M4GN: Micro–Meso–Macro Mesh-based Graph Network for Dynamic Simulations

Anonymous authors
Paper under double-blind review

Abstract

Dynamic systems often exhibit intricate interactions that span from localized, fine-scale processes to broad, global effects. Accurately modeling these systems therefore demands methods that account for both localized dynamics and extended global dependencies while remaining computationally tractable. However, existing surrogate models often struggle to balance precision and scalability, especially for large datasets, complex mesh topologies, and long-range effects. In this paper, we introduce M4GN, a physics-informed hierarchical model designed to address the aforementioned challenges by aligning its framework with the inherent behaviors in dynamic simulations. M4GN comprises three stages: a micro-level stage for fine-grained local dynamics, a macro-level stage for far-reaching global interactions, and a meso-level stage that facilitates effective information exchange between these levels by aligning mesh hierarchy with physical properties. Experimental results show that M4GN achieves superior accuracy, excels at modeling long-range interactions, and maintains high computational efficiency. Moreover, M4GN generalizes well to larger physical domains, making it particularly suitable for complex, large-scale dynamic simulations. All code and data will be released upon acceptance.

1 Introduction

Numerically solving partial differential equations (PDEs) to model dynamic systems is fundamental in science and engineering but is often computationally intensive, especially in time-sensitive applications requiring rapid inference. This has prompted increased attention across various scientific disciplines, ranging from solid mechanics (Haghighat et al., 2021) to quantum physics (Sellier et al., 2019), towards the adoption of learning-based surrogate models (Sun et al., 2020). These models aim to expedite numerical simulations, addressing the computational challenges associated with traditional solvers.

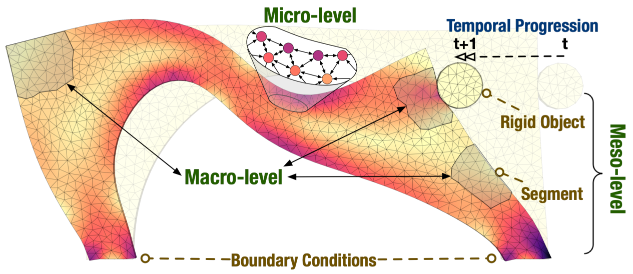


Figure 1: Visualization of multi-scale dynamics—encompassing micro-, meso-, and macro-level interactions—during a solid mechanics simulation from time t to $t+1$. In this scenario, a rigid object is interacting with a hyperelastic beam that is anchored to the ground, resulting in deformation.

Among these surrogate models, mesh-based Graph Neural Networks (GNNs) (Belbute-Peres et al., 2020; Pfaff et al., 2020) have shown promising results in simulating dynamic systems on unstructured meshes but often struggle with accuracy and efficiency on large or complex datasets. Their reliance on deep message-passing can lead to over-smoothing (Chen et al., 2020; Yang et al., 2020) and increased computational costs (Fortunato et al., 2022; Cao et al., 2023). To address these issues, recent techniques propose constructing coarser subgraphs via spatial proximity (Liu et al., 2021; Janny et al., 2023), coarsening (Fortunato et al., 2022; Cao et al., 2023), or random pooling (Li et al., 2020). However, these approaches present significant drawbacks: they may discard essential mesh structures, causing distorted local connectivity and diminished mesh quality; neglect crucial physical properties, resulting in physically inconsistent outcomes; or demand extensive manual intervention, which reduces scalability. These limitations highlight the need for a surrogate model that not only mitigates over-smoothing and high computational overhead but also preserves mesh and physical properties by accounting for behaviors of real-world dynamic systems.

Dynamic systems naturally unfold across multiple scales, progressing from localized (*micro-level*) interactions through transitional (*meso-level*) processes to emergent (*macro-level*) behavior (see Figure 1). Examples span ocean waves (Booij & Holthuijsen, 1987), atmospheric convection (Emanuel, 1994), structural vibrations (Fahy, 2007), and seismic waves (Kennett, 2009), all of which illustrate how fine-scale dynamics ultimately shape large-scale phenomena. At the *micro-level*, small-scale elements—such as currents within fluid flows or microstructural defects in materials—generate localized correlations that can significantly influence broader patterns (Li et al., 2009; Cubuk et al., 2017; Wiewel et al., 2019). As these localized effects propagate and synchronize at the *meso-level*, clusters or segments of interacting components form intermediate-scale structures that link fine-grained details to overarching system trends (Svedin et al., 2005). Over time, these meso-level interactions coalesce into cohesive, system-wide dynamics at the *macro-level*, reflecting robust emergent properties (O'Connor, 2020). Motivated by these hierarchical dynamics, we introduce the Micro–Meso–Macro Mesh-based Graph Network (M4GN), an innovative framework designed to capture multi-scale behaviors in dynamic simulations. By aligning with the intrinsic architecture of real-world systems, M4GN outperforms state-of-the-art methods across multiple evaluation metrics on both solid and fluid datasets. Main contributions of this paper are summarized as follows:

- We propose a novel hierarchical framework that incorporates a micro-level stage for fine-grained local dynamics, a macro-level stage for far-reaching global interactions, and a meso-level stage that enables effective information exchange between these levels by aligning mesh hierarchy with physical properties.
- We demonstrate that M4GN achieves high prediction accuracy and superior mesh quality while maintaining computational efficiency, effectively addressing the common trade-off present in existing methods.
- We present the DeformingBeam dataset and its scaled-up version, providing a comprehensive framework for evaluating mesh-based simulation models; using this dataset, we show that M4GN demonstrates strong scalability to larger and more complex domains.

2 Related Works

2.1 GNNs for Dynamic System Simulation

The application of Graph Neural Networks (GNN) for dynamic system prediction is an emerging research area in scientific machine learning due to their versatility and effectiveness (Mrowca et al., 2018; Belbute-Peres et al., 2020; Rubanova et al., 2021). Unlike image-based learning methods such as Convolutional Neural Networks (CNNs) (Um et al., 2018; Ummenhofer et al., 2019), GNNs can directly handle unstructured simulation meshes, making them well-suited for simulating systems with complex domain boundaries while ensuring spatial invariance and locality (Battaglia et al., 2018; Wu et al., 2020). The initial application of GNNs to physics-based simulations focused on deformable solids and fluids, with MeshGraphNets (MGN) being a pioneering work in this area (Pfaff et al., 2020). Building on this foundation, various MGN variants have been proposed: integrating GNNs with Physics-Informed Neural Networks (PINNs) (Gao et al., 2022), enabling long-term predictions by combining GraphAutoEncoder (GAE) and Transformer models (Han et al., 2022), and directly predicting steady states through multi-layer readouts (Harsch & Riedelbauch, 2021).

2.2 Hierarchical Models in GNNs for Long-range Dynamic Propagation

To mitigate the over-smoothing issue (Li et al., 2018) in GNNs when applied to large or complex datasets, several hierarchical models have been introduced recently. These hierarchical models can be categorized into two types. The first type includes dual-level structures. For instance, GMR-GMUS (Han et al., 2022) utilizes a pooling method to select pivotal nodes through uniform sampling. Similarly, the EAGLE (Janny et al., 2023) employs a clustering-based pooling method along with transformer mechanism, showing promising performance in fluid dynamics. MS-MGN (Fortunato et al., 2022) proposes a dual-layer framework that passes messages at both fine and coarse resolutions for mesh-based simulation learning. The second type encompasses multi-level structures. One such model, BSMS-GNN (Cao et al., 2023), analyzes limitations of existing pooling strategies and introduces a bi-stride pooling method using breadth-first search (BFS) to select nodes. (Yu et al., 2023) propose a similar hierarchical structure as (Cao et al., 2023) but with two different transformers to enable long-range interactions.

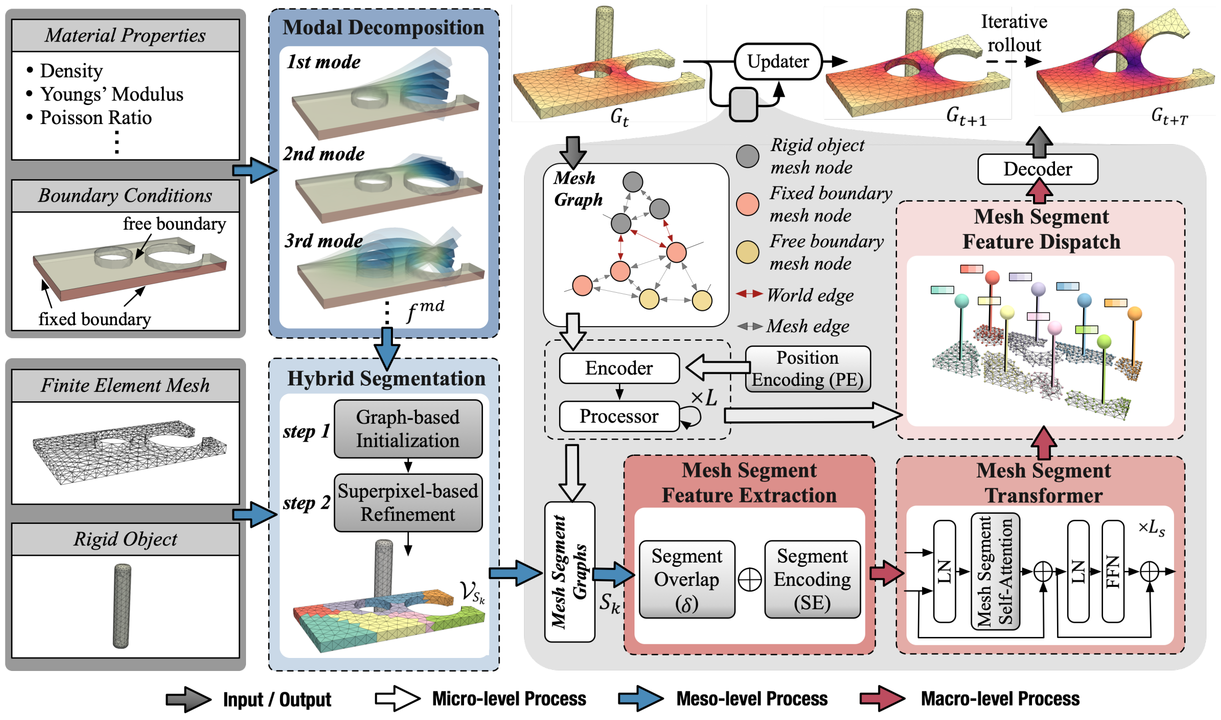


Figure 2: Micro-Meso-Macro Mesh-based Graph Network (M4GN)

3 Methodology

This section introduces the M4GN framework (see Figure 2), beginning with a concise formulation of the problem (Section 3.1). The framework itself is divided into three key stages. First, Micro-level Information Exchange (Section 3.2) focuses on fine-grained local dynamics and how individual nodes interact within the mesh. Next, Meso-level Information Alignment (Section 3.3) explains the process of synchronizing mesh structures with the underlying physical properties. Finally, Macro-level Information Exchange (Section 3.4) addresses large-scale global interactions that dictate emergent system behaviors.

3.1 Problem Definition

Let $G = (\mathcal{V}, \mathcal{E})$ be a mesh graph with $\mathcal{V}$ being the set of nodes and $\mathcal{E}$ being the set of edges. The graph has $N = |\mathcal{V}|$ nodes and $E = |\mathcal{E}|$ edges, with adjacency matrix $A \in \mathbb{R}^{N\times N}$ represents graph connectivity. The dynamic system simulation task is to learn a forward model of the dynamic quantities of the mesh graph at next time step $\hat{G}_{t+1}$ given the current mesh graph G_t and (optionally) a history of previous mesh graphs $\{G_{t-1}, \ldots, G_{t-h}\}$. Finally, the rollout trajectory can be generated through the simulator iteratively based on the previous prediction: $G_t, \hat{G}_{t+1}, \ldots, \hat{G}_{t+T}$, where T is the total simulation steps. In this paper, the proposed model (M4GN) can simulate both Eulerian and Lagrangian systems (Bontempi & Faravelli, 1998). In Eulerian systems, which model the evolution of continuous fields like velocity over a fixed mesh, the graph $\mathcal{E}$ includes only mesh-related edges $\mathcal{E}^M$. Conversely, in Lagrangian systems, where the mesh represents a moving and deforming surface or volume, additional world edges $\mathcal{E}^W$ are incorporated into the graph. These edges enable the model to learn external dynamics such as collision and contact. The node features of node i are denoted by $\mathbf{x}_i$, while the features for an edge between node i and j are indicated by $\mathbf{e}_{ij}$.

3.2 Micro-level Information Exchange

In the micro-level information exchange stage, each node engages in the exchange of information with its neighboring nodes. This process holds particular significance in dynamic systems, where the behavior of adjacent nodes is closely intertwined (Booij & Holthuijsen, 1987; Emanuel, 1994; Fahy, 2007; Kennett, 2009). Furthermore, the micro-level exchange module serves a crucial role in addressing discontinuities that may arise at the boundaries of adjacent mesh segments (Lai et al., 2009). By prioritizing micro-level information exchange, we effectively mitigate discontinuities introduced by subsequent macro-level operations.

We adopt the Encoder-Process-Decoder (EPD) (Pfaff et al., 2020) network structure for our micro-level information exchange as it has shown superior performance in dealing with mesh-based graphs. For a given graph G_t at time t, the model begins with extracting node and edge features through two separate Multi-Layer Perceptrons (MLPs):

$$\mathbf{h}^0_{i,t} = f_n(\mathbf{x}_{i,t}), \quad \mathbf{h}^{M,0}_{ij,t} = f^M_e(\mathbf{e}^M_{ij,t}), \quad \mathbf{h}^{W,0}_{ij,t} = f^W_e(\mathbf{e}^W_{ij,t}), \tag{1}$$

where $\mathbf{x}_{i,t}$, $\mathbf{e}^M_{ij,t} \in \mathcal{E}^M$, and $\mathbf{e}^W_{ij,t} \in \mathcal{E}^W$ denote node feature, mesh edge feature, and world edge feature vector at time t, respectively. For Lagrangian systems, world edges are created by spatial proximity, where for a fixed radius r_W, a world edge is added between nodes i and j when $|\mathbf{x}_i - \mathbf{x}_j| < r_W$, excluding node pairs already connected in the mesh. The outputs of two MLPs (i.e. f_n and f_e) for node and edge are denoted as $\mathbf{h}^0_{i,t}$ and $\mathbf{h}^0_{ij,t}$, respectively. Then, a L-step message passing (MP) is performed such that each node can receive and aggregate information from neighboring nodes within L steps of edge traversing. For each MP from 1 to L, the node and edge representations are updated as:

$$\mathbf{h}^l_{i,t} = f^l_n(\mathbf{h}^{l-1}_{i,t}, \sum_{j \in Adj(i)} \mathbf{h}^{M,l-1}_{ij,t}, \sum_{j \in Adj(i)} \mathbf{h}^{W,l-1}_{ij,t}), \tag{2}$$

$$\mathbf{h}^{M,l}_{ij,t} = f^l_e(\mathbf{h}^{M,l-1}_{ij,t}, \mathbf{h}^{l-1}_{i,t}, \mathbf{h}^{l-1}_{j,t}), \tag{3}$$

$$\mathbf{h}^{W,l}_{ij,t} = f^l_e(\mathbf{h}^{W,l-1}_{ij,t}, \mathbf{h}^{l-1}_{i,t}, \mathbf{h}^{l-1}_{j,t}), \tag{4}$$

where $Adj(i)$ denotes all adjacent nodes of node i. Up until this point, the node and edge information of the graph G_t are updated. Additionally, we implement a technique from (Godwin et al., 2021), which involves corrupting the input graph with noise and adding a noise-correcting node-level loss. We evaluate the impact of varying the number of message passing steps during micro-level information exchange step, where details can be found in Appendix D.1.

3.3 Meso-level Information Alignment

In the meso-level information alignment stage, the model facilitates interactions among clusters of nodes, capturing intermediate-scale structures that bridge local and global dynamics. This level is crucial for representing transitional processes where groups of interacting components form coherent substructures within the mesh.

3.3.1 Mathematical Notation

We define the segmentation policy $\pi(G) = f_s(G, I)$, where the segmentation function f_s takes the input graph G and prior physical information I (e.g., boundary conditions, material properties), and outputs a set of graph segments $\{S^0_1, S^0_2 \ldots, S^0_K\}$. The superscript 0 denotes non-overlapping segmentation. For each segment $S_k = (\mathcal{V}_{S_k}, \mathcal{E}_{S_k})$, the set of nodes $\mathcal{V}_{S_k} \subseteq \mathcal{V}$ and $\mathcal{E}_{S_k} \subseteq \mathcal{E}$ are subsets of the original graph G. The union of all segments reconstructs the original graph, such that $\mathcal{V} = \cup \mathcal{V}^0_{S_k}$ and $\mathcal{E} = \cup \mathcal{E}^0_{S_k}$.

In some cases, it may be beneficial to allow for overlapping segments, where nodes in $\mathcal{V}$ can belong to more than one segment. This overlap helps create smoother transitions between segments and reduces discontinuities at segment boundaries. We define the overlap amount by $\delta \in \mathbb{N}$, with $\delta = 0$ representing no overlap. For $\delta > 0$, the node set $\mathcal{V}_{S_k}^{\delta}$ is defined recursively as as $\mathcal{V}_{S_k}^{\delta} = \mathcal{V}_{S_k}^{\delta-1} \cup \{Adj(i) \mid i \in \mathcal{V}_{S_k}^{\delta-1}\}$. To simplify the presentation, we disregard the superscript δ in the remainder of this paper. The effect of adding overlapping segments is discussed in our ablation study, as shown in Table 4.

3.3.2 Modal Decomposition

Modal decomposition is a fundamental technique for extracting dominant spatiotemporal patterns, or *modes*, from complex physical systems (Fu & He, 2001; Schmid et al., 2011; Taira et al., 2017). Each mode encapsulates coherent behavior—such as a characteristic deformation shape or flow structure—allowing a reduced but meaningful representation of the underlying dynamics. In complex physical simulations, these dominant modes can effectively guide downstream tasks such as mesh segmentation, where the domain is subdivided based on physical coherence (Yang et al., 2016; Huang et al., 2009). In this work, we employ two different modal decomposition approaches to address *solid* and *fluid* problems separately, given their distinct physical behaviors (Bathe, 2001). The pseudo code of the modal decomposition module can be found in Algorithm 1.

Structural Modal Analysis: For solids, the decomposition naturally arises from the mass–stiffness relationship in elastodynamics, capturing genuine dynamic displacements (Andersen, 2006). Let $\mathbf{K}$ be the global stiffness matrix and $\mathbf{M}$ the global mass matrix arising from finite element assembly. The free vibration modes of a structure are obtained by solving the generalized eigenvalue problem:

$$\mathbf{K}\boldsymbol{\phi} \;=\; \lambda \mathbf{M}\boldsymbol{\phi}, \tag{5}$$

where λ represents the square of the natural frequency, and $\boldsymbol{\phi} = (\phi_1, \phi_2, \dots, \phi_{\text{dim}})$ is the corresponding *structural modes*, whose dimension (dim) matches the number of displacement components (e.g., 2D or 3D). Physically, each mode shape indicates a fundamental deformation pattern under vibrational motion (Fu & He, 2001; Wilson, 2002), which tied to the solid's geometry, boundary conditions, and material parameters. In practice, it is typical to selects the first m modes ($\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_m$) to construct an m-dimensional feature at each mesh node i: $f_i^{md} \;=\; \big(\boldsymbol{\phi}_1(i), \boldsymbol{\phi}_2(i), \dots, \boldsymbol{\phi}_m(i)\big)$.

Laplacian Eigenfunctions: In fluid contexts, particularly when lacking multiple snapshots or a steady base flow (Wang et al., 2024), Laplacian eigenfunctions (Grebenkov & Nguyen, 2013) are used to capture geometry- and boundary-driven harmonic modes by solving:

$$-\nabla^2\phi \;=\; \lambda\,\phi, \text{ subject to boundary constraints,} \tag{6}$$

yielding *harmonic modes* $\phi_1, \dots, \phi_m$. These modes serve as a practical proxy for flow-related structures, providing a minimal but informative decomposition that respects the domain shape and boundary conditions (De Witt et al., 2012; Taira et al., 2017). Similar to the solid case, each node i in the fluid mesh is associated with a feature vector:$f_i^{md} \;=\; \big(\phi_1(i), \phi_2(i), \dots, \phi_m(i)\big)$.

3.3.3 Hybrid Segmentation

As discussed in Section 1, to avoid the uninterpretable and potentially erroneous dynamics that coarsened graphs or added edges might introduce, we propose preserving the original mesh structure and facilitating long-range information exchange through communication between segmented mesh graphs. By leveraging dominant modes identified in the modal decomposition module (Section 3.3.2), we ensure these segments remain physically coherent and well-structured for effective communication. Grouping elements with similar physical properties enhances model convergence by minimizing discontinuities within each segment (Diao et al., 2023), while grouping nodes with similar behaviors streamlines learning and ensures uniform handling of similar interactions (Dolean et al., 2015).

Overview: Traditional graph segmentation methods (Alpert & Yao, 1995; Delingette, 1999) often prioritize geometric properties and computational efficiency over underlying physical attributes. Conversely, superpixel approaches (Veksler et al., 2010; Achanta et al., 2012) group pixels based on user-defined similarity measures but rely on careful cluster-center initialization to maintain segmentation quality. To merge the strengths of both, we apply a *graph-based method* (f_{gb}) for initial mesh segmentation and refine it using a *superpixel-based* *method* (f_{sb}), guided by modal decomposition features. This hybrid approach offers efficient geometric partitioning alongside adaptive, feature-based refinement, producing high-quality mesh segments adaptable to diverse dynamic systems.

Detailed Methodology: In the hybrid segmentation module, we first use METIS (Karypis & Kumar, 1998) for initial mesh segmentation due to its great balance of partition quality and speed. Formally, given a graph G, the partition function f_{gb} will split it into K non-overlapped mesh-segment graphs: $\{S_1, \dots, S_K \mid S_i \cap S_j = \varnothing, \forall i \neq j\} = f_{gb}(G)$. Then, we apply SLIC (Achanta et al., 2012), the state-of-the-art superpixel-based clustering methods, to these mesh segments to iteratively update the segmentation centroids $\{C_1, \dots, C_K\}$ and corresponding node assignments using information obtained from modal decomposition. It is worth to note that standard modal decomposition does not account for external obstacles (Fu & He, 2001). Therefore, in models with moving rigid objects, these information will need to be incorporated separately.

For node i in graph G, we represent it by its spatial coordinates $\mathbf{x}_i$, features related to rigid object or obstacle f_i^{obs}, and features obtained from modal decomposition f_i^{md}. For a given mesh segment S_k containing $|\mathcal{V}_{S_k}|$ nodes, we define its centroid C_k as its mean value along the features:

$$C_k = [\mathbf{x}_{C_k}, f_{C_k}^{obs}, f_{C_k}^{md}]^T = \frac{1}{|\mathcal{V}_{S_k}|} \sum_{i \in \mathcal{V}_{S_k}} [\mathbf{x}_i, f_i^{obs}, f_i^{md}]^T. \tag{7}$$

Within each iteration, we improve the mesh segmentation by minimizing a distance measure that considers both physical similarity and spatial proximity. The distance measure $d(i, C_k)$ between a node $i \in \mathcal{V}$ and a segment's centroid C_k is defined as:

$$d(i, C_k) = \|f_i^{obs} - f_{C_k}^{obs}\| + \|f_i^{md} - f_{C_k}^{md}\| + \tau\|\mathbf{x}_i - \mathbf{x}_{C_k}\|, \tag{8}$$

where τ is used to control the compactness of a mesh segment.

The pseudo code of the hybrid segmentation module can be found in Algorithm 2. In Appendix B.4, we present a comprehensive comparison of various segmentation methods and their variants based on different distance measures. Additionally, we evaluate the impact of varying the number of mesh segments on model performance in Appendix D.3 and Appendix E.2. We also introduce several metrics to measure quality of different mesh segmentation, specifically to understand the intra-segment and inter-segment characteristics, which can be found in Appendix C.3

3.4 Macro-level Information Exchange

At the macro-level information exchange stage, the model captures long-range dependencies and global interactions that govern the emergent behavior of the entire system. This stage is essential for understanding large-scale trends and patterns that arise from the collective dynamics of the system's components (Svedin et al., 2005). The macro-level module aggregates information from the meso-level, allowing the model to synthesize comprehensive system-wide insights and make predictions that reflect the overall state of the dynamic system.

3.4.1 Mesh Segment Feature Extraction

Segment Encoding (SE) – In order to extract a global feature for each mesh segment, we perform average pooling on all node vectors in S_k and apply a MLP (f_s) to get the fixed-sized segment embedding:

$$\mathbf{h}_{S_k,t} = f_s(\frac{1}{|\mathcal{V}_{S_k}|} \sum_{i \in \mathcal{V}_{S_k}} \mathbf{h}_{i,t}^L). \tag{9}$$

Position Encoding (PE) – As dynamic effect propagates continuously over mesh domains, knowing relative location among segments could provide extra information for next-step macro-level information exchange and increase expressivity of the network. Mathematically, for each pair of mesh segment graph, $\{S_i, S_j\}$, their relative positional information can be obtained through segment-level adjacency matrix $A^K \in \mathbb{R}^{K \times K}$:

$$A^K_{S_i S_j} = |\mathcal{V}_{S_i} \cap \mathcal{V}_{S_j}| = \mathrm{Cut}(\mathcal{V}_{S_i}, \mathcal{V}_{S_j}) = \sum_{m \in \mathcal{V}_{S_i}} \sum_{n \in \mathcal{V}_{S_j}} A_{mn}, \tag{10}$$

where $\mathrm{Cut}(\cdot)$ is a graph operator that counts the number of connection edges between node clusters in mesh segment graph S_i and S_j. We follow the strategy in (Rampášek et al., 2022) that uses random-walk structural encoding (RWSE) (Dwivedi et al., 2021) for PE calculation. Then the PE for the k-th segment, denoted as $\mathbf{p}_{S_k,t}$, is processed through an MLP layer (f_{sp}) and then added to update the SE from Eq (9) as follows: $\mathbf{h}_{S_k,t} \leftarrow \mathbf{h}_{S_k,t} + f_{sp}(\mathbf{p}_{S_k,t})$.

We can further enhance the network's expressivity by adding absolute PE to the graph nodes. We use an MLP (f_{np}) to process each node's PE ($\mathbf{p}_{i,t}$), calculated with a similar approach as segment level, and add it to the input node feature. Thus, Eq (1) becomes $\mathbf{h}^0_{i,t} = f_n(\mathbf{x}_{i,t} + f_{np}(\mathbf{p}_{i,t}))$. By incorporating node PE directly into the input features, these features participate in the micro-level information exchange described in Section 3.2, potentially improving the continuity of the extracted mesh segment features. Table 5 presents ablation studies show how adding or omitting PE affects prediction results.

3.4.2 Mesh Segment Transformer

We construct a fully connected mesh segment graph, where the i-th mesh segment feature is represented by $\mathbf{h}_{S_i}$. Note that since the transformer operates on mesh segments rather than individual mesh nodes, and the total number of mesh segments (K) is significantly smaller than the total number of mesh nodes (N), the computational cost of our transformer is substantially reduced compared to a traditional graph transformer that operates on graph nodes (i.e. $O(K^2) \ll O(N^2)$). The l-th block of the mesh segment transformer layer is defined as follows:

$$\mathbf{a}^{k,l}_{S_i S_j} = \mathrm{softmax}_{S_j}\left(\frac{\mathbf{Q}^{k,l}_h \mathrm{LN}(\mathbf{h}^l_{S_i}) \cdot \mathbf{K}^{k,l}_h \mathrm{LN}(\mathbf{h}^l_{S_j})}{\sqrt{d_h}}\right), \tag{11}$$

$$\bar{\mathbf{h}}^l_{S_i} = \|^H_{k=1} \sum_{j=1}^{K} \mathbf{a}^{k,l}_{S_i S_j} (\mathbf{V}^{k,l}_h \mathrm{LN}(\mathbf{h}^l_{S_j})), \tag{12}$$

$$\mathbf{h}^{l+1}_{S_i} = \mathbf{h}^l_{S_i} + \mathbf{O}^l_h \bar{\mathbf{h}}^l_{S_i} + \mathrm{FFN}^l_h(\mathrm{LN}(\mathbf{h}^l_{S_i} + \mathbf{O}^l_h \bar{\mathbf{h}}^l_{S_i})), \tag{13}$$

where $\mathbf{a}^{k,l}_{S_i S_j}$ is self-attention weight between S_i and S_j. $\mathbf{Q}^{k,l}_h, \mathbf{K}^{k,l}_h, \mathbf{V}^{k,l}_h \in \mathbb{R}^{d_h \times d}$ are trainable parameters, and $\mathbf{O}^l_h \in \mathbb{R}^{d \times d}$ is the learned output project matrix. $k = 1$ to H denotes the number of attention heads, and $\|$ denotes concatenation. d_h is the dimension of mesh segment feature for each head, and d is the input and output dimension. We adopt a Pre-Layer Norm architecture (Xiong et al., 2020), which is denoted as $\mathrm{LN}(\cdot)$, and the point-wise Feed Forward Network is represented as $\mathrm{FFN}(\cdot)$. The mesh segment transformer module facilitates information exchange among all mesh segments, updating the feature of each segment $\mathbf{h}_{S_i}$ after passing through L_S mesh segment transformer blocks.

3.4.3 Mesh Segment Feature Dispatch and Training

The mesh segment feature dispatch module (as shown in Figure 2) integrates information obtained from both macro-level and micro-level exchanges. Specifically, the final feature for node i at time step t is updated as $\mathbf{h}_{i,t} \leftarrow [\mathbf{h}_{i,t}, \mathbf{h}_{S_i,t}]$ where $i \in \mathcal{V}_{S_i}$. This ensures that each node incorporates information from both neighboring mesh nodes and spatially distant, yet correlated regions. Finally, we train our dynamics model by supervising on the per-node output features $\hat{\mathbf{x}}_{i,t+1}$, produced by feeding $\mathbf{h}_{i,t}$ into a MLP-based decoder, using a L_2 loss between $\hat{\mathbf{x}}_{i,t+1}$ and the corresponding ground truth values $\mathbf{x}_{i,t+1}$.

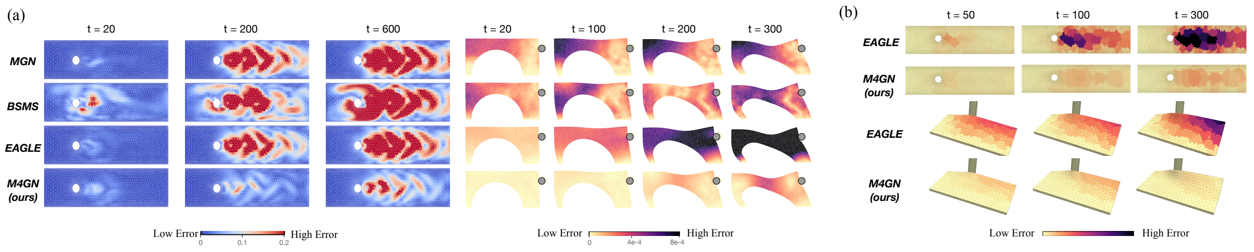


Figure 3: (a) Comparison of prediction results across different models, with each node in the plots color-coded according to its RMSE error over t-step rollouts. Result shows that M4GN achieves notably lower RMSE errors in areas where other methods struggle, particularly at later time steps and in regions further from the inlet or contact point; (b) Visualization of how segmentation aligns with system dynamics, where meshes are colored based on the average prediction error within each segment. Our approach consistently produces uniform segment colors across time steps, demonstrating that nodes within each segment exhibit similar dynamic behaviors and that the segments maintain high continuity.

Table 1: Comparison of results with state-of-the-art methods across three datasets, where each model is trained independently for each dataset. Prediction accuracy is evaluated using Root Mean Square Error (RMSE), with the output being the 2D velocity field for CylinderFlow and the 3D position for DeformingBeam and DeformingPlate. Errors are reported for 1-step rollout, 50-step rollouts, and the entire trajectory. Mesh quality is assessed using four different metrics. Results are averaged over three experiments with different random seeds and presented as mean and standard deviation.

		Mesh Quality Metrics ↓				Prediction Error Metrics ↓		
Dataset	Model	GF$_h$ ($\times 10^{-3}$)	GF$_c$ ($\times 10^{-6}$)	MC ($\times 10^{-3}$)	AR ($\times 10^{-3}$)	RMSE-1 ($\times 10^{-5}$)	RMSE-50 ($\times 10^{-4}$)	RMSE-all ($\times 10^{-4}$)
Cylinder Flow	GCN	-	-	-	-	764 ± 32	425 ± 82	1887 ± 358
	g-U-Net	-	-	-	-	423 ± 4	199 ± 37	843 ± 141
	MGN	-	-	-	-	274 ± 15	64.4 ± 3.4	481 ± 53
	BSMS-GNN	-	-	-	-	**202 ± 24**	280 ± 9	1373 ± 90
	EAGLE	-	-	-	-	507 ± 25	71.5 ± 3.2	583± 29
	M4GN (Ours)	-	-	-	-	320 ± 29	**63.6 ± 2.6**	**372 ± 27**
Deforming Plate	GCN	24.0 ± 0.6	323 ± 4	11.0 ± 0.3	9.33 ± 0.57	34.8 ± 0.6	26.1 ± 0.1	169 ± 1
	g-U-Net	36.1 ± 8.5	452± 125	20.1 ± 0.5	12.4 ± 4.3	41.2 ± 0.2	30.4 ± 0.8	179 ± 7
	MGN	12.7 ± 0.9	248 ± 12	9.25 ± 0.39	5.34 ±0.26	**22.8 ± 0.2**	20.0 ± 0.4	147 ± 3
	BSMS-GNN	23.8 ± 2.6	170 ± 13	18.3 ± 4.4	15.4 ± 5.9	30.3 ± 5.6	23.7 ± 3.5	118 ± 4
	EAGLE	6.75 ± 0.8	41.1 ± 2.6	5.56 ± 0.12	3.31 ± 0.04	36.4 ± 5.2	5.63 ± 1.7	38.7 ± 1.8
	M4GN (Ours)	**4.29 ± 0.07**	**7.05 ± 1.05**	**4.82 ± 0.06**	**2.67 ± 0.06**	26.7 ± 0.5	**3.03 ± 0.16**	**26.5 ± 2.4**
Deforming Beam	GCN	4.91 ± 0.36	3.53 ± 0.51	54.8 ± 8.2	69.5 ± 3.8	7.25 ± 0.12	5.08 ± 0.11	30.7 ± 4.1
	g-U-Net	4.91 ± 0.50	3.55 ± 0.73	34.7 ± 1.8	31.5 ± 1.2	7.28 ± 0.39	5.09 ± 0.23	31.7 ± 4.0
	MGN	0.82 ± 0.04	0.12 ± 0.01	16.9 ± 0.1	7.43 ± 0.10	4.43± 0.08	2.41 ± 0.16	4.72 ± 0.27
	BSMS-GNN	0.99 ± 0.03	0.21 ± 0.04	32.5 ± 0.5	16.1 ± 0.3	6.86 ± 0.09	1.95 ± 0.22	4.98 ± 0.71
	EAGLE	0.64 ± 0.04	0.17 ± 0.01	5.98 ± 0.43	5.17 ± 0.37	1.51 ± 0.04	0.67 ± 0.12	4.22 ± 0.30
	M4GN (Ours)	**0.31 ± 0.01**	**0.05 ± 0.00**	**5.26 ± 0.04**	**3.08 ± 0.06**	**1.17 ± 0.01**	**0.34 ± 0.02**	**1.87 ± 0.12**

4 Experiment

4.1 Experiment Setup

Datasets – We use two public datasets from (Pfaff et al., 2020): *CylinderFlow* (fluid flows around a cylinder) and *DeformingPlate* (elastic plate deformed by an actuator). We also create a new dataset, *DeformingBeam*, featuring a hyperelastic beam deformed by an actuator in a 3D mesh. Details of the datasets can be found in the appendix A. We create *DeformingBeam* dataset for three major reasons: (1) This dataset exhibits long-range interactions, with the largest graph diameter compared to the other two datasets (Table 2). (2) The inclusion of diverse mesh structures significantly increases the complexity of the underlying physics, making the task more challenging. (3) The dataset allows for the generation of directly scaled-up versions, enabling comprehensive generalization tests.

M4GN and Baselines – As a default configuration for our M4GN model, we use 7 message passing steps in the mesh graph network. The mesh segment transformer adopts 4 self-attention layers with 8 heads. We compare our method to five baseline models: 1) *GCN* (Kipf & Welling, 2016; Belbute-Peres et al., 2020), a basic GNN structure widely used for simulating fluid dynamics; 2) *g-U-Nets* (Gao & Ji, 2019; Alsentzer et al., 2020), a representative method that incorporates graph pooling modules to enhance long-range interactions; 3) *MeshGraphNets* (MGNs) (Pfaff et al., 2020), a single-level GNN architecture that achieves exceptional performance and generalizability across various dynamic systems; 4) *BSMS-GNN* (Cao et al., 2023), a recent work featuring a multi-level hierarchical GNN architecture that aims to enhance computational efficiency in simulating physical systems; and 5) *EAGLE*(Janny et al., 2023), a recent work presenting a clustering-based pooling method along with transformer to enhance performance on large-scale turbulent fluid dynamics. Detailed descriptions of the these models and training procedures can be found in Appendix B.

Metrics – In addition to traditional accuracy metrics, we introduce mesh quality metrics to assess the integrity of the predicted mesh in Lagrangian systems, where the mesh moves with the material. Maintaining mesh quality is crucial in these systems because changes in mesh elements over time can lead to numerical errors and misrepresentation of dynamic behaviors. Conversely, in Eulerian systems with a fixed mesh, mesh quality is less critical since the mesh remains static. To achieve holistic assessment of the predicted meshes, four mesh quality metrics are used: Hausdorff distance and Chamfer distance based Geometric Fidelity (GF), Mesh Continuity (MC), and Aspect Ratio (AR). Details description of these metrics can be found in Appendix C.2.

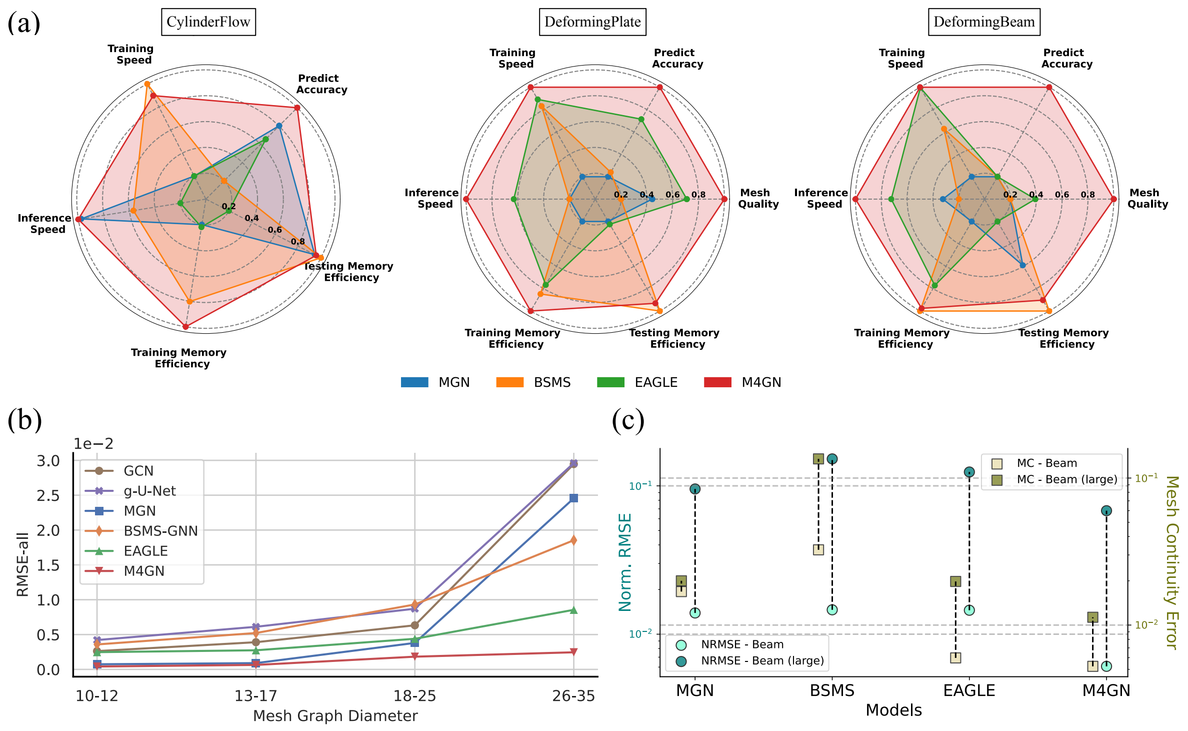


Figure 4: (a) Radar charts comparing the performance of difference models across different metrics under three datasets. For each metric, values are normalized to a 0.2–1.0 scale, where 1.0 represents the best performance. The concentric circles show normalized values from 0.2 (innermost) to 1.0 (outermost). Larger filled areas indicate better overall performance; (b) Illustration of how varying graph diameter impacts prediction accuracy across different models on the DeformingPlate dataset. RMSE-all is averaged over selected cases within a given graph diameter range and across all time steps; (c) Illustration of how each model's prediction accuracy (Normalized RMSE) and mesh quality (MC) change when generalizing from DeformingBeam to its scaled-up version, DeformingBeam (large).

4.2 Results and Discussion

4.2.1 Outstanding Performance of M4GN Across Multiple Datasets

The results in Table 1 and Figure 3 demonstrate the superior performance of our M4GN model compared to other baselines across various evaluation metrics. Specifically, for the CylinderFlow dataset, M4GN achieves a remarkable 36% reduction in test RMSE-all compared to the second-best performing model, EAGLE. This improvement is even more pronounced for the DeformingPlate dataset, where M4GN reduces the test RMSE-all by 42%. Similarly, for DeformingBeam dataset, M4GN demonstrates a 51% reduction in test RMSE-all. Such exceptional performance in 50-step and longer-step predictions underscores its enhanced capability for long-term predictions. In addition to achieving superior prediction accuracy, M4GN demonstrates excellent mesh quality, with up to a 48% reduction in GF and a 14% reduction in MC compared to the second-best model across both Lagrangian system datasets.

4.2.2 Achieving a Balance Between Accuracy and Efficiency

According to Figure 4(a), the MGN model performs well on small-diameter datasets like CylinderFlow, effectively capturing short-range effects. However, its performance drops on larger datasets like DeformingPlate and DeformingBeam due to oversmoothing and slow inference caused by excessive message passing and world-edges, reducing overall efficiency. The BSMS model excels in memory efficiency due to its bi-stride pooling, but this comes at the expense of mesh quality and accuracy, as the pooling introduces spatially insignificant edges. It also has slower inference times due to the complexity of reconstructing fine-grained details and managing long-range dynamics. The EAGLE model performs adequately but struggles with long-range effects due to its graph clustering and pooling methods, which lack physics-informed guidance. While it shows reasonable efficiency in DeformingPlate and DeformingBeam, its computational performance declines dramatically on CylinderFlow due to the increased amount of clusters needed under dense meshes. M4GN achieves the largest filled areas across all three datasets, demonstrating high prediction accuracy, superior mesh quality, and strong computational efficiency. More comprehensive evaluation results can be found in Table 7.

4.2.3 Effective Long-Range Dynamics and Scalability to Large Datasets

M4GN's remarkable performance stems from its ability to effectively handle long-range dynamic effects. In Figure 4(b), the relationship between graph diameter and RMSE-all for the DeformingPlate dataset is shown. While other models experience a significant rise in prediction error as the problem diameter increases, M4GN only shows a slight increase, demonstrating its superior performance on larger graphs and long-range interactions. Figure 4(c) further illustrates how each model's prediction accuracy (Normalized RMSE) and mesh quality change when generalizing from DeformingBeam to its scaled-up version, DeformingBeam (large). M4GN consistently achieves the lowest prediction error and mesh continuity error when tested on the larger-scale dataset with a model trained on the smaller scale. This highlights M4GN's robust generalization capabilities, making it well-suited for complex, large-scale dynamic systems. Detailed generalization results are presented in Table 6.

4.3 Additional Studies

We conducted additional studies to comprehensively evaluate model performance, hyperparameter selection, and the impact of key architectural designs, with detailed results and discussions provided in the appendices. Metrics for mesh quality and evaluations are presented in Appendix C.2, while segmentation quality metrics and related evaluations are detailed in Appendix C.3. Visualization and discussion of segmentation alignment with system dynamics are included in Appendix C.4. Ablation studies of our approach are discussed in Appendix D, covering the effect of different message-passing steps at the micro-level stage (Appendix D.1); the influence of segment extraction methods and segment count at the meso-level stage (Appendix D.2); and the impact of positional encoding and segment overlap at the macro-level stage (Appendix D.3). Additionally, a comprehensive analysis of generalization performance is provided in Appendix E, and further insights into computational efficiency are included in Appendix F.

5 Conclusion

In this paper, we introduced the Micro-Meso-Macro Mesh-based Graph Network (M4GN), a novel approach that enhances dynamic system simulations through a hierarchical pipeline. Our extensive evaluations demonstrate that M4GN outperforms traditional models, offering significant improvements in accuracy and computational efficiency, particularly in scenarios involving long-range dynamics and larger physical domains. The adaptability of M4GN to large-scale graphs underscores its potential for real-world applications in complex physical systems. However, the method has limitations, including the absence of hard constraints on contact meshes, which can result in overlapping meshes, and it has no guarantees on physical consistency at segmentation interfaces. These are important areas for future work to improve the robustness and applicability of the model.

Broader Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning for Physics, Surrogate Modeling, and Dynamic System Simulation. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

Charles J Alpert and So-Zen Yao. Spectral partitioning: The more eigenvectors, the better. In *Proceedings of the 32nd annual ACM/IEEE design automation conference*, pp. 195–200, 1995.

Emily Alsentzer, Samuel Finlayson, Michelle Li, and Marinka Zitnik. Subgraph neural networks. *Advances in Neural Information Processing Systems*, 33:8017–8029, 2020.

Lars Andersen. Linear elastodynamic analysis. 2006.

Klaus-Jürgen Bathe. *Computational fluid and solid mechanics*. Elsevier, 2001.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. arxiv 2018. *arXiv preprint arXiv:1806.01261*, 2018.

Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *international conference on machine learning*, pp. 2402–2411. PMLR, 2020.

Franco Bontempi and Lucia Faravelli. Lagrangian/eulerian description of dynamic system. *Journal of Engineering Mechanics*, 124(8):901–911, 1998.

Nico Booij and Leo H Holthuijsen. Propagation of ocean waves in discrete spectral wave models. *Journal of Computational Physics*, 68(2):307–326, 1987.

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.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3438–3445, 2020.

Ekin Dogus Cubuk, RJS Ivancic, Samuel S Schoenholz, DJ Strickland, Anindita Basu, ZS Davidson, Julien Fontaine, Jyo Lyn Hor, Y-R Huang, Y Jiang, et al. Structure-property relationships from universal signatures of plasticity in disordered solids. *Science*, 358(6366):1033–1037, 2017.

Tyler De Witt, Christian Lessig, and Eugene Fiume. Fluid simulation using laplacian eigenfunctions. *ACM Transactions on Graphics (TOG)*, 31(1):1–11, 2012.

Hervé Delingette. General object reconstruction based on simplex meshes. *International journal of computer vision*, 32:111–146, 1999.

Yu Diao, Jianchuan Yang, Ying Zhang, Dawei Zhang, and Yiming Du. Solving multi-material problems in solid mechanics using physics-informed neural networks based on domain decomposition technology. *Computer Methods in Applied Mechanics and Engineering*, 413:116120, 2023.

Victorita Dolean, Pierre Jolivet, and Frédéric Nataf. *An introduction to domain decomposition methods: algorithms, theory, and parallel implementation*. SIAM, 2015.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*, 2021.

Kerry A Emanuel. *Atmospheric convection*. Oxford University Press, USA, 1994.

Frank J Fahy. *Sound and structural vibration: radiation, transmission and response*. Elsevier, 2007.

Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. Multiscale meshgraphnets. In *ICML 2022 2nd AI for Science Workshop*, 2022.

Zhi-Fang Fu and Jimin He. *Modal analysis*. Elsevier, 2001.

Han Gao, Matthew J Zahr, and Jian-Xun Wang. Physics-informed graph neural galerkin networks: A unified framework for solving pde-governed forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 390:114502, 2022.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pp. 2083–2092. PMLR, 2019.

Jonathan Godwin, Michael Schaarschmidt, Alexander Gaunt, Alvaro Sanchez-Gonzalez, Yulia Rubanova, Petar Veličković, James Kirkpatrick, and Peter Battaglia. Simple gnn regularisation for 3d molecular property prediction & beyond. *arXiv preprint arXiv:2106.07971*, 2021.

Denis S Grebenkov and B-T Nguyen. Geometrical structure of laplacian eigenfunctions. *siam REVIEW*, 55 (4):601–667, 2013.

Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021.

Xu Han, Han Gao, Tobias Pfaff, Jian-Xun Wang, and Li-Ping Liu. Predicting physics in mesh-reduced space with temporal attention. *arXiv preprint arXiv:2201.09113*, 2022.

Lukas Harsch and Stefan Riedelbauch. Direct prediction of steady-state flow fields in meshed domain with graph networks. *arXiv preprint arXiv:2105.02575*, 2021.

Qi-Xing Huang, Martin Wicke, Bart Adams, and Leonidas Guibas. Shape decomposition using modal analysis. In *Computer Graphics Forum*, volume 28, pp. 407–416. Wiley Online Library, 2009.

Daniel P Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.

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.

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

Brian Kennett. *Seismic wave propagation in stratified media.* ANU Press, 2009.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

W Michael Lai, David Rubin, and Erhard Krempl. *Introduction to continuum mechanics.* Butterworth-Heinemann, 2009.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

X Li, HS Yu, and XS Li. Macro–micro relations in granular mechanics. *International Journal of Solids and Structures*, 46(25-26):4331–4341, 2009.

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, 2020.

Wenzhuo Liu, Mouadh Yagoubi, and Marc Schoenauer. Multi-resolution graph neural networks for pde approximation. In *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part III 30*, pp. 151–163. Springer, 2021.

Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li F Fei-Fei, Josh Tenenbaum, and Daniel L Yamins. Flexible neural representation for physics prediction. *Advances in neural information processing systems*, 31, 2018.

Timothy O'Connor. Emergent properties. 2020.

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

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Yulia Rubanova, Alvaro Sanchez-Gonzalez, Tobias Pfaff, and Peter Battaglia. Constraint-based graph network simulator. *arXiv preprint arXiv:2112.09161*, 2021.

Peter J Schmid, Larry Li, Matthew P Juniper, and Oliver Pust. Applications of the dynamic mode decomposition. *Theoretical and computational fluid dynamics*, 25:249–259, 2011.

Jean Michel Sellier, Gaétan Marceau Caron, and Jacob Leygonie. Signed particles and neural networks, towards efficient simulations of quantum systems. *Journal of Computational Physics*, 387:154–162, 2019.

Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

Uno Svedin et al. *Micro, meso, macro: Addressing complex systems couplings.* World Scientific, 2005.

Kunihiko Taira, Steven L Brunton, Scott TM Dawson, Clarence W Rowley, Tim Colonius, Beverley J McKeon, Oliver T Schmidt, Stanislav Gordeyev, Vassilios Theofilis, and Lawrence S Ukeiley. Modal analysis of fluid flows: An overview. *Aiaa Journal*, 55(12):4013–4041, 2017.

Kiwon Um, Xiangyu Hu, and Nils Thuerey. Liquid splash modeling with neural networks. In *Computer Graphics Forum*, volume 37, pp. 171–182. Wiley Online Library, 2018.

Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2019.

Olga Veksler, Yuri Boykov, and Paria Mehrani. Superpixels and supervoxels in an energy optimization framework. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part V 11*, pp. 211–224. Springer, 2010.

Haixin Wang, Yadi Cao, Zijie Huang, Yuxuan Liu, Peiyan Hu, Xiao Luo, Zezheng Song, Wanjia Zhao, Jilin Liu, Jinan Sun, et al. Recent advances on machine learning for computational fluid dynamics: A survey. *arXiv preprint arXiv:2408.12171*, 2024.

Steffen Wiewel, Moritz Becher, and Nils Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum*, volume 38, pp. 71–82. Wiley Online Library, 2019.

Edward L Wilson. Three-dimensional static and dynamic analysis of structures. *Computers and structures, Inc*, 1, 2002.

Tong Wu, Liang Pan, Junzhe Zhang, Tai Wang, Ziwei Liu, and Dahua Lin. Balanced chamfer distance as a comprehensive metric for point cloud completion. *Advances in Neural Information Processing Systems*, 34: 29088–29100, 2021.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek Abdelzaher. Revisiting over-smoothing in deep gcns. *arXiv preprint arXiv:2003.13663*, 2020.

Chen Yang, Shuai Li, Yu Lan, Lili Wang, Aimin Hao, and Hong Qin. Coupling time-varying modal analysis and fem for real-time cutting simulation of objects with multi-material sub-domains. *Computer Aided Geometric Design*, 43:53–67, 2016.

Youn-Yeol Yu, Jeongwhan Choi, Woojin Cho, Kookjin Lee, Nayong Kim, Kiseok Chang, ChangSeung Woo, Ilho Kim, SeokWoo Lee, Joon Young Yang, et al. Learning flexible body collision dynamics with hierarchical contact mesh transformer. *arXiv preprint arXiv:2312.12467*, 2023.

Olek C Zienkiewicz and Robert L Taylor. *The finite element method set.* Elsevier, 2005.

Appendix: Table of Contents

A Datasets

A.1 Datasets for Unstructured Mesh-based Simulations

Mesh-based dynamics simulation datasets have been developed as benchmarks to evaluate the performance of proposed models. As a cornerstone in the field, MeshGraphNets Pfaff et al. (2020) introduced a collection of datasets, encompassing cloth simulation, materials deformation and fluid flow, showcasing the versatility of GNNs in various problems involving unstructured mesh simulations. These datasets have been extensively adopted as benchmarks for developing new models. As the field shifts towards tackling more complex and large-scale systems, EAGLE Janny et al. (2023) presented a large-scale fluid dynamics dataset capturing unsteady and turbulent airflows. Similarly, BSMS-GNN Cao et al. (2023) provides the InflatingFont dataset, which focuses on the quasi-static inflation of enclosed elastic surfaces.

To demonstrate our model's generality across diverse dynamics and mesh configurations, we employed the *CylinderFlow* and *DeformingPlate* datasets in this study. These widely-used datasets from MeshGraphNets encompass both Eulerian and Lagrangian systems, providing a comprehensive evaluation of our model's performance across different simulation paradigms. Additionally, we developed the *DeformingBeam* dataset, which features meshes with a large graph diameter and complex long-range interactions spanning distant regions of the mesh. Existing datasets often lack this level of complexity, limiting their effectiveness in testing advanced models. We also generated a scaled-up version of the DeformingBeam dataset, enabling the evaluation of generalization performance from small-scale to large-scale scenarios, an important consideration for industrial-level simulations. The details of the investigated datasets are desbribed below and in Table 2.

A.2 Datasets Details

CylinderFlow – This public dataset includes simulations of transient incompressible flow around a cylinder, with varying diameters and locations, on a fixed 2D Eulerian mesh. In all fluid domains, the node type distinguishes fluid nodes, wall nodes and inflow/outflow boundary nodes. The inlet boundary conditions are given by a prescribed parabolic profile, $u_{in} = u_0[1 - 4(y/H)]$ where u_0 and H are the centerline velocity and the distance between the sidewalls, respectively. The dataset contains 1000 training simulations, 100 validation simulations and 100 test simulations.

DeformingPlate – This public dataset includes simulations of hyperelastic plates deformed by a moving obstacle, with variations in plate design and obstacle design. The node types are plate nodes, handle nodes that are fixed and obstacle nodes. This dataset contains 1200 training simulations, 100 validation simulations and 100 test simulations.

DeformingBeam – This dataset is generated using *solids4foam* which a toolbox for performing solid mechanics and fluid-solid interaction simulations in OpenFOAM and foam-extend. A nearly incompressible neo-Hookean model is used where the material properties are density ρ_0 = 1000 kg/m^3, Youngs's modulus E = 1 MPa and Poisson's ratio ν = 0.4. The beam comes in different geometries with various initial conditions and boundary conditions. The node types are plate nodes, handle nodes that are fixed and obstacle nodes. This dataset contains 355 training simulations, 40 validation simulations and 60 test simulations.

DeformingBeam (large) – A large domain DeformingBeam dataset is created for generalization studies. The physical domain size is doubled. The size of the mesh cell is kept consistent with the regular DeformingBeam dataset. This generalization dataset has 112 simulations.

Table 2: Detailed information for each dataset.

Dataset	Avg. # Nodes	# Steps	Mesh Type	Graph Diameter	Node Feature	Edge Feature	Output
CylinderFlow	1885	600	Triangle, Eulerian, 2D	11	$\mathbf{v}_i, \mathbf{n}_i$	$\mathbf{m}_{ij}, \lvert\mathbf{m}_{ij}\rvert$	$\dot{\mathbf{v}}_i$
DeformingPlate	1271	400	Tetrahedron, Lagrangian, 3D	16.9 ± 5.8	$\mathbf{x}_i, \dot{\mathbf{x}}_{\text{OBS}}, \mathbf{n}_i$	$\mathbf{x}_{ij}, \lvert\mathbf{x}_{ij}\rvert, \mathbf{m}_{ij}, \lvert\mathbf{m}_{ij}\rvert$	$\dot{\mathbf{x}}_i$
DeformingBeam	1542	400	Prism, Lagrangian, 3D	41.3 ± 11.8	$\mathbf{x}_i, \dot{\mathbf{x}}_{\text{OBS}}, \mathbf{n}_i$	$\mathbf{x}_{ij}, \lvert\mathbf{x}_{ij}\rvert, \mathbf{m}_{ij}, \lvert\mathbf{m}_{ij}\rvert$	$\dot{\mathbf{x}}_i$
DeformingBeam (large)	4540	400	Prism, Lagrangian, 3D	82.1 ± 23.0	$\mathbf{x}_i, \dot{\mathbf{x}}_{\text{OBS}}, \mathbf{n}_i$	$\mathbf{x}_{ij}, \lvert\mathbf{x}_{ij}\rvert, \mathbf{m}_{ij}, \lvert\mathbf{m}_{ij}\rvert$	$\dot{\mathbf{x}}_i$

B Model Details

B.1 Overall Information

The GNN part of M4GN adopts the encoder and graph processor in the MGN model Pfaff et al. (2020). The basic building block is Multi-Layer Perceptron (MLP). The MLP has 3 layers, a hidden dimension of 128, ReLU activation and single layer of Layer Normalization at the end. The node encoder and edge encoder(s) are 3-layer MLPs. By default, the M4GN has 7 message passing steps in the GNN. The mesh segment transformer consists of 4 self-attention layers, each with 8 heads. The output decoder is a 3-layer MLP without Layer Normalization. For DeformingPlate and DeformingBeam, M4GN only considers world edges between contacting mesh objects. The world edge radius is set to 0.01 for DeformingPlate and 0.002 for DeformingBeam. As both the DeformingBeam and DeformingPlate datasets feature a rigid object with quasi-static motion, we use only the first mode from our modal decomposition, which sufficiently captures the largest-scale deformation pattern. For the CylinderFlow dataset, we employ 6 modes, determined by an energy threshold criterion, ensuring a more comprehensive representation of the flow's multi-scale dynamics.

B.2 Baselines

GCN – The GCN model consists of 15 GCN layers with a hidden dimension of 128. The GCN model does not have edge input. Node input includes mesh position $\mathbf{x}_i$ for CylinderFlow. The implementation is from PyTorch Geometric.

g-U-Net – The g-U-Net model is a modified version from PyTorch Geometric. Instead of GCN layers, it is built using the GNN layers similar to MGN. The level of scale is 7 for CylinderFlow, 6 for DeformingPlate and 4 for DeformingBeam.

MGN – Our implementation of MGN follows the one described in Pfaff et al. (2020). The processor of MGN contains 15 MP steps. World edges are constructed as specified in the paper, with a world edge radius of 0.03 for DeformingPlate and 0.003 for DeformingBeam.

BSMS-GNN – We followed the BSMS-GNN implementation Cao et al. (2023) from `https://github.com/Eydcao/BSMS-GNN`. We introduced a modification to the original code by incorporating output normalization, which we observed to enhance the model's performance. For CylinderFlow and DeformingPlate, we used the same number of multi-scale levels as specified in the BSMS-GNN paper, at 7 and 6 levels, respectively. The number of multi-scale levels for DeformingBeam is set at 4 as an optimal configuration.

EAGLE – The implementation of EAGLE follows the paper Janny et al. (2023) and the code repository `https://github.com/eagle-dataset/EagleMeshTransformer`. We set the number of nodes per cluster at 20, which offers a balanced performance and efficiency according to the paper. This results in 94 clusters for CylinderFlow, 64 for DeformingPlate, and 38 for DeformingBeam. In addition, we add contacting world edges in EAGLE implementation for DeformingPlate and DeformingBeam to improve the performance. The world edges are added the same as in M4GN.

B.3 Training Details

During training, random Gaussian noise is added to the spatial node inputs, as described in Pfaff et al. (2020). For CylinderFlow, all models use a noise scale of 0.02. For DeformingPlate, all models use a noise scale of 0.003. For DeformingBeam, EAGLE and M4GN use a noise scale of 1e-4 and other models use a noise scale of 1e-3.

For GCN, g-U-Net, MGN, EAGLE and M4GN, we adopt the same training scheme: For CylinderFlow and DeformingPlate, we trained the model for 2M steps. The learning rate starts at 1e-4 and exponentially decays to 1e-6 from 1M to 2M steps. For DeformingBeam, we trained the model for 1M steps. The learning rate starts at 1e-4 and exponentially decays to 1e-6 from 500K to 1M steps.

For BSMS-GNN, we adopt the training scheme from the original implementation. Models for CylinderFlow and DeformingPlate were trained for 50 epochs, corresponding to 3.75M and 3M training steps, respectively. DeformingBeam model was trained for 100 epochs, corresponding to 1.775M training steps.

Across all models and datasets, we use a batch size of 8. Experiments were conducted using PyTorch distributed training over two Nvidia Tesla P100 GPUs.

B.4 Hybrid Mesh Segmentation Details

In Figure 5, several cases are selected from each dataset to illustrate the difference of each mesh graph segmentation methods. It's worth to note that the graph will be partitioned only once during the training and testing phase for each simulation, and this partitioning will remain consistent across all time steps. This is because the segmentation is based solely on the system's properties and initial conditions prior to the start of the simulation.

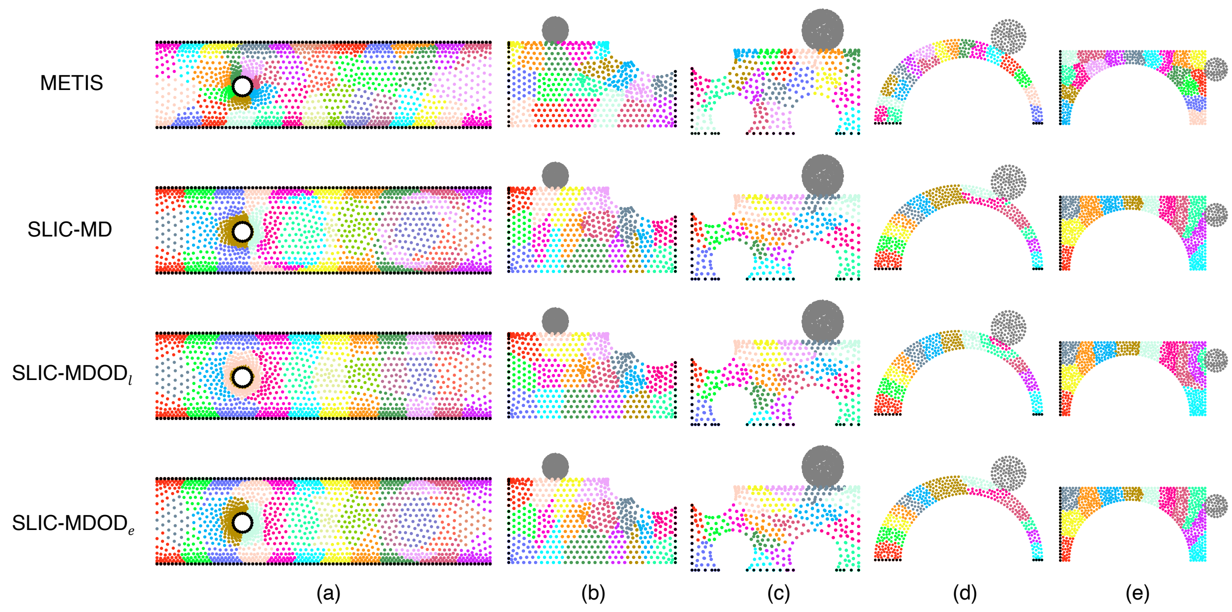


Figure 5: Illustration of different segmentation methods under various cases: (a):CylinderFlow; (b)(c): DeformingPlate; (d)(e): DeformingBeam. Mesh nodes are colored based on segment id and all boundary nodes are colored in black.

The pseudo code of the hybrid segmentation module proposed in this work can be found in Algorithm 2. Here, METIS (Karypis & Kumar, 1998) is a graph partitioning technique that efficiently divides meshes into approximately equal-sized partitions. It leverages multilevel partitioning algorithms to minimize the edge-cut or communication costs between the resulting partitions. We employ METIS due to its versatility in creating a user-specified number of equal-sized mesh segments. SLIC (Achanta et al., 2012) is a clustering algorithm employed for partitioning data. In our approach, we adapt SLIC to segment the mesh based on physics-informed features. These features could guide SLIC to create a segmentation that captures the underlying physics of the system. The consequent mesh segments can potentially enable efficient macro-level information exchange tailored to the system's dynamics. Concretely, for each node i, we incorporate physics-informed feature f_i^{md} derived from modal decomposition. Additionally, we augment these features by concatenating a measure of the shortest distance to obstacle nodes d_i^{obs}. To ensure that this measure dominates when d_i^{obs} is small, we apply either an exponential or logarithmic transformation, defined as:

$$f_{\exp}(d) = \exp(-d), \quad f_{\log}(d) = \log(d). \tag{14}$$

Depending on the selection of features and the transformation function, we design 3 variants of SLIC:

Algorithm 1: Modal Decomposition

```
1: Case Type: solid or fluid
2: Input: Finite element mesh, boundary conditions, material properties for solid (e.g. $E, \nu, \rho$), number of
   modes $m$
3: Build Finite Element Basis:
4:    Define shape functions on each element using the node connectivity
5:    Enumerate degrees of freedom (DOFs) for each node/component
6: if Case Type = solid then
7:    Structural Modal Analysis
8:    Assemble stiffness matrix $\mathbf{K}$ (using elasticity)
9:    Assemble mass matrix $\mathbf{M}$ (using density)
10:   Apply boundary conditions to eliminate fixed DOFs
11:   Solve $\mathbf{K}\boldsymbol{\phi} = \lambda\,\mathbf{M}\boldsymbol{\phi}$ for the first $m$ modes
12:   Output: Eigenpairs $\{(\lambda_i,\, \boldsymbol{\phi}_i)\}_{i=1}^{m}$ (structural modes)
13: else if Case Type = fluid then
14:   Laplacian Eigenfunctions
15:   Assemble Laplacian matrix
16:   Assemble $L^2$-type matrix
17:   Apply Dirichlet constraints on boundary nodes
18:   Solve $-\nabla^2\phi = \lambda\,\phi$ for the first $m$ modes
19:   Output: Eigenpairs $\{(\lambda_i,\, \phi_i)\}_{i=1}^{m}$ (harmonic modes)
20: end if
21: Return: $m$-dimensional feature vector $f_i^{md} = (\phi_1(i), \phi_1(i), \dots \phi_m(i))$ at each mesh node $i$
```

- SLIC-MD: $f_i = f_i^{md}$
- SLIC-MDOD$_l$: $f_i = \left[f_{\log}(d_i^{obs}), f_i^{md}\right]^T$
- SLIC-MDOD$_e$: $f_i = \left[f_{\exp}(d_i^{obs}), f_i^{md}\right]^T$

After we have the physics-informed feature, we can apply the SLIC algorithm to get the mesh node segments.

B.4.1 Mesh Segment Hyperparameter Selection

The compactness parameter τ in the SLIC algorithm controls the trade-off between physics-guided feature similarity and spatial proximity. Our goal is to choose τ such that the resulting segmentation captures both underlying physical patterns and spatial coherence (i.e., grouping nodes that are close to each other). For CylinderFlow and DeformingPlate, we set $\tau = 1.0$, which provides a balanced segmentation. For DeformingBeam, we set a lower τ at 0.5 to promote a better alignment with physical features. The cluster size S is determined such that the domain area satisfy: Domain Area $= KS^2$, where K is the number of segments and the domain area is given by $(x_{\max} - x_{\min})(y_{\max} - y_{\min})$. The average cluster size S for CylinderFlow, DeformingPlate and DeformingBeam is set to be $\sqrt{0.656/K}$, $\sqrt{0.125/K}$ and $\sqrt{0.005/K}$, respectively.

Systematically choosing the optimal number of segments K requires both domain insight and practical experimentation. In our experience, two main factors drive the choice of K: (1) the total mesh size (N) and (2) local variations in mesh density. For instance, CylinderFlow is particularly dense near boundaries, which benefits from a larger K, whereas DeformingBeam/DeformingPlate have more uniformly distributed nodes, so a smaller K can suffice.

To make this selection concrete, we typically perform a short hyperparameter sweep over a small set of candidate values for K. A simple heuristic is to pick K values on a roughly geometric or linear scale, for instance: $K \in \{\sqrt{N}/2, \sqrt{N}, 2\sqrt{N}, ...\}$ up to a point where adding more segments no longer improves validation metrics (e.g., prediction accuracy, mesh quality). In practice, testing each candidate K on a subset (e.g., 10%) of the training data is typically enough to identify a near-optimal configuration, and then we

Algorithm 2: Hybrid Mesh Segmentation

```
1:  **Input:**
2:      Initial mesh graph $G = (\mathcal{V}, \mathcal{E})$
3:      Perform modal decomposition and computed mesh node feature $f_i$
4:      Number of segments $K$, compactness parameter $\tau$, average cluster size $S$
5:  **Output:** Mesh node segmentation $\{\mathcal{V}_{S_k}\}_{k=1}^{K}$
    ⇒ ***Graph-based Mesh Segment Initialization***
6:  **Coarsening Phase:**
7:  $G_{\text{coarse}} \leftarrow G$
8:  **while** size of $G_{\text{coarse}}$ is larger than threshold **do**
9:      Combine pairs of connected nodes in $G_{\text{coarse}}$ to form a coarser graph
10:     $G_{\text{coarse}} \leftarrow$ coarsened graph
11: **end while**
12: **Initial Partitioning:**
13: Partition $G_{\text{coarse}}$ into $K$ segments using a standard partitioning method (e.g., spectral partitioning)
14: **Uncoarsening and Refinement Phase:**
15: **while** $G_{\text{coarse}} \neq G$ **do**
16:     Expand $G_{\text{coarse}}$ to the next finer graph $G_{\text{fine}}$
17:     Project partitions onto $G_{\text{fine}}$
18:     Refine the partitioning on $G_{\text{fine}}$ to improve quality
19:     $G_{\text{coarse}} \leftarrow G_{\text{fine}}$
20: **end while**
21: Obtain initial clusters $\{\mathcal{V}_{S_k}\}_{k=1}^{K}$ from the final partitioning, which will be updated next
    ⇒ ***Superpixel-based Mesh Segment Refinement***
22: **repeat**
23:     **for** each mesh segment centroid $C_k$ **do**
24:         Update $C_k$ by averaging over all mesh nodes assigned to it:
                $$C_k = [x_{C_k}, f_{C_k}]^T = \frac{1}{|\mathcal{V}_{S_k}|} \sum_{i \in \mathcal{V}_{S_k}} [x_i, f_i]^T$$
            where $\mathcal{V}_{S_k}$ is the set of mesh nodes assigned to segment $S_k$
25:     **end for**
26:     **for** each mesh node $i \in V$ **do**
27:         Compute the distance measure $d(i, C_k)$ to each cluster center $C_k$ using:
                $$d(i, C_k) = \|f_i - f_{C_k}\| + \tau \|x_i - x_{C_k}\|$$
            where $x_i$ and $x_{C_k}$ are the spatial coordinates, $f_i$ and $f_{C_k}$ are the physics-guided features.
28:         Assign mesh node $i$ to the nearest segment centroid $C_k$ if $d(i, C_k) \leq S$
29:     **end for**
30: **until** convergence or a maximum number of iterations is reached
```

finalize training with that K on the full dataset. This strategy is computationally manageable and provides a principled way to tailor K to new domains.

C Additional Evaluation Metrics and Results

C.1 Overall Results

The results in Table 1 demonstrate the superior performance of our M4GN model compared to other baselines across various evaluation metrics. Specifically, for the CylinderFlow dataset, M4GN achieves a remarkable 36% reduction in test RMSE-all compared to the second-best performing model, EAGLE. This improvement is even more pronounced for the DeformingPlate dataset, where M4GN reduces the test RMSE-all by 42%. Similarly, for DeformingBeam dataset, M4GN demonstrates a 51% reduction in test RMSE-all. Such exceptional performance in 50-step and longer-step predictions underscores its enhanced capability for long-term predictions. In addition to achieving superior prediction accuracy, M4GN demonstrates excellent mesh quality, with up to a 48% reduction in GF and a 14% reduction in MC compared to the second-best model across both Lagrangian system datasets.

C.2 Metrics for Mesh Quality Measure

Hausdorff Distance – The Hausdorff Distance measures how well the mesh with the predicted node positions conforms to the system's true geometry. It is defined as:

$$\mathrm{GF}_h(\mathcal{V}, \hat{\mathcal{V}}) = \max\Big\{h(\mathcal{V}, \hat{\mathcal{V}}), h(\hat{\mathcal{V}}, \mathcal{V})\Big\}, \tag{15}$$

where $h(\mathcal{V}, \hat{\mathcal{V}}) = \sup_{\mathbf{x}\in\mathcal{V}} \inf_{\hat{\mathbf{x}}\in\hat{\mathcal{V}}} \|\mathbf{x} - \hat{\mathbf{x}}\|$ is the directed Hausdorff distance (Huttenlocher et al., 1993) from the ground-truth node set $\mathcal{V}$ to the predicted node set $\hat{\mathcal{V}}$.

Chamfer Distance – The Chamfer Distance (Wu et al., 2021) measures the average distance between points on the predicted mesh and the true mesh, providing a balanced assessment of GF_h. Unlike the Hausdorff Distance, which focuses on the maximum deviation, the Chamfer Distance is sensitive to the overall distribution of errors across the mesh surfaces. As both Chamfer and Hausdorff distance are measures for GF, we name them as GF_c and GF_h for simplicity, respectively. The Chamfer distance is mathematically defined as:

$$\mathrm{GF}_c(\mathcal{V}, \hat{\mathcal{V}}) = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{x}\in\mathcal{V}} \min_{\hat{\mathbf{x}}\in\hat{\mathcal{V}}} \|\mathbf{x} - \hat{\mathbf{x}}\|^2 + \frac{1}{|\hat{\mathcal{V}}|}\sum_{\hat{\mathbf{x}}\in\hat{\mathcal{V}}} \min_{\mathbf{x}\in\mathcal{V}} \|\hat{\mathbf{x}} - \mathbf{x}\|^2, \tag{16}$$

where $\mathcal{V}$ and $\hat{\mathcal{V}}$ are the set of vertices in the ground-truth and predict mesh, respectively. $|\mathcal{V}|$ and $|\hat{\mathcal{V}}|$ denote the number of vertices in each mesh.

Mesh Continuity – Mech Continuity evaluates the uniformity of predicted mesh cell sizes to ensure stability and is defined as

$$\mathrm{MC} = \frac{1}{C}\sum_{i=1}^{C} \frac{\max_{c_j\in \mathrm{Adj}(c_i)} V(c_j)}{\min_{c_j\in \mathrm{Adj}(c_i)} V(c_j)}, \tag{17}$$

where $\mathrm{Adj}(c_i)$ is the neighboring cells of cell c_i, and $V(c_i)$ calculates the volumetric area for c_i.

Aspect Ratio (error) – The Aspect Ratio (Zienkiewicz & Taylor, 2005) metric assesses the shape quality of individual 2D or 3D mesh elements and is widely used in finite element method (FEM) literature to evaluate how closely each element approaches the ideal shape, such as an equilateral triangle or a regular tetrahedron. For example, for triangular meshes, the aspect ratio is defined as $\frac{L_{\max}}{2\sqrt{\sqrt{3}A}}$, where $L_{\max}$ is the longest edge length, A is the area of the triangle. For tetrahedra mesh, it is defined as $\frac{\sqrt{6}L_{\max}}{V^{1/3}}$, where V the volume of the tetrahedron. High aspect ratios indicate elongated or distorted elements, which can cause numerical instability and reduce simulation accuracy. By analyzing the aspect ratios across all elements, we can assess the overall uniformity and regularity of the mesh. To evaluate the accuracy of the predicted mesh compared to the ground truth, we calculate the aspect ratio for both the predicted and actual meshes. The Aspect

Ratio Error is then determined as the L_1 distance between these two values. This error metric quantifies the deviation in shape quality between the predicted and true meshes, providing a direct measure of how well the prediction preserves the ideal element shapes. Incorporating the Aspect Ratio Error allows for a more precise evaluation of mesh quality and prediction accuracy, ensuring that the segmented meshes maintain the necessary geometric properties for reliable simulations.

C.3 Segmentation Quality Metrics

In order to rigorously evaluate the quality of our physics-informed mesh segmentation and its impact on the prediction of system dynamics, it is essential to consider metrics that assess both inter-segment and intra-segment characteristics. We introduce three such metrics —*Conductance*, *Edge Cut Ratio*, and *Silhouette Score* — which provide a comprehensive assessment of segmentation quality by quantifying the cohesion within segments and the separation between segments. The necessity of these metrics arises from the need to ensure that segments are well-separated, minimizing unnecessary interactions between dissimilar regions (inter-segment quality), and that nodes within the same segment share similar properties or behaviors (intra-segment quality).

Moreover, in our hierarchical model architecture, the intra-segment quality pertains to the micro-level information exchange stage. High intra-segment quality facilitates accurate modeling of local dynamics within each segment by ensuring that nodes are cohesive and share similar dynamic behaviors. Conversely, the inter-segment quality directly relates to the macro-level information exchange stage. High inter-segment quality ensures efficient communication between segments by reducing redundant or irrelevant interactions, which is crucial for capturing global dynamics across the entire mesh. Below are the details of three metrics to measure segmentation quality.

Conductance – Conductance measures the fraction of total edge connections that cross between different segments relative to the total connections of the segments. It assesses how well the segmentation minimizes inter-segment connections while maintaining intra-segment cohesion. Let $G = (\mathcal{V}, \mathcal{E})$ as an undirected graph representing the mesh, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Let S be a segment and $\bar{S} = G \setminus S$ be its complement. The conductance of segment S is defined as:

$$\text{Conductance} = \frac{\left|\{(u,v) \in \mathcal{E} \mid u \in S,\ v \in \bar{S}\}\right|}{\min\left(\text{vol}(S),\ \text{vol}(\bar{S})\right)}, \tag{18}$$

where the numerator is the number of edges crossing between S and $\bar{S}$. The volumn of segment S is given by $\text{vol}(S) = \sum_{u \in S} \deg(u)$, where $\deg u$ is the degree of node u (the number of edges connected to u).

Edge Cut Ratio – The Edge Cut Ratio quantifies the proportion of edges that are cut by the segmentation relative to the total number of edges in the mesh. It is defined as:

$$\text{Edge Cut Ratio} = \frac{|\{(u,v) \in \mathcal{E} \mid \text{Seg}(u) \neq \text{Seg}(v)\}|}{E}, \tag{19}$$

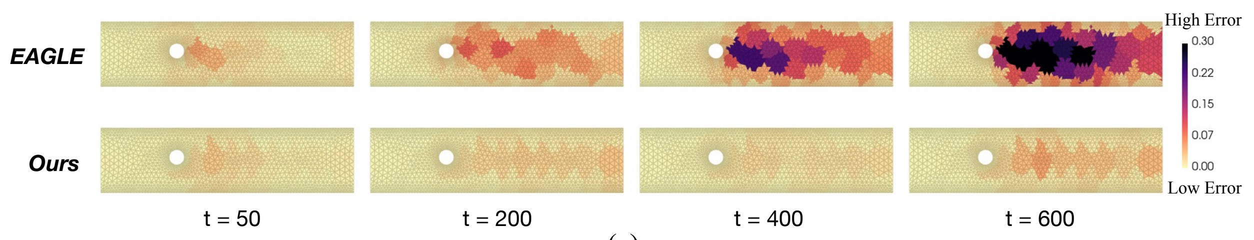
where the denominator is the number of edges that connect nodes in different segment. $\text{Seg}(u)$ denotes the segment to which node u belongs and $E = |\mathcal{E}|$ is the total number of edges.

Silhouette Score – For each node i, the Silhouette Score evaluates how similar i is to nodes in its own segment compared to nodes in other segments. It is defined as:

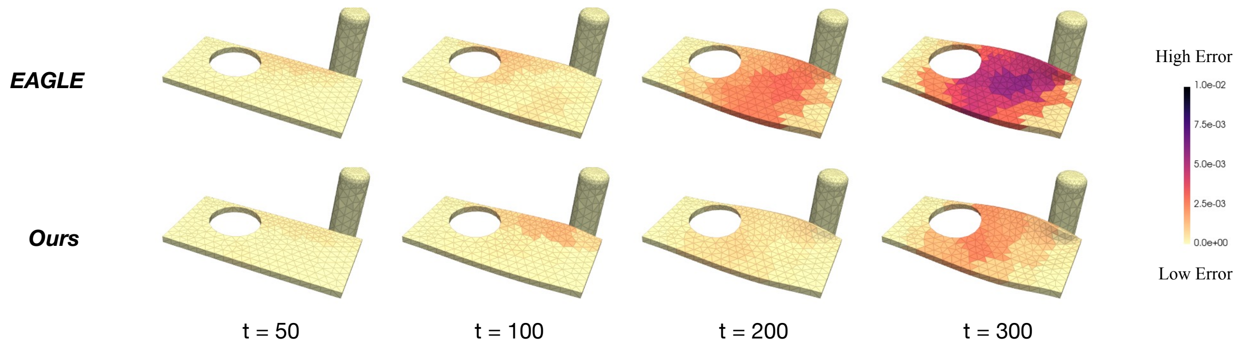
$$\text{Silhouette Score} = \frac{1}{N} \sum_{u=1}^{N} \frac{b(i) - a(i)}{\max\{a(i),\ b(i)\}}, \tag{20}$$

where N is the total number of nodes, $a(i)$ is the average dissimilarity of node i with all other nodes in the same segment and $b(i)$ is the lowest average dissimilarity of node i to any other segment to which i does not belong. To be more specific $a(i) = \frac{1}{|S_i|-1} \sum_{\substack{j \in S_i \\ j \neq i}} d(i,j)$, $b(i) = \min_{S' \neq S_i} \left(\frac{1}{|S'|} \sum_{j \in S'} d(i,j) \right)$, where S_i is the segment containing node i and $d(i,j)$ can be any appropriate distance metric, such as Euclidean distance based on node features or positions.

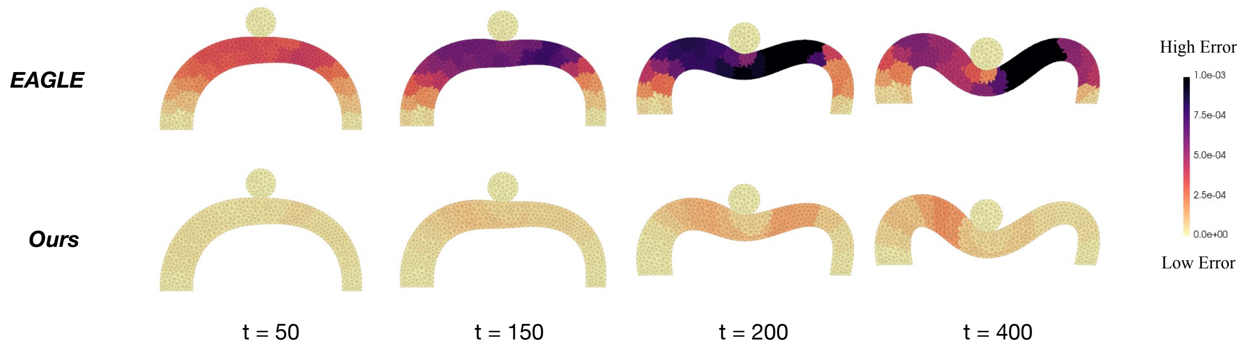
By combining these metrics, we achieve a comprehensive evaluation of segmentation quality that covers both the internal cohesion of segments and their external separation. Having these metrics, along with prediction result metrics, can better help us understand the effect of segmentation on the predicted system dynamics. These metrics can be used to help finding better physics-informed segment features and determining the optimal segmentation number (results and discussion in Appendix D.3).



(a)



(b)



(c)

Figure 6: Visualization of simulation rollouts over time for three datasets, comparing our segmentation method with EAGLE. Nodes are colored based on the average prediction error within their segments. Our method consistently produces uniform segment colors across time steps, indicating that nodes within each segment share similar dynamic behaviors and that segments maintain high continuity. For example, in (a), segmentation follows periodic wave patterns in fluid dynamics, while in (c), it reflects symmetrical system dynamics with symmetric segment coloring. These visualizations demonstrate that our segmentation effectively captures the temporal and spatial dynamics of the system, outperforming EAGLE.

C.4 Visualization of Segmentation Alignment with System Dynamics

To visualize the how predict mesh properties in each segment various through time for different method, we include additional visualization results in Figure 6, where mesh nodes at each predicted time step are colored based on the average prediction error within their segments. In all three datasets, our method produces segments with uniform colors across time steps, indicating that nodes within the same segment share similar dynamic behaviors and different segments have little discrepancies or maintain high continuity. This means that our segmentation effectively groups regions with coherent dynamic interactions, ensuring consistent modeling and accurate prediction of the system's evolution over time. Such consistent segmentation enhances the model's ability to capture and represent the underlying physical properties, leading to more reliable and stable simulation outcomes. Specifically, our methods is able to accurately capture dynamic patterns of periodic wake formations shown in Figure 6(a). Also, in case with inherent symmetry, such as the Figure 6(c), our method successfully generates symmetric segments that share similar dynamic properties.

Our segmentation method effectively aligns with system dynamics but could be sensitive to initial setup parameters, potentially missing important physical dependencies in complex systems. To address this, future work could incorporate adaptive clustering techniques based on system dynamics. This would improve the physical relevance and robustness of our segmentation, making it more versatile for a wider range of dynamic environments.

D Ablation Studies

D.1 Micro-level Information Exchange

According to Figure 7, with fewer message passing steps, each node updates only based on immediate neighbors, resulting in higher prediction errors and mesh discontinuities. As more steps are introduced, nodes gather information from a broader neighborhood, leading to more accurate predictions and smoother mesh transitions. The early iterations of message passing yield the most noticeable improvements, as nodes rapidly gather useful information from their surrounding environment. Later iterations primarily serve to fine-tune the mesh continuity and reduce local errors, but the impact on overall accuracy diminishes. Interestingly, increasing the number of message-passing steps beyond a certain point continues to improve mesh quality, but prediction accuracy may degrade. This suggests the occurrence of oversmoothing, where the model excessively homogenizes node features, or overfitting, where the model starts to memorize local information rather than generalize. This phenomenon highlights the importance of carefully selecting the number of message-passing steps during micro-level information exchange step to strike the right balance between improving prediction accuracy and maintaining mesh quality.

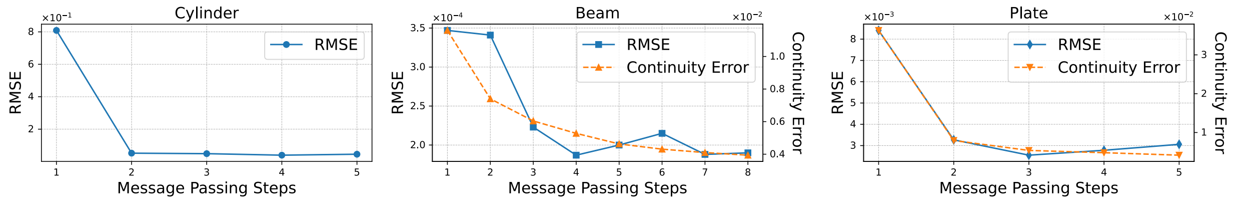


Figure 7: Ablation study on the impact of varying message-passing steps in the micro-level information exchange on prediction performance across three datasets.

D.2 Meso-level Information Alignment

Comparison of Different Segment Extraction Methods – As shown in Table 3, mesh segment method can have a large impact on the result. By comparing SLIC and METIS results, we find a 16%, 27% and 7% improvement in RMSE-all for CylinderFLow, DeformingPlate and DeformingBeam. The best segment method for all three datasets is SLIC-MDOD$_e$.

To thoroughly evaluate the different segmentation methods, we utilize the three metrics -Conductance, Edge Cut Ratio, and Silhouette Score - introduced in Appendix C.3 to assess both inter-segment and intra-segment qualities of mesh partitions, providing a comprehensive understanding of each method's effectiveness. We then analyzed the correlation between these segmentation metrics and overall dynamic

Table 3: Ablation study on different segment extraction methods over different dataset.

Segmentation Method	Dataset	GF$_h$ ↓	GF$_c$ ↓	MC ↓	Aspect Ratio ↓	RMSE-1	RMSE-all
METIS	Cylinder	-	-	-	-	3.44e-03	4.59e-02
	Plate	5.32e-03	1.36e-05	5.33e-03	2.97e-03	2.67e-04	3.29e-03
	Beam	3.88e-04	5.61e-08	5.18e-03	3.09e-03	1.15e-05	2.16e-04
SLIC-MD	Cylinder	-	-	-	-	3.16e-03	5.62e-02
	Plate	5.10e-03	8.38e-6	4.67e-03	2.53e-03	2.74e-04	3.02e-03
	Beam	3.81e-04	5.45e-08	5.32e-03	3.20e-03	1.17e-05	2.32e-04
SLIC-MDOD$_l$	Cylinder	-	-	-	-	4.16e-03	5.29e-02
	Plate	4.84e-03	7.23e-06	4.56e-03	2.47e-03	2.68e-04	2.82e-03
	Beam	3.53e-03	5.10e-08	5.29e-03	3.40e-03	1.22e-05	2.25e-04
SLIC-MDOD$_e$	Cylinder	-	-	-	-	3.09e-03	3.86e-02
	Plate	4.26e-03	6.49e-06	4.73e-03	2.58e-03	2.71e-04	2.40e-03
	Beam	3.02e-03	4.47e-08	5.31e-03	3.08e-03	1.17e-05	2.01e-04

system performance, including mesh quality and prediction error, as illustrated in Figure 9 (a-c). Our findings indicate that segmentation methods incorporating physics-informed features, particularly those utilizing obstacle distances with exponential transformations, generally enhance model performance across various datasets. This improvement can be attributed to three key factors: (1) *Alignment with Dynamics*, where segmentation reflecting physical influences enables more effective learning of the system's dynamics; (2) *Enhanced Segment Quality*, achieved through improved intra-segment cohesion and minimized inter-segment interactions, facilitating better learning of localized patterns; and (3) *Benefit to Learning*, where emphasizing critical regions via exponential transformations allows the model to focus on areas with significant dynamic changes, thereby enhancing prediction accuracy. These results demonstrate that the choice of segmentation method impacts the model's ability to learn dynamic behaviors, and the introduction of additional metrics reveals that physics-informed segmentation effectively aligns mesh partitions with the system's inherent physical properties, thereby benefiting the learning process.

Influence of Segment Count on Performance – Table 4 and Table 5 present the RMSE-1, RMSE-all, and various mesh quality metrics as the total number of mesh segments is varied during training on three different datasets. In general, M4GN maintains stable performance with relatively low variance, indicating that results are not highly sensitive to segment count. This robustness ensures reliable accuracy across different mesh granularities. However, increasing the number of segments—thereby reducing finite elements per segment—can lead to slight decreases in accuracy and performance.

To comprehensively evaluate the effect of segment number and determine the optimal segmentation for a given dataset, we analyzed prediction accuracy across a wide range of segment counts (from 3 to 51) during training on the DeformingBeam dataset. The impact of varying the number of mesh segments on prediction accuracy is illustrated in Figure 8 and Figure 9(d). According to the plots, we identify 19 segments as the optimal number. At this segmentation level, the model achieves the lowest RMSE and Chamfer Distance, indicating high prediction accuracy and precise shape representation. The Hausdorff Distance is also minimized, reflecting excellent alignment between the predicted and true meshes. While the Silhouette score peaks at 9 segments—suggesting well-defined and compact clusters—the slight decrease at 19 segments is offset by significant gains in other performance metrics. Choosing a lower number of segments, such as 3 or 9, may result in higher Silhouette scores but can compromise mesh detail and prediction accuracy due to insufficient spatial granularity. Conversely, selecting a higher number of segments beyond 19 shows diminishing returns, with only marginal improvements or slight degradations in some metrics and a continued decline in Silhouette scores, potentially indicating over-segmentation and unnecessary computational complexity.

In conclusion, when presented with a new dataset, the optimal number of segments can be determined by first computing Silhouette scores for various segment counts to assess cluster cohesion and separation without requiring model training. This provides initial guidance on meaningful segmentation levels. Subsequently, training the model with different segment numbers and evaluating performance metrics like RMSE, Hausdorff Distance, and Chamfer Distance will help identify the point where performance improvements plateau or begin to reverse, indicating the optimal balance between segmentation detail and model efficacy.

D.3 Macro-level Information Exchange

Influence of Positional Encoding on Performance – Table 5 and Figure 10(a) shows the effect of adding positional encoding for small and large number of segments across three datasets. According to the results, we identified several key findings. Firstly, the effectiveness of PE depends on the number of segments: in the CylinderFlow and Deforming Plate datasets, incorporating PE with fewer segments improves performance across multiple metrics by reducing positional ambiguity. With low segment counts, each segment covers larger, more diverse areas, limiting the model's spatial detail and understanding of segment relationships. PE provides explicit positional information, allowing the model to distinguish distinct regions within the same segment and better comprehend their interactions. However, as the number of segments increases and spatial resolution improves, the benefits of PE diminish and may even introduce unnecessary complexity that hinders performance. Additionally, dataset-specific factors influence PE's effectiveness; for example, the DeformingBeam dataset, with its complex geometry and deformation, did not benefit from PE. This indicates that PE's success depends not only on segment count but also on how well the PE implementation aligns with the dataset's unique characteristics.

Consequently, tailored PE approaches that consider specific geometry and deformation patterns are necessary for complex systems to achieve significant performance gains. In summary, while PE enhances the performance of graph-based networks, further advancements are needed to develop optimal encoding strategies that consistently improve performance across diverse dynamic systems.

Influence of Segment Overlap on Performance – Table 4 and Figure 10(b) illustrate the effect of adding segment overlap for small and large number of segments across three datasets. According to the results, the effectiveness of adding overlap between segments ($\delta > 0$) depends on both the segment count and the characteristics of the dataset, such as dimensionality, mesh type, and system dynamics. Overlapping segments are more beneficial with higher segment counts where discontinuities are more prevalent. In Eulerian systems, overlaps enhance the capture of complex interactions and smooth transitions on fixed meshes, leading to improved representation of fluid dynamics. Conversely, in Lagrangian systems where meshes move with the material, overlaps can create redundancy and complicate connectivity, with their impact on model performance varying based on mesh structures and deformation behaviors. For example, in the Deforming Beam dataset, which uses a prism mesh suited for directional deformation, overlapping segments improve performance by facilitating smooth transitions along its mesh surface, especially with a higher number of segments. In contrast, the Deforming Plate dataset employs a tetrahedral mesh with complex, isotropic deformations, where overlaps introduce unnecessary complexity and redundancy, resulting in decreased performance. Therefore, despite both being 3D Lagrangian systems, the different mesh types and deformation patterns explain why overlapping segments benefit the Deforming Beam but not the Deforming Plate.

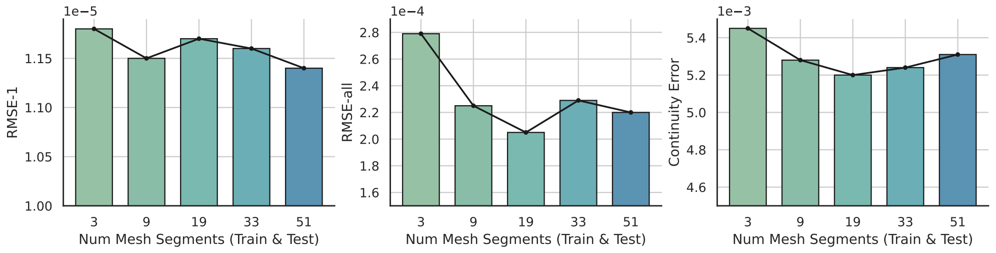


Figure 8: Impact of varying mesh segment numbers during training on prediction accuracy under the DeformingBeam dataset. The number of mesh segments remains consistent during both training and testing. In general, M4GN maintains stable performance with relatively low variance, indicating that results are not highly sensitive to segment count. This robustness ensures reliable accuracy across different mesh granularities. However, increasing the number of segments—thereby reducing finite elements per segment—can lead to slight decreases in accuracy and performance. More detailed analysis on the effect of segmentation numbers to various metrics can be found in Figure 9(d).

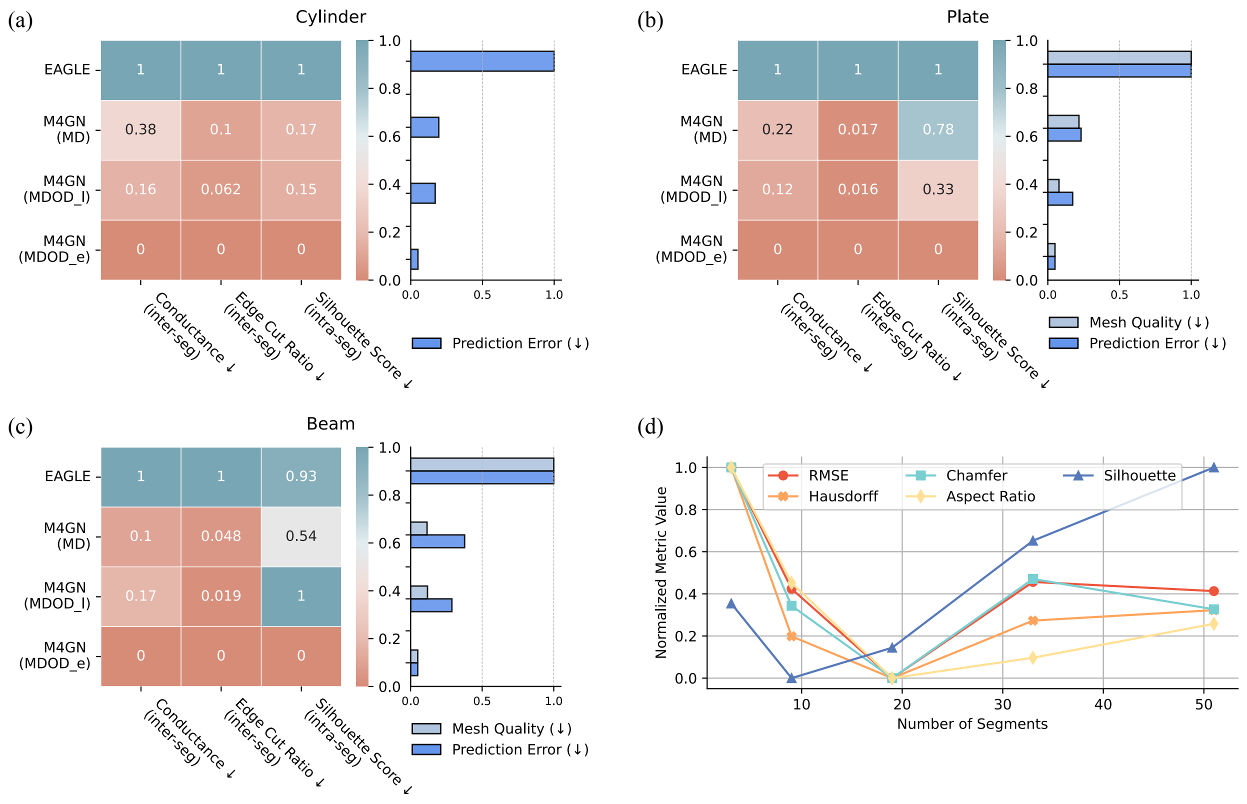


Figure 9: **(a-c)** Evaluation of different segmentation methods under three datasets. The heatmap (left) presents normalized Conductance, Edge Cut Ratio, and reversed Silhouette Score for EAGLE and four M4GN variants. Metrics are scaled between 0 and 1, with Silhouette Scores reversed to ensure consistent evaluation criteria, where lower values indicate better segmentation quality. The sidebar plot (right) depicts normalized Prediction Error and Mesh Quality, with a minimum value of 0.05 applied to avoid invisible bars. These figures evaluate segmentation quality across multiple metrics and demonstrate how different segmentation methods influence model accuracy and mesh quality, emphasizing the advantages of our physics-informed segmentation strategies; **(d)** Dependence of various performance metrics on the number of segments in M4GN under Deforming Beam dataset. The plot illustrates how the normalized values of several performance metrics vary with the number of segments. Each metric is represented by a distinct curve, demonstrating the relationship between segment number and overall performance. This figure evaluates the effect of segment number and guides the selection of the optimal number of segments for balanced performance across all metrics.

Table 4: Ablation study of number of segments, and effect of adding segment overlap.

Dataset	N_{seg}	$\delta > 0$	GF$_h$ ↓	GF$_c$ ↓	MC ↓	Aspect Ratio ↓	RMSE-1	RMSE-all
Cylinder	16		-	-	-	-	3.16e-03	5.03e-02
	16		-	-	-	-	3.19e-03	5.35e-02
	36		-	-	-	-	3.41e-03	4.42e-02
	36		-	-	-	-	3.09e-03	3.86e-02
Plate	9		4.98e-03	9.58e-06	5.01e-03	2.83e-03	2.77e-04	3.88e-03
	9		5.32e-03	9.87e-06	5.24e-03	2.95e-03	2.83e-04	2.98e-03
	19		4.51e-03	6.91e-06	4.73e-03	2.58e-03	2.71e-04	2.40e-03
	19		4.76e-03	7.01e-06	4.81e-03	2.85e-03	2.77e-04	3.59e-03
Beam	9		3.46e-04	5.23e-08	5.17e-03	3.31e-03	1.14e-05	2.39e-04
	9		3.38e-04	5.07e-08	5.31e-03	3.30e-03	1.15e-05	2.40e-04
	19		3.57e-04	4.92e-08	5.24e-03	3.29e-03	1.18e-05	2.28e-04
	19		3.19e-04	4.73e-08	5.31e-03	3.08e-03	1.17e-05	2.01e-04

Table 5: Ablation study of number of segments and whether to add PE or not.

Dataset	N_{seg}	PE	GF$_h$ ↓	GF$_c$ ↓	MC ↓	Aspect Ratio ↓	RMSE-1	RMSE-all
Cylinder	16		-	-	-	-	3.19e-03	5.35e-02
	16		-	-	-	-	3.34e-03	4.76e-02
	36		-	-	-	-	3.09e-03	3.86e-02
	36		-	-	-	-	3.00e-03	3.80e-02
Plate	9		4.81e-03	9.33e-06	5.01e-03	2.83e-03	2.77e-04	3.88e-03
	9		4.24e-03	6.72e-06	5.04e-03	2.81e-03	2.84e-04	2.72e-03
	19		5.10e-03	6.51e-06	4.73e-03	2.58e-03	2.71e-04	2.40e-03
	19		5.13e-03	6.78e-06	4.74e-03	2.64e-03	2.68e-04	2.91e-03
Beam	9		3.75e-04	5.89e-08	5.31e-03	3.30e-03	1.15e-05	2.40e-04
	9		3.51e-04	5.33e-08	5.18e-03	3.31e-03	1.17e-05	2.56e-04
	19		3.22e-04	4.86e-08	5.31e-03	3.08e-03	1.17e-05	2.01e-04
	19		3.19e-04	4.84e-08	5.27e-03	3.18e-03	1.15e-05	2.21e-04

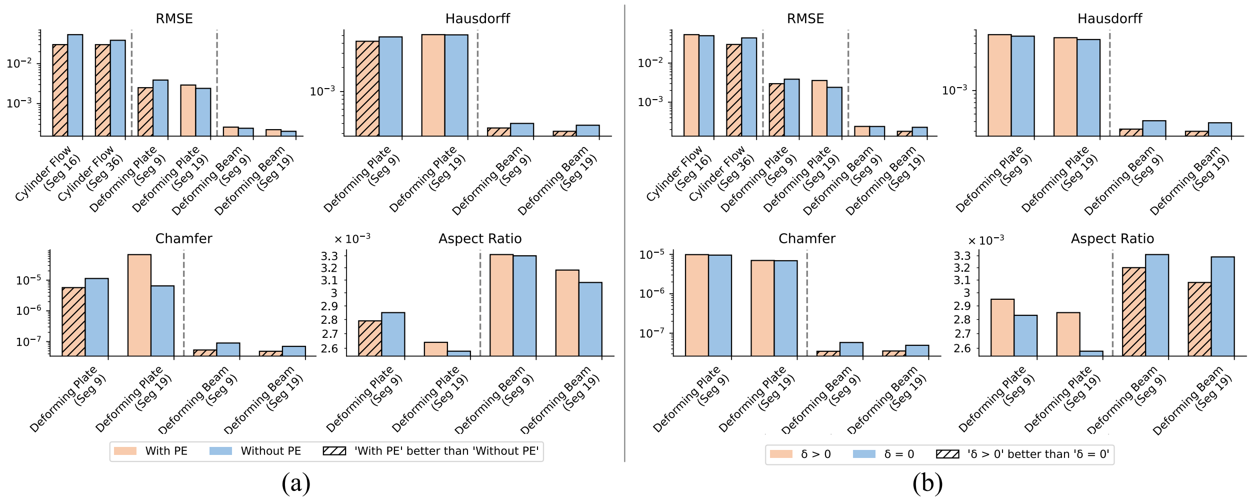


Figure 10: Ablation study on the effects of position encoding and segment overlap across datasets with varying segment numbers. The figure presents the performance metrics for models both with and without the position encoder (a), and with and without considering segment overlap (b) across three distinct datasets, each characterized by a different number of segments. By comparing these conditions, the study highlights how the inclusion of position encoding and the handling of segment overlap influence overall performance, thereby informing the selection of optimal model configurations.

E Generalization Studies

To evaluate the generalizability of our M4GN model, we created a larger-scale DeformingBeam dataset, detailed in Appendix A.

E.1 Performance on Larger-Scale Datasets

Table 6 summarizes the generalization performance of various models trained on the DeformingBeam dataset and directly applied to DeformingBeam(large), a scaled-up version. The results demonstrate that M4GN consistently outperforms all other models across all metrics. In terms of mesh quality, M4GN achieves a 53% improvement over both EAGLE and BSMS for Geometric Fidelity (GF). Similarly, for Mesh Continuity (MC), M4GN achieves the best performance with a value of 1.08e-02, representing a 45% improvement over EAGLE, the next-best model. For the RMSE metrics, M4GN delivers the lowest RMSE-1, RMSE-50, and RMSE-all. Notably, M4GN's RMSE-all is 46% lower than EAGLE. These findings suggest that M4GN not only preserves prediction accuracy but also enhances mesh quality when generalizing to larger-scale data, significantly surpassing state-of-the-art models in both accuracy and mesh quality. This demonstrates M4GN's robust generalization ability, making it highly suitable for complex, large-scale dynamic systems.

Table 6: Generalization performance of our method and five baseline models on the scaled-up DeformingBeam dataset. M4GN demonstrates superior accuracy and mesh quality when generalizing to an unseen dataset with a denser mesh and more extensive long-range dynamic effects.

Method	$GF_h \downarrow$	$GF_c \downarrow$	MC ↓	Aspect Ratio ↓	RMSE-1	RMSE-50	RMSE-all
GCN	2.18e-02	3.28e-05	1.21e-01	1.69e-01	2.57e-04	1.95e-03	1.11e-02
g-U-Net	1.94e-02	2.80e-05	4.56e-02	7.01e-02	1.60e-04	1.87e-03	1.01e-02
MGN	2.32e-02	1.43e-05	2.00e-02	2.57e-02	1.34e-04	1.43e-03	6.42e-03
BSMS	1.72e-02	3.34e-05	1.35e-01	1.17e-01	4.47e-04	3.19e-03	1.03e-02
EAGLE	1.69e-02	2.20e-05	1.98e-02	5.15e-02	8.42e-05	1.45e-03	8.37e-03
M4GN	**7.96e-03**	**5.35e-06**	**1.08e-02**	**2.24e-02**	**5.47e-05**	**9.20e-04**	**4.58e-03**

E.2 Effect of Mesh Segment Count on Generalization

Generalization with Varying Segment Counts During Testing – Across three datasets, we perform generalization studies where the model is tested using a varying number of segments. The results in Figure 11 illustrate the generalization performance. Pink columns are the references for regular testing and the others are generalization to different number of segments from training. Overall, the M4GN model can generalize very well to different number of segments during testing.

Impact of Segment Count During Training and Testing – Equipped with message passing and transformer mechanisms, M4GN can handle an arbitrary number of segments. Figure 12 shows the generalization performance of our M4GN model to larger domain as heatmaps, where models trained with a specific number of segment under deformingBeam dataset are tested with varying number of segments under deformingBeam (large). We observe that better results are seen when the number of nodes per segment during training is less than or equal to that in the generalizing domain, or when the number of segments is greater. Overall, we demonstrate M4GN's robustness and adaptability in generalizing to larger domains with varying mesh segments, making it highly suitable for real-world applications involving large and diverse mesh graphs.

F Computational Efficiency Analysis

F.1 Performance Comparision

Table 7 listed the training time, test time and number of parameters for four models MGN, BSMS-GNN, EAGLE and M4GN across three datasets. The RMSE-all is also listed as performance reference. Our M4GN model has comparable or better efficiency compared with other models. Notebly, the M4GN model has exceptional efficiency with RMSE-all better than other baselines.

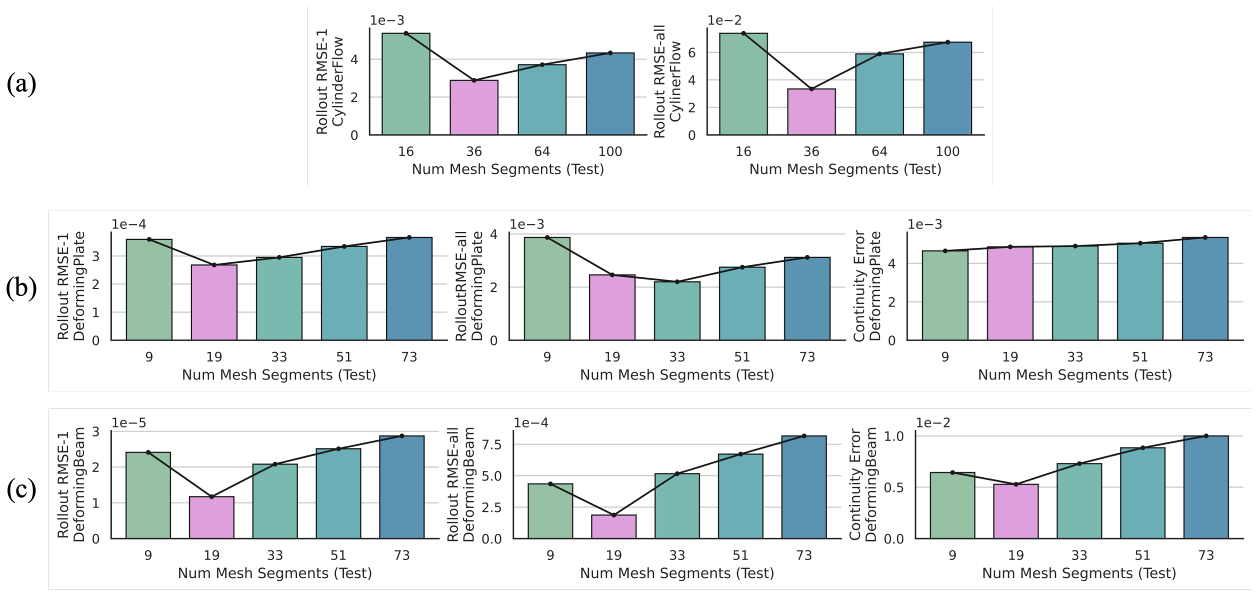


Figure 11: Generalization performance of our method under varying segment counts during testing over three datasets. (a) CylinderFlow: effect of number of segments for test set on different metrics, where model is trained under 36 segments (colored in pink); (b) DeformingPlate: effect of number of segments for test set on different metrics, where model is trained under 19 segments (colored in pink); (c) DeformingBeam: effect of number of segments for test set on different metrics, where model is trained under 19 segments (colored in pink). This figure illustrates that our M4GN model, despite being trained with a fixed number of mesh segments, maintains strong accuracy and mesh quality when tested with varying numbers of mesh segments.

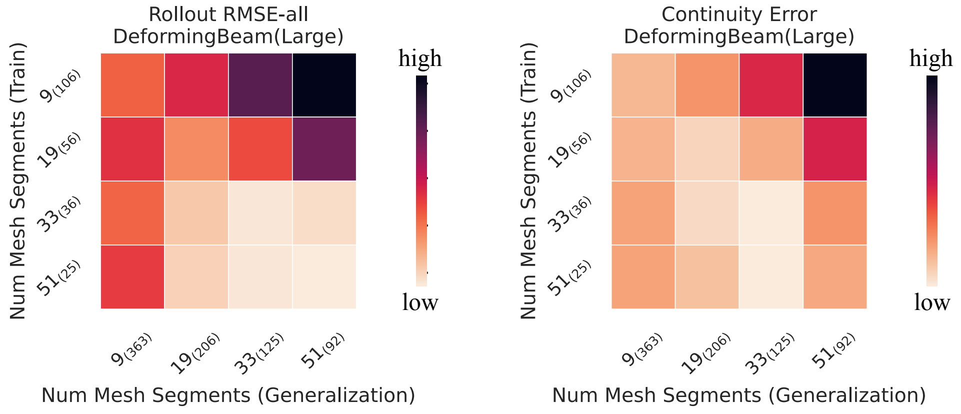


Figure 12: Generalization performance of our method on larger domains under different number of mesh segmentation during training and testing. The subscript of each mesh segment indicating the average number of nodes per segment. M4GN demonstrates robustness and adaptability in handling larger domains with varying mesh segments, making it well-suited for real-world applications involving large and complex mesh structures.

Table 7: Comprehensive evaluation of our method alongside MGN, BSMS, and EAGLE under three datasets. M4GN consistently delivers stable, competitive efficiency while maintaining high accuracy and superior mesh quality.

Dataset	Model	RMSE-all	MC ↓	Train Time per step [ms] ↓	Train Memory [MB] ↓	Test Time per step [ms] ↓	Test Memory [MB] ↓	Train Time total [h] ↓
Cylinder	MGN	4.81e-02	-	66.7	698.5	20.2	67.2	37.1
	BSMS	1.37e-01	-	54.7	430.3	23.8	57.9	30.4
	EAGLE	5.83e-02	-	69.5	618.7	28.8	230.8	38.6
	M4GN	3.80e-02	-	56.2	366.6	20.0	65.0	31.2
Plate	MGN	1.47e-02	9.25e-03	131.9	6021.5	36.2	445.5	73.3
	BSMS	1.18e-02	1.83e-02	83.9	910.1	37.7	77.9	46.6
	EAGLE	3.87e-03	5.56e-03	81.2	1090.8	32.4	362.7	45.1
	M4GN	2.65e-03	4.82e-03	76.5	648.1	29.3	103.3	42.5
Beam	MGN	4.72e-04	1.69e-02	79.1	1074.4	28.6	83.8	22.0
	BSMS	4.98e-04	3.25e-02	61.8	213.7	30.7	35.6	17.2
	EAGLE	4.22e-04	5.98e-03	53.5	410.3	26.0	153.5	14.9
	M4GN	1.87e-04	5.26e-03	53.4	234.5	24.2	47.1	14.8

F.2 Compelxity Analysis

M4GN is composed of four key components: an Encoder-Process-Decoder (EPD) network operating on mesh graphs, modal decomposition, hybrid mesh segmentation, and a mesh segment transformer. Since modal decomposition and mesh segmentation are performed only once at the initial time step, their computational cost is excluded from our analysis. We focus instead on the computational complexity of the EPD and mesh segment transformer components. The complexity of the EPD is: $O(L_1|\mathcal{V}|d^2 + L_1|\mathcal{E}|d^2)$, where L_1 is the number of message passing layer, d is the feature dimension, $|\mathcal{V}|$ is the number of mesh nodes and $|\mathcal{E}|$ is the number of mesh edges. The complexity of mesh segment transformer is $O(L_2K^2d + L_2Kd^2)$, where L_2 is the number of multi-head attention layers, K is the number of segments, and d is the feature dimension. The overall time complexity is $O(L_1|\mathcal{V}|d^2 + L_1|\mathcal{E}|d^2 + L_2K^2d + L_2Kd^2)$.

G Qualitative Results

Figure 13, 14, 15, and 16 illustrate selected rollout results for all three datasets under different models.

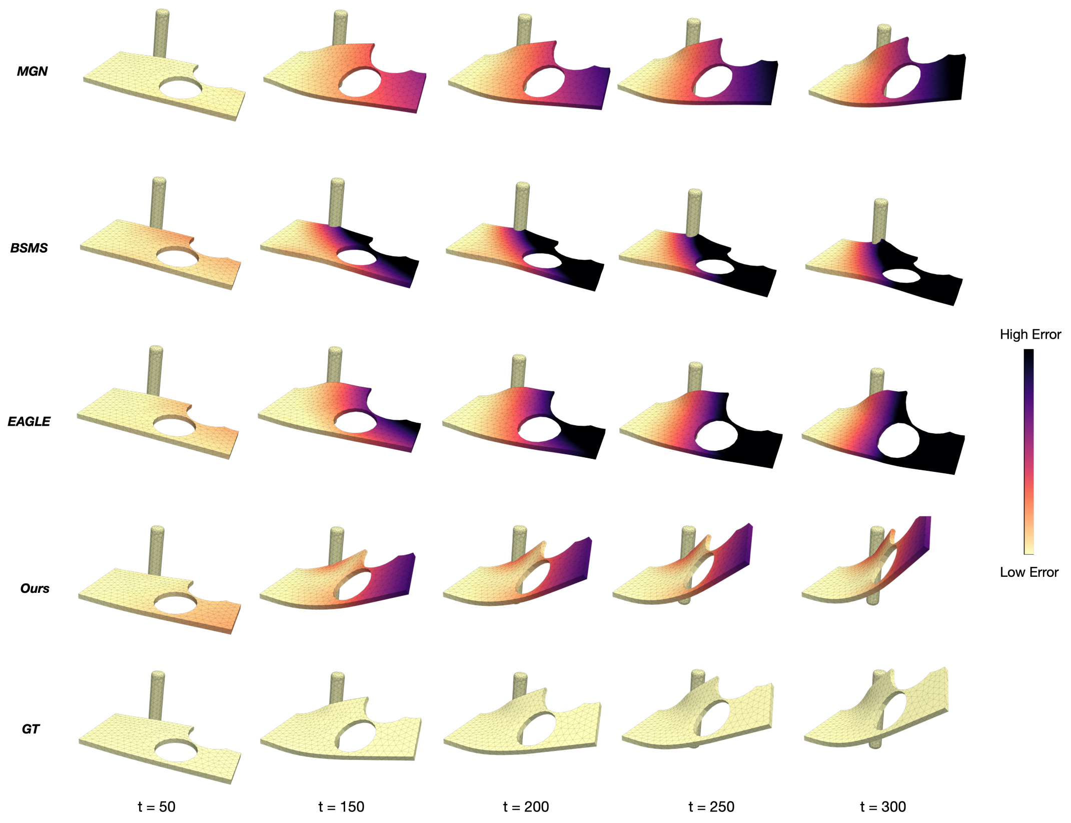


Figure 13: Additional simulation results for different models under DeformingPlate dataset.

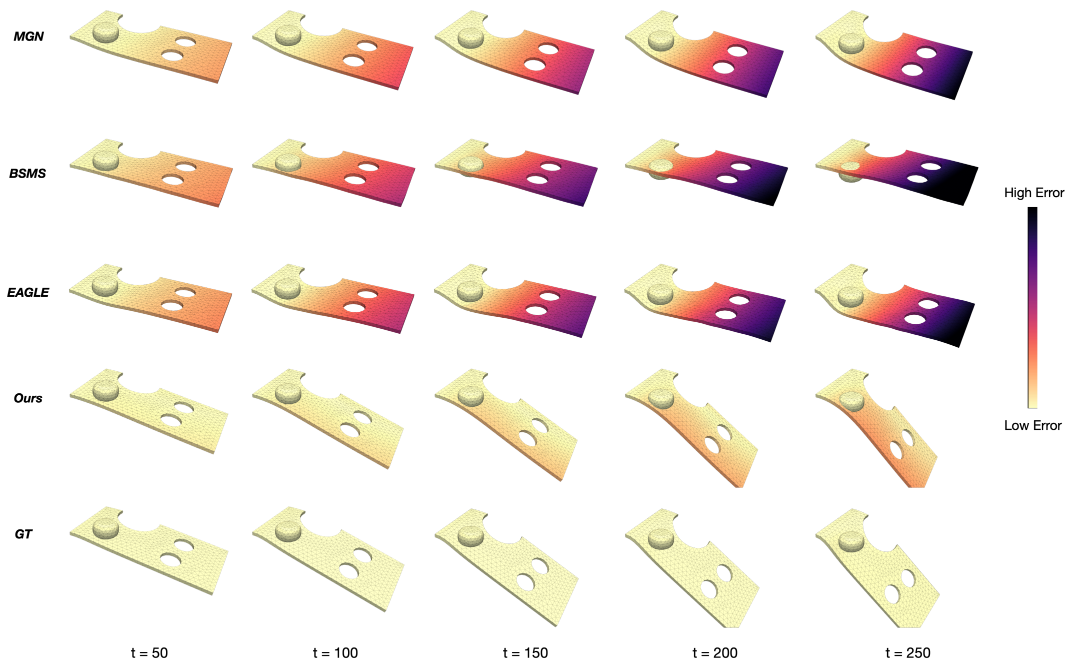


Figure 14: Additional simulation results for different models under DeformingPlate dataset.

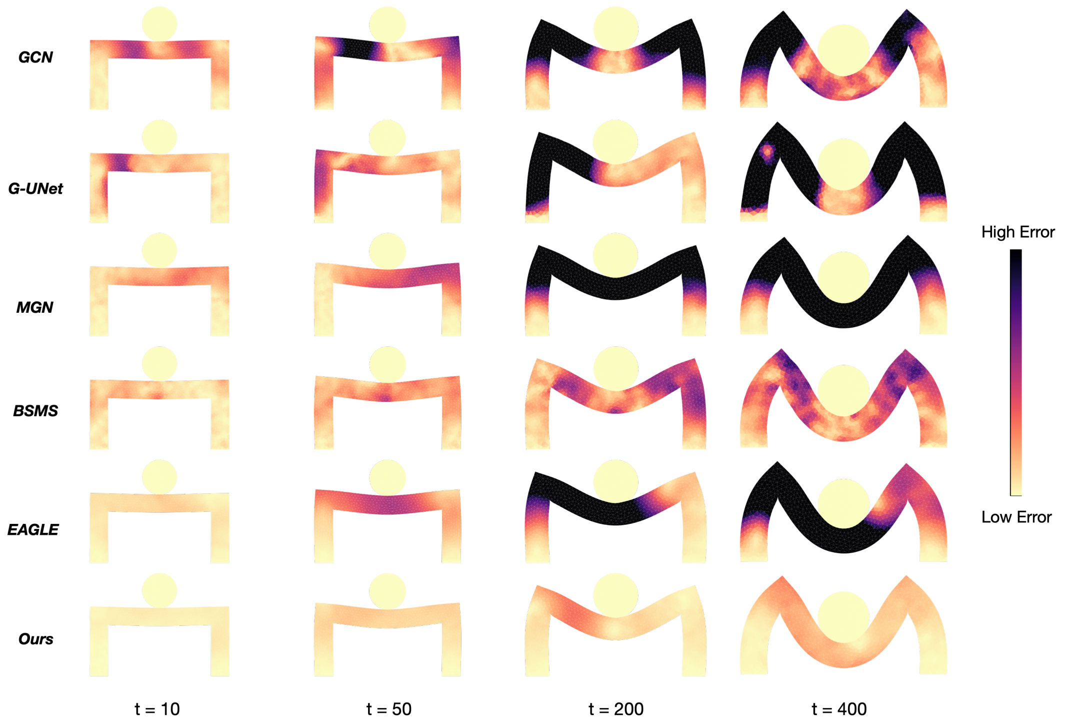


Figure 15: Additional simulation results for different models under DeformingBeam dataset

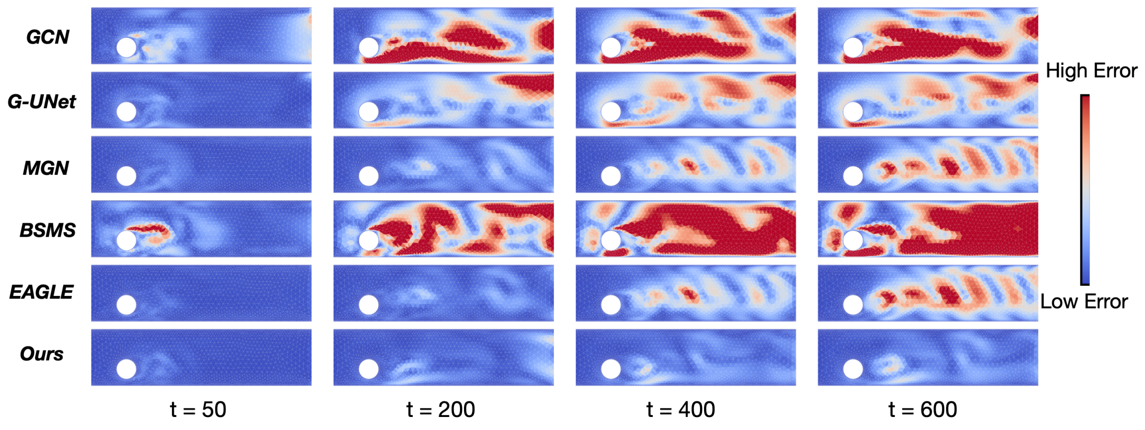


Figure 16: Additional simulation results for different models under CylinderFlow dataset