Future Matters for Present: Towards Effective Physical Simulation over Meshes

Xiao Luo
xiaoluo@cs.ucla.edu
University of California, Los Angeles
Los Angeles, CA, USA

Junyu Luo
luojunyu@stu.pku.edu.cn
Peking University
Beijing, China

Huiyu Jiang
huiyujiang@ucsb.edu
University of California, Santa Barbara
Los Angeles, CA, USA

Hang Zhou
hgzhou@ucdavis.edu
University of California, Davis
Los Angeles, CA, USA

Zhiping Xiao
patxiao@uw.edu
University of Washington
Seattle, WA, USA

Wei Ju
juwei@pku.edu.cn
Peking University
Beijing, China

Carl Yang
j.carlyang@emory.edu
Emory University
Atlanta, GA, USA

Ming Zhang
mzhang_cs@pku.edu.cn
Peking University
Beijing, China

Yizhou Sun
yzsun@cs.ucla.edu
University of California, Los Angeles
Los Angeles, CA

ABSTRACT

This paper investigates the problem of learning mesh-based physical simulations, which is a crucial task with applications in fluid mechanics and aerodynamics. Recent works typically utilize graph neural networks (GNNs) to produce next-time states on irregular meshes by modeling interacting dynamics, and then adopt iterative rollouts for the whole trajectories. However, these methods cannot achieve satisfactory performance in long-term predictions due to the failure of capturing long-term dependency and potential error accumulations. To tackle this, we introduce a new future-to-present learning perspective, and further develop a simple yet effective approach named Foresight And InteRpolation (FAIR) for long-term mesh-based simulations. The main idea of our FAIR is to first learn a graph ODE model for coarse long-term predictions and then refine short-term predictions via interpolation. Specifically, FAIR employs a continuous graph ODE model that incorporates past states into the evolution of interacting node representations, which is capable of learning coarse long-term trajectories under a multi-task learning framework. Then, we leverage a channel aggregation strategy to summarize the trajectories for refined short-term predictions, which can be illustrated using an interpolation process. Through pyramid-like alternative propagation between the foresight step and refinement step, our proposed framework FAIR can generate accurate long-term trajectories, achieving a significant error reduction compared with the best baseline on four benchmark datasets. Extensive ablation studies and visualization further validate the superiority of our proposed FAIR.

KDD '25, August 3–7, 2025, Toronto, ON, Canada

ACM ISBN 979-8-4007-1245-6/25/08
https://doi.org/10.1145/3690624.3709340

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Information systems** → **Physical data models**.

KEYWORDS

Physical simulation, dynamical system modeling, graph learning, neural ODE

ACM Reference Format:
Xiao Luo, Junyu Luo, Huiyu Jiang, Hang Zhou, Zhiping Xiao, Wei Ju, Carl Yang, Ming Zhang, and Yizhou Sun. 2025. Future Matters for Present: Towards Effective Physical Simulation over Meshes. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3690624.3709340

1 INTRODUCTION

Physical simulations are of paramount importance for understanding fundamental principles in various domains [77, 90], including mechanics [45, 62, 75], electromagnetics [54], biology [74] and acoustics [48]. The majority of studies [58] utilize mesh-based finite element systems to describe complicated physics by simulating the interactions of mesh points. To achieve the optimal use of resource budgets for unstructured surfaces, they usually allocate greater resolution to regions of interest where more accurate analysis is expected, resulting in complicated irregular mesh structures [11, 20, 37, 40]. Traditional numerical solvers usually require a heavy computational burden, and thus efficient data-driven simulators have drawn ever-lasting interest recently.

With the rapid development of deep learning techniques, several data-driven simulators have been recently proposed to learn numerical simulations on structured grids [15, 30, 61, 71]. These approaches usually leverage convolution neural networks (CNNs) to extract spatial semantics for future predictions. However, they have strict requirements of data structures. To adapt to irregular meshes with high efficiency, graph machine learning-based approaches have received more attention gradually [4, 58, 65, 67, 82].

The majority of them first construct a geometric graph where mesh points are considered as nodes and utilize graph neural networks (GNNs) to model the interacting dynamics in physical systems. In particular, they first map observations of nodes into the embedding space and then follow the paradigm of message passing [1, 31, 78, 83, 85, 89, 93], which aggregates edge information from the neighbors of each node to update the node representation in a progressive fashion. Finally, a decoder is adopted to output the prediction of trajectories at the next time step.

In reality, long-term forecasting [34, 35, 52, 79, 94, 95] is a practical yet challenging scenario for physical simulations. Existing methods [4, 58, 67] usually rely on an autoregressive strategy, which utilizes the current states for the next-time predictions and then feeds them back as input in an iterative manner. However, these one-step predictors often struggle to capture the long-term system dynamics, which could be governed by underlying partial differential equations (PDEs) [72]. Additionally, they are prone to accumulating errors over iterative rollouts, which degrades the performance of their long-term predictions. Given these accumulated errors, it is highly anticipated to include future states into the prediction procedure to provide a foresight of systems, which not only enhances long-term predictions, but also infers extra knowledge connected with underlying PDEs to refine the short-term predictions.

Towards this end, we provide a new perspective that learns from the future for present predictions (see Figure 1), and propose a simple yet effective approach named Foresight And InteRpolation (FAIR) for long-term mesh-based simulations. In particular, FAIR takes a two-stage learning paradigm which first generates coarse long-term predictions using a continuous graph ordinary differential equation (ODE) model, and then refines these into accurate short-term predictions through interpolation. In the first stage, we extend neural ODEs [5, 53] into graph ODE twins to provide coarse foresight, which consists of two coupled graph ODEs to model the evolution of both mesh node and edge representations using the neighborhood aggregation mechanism. To enhance the capacity to capture non-linear complex patterns, we incorporate historical states to augment the current embedding in ODEs. Our graph ODE model has the flexibility to generate various predictions of different steps ahead, which is optimized for a multi-task learning framework [50]. In summary, our model can not only accord with the continuous nature of real-world systems, but also capture long-term dependencies with limited error accumulation. In the second stage, we employ a channel aggregation strategy to summarize the future predictions and current observations for interpolation, which is followed by further neighborhood aggregation in the observation space to refine the short-term predictions. More importantly, through the alternative foresight step and refinement step, our FAIR can provide refined long-term trajectories.

We validate the effectiveness of FAIR via extensive experiments on four benchmark datasets. From the experimental results, our FAIR achieves significant improvements over various baseline models. In particular, our proposed FAIR achieves a significant error reduction compared with the best baseline. We also provide extensive ablation studies, sensitivity analysis and visualization to validate the effectiveness of our proposed FAIR. We also provide extensive efficiency analysis and results on noisy data.

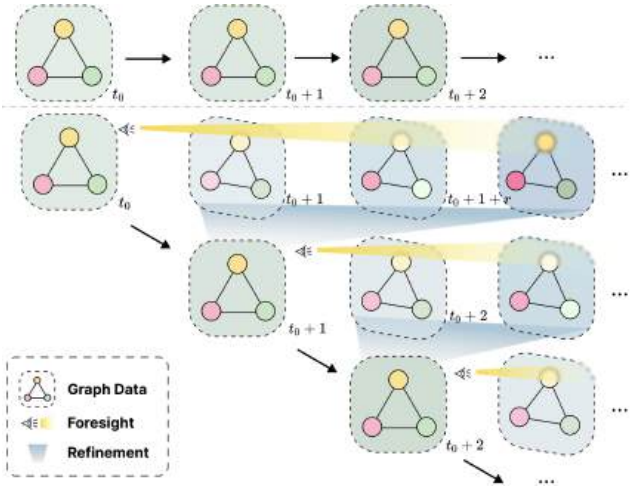

Figure 1: Previous approaches adopt the current states for next-time predictions while our FAIR includes alternative foresight and refinement.

2 PRELIMINARIES

2.1 Problem Formulation

In this work, we aim to learn a neural simulator that uses neural operations to approximate the ground-truth physics dynamics on irregular meshes, which are usually driven by underlying PDEs such as Navier–Stokes equations [58]. The dynamic system with both spatial correlations can be characterized using a mesh graph $G = (\mathcal{V}, \mathcal{E})$ with a set of mesh nodes $\mathcal{V}$ and an edge set $\mathcal{E}$. Given current states of mesh points, *i.e.*, $\boldsymbol{X}^{t_0} \in \mathbb{R}^{N\times F}$ and (optionally) a series of historical states, *i.e.*, $\{\boldsymbol{X}^{t_0-1}, \cdots, \boldsymbol{X}^{t_0-\tau}\}$ with the trajectory length τ, the objective is to predict the future trajectories for all nodes $\boldsymbol{X}^t \in \mathbb{R}^{N\times F}$ $(t_0 < t \leq t_0 + T)$, where F is the attribute dimension and N is the number of mesh nodes. We use prediction errors with respect to the ground truth to evaluate the performance.

2.2 Graph Neural Networks (GNNs)

GNNs are a class of neural networks that operate directly on graph-structured data [25, 76, 84]. They have been extensively studied for approximating pairwise interactions in mesh-based physical systems [58, 65, 67, 80]. GNNs usually follow the paradigm of message passing [31, 73, 83], where neighborhood information is aggregated to update the node representations in an iterative manner. Given the node embedding of $v \in \mathcal{V}$ at the layer l, the updated rule can be written as follows:

$$\boldsymbol{v}_i^{(l+1)} = \mathrm{COM}\{\boldsymbol{v}_i^{(l)}, \mathrm{AGG}(\{\boldsymbol{v}_j^{(l)} | j \in \mathcal{N}(i)\})\}, \tag{1}$$

where $\mathcal{N}(i)$ denotes the neighbours of node i. $\mathrm{COM}(\cdot)$ and $\mathrm{AGG}(\cdot)$ denotes the combination and aggregation operators, respectively. A decoder is used to map Through this, GNNs can capture the complex interaction among mesh points, which reveals how the system changes from time step t_0 to time step $t_0 + 1$ [4, 42, 87]. To generate long-term predictions, these approaches usually feed the predictions back as the input in the rollout procedure.

2.3 Neural Ordinary Differential Equations (ODEs) for Dynamical System Modeling

A complex physical system can be described by a series of coupled nonlinear ordinary differential equations [21, 67]:

$$\frac{d\boldsymbol{h}_i^t}{dt} = \Phi(\boldsymbol{h}_1^t, \boldsymbol{h}_2^t, \cdots, \boldsymbol{h}_N^t), \tag{2}$$

where $\boldsymbol{h}_i^t$ is the state for object i at time step t and $\Phi(\cdot)$ is a function for capturing the interaction among objects, which can be a neural network automatically learned from data [5, 26, 27, 88]. Given the initial states $\boldsymbol{h}_1^{t_0}, \boldsymbol{h}_2^{t_0} \cdots \boldsymbol{h}_N^0$ for all objects, the latent states of trajectories at arbitrary time steps can be calculated with a black-box ODE solver as follows:

$$\boldsymbol{h}_i^t = \boldsymbol{h}_i^{t_0} + \int_{s=t_0}^{t} \Phi\left(\boldsymbol{h}_1^s, \boldsymbol{h}_2^s, \cdots, \boldsymbol{h}_N^s\right) ds. \tag{3}$$

We model interacting dynamics in mesh-based physical systems by learning neural ODEs in the latent space, with a GNN as the ODE function Φ to model the continuous interaction among mesh points. The latent initial states $\boldsymbol{h}_1^{t_0}, \boldsymbol{h}_2^{t_0}, \cdots, \boldsymbol{h}_N^{t_0}$ are computed via an encoder and the decoder recovers the whole trajectory $\boldsymbol{X}^t$ ($t_0 < t \leq T$) based on the latent states at each time step.

3 THE PROPOSED FAIR

3.1 Framework Overview

In this paper, we study an underexplored yet important problem of long-term mesh-based simulations and introduce a new approach named FAIR from a future-to-present perspective. Existing methods [58, 67, 82] usually utilize the current states for next-time predictions, followed by iterative rollouts to predict whole trajectories while our FAIR allows the model to maintain foresight throughout the evolution by tracking long-term future predictions. Specifically, our proposed FAIR incorporates a continuous graph ODE model that is enhanced with past states to model the dynamics of both mesh nodes and edges. In this way, we can generate coarse long-term trajectories under a multi-task learning framework. To further improve the accuracy of our short-term predictions, we employ a channel aggregation strategy, which refines the trajectory through an interpolation process. Finally, we adopt pyramid-like propagation for the whole trajectories. An overview of our proposed FAIR can be found in Figure 2 and we will elaborate on the details as follows.

3.2 Coarse Foresight with Graph ODE Twins

The key insight of our proposed FAIR is to learn from long-term trajectories. As a preliminary step, it is crucial to generate high-quality long-term trajectories based on historical predictions. Instead of using inefficient iterative rollouts [4, 39, 58], we follow the idea of neural ODEs [5] to model the continuous evolution of both mesh nodes and edges within dynamic systems. To learn the evolution between mesh points, we extend neural ODEs into graph ODE twins, which employ a neighborhood aggregation mechanism to update the representations of both nodes and edges and are flexible to produce outputs at any given timestamp. In particular, FAIR leverages an encoder-ODE-decoder architecture where both the encoder and decoder components are built upon message passing neural networks (MPNNs). The effectiveness of the graph ODE twins is further enhanced by augmenting latent states with historical data, thereby improving the capability to capture evolving patterns under potential noise.

MPNN-based Encoder. To begin with, we first generate state representations for mesh nodes and their associated edge using a message passing mechanism [31]. Specifically, both node and edge embeddings are initialized using feed-forward networks (FFNs) as:

$$\boldsymbol{v}_i^{(0)} = f^n\left(\boldsymbol{x}_i\right), \boldsymbol{e}_{ij}^{(0)} = f^e\left(\boldsymbol{p}_i - \boldsymbol{p}_j\right), \tag{4}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{p}_i$ are the feature and position vectors of node i, respectively. $f^n(\cdot)$ and $f^e(\cdot)$ are implemented by two FNNs for nodes and edges, respectively. Then, we stack a range of MPNN layers to learn semantics from geometric graphs in an iterative manner. The updating rule for each node i can be summarized as:

$$\boldsymbol{v}_i^{(l+1)} = \psi^n(\boldsymbol{v}_i^{(l)}, \sum_{j \in \mathcal{N}(i)} \boldsymbol{e}_{ij}^{(l)}), \boldsymbol{e}_{ij}^{(l+1)} = \psi^e(\boldsymbol{v}_i^{(l)}, \boldsymbol{v}_j^{(l)}, \boldsymbol{e}_{ij}^{(l)}), \tag{5}$$

where $\boldsymbol{v}_i^{(l)}$ and $\boldsymbol{e}_{ij}^{(l)}$ are the node and edge embeddings at the layer l, respectively. $\mathcal{N}(i)$ denotes the neighbors of node i. ψ^e and ψ^n are two FFNs for feature transformations. After stacking L layers, we can generate discriminative node end edges representations for the current timestamp t_0, *i.e.*, $\boldsymbol{h}_i^{t_0} = \boldsymbol{v}_i^{(L)}$ and $\boldsymbol{r}_{ij}^{t_0} = \boldsymbol{e}_{ij}^{(L)}$ for the subsequent generative model.

Graph ODE. Neural ODEs are commonly used to model dynamical systems with continuous evolution, which can output flexible predictions at any given timestamps. While previous approaches often integrate neighborhood information into ODEs to model interacting dynamics [19, 26, 27], they typically fall short in explicitly capturing the evolving dynamics of edges. To address this gap, we introduce a propagator named graph ODE twins, which adopts separate neural ODEs to model the evolution of both nodes and edges. Furthermore, we notice that data-driven prediction models [26, 27] often struggle to accurately deduce continuous evolution solely based on the current state. To mitigate this, we incorporate historical states as supplementary data in our graph ODE model. In practice, we use them to augment the current embeddings in ODEs, thereby enhancing the capacity to capture the evolving dynamics as well. In formulation, we generate augmented embeddings at the timestamp t as follows:

$$\bar{\boldsymbol{h}}_i^t = \begin{bmatrix} \boldsymbol{h}_i^t \\ \boldsymbol{h}_i^{t-1} \end{bmatrix}, \bar{\boldsymbol{r}}_{ij}^t = \begin{bmatrix} \boldsymbol{r}_{ij}^t \\ \boldsymbol{r}_{ij}^{t-1} \end{bmatrix}, \tag{6}$$

where $\boldsymbol{h}_i^{t-1}$ and $\boldsymbol{r}_{ij}^{t-1}$ are from the last timestamps. Then, we model the evolution using both augmented node embeddings and edge embeddings based on the following ODEs:

$$\begin{aligned} \frac{d\boldsymbol{h}_i^t}{dt} &= \Phi^n(\bar{\boldsymbol{h}}_i^t, \sum_{j \in \mathcal{N}(i)} \bar{\boldsymbol{r}}_{ij}), \\ \frac{d\boldsymbol{r}_{ij}^t}{dt} &= \Phi^e(\bar{\boldsymbol{r}}_{ij}^t, \bar{\boldsymbol{h}}_i^t, \bar{\boldsymbol{h}}_j^t), \end{aligned} \tag{7}$$

where Φ^n and Φ^e are implemented by two FFNs. Through a standard ODE solver, we are able to output the hidden embeddings for the future timestamps ranging from t_1 to t_L at one time with the step

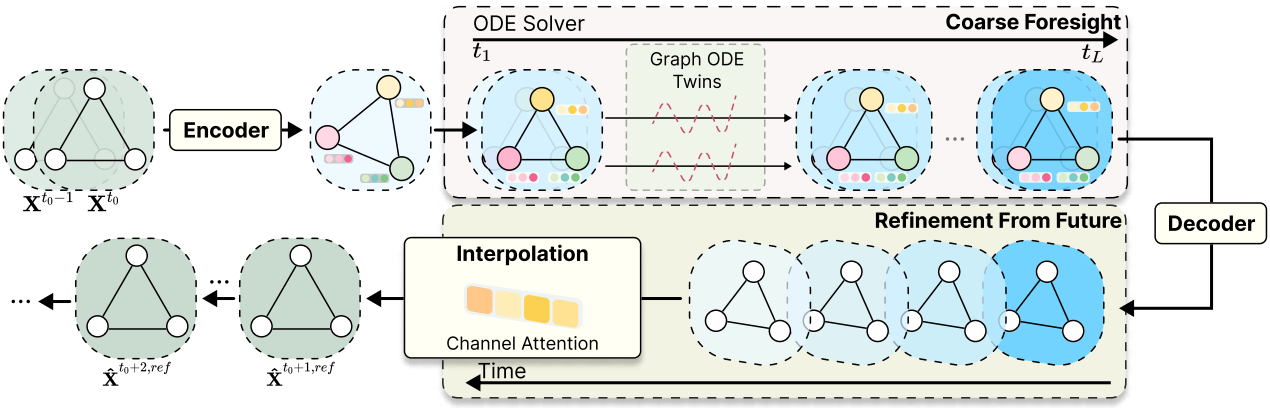

Figure 2: An overview of the proposed FAIR. Our innovation is to leverage a new future-to-present learning perspective to effectively capture long-term dynamics. In particular, FAIR adopts an MPNN-based encoder to generate node representations, which are fed into graph ODE twins to generate trajectory predictions at different timestamps. These predictions are aggregated with channel attention to refine the short-term predictions. These foresight and refinement steps are conducted alternatively for accurate long-term predictions.

size r and the number of predictions L (*i.e.*, $t_l = t_0 + rl - r + 1$). Different r indicates a different horizon of predictions. Our graph ODE model is a special case of delay differential equations (DDEs) [2], *i.e.*, $\frac{d\boldsymbol{h}^t}{dt} = \phi(\boldsymbol{h}^t, \boldsymbol{h}^{t-\tau}, t)$, which has been shown to have an improved capacity for capturing non-linear dynamics [97]. We further provide a theorem to show that our graph ODE has a unique absolutely continuous solution. To begin, denote $\boldsymbol{y}^t = (\boldsymbol{h}_1^t, \ldots, \boldsymbol{h}_N^t, \boldsymbol{r}_{12}^t, \ldots, \boldsymbol{r}_{N-1,N}^t)$, and then our system can be represented as:

$$\begin{cases} \dfrac{d\boldsymbol{y}^t}{dt} = \Phi(\boldsymbol{y}^t, \boldsymbol{y}^{t-1}), & t \in [t_0, t_0 + T] \\ \boldsymbol{y}(t) = \boldsymbol{y}(t_0 - 1), & t \in [t_0 - 1, t_0), \end{cases} \tag{8}$$

where we add the definition of $\boldsymbol{y}$ in the interval $[t_0 - 1, t_0]$ to make sure our system is well-defined.

Lemma 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. 8 has a unique absolutely continuous solution.*

The proof has been shown in Appendix A. Note that the existence of unique solutions is still a desirable property as it makes the learning problem well-defined and the trajectory predictions consistent [33]. Non-unique solutions would bring challenges in training and inference. The lemma is a starting point to justify the ODE modeling approach.

MPNN-based Decoder. In the end, we adopt a decoder $\psi_{dec}(\cdot)$ to generate the predictions at any given timestamps as follows:

$$\hat{\boldsymbol{x}}_i^t = \psi_{\text{dec}}(\{\boldsymbol{h}_i^t\}_{i\in\mathcal{V}}, \{\boldsymbol{p}_{ij}\}_{(i,j)\in\mathcal{E}}), \tag{9}$$

where $\boldsymbol{p}_{ij} = \boldsymbol{p}_i - \boldsymbol{p}_j$ is reused to provide position information. The architecture of the decoder is the same as the MPNN in the encoder to ensure effective neighborhood learning. To train our graph ODE twins, we minimize the mean square error for different timestamps:

$$\mathcal{L}_{\text{ode}} = \sum_{l=1}^{L}\sum_{i=1}^{N} \|\boldsymbol{x}_i^{t_l} - \hat{\boldsymbol{x}}_i^{t_l}\|_2, \tag{10}$$

in which each timestamp t_l corresponds to a different t_l-step ahead trajectory prediction task.

Comparison with One-step Predictors. Current one-step predictors generate states for the next-time states [4, 58, 67] and then proceed in an autoregressive manner for entire trajectories. In contrast, our graph ODE twins have two strengths as follows. Firstly, our approach is capable of capturing the continuous interactive dynamics that naturally occur in the mesh-based physical system. Secondly, our approach is optimized under the framework of multi-task learning. In particular, we generate predictions with different time steps, each of which corresponds to a task. These tasks share the same encoder and decoder. This strategy enables the model to learn long-term dependency with limited error accumulation, thereby enhancing the learning process.

3.3 Refinement with Interpolation

While our proposed graph ODE twins model is effective, it has the potential to underfit long-term trajectories in the multi-task learning framework. To address this issue, we further introduce a refinement module, which uses the coarse long-term trajectories to improve the accuracy of short-term predictions, *i.e.*, $\hat{\boldsymbol{x}}_i^{t_1}$. Considering that both future states beyond the target one, *i.e.*, $\{\hat{\boldsymbol{x}}_i^{t_l}\}_{l=2}^{L}$ and current states, *i.e.*, $\hat{\boldsymbol{x}}_i^{t_0}$, are both available, this refinement process can be viewed as a form of interpolation.

To be specific, we introduce a set of learnable parameters to serve as channel attention, which are the weights for interpolation. Moreover, the message passing procedure is performed in the observation space rather than the embedding space to learn the offset with enhanced efficiency. In formulation, we define the learnable weights as $\boldsymbol{w}^l \in \mathbb{R}^F$, and the aggregated observation $\boldsymbol{z}_i^{t_1}$ for node i can be written as follows:

$$\boldsymbol{z}_i^{t_1} = \boldsymbol{x}_i^{t_0} \odot \boldsymbol{w}^0 + \sum_{l=1}^{L} \hat{\boldsymbol{x}}_i^{t_l} \odot \boldsymbol{w}^l, \tag{11}$$

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. 9;
        Minimize the mean square error for these timestamps in Eqn. 10;
        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. 9;
        Generate the refined predictions from Eqn. 13;
        Minimize the mean square error for the target in Eqn. 14
        Update the parameters using gradient descent;
    end for
end while
```

where target states $\hat{\boldsymbol{x}}_i^{t_1}$ is also involved for a more comprehensive offset mining and $\odot$ denotes element-wise product of two vectors. Compared to standard interpolation, our approach facilitates adaptive information summarization, thereby enhancing the capacity to capture complex patterns. Finally, we stack several MPNNs for neighborhood interaction, which outputs the final predictions of the offset as follows:

$$\hat{\boldsymbol{x}}_i^{t_1,\text{off}} = \psi_{\text{ref}}(\{\boldsymbol{z}_i^{t_1}\}_{i\in\mathcal{V}}, \{\boldsymbol{p}_{ij}\}_{(i,j)\in\mathcal{E}}), \tag{12}$$

where ψ_{ref} has a similar architecture to the MPNN-based decoder, but with shallower layers. In the end, the refined predictions can be obtained by combining coarse predictions and offsets:

$$\hat{\boldsymbol{x}}_i^{t_1,\text{ref}} = \hat{\boldsymbol{x}}_i^{t_1} + \hat{\boldsymbol{x}}_i^{t_1,\text{off}}. \tag{13}$$

The mean square error is minimized for the target observations:

$$\mathcal{L}_{\text{re}} = \sum_{i=1}^{N} \|\hat{\boldsymbol{x}}_i^{t_1,\text{ref}} - \boldsymbol{x}_i^{t_1}\|_2. \tag{14}$$

In contrast to coarse foresight over multiple timestamps, our refinement module targets a single timestamp, enabling us to further minimize the training loss. Unlike previous one-step predictors [4], our proposed FAIR uses future predictions generated by the graph ODE twins for interpolation, which is empirically simpler than extrapolation. Our proposed FAIR employs a two-stage optimization strategy. In the first stage, we train the model to generate coarse long-term trajectories using a multi-task learning framework. In the second stage, we shift our focus to fine-tuning short-term predictions and eliminate additional supervision in Eqn. 10. A comprehensive summary of the learning algorithm can be found in Algorithm 1. This approach can be illustrated as a knowledge distillation framework [6, 17, 55] where the teacher model (*i.e.*, graph ODE twins) gains broad and generalized knowledge from multiple tasks, while the student model (*i.e.*, refinement module) focuses on the specific target, which is capable of benefiting from the foresight provided by the teacher model as well. In addition, our foresight steps would generate a range of coarse predictions with potential noise, which would serve as the perturbation to the input for the refinement step to release potential overfitting.

Pyramid-like Propagation. To generate long-term predictions, we alternatively conduct foresight generation and interpolation, resulting in a pyramid-like architecture as illustrated in Figure 1. By employing the graph ODE model, our FAIR is able to capture the long-term dynamics governed by the underlying rules. In addition, during our pyramid-like alternative propagation, our FAIR gains insight into future states which helps in mitigating potential error accumulation. A comprehensive summary of the inference algorithm can be found in Algorithm 2.

Error Bound. To obtain the stochastic error bound of Eqn. 10 in Section 3.2, we assume the $\boldsymbol{x}_i^t$ follows the regression model $\boldsymbol{x}_i^t = f(\boldsymbol{h}_i^t) + \boldsymbol{\varepsilon}_i$, where $\boldsymbol{h}_i^t$ denotes an informative representation of the system state and the noise term accounts for intrinsic stochasticity or unobserved factors. $f : \mathbb{R}^d \mapsto \mathbb{R}^F$ is the true regression function. Moreover, we make the following assumptions: (1) $\boldsymbol{h}_i^t$ has a density function $f_{\boldsymbol{h}^t}$ that is bounded away from 0 and infinity; (2) The regression error $\boldsymbol{\varepsilon}_i$ are Gaussian random samples with zero mean and covariance matrix $\sigma_\varepsilon \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix of dimension F. The first assumption is made to avoid pathological cases and the second assumption is common [22, 66] in regression settings and can be justified by the central limit theorem. Assume the $\hat{f}$ is the minimizer of the empirical loss $\hat{f} = \arg\min_{f\in\mathcal{F}} \mathcal{L}_{ode}$, where $\mathcal{F}$ is the class of neural network in the MLP. We are interested in the error bound $\mathcal{E}(\hat{f}, f) := \sum_{l=1}^{L} \mathbb{E}\left[\|\hat{f}(\boldsymbol{h}^{t_l}) - f(\boldsymbol{h}^{t_l})\|^2\right]$, where $\boldsymbol{h}^{t_l}$ is a independent copy of $\boldsymbol{h}_i^{t_l}$ the the expectation is taken over both random elements $\hat{f}$ and $\boldsymbol{h}^{t_l}$. On the basis of [22, 66],the following lemma presents the generalization error of our learning framework for coarse foresight.

Lemma 2. *Assume that there exists a fixed function F_e such that $\sup_{f\in\mathcal{F}} \|f\|_{L^\infty} \le F_e$ and denote $M_{\mathcal{F}} = \max\{1, \|F_e\|_{L^\infty}\}$. For all $\delta > 0$ and denote the δ-covering number of $\mathcal{F}$ in L-infinity norm by $\mathcal{N}_\delta := \mathcal{N}(\mathcal{F}, \mathcal{L}_\infty, \delta)$, the generalization error of our proposed model is bounded by:*

$$\begin{aligned}\mathcal{E}(\hat{f}, f) \le & 4 \min_{f^*\in\mathcal{F}} \sum_{l=1}^{L} \mathbb{E}\|f^*(\boldsymbol{h}^{t_l}) - f(\boldsymbol{h}^{t_l})\|^2 + (8F^{\frac{3}{2}}\sigma_\epsilon + 36FM_{\mathcal{F}})\delta L \\ & + \frac{2}{N}\left\{\left(\frac{37}{9}FM_{\mathcal{F}}^2 + 16\sigma_\varepsilon^2\right)\log\mathcal{N}_\delta + 4(3FM_{\mathcal{F}}^2) + 4\sigma_\varepsilon^2\right\} L.\end{aligned} \tag{15}$$

The proof is omitted due to the space limitation. The first term on the right-hand side of Equ. 15 represents the approximation bias. This term reflects the expressive capabilities of the neural network. The last two terms correspond to the stochastic error. Notably, a smaller value of δ leads to an increased covering number, denoted by $\log\mathcal{N}_\delta$. This scenario illustrates a fundamental bias-variance

Table 1: The RMSE results of the compared methods over different time steps of 1, 50, and all time steps. The best results are displayed in bold. Partial results are consistent with [4]. OOM indicates out-of-memory.

Dataset	CylinderFlow RMSE ($\times 10^{-3}$) ↓			Airfoil RMSE ($\times 10^{-1}$) ↓			DeformingPlate RMSE ($\times 10^{-4}$) ↓			InflatingFont RMSE ($\times 10^{-4}$) ↓			Improvement
	1	50	all	1	50	all	1	50	all	1	50	all	
GraphUNets [16]	8.09	187	1650	2.93	117	611	2.03	5.19	54.6	OOM	OOM	OOM	53.0%
GNS [65]	2.61	50.7	176	5.29	175	639	2.23	3.21	17.2	2.14	36.9	50.7	45.4%
MeshGraphNet [58]	2.26	43.9	107	4.35	166	695	1.98	2.88	15.1	1.95	17.8	36.5	34.9%
MS-GNN-Grid [38]	2.20	27.4	84.9	2.68	122	556	2.20	**2.78**	14.8	1.87	32.4	37.8	28.4%
GMR-GMUS [21]	2.25	24.4	87.1	2.29	108	414	2.96	3.41	23.2	2.42	13.6	29.9	26.1%
GPMS [24]	2.10	23.9	74.9	2.46	110	426	2.74	3.68	14.8	2.49	11.9	27.3	21.3%
BSMS-GNN [4]	2.04	24.2	83.7	2.88	110	421	2.87	3.18	16.9	1.77	10.8	22.0	19.8%
FAIR (Ours)	**1.75**	**22.6**	**62.4**	**1.88**	**95**	**405**	**1.92**	2.82	**13.7**	**0.79**	**10.6**	**17.8**	-

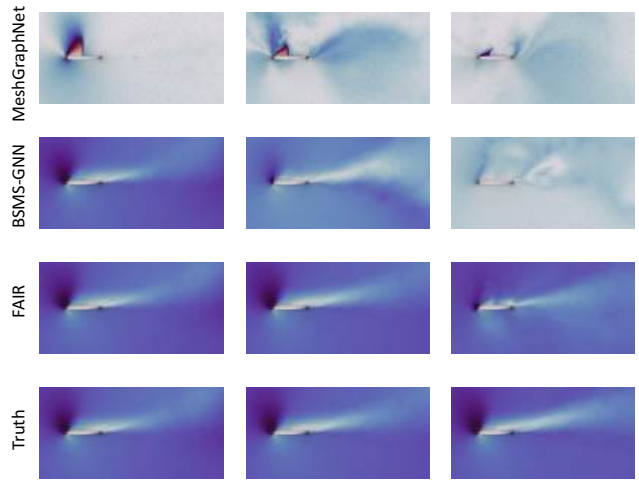

Figure 3: Visualization of different methods on *Airfoil* at different time steps among 1, 100, and 200.

trade-off as follows. On the one hand, larger networks typically exhibit superior learning abilities with a reduced approximation error. On the other hand, the covering number also increases along with the size of the network, which indicates a higher variance.

4 EXPERIMENTS

4.1 Experimental Setup

4.1.1 Datasets. To evaluate our proposed FAIR, we employ four mesh-based benchmark physics simulation datasets [4, 58] as follows: (1) *CylinderFlow*, which simulates the flow of an incompressible fluid around a cylinder; (2) *Airfoil*, which focuses on the simulating compressible flow around an airfoil; (3) *DeformingPlate*, which involves the simulation of elastic plate deformation using an actuator; and (4) *InflatingFont*, which depicts the inflation of enclosed elastic surfaces. For each dataset, the train/val/test splits follow the recent work [4].

4.1.2 Baselines. To validate the effectiveness of our FAIR, we compare it with a range of neural simulation baselines including GraphUNets [16], GNS [65], MeshGraphNet [58], MS-GNN-GRID [38], GMR-GMUS [21], GPMS [24], and BSMS-GNN [4].

4.1.3 Implementation Details. The root mean square deviation (RMSE) is taken as the metric to evaluate the performance. We follow the same experimental settings in [4], which are the latest and easily reproducible are not reproducible, which is different from ours [4]. We vary the prediction lengths to show the performance in both short-term and long-term forecasting tasks. We set the step size r and the future prediction number L as 2 and 3 as default, respectively, and parameter sensitivity can be seen in Sec. 4.5. We execute all experiments on a single A100 GPU, including the speed test. For the results of MeshGraphNet, MS-GNN-GRID and BSMS-GNN, we quote their results reported by [4]. Multiple steps ahead are predicted by rollout for the baselines.

4.2 Quantitative Comparison

The compared performance of seven competing approaches on four datasets is recorded in Table 1. From the results, we can have three observations as follows. Firstly, GraphUNets achieve the worse performance compared with the other methods targeting at dynamical system modeling. This indicates the difficulties of the mesh-based physics simulations and we need to design special approaches for this problem. Secondly, it can be observed that our proposed FAIR performs the best across all four datasets in terms of both short-term and long-term forecasting. In particular, compared to the best baseline BSMS-GNN, our FAIR achieves an average error reduction of 34.4% in 1-step simulations and 16.8% in all-step simulations. The significant performance improvement of our proposed FAIR compared with baselines can be attributed to two factors: (1) The introduction of our graph ODE twins in our foresight step. The graph ODE can significantly reduce the error accumulation in the multi-task learning framework, which also provides future information for short-term predictions; (2) The introduction of our refinement step. This step can leverage coarse future predictions to refine short-term predictions with channel aggregation, thus mitigating the underfitting resulting from multi-task learning. Thirdly, the performance of our proposed FAIR on MS-GNN-Grid is a little worse

Table 2: Ablation studies of different variants on four datasets.

Dataset	CylinderFlow RMSE ($\times 10^{-3}$) ↓			Airfoil RMSE ($\times 10^{-1}$) ↓			DeformingPlate RMSE ($\times 10^{-4}$) ↓			InflatingFont RMSE ($\times 10^{-4}$) ↓			Improvement
	1	50	all	1	50	all	1	50	all	1	50	all	
FAIR w/o ODE	1.82	24.3	75.5	1.94	102	595	1.95	2.93	16.3	0.92	11.4	22.1	11.0%
FAIR w/o R	2.03	27.2	67.5	2.31	110	543	2.16	3.14	16.1	1.12	12.1	21.6	16.0%
FAIR w/o MTL	1.91	25.0	94.1	2.07	108	602	1.99	3.16	16.9	1.21	13.6	25.0	18.7%
FAIR w/ E	1.98	25.2	78.1	2.02	103	440	2.16	3.15	15.0	0.93	11.5	21.0	11.1%
FAIR w/ Cubic	2.37	35.1	108	2.79	105	585	2.34	4.01	23.8	1.13	15.8	27.3	30.4%
FAIR (Full Model)	**1.75**	**22.6**	**62.4**	**1.88**	**95**	**405**	**1.92**	**2.82**	**13.7**	**0.79**	**10.6**	**17.8**	-

than MS-GNN-Grid. The potential reason is the high complexity of DeformingPlate makes it harder to generate accurate foresight, which could deteriorate the model performance.

4.3 Visualization

In this part, we conduct visualization to compare our FAIR with representative baselines and the ground truth. The compared results at different time steps on *Airfoil* and *CylinderFlow* are shown in Figure 3 and Figure 4, respectively. From the results, we can make the following observations: (1) We can find that serious error accumulation occurs for one-step predictors (*i.e.*, MeshGraphNet and BSMS-GNN). For example, in the last frame of Figure 3, MeshGraphNet and BSMS-GNN have a huge gap compared with the ground truth. (2) Our FAIR can make precise long-term predictions consistently. The potential reason is that our ODE-based model can capture the continuous dynamics in physical systems and mitigate the error accumulation during propagation. In particular, all the baselines fail to reflect correct flow fields at the last time step while our FAIR can still approximate the ground truth. (3) Our proposed FAIR can also make accurate short-term predictions, which demonstrates that future foresight can benefit short-term predictions as well. Moreover, Figure 5 shows the difference on 3D dataset *DeformationPlate*. We can observe that our proposed methods can suffer limited errors for long-term predictions, which validates the superiority of our proposed FAIR.

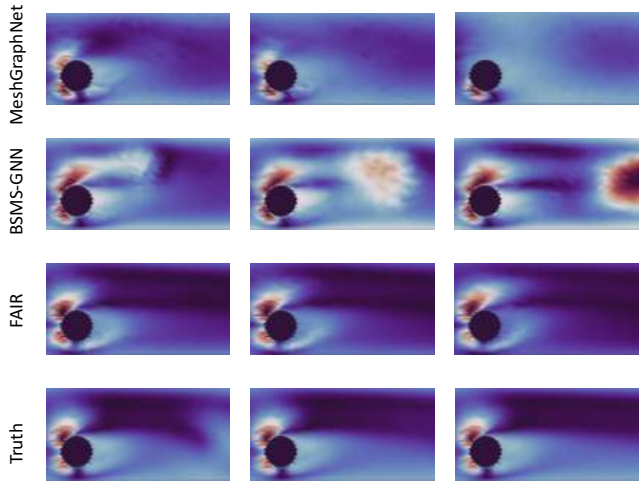

Figure 4: Visualization of different methods on *CylinderFlow* at multiple time steps. We render the velocity in the fluid field at different time steps among 100, 300 **and** 500**.**

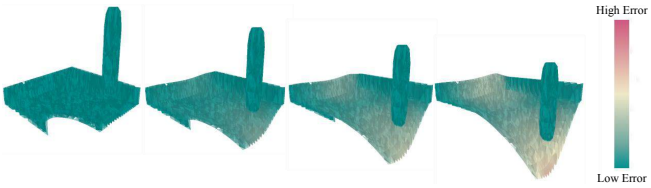

Figure 5: Visualization of FAIR on 3D dataset *DeformingPlate* at different time steps among 1, 100, 200 **and** 300**.**

4.4 Ablation Study

To evaluate the effectiveness of different subcomponents in our proposed FAIR, we introduce several model variants as follows: (1) FAIR w/o ODE, which removes the graph ODE twins module and utilizes one-step predictors for coarse foresight. (2) FAIR w/o R, which removes the refinement from the future module and outputs the results using graph ODE. (3) FAIR w/o MTL, which removes the multi-task learning framework and involves the next-time predictions in refinement. (4) FAIR w/ E, which utilizes the end-to-end training manner to train the whole model. (5) FAIR w/ Cubic, which utilizes the cubic interpolation instead of our refinement module. The compared results are presented in Table 2. From these results, we have the following observations. *Firstly*, by comparing FAIR w/o ODE with the full model, we can validate that the graph ODE twins module can capture continuous dynamics to provide high-quality future information for accurate predictions. *Secondly*, our full model achieves better performance than FAIR w/o R, which validates that the refinement step is indispensable by preventing potential underfitting in the multi-task learning framework. *Thirdly*, FAIR w/o MTL performs much worse than the full model, which validates that future information is capable of making a critical contribution to effective long-term mesh-based simulations. *Fourthly*, FAIR w/ E performs much worse than our full model. The potential reason is that the end-to-end training would involve all these modules simultaneously, which makes the training more difficult. *Fifthly*, the performance of FAIR w/ Cubic is worse than that of our full model, which can validate that cubic interpolation cannot learn

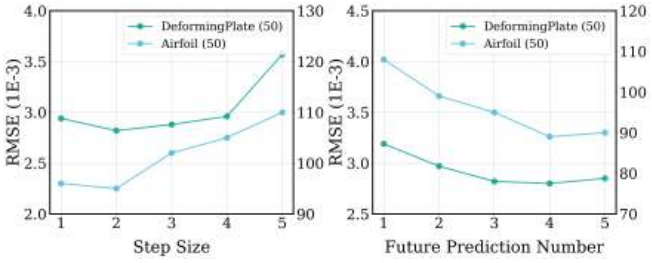

Figure 6: The performance with respect to different step sizes (horizon) r and prediction numbers L on *DeformingPlate* and *Airfoil*.

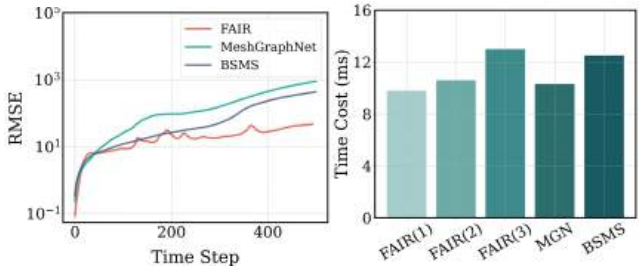

Figure 7: Left. The RMSE results of our proposed FAIR and two baselines with respect to different time steps on *Airfoil* (one sample). Right. The comparison of the running time of FAIR (with different future prediction numbers), BSMS (BSMS-GNN), and MGN (MeshGraphNet).

the relationships between different nodes during refinement while cubic interpolation considers each node independently.

4.5 Further Analysis

4.5.1 Sensitivity Analysis. We investigate the impact of different parameters on the performance of FAIR, *i.e.*, the step size r, and the number of future predictions L. First, we vary r in $\{1, 2, 3, 4, 5\}$ with the other parameters fixed and the results are shown in Figure 6 (Left). We can observe that the errors first decrease and then increase as r rises. The potential reason is that when r is small, a larger r can provide a larger horizon, while too large r would be far away from our target, which makes the interpolation unreliable. Second, we vary the prediction number L in $\{1, 2, 3, 4, 5\}$, and the results can be found in Figure 6 (Right). We can observe an error reduction when L rises before saturation, indicating that including more future information boosts model performance.

4.5.2 Predictions at Different Time Steps. We compare the prediction errors of our proposed FAIR and two baselines, i.e., MeshGraphNet and BSMS at different time steps in terms of RMSE on *Airfoil*. The results are shown in Figure 7 (Left). We can find that our proposed FAIR exhibits stronger modeling capabilities with relatively lower errors at large time steps while both two baselines suffer from serious error accumulations. This validates that our proposed FAIR utilizes foresight to reduce the error accumulation for long-term predictions.

Table 3: The compared results ($\times 10^{-3}$) with respect to different step times on the weather dataset.

Method	MAE@3	MSE@3	MAE@5	MSE@5
LSTM [23]	46.9	76.7	58.1	88.3
GRU [7]	48.2	80.2	60.0	90.6
HOPE [44]	52.8	81.8	62.8	93.7
FAIR (Ours)	45.2	71.0	59.8	85.5

4.5.3 Efficiency Analysis. In this part, we analyze the efficiency of MeshGraphNet, BSMS-GNN, and the proposed FAIR by varying the number of future predictions L. The computational time of one-step predictions on a single NVIDIA A100 GPU is reported in Figure 7 (Right). From the results, we have two different observations. Firstly, our proposed FAIR achieves the performance increasement of 19.6% compared with the best baseline with similar efficiency, which validates the application value of our method. Secondly, we can find that the time cost of the proposed FAIR slightly increases as L rises. Considering there is a trade-off between efficiency and effectiveness, we set L as 3 in our implementation as default.

4.5.4 Performance on Noisy Data. Lastly, we compare the performance of four different methods on a real-world noisy weather dataset[1], which records authentic radar reflectivity maps at different selected locations. The compared results at different time steps are shown in Table 3. From the results, we can validate the superiority of our proposed FAIR in comparison to these baselines. The potential reason is that our FAIR follows a multi-task learning procedure, which is more robust to the noisy data.

5 RELATED WORK

5.1 Learning-based Physics Simulations

Physics simulations can be advantageous in science when model parameters or boundary conditions are insufficient. Due to their great efficacy, learning-based physics simulations are gaining popularity in a variety of domains such as computational fluid dynamics [18, 24, 49, 60, 68, 69, 86]. Initially, convolutional neural network-based methods are often built to learn from regular grids [57]. Recently, MeshGraphNet [58] makes an attempt to incorporate GNNs into learning mesh-based physics simulations on irregular meshes, followed by several extensions [4, 65, 67]. However, these algorithms often generate next-step predictions using current states, which fails to make accurate long-term predictions. [21] adopts an encoder-decoder structure to compress data for long sequence modeling with Transformer. Recent work [25] uses differentiable orthogonal spline collocation for interpolation to increase the forecasting efficiency while our work utilizes adaptive interpolation for refinement of short-term predictions based on our graph ODE predictions.

5.2 Long-term Forecasting

Based on historical observations, long-term forecasting [9, 32, 34, 36, 46, 47, 52, 79, 94, 96] aims to make predictions for a long horizon with various applications with weather forecasting [3, 92]

[1] https://tianchi.aliyun.com/competition/entrance/231596

and economic analysis [8]. A range of Transformer-based architectures [41, 70, 95, 96] have been introduced for long-term forecasting, which can get rid of gradient vanishing and exploding in recurrent neural networks (RNNs) [64]. These approaches usually focus on modeling single-agent systems. Recent work [39] aims to capture all frequency components in PDE solutions to enhance long-term forecasting and utilize diffusion models for refinement. In contrast, our work utilizes graph ODE twins for long-term foresight in a multi-task learning framework and learning interpolation methods for refinement.

5.3 Continuous GNNs

Neural ODEs [5] have been shown effective in forwarding networks in a continuous way. In recent years, several works have combined GNNs with neural ODEs [59, 81], which have been applied in exploring both static and dynamic graphs. In static graphs, continuous GNNs can be adopted to mitigate the overfitting of discrete GNNs [81, 91]. In contrast, several works have adopted continuous GNNs to extract continuous spatio-temporal relationships for traffic forecasting [13, 28] and social network analysis [43]. In this work, enhanced by incorporating the past states, our proposed FAIR utilizes graph ODE twins to model the evolving dynamics of both nodes and edges for coarse foresight.

6 CONCLUSION

In this paper, we study the problem of long-term mesh-based physics simulations and propose a new approach FAIR to address it. Our FAIR utilizes a future-to-present perspective, which consists of two steps, *i.e.*, foresight and refinement, for accurate simulations. In the first step, we use a graph ODE model that integrates previous states to learn coarse long-term trajectories using a multi-task learning framework. In the second step, we employ a channel aggregation strategy to aggregate the trajectories for refined short-term predictions. Our proposed FAIR can provide accurate long-term trajectories by the alternative propagation of foresight and refinement. We believe that our study provides a brand-new perspective in learning long-term mesh-based simulations. In the future work, we plan to expand the proposed FAIR to accommodate more complex simulations in physical and biological applications such as molecular dynamics simulations and protein structural analysis. We will further try to increase the stability of the end-to-end optimization.

ACKNOWLEDGMENT

This work was partially supported by NSF 2211557, NSF 1937599, NSF 2119643, NSF 2303037, NSF 2312501, NSF 2319449, NSF 2312502, NASA, SRC JUMP 2.0 Center, Amazon Research Awards, and Snapchat Gifts. We would like to thank Yadi Cao for his valuable help throughout this project.

REFERENCES

[1] Ferran Alet, Adarsh Keshav Jeewajee, Maria Bauza Villalonga, Alberto Rodriguez, Tomas Lozano-Perez, and Leslie Kaelbling. 2019. Graph element networks: adaptive, structured computation and memory. In *ICML*.
[2] Balakumar Balachandran, Tamás Kalmár-Nagy, and David E Gilsinn. 2009. *Delay differential equations.* Springer.
[3] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. 2023. Accurate medium-range global weather forecasting with 3D neural networks. *Nature* (2023).
[4] Yadi Cao, Menglei Chai, Minchen Li, and Chenfanfu Jiang. 2023. Efficient Learning of Mesh-Based Physical Simulation with Bi-Stride Multi-Scale Graph Neural Network. In *ICML*.
[5] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. In *NeurIPS*.
[6] Jang Hyun Cho and Bharath Hariharan. 2019. On the efficacy of knowledge distillation. In *ICCV*.
[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
[8] Alexander Chudik, Kamiar Mohaddes, M Hashem Pesaran, Mehdi Raissi, and Alessandro Rebucci. 2021. A counterfactual economic analysis of Covid-19 using a threshold augmented multi-country model. *Journal of International Money and Finance* (2021).
[9] Patrick Dendorfer, Vladimir Yugay, Aljosa Osep, and Laura Leal-Taixé. 2022. Quo Vadis: Is Trajectory Forecasting the Key Towards Long-Term Multi-Object Tracking?. In *NeurIPS*.
[10] Fabio V Difonzo, Paweł Przybyłowicz, and Yue Wu. 2024. Existence, uniqueness and approximation of solutions to Carathéodory delay differential equations. *J. Comput. Appl. Math.* (2024).
[11] Qiujie Dong, Zixiong Wang, Manyi Li, Junjie Gao, Shuangmin Chen, Zhenyu Shu, Shiqing Xin, Changhe Tu, and Wenping Wang. 2023. Laplacian2mesh: Laplacian-based mesh understanding. *IEEE Transactions on Visualization and Computer Graphics* (2023).
[12] Thomas D Economon, Francisco Palacios, Sean R Copeland, Trent W Lukaczyk, and Juan J Alonso. 2016. SU2: An open-source suite for multiphysics simulation and design. *Aiaa Journal* (2016).
[13] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. 2021. Spatial-temporal graph ode networks for traffic flow forecasting. In *KDD*. 364–373.
[14] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).
[15] Stathi Fotiadis, Eduardo Pignatelli, Mario Lino Valencia, Chris Cantwell, Amos Storkey, and Anil A Bharath. 2020. Comparing recurrent and convolutional neural networks for predicting wave propagation. *arXiv preprint arXiv:2002.08981* (2020).
[16] Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. In *ICML*.
[17] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision* (2021).
[18] Xiaoxiao Guo, Wei Li, and Francesco Iorio. 2016. Convolutional neural networks for steady flow approximation. In *KDD*.
[19] Jayesh K Gupta, Sai Vemprala, and Ashish Kapoor. 2022. Learning Modular Simulations for Homogeneous Systems. In *NeurIPS*.
[20] Igor Guskov, Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. 2002. Hybrid meshes: multiresolution using regular and irregular refinement. In *Annual Symposium on Computational Geometry*.
[21] Xu Han, Han Gao, Tobias Pfaff, Jian-Xun Wang, and Li-Ping Liu. 2022. Predicting physics in mesh-reduced space with temporal attention. *arXiv preprint arXiv:2201.09113* (2022).
[22] Satoshi Hayakawa. 2019. Note on the generalization bounds of the empirical risk minimizer. (2019).
[23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[24] Yeping Hu, Bo Lei, and Victor M Castillo. 2023. Graph Learning in Physical-informed Mesh-reduced Space for Real-world Dynamic Systems. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4166–4174.
[25] Chuanbo Hua, Federico Berto, Michael Poli, Stefano Massaroli, and Jinkyoo Park. 2024. Learning Efficient Surrogate Dynamic Models with Graph Spline Networks. *Advances in Neural Information Processing Systems* 36 (2024).
[26] Zijie Huang, Yizhou Sun, and Wei Wang. 2020. Learning continuous system dynamics from irregularly-sampled partial observations. In *NeurIPS*.
[27] Zijie Huang, Yizhou Sun, and Wei Wang. 2021. Coupled Graph ODE for Learning Interacting System Dynamics. In *KDD*.
[28] Ming Jin, Yu Zheng, Yuan-Fang Li, Siheng Chen, Bin Yang, and Shirui Pan. 2022. Multivariate time series forecasting with dynamic graph neural odes. *IEEE Transactions on Knowledge and Data Engineering* (2022).
[29] Patrick Kidger, Ricky TQ Chen, and Terry J Lyons. 2021. " Hey, that's not an ODE": Faster ODE Adjoints via Seminorms.. In *ICML*.
[30] Byungsoo Kim, Vinicius C Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. 2019. Deep fluids: A generative network for parameterized fluid simulations. In *Computer graphics forum*.
[31] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[32] Miltiadis Kofinas, Naveen Nagaraja, and Efstratios Gavves. 2021. Roto-translated local coordinate frames for interacting dynamical systems. *NeurIPS* (2021).
[33] Lingkai Kong, Jimeng Sun, and Chao Zhang. 2020. Sde-net: Equipping deep neural networks with uncertainty estimates. In *ICML*.

[34] Shiyong Lan, Yitong Ma, Weikang Huang, Wenwu Wang, Hongyu Yang, and Pyang Li. 2022. Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In *ICML*.

[35] Yan Li, Xinjiang Lu, Haoyi Xiong, Jian Tang, Jiantao Su, Bo Jin, and Dejing Dou. 2023. Towards long-term time-series forecasting: Feature, pattern, and distribution. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1611–1624.

[36] Chumeng Liang, Zherui Huang, Yicheng Liu, Zhanyu Liu, Guanjie Zheng, Hanyuan Shi, Kan Wu, Yuhao Du, Fuliang Li, and Zhenhui Jessie Li. 2023. CBLab: Supporting the Training of Large-scale Traffic Control Policies with Scalable Traffic Simulation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4449–4460.

[37] Yaqian Liang, Shanshan Zhao, Baosheng Yu, Jing Zhang, and Fazhi He. 2022. Meshmae: Masked autoencoders for 3d mesh data analysis. In *European Conference on Computer Vision*.

[38] Mario Lino, Chris Cantwell, Anil A Bharath, and Stathi Fotiadis. 2021. Simulating continuum mechanics with multi-scale graph neural networks. *arXiv preprint arXiv:2106.04900* (2021).

[39] Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. 2024. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems* 36 (2024).

[40] Lu Liu, Jie Wu, and Shunying Ji. 2022. DEM–SPH coupling method for the interaction between irregularly shaped granular materials and fluids. *Powder Technology* (2022).

[41] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *NeurIPS*.

[42] Andreas Look, Melih Kandemir, Barbara Rakitsch, and Jan Peters. 2023. Cheap and Deterministic Inference for Deep State-Space Models of Interacting Dynamical Systems. *arXiv preprint arXiv:2305.01773* (2023).

[43] Linhao Luo, Gholamreza Haffari, and Shirui Pan. 2023. Graph sequential neural ode process for link prediction on dynamic and sparse graphs. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 778–786.

[44] Xiao Luo, Jingyang Yuan, Zijie Huang, Huiyu Jiang, Yifang Qin, Wei Ju, Ming Zhang, and Yizhou Sun. 2023. HOPE: High-order Graph ODE For Modeling Interacting Dynamics. In *ICML*. 23124–23139.

[45] Yingtao Luo, Qiang Liu, Yuntian Chen, Wenbo Hu, Tian Tian, and Jun Zhu. 2023. Physics-Guided Discovery of Highly Nonlinear Parametric Partial Differential Equations. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1595–1607.

[46] Priyanka Malhan and Monika Mittal. 2022. A novel ensemble model for long-term forecasting of wind and hydro power generation. *Energy Conversion and Management* (2022).

[47] Karttikeya Mangalam, Yang An, Harshayu Girase, and Jitendra Malik. 2021. From goals, waypoints & paths to long term human trajectory forecasting. In *CVPR*.

[48] Steffen Marburg and Bodo Nolte. 2008. *Computational acoustics of noise propagation in fluids: finite and boundary element methods*. Vol. 578. Springer.

[49] Romit Maulik, Bethany Lusch, and Prasanna Balaprakash. 2021. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids* (2021).

[50] Michinari Momma, Chaosheng Dong, and Jia Liu. 2022. A multi-objective/multi-task learning framework induced by pareto stationarity. In *ICML*.

[51] COMSOL Multiphysics. 1998. Introduction to comsol multiphysics. *COMSOL Multiphysics, Burlington, MA, accessed Feb* (1998).

[52] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730* (2022).

[53] Alexander Norcliffe, Cristian Bodnar, Ben Day, Jacob Moss, and Pietro Liò. 2021. Neural ode processes. *arXiv preprint arXiv:2103.12413* (2021).

[54] D Pardo, L Demkowicz, C Torres-Verdin, and M Paszynski. 2007. A self-adaptive goal-oriented hp-finite element method with electromagnetic applications. Part II: Electrodynamics. *Computer Methods in Applied Mechanics and Engineering* (2007).

[55] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. 2019. Relational knowledge distillation. In *CVPR*.

[56] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).

[57] Jiang-Zhou Peng, Siheng Chen, Nadine Aubry, Zhihua Chen, and Wei-Tao Wu. 2020. Unsteady reduced-order model of flow over cylinders based on convolutional and deconvolutional neural network structure. *Physics of Fluids* (2020).

[58] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. 2021. Learning mesh-based simulation with graph networks. In *ICLR*.

[59] Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. 2019. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532* (2019).

[60] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. 2020. Scalable differentiable physics for learning and control. *arXiv preprint arXiv:2007.02168* (2020).

[61] Chengping Rao, Pu Ren, Qi Wang, Oral Buyukozturk, Hao Sun, and Yang Liu. 2023. Encoding physics to learn reaction–diffusion processes. *Nature Machine Intelligence* (2023).

[62] Zhengyong Ren and Jingtian Tang. 2010. 3D direct current resistivity modeling with unstructured mesh by adaptive finite-element method. *Geophysics* (2010).

[63] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference*.

[64] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* (2020).

[65] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. 2020. Learning to simulate complex physics with graph networks. In *ICML*.

[66] Johannes Schmidt-Hieber. 2020. Nonparametric regression using deep neural networks with ReLU activation function. *The Annals of Statistics* 48, 4 (2020).

[67] Yidi Shao, Chen Change Loy, and Bo Dai. 2022. Transformer with Implicit Edges for Particle-Based Physics Simulation. In *ECCV*.

[68] Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. 2022. Do differentiable simulators give better policy gradients?. In *ICML*.

[69] Makoto Takamoto, Francesco Alesiani, and Mathias Niepert. 2023. Learning Neural PDE Solvers with Parameter-Guided Channel Attention. *arXiv preprint arXiv:2304.14118* (2023).

[70] Binh Tang and David S Matteson. 2021. Probabilistic transformer for time series analysis. In *NeurIPS*.

[71] Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. 2017. Accelerating eulerian fluid simulation with convolutional networks. In *ICML*.

[72] Arnaud Vadeboncoeur, Ieva Kazlauskaite, Yanni Papandreou, Fehmi Cirak, Mark Girolami, and Omer Deniz Akyildiz. 2023. Random grid neural processes for parametric partial differential equations. In *ICML*.

[73] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

[74] Qixuan Wang and Hao Wu. 2021. Mathematical modeling of chemotaxis guided amoeboid cell swimming. *Physical Biology* (2021).

[75] Shan Wang, J González-Cao, H Islam, M Gómez-Gesteira, and C Guedes Soares. 2022. Uncertainty estimation of mesh-free and mesh-based simulations of the dynamics of floaters. *Ocean Engineering* (2022).

[76] Junkang Wei, Siyuan Chen, Licheng Zong, Xin Gao, and Yu Li. 2022. Protein–RNA interaction prediction with deep learning: structure matters. *Briefings in bioinformatics* (2022).

[77] Dongxia Wu, Ruijia Niu, Matteo Chinazzi, Alessandro Vespignani, Yi-An Ma, and Rose Yu. 2023. Deep bayesian active learning for accelerating stochastic simulation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2559–2569.

[78] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *ICML*.

[79] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*.

[80] Tailin Wu, Qinchen Wang, Yinan Zhang, Rex Ying, Kaidi Cao, Rok Sosic, Ridwan Jalali, Hassan Hamam, Marko Maucec, and Jure Leskovec. 2022. Learning large-scale subsurface simulations with a hybrid graph network simulator. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4184–4194.

[81] Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. 2020. Continuous graph neural networks. In *ICML*. 10432–10441.

[82] Jiayang Xu, Aniruddhe Pradhan, and Karthikeyan Duraisamy. 2021. Conditionally parameterized, discretization-aware neural networks for mesh-based modeling of physical systems. In *NeurIPS*.

[83] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *ICLR*.

[84] Fangkai Yang, Jue Zhang, Lu Wang, Bo Qiao, Di Weng, Xiaoting Qin, Gregory Weber, Durgesh Nandini Das, Srinivasan Rakhunathan, Ranganathan Srikanth, et al. 2023. Contextual Self-attentive Temporal Point Process for Physical Decommissioning Prediction of Cloud Assets. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5372–5381.

[85] Haoran Yang, Hongxu Chen, Shirui Pan, Lin Li, Philip S Yu, and Guandong Xu. 2022. Dual Space Graph Contrastive Learning. In *WWW*.

[86] Shan Yang, Junbang Liang, and Ming C Lin. 2017. Learning-based cloth material recovery from video. In *ICCV*.

[87] Çağatay Yıldız, Melih Kandemir, and Barbara Rakitsch. 2022. Learning interacting dynamical systems with latent Gaussian process ODEs. *NeurIPS* (2022).

[88] Tehrim Yoon, Sumin Shin, and Eunho Yang. 2022. Learning Polymorphic Neural ODEs with Time-evolving Mixture. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[89] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *IJCAI*.

[90] Guozhen Zhang, Zihan Yu, Depeng Jin, and Yong Li. 2022. Physics-infused machine learning for crowd simulation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2439–2449.
[91] Yanfu Zhang, Shangqian Gao, Jian Pei, and Heng Huang. 2022. Improving social network embedding via new second-order continuous graph neural networks. In *KDD*. 2515–2523.
[92] Yuchen Zhang, Mingsheng Long, Kaiyuan Chen, Lanxiang Xing, Ronghua Jin, Michael I Jordan, and Jianmin Wang. 2023. Skilful nowcasting of extreme precipitation with NowcastNet. *Nature* (2023).
[93] Yao Zhang, Yun Xiong, Dongsheng Li, Caihua Shan, Kan Ren, and Yangyong Zhu. 2021. CoPE: Modeling Continuous Propagation and Evolution on Interaction Graph. In *CIKM*.
[94] Jingyu Zhao, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, and Guangjian Tian. 2020. Do RNN and LSTM have long memory?. In *ICML*.
[95] Tian Zhou, Ziqing Ma, Qingsong Wen, Liang Sun, Tao Yao, Wotao Yin, Rong Jin, et al. 2022. Film: Frequency improved legendre memory model for long-term time series forecasting. *NeurIPS* (2022).
[96] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *ICML*.
[97] Qunxi Zhu, Yao Guo, and Wei Lin. 2021. Neural delay differential equations. *arXiv preprint arXiv:2102.10801* (2021).

A PROOF OF LEMMA 1

To begin with, we introduce a theorem from [10].

Theorem 3. *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:

$$\|f(t, x, z)\| \leq K(t)(1 + \|x\|)(1 + \|z\|), \tag{16}$$

(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:

$$\|f(t, x_1, z) - f(t, x_2, z)\| \leq L_U(t)(1 + \|z\|)\|x_1 - x_2\|. \tag{17}$$

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

$$\begin{cases} x'(t) = f(t, x(t), x(t-\tau)), & t \in [0, (n+1)\tau] \\ x(t) = x_0, & t \in [-\tau, 0) \end{cases}, \tag{18}$$

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

$$\sup_{0 \leq t \leq \tau} \left\|\phi_j(t)\right\| \leq K_j, \tag{19}$$

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

$$\begin{aligned} K_j = &\left(1 + K_{j-1}\right)\left(1 + \|K\|_{L^1([j\tau,(j+1)\tau])}\right) \\ &\times \exp\left(\left(1 + K_{j-1}\right)\|K\|_{L^1([j\tau,(j+1)\tau])}\right). \end{aligned} \tag{20}$$

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

$$\boldsymbol{m}_1^t = \sigma(\boldsymbol{W}_0 \boldsymbol{y}^t + \boldsymbol{W}_1 \boldsymbol{y}^{t-1} + \boldsymbol{b}_1), \tag{21}$$

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:

$$\boldsymbol{m}_i^t = \sigma(\boldsymbol{W}_i \boldsymbol{m}_{i-1}^t + \boldsymbol{b}_i), \tag{22}$$

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:

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

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

Third, note the ReLU activation function satisfies:

$$|\sigma(x) - \sigma(y)| \leq |x - y|. \tag{23}$$

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

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

which means the assumption (A4) is satisfied.

Then, based on Theorem 3, we can claim that our graph ODE system Eqn. 8 has a unique absolutely continuous solution. □

B ALGORITHM

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
X̂^{t_0,ref} = X^{t_0}
while t < t_0 + T do
    // Foresight Step
    Feed X̂^{t,ref} into the graph ODE;
    Generate the predictions X̂^{t+1}, X̂^{t+1+r}, ··· , X̂^{t+1+rL−r} using Eqn. 9;
    // Refinement Step
    Generate the refined predictions X̂^{t+1,ref} from Eqn. 13;
    t ← t + 1
end while
```

Table 4: Details of four datasets. 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

C DATASET DETAILS

Four physics simulation benchmark datasets are utilized to evaluate our proposed FAIR and the compared baselines with details listed in Table 4.

CylinderFlow simulates the incompressible Navier-Stokes flow of water around a cylinder on a fixed 2D Eulerian mesh generated by COMSOL [51]. 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 [12]. 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 [4], including $1,400$ 2×2-Chinese character matrices. *InflatingFont* has more complicated structures, with at least twice node numbers, and 70 times edge numbers. We aim to predict the future position of every mesh node.

D BASELINE DETAILS

We compare our FAIR with a range of state-of-the-art methods, *i.e.*, GraphUNets [16], GNS [65], MeshGraphNet [58], MS-GNN-GRID [38], GMR-GMUS [21], GPMS [24], and BSMS-GNN [4]. Their details are elaborated as follows:

- GraphUNets [16] proposes new pooling and unpooling operations, which can be implemented in an UNet-style architecture [63]. We have replaced the original GCN layers into our message passing layers following [4].
- GNS [65] 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 [4].
- MeshGraphNet [58] 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.
- MS-GNN-GRID [38] is a well-known method of modeling the hierarchy with spatial structures, which introduces a novel multi-scale graph neural network model, designed to enhance and accelerate predictions in continuum mechanics simulations. Following [4, 38], MS-GNN-GRID is implemented using the finest edge encoder, an additional aggregation module for node and edge representations, and a node returning module.
- GMR-GMUS [21] is an extension to MeshGraphNet for mesh-based simulations. It first compresses mesh data into hidden embeddings and then utilizes a Transformer architecture to extract the spatio-temporal relationships with efficiency.
- GPMS [24] focuses on the region of interest in the mesh graph and then utilizes a two-stage framework with prior knowledge injected. The model is evaluated in two physical systems.
- BSMS-GNN [4] 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 [63]. 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 [56], PyG [14], and TorchDiffEq [29]. To ensure fairness, we implement our approach with a publicly available codebase by BSMS-GNN [4]. The random seed is fixed. 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 each epoch, aiming to enhance the adaptability of the model to process noisy inputs. Furthermore, to maintain fairness, we set the layer numbers for both encoder and decoder to 7, while MeshGraphNet and BSMS-GNN both have 15 layers. Each layer includes the encoder, processor, and decoder, all activated by ReLU and embedded within two hidden-layer MLPs. Residual connections are added in all MLPs, while a LayerNorm layer normalizes the outputs from MLPs apart from the nodal decoder.

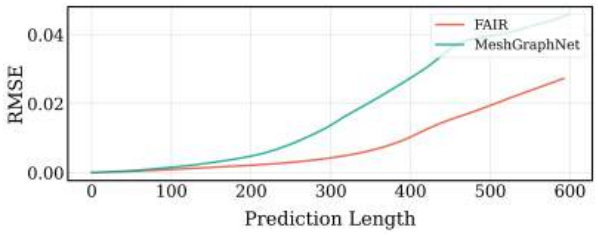

Figure 8: RMSE of our FAIR and MeshGraphNet with respect to different prediction lengths on *CylinderFlow*.

F PREDICTIONS AT DIFFERENT TIME STEPS

Figure 8 records the prediction errors of our proposed FAIR and MeshGraphNet at 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.