A PROOF OF LEMMA 4.1

Lemma 4.1. *Suppose we have an FFN Φ with all absolute values of weights and biases bounded by satisfy M, B respectively. Besides ReLU is adopted as the activation function. Then, our Eqn. 7 has a unique absolutely continuous solution.*

To begin with, we introduce a theorem from (Difonzo et al., 2024).

Theorem A.1. *Let $n \in \mathbb{N} \cup \{0\}, \tau \in (0, +\infty), x_0 \in \mathbb{R}^d$ and let f satisfy the following assumptions:*

(A1) For all $t \in [0, (n+1)\tau]$, $f(t, \cdot, \cdot) \in C\left(\mathbb{R}^d \times \mathbb{R}^d; \mathbb{R}^d\right)$,

(A2) For all $(x, z) \in \mathbb{R}^d \times \mathbb{R}^d$, $f(\cdot, x, z)$ is Borel measurable,

(A3) There exists $K : [0, (n+1)\tau] \to [0, +\infty)$ such that $K \in L^1([0, (n+1)\tau])$ and for all $(t, x, z) \in [0, (n+1)\tau] \times \mathbb{R}^d \times \mathbb{R}^d$, we have:

$$\|f(t, x, z)\| \leq K(t)(1 + \|x\|)(1 + \|z\|), \tag{14}$$

(A4) For every compact set $U \subset \mathbb{R}^d$, there exists $L_U : [0, (n+1)\tau] \mapsto [0, +\infty)$ such that $L_U \in L^1([0, (n+1)\tau])$ and for all $t \in [0, (n+1)\tau], x_1, x_2 \in U, z \in \mathbb{R}^d$, we have:

$$\|f(t, x_1, z) - f(t, x_2, z)\| \leq L_U(t)(1 + \|z\|)\|x_1 - x_2\| \tag{15}$$

Then there exists a unique absolutely continuous solution $x = x(x_0, f)$ to the following system:

$$\begin{cases} x'(t) = f(t, x(t), x(t-\tau)), & t \in [0, (n+1)\tau] \\ x(t) = x_0, & t \in [-\tau, 0) \end{cases}, \tag{16}$$

such that for $j = 0, 1, \ldots, n$ we have:

$$\sup_{0 \leq t \leq \tau} \|\phi_j(t)\| \leq K_j, \tag{17}$$

where $\phi_{-1}(t) = x_0, \phi_j(t) = x(t + j\tau), \quad$ for $j = 0, 1, \ldots, n$, $K_{-1} := \|x_0\|$ and

$$K_j = (1 + K_{j-1})\left(1 + \|K\|_{L^1([j\tau,(j+1)\tau])}\right) \cdot \exp\left((1 + K_{j-1})\|K\|_{L^1([j\tau,(j+1)\tau])}\right). \tag{18}$$

Proof. Firstly, the function Φ is learnable FFNs, and the first layer can be represented as:

$$\boldsymbol{m}_1^t = \sigma(\boldsymbol{W}_0 \boldsymbol{y}^t + \boldsymbol{W}_1 \boldsymbol{y}^{t-1} + \boldsymbol{b}_1), \tag{19}$$

where $\boldsymbol{W}_0$ and $\boldsymbol{W}_1$ are two weight matrices, and $\boldsymbol{b}_1$ denotes the biases. The i layers can be written as:

$$\boldsymbol{m}_i^t = \sigma(\boldsymbol{W}_i \boldsymbol{m}_{i-1}^t + \boldsymbol{b}_i), \tag{20}$$

where $\boldsymbol{W}_i$ and $\boldsymbol{b}_i$ are corresponding weights and biases. These functions are continuous and Borel measurable. Therefore, Φ also satisfies assumptions (A1), (A2).

Secondly, given any inputs $\boldsymbol{y}^t, \boldsymbol{y}^{t-1}$, we can see:

$$\begin{aligned} \|\Phi(\boldsymbol{y}^t, \boldsymbol{y}^{t-1})\| &= \|\boldsymbol{m}_l^t(\cdots \boldsymbol{m}_1^t(\boldsymbol{y}^t, \boldsymbol{y}^{t-1}))\| \\ &\leq \|\boldsymbol{m}_l^t(\cdots \boldsymbol{m}_2^t(M\|\boldsymbol{y}^t\| + M\|\boldsymbol{y}^{t-1}\| + B))\| \\ &\leq \cdots \leq M^l\|\boldsymbol{y}^t\| + M^l\|\boldsymbol{y}^{t-1}\| + \frac{M^l - 1}{M - 1}B \\ &\leq L(1 + \|\boldsymbol{y}^t\|)(1 + \|\boldsymbol{y}^{t-1}\|), \end{aligned}$$

where $L = \max\{M^l, \frac{M^l - 1}{M-1}B\}$. Thus assumption (A3) is satisfied.

Third, note the ReLU activation function satisfies:

$$|\sigma(x) - \sigma(y)| \leq |x - y|. \tag{21}$$

Given any $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{z}$, we have:

$$\|\boldsymbol{m}_1^t(\boldsymbol{x}_1, \boldsymbol{z}) - \boldsymbol{m}_1^t(\boldsymbol{x}_2, \boldsymbol{z})\| \leq \|\boldsymbol{W}_0(\boldsymbol{x}_1 - \boldsymbol{x}_2)\| \leq M\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|. \tag{22}$$

Then, the following inequation holds:

$$\|\boldsymbol{m}_2^t(\boldsymbol{m}_1^t(\boldsymbol{x}_1,\boldsymbol{z}))-\boldsymbol{m}_2^t(\boldsymbol{m}_1^t(\boldsymbol{x}_2,\boldsymbol{z}))\| \leq \|W_2(\boldsymbol{m}_1^t(\boldsymbol{x}_1,\boldsymbol{z})-\boldsymbol{m}_1^t(\boldsymbol{x}_2,\boldsymbol{z}))\| \leq M^2\|\boldsymbol{x}_1-\boldsymbol{x}_2\| \quad (23)$$

Thus, we can conclude:

$$\|\Phi(\boldsymbol{x}_1,\boldsymbol{z})-\Phi(\boldsymbol{x}_2,\boldsymbol{z})\| \leq M^l\|\boldsymbol{x}_1-\boldsymbol{x}_2\|, \quad (24)$$

which means the assumption (A4) is satisfied.

Then, based on Theorem A.1, we can claim that our graph ODE system Eqn. 7 has a unique absolutely continuous solution. □

B ALGORITHM

The training algorithm of our FAIR is summarized in Algorithm 1.

Algorithm 1 Learning Algorithm of FAIR

Input: The mesh graph G, a sequence of observations $G^{t_0:t_0+T}=\{G^{t_0},\cdots,G^{t_0+T}\}$.
Output: Parameters in our FAIR.

Initialize model parameters;
// Foresight Step
while not convergence **do**
  for each training sequence **do**
    Feed each sample into the graph ODE;
    Generate the predictions at the given timestamps using Eqn. 8;
    Minimize the mean square error for these timestamps in Eqn. 9;
    Update the parameters using gradient descent;
  end for
end while
// Refinement Step
while not convergence **do**
  for each training sequence **do**
    Generate the predictions at the given timestamps using Eqn. 8;
    Generate the refined predictions from Eqn. 12;
    Minimize the mean square error for the target in Eqn. 13
    Update the parameters using gradient descent;
  end for
end while

The inference algorithm of our FAIR is summarized in Algorithm 2.

Algorithm 2 Inference Algorithm of FAIR

Input: The mesh graph G, a sequence of observations $G^{t_0:t_0+T}=\{G^{t_0}\}$.
Output: Parameters in our FAIR.

$t=t_0$
$\hat{\boldsymbol{X}}^{t_0,ref}=\boldsymbol{X}^{t_0}$
while $t<t_0+T$ **do**
  // Foresight Step
  Feed $\hat{\boldsymbol{X}}^{t,ref}$ into the graph ODE;
  Generate the predictions $\hat{\boldsymbol{X}}^{t+1},\hat{\boldsymbol{X}}^{t+1+r},\cdots,\hat{\boldsymbol{X}}^{t+1+rL-r}$ using Eqn. 8;
  // Refinement Step
  Generate the refined predictions $\hat{\boldsymbol{X}}^{t+1,ref}$ from Eqn. 12;
  $t\leftarrow t+1$
end while

C Dataset Details

Four physics simulation benchmark datasets are utilized to evaluate our proposed FAIR and the compared baselines with details listed in Table 3.

CylinderFlow simulates the incompressible Navier-Stokes flow of water around a cylinder on a fixed 2D Eulerian mesh generated by COMSOL (Multiphysics, 1998). This mesh has an irregular structure with varying edge lengths in different regions. The simulation consists of 600 time steps, with an interval of 0.01s between each step. Node attributes in the system include mesh position, node type, velocity, and pressure. Node types can be divided into three different categories in fluid domains, *i.e.*, fluid nodes, wall nodes, and inflow/outflow boundary nodes. We predict the velocity values in both directions.

Airfoil simulates the aerodynamics around the cross-section of an airfoil wing for compressible Navier-Stokes flow by SU2 (Economon et al., 2016). As the edge lengths of the mesh range between 2×10^{-4}m to 3.5m, the mesh structure is highly irregular. Each trajectory containing $5,200$ nodes has 500 time steps with an interval of 0.008s. Node attributes include mesh position, node type, velocity, pressure, and density. We aim to predict the velocity, density, and pressure in the future.

DeformingPlate is a hyper-elastic plate in the structural mechanical system, deformed by a kinematic actuator, simulated with a quasi-static simulator COMSOL. Each trajectory has 400 time steps with $1,200$ nodes average. A one-hot vector for each type of node distinguishes actuators from plates in the Lagrangian tetrahedral mesh. In addition, node type, position, and velocity are fed to predict the whole trajectories.

InflatingFont is from BSMS-GNN (Cao et al., 2023), including $1,400$ 2×2-character matrices in Chinese. *InflatingFont* has more complex geometric shapes, 2 to 8 times the number of nodes, and 70 times the number of contact edges. We aim to predict the future position of every mesh node.

For each dataset, the train/val/test splits follow the recent work (Cao et al., 2023).

Table 3: All of our four datasets are listed in detail. The system describes the underlying PDE: hypere-lastic flow, or a compressible or incompressible Navier-Stokes flow. Simulation data is generated using a solver. In *DeformingPlate* and *InflatingFont*, there is no time step since it is a quasi-static simulation.

Dataset	Nodes (avg)	Edge (avg)	Type	Steps
CylinderFlow	1885	5424	Eulerian	600
Airfoil	5233	15449	Eulerian	500
DeformingPlate	1271	4611	Lagrangian	400
InflatingFont	13177	39481	Lagrangian	100

D Baseline Details

We compare our FAIR with a range of state-of-the-art methods, *i.e.*, GraphUNets (Gao & Ji, 2019), GNS (Sanchez-Gonzalez et al., 2020), MeshGraphNet (Pfaff et al., 2021), MS-GNN-GRID (Lino et al., 2021), Social-ODE (Wen et al., 2022) and BSMS-GNN (Cao et al., 2023). Their details are elaborated as follows:

- GraphUNets (Gao & Ji, 2019) proposes new pooling and unpooling operations, which can be implemented in an UNet-style architecture (Ronneberger et al., 2015). We have replaced the original GCN layers into our message passing layers following (Cao et al., 2023).
- GNS (Sanchez-Gonzalez et al., 2020) is the pioneering work on physical simulations, which leverage graphs to depict systems and model dynamics using message passing neural networks. This work demonstrates that graph neural networks have the ability to capture long-range interactions. We employ 15 message passing layers as in Cao et al. (2023).
- MeshGraphNet (Pfaff et al., 2021) is an effective framework for mesh-based physical simulations, which combine graph neural networks and re-mesh techniques to learn the dynamics for next-time

predictions. Based on the basic node modeling of graph networks, MeshGraphNet introduces additional edge encoders. The edge representation is updated during each MeshGraphNet layer. We also employ 15 message passing layers as in Cao et al. (2023).

- MS-GNN-GRID (Lino et al., 2021) is a representative work for those building the hierarchy with spatial proximity, which introduces a novel multi-scale graph neural network model, designed to enhance and accelerate predictions in continuum mechanics simulations. Following (Lino et al., 2021; Cao et al., 2023), MS-GNN-GRID is implemented using the finest edge encoder, an additional aggregation module for node and edge representations, and a node returning module.
- Social-ODE (Wen et al., 2022) is a latent ordinary differential equation generative model, which can understand and predict dynamics from irregularly-sampled partial observations with underlying graph structures. In our implementation, we use a similar structure paradigm with our approach, *i.e.*, Encoder-ODE-Decoder pattern, where both the encoder and decoder consist of 7 layer networks for node and edge modeling.
- BSMS-GNN (Cao et al., 2023) is a framework that introduces a bi-stride pooling strategy for large-scale physical simulations, addressing existing challenges associated with scaling complexity, over-smoothing, and incorrect edge introductions. BSMS-GNN follows the design paradigm of UNet (Ronneberger et al., 2015). We adhere to the original network configurations and utilize several UNet layers for experiments.

E Implementation Details

In this paper, we present an extensive series of experiments leveraging the frameworks of PyTorch (Paszke et al., 2017), PyG (Fey & Lenssen, 2019), and TorchDiffEq (Kidger et al., 2021). To ensure fairness, we implement our approach with a publicly available codebase by BSMS-GNN (Cao et al., 2023). We execute all experiments on a single A100 GPU, including the speed test. To maximize training efficiency, we set the batch size across all experiments at the maximum level supported by the GPU memory. Our optimization strategy includes the use of Adam optimizer, with a learning rate set at $1e-4$, with an exponential learning rate decay strategy.

Following BSMS-GNN, we apply Gaussian noise to each original trajectory at the start of every epoch, aiming to enhance the adaptability of the model to process noisy inputs. Furthermore, to maintain fairness, we set the layer number for both encoder and decoder to 7, while MeshGraphNet and BSMS-GNN both have 15 layers. Each layer includes the nodal encoder, processor, and nodal decoder, all activated by ReLU and embedded within two hidden-layer MLPs. The MLPs have a residual connection, while a LayerNorm normalizes all MLP outputs except for the nodal decoder. Our code is available at `https://anonymous.4open.science/r/FAIR` anonymously. We will make the code public after the anonymity period.

F More Results

F.1 Predictions at Different Time Steps

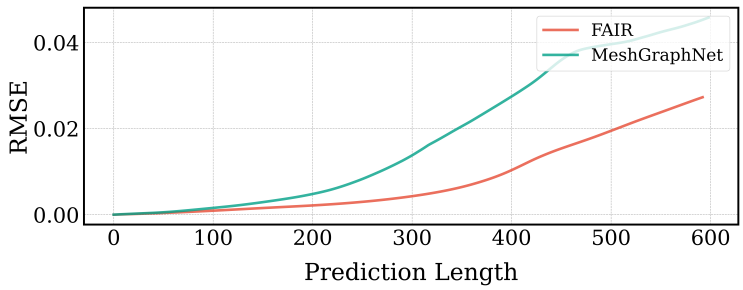

Figure 6: RMSE of our FAIR and MeshGraphNet with respect to different prediction lengths on *CylinderFlow*.

Figure 6 records the prediction errors of our proposed FAIR and MeshGraphNet with different time steps in terms of RMSE on *CylinderFlow*. From the results, we can observe that our proposed

FAIR demonstrates stronger modeling capabilities with relatively lower errors at large-time steps. In contrast, MeshGraphNet suffers from serious error accumulations, with the prediction error about twice as large as ours at the final time step.

F.2 VISUALIZATION

Figure 7 shows more visualization at a range of time steps on *CylinderFlow*. Here we utilize a different instance to show the results at the time steps among $\{1, 10, 50, 100, 300, 400\}$. From the results, we can validate the superiority of the proposed FAIR again.

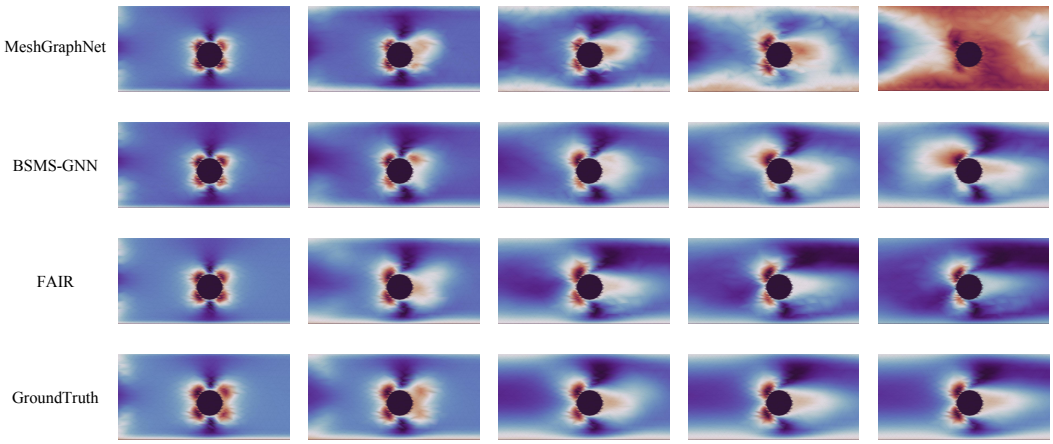

Figure 7: Visualization of *CylinderFlow* at multiple time steps.

F.3 INFLUENCE OF DIFFERENT STEP SIZES

Figure 8 shows the results with respect to different step sizes r on *Airfoil*. From the results, it can be found that the prediction errors first decrease and then increase, which achieves the minimum when r is around 2. This observation is consistent with that in Sec. 5.3.

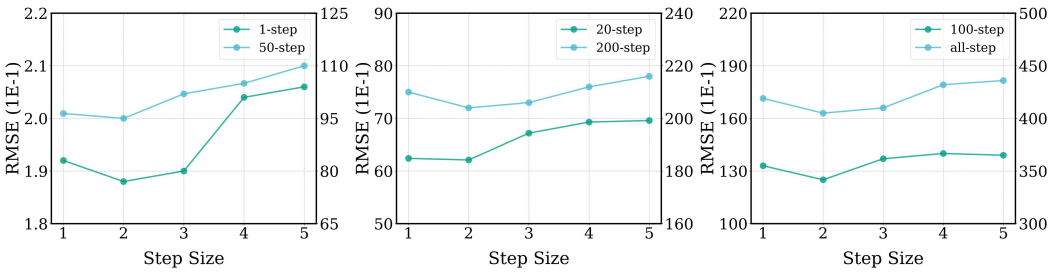

Figure 8: The performance with respect to different step sizes r at different time steps on *Airfoil*.

F.4 INFLUENCE OF FUTURE PREDICTION NUMBERS

Figure 9 records the results with respect to different future prediction numbers L on *Airfoil*. From the results, we can observe that prediction errors would decrease generally till saturation and the errors would not dramatically decrease when L is over 3 in most cases. Due to the trade-off between effectiveness and efficiency, we set L to 3 as default.

F.5 VISUALIZATION OF ERRORS

Figure 11 visualizes the prediction errors of different approaches compared with the ground truth on *CylinderFlow*, respectively. From top to bottom, we show the results at the time steps 1, 10, 50,

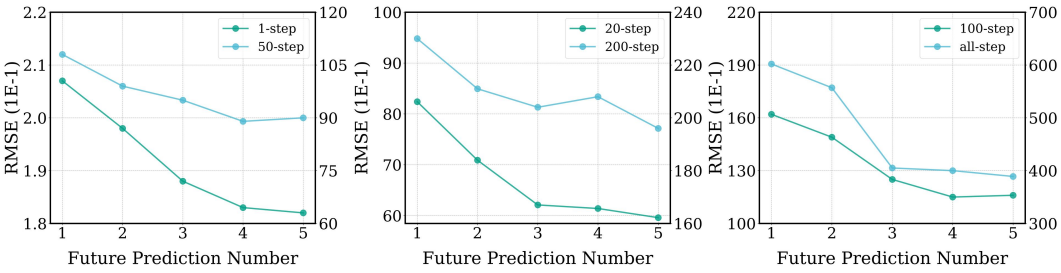

Figure 9: The performance with respect to different prediction lengths L at different time steps on *Airfoil*.

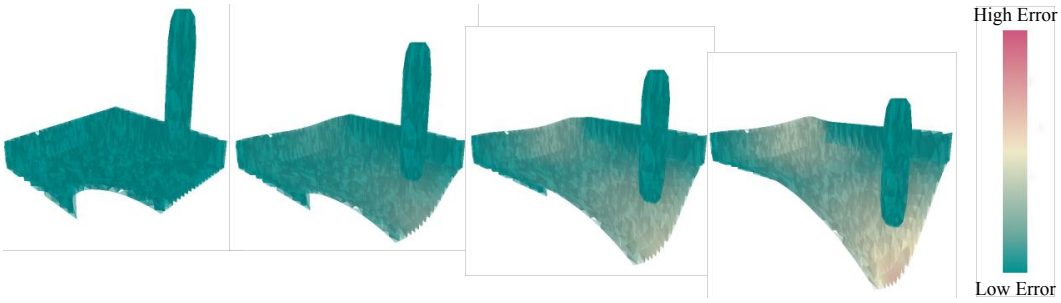

Figure 10: Visualization of FAIR on 3D dataset *DeformingPlate*, with the time steps among $1, 100, 200$ and 300.

100, 300 and 400. We can observe that although both baselines and our FAIR can make accurate short-term predictions with limited prediction errors, these baselines would suffer from serious error accumulation in long-term forecasting. Figure 12 visualizes the prediction errors on *Airfoil* and we can find that our FAIR is consistently superior to the compared baselines. Moreover, Figure F.4 shows the difference on 3D dataset *DeformationPlate*.

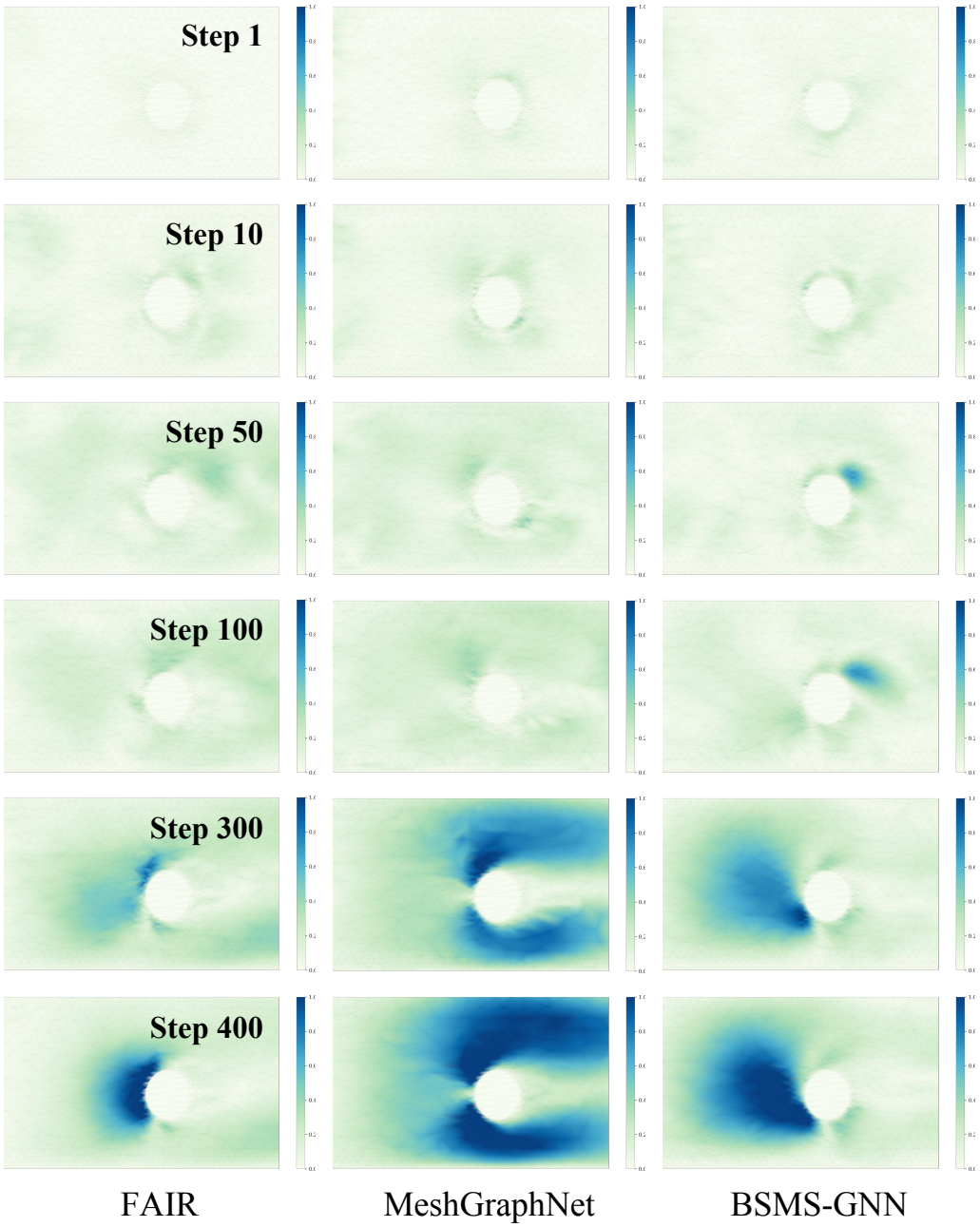

Figure 11: Visualization of error on *CylinderFlow* with multiple time steps.

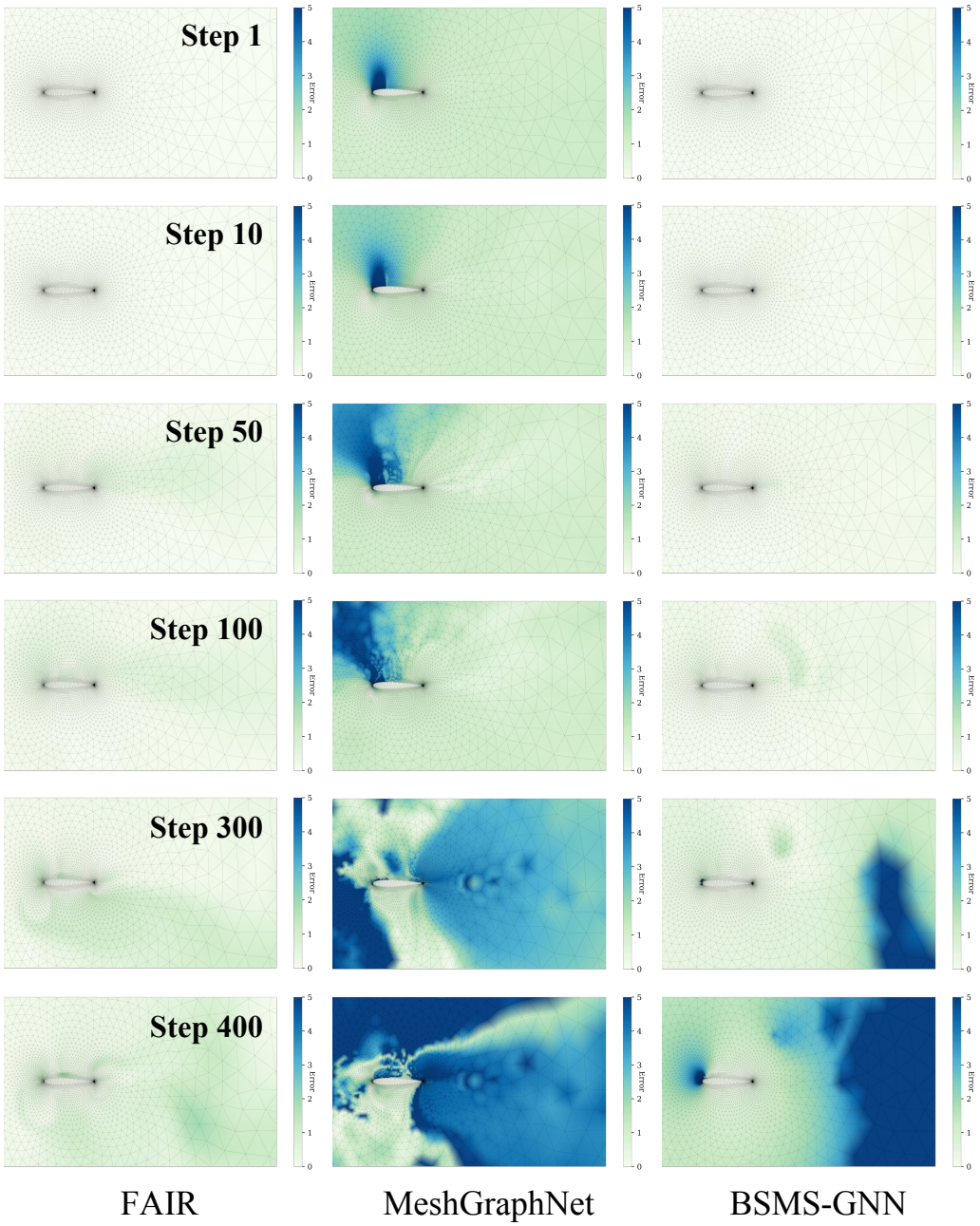

Figure 12: Visualization of error on *Airfoil* at multiple time steps.