MESH-AWARE TOKENIZATION FOR STRUCTURED GRAPH TRANSFORMERS: LEARNING PHYSICAL SIMULATION ON HIGHER-ORDER MESHES

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a deterministic mesh-aware tokenization strategy for graph-transformer-based physical simulation on higher-order meshes. Instead of learning token assignments, the proposed method constructs structured tokens directly from the corner nodes of higher-order finite elements and couples them to the fine mesh through parameter-free aggregation and interpolation operators derived from finite-element shape functions. This provides a discretization-aligned interface for global transformer attention without converting higher-order meshes into first-order graphs. Building on this formulation, we propose the Interpolation Mesh Graph Token Transformer (IMGTT), which combines local MeshGraphNet-style message-passing with iteratively coupled global attention over mesh-derived tokens. Unlike the dual-graph MS-IMGN architecture, IMGTT replaces coarse message-passing with transformer processing while preserving direct higher-order compatibility. Across static and dynamic benchmarks and element orders one through five, IMGTT consistently improves long-horizon rollout behavior over learned-token MGN-T while retaining competitive predictive accuracy across all evaluated domains. Compared with the native higher-order MS-IMGN baseline, IMGTT reduces processor parameter count by 72% and achieves approximately $3\times$ faster training and inference. These results demonstrate that structured mesh-derived tokenization provides an effective interface between higher-order graph simulation and global transformer processing.

1 INTRODUCTION

Learning-based physical simulation on discretized domains has emerged as a scalable alternative to traditional numerical solvers. Mesh-based graph neural networks, particularly MeshGraphNet Pfaff et al. (2021) and related developments, have demonstrated strong performance in solid mechanics, fluid dynamics, and multiphysics systems while supporting irregular geometries and varying mesh resolutions.

Higher-order finite-element discretizations are widely used in engineering simulation because they provide improved accuracy, geometric representation, and convergence compared with first-order meshes. However, most learned mesh simulators operate only on first-order connectivity graphs. Higher-order meshes must therefore be converted into first-order representations, as illustrated in Figure 1. Such conversions either simplify the original higher-order structure or increase graph size by subdivision, while also increasing graph distances and the cost of long-range information propagation in the case of message-passing neural networks.

The Interpolation Mesh-Graph Net (IMGN) Sebestyen & Kim (2025) addressed higher-order compatibility by coupling fine and coarse graph representations through FEM-inspired aggregation and interpolation operators. Although this preserves the native discretization and improves information propagation, both representations remain message-passing graphs, making long-range interactions still computationally expensive. Conversely, MeshGraphNet-Transformer (MGN-T) Iparraguirre et al. (2026) improves global propagation by mapping first-order graph nodes to learned latent tokens and applying transformer attention. Its tokenization, however, is learned rather than explicitly tied to the underlying mesh discretization and does not directly support higher-order meshes.

In this work, we use FEM shape-function operators as a deterministic graph-tokenization interface

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

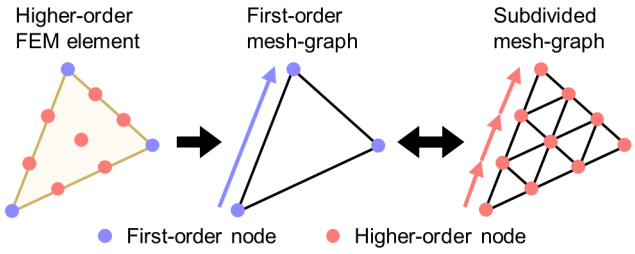


Figure 1: Higher-order meshes provide accurate engineering discretizations, but most learned simulators require conversion to first-order graph representations. This either oversimplifies the native higher-order mesh or increases graph size through subdivision. Both approaches weaken the consistency with the original higher-order discretization.

for global transformer processing. Coarse corner nodes of higher-order elements form a structured token set, while parameter-free aggregation and interpolation operators exchange information with the fine higher-order graph. Unlike IMGN, the coarse representation is no longer processed as a second graph with explicit coarse edges but instead the global interactions occur through transformer attention over mesh-derived tokens. Positional encodings provide spatial information to the token representation, while repeated aggregation, attention, interpolation, and local message-passing iteratively couple global and fine-scale information.

Building on this formulation, we propose the Interpolation Mesh Graph Token Transformer (IMGTT), which combines MeshGraphNet-style local processing with structured global attention while operating directly on higher-order discretizations. This design retains the geometric inductive bias of mesh-based processing, eliminates learned token assignment, and substantially reduces processor size and runtime relative to the dual-graph IMGN formulation. Experiments across static and dynamic higher-order benchmarks show improved rollout stability over learned-token MGN-T while providing a favorable trade-off between prediction accuracy and computational efficiency.

1.1 Related work

Graph neural networks are widely used for learned physical simulation on irregular domains. Early graph-network formulations Battaglia et al. (2018) and Message-Passing Neural Networks Gilmer et al. (2017) established iterative neighborhood aggregation for latent interaction modeling, while Graph Network-based Simulators (GNS) Sanchez-Gonzalez et al. (2020) demonstrated autoregressive simulation of fluids, rigid bodies, and particle systems. MeshGraphNet (MGN) Pfaff et al. (2021) extended this concept to computational meshes, but its local message-passing requires increasingly many propagation steps as the graph size and interaction distance increase.

Multiscale approaches address this limitation using hierarchical representations. MultiScale-MGN Lino et al. (2021) and GraphCast Lam et al. (2023) employ multiple graph resolutions, while FEM-inspired hyper-graph neural networks Gao et al. (2023), MAgNET Deshpande et al. (2024), and Bi-Stride Multi-Scale GNNs Cao et al. (2023) construct hierarchical graph levels. X-MeshGraphNet Nabian et al. (2024) and adaptive multiscale methods Perera & Agrawal (2024) further explore coarse-fine processing for large-scale simulation. These methods nevertheless operate only on first-order mesh-based graph discretizations. Neural-operator approaches, including DeepONet Lu et al. (2019), Fourier Neural Operators Li et al. (2020a), Physics-Informed Neural Operators Li et al. (2021), and Graph Neural Operators Li et al. (2020b), provide global operator learning beyond local message-passing but are also not specifically designed for direct processing of higher-order element-based discretizations.

IMGN Sebestyen & Kim (2025) directly addressed higher-order FEM meshes by coupling fine and coarse graph representations based on a single higher order mesh, through aggregation and interpolation operators derived from element shape functions. This preserves the native higher-order discretization and improves multiscale information exchange, but both resolutions remain dependent on local message propagation.

Graph-transformer methods provide an alternative mechanism for long-range interaction. Graph Transformer Networks Yun et al. (2019) introduced learned graph transformations through adaptive meta-path construction, while Graphormer Ying et al. (2021) incorporated structural graph informa-

tion into global node-level self-attention. GraphGPS Rampášek et al. (2023) established a general hybrid framework combining local message-passing with global transformer processing. For physical simulation, MeshGraphNet-Transformer Iparraguirre et al. (2026) augments MGN with learned token-based transformer processing, while related transformer-based mesh methods Garnier et al. (2025) and hybrid local-global attention models Patrignani & Pinho (2026) further demonstrate the effectiveness of combining message-passing and attention. Existing approaches, however, remain primarily based on first-order mesh representations, leaving higher-order discretization compatibility and structured global tokenization largely unresolved.

Table 1: Properties of compared models. MGN and MGN-T are indirect baselines because they require higher-order meshes to be converted to first-order graph representations, while MS-IMGN provides the direct native higher-order mesh-graph baseline.

Property	**MGN**	**(MS-)IMGN**	**MGN-T**	**IMGTT**
Native higher-order	No	Yes	No	Yes
Needs first-order conversion	Yes	No	Yes	No
Fine local MPNN	Yes	Yes	Yes	Yes
Global attention	No	No	Yes	Yes
Coarse representation	None	Graph	Learned tokens	Structured tokens
Coarse mapping	None	Shape function	Learned	Shape function
Fine/coarse coupling	None	Interleaved	Sequential	Interleaved
Comparison role	Indirect	Direct baseline	Indirect	Proposed

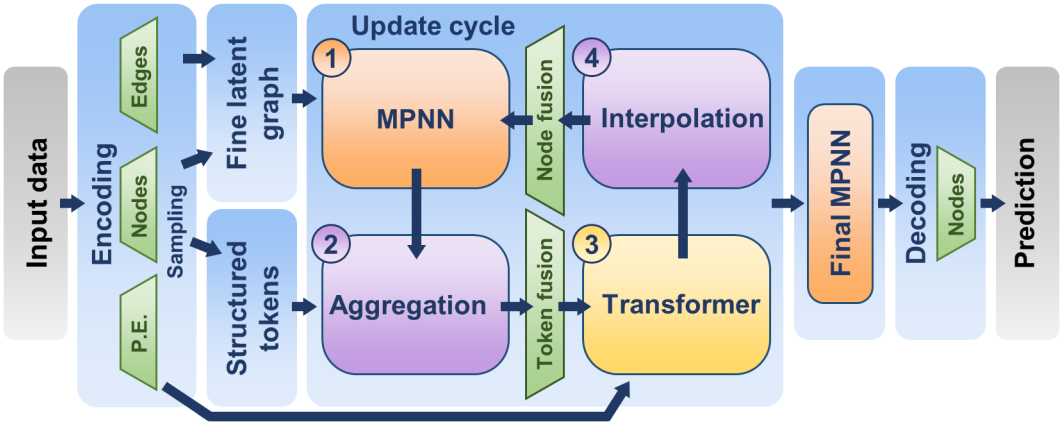


Figure 2: The general flow of the proposed IMGTT forward pass algorithm. Here, P.E. denotes the positional encodings of the higher-order element corner nodes that form the structured tokens.

2 Model Architecture

Let the fine graph be $\mathcal{G}_f = (\mathcal{V}_f, \mathcal{E}_f, \mathcal{E}_w)$, with node features $\mathbf{x}_i \in \mathbb{R}^{d_v}$, edge features $\mathbf{e}_{ij} \in \mathbb{R}^{d_e}$ and optional world-edge features $\mathbf{w}_{ij} \in \mathbb{R}^{d_w}$ that might model domain-dependent effects, such as proximity-based contact phenomena. This graph contains every node of the higher-order mesh.

The proposed model (Figure 2) follows the encoder-processor-decoder architecture established in Battaglia et al. (2018) and adopted by Sanchez-Gonzalez et al. (2020) among many other GNN-based developments. The fine mesh-graph is encoded into a latent graph representation, with latent node, mesh-edge and world-edge feature sets as h_i, g_{ij}, q_{ij} respectively.

From this, we define a sampled coarse node set $\mathcal{V}_c \subset \mathcal{V}_f$, consisting of the first-order corner nodes of the higher-order mesh elements. This node set is directly used as the structured transformer token representation, defined as $Z = \{z_a\}_{a \in \mathcal{V}_c}$, where each coarse token z_a corresponds to a sampled coarse node $a \in \mathcal{V}_c$. The coarse token features are augmented with sinusoidal standing-wave positional encodings $P = \{p_a\}_{a \in \mathcal{V}_c}$, following the positional encoding strategy of the MGN-T model,

to provide spatial information between the coarse tokens during the transformer processing.
For the interpolation formulation, let $\mathcal{K}(i)$ denote the set of higher-order elements containing fine node $i \in \mathcal{V}_f$ and let $\mathcal{S}(k)$ denote the index set of linear shape functions associated with element $k \in \mathcal{K}(i)$. Since each linear shape function corresponds to one coarse corner node of the element, the notation $z_{a(n,k)}$ denotes the coarse token associated with shape function $n \in \mathcal{S}(k)$ of element k. This formulation enables element-wise operations between the coarse token representation and the fine higher-order mesh discretization.

2.1 Mesh-Aware Tokenization

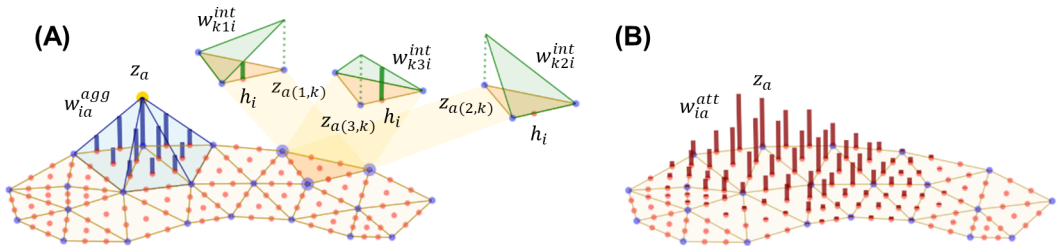


Figure 3: (A) Interpolation-based aggregation weights map fine nodes to coarse tokens, while interpolation weights map coarse tokens of each element back to fine nodes, as in IMGTT. (B) Learned soft-clustering tokenization, where attention weights perform node-to-token aggregation and token-to-node de-clustering, as in MGN-T.

The central component of the proposed framework is the mesh-aware tokenization mechanism based on linear interpolation operators, shown in Figure 3. Unlike learned tokenization methods that predict token assignments dynamically during both training and inference, as summarized in Table 1, the aggregation and interpolation operators are derived directly from the higher-order mesh structure in the undeformed configuration. Consequently, the tokenization operator remains free of learnable parameters, discretization-consistent, and invariant throughout rollout simulations. The aggregation and interpolation operators follow the formulation introduced in IMGN Sebestyen & Kim (2025), where multiple mesh-graph resolutions are coupled using mappings derived from element-wise linear interpolation used in conventional FEM Bathe (1996).
Here, $\mathbf{FC}_a$ represents the aggregated fine-to-coarse message received by coarse token $a \in \mathcal{V}_c$:

$$\mathbf{FC}_a = \sum_{i \in \mathcal{N}(a)} \hat{\mathbf{h}}_i \, w_{ia}^{\text{agg}} \tag{1}$$

where $\hat{\mathbf{h}}_i$ are the intermediate fine-node latent features of node i after local message-passing and w_{ia}^{agg} are the aggregation weights derived from linear interpolation cover functions and $\mathcal{N}(a)$ denotes the set of fine nodes contributing to coarse token a. The resulting aggregated message is $\mathbf{FC}_a$, which is a weighted summation of the fine nodes per coarse token.
In the interpolation step, the coarse tokens are interpolated back to the fine graph through the corresponding interpolation operators, producing the coarse-to-fine messages:

$$\mathbf{CF}_i = \frac{1}{|\mathcal{K}(i)|} \sum_{k \in \mathcal{K}(i)} \sum_{n \in \mathcal{S}(k)} \mathbf{z}_{a(n,k)} \, w_{kni}^{\text{int}}. \tag{2}$$

where $a(n,k)$ denotes the coarse token corresponding to shape function n of element k.
Unlike the aggregation step, the interpolation operation is performed element-wise rather than through direct node-wise summation. Each coarse token $\mathbf{z}_a$ is multiplied by the corresponding interpolation weight w_{kni}^{int} associated with shape function n of element k. The weighted contributions are first summed across the shape functions within each element and then averaged across all elements containing the fine node i. This averaging is necessary because the fine nodes may belong to multiple higher-order elements, ensuring a single message for each fine-node representation.

In contrast to learned tokenization methods, the proposed operator-based tokenization is deterministic and parameter-free. Token identities and the fine-to-coarse and coarse-to-fine mappings are defined directly by the mesh discretization and remain fixed throughout training, inference, and autoregressive rollouts. At each unique mesh discretization, the token set is constructed from the corresponding coarse corner-node representation, without requiring a predefined token budget or learned reassignment. The aggregation and interpolation operators therefore provide a fixed discretization-aligned interface between fine and coarse representations, preserving the underlying mesh geometry while avoiding variability from learned token assignments. The computational cost of transformer processing nevertheless remains dependent on the resulting number of coarse tokens.

2.2 Encoding

The raw fine-node, fine-edge and world-edge features are first embedded into a common hidden dimension:

$$\mathbf{h}_i^0 = \phi_v^{enc}(\mathbf{x}_i),\ \mathbf{g}_{ij}^0 = \phi_e^{enc}(\mathbf{e}_{ij}),\ \mathbf{q}_{ij}^0 = \phi_w^{enc}(\mathbf{w}_{ij}) \tag{3}$$

If the world-edges are not used in the given system, they are simply omitted.
The coarse tokens are initialized by selecting the corresponding encoded fine nodes that are spatially located at the corners of the elements:

$$\mathbf{Z}^0 = \{\mathbf{z}_a^0\}_{a\in\mathcal{V}_c}, \qquad \mathbf{z}_a^0 = \mathbf{h}_a^0. \tag{4}$$

The $\mathbf{X}_a$ initial mesh-space positions of the coarse tokens are processed by PE sinusoidal positional encoding and projected into the hidden space by $\mathbf{W}_p$ linear projection:

$$\mathbf{p}_a = \mathbf{W}_p\,(\mathrm{PE}(\mathbf{X}_a)), \qquad a \in \mathcal{V}_c \tag{5}$$

2.3 Processor block

At each layer l, the model first performs fine-scale local message-passing. The $\mathbf{g}_{ij}$ mesh edges are updated using the receiver and sender node states with ϕ_e^l MLP. If $\mathbf{q}_{ij}$ world-edges are used, they are updated analogously with ϕ_w^l MLP:

$$\mathbf{g}_{ij}^{l+1} = \mathbf{g}_{ij}^l + \phi_e^l\left(\left[\mathbf{g}_{ij}^l, \mathbf{h}_i^l, \mathbf{h}_j^l\right]\right) \tag{6}$$

$$\mathbf{q}_{ij}^{l+1} = \mathbf{q}_{ij}^l + \phi_w^l\left(\left[\mathbf{q}_{ij}^l, \mathbf{h}_i^l, \mathbf{h}_j^l\right]\right) \tag{7}$$

The incoming mesh-edge messages are summed at each receiver node. In case world-edges are present, the world-edge messages are aggregated separately:

$$\bar{\mathbf{g}}_i^{l+1} = \sum_{j:(i,j)\in\mathcal{E}_f} \mathbf{g}_{ij}^{l+1}, \quad \bar{\mathbf{q}}_i^{l+1} = \sum_{j:(i,j)\in\mathcal{E}_w} \mathbf{q}_{ij}^{l+1} \tag{8}$$

The node state is then updated by ϕ_v^l MLP:

$$\hat{\mathbf{h}}_i^{l+1} = \mathbf{h}_i^l + \phi_v^l\left(\left[\mathbf{h}_i^l, \bar{\mathbf{g}}_i^{l+1}, \bar{\mathbf{q}}_i^{l+1}\right]\right) \tag{9}$$

After the local fine-scale message-passing, the fine node features are aggregated to the coarse tokens using predefined aggregation weights, according to Equation 1:

$$\mathbf{FC}_a^{l+1} = \sum_{i\in\mathcal{N}(a)} \hat{\mathbf{h}}_i^{l+1}\, w_{ia}^{\mathrm{agg}} \tag{10}$$

The aggregated messages are fused into the current coarse token state with ϕ^l_{coarse} fusion MLP, where, $\hat{\mathbf{z}}^l_a$ denotes the latent state of coarse token $\mathbf{z}^l_a$ after the fusion step.

$$\hat{\mathbf{z}}^l_a = \mathbf{z}^l_a + \phi^l_{\text{coarse}}\left(\left[\mathbf{z}^l_a, \mathbf{FC}^{l+1}_a\right]\right) \tag{11}$$

Before applying the Transformer, the projected positional encoding is added to the coarse latent tokens at each updating step, providing spatial information about the coarse tokens:

$$\widetilde{\mathbf{z}}^l_a = \hat{\mathbf{z}}^l_a + \mathbf{p}_a. \tag{12}$$

Accordingly, $\widetilde{\mathbf{Z}}^l$ denotes the coarse-token latent representation after the fusion with the aggregated fine-node messages and the addition of the positional encoding.

$$\widetilde{\mathbf{Z}}^l = \left[\widetilde{\mathbf{z}}^l_1, \widetilde{\mathbf{z}}^l_2, \ldots, \widetilde{\mathbf{z}}^l_A\right] \qquad A = |\mathcal{V}_c| \tag{13}$$

The resulting token representation is then processed through the transformer layer, where $\mathbf{Z}^{l+1} = \left\{\mathbf{z}^{l+1}_a\right\}_{a\in\mathcal{V}_c}$ denotes the updated coarse-token latent representation.

$$\mathbf{Z}^{l+1} = \text{Transformer}^l\left(\widetilde{\mathbf{Z}}^l\right) \tag{14}$$

The updated coarse tokens are linearly interpolated back to the fine representation, according to Equation 2:

$$\mathbf{CF}^{l+1}_i = \frac{1}{|\mathcal{K}(i)|}\sum_{k\in\mathcal{K}(i)}\sum_{n\in\mathcal{S}(k)} \mathbf{z}^{l+1}_{a(n,k)}\, w^{\text{int}}_{kni}. \tag{15}$$

The interpolated messages provide each fine node with global coarse-level context derived from the transformer. These are fused into the fine latent nodes with ϕ^l_{fine} fusion MLP:

$$\mathbf{h}^{l+1}_i = \hat{\mathbf{h}}^{l+1}_i + \phi^l_{\text{fine}}\left(\left[\hat{\mathbf{h}}^{l+1}_i, \mathbf{CF}^{l+1}_i\right]\right). \tag{16}$$

This step injects global Transformer information back into the fine-scale MGN representation.
The processing block is repeated $L-1$ times, shown in detail in Algorithm 1 in Appendix A. After the last updating cycle, one additional fine-scale message-passing update is conducted with $\mathbf{h}^L_i$ output and no coarse transformer pass afterwards. This way, the message-passing blocks at the beginning and at the very end of the updating cycle act as local pre- and post-processing on the fine node features. In our experiments, four update cycles were sufficient for all benchmarks.

2.4 Decoding

Lastly, the output is predicted from the final fine-node latent state with ϕ^{dec}_v nodal decoder MLP. For node-wise simulation targets, $\hat{\mathbf{y}}_i$ may represent quantities such as displacement or velocity increment, derivatives such as acceleration, or absolute system variables like pressure or mechanical stress.

$$\hat{\mathbf{y}}_i = \phi^{dec}_v\left(\mathbf{h}^L_i\right) \tag{17}$$

3 Experimental domains

We evaluate the proposed framework on static and dynamic simulation domains defined on Lagrangian and Eulerian meshes, including cloth dynamics, fluid flow, and linear elasticity. The experiments use higher-order benchmarks introduced in the IMGN framework, including FlagDynamics and CylinderFlow from Pfaff et al. (2021), and a linear elastostatics dataset with stress concentration

around crack-like geometries. The datasets are created by interpolating the high-resolution first-order simulations onto higher-order discretizations, enabling controlled evaluation across element orders and mesh resolutions while preserving identical system behavior, as explained by Sebestyen & Kim (2025). Using shared reference trajectories avoids confounding model differences with trajectory divergence caused by independently generated trajectories. Therefore, the experiments test the learned simulation across higher-order representations, without the numerical accuracy differences between independently solved higher-order FEM discretizations.

Most graph-based simulators, including MGN and MGN-T, operate on first-order mesh graphs and cannot be applied directly to higher-order discretizations. As a consequence, higher-order meshes are converted into equivalent first-order graph representations for the MGN and MGN-T baseline comparison. In contrast, IMGN and the proposed IMGTT operate directly on the original higher-order meshes, preserving their geometric structure and mesh-derived inductive biases.

Although IMGTT uses two coupled representation levels, similar to MultiScale-IMGN, the simple IMGTT name is used for simplicity since no single-scale variant is considered.

4 Model configuration and evaluation

All models are implemented in PyTorch Paszke et al. (2019). MLPs use four layers with latent width 128, except the two-layer residual fusion MLPs. The positional projection W_p is a single linear layer without activation or normalization. Transformer layers also use latent width 128, four attention heads, and GELU activation. Continuous physical input features are normalized online using Welford statistics, while the intermediate representations use LayerNorm and the decoder output uses no activation nor normalization.

The models are trained for single-step prediction using Huber loss Huber (1964) with $\delta = 1.0$ and mean reduction, which was more stable than mean squared error. We use Adam optimization, gradient clipping, and a batch size of one graph. The learning rate decays exponentially from 10^{-4} to 10^{-5} over 1.5M steps, then linearly to 10^{-6} over the next 1M steps, and remains constant after. Flag dynamics models are trained for 5M steps, while fluid dynamics and linear elastostatics models use 2.5M steps as longer training caused overfitting on these datasets.

The performance is evaluated on 100 test simulations which are separate from the training and validation set in each domain with element orders from one through five using RMSE. RMSE-1 measures one-step prediction from ground-truth inputs, RMSE-50 evaluates autoregressive rollout over the first 50 steps limiting longer trajectory dependent error accumulation. RMSE-all evaluates the whole rollout. These metrics respectively characterize local prediction accuracy, short-term temporal consistency, and long-term rollout stability, with RMSE-all additionally reflecting accumulated trajectory error. The comprehensive error tables can be found in Appendix A.

5 Results and discussion

The MGN and IMGN/MS-IMGN accuracy results are taken from the published IMGN benchmark Sebestyen & Kim (2025), which uses the same datasets and evaluation protocol. These models were reimplemented in the current framework for implementation testing and computational profiling, while the published benchmark values are retained for the accuracy comparison due to the high computational cost of retraining the complete baseline ablation benchmark. Results are reported as mean RMSE with standard deviation over the 100 test simulations. RMSE-1, RMSE-50, and RMSE-all values are provided in Appendix A. For linear elastostatics, IMGN denotes the single-scale model, while the dynamic benchmarks use multi-scale MS-IMGN.

For linear elastostatics (Figure 4A), IMGTT achieves lower position RMSE than MGN-T across all element orders and shows substantially less degradation than MGN at higher orders. Stress errors are comparable to MGN-T for first- and second-order elements, while IMGTT becomes more accurate and less variable for orders three through five. IMGN remains the strongest higher-order baseline for most cases, indicating that the transformer-based formulation trades some single-step accuracy for substantially lower computational cost.

For flag dynamics (Figure 4B), IMGTT consistently improves over MGN-T in one-step prediction and achieves the lowest acceleration RMSE-1 across all element orders. The advantage becomes more pronounced during rollout, as IMGTT reduces RMSE-50 for position, velocity, and acceleration at every order relative to MGN-T. Over the full trajectory, position and velocity errors remain

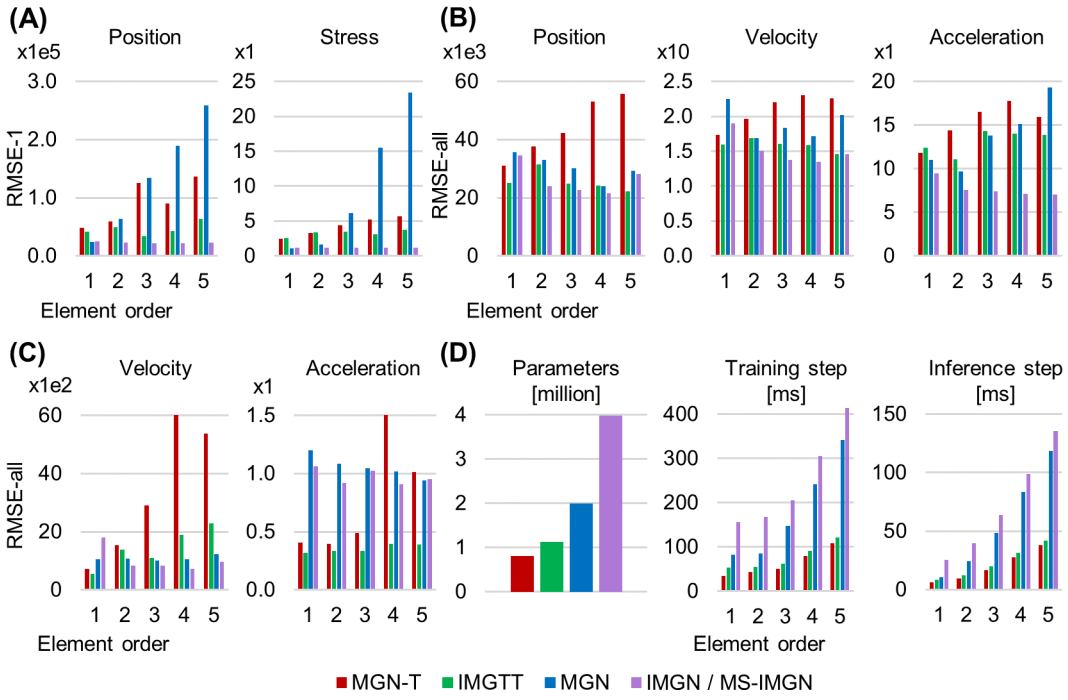


Figure 4: (A) Linear elastostatics position and stress RMSE-1, (B) flag dynamics position, velocity and acceleration RMSE-all, (C) fluid dynamics velocity and acceleration RMSE-all by element order from first to fifth. The excessively large fluid dynamics errors of the MGN-T model in 4th order are truncated for visualization. (D) Comparing processor parameter count reported in millions and FP64 runtime in training and inference. The displayed errors are mean values.

lower than MGN-T for all orders, with substantially reduced standard deviation at higher orders, indicating smaller variability across multiple trajectories. MS-IMGN remains strongest for full-rollout acceleration, while IMGTT provides similar or better performance in position and velocity. Thus, IMGTT is a favorable compromise between long-term accuracy and computational cost.

Fluid dynamics (Figure 4C) shows the clearest improvement in long-horizon transformer stability. IMGTT achieves lower one-step acceleration error than MGN-T for four of five element orders and generally improves short-rollout performance. During long rollouts, IMGTT reduces velocity error relative to MGN-T at every order. Furthermore, IMGTT avoids the severe high-order instability observed for MGN-T at element order four, where MGN-T shows rollout divergence producing errors that exceed the plotted range and are therefore truncated for visualization. Most notably, IMGTT achieves the lowest full-rollout acceleration RMSE and standard deviation among all evaluated models for every element order. These results show that IMGTT improves both long-term accuracy and trajectory consistency in the fluid benchmark, while conventional message-passing baselines remain competitive in velocity prediction.

Overall, IMGTT does not uniformly minimize error across all quantities and domains, but consistently improves the rollout behavior of the learned-token transformer baseline. The strongest gains occur at higher element orders and during long autoregressive rollouts, where both mean error and trajectory-to-trajectory variability increase substantially for MGN-T. Combined with the computational complexity discussed below, this supports IMGTT as an efficient higher-order alternative that retains competitive accuracy while providing more robust global information propagation.

5.1 Computational complexity

Although IMGTT does not achieve the lowest error on every benchmark, it provides a strong trade-off between accuracy, rollout stability, and computational efficiency. In the dynamic flag and fluid domains, it is more stable than MGN-T in long-horizon rollouts while remaining substantially

cheaper and faster than MS-IMGN.

Under the shared encoder-processor-decoder configuration without world edges, MGN uses 15 message-passing steps with 30 latent MLPs, while MS-IMGN uses 60 MLPs across two coupled graph representations. MGN-T uses four message-passing steps and two transformer layers, while IMGTT uses five message-passing steps and four transformer layers. The resulting processor sizes are approximately 0.80M parameters for MGN-T, 1.13M for IMGTT, 1.99M for MGN, and 3.98M for MS-IMGN. Consequently, IMGTT uses only 28% of the MS-IMGN parameters, a 72% reduction, while remaining only moderately larger than MGN-T (Figure 4D).

Runtime measurements for all models are obtained from our own implementations under the same hardware, precision (FP64) on an RTX A6000 GPU, excluding data loading and graph preprocessing. Training times are averaged over the whole run, and inference times over all steps of 100 test rollouts. Preliminary FP32 experiments showed reproducibly less stable behavior during training and inference. This is consistent with sensitivity to non-associative CUDA scatter reductions, where small ordering-dependent numerical differences can be amplified by deep iterative processing.

Across element orders 1-5, IMGTT reduces MS-IMGN training time from approximately 156-415 ms to 53-121 ms per step, corresponding to a 2.94-3.42$\times$ speedup. Inference time decreases from about 25-135 ms to 8-42 ms, achieving 2.93-3.22$\times$ speedup. Compared with MGN-T, IMGTT incurs only a modest increase in parameters and runtime while providing improved rollout stability and direct higher-order compatibility through deterministic mesh-aware tokenization.

Overall, IMGTT retains structured multiscale higher-order processing at a fraction of the computational cost of MS-IMGN, making it an efficient alternative for learned higher-order mesh-based simulation.

5.2 Limitations and Future Work

A key limitation of IMGTT arises from transformer-based processing. Although global attention improves information propagation and rollout stability, it is less physically interpretable than explicit message-passing. In MPNN-based simulators, information propagates along mesh edges whose latent features directly encode spatial relationships such as relative position. Transformer layers instead operate on token embeddings without explicit latent edges and must infer relative spatial relationships from the less direct positional encoding.

This trade-off is reflected in the results as well. IMGTT improves rollout stability and computational efficiency but does not always achieve the lowest numerical error. In spatially fine-grained domains, explicit message-passing models may remain more accurate because geometric relationships are encoded directly, but at a substantially higher computational cost. Like MGN-T, IMGTT retains the $O(N_c^2)$ complexity with respect to the number of coarse tokens due to global self-attention, although the attention operates only on the reduced token set rather than the full fine mesh. As a consequence, the structured transformer-based IMGTT can be considered as an efficient and stable alternative rather than a universal replacement for high-resolution message-passing.

Future work will investigate hybrid attention mechanisms that retain information from the geometric structure while preserving efficient token-based global propagation. Mesh-informed attention weights and analysis of learned attention patterns may further improve locality, interpretability, and understanding of when explicit message-passing remains necessary.

6 Conclusion

We introduced a deterministic mesh-aware tokenization strategy for graph transformers on higher-order discretizations. By using finite-element shape-function operators as a parameter-free interface between fine mesh nodes and structured coarse tokens, the proposed method enables global transformer processing while preserving direct compatibility with the underlying higher-order mesh. Building on this formulation, IMGTT combines local mesh-graph message-passing with iteratively coupled global attention, replacing the expensive coarse message-passing graph of MS-IMGN with a structured token representation. Across static and dynamic benchmarks, IMGTT provides competitive prediction accuracy, improved long-horizon robustness over learned-token MGN-T, and substantially lower processor size and runtime than the native higher-order MS-IMGN baseline. These results demonstrate that mesh-derived structured tokenization provides an effective interface for higher-order graph simulation with efficient global transformer processing.

AI USE STATEMENT

In this work, we used generative AI tools for literature research, conceptual design, code implementation and optimization. Additionally, we used generative AI tools for improving wording, grammar and spell checking.
We have reviewed all AI-assisted work, including LLM-generated research ideas and related works through a manual literature survey. All LLM-generated code was verified and tested.
We have not used generative AI tools for data generation or processing, numerical data analysis, model evaluation and qualitative or quantitative result assessment. All conclusions and claims are made based on evaluation by the authors.
We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide anonymized source code, experiment configurations, and instructions for training and evaluating the models used in the key experiments. The experiments use the datasets of the IMGN benchmark as described in Sebestyen & Kim (2025). Due to their size and source, the datasets are not redistributed with the submission but may be requested from the authors of the cited work. Model equations, algorithmic details, and implementation procedures are provided in Section 2 and the Appendix A.

REFERENCES

Klaus-Jürgen Bathe. *Finite element procedures*. Prentice Hall, Upper Saddle River, NJ, 1996. ISBN 0-13-301458-4.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew M. Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL http://arxiv.org/abs/1806.01261.

Yadi Cao, Menglei Chai, Minchen Li, and Chenfanfu Jiang. Bi-stride multi-scale graph neural network for mesh-based physical simulation, 2023.

Saurabh Deshpande, Stéphane P.A. Bordas, and Jakub Lengiewicz. Magnet: A graph u-net architecture for mesh-based simulations. *Engineering Applications of Artificial Intelligence*, 133:108055, 2024. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2024.108055. URL https://www.sciencedirect.com/science/article/pii/S0952197624002136.

Rui Gao, Indu Kant Deo, and Rajeev K. Jaiman. A finite element-inspired hypergraph neural network: Application to fluid dynamics simulations, 2023.

Paul Garnier, Vincent Lannelongue, Jonathan Viquerat, and Elie Hachem. Training transformers for mesh-based simulations, 2025. URL https://arxiv.org/abs/2508.18051.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017. URL http://arxiv.org/abs/1704.01212.

Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964. doi: 10.1214/aoms/1177703732. URL https://doi.org/10.1214/aoms/1177703732.

Mikel M. Iparraguirre, Iciar Alfaro, David Gonzalez, and Elias Cueto. Meshgraphnet-transformer: Scalable mesh-based learned simulation for solid mechanics, 2026. URL https://arxiv.org/abs/2601.23177.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Graphcast: Learning skillful medium-range global weather forecasting, 2023.

Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *CoRR*, abs/2010.08895, 2020a. URL https://arxiv.org/abs/2010.08895.

Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *CoRR*, abs/2003.03485, 2020b. URL https://arxiv.org/abs/2003.03485.

Zongyi Li, Hongkai Zheng, Nikola B. Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *CoRR*, abs/2111.03794, 2021. URL https://arxiv.org/abs/2111.03794.

Mario Lino, Chris Cantwell, Anil A. Bharath, and Stathi Fotiadis. Simulating continuum mechanics with multi-scale graph neural networks, 2021.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *CoRR*, abs/1910.03193, 2019. URL http://arxiv.org/abs/1910.03193.

Mohammad Amin Nabian, Chang Liu, Rishikesh Ranade, and Sanjay Choudhry. X-meshgraphnet: Scalable multi-scale graph neural networks for physics simulation, 2024. URL https://arxiv.org/abs/2411.17164.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Luca Patrignani and Silvestre T. Pinho. Graph neural networks with hybrid local-global attention for effective prediction of mechanical response in structures. *Computer Methods in Applied Mechanics and Engineering*, 452:118753, 2026. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2026.118753. URL https://www.sciencedirect.com/science/article/pii/S0045782526000277.

Roberto Perera and Vinamra Agrawal. Multiscale graph neural networks with adaptive mesh refinement for accelerating mesh-based simulations. *Computer Methods in Applied Mechanics and Engineering*, 429:117152, 2024. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2024.117152. URL https://www.sciencedirect.com/science/article/pii/S0045782524004080.

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

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer, 2023. URL https://arxiv.org/abs/2205.12454.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. *CoRR*, abs/2002.09405, 2020. URL https://arxiv.org/abs/2002.09405.

Jozsef Adam Sebestyen and Do-Nyun Kim. Interpolation mesh-graph net using disconnected spatial graphs of higher order element based meshes. *Engineering Applications of Artificial Intelligence*, 159:111535, 2025. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2025.111535. URL https://www.sciencedirect.com/science/article/pii/S0952197625015374.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation?, 2021. URL https://arxiv.org/abs/2106.05234.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. Graph transformer networks. *CoRR*, abs/1911.06455, 2019. URL http://arxiv.org/abs/1911.06455.

A Appendix

Lowest mean RMSE values are indicated with bold font in the following tables.

Table 2: Linear elastostatics RMSE-1 by Element Order (E. O.).

E.O.	**MGN-T**	**IMGTT**	**MGN**	**IMGN**
Position mean RMSE-1 and standard deviation ($\times 10^5$).				
1	0.484 ± 0.763	0.419 ± 0.688	$\mathbf{0.241 \pm 0.280}$	0.250 ± 0.335
2	0.594 ± 0.795	0.489 ± 0.804	0.638 ± 1.042	$\mathbf{0.227 \pm 0.295}$
3	1.250 ± 2.028	0.338 ± 0.475	1.339 ± 2.178	$\mathbf{0.212 \pm 0.247}$
4	0.898 ± 1.494	0.426 ± 0.581	1.892 ± 2.434	$\mathbf{0.217 \pm 0.291}$
5	1.358 ± 2.185	0.641 ± 1.014	2.589 ± 2.997	$\mathbf{0.231 \pm 0.301}$
Stress mean RMSE-1 and standard deviation.				
1	2.441 ± 2.494	2.530 ± 2.704	$\mathbf{1.059 \pm 1.107}$	1.133 ± 1.306
2	3.322 ± 3.445	3.393 ± 3.519	1.652 ± 1.721	$\mathbf{1.170 \pm 1.415}$
3	4.406 ± 4.874	3.458 ± 3.671	6.087 ± 6.368	$\mathbf{1.149 \pm 1.413}$
4	5.200 ± 6.329	3.107 ± 3.497	15.48 ± 14.29	$\mathbf{1.124 \pm 1.265}$
5	5.709 ± 6.485	3.731 ± 4.159	23.420 ± 19.170	$\mathbf{1.179 \pm 1.498}$

Table 3: Flag dynamics RMSE-1 by Element Order (E. O.).

E.O.	MGN-T	IMGTT	MGN	MS-IMGN
Position mean RMSE-1 and standard deviation ($\times 10^3$).				
1	1.062 ± 0.409	1.031 ± 0.383	0.628 ± 0.120	$\mathbf{0.610 \pm 0.116}$
2	0.862 ± 0.308	0.822 ± 0.275	0.494 ± 0.085	$\mathbf{0.484 \pm 0.080}$
3	0.745 ± 0.250	0.703 ± 0.230	0.424 ± 0.076	$\mathbf{0.417 \pm 0.067}$
4	0.675 ± 0.218	0.639 ± 0.201	0.388 ± 0.072	$\mathbf{0.382 \pm 0.061}$
5	0.616 ± 0.201	0.589 ± 0.182	0.369 ± 0.076	$\mathbf{0.355 \pm 0.063}$
Velocity mean RMSE-1 and standard deviation ($\times 10$).				
1	0.521 ± 0.309	0.515 ± 0.295	0.513 ± 0.165	$\mathbf{0.504 \pm 0.160}$
2	0.464 ± 0.244	0.454 ± 0.230	0.430 ± 0.115	$\mathbf{0.420 \pm 0.111}$
3	0.428 ± 0.214	0.418 ± 0.200	0.387 ± 0.098	$\mathbf{0.376 \pm 0.092}$
4	0.416 ± 0.204	0.406 ± 0.189	0.379 ± 0.089	$\mathbf{0.366 \pm 0.081}$
5	0.398 ± 0.197	0.387 ± 0.179	0.367 ± 0.093	$\mathbf{0.349 \pm 0.082}$
Acceleration mean RMSE-1 and standard deviation.				
1	6.237 ± 4.138	$\mathbf{6.158 \pm 3.957}$	7.644 ± 2.651	7.505 ± 2.586
2	5.690 ± 3.248	$\mathbf{5.601 \pm 3.085}$	6.408 ± 1.837	6.258 ± 1.798
3	5.357 ± 2.881	$\mathbf{5.258 \pm 2.705}$	5.817 ± 1.554	5.620 ± 1.481
4	5.342 ± 2.820	$\mathbf{5.225 \pm 2.636}$	5.828 ± 1.382	5.592 ± 1.288
5	5.184 ± 2.771	$\mathbf{5.044 \pm 2.539}$	5.646 ± 1.442	5.345 ± 1.303

Table 4: Flag dynamics RMSE-50 by Element Order (E. O.).

E.O.	MGN-T	IMGTT	MGN	MS-IMGN
Position mean RMSE-50 and standard deviation ($\times 10^3$).				
1	5.562 ± 4.898	$\mathbf{3.629 \pm 4.217}$	24.92 ± 6.699	10.660 ± 4.940
2	15.12 ± 11.94	7.956 ± 7.016	7.115 ± 5.314	$\mathbf{4.236 \pm 4.487}$
3	23.70 ± 23.35	$\mathbf{5.632 \pm 5.529}$	8.922 ± 5.717	6.219 ± 4.799
4	24.32 ± 20.89	8.769 ± 9.130	9.129 ± 4.540	$\mathbf{5.881 \pm 4.379}$
5	18.67 ± 16.49	$\mathbf{6.236 \pm 6.651}$	9.704 ± 4.928	6.446 ± 5.017
Velocity mean RMSE-50 and standard deviation ($\times 10$)				
1	1.100 ± 0.821	$\mathbf{0.931 \pm 0.788}$	2.073 ± 0.889	1.231 ± 0.708
2	1.671 ± 1.221	1.097 ± 0.862	1.101 ± 0.755	$\mathbf{0.898 \pm 0.717}$
3	2.101 ± 1.816	$\mathbf{0.897 \pm 0.841}$	1.142 ± 0.761	0.909 ± 0.755
4	2.198 ± 1.666	1.193 ± 1.142	1.139 ± 0.681	$\mathbf{0.930 \pm 0.775}$
5	1.829 ± 1.500	$\mathbf{0.924 \pm 0.913}$	1.294 ± 0.757	0.948 ± 0.818
Acceleration mean RMSE-50 and standard deviation.				
1	12.92 ± 7.374	11.18 ± 6.638	12.29 ± 6.058	$\mathbf{9.699 \pm 4.912}$
2	13.49 ± 8.360	10.04 ± 6.126	10.24 ± 4.862	$\mathbf{8.480 \pm 4.338}$
3	11.35 ± 7.151	7.988 ± 5.293	9.492 ± 5.029	$\mathbf{7.592 \pm 4.848}$
4	11.45 ± 6.737	9.912 ± 7.256	9.575 ± 5.312	$\mathbf{7.979 \pm 4.855}$
5	9.280 ± 7.076	$\mathbf{7.094 \pm 5.623}$	11.04 ± 6.938	7.723 ± 5.049

Table 5: Flag dynamics RMSE-all by Element Order (E. O.).

E.O.	MGN-T	IMGTT	MGN	MS-IMGN
Position mean RMSE-all and standard deviation ($\times 10^3$).				
1	31.13 ± 30.81	$\mathbf{25.14 \pm 23.47}$	35.74 ± 11.86	34.59 ± 25.32
2	37.74 ± 37.88	31.52 ± 28.46	33.07 ± 23.74	$\mathbf{23.87 \pm 12.79}$
3	42.19 ± 30.40	24.93 ± 18.73	30.19 ± 15.25	$\mathbf{22.60 \pm 19.58}$
4	53.12 ± 48.19	24.15 ± 19.09	24.02 ± 11.35	$\mathbf{21.62 \pm 20.68}$
5	55.62 ± 108.5	$\mathbf{22.28 \pm 21.44}$	29.27 ± 15.48	28.26 ± 53.99
Velocity mean RMSE-all and standard deviation ($\times 10$).				
1	1.732 ± 1.059	$\mathbf{1.594 \pm 0.925}$	2.250 ± 0.710	1.897 ± 0.619
2	1.960 ± 1.163	1.692 ± 0.995	1.689 ± 0.539	$\mathbf{1.501 \pm 0.517}$
3	2.204 ± 1.448	1.602 ± 0.770	1.837 ± 0.598	$\mathbf{1.379 \pm 0.444}$
4	2.300 ± 1.500	1.589 ± 0.956	1.714 ± 0.582	$\mathbf{1.344 \pm 0.472}$
5	2.262 ± 1.480	$\mathbf{1.456 \pm 0.761}$	2.021 ± 0.724	1.459 ± 0.678
Acceleration mean RMSE-all and standard deviation.				
1	11.82 ± 7.807	12.39 ± 8.405	11.03 ± 4.910	$\mathbf{9.462 \pm 3.838}$
2	14.38 ± 11.16	11.08 ± 7.799	9.680 ± 5.177	$\mathbf{7.516 \pm 2.195}$
3	16.54 ± 10.96	14.35 ± 7.798	13.83 ± 10.77	$\mathbf{7.436 \pm 2.539}$
4	17.75 ± 9.345	14.03 ± 7.458	15.09 ± 10.21	$\mathbf{7.088 \pm 2.527}$
5	15.95 ± 7.631	13.88 ± 6.043	19.29 ± 13.78	$\mathbf{7.016 \pm 2.671}$

Table 6: Fluid dynamics RMSE-1 by Element Order (E. O.).

E.O.	MGN-T	IMGTT	MGN	MS-IMGN
Velocity mean RMSE-1 and standard deviation ($\times 10^2$).				
1	0.588 ± 0.341	$\mathbf{0.558 \pm 0.341}$	0.819 ± 0.280	0.858 ± 0.277
2	$\mathbf{0.534 \pm 0.298}$	0.742 ± 0.381	0.737 ± 0.279	0.822 ± 0.313
3	0.552 ± 0.319	$\mathbf{0.504 \pm 0.261}$	0.640 ± 0.251	0.621 ± 0.240
4	0.586 ± 0.304	0.548 ± 0.310	0.591 ± 0.227	$\mathbf{0.524 \pm 0.192}$
5	0.552 ± 0.316	0.531 ± 0.314	0.573 ± 0.219	$\mathbf{0.484 \pm 0.180}$
Acceleration mean RMSE-1 and standard deviation.				
1	0.064 ± 0.126	$\mathbf{0.059 \pm 0.107}$	0.139 ± 0.118	0.154 ± 0.122
2	$\mathbf{0.054 \pm 0.107}$	0.061 ± 0.138	0.120 ± 0.095	0.117 ± 0.093
3	0.052 ± 0.111	$\mathbf{0.048 \pm 0.095}$	0.108 ± 0.084	0.098 ± 0.076
4	0.055 ± 0.122	$\mathbf{0.049 \pm 0.107}$	0.108 ± 0.084	0.088 ± 0.068
5	0.052 ± 0.118	$\mathbf{0.048 \pm 0.116}$	0.108 ± 0.085	0.081 ± 0.062

Table 7: Fluid dynamics RMSE-50 by Element Order (E. O.).

E.O.	MGN-T	IMGTT	MGN	MS-IMGN
Velocity mean RMSE-50 and standard deviation ($\times 10^2$).				
1	3.429 ± 4.642	$\mathbf{2.993 \pm 3.432}$	3.486 ± 1.463	6.174 ± 2.010
2	4.599 ± 4.572	5.670 ± 7.193	3.142 ± 1.324	$\mathbf{3.079 \pm 1.476}$
3	4.701 ± 4.758	3.719 ± 3.312	3.231 ± 1.052	$\mathbf{2.702 \pm 1.054}$
4	5.404 ± 5.475	5.328 ± 5.616	3.089 ± 1.479	$\mathbf{2.360 \pm 1.016}$
5	5.820 ± 5.627	4.820 ± 5.945	3.442 ± 1.713	$\mathbf{2.244 \pm 0.935}$
Acceleration mean RMSE-50 and standard deviation.				
1	0.354 ± 0.610	$\mathbf{0.276 \pm 0.514}$	0.384 ± 0.224	0.523 ± 0.297
2	0.385 ± 0.538	0.443 ± 0.703	0.338 ± 0.177	$\mathbf{0.307 \pm 0.158}$
3	0.378 ± 0.533	0.333 ± 0.416	0.324 ± 0.172	$\mathbf{0.267 \pm 0.139}$
4	0.429 ± 0.585	0.395 ± 0.521	0.351 ± 0.211	$\mathbf{0.267 \pm 0.173}$
5	0.424 ± 0.581	0.405 ± 0.619	0.382 ± 0.252	$\mathbf{0.247 \pm 0.146}$

Table 8: Fluid dynamics RMSE-all by Element Order (E. O.).

E.O.	MGN-T	IMGTT	MGN	MS-IMGN
Velocity mean RMSE-all and standard deviation ($\times 10^2$).				
1	7.145 ± 9.856	$\mathbf{5.385 \pm 5.092}$	10.52 ± 6.526	18.04 ± 8.218
2	15.45 ± 18.44	13.81 ± 11.53	10.68 ± 5.697	$\mathbf{8.376 \pm 4.796}$
3	28.93 ± 125.9	10.91 ± 11.26	10.14 ± 5.266	$\mathbf{8.281 \pm 5.732}$
4	617.6 ± 3546	18.80 ± 22.06	10.43 ± 6.931	$\mathbf{7.306 \pm 5.488}$
5	53.84 ± 263.8	22.88 ± 28.25	12.21 ± 7.449	$\mathbf{9.544 \pm 5.150}$
Acceleration mean RMSE-all and standard deviation.				
1	0.409 ± 0.964	$\mathbf{0.317 \pm 0.701}$	1.201 ± 1.510	1.064 ± 1.346
2	0.395 ± 0.826	$\mathbf{0.337 \pm 0.692}$	1.085 ± 1.434	0.918 ± 1.216
3	0.490 ± 2.175	$\mathbf{0.335 \pm 0.745}$	1.045 ± 1.361	1.026 ± 1.358
4	5.998 ± 21.97	$\mathbf{0.395 \pm 0.751}$	1.019 ± 1.383	0.910 ± 1.250
5	1.012 ± 3.237	$\mathbf{0.388 \pm 0.707}$	0.943 ± 1.134	0.951 ± 1.294

Table 9: Number of parameters in the processor block of each model and the training and inference step times.

E.O.	MGN-T	IMGTT	MGN	MS-IMGN
Million parameters				
	0.80	1.13	1.99	3.98
Training step time [ms] by element order				
1	35.00	52.88	82.61	156.5
2	43.74	54.66	85.04	167.8
3	50.18	61.80	147.9	205.9
4	79.66	91.13	241.3	306.1
5	108.1	121.3	341.1	414.5
Inference step time [ms] by element order				
1	6.157	8.608	10.61	25.24
2	9.300	12.48	24.51	39.44
3	16.65	20.07	48.49	63.86
4	27.74	31.50	83.60	98.94
5	38.13	42.08	118.5	135.5

Algorithm 1 IMGTT forward pass

Input: fine-node features $\mathbf{X}_f$, fine-edge features $\mathbf{E}_f$, coarse-token positional features $\mathbf{P}_c$

Output: prediction $\hat{\mathbf{Y}}$

$\mathbf{H}_f^0 \leftarrow \mathrm{NodeEncoder}(\mathbf{X}_f)$
$\mathbf{G}_f^0 \leftarrow \mathrm{EdgeEncoder}(\mathbf{E}_f)$
$\mathbf{Z}^0 \leftarrow \mathrm{CoarseInit}(\mathbf{H}_f^0)$
$\mathbf{P}_c \leftarrow \mathrm{PositionalEncoder}(\mathbf{P}_c)$
for $\ell = 0$ to $L-2$ **do**
  $\hat{\mathbf{H}}_f^{\ell+1}, \mathbf{G}_f^{\ell+1} \leftarrow \mathrm{MGNBlock}_\ell(\mathbf{H}_f^\ell, \mathbf{G}_f^\ell)$
  $\mathbf{FC}^{\ell+1} \leftarrow \mathrm{Aggregate}(\hat{\mathbf{H}}_f^{\ell+1})$
  $\hat{\mathbf{Z}}^\ell \leftarrow \mathrm{Fuse}_{\mathrm{coarse}}(\mathbf{Z}^\ell, \mathbf{FC}^{\ell+1})$
  $\widetilde{\mathbf{Z}}^\ell \leftarrow \hat{\mathbf{Z}}^\ell + \mathbf{P}_c$
  $\mathbf{Z}^{\ell+1} \leftarrow \mathrm{Transformer}_\ell(\widetilde{\mathbf{Z}}^\ell)$
  $\mathbf{CF}^{\ell+1} \leftarrow \mathrm{Interpolate}(\mathbf{Z}^{\ell+1})$
  $\mathbf{H}_f^{\ell+1} \leftarrow \mathrm{Fuse}_{\mathrm{fine}}(\hat{\mathbf{H}}_f^{\ell+1}, \mathbf{CF}^{\ell+1})$
end for
$\mathbf{H}_f^L, \mathbf{G}_f^L \leftarrow \mathrm{MGNBlock}_{L-1}(\mathbf{H}_f^{L-1}, \mathbf{G}_f^{L-1})$
$\hat{\mathbf{Y}} \leftarrow \mathrm{Decoder}(\mathbf{H}_f^L)$
return $\hat{\mathbf{Y}}$

Note: the optional world-edges are omitted for simplicity.