

ADAPTIVE GRAPH REWIRING TO MITIGATE OVER-SQUASHING IN MESH-BASED GNNs FOR FLUID DYNAMICS SIMULATIONS

006 **Anonymous authors**

007 Paper under double-blind review

ABSTRACT

012 Mesh-based simulation using Graph Neural Networks (GNNs) has been recognized as a promising approach for modeling fluid dynamics. However, the mesh refinement techniques which allocate finer resolution to regions with steep gradients can induce the over-squashing problem in mesh-based GNNs, which prevents the capture of long-range physical interactions. Conventional graph rewiring methods attempt to alleviate this issue by adding new edges, but they typically complete all rewiring operations before applying them to the GNN. These approaches are physically unrealistic, as they assume instantaneous interactions between distant nodes and disregard the distance information between particles. To address these limitations, we propose a novel framework, called Adaptive Graph Rewiring in Mesh-Based Graph Neural Networks (AdaMeshNet), that introduces an adaptive rewiring process into the message-passing procedure to model the gradual propagation of physical interactions. Our method computes a rewiring delay score for bottleneck nodes in the mesh graph, based on the shortest-path distance and the velocity difference. Using this score, it dynamically selects the message-passing layer at which new edges are rewired, which can lead to adaptive rewiring in a mesh graph. Extensive experiments on mesh-based fluid simulations demonstrate that AdaMeshNet outperforms conventional rewiring methods, effectively modeling the sequential nature of physical interactions and enabling more accurate predictions. Our source code is available at <https://anonymous.4open.science/r/AdaMeshNet-9321>.

1 INTRODUCTION

035 Fluid dynamics has seen various attempts to solve the Navier-Stokes equations (Temam, 1977).
 036 Since analytical solutions for complex physics are unobtainable, numerical methods such as the finite element method (FEMs) (Madenci & Guven, 2006; Stolarski et al., 2018; Dhatt et al., 2012)
 037 have been widely adopted to solve the differential equations by discretizing them in space and time.
 038 As a key strategy for enhancing the accuracy of these numerical methods, the mesh refinement technique (Löhner, 1995; Liu et al., 2022) generates adaptive meshes by increasing the resolution of
 039 specific regions that require detailed analysis, such as areas with sharp gradients involving unstructured
 040 surfaces in complex dynamics problems. The adaptive meshes are used to focus computational
 041 resources on the most critical areas, which enables high accuracy in mesh simulations without the
 042 need to compute the entire domain at high resolution, even with limited computational power.
 043

044 Recently, graph neural networks (GNNs) have been widely used for mesh simulations by leveraging
 045 these advantages of adaptive meshes. In particular, MeshGraphNets (MGN) (Pfaff et al., 2020) have
 046 proven effective at approximating simulation results on unstructured meshes by propagating local
 047 physical interactions between nodes via message passing (Sanchez-Gonzalez et al., 2020; Fortunato
 048 et al., 2022; Nabian et al., 2024).

049 A key challenge in applying GNNs to fluid dynamics simulations is balancing mesh refinement with
 050 the propagation of interactions. Specifically, regions with sharp gradients, such as boundary layers
 051 and turbulence, require higher-density mesh structures for accurate simulation (Katz & Sankaran,
 052 2011; Baker, 2005). However, these fine mesh structures cause the **over-squashing** problem when
 053 physical interactions are propagated through the graph (Topping et al., 2022; Di Giovanni et al.,
 2023; Black et al., 2023). Over-squashing refers to the progressive compression of information from

054 distant nodes as it passes through multiple layers in GNNs. This compression becomes more severe
 055 in the fine mesh areas, which makes it difficult to capture long-range interactions. In particular,
 056 fluid dynamics simulations require mesh refinement techniques to accurately capture complex flow
 057 phenomena, which makes this challenge especially pronounced compared to many other domains.

058 To solve the over-squashing problem, several graph rewiring approaches that account for graph
 059 topology have been developed (Gasteiger et al., 2019; Karhadkar et al., 2023; Nguyen et al., 2023).
 060 Recently, PIORF (Yu et al., 2025) introduced a graph rewiring approach specifically designed for
 061 fluid dynamics simulations, which considers not only graph topology but also physical quantities of
 062 the fluid. However, in existing approaches, all rewiring occurs before GNN training for fluid simu-
 063 lations and it forces nodes to interact as if they were immediate neighbors. This leads the model to
 064 lose information about their actual physical distance and gradual propagation, which is unrealistic
 065 for long-range interactions in fluids. In reality, phenomena such as boundary layers and turbulence
 066 affect distant particles only after a certain delay, since their influence propagates gradually through
 067 sequential collisions among neighboring particles. This highlights the need for a new rewiring ap-
 068 proach that explicitly accounts for the gradual propagation of physical interactions without loss of
 069 inter-node distance information during long-range interactions.

070 In this work, we theoretically demonstrate the over-squashing phenomenon inherent in MGN, which
 071 is widely used as a mesh-based GNN model. Additionally, to address this issue, we propose a novel
 072 framework, called Adaptive Graph Rewiring to Mitigate Over-Squashing in Mesh-Based GNNs
 073 for Fluid Dynamics Simulations (AdaMeshNet). The key idea is to dynamically rewire new edges
 074 during the message-passing process by considering the gradual propagation of physical interactions
 075 in fluid simulations. We first detect bottleneck nodes in the graph based on Ollivier–Ricci curvature
 076 (ORC) (Ollivier, 2009). We then compute the distances between these bottleneck nodes and nodes
 077 with large velocity differences, and subsequently calculate the rewiring delay score using both the
 078 distances and the velocity differences. The rewiring delay score quantifies the degree of rewiring
 079 delay and serves to determine the layer at which rewiring should be performed during the message-
 080 passing process. Based on the computed rewiring delay scores, we rewire bottleneck nodes with
 081 nodes of high velocity gradients at each layer of the message passing process. This approach applies
 082 rewiring delays based on curvature and physical quantity, which enables simulations to consider
 083 the gradual propagation of interactions. Therefore, our model provides an effective solution to the
 084 over-squashing problem in fluid simulations by performing adaptive graph rewiring in the message
 085 passing process.

086 To validate our approach, we conducted extensive experiments on two fluid dynamics datasets:
 087 CylinderFlow and Airfoil. Our AdaMeshNet framework was compared with leading static rewiring
 088 methods, all implemented on the MGN model. The results demonstrate that AdaMeshNet achieves
 089 more accurate predictions of key physical quantities, such as velocity and pressure. Furthermore, it
 090 produces velocity contours that more closely match the ground truth, particularly in capturing com-
 091 plex phenomena like wavelike propagation. These findings highlight our model’s ability to effec-
 092 tively solve the over-squashing problem by considering the gradual nature of physical interactions,
 093 a crucial aspect often overlooked by existing methods.

094 In summary, our main contributions are summarized as follows:

- 095 • We provide a theoretical demonstration of the over-squashing problem in MGN.
- 096 • We propose an adaptive rewiring method that considers the gradual propagation of physical
 097 interactions to address the over-squashing problem in fluid simulations.
- 098 • We demonstrate that our model outperforms existing rewiring methods in our experiments.

100 2 PRELIMINARIES

101 2.1 PROBLEM DEFINITION

102 The goal of our task is to train a model that predicts the dynamic quantity of the mesh at time
 103 $t + 1$, using the current mesh \mathcal{M}_t at time t and past meshes $\{\mathcal{M}_{t-1}, \mathcal{M}_{t-2}, \dots, \mathcal{M}_{t-h}\}$. Our fluid
 104 dynamics simulations are based on the Euler system, which models physical quantities that change
 105 over time on the fixed mesh coordinates and incorporates these changes into the simulation.

106 The mesh \mathcal{M}_t is transformed into a multi-graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$. The mesh nodes and edges are
 107 mapped to graph nodes \mathcal{V} and bidirectional edges \mathcal{E} , respectively. A denotes the adjacency matrix,

108 and we define $\tilde{A} = A + I$, which is the adjacency matrix augmented with self-loops. We then define
 109 \hat{A} as the normalized augmented adjacency matrix, i.e., $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, where $\tilde{D} = D + I$ and D
 110 is the diagonal degree matrix. Each node has node features consisting of the dynamic feature q_i and a
 111 one-hot vector that represents the node type n_i , which includes fluid, wall, inflow, and outflow nodes.
 112 Each edge has features m_{ij} , which include connection information such as the distance between two
 113 particles, as well as the relative displacement vector $\mathbf{d}_{ij} = \mathbf{d}_i - \mathbf{d}_j$ and its norm $|\mathbf{d}_{ij}|$ to achieve
 114 spatial invariance.

115

116 2.2 MESHGRAPHNETS

117 MeshGraphNets (MGN) (Pfaff et al., 2020) is a GNN model designed to predict the dynamics of
 118 physical systems based on mesh simulations. The model first encodes the physical simulation data
 119 as graphs. Then, it updates node and edge embeddings through multi-layer message passing and
 120 predicts the physical quantities at the next time step based on the embeddings.

121

122 The processor, which plays a central role in this message-passing mechanism, is composed of L
 123 message-passing blocks. Each block sequentially performs edge and node updates to propagate in-
 124 formation through the graph. Specifically, the edge embedding at layer $l + 1$ is updated through f_E ,
 125 which takes as input two node embeddings at layer l and the edge embedding connecting them. Next,
 126 the node embedding at layer $l + 1$ is updated through f_V , which takes as input the node embedding
 127 at layer l and the updated edge embedding at layer $l + 1$. The detailed procedure of the processor is
 128 as follows:

129

$$\mathbf{e}_{ij}^{(\ell+1)} = f_E \left(\mathbf{e}_{ij}^{(\ell)}, \mathbf{h}_i^{(\ell)}, \mathbf{h}_j^{(\ell)} \right), \quad \mathbf{h}_i^{(\ell+1)} = f_V \left(\mathbf{h}_i^{(\ell)}, \sum_{j=1}^n \hat{A}_{ij} \mathbf{e}_{ij}^{(\ell+1)} \right), \quad (1)$$

130

131

132 where $\mathbf{h}_i^{(l)}$ and $\mathbf{h}_j^{(l)}$ denote the node embeddings at layer l , and $\mathbf{e}_{ij}^{(l)}$ denotes the edge embedding at
 133 layer l . f_E and f_V are implemented as multi-layer perceptrons (MLPs) with residual connections.

134

135 2.3 OLLIVIER–RICCI CURVATURE ON GRAPHS

136

137 The Ricci curvature in differential geometry represents the dispersion of geodesics on a Riemann-
 138 nian manifold. Ollivier–Ricci curvature (ORC) (Ollivier, 2009) extends Ricci curvature to graphs by
 139 replacing geodesics with shortest paths between nodes, and by interpreting dispersion in terms of
 140 the probability distribution of a random walk. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and nodes $i, j \in \mathcal{V}$, the
 141 Ollivier–Ricci curvature (ORC) $\kappa(i, j)$ of an edge $(i, j) \in \mathcal{E}$ is computed as follows:

142

143

$$\kappa(i, j) = 1 - \frac{W_1(P_i, P_j)}{d_{\mathcal{G}}(i, j)}, \quad (2)$$

144

145 where W_1 is the 1st-order Wasserstein distance, P_i denotes the probability distribution of a random
 146 walk starting from node i , and $d_{\mathcal{G}}(i, j)$ is the shortest path distance between nodes i and j . The
 147 1st-order Wasserstein distance $W_1(P_i, P_j)$ between P_i and P_j is computed as follows:

148

149

150

$$W_1(P_i, P_j) = \inf_{\pi \in \Pi(P_i, P_j)} \left(\sum_{(p, q) \in \mathcal{V}^2} \pi(p, q) d_{\mathcal{G}}(p, q) \right), \quad (3)$$

151

152

153

154 where $\Pi(P_i, P_j)$ is the set of joint probability distributions that have P_i and P_j as their marginals.
 155 The probability $P_i(p)$ that a 1-step random walker starting from node i reaches node p is defined as
 156 follows:

157

158

159

$$P_i(p) = \begin{cases} \frac{1}{\deg(i)} & \text{if } p \in \mathcal{N}_i \\ 0 & \text{if } p \notin \mathcal{N}_i, \end{cases} \quad (4)$$

160

161

162 where $\deg(i)$ denotes the degree of node i , and \mathcal{N}_i represents the set of neighboring nodes of i .
 163 Equation 4 indicates that the random walker moves to one of the neighboring nodes with equal
 164 probability.

165 The ORC $\kappa(i, j)$ represents the degree of dispersion of geodesics, with different ranges indicating
 166 distinct structural implications of information flow. Its value indicates whether information is likely
 167 to converge, flow stably, or diverge as follows: 1) $\kappa(i, j) > 0$ (Convergence): When the ORC value

162 is positive, geodesics tend to converge. This suggests that information is likely to concentrate at
 163 certain points, which can lead to efficient integration and processing. **2)** $\kappa(i, j) = 0$ (Stable Flow):
 164 A zero value indicates that geodesics remain parallel. This implies a stable and uniform flow of
 165 information without the formation of bottlenecks. **3)** $\kappa(i, j) < 0$ (Divergence): When the ORC
 166 value is negative, geodesics tend to diverge. This suggests the presence of bottlenecks or structural
 167 constraints that can reduce the efficiency of information transfer. It is also worth noting that Topping
 168 et al. (2022) observed that highly negative ORC values can contribute to the over-squashing problem,
 169 a phenomenon where information becomes compressed and difficult to propagate effectively.
 170

3 METHODOLOGY

3.1 ANALYSIS OF THE OVER-SQUASHING PHENOMENON IN MESH-BASED GNN

174 We provide a theoretical analysis of how well MGN captures long-range interactions in scenarios
 175 with a large number of distant neighbor nodes. We assume that the graph \mathcal{G} has node features $X \in$
 176 $\mathbb{R}^{n \times p_0}$, where $\mathbf{x}_i \in \mathbb{R}^{p_0}$ is the feature vector of node $i = 1, \dots, n = |\mathcal{V}|$. The hidden representations
 177 $\mathbf{h}_i^{(\ell)}$ and $\mathbf{e}_{ij}^{(\ell)}$, as computed by Equation 1, are differentiable with respect to the input node features
 178 $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, provided that f_V and f_E are differentiable functions. We evaluate how much the node
 179 $\mathbf{h}_i^{(\ell)}$ and edge $\mathbf{e}_{ij}^{(\ell)}$ are influenced by the input features \mathbf{x}_s of a node s located at distance r from node
 180 i . To this end, we utilize the Jacobians $\partial \mathbf{h}_i^{(r)} / \partial \mathbf{x}_s$ and $\partial \mathbf{e}_{ij}^{(r)} / \partial \mathbf{x}_s$ as follows.
 181

182 **Lemma 1.** *Assume a message-passing scheme for mesh simulation in Equation 1. Let $i, j, s \in \mathcal{V}$ be
 183 nodes in the graph \mathcal{G} , where j is a neighbor of i and the s is an r -hop neighbor of i , i.e., $j \in \mathcal{N}_i$ and
 184 $d_{\mathcal{G}}(i, s) = r$. If $|\partial_2 f_V| \leq \alpha_e$, $|\partial_3 f_E| \leq \beta_h$ for $0 \leq l \leq r - 1$, then*

$$185 \left| \frac{\partial \mathbf{h}_i^{(r)}}{\partial \mathbf{x}_s} \right| \leq (\alpha_e \beta_h)^r \left(\hat{A}^r \right)_{is}, \quad \left| \frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s} \right| \leq \alpha_e^{r-1} \beta_h^r \left(\hat{A}^{r-1} \right)_{js}.$$

189 *Proof.* Since $d_{\mathcal{G}}(i, s) = r$, note that the Jacobians $\frac{\partial \mathbf{h}_i^{(r-1)}}{\partial \mathbf{x}_s}$, $\frac{\partial \mathbf{h}_j^{(r-2)}}{\partial \mathbf{x}_s}$ and $\frac{\partial \mathbf{e}_{ij}^{(r-1)}}{\partial \mathbf{x}_s}$ are zero matrices.
 190

191 Then, we can recursively expand $\frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s}$ as follows:

$$192 \frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s} = \frac{\partial f_E}{\partial \mathbf{e}_{ij}^{(r-1)}} \frac{\partial \mathbf{e}_{ij}^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_E}{\partial \mathbf{h}_i^{(r-1)}} \frac{\partial \mathbf{h}_i^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_E}{\partial \mathbf{h}_j^{(r-1)}} \frac{\partial \mathbf{h}_j^{(r-1)}}{\partial \mathbf{x}_s} \\ 193 = \frac{\partial f_E}{\partial \mathbf{e}_{ij}^{(r-1)}} \frac{\partial \mathbf{e}_{ij}^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_E}{\partial \mathbf{h}_i^{(r-1)}} \frac{\partial \mathbf{h}_i^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_E}{\partial \mathbf{h}_j^{(r-1)}} \left(\frac{\partial f_V}{\partial \mathbf{h}_j^{(r-2)}} \frac{\partial \mathbf{h}_j^{(r-2)}}{\partial \mathbf{x}_s} + \frac{\partial f_V}{\partial \mathbf{z}_j^{(r-1)}} \sum_k \hat{A}_{jk} \frac{\partial \mathbf{e}_{jk}^{(r-1)}}{\partial \mathbf{x}_s} \right) \\ 194 = \frac{\partial f_E}{\partial \mathbf{h}_j^{(r-1)}} \frac{\partial f_V}{\partial \mathbf{z}_j^{(r-1)}} \sum_k \hat{A}_{jk} \frac{\partial \mathbf{e}_{jk}^{(r-1)}}{\partial \mathbf{x}_s} = \dots = \sum_{j_2, \dots, j_{r-1}} \hat{A}_{jj_2} \hat{A}_{j_2 j_3} \dots \hat{A}_{j_{r-1} s} \cdot J_{i j_2 \dots j_{r-1} s}(X),$$

200 where $\mathbf{z}_j^{(r-1)} = \sum_k \hat{A}_{jk} \mathbf{e}_{jk}^{(r-1)}$ and $J_{i j_2 \dots j_{r-1} s}(X)$ is the product of r third partial derivatives of
 201 f_E and $r - 1$ second partial derivatives of f_V with j_l indicating the index of i 's l -hop neighbors.
 202

203 Since $|J_{i j_2 \dots j_{r-1} s}(X)| \leq \alpha_e^{r-1} \beta_h^r$ holds, we obtain

$$204 \left| \frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s} \right| \leq \sum_{j_2, \dots, j_{r-1}} \hat{A}_{jj_2} \hat{A}_{j_2 j_3} \dots \hat{A}_{j_{r-1} s} \alpha_e^{r-1} \beta_h^r = \alpha_e^{r-1} \beta_h^r \left(\hat{A}^{r-1} \right)_{js}.$$

208 Using this result, we can also derive the upper bound of $\left| \frac{\partial \mathbf{h}_i^{(r)}}{\partial \mathbf{x}_s} \right|$ as follows:

$$210 \left| \frac{\partial \mathbf{h}_i^{(r)}}{\partial \mathbf{x}_s} \right| = \left| \frac{\partial f_V}{\partial \mathbf{h}_i^{(r-1)}} \frac{\partial \mathbf{h}_i^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_V}{\partial \mathbf{z}_i^{(r)}} \sum_j \hat{A}_{ij} \frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s} \right| \\ 211 \leq \alpha_e \sum_{j, j_2, \dots, j_{r-1}} \hat{A}_{ij} \hat{A}_{j j_2} \dots \hat{A}_{j_{r-1} s} \alpha_e^{r-1} \beta_h^r = (\alpha_e \beta_h)^r \left(\hat{A}^r \right)_{is}.$$

213 \square

Lemma 1 shows that if f_V and f_E have bounded derivatives, the extent of message propagation in a mesh-based GNN is controlled by powers of \hat{A} . Intuitively, as the hop distance r increases, the number of r -hop neighbors within the receptive field $B_r(i) = \{j \in \mathcal{V} \mid d_{\mathcal{G}}(i, j) \leq r\}$ grows rapidly. Because information from this expanding set of neighbors must ultimately be compressed into a fixed-size vector, the influence of each individual neighbor necessarily diminishes with increasing r . This diminishing influence is reflected in the upper bound of the Jacobian terms derived in Lemma 1, which decay exponentially as a function of the hop distance. This result is precisely what gives rise to the over-squashing phenomenon. More detailed derivation is provided in Appendix A.

3.2 ADAPTIVE GRAPH REWIRING IN MESH-BASED GNN

To address the over-squashing problem analyzed in Section 3.1, we propose a novel graph rewiring method for mesh simulations. Recall that in existing rewiring methods (Gasteiger et al., 2019; Karhadkar et al., 2023; Nguyen et al., 2023; Yu et al., 2025), all rewiring occurs before GNN training. This causes two distant particles to interact instantaneously, as if they were neighboring particles, which does not sufficiently reflect actual physical conditions. For example, boundary layers or turbulence that can affect distant particles propagate their influence sequentially through collisions between adjacent particles, leading to a certain delay before the influence reaches the distant particles. Therefore, we propose adaptive graph rewiring that considers the gradual propagation of physical interactions in mesh-based GNN. Figure 1 illustrates the differences between existing rewiring methods and our proposed approach within a mesh graph.

3.2.1 PREPROCESSING

Identifying bottleneck nodes. We identify bottleneck nodes that cause over-squashing in the mesh graph based on ORC. We calculate obtain the node-level curvature γ_i to summarize the local geometry around node i as follows:

$$\gamma_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \kappa(i, j), \quad (5)$$

which represents the mean of the κ values of all edges connected to node i . Based on the computed γ_i , we define the set of nodes $\mathcal{V}_{\text{lowORC}} \subset \mathcal{V}$ whose curvatures belong to the lowest $a\%$ as follows:

$$\mathcal{V}_{\text{lowORC}} = \{v_i \in \mathcal{V} \mid \gamma_i \leq \text{Percentile}_a(\{\gamma_j\}_{j \in \mathcal{V}})\}. \quad (6)$$

Calculating the rewiring delay score. To avoid performing all rewiring at once, we dynamically rewire each edge during the message-passing process. Therefore, we aim to compute the rewiring delay score, which indicates the degree of delay required to rewire each new edge. To this end, we first select the optimal connection pair for each bottleneck node to resolve the bottleneck. Yu et al. (2025) has demonstrated that rewiring nodes with large velocity differences can effectively resolve the over-squashing problem in fluid simulations. Inspired by this, we determine the optimal connection node v_{i^*} based on the velocity difference for each $v_i \in \mathcal{V}_{\text{lowORC}}$ as follows:

$$i^* = \underset{j \text{ s.t. } v_j \in \mathcal{V} \setminus \{v_i\}}{\text{argmax}} \|\mathbf{v}_i - \mathbf{v}_j\| \quad \forall v_i \in \mathcal{V}_{\text{lowORC}}, \quad (7)$$

where \mathbf{v}_i and \mathbf{v}_j represent the velocities of v_i and v_j , respectively. Finally, we compute the rewiring delay score $s_{\text{delay}}(i, i^*)$ based on the velocity difference between v_i and v_{i^*} , and the shortest path distance $d_{\mathcal{G}}$ as follows:

$$s_{\text{delay}}(i, i^*) = \min \left(\frac{\beta \cdot d_{\mathcal{G}}(i, i^*)}{\|\mathbf{v}_i - \mathbf{v}_{i^*}\|}, L \right), \quad (8)$$

where L represents the total number of message-passing blocks and β is a hyperparameter that controls the influence of distance and velocity. $s_{\text{delay}}(i, i^*)$ represents the degree of delay required for v_i and v_{i^*} to be rewired, and it determines the layer index at which the two nodes are rewired during the message-passing process. As the distance increases, s_{delay} increases, and conversely, as the velocity difference increases, s_{delay} decreases. Specifically, as distance increases, long-range interactions require more time to propagate, so we set s_{delay} to a larger value. Additionally, as the velocity difference increases, the bottleneck node have a greater influence on long-range interactions, so we set s_{delay} to a smaller value.

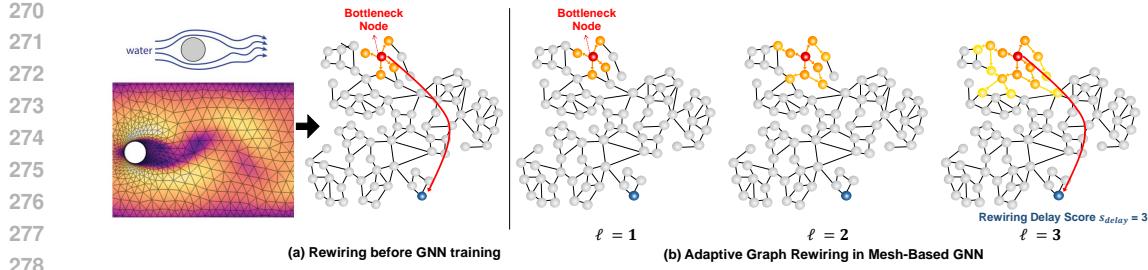


Figure 1: Comparison of static rewiring and adaptive graph rewiring (AdaMeshNet).

3.2.2 ENCODER

The encoder maps the node v_i and edge e_{ij} into latent vectors using a Multi-Layer Perceptron (MLP). Specifically, the node and edge embeddings are denoted as \mathbf{h}_i and \mathbf{e}_{ij} , respectively, and are obtained via separate MLPs as follows:

$$\mathbf{h}_i = \text{MLP}_v(v_i), \quad \mathbf{e}_{ij} = \text{MLP}_e(e_{ij}). \quad (9)$$

3.2.3 PROCESSOR

Updating nodes for rewiring. We update the neighboring nodes based on the rewiring delay score s_{delay} computed in Section 3.2.1. This update process is performed at each layer, and the overall update procedure is as follows:

$$\mathcal{N}_i^0 = \{j | (i, j) \in \mathcal{E}\}, \quad \mathcal{N}_i^{l+1} = \mathcal{N}_i^l \cup \{i^* | l < s_{\text{delay}}(i, i^*) \leq l + 1\}. \quad (10)$$

Specifically, the initial neighboring nodes are the same as the neighbors connected by the edges \mathcal{E} derived from the mesh graph. As layer l increases, new neighboring nodes are added to the neighbor set \mathcal{N}_i^l based on the s_{delay} , updating \mathcal{N}_i^l . As a result, a neighbor set \mathcal{N}_i^l for v_i is assigned at each layer l , and as l increases, the neighboring nodes within \mathcal{N}_i^l are progressively accumulated.

Edge update. Each message-passing block consists of an edge update and a node update. Each block contains a separate set of network parameters and is applied sequentially to the output of the previous block. The edge embedding \mathbf{e}_{ij}^{l+1} at layer $l + 1$ is updated based on \mathbf{e}_{ij}^l , \mathbf{h}_i^l , and \mathbf{h}_j^l as:

$$\mathbf{e}_{ij}^{l+1} = f_E(\mathbf{e}_{ij}^l, \mathbf{h}_i^l, \mathbf{h}_j^l), \quad j \in \mathcal{N}_i^{l+1}. \quad (11)$$

Note that j used in the edge update belongs to the neighbor set \mathcal{N}_i^{l+1} of v_i , which is determined based on the rewiring delay score s_{delay} . Thus, the edge embedding \mathbf{e}_{ij}^{l+1} is updated using the newly rewired neighboring nodes at each layer.

Node update. Next, the node embedding \mathbf{h}_i^{l+1} at layer $l + 1$ is updated based on \mathbf{h}_i^l and \mathbf{e}_{ij}^{l+1} as:

$$\mathbf{h}_i^{l+1} = f_V \left(\mathbf{h}_i^l, \sum_{j \in \mathcal{N}_i^{l+1}} \mathbf{e}_{ij}^{l+1} \right), \quad (12)$$

where \mathbf{e}_{ij}^{l+1} is the edge embedding obtained from the edge update. Note that j in \mathbf{e}_{ij}^{l+1} belongs to \mathcal{N}_i^{l+1} , which is determined based on s_{delay} . Thus, the node embedding \mathbf{h}_i^{l+1} is updated using the newly rewired edges at each layer.

3.2.4 DECODER AND STATE UPDATER

To predict the state at time $t + 1$ from time t , the decoder uses an MLP to transform the outputs o_i , such as the velocity gradient $\hat{\mathbf{v}}_i$, density gradient $\hat{\rho}_i$, and pressure gradient \hat{p}_i . The updator computes the dynamic quantity \hat{q}_i^{t+1} at the next step based on the outputs o_i obtained from the decoder, using the forward-Euler integrator. For example, the velocity $\hat{\mathbf{v}}_i^t$ is used to compute the velocity $\hat{\mathbf{v}}_i^{t+1}$ at time $t + 1$ as follows:

$$\hat{\mathbf{v}}_i^{t+1} = \hat{\mathbf{v}}_i^t + \mathbf{v}_i^t. \quad (13)$$

Finally, the output nodes \mathcal{V} are updated using q_i^{t+1} , and \mathcal{M}_{t+1} is generated based on the updated output nodes \mathcal{V} .

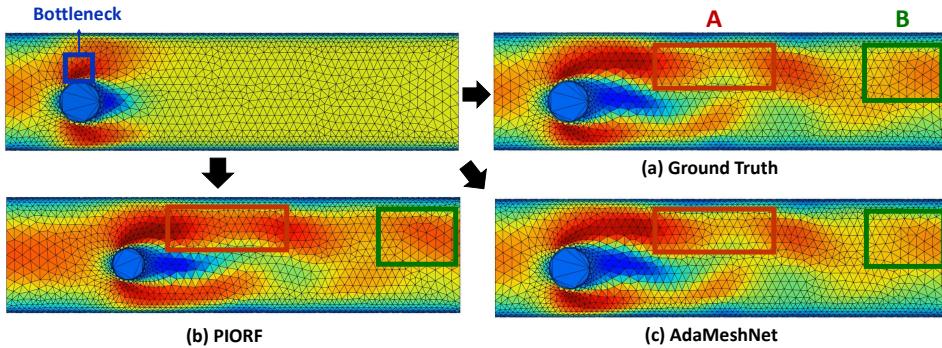


Figure 2: Physical interpretation based on visualization in Cylinder Flow.

3.2.5 PHYSICAL INTERPRETATION BASED ON VISUALIZATION

One of the most effective methods for analyzing fluid motion is to visualize velocity contours from fluid simulations. In Figure 2, we visualize how velocity contours propagate from the initial state in a Cylinder Flow. We compare our proposed model, AdaMeshNet, with a state-of-the-art static rewiring method, PIORF (Yu et al., 2025). In the visualizations, a red mesh indicates high velocity values, while a blue mesh indicates low velocity values. In Figure 2(b), the PIORF method fails to accurately capture the wavelike propagation of velocity in Region A. In Region B, PIORF generates an overshooting phenomenon producing velocities faster than the ground truth, since it instantly transmits interactions as if they were from adjacent particles, without considering the inherent delay. In contrast, Figure 2(c) shows that AdaMeshNet produces velocity values that are very similar to the ground truth by mimicking the physical reality of the gradual propagation of long-range interactions. Ultimately, AdaMeshNet models key fluid dynamics principles of physical interaction delay and propagation, going beyond simple graph structure improvements to enable predictions that are closer to real-world simulations.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUPS

Datasets. For evaluation of the models, we use Cylinderflow and Airfoil. They all operate on the basis of the Navier–Stokes equations (Temam, 1977), but the fluid behaves differently in each case. Specifically, CylinderFlow exhibits a laminar flow, where fluid particles move in a regular and orderly manner, whereas Airfoil produces a high-speed turbulent model, where fluid particles move in a disordered manner. Each dataset includes 1,000 flow results, each with 600 time steps. Details on datasets can be found in Appendix C.

Baselines. We use DIGL (Gasteiger et al., 2019), SDRF (Topping et al., 2022), BORF (Nguyen et al., 2023), and PIORF (Yu et al., 2025) as baselines. All of these baselines follow a static rewiring approach, completing all rewiring before applying the GNN. In our experiments, these methods were implemented based on the MGN model (Pfaff et al., 2020) as the backbone. For all models, we used 15 message-passing layers and set the hidden vector size of MLPs to 128. Details on baselines can be found in Appendix B.

4.2 PREDICTION OF PHYSICAL QUANTITIES

Tables 1 and 2 show the results of physical quantity predictions for the Cylinder Flow and Airfoil datasets, respectively. We measured the root-mean-square error (RMSE) for velocity, pressure, and density across a single prediction step, a 50-step rollout, and the full trajectory rollout. AdaMeshNet achieved the lowest RMSE across all metrics when compared to existing static graph rewiring methods. The superior performance of AdaMeshNet on both datasets indicates its effectiveness in predicting both laminar and turbulent flows. This demonstrates the efficiency of our fluid dynamics simulation method, which adaptively connects new edges based on rewiring delay scores.

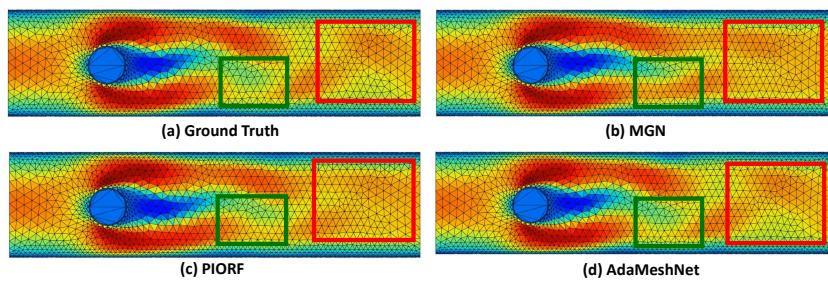
Figures 3 and 5 present visualizations of the velocity magnitude contours for two additional datasets. The red mesh indicates high velocity values, while the blue mesh indicates low velocity values. The

378
379
380
381 Table 1: RMSE results on the Cylinder Flow dataset.
382
383
384
385
386

Method	velocity ($\times 10^{-3}$)			pressure ($\times 10^{-3}$)		
	1-step	rollout-50	rollout-all	1-step	rollout-50	rollout-all
MGN	2.95 \pm 0.99	9.43 \pm 4.36	53.23 \pm 39.24	97.18 \pm 20.85	26.02 \pm 4.49	11.03 \pm 6.25
DIGL	2.64 \pm 1.53	10.50 \pm 6.79	62.35 \pm 40.36	98.62 \pm 22.53	26.47 \pm 5.24	11.47 \pm 5.93
SDRF	2.45 \pm 0.54	7.53 \pm 3.52	49.23 \pm 41.93	73.53 \pm 21.76	24.68 \pm 5.63	9.32 \pm 6.16
BORF	2.34 \pm 0.12	6.30 \pm 3.70	48.10 \pm 37.20	64.74 \pm 20.82	20.72 \pm 7.52	9.36 \pm 7.95
PIORF	1.97 \pm 0.78	7.68 \pm 3.18	47.88 \pm 38.59	57.46 \pm 19.92	19.25 \pm 8.03	7.74 \pm 5.31
AdaMeshNet	1.69 \pm 0.56	5.21 \pm 2.97	40.37 \pm 38.82	48.15 \pm 19.48	12.47 \pm 7.18	5.86 \pm 4.49

387
388
389
390 Table 2: RMSE results on the Airfoil dataset.
391
392
393
394
395

Method	velocity			density ($\times 10^{-2}$)		
	1-step	rollout-50	rollout-all	1-step	rollout-50	rollout-all
MGN	9.42 \pm 3.13	22.34 \pm 8.39	61.42 \pm 32.35	13.14 \pm 5.13	13.88 \pm 5.93	15.14 \pm 6.49
DIGL	9.47 \pm 3.46	20.73 \pm 7.35	63.75 \pm 29.52	11.91 \pm 4.24	12.47 \pm 5.79	14.93 \pm 6.39
SDRF	7.09 \pm 2.75	15.24 \pm 3.90	44.25 \pm 41.66	13.30 \pm 4.82	14.93 \pm 5.14	16.38 \pm 5.92
BORF	7.51 \pm 3.27	16.33 \pm 2.88	58.24 \pm 28.32	8.01 \pm 1.95	7.91 \pm 3.44	9.81 \pm 4.21
PIORF	6.42 \pm 2.25	14.37 \pm 3.95	47.52 \pm 35.48	9.15 \pm 2.20	10.03 \pm 4.39	12.20 \pm 6.13
AdaMeshNet	3.25 \pm 1.04	7.76 \pm 6.25	28.67 \pm 30.46	4.98 \pm 2.31	4.87 \pm 2.47	7.01 \pm 5.16

400
401
402
403
404 Figure 3: Velocity magnitude contours on the Cylinder Flow dataset.
405
406

407 red and green boxes in these figures highlight that our method produces velocity contours that are
408 more similar to the ground truth. Specifically, our approach more accurately visualizes the wavelike
409 propagation of velocity to neighboring nodes compared to other methods. This is because our
410 adaptive graph rewiring module more precisely considers inter-particle interactions, allowing it to capture
411 long-range interactions more effectively. Please refer to Section E for more velocity contours.

412
413 4.3 ABLATION STUDIES

414 Figure 4 shows the results of ablation studies to examine the effectiveness of our proposed model.
415 Specifically, we perform ablation studies by excluding the distance term d_g in the numerator (i.e.,
416 w/o d_g), and the velocity difference term $|\mathbf{v}_i - \mathbf{v}_{i^*}|$ in the denominator (i.e., w/o velocity) from
417 Equation 8. We also evaluate the model performance by incorporating the information regarding
418 d_g into the edge weight without including d_g in Equation 8 (i.e., weighted edges). We obtained
419 the following observations: **1)** Excluding d_g and \mathbf{v} from s_{delay} leads to a performance degradation
420 compared to our final model. In particular, removing d_g significantly reduces performance, since
421 the distance information between two nodes is no longer considered when new edges are connected.
422 This indicates that distance information must be sufficiently accounted for when computing the
423 degree of rewiring delay. **2)** Including d_g as an edge weight does not substantially improve performance.
424 This is because, unlike s_{delay} , edge weights cannot explicitly consider the rewiring delay.
425 This result highlights that considering temporal delay based on distance information contributes to
426 performance improvement. **3)** The final model with all components included achieves the best per-
427 formance. This demonstrates that our adaptive rewiring approach, which considers temporal delay
428 and gradual propagation based on both velocity and distance, is the most effective.

428
429 4.4 HYPERPARAMETER ANALYSIS

430 In this section, we analyze the sensitivity to the pooling ratio α in Equation 6, which determines
431 the number of edges to be rewired, and the hyperparameter β in Equation 8, which represents the
432 influence of distance and velocity.

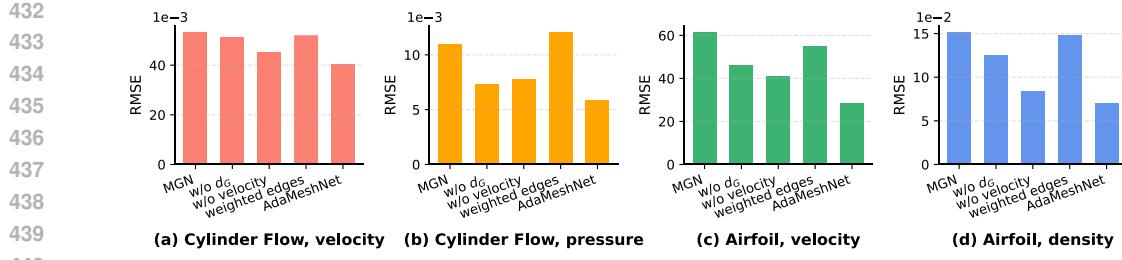


Figure 4: Ablation studies on Cylinder Flow and Airfoil.

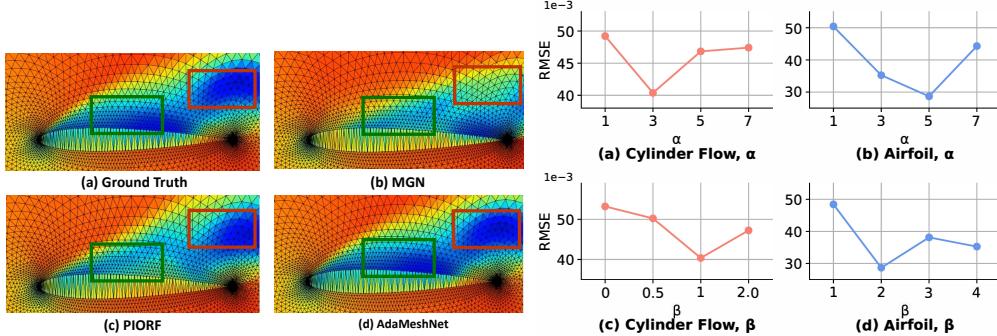


Figure 5: Velocity magnitude contours on Airfoil.

Figure 6: Impact of α and β .

Effect of pooling ratio α . Figure 6(a) and (b) show the velocity RMSE for rollout-all over various α s. The results show that for the CylinderFlow dataset, the lowest RMSE is achieved at $\alpha = 3\%$, while for the Airfoil dataset, the optimal performance is achieved at $\alpha = 5\%$. These findings indicate that if α is too low, the number of newly rewired nodes is insufficient to effectively capture long-range interactions. Conversely, if α is too high, the model risks losing the original graph topology. This analysis highlights the importance of selecting an optimal α value to balance the preservation of original structure with the ability to capture broader, long-range dependencies. Regarding the training time analysis according to the alpha value, please refer to the Appendix D.

Effect of hyperparameter β . Figure 6(c) and (d) show the velocity RMSE for rollout-all over various β s. The results indicate that the lowest RMSE is achieved for the Cylinder Flow when $\beta = 1$, while for the Airfoil, the optimal performance is achieved at $\beta = 2$. A lower β value places relatively more weight on the influence of velocity than on distance in determining s_{delay} , whereas a higher β places more weight on distance than on velocity. Airfoil has a wider range of particle velocity values compared to the Cylinder Flow, which can cause the influence of velocity to become overly dominant. To reduce this effect, the optimal β is a higher value that increases the influence of distance d_G . This demonstrates that the optimal β value can be controlled by adjusting the relative influence of velocity and distance, allowing our method to adapt to different graph properties such as velocity distribution.

5 CONCLUSION

In this work, we addressed the over-squashing problem inherent in MeshGraphNets (MGN) for fluid dynamics simulations by introducing AdaMeshNet, a novel adaptive graph rewiring framework. Unlike previous static rewiring methods that treat distant nodes as immediate neighbors, our approach adaptively rewrites edges during the message-passing process, considering the gradual propagation of physical interactions. We propose a new rewiring delay score based on velocity difference and inter-node distance. This score determines the layer at which new edges are added, allowing our model to more realistically simulate the time-delayed effects of long-range interactions. Experimental results confirm that AdaMeshNet outperforms existing static rewiring methods, and our visualizations highlight its superior ability to accurately capture complex flow phenomena. This work represents a significant step forward in developing more accurate and physically-grounded GNNs for computational fluid dynamics.

486
487
ETHIC STATEMENT488
489
490
491
492
493
494
This research complies with the ICLR Code of Ethics. All experiments were performed on pub-
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
licly accessible and widely adopted benchmark datasets, which contain no personally identifiable
or sensitive information, thereby minimizing privacy concerns. Our research is intended to enhance
application in real-world fluid dynamics without enabling harmful uses or misuse. We are dedicated
to maintaining scientific integrity and provide anonymized source code to guarantee transparency
and reproducibility. After careful consideration of the potential impacts of this work, we conclude
that it does not present notable ethical concerns.501
502
503
504
505
506
REPRODUCIBILITY STATEMENT507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
We provide all the necessary details in Section 4 and the Appendix to ensure the reproducibil-
ity of our study. In addition, our source code is available at <https://anonymous.4open.science/r/AdaMeshNet-9321>.501
502
503
504
505
506
REFERENCES507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Hugo Attali, Davide Buscaldi, and Nathalie Pernelle. Delaunay graph: Addressing over-squashing
and over-smoothing using delaunay triangulation. In *Proceedings of the 41st International Con-
ference on Machine Learning*, 2024.507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Timothy Baker. On the relationship between mesh refinement and solution accuracy. In *17th AIAA
Computational Fluid Dynamics Conference*, 2005.507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Federico Barbero, Ameya Velingker, Amin Saberi, Michael Bronstein, and Francesco Di Giovanni.
Locality-aware graph rewiring in GNNs. In *Proceedings of the 12th International Conference on
Learning Representations*, 2024.507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Predic-
tion of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*,
64(2):525–545, 2019.507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in
gnns through the lens of effective resistance. In *Proceedings of the 40th International Conference
on Machine Learning*, 2023.507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Emmanuel De Bézenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes:
Incorporating prior scientific knowledge. In *Proceedings of the 6th International Conference on
Learning Representations*, 2018.507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Gouri Dhatt, Emmanuel Lefrançois, and Gilbert Touzot. *Finite element method*. John Wiley & Sons,
2012.507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael
Bronstein. On over-squashing in message passing neural networks: The impact of width, depth,
and topology. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Federico Errica, Henrik Christiansen, Viktor Zaverkin, Takashi Maruyama, Mathias Niepert, and
Francesco Alesiani. Adaptive message passing: A general framework to mitigate oversmoothing,
oversquashing, and underreaching. *arXiv preprint arXiv:2312.16560*, 2023.507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Lukas Fesser and Melanie Weber. Mitigating over-smoothing and over-squashing using augmenta-
tions of forman-ricci curvature. In *Learning on Graphs Conference*, 2023.507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Ben Finkelshtein, Xingyue Huang, Michael Bronstein, and Ismail Ilkan Ceylan. Cooperative graph
neural networks. *arXiv preprint arXiv:2310.01267*, 2023.507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. MultiScale
MeshGraphNets. *arXiv preprint arXiv:2210.00612*, 2022.507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph
learning. *Advances in Neural Information Processing Systems* 32, 2019.

540 Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow ap-
 541 proximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge*
 542 *Discovery and Data Mining*, 2016.

543 Daniel Holden, Bang Chi Duong, Sayantan Datta, and Derek Nowrouzezahrai. Subspace neural
 544 physics: Fast data-driven interactive simulation. In *Proceedings of the 18th annual ACM SIG-*
 545 *GRAPH/Eurographics Symposium on Computer Animation*, 2019.

546 Kedar Karhadkar, Pradeep Kr. Banerjee, and Guido Montúfar. FoSR: First-order spectral rewiring
 547 for addressing oversquashing in GNNs. In *Proceedings of the 11th International Conference on*
 548 *Learning Representations*, 2023.

549 Aaron Katz and Venkateswaran Sankaran. Mesh quality effects on the accuracy of CFD solutions
 550 on unstructured meshes. *Journal of Computational Physics*, 230(20):7670–7686, 2011.

551 Sangseung Lee and Donghyun You. Data-driven prediction of unsteady flow over a circular cylinder
 552 using deep learning. *Journal of Fluid Mechanics*, 879:217–254, 2019.

553 Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua Tenenbaum, and Antonio Torralba. Learning particle
 554 dynamics for manipulating rigid bodies, deformable objects, and fluids. In *Proceedings of the 7th*
 555 *International Conference on Learning Representations*, 2019.

556 Lu Liu, Jie Wu, and Shunying Ji. DEM-SPH coupling method for the interaction between irregularly
 557 shaped granular materials and fluids. *Powder Technology*, 400:117249, 2022.

558 Rainald Löhner. Mesh adaptation in fluid mechanics. *Engineering Fracture Mechanics*, 50(5-6):
 559 819–847, 1995.

560 Ran Luo, Tianjia Shao, Huamin Wang, Weiwei Xu, Xiang Chen, Kun Zhou, and Yin Yang. NNWarp:
 561 Neural network-based nonlinear deformation. *IEEE Transactions on Visualization and Computer*
 562 *Graphics*, 26(4):1745–1759, 2018.

563 Erdogan Madenci and Ibrahim Guven. *The finite element method and applications in engineering*
 564 using ANSYS®. Springer, 2006.

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

568 Khang Nguyen, Nong Minh Hieu, Vinh Duc Nguyen, Nhat Ho, Stanley Osher, and Tan Minh
 569 Nguyen. Revisiting over-smoothing and over-squashing using Ollivier-Ricci curvature. In *Pro-*
 570 *ceedings of the 40th International Conference on Machine Learning*, 2023.

571 Yann Ollivier. Ricci curvature of Markov chains on metric spaces. *Journal of Functional Analysis*,
 572 256(3):810–864, 2009.

573 Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based
 574 simulation with graph networks. In *Proceedings of the 8th International Conference on Learning*
 575 *Representations*, 2020.

576 Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter
 577 Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the*
 578 *37th International Conference on Machine Learning*, 2020.

579 Dai Shi, Andi Han, Lequan Lin, Yi Guo, and Junbin Gao. Exposition on over-squashing problem on
 580 gnns: Current methods, benchmarks and challenges. *arXiv preprint arXiv:2311.07073*, 2023.

581 Tadeusz Stolarski, Yuji Nakasone, and Shigeka Yoshimoto. *Engineering analysis with ANSYS soft-*
 582 *ware*. Butterworth-Heinemann, 2018.

583 Roger Temam. *Navier-Stokes Equations: Theory and Numerical Analysis*. American Mathematical
 584 Society, 1977.

585 Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep learning methods for
 586 Reynolds-averaged Navier-Stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020.

594 Jake Topping, Francesco Di Giovanni, Benjamin Chamberlain, Xiaowen Dong, and Michael Bron-
 595 stein. Understanding over-squashing and bottlenecks on graphs via curvature. In *Proceedings of*
 596 *the 10th International Conference on Learning Representations*, 2022.

597 Domenico Tortorella and Alessio Micheli. Leave graphs alone: Addressing over-squashing without
 598 rewiring. *arXiv preprint arXiv:2212.06538*, 2022.

600 Kiwon Um, Xiangyu Hu, and Nils Thuerey. Liquid splash modeling with neural networks. *Computer*
 601 *Graphics Forum*, 37(8):171–182, 2018.

602 Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-
 603 informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD*
 604 *International Conference on Knowledge Discovery and Data Mining*, 2020.

606 Steffen Wiewel, Moritz Becher, and Nils Thürey. Latent space physics: Towards learning the tem-
 607 poral evolution of fluid flow. *Computer Graphics Forum*, 38(2):71–82, 2019.

608 You Xie, Aleksandra Franz, Mengyu Chu, and Nils Thuerey. tempoGAN: A temporally coherent,
 609 volumetric GAN for super-resolution fluid flow. *ACM Transactions on Graphics*, 37(95):1–15,
 610 2018.

611 Youn-Yeol Yu, Jeongwhan Choi, Jaehyeon Park, Kookjin Lee, and Noseong Park. PIORF: Physics-
 612 informed Ollivier-Ricci flow for long-range interactions in mesh graph neural networks. In *Pro-
 613 ceedings of the 13th International Conference on Learning Representations*, 2025.

615 Yao Zhang, Woong Je Sung, and Dimitri N Mavris. Application of convolutional neural network
 616 to predict airfoil lift coefficient. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics,*
 617 *and Materials Conference*, 2018.

618
 619
 620
 621
 622
 623
 624
 625
 626
 627
 628
 629
 630
 631
 632
 633
 634
 635
 636
 637
 638
 639
 640
 641
 642
 643
 644
 645
 646
 647

648 A DETAILED PROOF OF LEMMA 1
649650 The following equations describe the message-passing scheme used in MGN:
651

652
$$\mathbf{e}_{ij}^{(r)} = f_E \left(\mathbf{e}_{ij}^{(r-1)}, \mathbf{h}_i^{(r-1)}, \mathbf{h}_j^{(r-1)} \right), \quad (14)$$

653
$$\mathbf{h}_i^{(r)} = f_V \left(\mathbf{h}_i^{(r-1)}, \sum_{j=1}^n \hat{A}_{ij} \mathbf{e}_{ij}^{(r)} \right) = f_V \left(\mathbf{h}_i^{(r-1)}, \sum_{j=1}^n \hat{A}_{ij} f_E \left(\mathbf{e}_{ij}^{(r-1)}, \mathbf{h}_i^{(r-1)}, \mathbf{h}_j^{(r-1)} \right) \right). \quad (15)$$

656 Based on the above equations, we can expand their derivatives as follows:
657

658
$$\frac{\partial \mathbf{h}_i^{(r)}}{\partial \mathbf{x}_s} = \frac{\partial f_V}{\partial \mathbf{h}_i^{(r-1)}} \frac{\partial \mathbf{h}_i^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_V}{\partial \mathbf{z}_i^{(r)}} \sum_j \hat{A}_{ij} \frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s}, \quad (16)$$

661
$$\frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s} = \frac{\partial f_E}{\partial \mathbf{e}_{ij}^{(r-1)}} \frac{\partial \mathbf{e}_{ij}^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_E}{\partial \mathbf{h}_i^{(r-1)}} \frac{\partial \mathbf{h}_i^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_E}{\partial \mathbf{h}_j^{(r-1)}} \frac{\partial \mathbf{h}_j^{(r-1)}}{\partial \mathbf{x}_s}. \quad (17)$$

664 where $\mathbf{z}_i^{(r)} = \sum_{j=1}^n \hat{A}_{ij} \mathbf{e}_{ij}^{(r)}$.
665666 First, to derive an upper bound of $\left| \frac{\partial \mathbf{h}_i^{(r)}}{\partial \mathbf{x}_s} \right|$, we plug Equation 17 into Equation 16 and obtain the
667 following expression:
668

669
$$\frac{\partial \mathbf{h}_i^{(r)}}{\partial \mathbf{x}_s} = \frac{\partial f_V}{\partial \mathbf{h}_i^{(r-1)}} \frac{\partial \mathbf{h}_i^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_V}{\partial \mathbf{z}_i^{(r)}} \sum_j \hat{A}_{ij} \left(\frac{\partial f_E}{\partial \mathbf{e}_{ij}^{(r-1)}} \frac{\partial \mathbf{e}_{ij}^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_E}{\partial \mathbf{h}_i^{(r-1)}} \frac{\partial \mathbf{h}_i^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_E}{\partial \mathbf{h}_j^{(r-1)}} \frac{\partial \mathbf{h}_j^{(r-1)}}{\partial \mathbf{x}_s} \right). \quad (18)$$

670 Note that s is an r -hop neighbor of i , while $\mathbf{h}_i^{(r-1)}$, $\mathbf{h}_j^{(r-2)}$, and $\mathbf{e}_{ij}^{(r-1)}$ are embeddings made by
671 aggregating the information from up to $(r-1)$ -hop neighbors of i . Thus, the Jacobians $\frac{\partial \mathbf{h}_i^{(r-1)}}{\partial \mathbf{x}_s}$,
672 $\frac{\partial \mathbf{h}_j^{(r-2)}}{\partial \mathbf{x}_s}$ and $\frac{\partial \mathbf{e}_{ij}^{(r-1)}}{\partial \mathbf{x}_s}$ are zero matrices, and this enables us to recursively expand $\frac{\partial \mathbf{h}_i^{(r)}}{\partial \mathbf{x}_s}$ as follows:
673

674
$$\begin{aligned} \frac{\partial \mathbf{h}_i^{(r)}}{\partial \mathbf{x}_s} &= \sum_j \hat{A}_{ij} \frac{\partial f_V}{\partial \mathbf{z}_i^{(r)}} \frac{\partial f_E}{\partial \mathbf{h}_j^{(r-1)}} \frac{\partial \mathbf{h}_j^{(r-1)}}{\partial \mathbf{x}_s} \\ &= \sum_j \hat{A}_{ij} \frac{\partial f_V}{\partial \mathbf{z}_i^{(r)}} \frac{\partial f_E}{\partial \mathbf{h}_j^{(r-1)}} \sum_k \hat{A}_{jk} \frac{\partial f_V}{\partial \mathbf{z}_j^{(r-1)}} \frac{\partial f_E}{\partial \mathbf{h}_k^{(r-2)}} \frac{\partial \mathbf{h}_k^{(r-2)}}{\partial \mathbf{x}_s} \\ &= \dots = \sum_{j_1, \dots, j_r} \hat{A}_{ij_1} \hat{A}_{j_1 j_2} \dots \hat{A}_{j_{r-1} j_r} \cdot J_{ij_1 \dots j_r}(X) \cdot \frac{\partial \mathbf{h}_j^{(0)}}{\partial \mathbf{x}_s}, \end{aligned} \quad (19)$$

680 where $J_{ij_1 \dots j_r}(X)$ represents the product of r second partial derivatives of f_V and r third partial
681 derivatives of f_E with j_l indicating the index of i 's l -hop neighbors. Since $\partial_{\mathbf{x}_s} \mathbf{h}_{j_r}^{(0)} = \partial_{\mathbf{x}_s} \mathbf{x}_{j_r} = \delta_{j_r s}$
682 holds, we obtain
683

684
$$\frac{\partial \mathbf{h}_i^{(r)}}{\partial \mathbf{x}_s} = \sum_{j_1, \dots, j_{r-1}} \hat{A}_{ij_1} \hat{A}_{j_1 j_2} \dots \hat{A}_{j_{r-1} j_r} \cdot J_{ij_1 \dots j_{r-1} s}(X) \quad (20)$$

696 Finally, since $|J_{ij_1 \dots j_{r-1} s}(X)| \leq (\alpha_e \beta_h)^r$ holds by the given assumptions, we obtain
697

698
$$\begin{aligned} \left| \frac{\partial \mathbf{h}_i^{(r)}}{\partial \mathbf{x}_s} \right| &\leq \sum_{j_1, \dots, j_{r-1}} \hat{A}_{ij_1} \hat{A}_{j_1 j_2} \dots \hat{A}_{j_{r-1} j_r} (\alpha_e \beta_h)^r \\ &= (\alpha_e \beta_h)^r \left(\hat{A}^r \right)_{is}. \end{aligned} \quad (21)$$

702 Second, to derive an upper bound of $\left| \frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s} \right|$. To this end, we plug Equation 16 into Equation 17 and
 703 obtain the following expression:
 704

$$705 \frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s} = \frac{\partial f_E}{\partial \mathbf{e}_{ij}^{(r-1)}} \frac{\partial \mathbf{e}_{ij}^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_E}{\partial \mathbf{h}_i^{(r-1)}} \frac{\partial \mathbf{h}_i^{(r-1)}}{\partial \mathbf{x}_s} + \frac{\partial f_E}{\partial \mathbf{h}_j^{(r-1)}} \left(\frac{\partial f_V}{\partial \mathbf{h}_j^{(r-2)}} \frac{\partial \mathbf{h}_j^{(r-2)}}{\partial \mathbf{x}_s} + \frac{\partial f_V}{\partial \mathbf{z}_j^{(r-1)}} \sum_k \hat{A}_{jk} \frac{\partial \mathbf{e}_{jk}^{(r-1)}}{\partial \mathbf{x}_s} \right). \quad (22)$$

706 As mentioned above, the Jacobians $\frac{\partial \mathbf{h}_i^{(r-1)}}{\partial \mathbf{x}_s}$, $\frac{\partial \mathbf{h}_j^{(r-2)}}{\partial \mathbf{x}_s}$ and $\frac{\partial \mathbf{e}_{ij}^{(r-1)}}{\partial \mathbf{x}_s}$ are zero matrices, and this enables
 707 us to recursively expand $\frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s}$ as follows:
 708

$$709 \frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s} = \frac{\partial f_E}{\partial \mathbf{h}_j^{(r-1)}} \frac{\partial f_V}{\partial \mathbf{z}_j^{(r-1)}} \sum_k \hat{A}_{jk} \frac{\partial \mathbf{e}_{jk}^{(r-1)}}{\partial \mathbf{x}_s} \quad 710$$

$$711 = \frac{\partial f_E}{\partial \mathbf{h}_j^{(r-1)}} \frac{\partial f_V}{\partial \mathbf{z}_j^{(r-1)}} \sum_k \hat{A}_{jk} \frac{\partial f_E}{\partial \mathbf{h}_k^{(r-2)}} \frac{\partial f_V}{\partial \mathbf{z}_k^{(r-2)}} \sum_m \hat{A}_{km} \frac{\partial \mathbf{e}_{km}^{(r-1)}}{\partial \mathbf{x}_s} \quad 712$$

$$713 = \dots = \sum_{j_2, \dots, j_r} \hat{A}_{jj_2} \hat{A}_{j_2 j_3} \dots \hat{A}_{j_{r-1} j_r} \cdot J_{jj_2 \dots j_{r-1}}(X) \cdot \frac{\partial \mathbf{e}_{j_{r-1} j_r}^{(1)}}{\partial \mathbf{x}_s} \quad 714$$

$$715 = \sum_{j_2, \dots, j_r} \hat{A}_{jj_2} \hat{A}_{j_2 j_3} \dots \hat{A}_{j_{r-1} j_r} \cdot J_{jj_2 \dots j_{r-1}}(X) \cdot \frac{\partial f_E}{\partial \mathbf{h}_{j_r}^{(0)}} \frac{\partial \mathbf{h}_{j_r}^{(0)}}{\partial \mathbf{x}_s} \quad 716$$

$$717 = \sum_{j_2, \dots, j_r} \hat{A}_{ij_2} \hat{A}_{j_2 j_3} \dots \hat{A}_{j_{r-1} j_r} \cdot J_{jj_2 \dots j_r}(X) \cdot \frac{\partial \mathbf{h}_{j_r}^{(0)}}{\partial \mathbf{x}_s}, \quad (23)$$

718 where $J_{jj_2 \dots j_r}(X)$ represents the product of $r-1$ second partial derivatives of f_V and r third partial
 719 derivatives of f_E with j_l indicating the index of i 's l -hop neighbors. Since $\partial_{\mathbf{x}_s} \mathbf{h}_{j_r}^{(0)} = \partial_{\mathbf{x}_s} \mathbf{x}_{j_r} = \delta_{j_r s}$
 720 holds, we obtain
 721

$$722 \frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s} = \sum_{j_2, \dots, j_{r-1}} \hat{A}_{jj_2} \hat{A}_{j_2 j_3} \dots \hat{A}_{j_{r-1} s} \cdot J_{jj_2 \dots j_{r-1} s}(X) \quad (24)$$

723 Finally, since $|J_{ij_2 \dots j_{r-1} s}(X)| \leq \alpha_e^{r-1} \beta_h^r$ holds by the given assumptions, we obtain
 724

$$725 \left| \frac{\partial \mathbf{e}_{ij}^{(r)}}{\partial \mathbf{x}_s} \right| \leq \sum_{j_2, \dots, j_{r-1}} \hat{A}_{jj_2} \hat{A}_{j_2 j_3} \dots \hat{A}_{j_{r-1} s} \alpha_e^{r-1} \beta_h^r \quad 726$$

$$727 = \alpha_e^{r-1} \beta_h^r \left(\hat{A}^{r-1} \right)_{js}. \quad (25)$$

728
 729
 730
 731
 732
 733
 734
 735
 736
 737
 738
 739
 740
 741
 742
 743
 744
 745
 746
 747
 748
 749
 750
 751
 752
 753
 754
 755

756 **B RELATED WORK**
757758 **B.1 HIGH-DIMENSIONAL PHYSICS MODELS**
759760 Deep learning-based modeling for high-dimensional physics problems has been actively used in
761 fluid dynamics (Bhatnagar et al., 2019; Zhang et al., 2018; Guo et al., 2016). Compared to complex
762 Finite Element Methods (FEM), deep learning-based approaches offer efficient execution times (Um
763 et al., 2018; Xie et al., 2018; Wiewel et al., 2019) and can be applied in real-world physical envi-
764 ronments where all parameters are not fully known (De Bézenac et al., 2018). Domain-specific loss
765 functions (Lee & You, 2019; Wang et al., 2020) or feature normalization that incorporates physical
766 knowledge (Thuerey et al., 2020) can help improve the performance of deep learning models.
767768 All the methods mentioned above use regular grid-based convolutions to model high-dimensional
769 physics problems. Holden et al. (2019) applied Principal Component Analysis (PCA) to cloth data
770 to reduce the dimensionality of high-dimensional systems and then performed simulations in the
771 reduced-dimensional space. Recent studies (Li et al., 2019; Sanchez-Gonzalez et al., 2020) have
772 utilized Graph Neural Networks (GNNs) to model physics systems such as fluid simulations. Con-
773 ventional FEM requires complex calculations and struggles to find accurate solutions when dealing
774 with nonlinear problems. In contrast, GNN-based methods can predict nonlinear problems more
775 quickly and accurately by learning these complex, nonlinear relationships directly from data (Luo
776 et al., 2018).
777778 **B.2 GRAPH REWIRING METHODS**
779780 Mesh refinement techniques (Löhner, 1995; Liu et al., 2022), which adaptively create high-
781 resolution meshes, can exacerbate the over-squashing problem. This leads to a loss of information as
782 long-range information is compressed into a fixed-size feature vector. To solve this problem, various
783 methods have been attempted to address over-squashing in GNNs (Fesser & Weber, 2023; Shi et al.,
784 2023; Finkelshtein et al., 2023; Barbero et al., 2024; Errica et al., 2023; Tortorella & Micheli, 2022).
785 To address this, various graph rewiring techniques have been proposed. Gasteiger et al. (2019) in-
786 troduced new edges based on diffusion distance to induce a smoother adjacency matrix. However,
787 this method is not suitable for tasks that require connecting long diffusion distances. Topping et al.
788 (2022) detects nodes with negative curvature and adds new edges from these nodes. (Karhadkar
789 et al., 2023) enhances the efficiency of information transfer by connecting edges that maximize
790 the spectral gap. Nguyen et al. (2023) propose connecting new edges based on the Ollivier-Ricci
791 curvature, which is designed to mitigate both over-smoothing and over-squashing simultaneously.
792 Attali et al. (2024) connect nodes based on Delaunay triangulation to make connections regular and
793 uniform, preventing information from being excessively concentrated on specific nodes. However,
794 since mesh-based simulations are already constructed with a regular grid-like structure similar to tri-
795 angulation, Delaunay triangulation offers little additional benefit to mesh graphs. All of these studies
796 employ a static approach, completing all rewiring before applying the GNN. Our method adaptively
797 rewires new edges during the message-passing process, considering the progressive propagation of
798 physical interactions.
799800 **C DATASETS**
801802 In this paper, we used the Cylinder Flow and Airfoil datasets, which are commonly used in fluid
803 simulations. Cylinder Flow represents a laminar flow model, where the fluid moves smoothly and
804 regularly, whereas Airfoil represents a turbulent flow model, where the fluid moves irregularly and
805 chaotically.
806807 **C.1 CYLINDER FLOW**
808809 The Cylinder Flow dataset contains physical quantities of a fluid as it flows around a cylinder. This
810 model has practical applications in various industrial fields, particularly in environments involv-
811 ing cylindrical pipes. The model can predict how fluid flow patterns change depending on the size
812 and position of the cylinder. This prediction ability can contribute to solving real-world engineer-
813 ing problems, such as designing cooling systems or improving fluid transportation efficiency. The
814 dataset includes 1,000 flow results, each with 600 time steps.
815

810
811

C.2 AIRFOIL

812
813
814
815
816
817
818
819

The Airfoil dataset includes physical quantities related to fluid flow around an aircraft wing. It contains complex turbulent phenomena, which helps our model learn to handle diverse flow conditions. An aircraft wing has a special cross-sectional shape called an airfoil. This shape causes air to flow over and under the wing at different speeds, and this velocity difference generates lift, which is the key force that allows an airplane to fly. This dataset is crucial for designing and validating the performance of wings in various aerospace applications, such as airplanes and helicopters. Specifically, the model can be used to predict how airflow changes around a wing and how this affects the stability of the aircraft. The Airfoil dataset also includes 1,000 flow simulations, each with 600 time steps.

820
821

D TRAINING TIME ANALYSIS

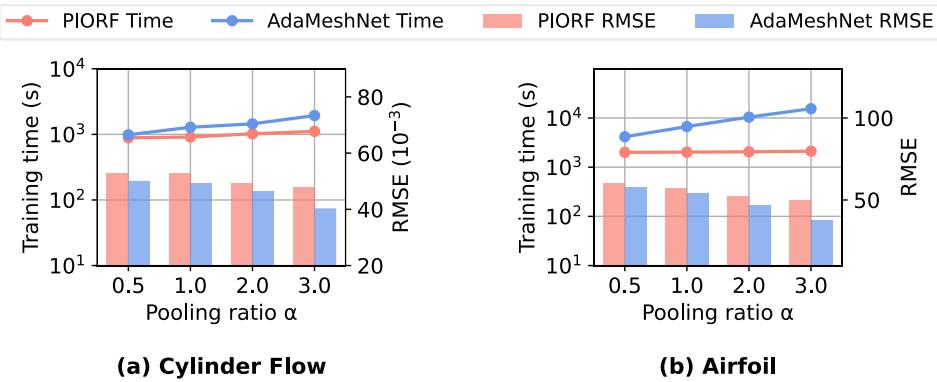
822
823
824
825
826
827
828
829
830
831
832
833834
835
836
837

Figure 7: Time efficiency on Cylinder Flow and Airfoil.

838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863

In this section, we measure the training time for mesh simulation to analyze the time efficiency. We compare the training time of our model with PIORF, the most time-efficient static rewiring method. Figure 7 shows the training time over various pooling ratios α on Cylinder Flow and Airfoil datasets. According to Figure 7, our AdaMeshNet model takes longer training time compared to the existing PIORF model, since it involves calculating the rewiring delay score during the message-passing process. Nevertheless, the result shows that as the pooling ratio α decreases, the training time of AdaMeshNet becomes comparable to that of PIORF. While AdaMeshNet is somewhat less efficient in terms of training time compared to PIORF, the bar graphs in Figure 7 show that it provides a significant advantage in terms of improved prediction accuracy. In real-world fluid dynamics simulations, even a small difference in accuracy can have a substantial impact on the overall reliability of the model, which makes a slight increase in training time acceptable. For instance, the Airfoil dataset can be used to design and validate wing performance. In the aerospace field, the performance of the wing is closely related to safety, making improvements in accuracy much more important than training time efficiency. Therefore, even with a slight increase in training time, our model, which significantly contributes to improving accuracy, is expected to have high applicability to real-world problems in fluid dynamics. In conclusion, while AdaMeshNet takes longer to train compared to PIORF, the extra time is spent on modeling the gradual propagation we propose, which can be seen as a reasonable cost to mimic more realistic models in complex fluid simulations.

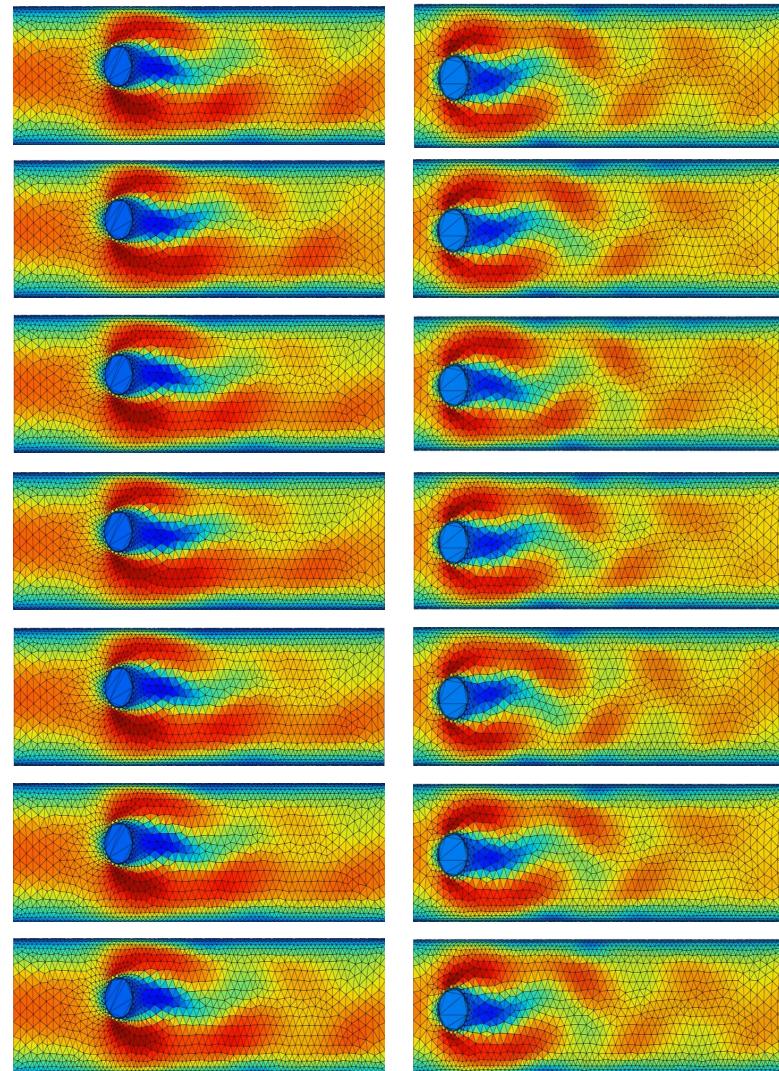
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
E OTHER VELCITY CONTOURS

Figure 8: Other Velocity Contours

918 F ALGORITHM
919920 **Algorithm 1:** Adaptive Graph Rewiring for Mesh-Based GNN Training
921

922 **Input :** Training mesh \mathcal{M}_t
 923 **Output:** Updated mesh \mathcal{M}_{t+1}

1 **for** $epoch = 1, 2, \dots, T$ **do**

2 **Preprocessing:** **for** node v_i in \mathcal{M}_t **do**
 | Calculate node curvature γ_i using Eq. 5
 | **end**
 5 Identify bottleneck nodes \mathcal{V}_{lowORC} using Eq. 6
 6 **for** each $v_i \in \mathcal{V}_{lowORC}$ **do**
 | 7 Select optimal connection node v_{i*} using Eq. 7
 | 8 Calculate rewiring delay score $s_{delay}(i, i^*)$ using Eq. 8
 | **end**
 10 **Encoder:** **for** each node v_i and edge e_{ij} in \mathcal{M}_t **do**
 | 11 Calculate node embedding \mathbf{h}_i using Eq. 9
 | 12 Calculate edge embedding \mathbf{e}_{ij} using Eq. 9
 | **end**
 13 **Processor:** **for** layer $l = 0, 1, \dots, L-1$ **do**
 | 14 **for** each $v_i \in \mathcal{V}_{lowORC}$ **do**
 | 15 **for** each optimal connection node v_{i*} **do**
 | | 17 If $l < s_{delay}(i, i^*) \leq l + 1$ Add v_{i*} to neighbor set \mathcal{N}_i^{l+1}
 | | **end**
 | | **end**
 | | **for** each node v_i in \mathcal{M}_t **do**
 | | 19 Initialize neighbor set \mathcal{N}_i^0 as direct neighbors from \mathcal{E}
 | | 20 **for** each node $v_j \in \mathcal{N}_i^{l+1}$ **do**
 | | | 22 Update edge embedding \mathbf{e}_{ij}^{l+1} using Eq. 11
 | | | 23 Update node embedding \mathbf{h}_i^{l+1} using Eq. 12
 | | | **end**
 | | | **end**
 | | | **end**
 | | | **end**
 | | | **Decoder and State Updater:** **for** each node v_i in \mathcal{V} **do**
 | | | 28 Compute the predicted state q_i^{t+1} using Eq. 13
 | | | **end**
 | | | **Update Mesh:** Update mesh \mathcal{M}_{t+1} based on the updated nodes \mathcal{V} and their corresponding states
 | | | **end**
 32 **end**

951
 952
 953
 954
 955
 956
 957
 958
 959
 960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 970
 971

972 **G NOTATIONS**
973974 In this section, we summarize the main notations used in this paper. Table 3 provides the main
975 notation and their descriptions.
976977 Table 3: Summary of the main notations used in this paper.
978

979 Notation	980 Description
$980 \mathcal{G} = (\mathcal{V}, \mathcal{E})$	A graph with a set of nodes \mathcal{V} and a set of edges \mathcal{E}
$981 n = \mathcal{V} $	Total number of nodes
$982 \mathcal{N}_i$	Set of neighbors for node i
$983 \mathbf{x}_i \in \mathbb{R}^{p_0}$	Initial feature vector of node i
$984 \mathbf{v}_i$	Velocity vector of node i
$985 d_{\mathcal{G}}(i, j)$	Shortest path distance between nodes i and j in graph \mathcal{G}
$986 l$	Layer index of the GNN
$987 L$	Total number of message-passing blocks (layers)
$988 \mathbf{h}_i^{(l)}$	Hidden representation (embedding) of node i at layer l
$989 \mathbf{e}_{ij}^{(l)}$	Hidden representation (embedding) of edge (i, j) at layer l
$990 f_V$	Node update function (MLP)
$991 f_E$	Edge update function (MLP)
$992 r$	Distance between two nodes in hops
$993 B_r(i)$	Set of nodes within r hops from node i (receptive field)
$994 \partial \mathbf{h}_i^{(r)} / \partial \mathbf{x}_s$	Jacobian of the hidden representation of node i at layer r w.r.t. the input feature of node s
$995 \hat{A}$	Normalized augmented adjacency matrix
$996 \alpha_e$	Upper bounds for the second partial derivatives of f_V
$997 \beta_h$	Upper bounds for the third partial derivatives f_E
$998 \kappa(i, j)$	Ollivier-Ricci Curvature of the edge (i, j)
$999 \gamma_i$	Average curvature of node i (local geometric information)
$1000 \mathcal{V}_{\text{lowORC}}$	Set of bottleneck nodes in the bottom $a\%$ of curvature
$1001 v_i^*$	Optimal node to be rewired with the bottleneck node v_i
$1002 s_{\text{delay}}(i, i^*)$	Rewiring delay score for the edge (i, i^*)
1003β	Hyperparameter used in calculating the delay score
$1004 \mathcal{N}_i^l$	Set of neighbors for node i at layer l (with rewiring applied)
$1005 \hat{\mathbf{v}}_i$	Predicted velocity gradient of node i from the decoder
$1006 \hat{\mathbf{v}}_i^{t+1}$	Predicted velocity of node i at time $t + 1$

1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025