table 8: Scaling behavior on DrivAerML with a fixed tile size of 2,048. Fractions denote retained surface cells. Time and memory are measured values for all five runs.

Sampled cells	Fraction	Time / epoch (s)	Peak memory (MiB)
440,623	5%	220	764
1,101,557	12.5%	572	1,754
4,406,231	50%	1,980	6,870
6,609,346	75%	3,010	10,350
8,812,462	100%	4,080	13,605

C VISUALIZATIONS

We first visualize the construction of the multiscale hierarchy, compare Balanced-Lloyd with Lloyd, and inspect the learned cross-scale weights. We then present field predictions and a batch of AirfRANS error maps across multiple examples.

C.1 MULTISCALE HIERARCHIES

Figures 4 and 5 show the complete spatial extent at three consecutive levels. Balanced-Lloyd forms local aggregates on each graph and selects their representatives for the next level. The resulting hierarchy reduces the number of nodes while retaining the geometry across levels.

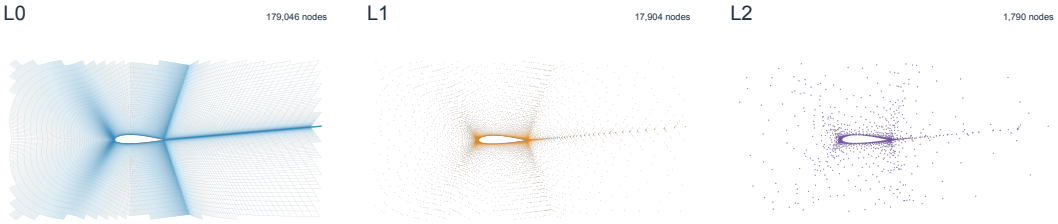

Figure 4: AirfRANS full-domain hierarchy, from L_0 to L_2. All three panels use the same spatial extent, showing the fine mesh and the two recursively constructed coarse graphs.

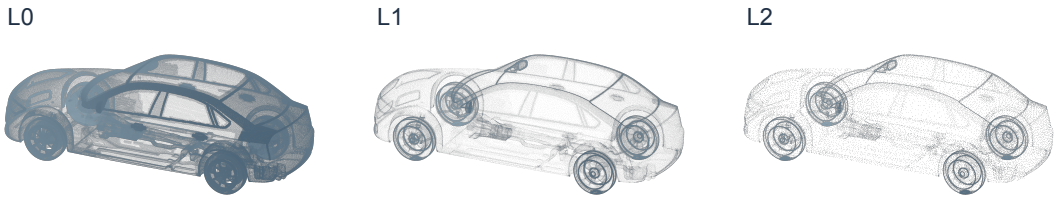

Figure 5: Full DrivAerML surface at L_0, L_1, and L_2, with 8,140,177, 814,017, and 81,401 nodes, respectively. The panels share the same viewpoint and scale. The visualization shows the complete surface at each level; edges are omitted for readability.

C.2 BALANCED-LLOYD AND LLOYD

Figure 6 first illustrates the local cross-scale connections produced by the two constructions. The enlarged patches expose how fine nodes connect to representatives at successive levels. The degree statistics below quantify the variation in neighborhood size across the complete graphs.

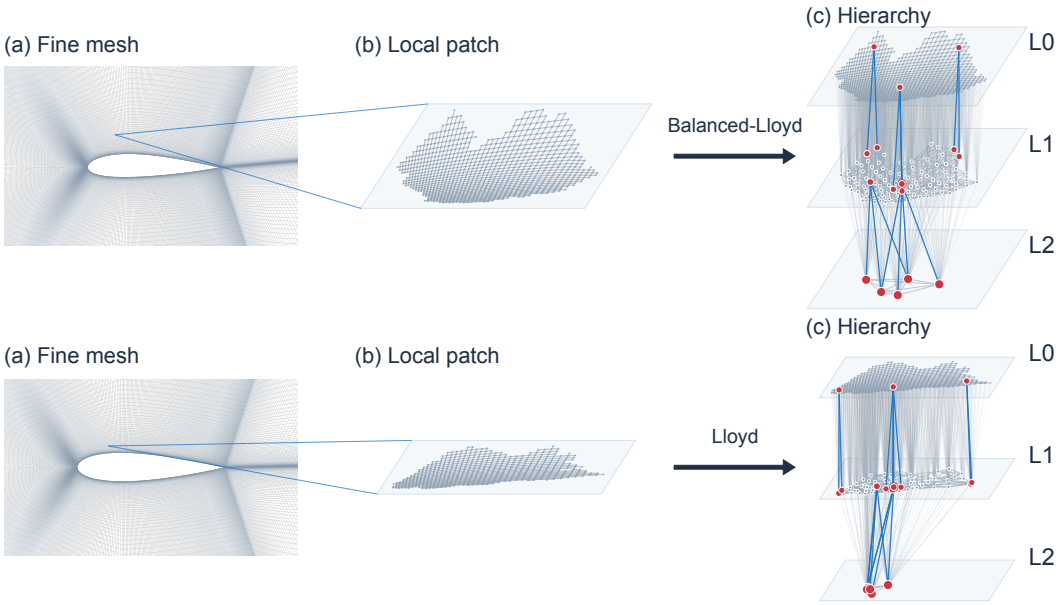

Figure 6: Local AirfRANS hierarchy examples for Balanced-Lloyd (top) and Lloyd (bottom). Each row shows the fine mesh, an enlarged patch, and connections across three levels. Red nodes and blue edges highlight selected cross-scale relations; the remaining nodes and edges are gray.

In-degree and aggregation balance. For directed Pool edges from fine to coarse nodes, a coarse node's in-degree counts the fine nodes in its aggregation neighborhood. Table 9 and Figure 7 show that Balanced-Lloyd concentrates this distribution. At $L_0 \to L_1$, the 95th percentile (P95) decreases from 35 to 25 and the maximum from 65 to 29. At $L_1 \to L_2$, P95 decreases from 55 to 37 and the maximum from 81 to 43. The mean degrees remain close, so the principal change is a reduction in the upper tail of neighborhood sizes.

Table 9: AirfRANS Pool degree statistics. In-degree is measured at coarse nodes; out-degree at fine nodes. Mean out-degree equals the number of Pool edges divided by the number of fine nodes.

		In-degree			Out-degree	
Transition	Construction	Mean	P95	Max.	Mean	Edges
$L_0 \to L_1$	Balanced-Lloyd	21.98	25	29	2.198	393,555
	Lloyd	22.25	35	65	2.225	398,375
$L_1 \to L_2$	Balanced-Lloyd	31.57	37	43	3.157	56,511
	Lloyd	32.50	55	81	3.249	58,167

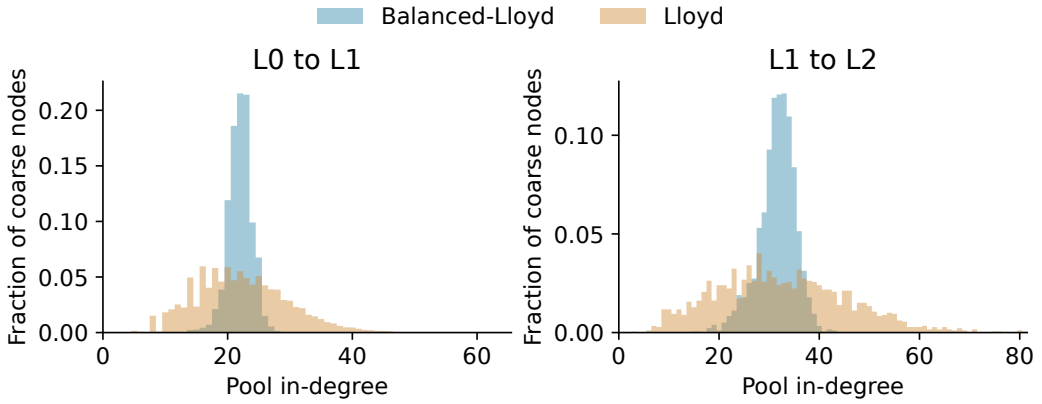

Figure 7: Pool in-degree distributions for Balanced-Lloyd and Lloyd, at $L_0 \to L_1$ (left) and $L_1 \to L_2$ (right). Each distribution is normalized by its number of coarse nodes.

Out-degree and sparse overlap. A fine node's Pool out-degree counts the coarse nodes that receive its features. Figure 8 compares the average out-degree at the two transitions. Balanced-Lloyd has means of 2.198 and 3.157, close to Lloyd's 2.225 and 3.249. Thus the reduction in coarse-neighborhood imbalance is accompanied by a similarly sparse average fan-out. For Lloyd, the out-degree P95 values are 3 and 5, respectively. Reversing the cross-scale edges exchanges Pool in-degree with Unpool out-degree, and Pool out-degree with Unpool in-degree.

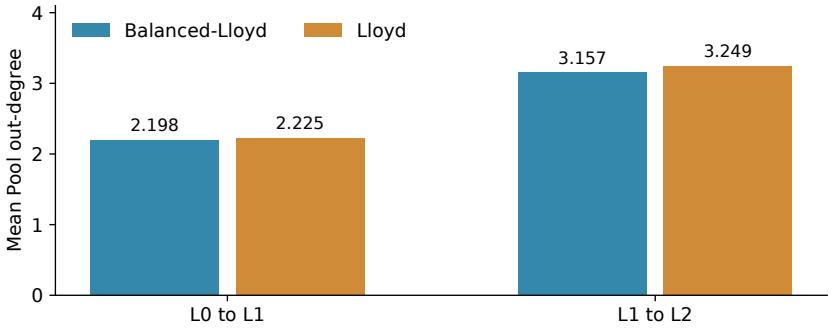

Figure 8: Mean Pool out-degree for Balanced-Lloyd and Lloyd at the two hierarchy transitions.

C.3 Cross-Scale Transfer Weights

Figure 9 expands the Pool and Unpool weight maps used in the architecture illustration. With fine nodes ordered by cluster membership, the strongest weights concentrate around a narrow diagonal band. This structure reflects local aggregation and reconstruction; the neighboring off-diagonal connections arise from overlap between aggregates. The sparse support restricts where information can flow, while attention learns the relative weights within that support.

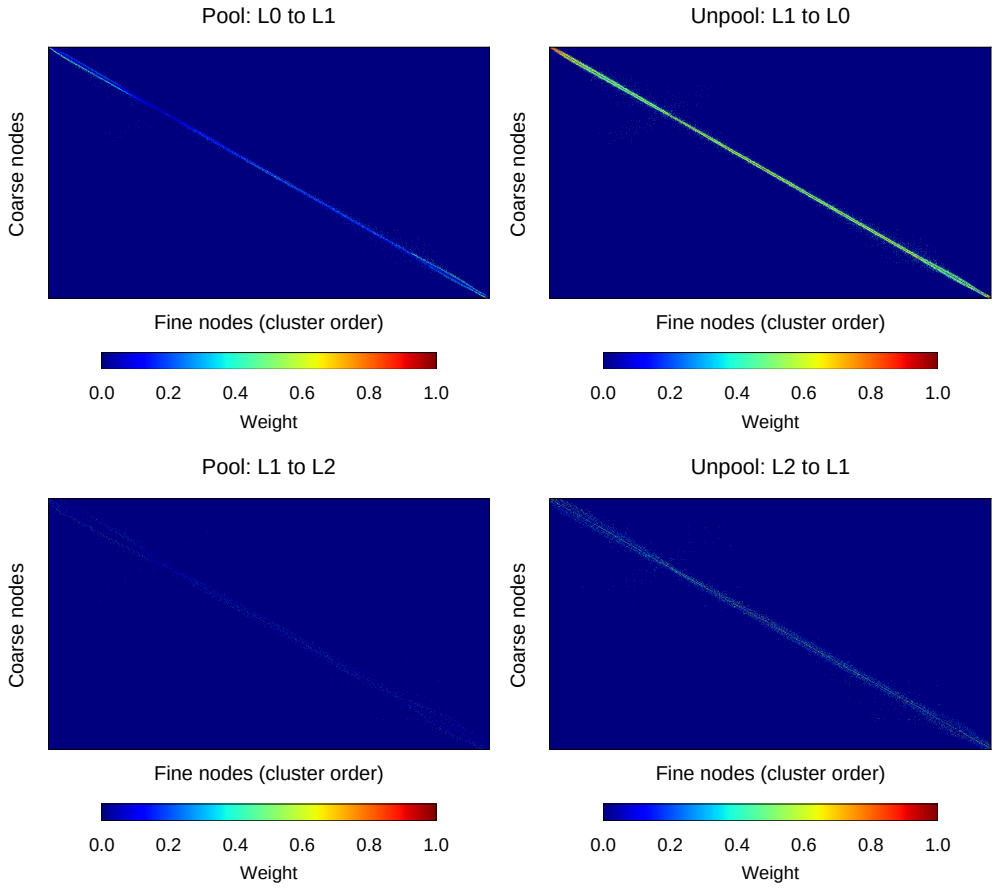

Figure 9: AirfRANS cross-scale attention weights, averaged over heads, for Pool (left) and Unpool (right). The top row corresponds to L_0 and L_1; the bottom row to L_1 and L_2. Fine nodes are ordered by cluster membership. Unpool matrices are transposed for display so that all panels place coarse nodes on the vertical axis and fine nodes on the horizontal axis. Color represents weight; dark blue includes absent connections and small weights.

C.4 Predictions Across Datasets

The following figures display the ground truth, AMS-GNO prediction, and pointwise absolute error in that order. The target and prediction share a color scale within each row, and the error uses a separate scale.

NASA-CRM. Figure 10 shows the surface pressure coefficient on a three-dimensional aircraft geometry.

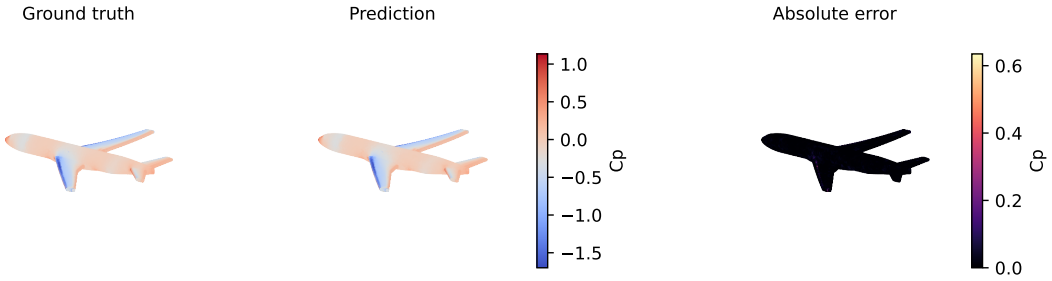

Figure 10: NASA-CRM surface pressure coefficient C_p: ground truth, AMS-GNO prediction, and absolute error.

DeepJEB. Figure 11 shows the von Mises stress field on a structural bracket.

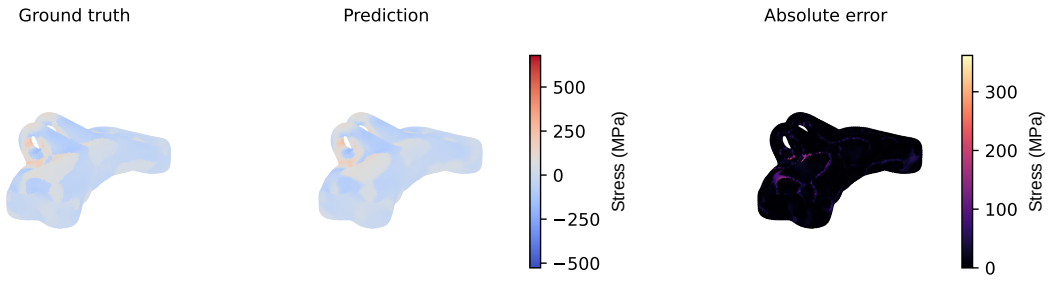

Figure 11: DeepJEB example 70: ground-truth von Mises stress, AMS-GNO prediction, and absolute error (MPa).

ShapeNet-Car. Figures 12 and 13 show the surface pressure and the three volume-velocity components for the same car example. These are the quantities supervised by the ShapeNet-Car training loss.

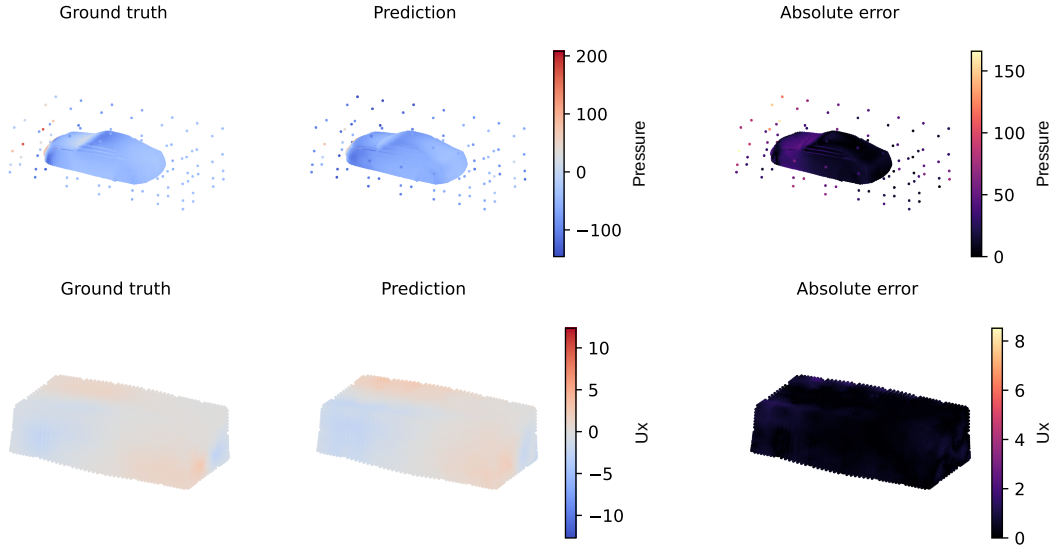

Figure 12: ShapeNet-Car surface pressure (top) and volume velocity U_x (bottom): ground truth, AMS-GNO prediction, and absolute error.

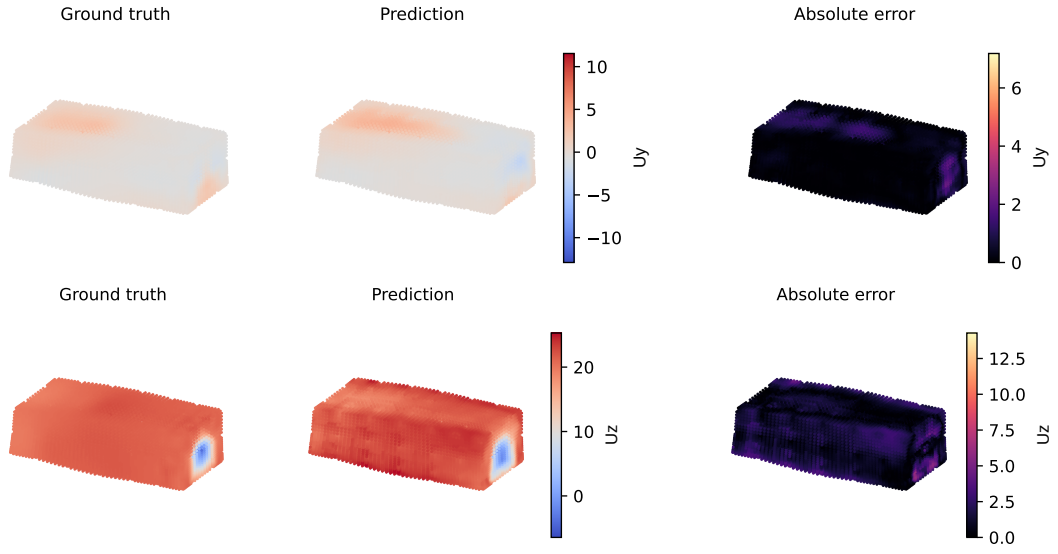

Figure 13: ShapeNet-Car volume velocities U_y (top) and U_z (bottom): ground truth, AMS-GNO prediction, and absolute error.

AirfRANS. Figures 14–17 show all four channels in the volume and on the airfoil surface: velocity components U_x and U_y, pressure p, and turbulent viscosity ν_t.

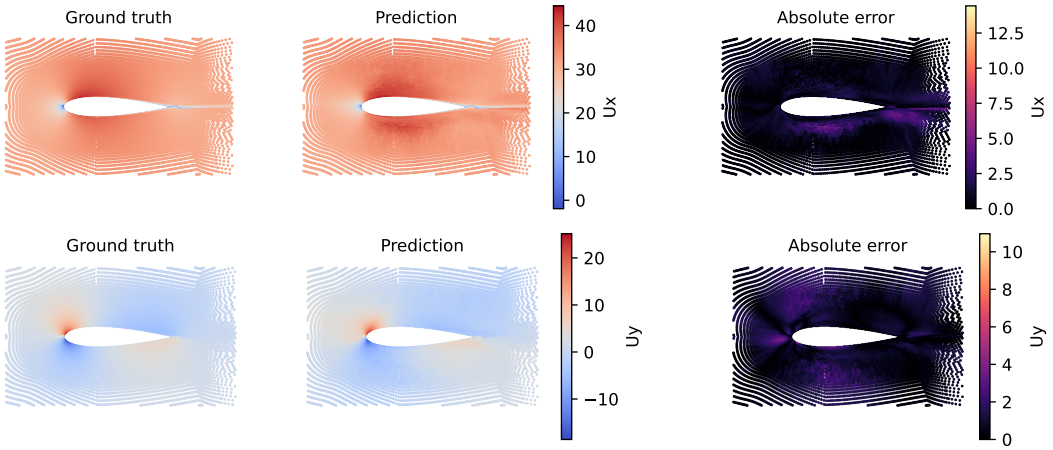

Figure 14: AirfRANS volume velocity: U_x (top) and U_y (bottom). Each row shows ground truth, prediction, and absolute error.

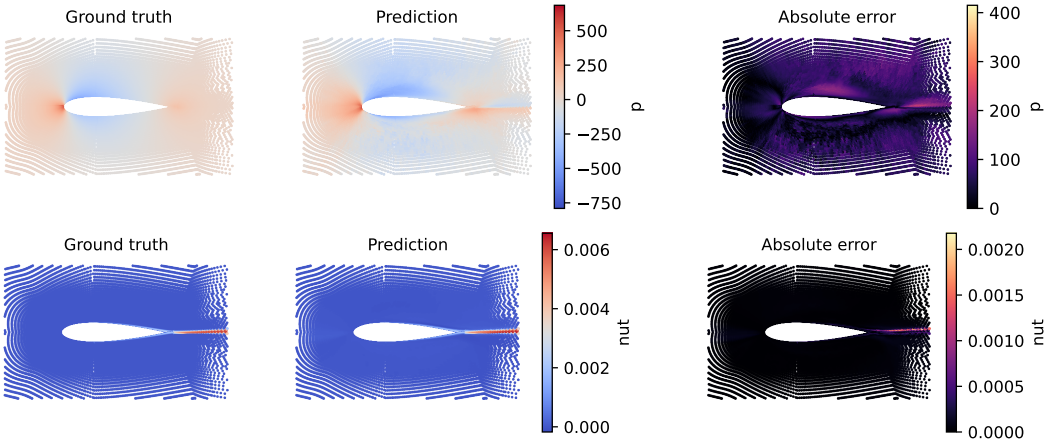

Figure 15: AirfRANS volume pressure p (top) and turbulent viscosity ν_t (bottom): ground truth, prediction, and absolute error.

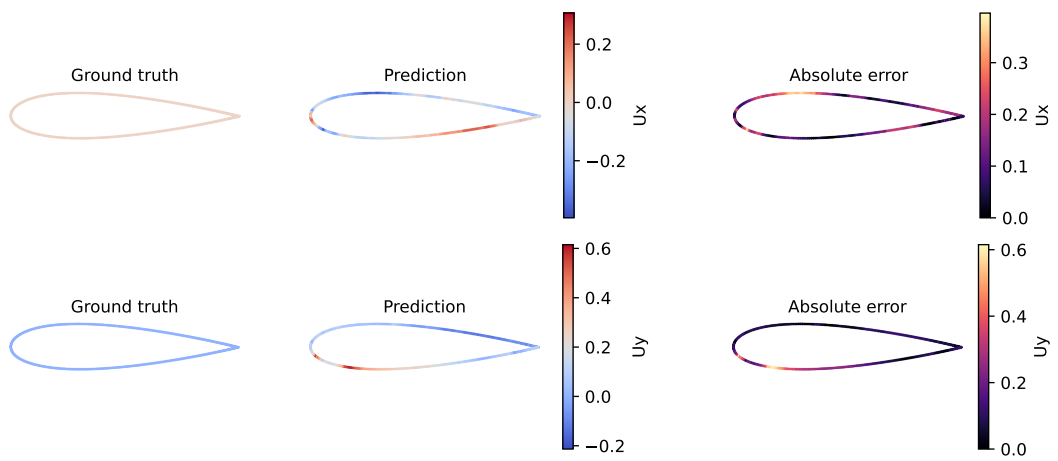

Figure 16: AirfRANS surface velocity: U_x (top) and U_y (bottom). Each row shows ground truth, prediction, and absolute error.

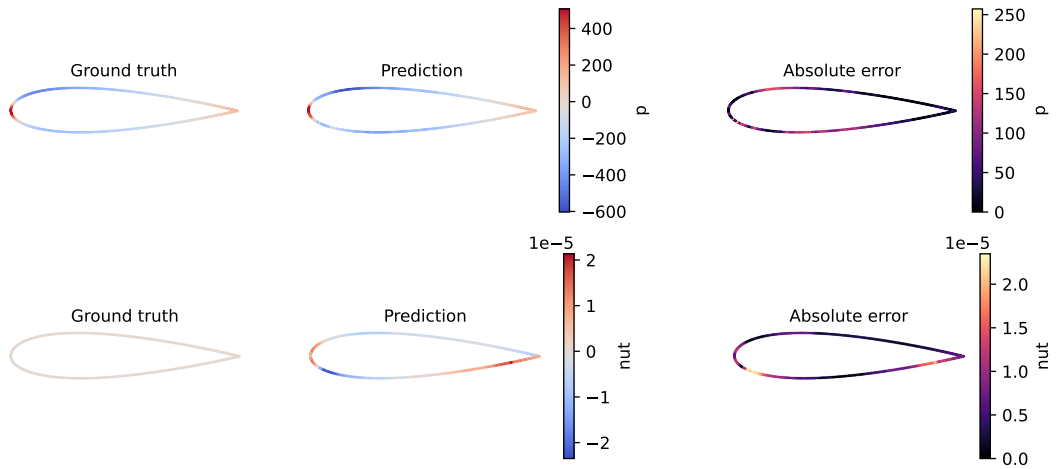

Figure 17: AirfRANS surface pressure p (top) and turbulent viscosity ν_t (bottom): ground truth, prediction, and absolute error.

C.5 AirfRANS Errors Across Multiple Samples

Figures 18–22 display the complete ten-example AirfRANS visualization batch, with exported indices 000–009. Presenting the full batch exposes variation in the location and magnitude of prediction errors across cases and provides a broader view than a single example. Each example is arranged as a two-by-two grid: errors in U_x and U_y above, and p and ν_t below. The individual color bars give the error scale of each panel.

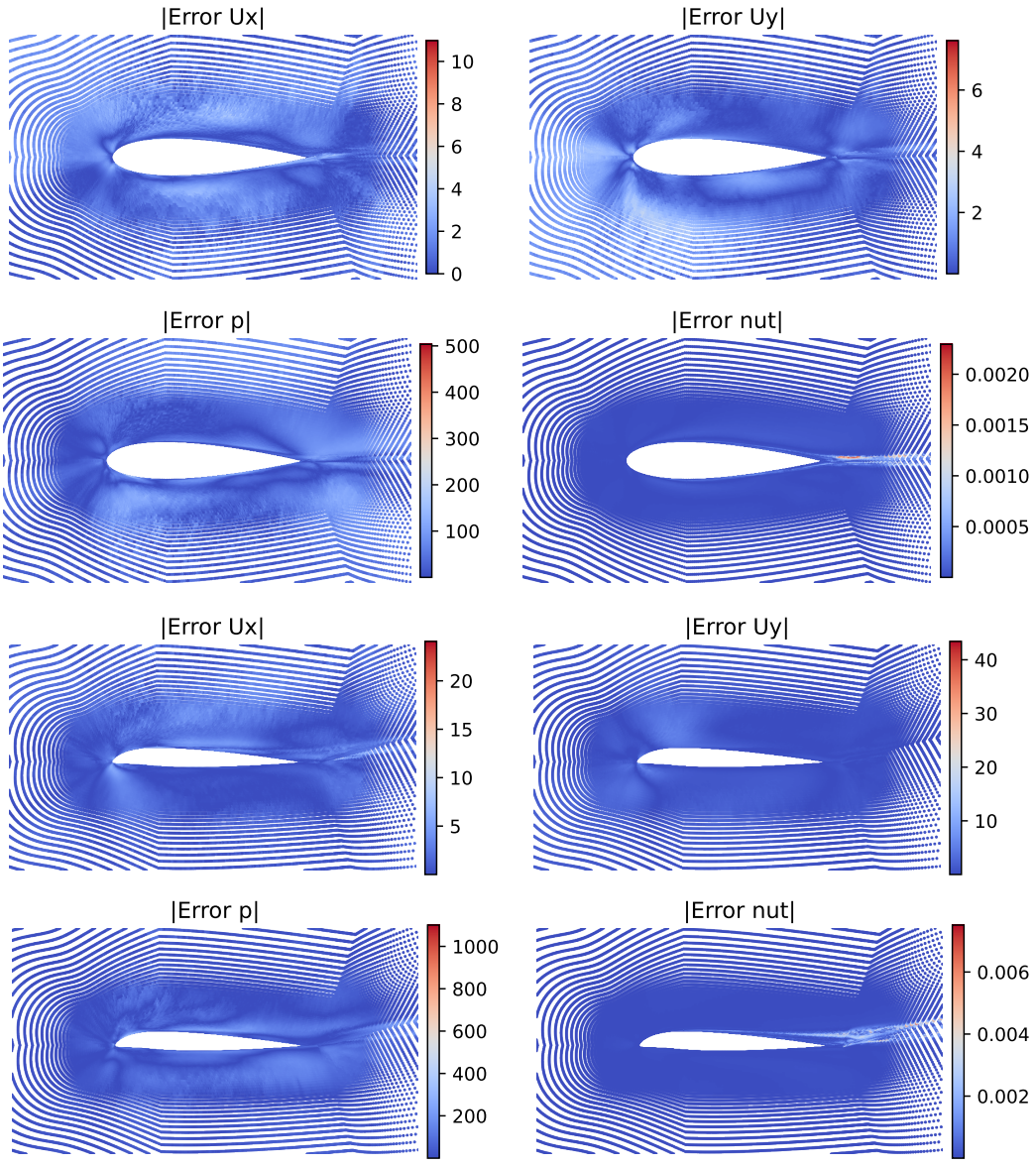

Figure 18: AirfRANS absolute errors for examples 000 (top) and 001 (bottom).

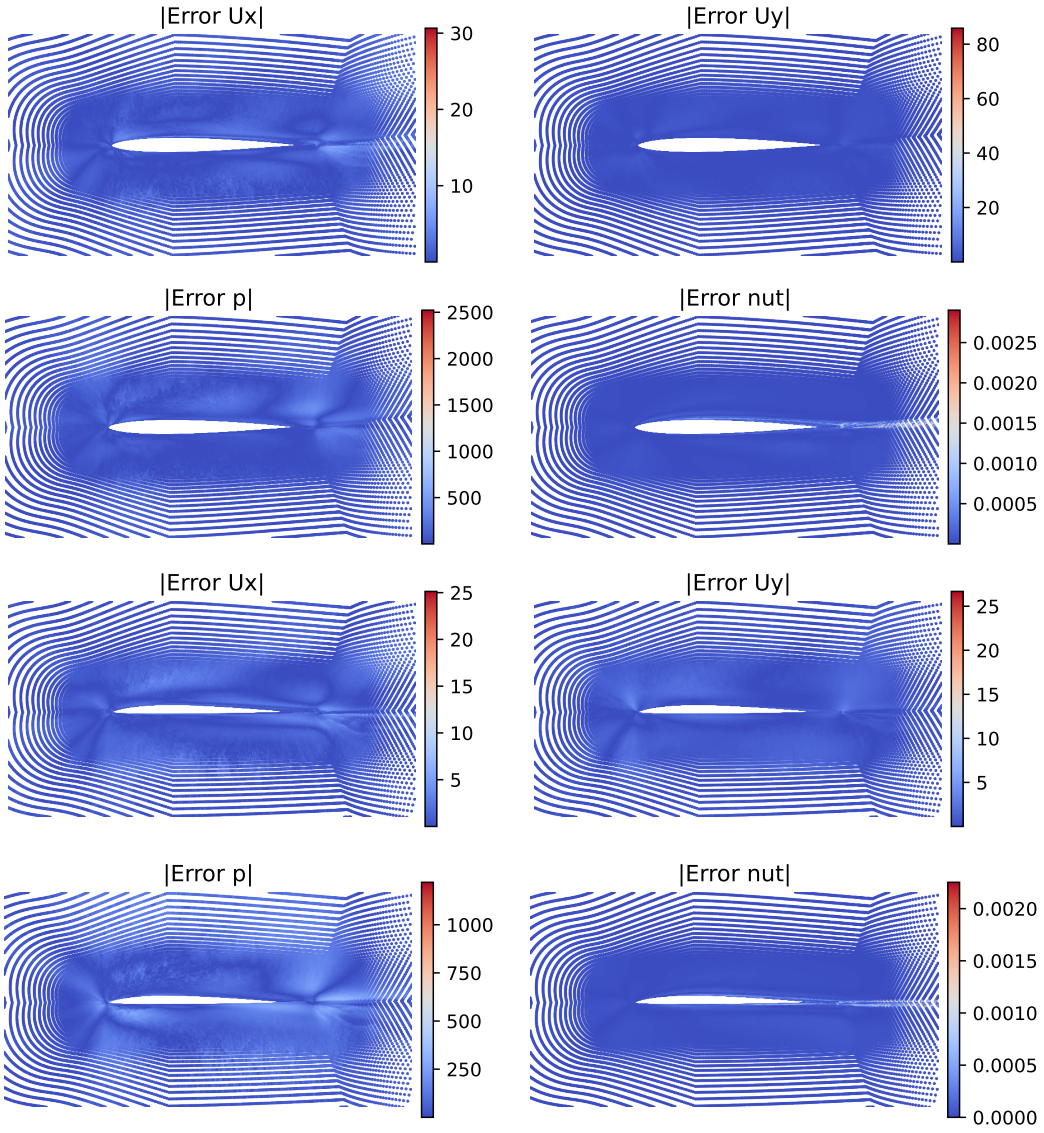

Figure 19: AirfRANS absolute errors for examples 002 (top) and 003 (bottom).

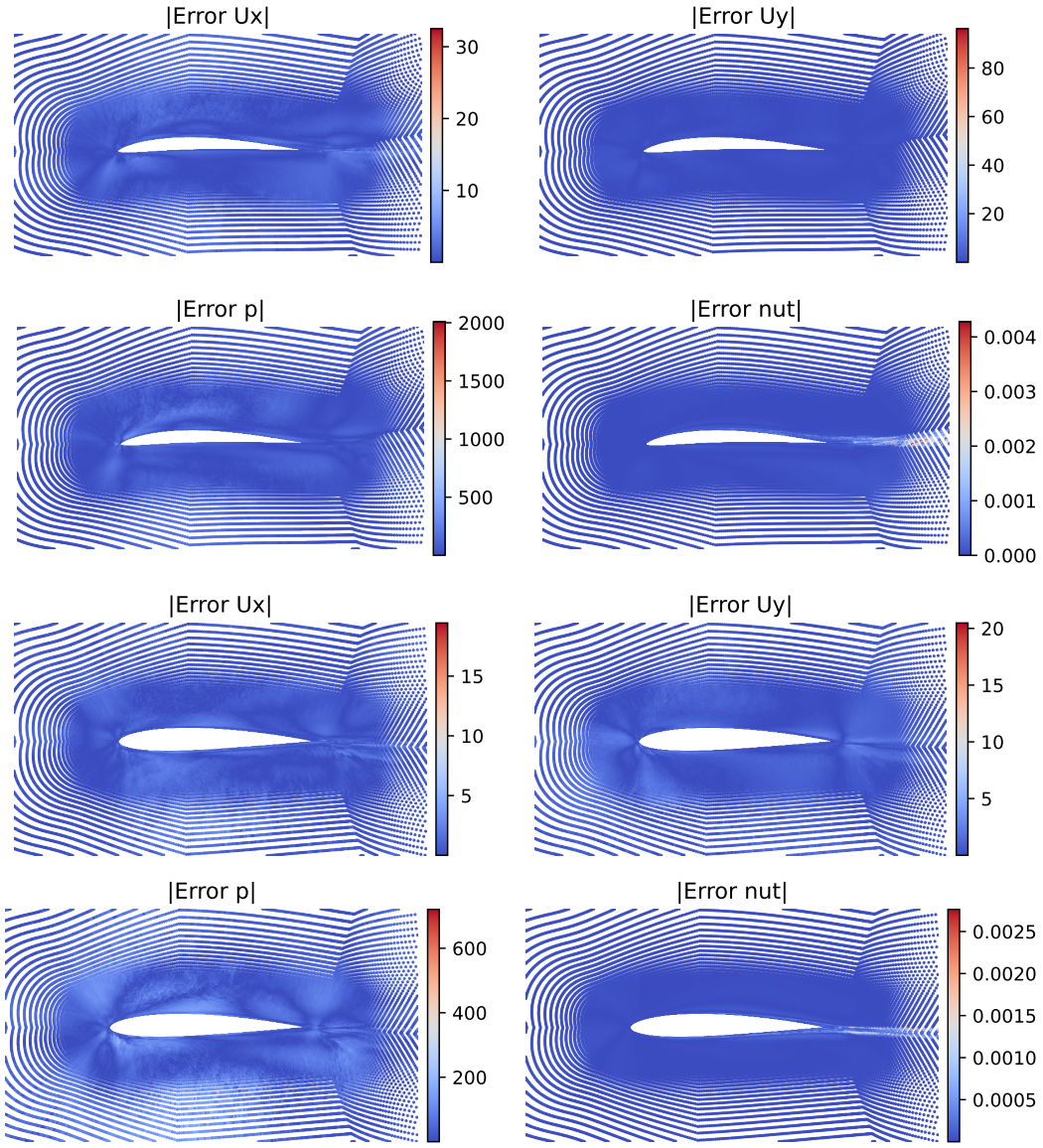

Figure 20: AirfRANS absolute errors for examples 004 (top) and 005 (bottom).

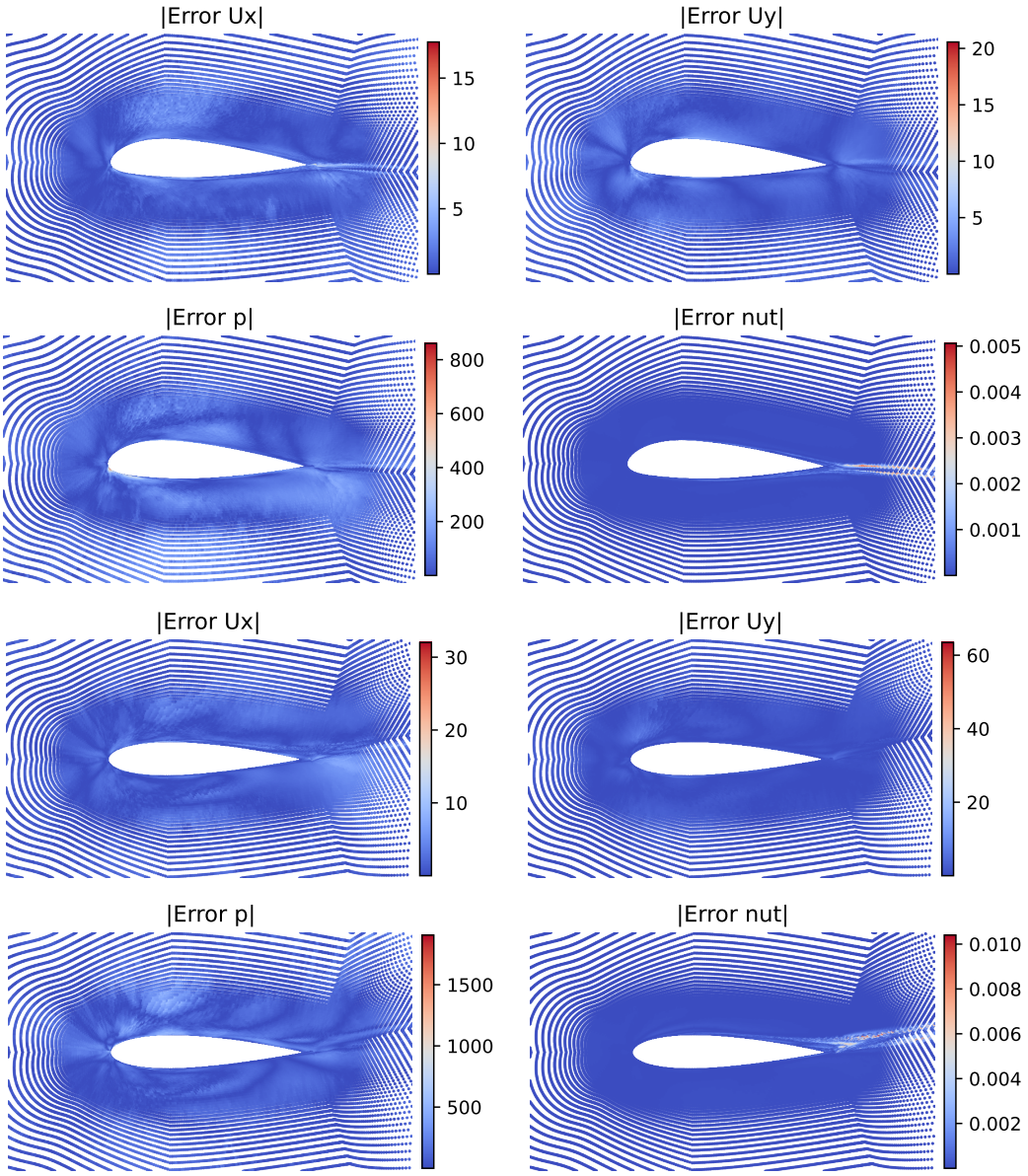

Figure 21: AirfRANS absolute errors for examples 006 (top) and 007 (bottom).

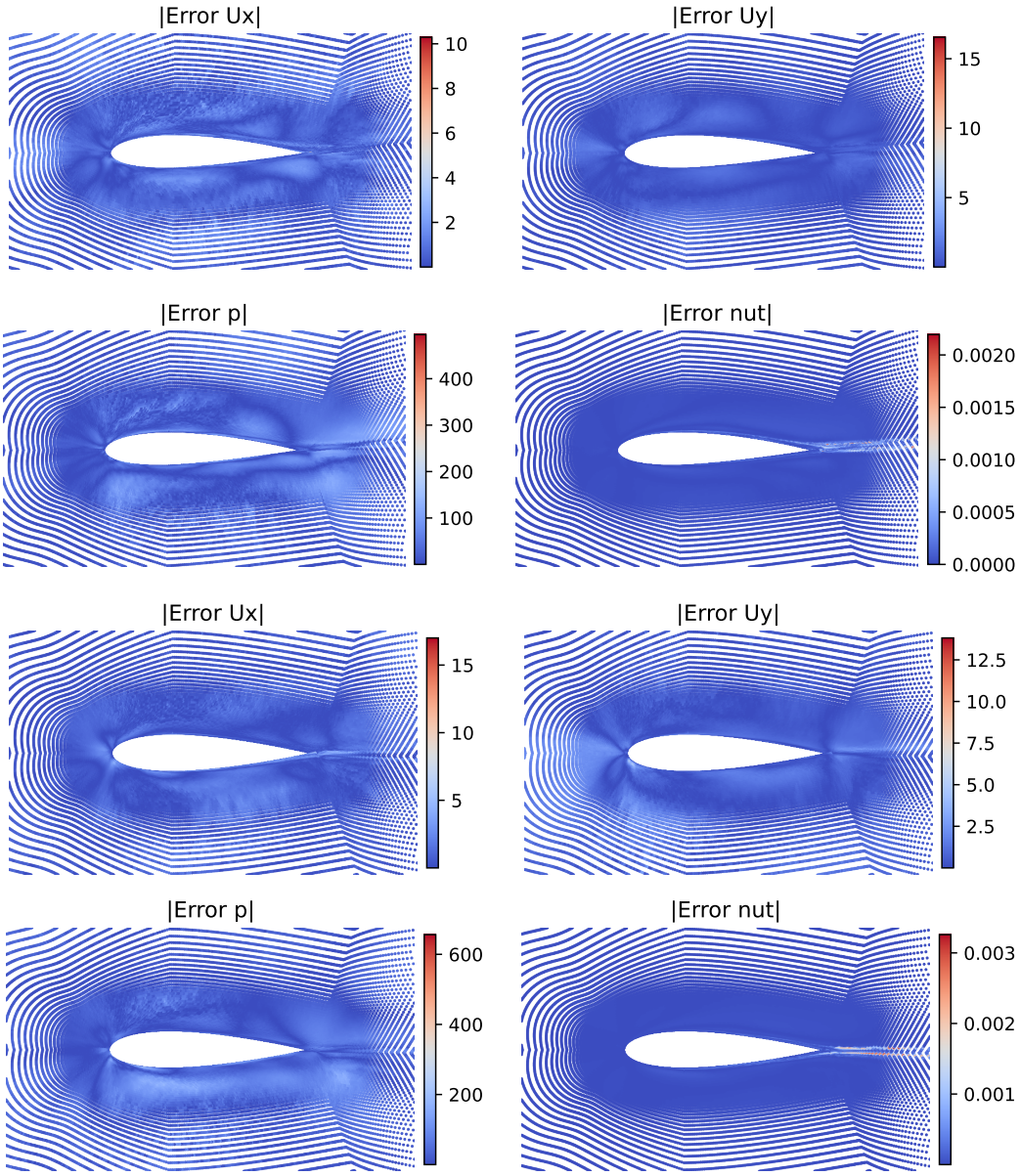

Figure 22: AirfRANS absolute errors for examples 008 (top) and 009 (bottom).