Article:
Dharma, D., Jimack, P.K. and Wang, H. (Accepted: 2026) Extending Graph Network Simulators for Particle-Based Simulations of Transient Flow Problems. Journal of Computational Science. ISSN: 1877-7503 (In Press)

Extending Graph Network Simulators for Particle-Based Simulations of Transient Flow Problems

Dody Dharma [a], Peter K. Jimack [b] and He Wang [c]

[a] School of EE and Informatics, Institut Teknologi Bandung, Jl. Ganesha 10, Bandung, 40132, Indonesia
[b] School of Computing, University of Leeds, Woodhouse Lane, Leeds, LS2 9JT, United Kingdom
[c] AI Centre, Department of Computer Science, University College London, Gower Street, London, WC1E 6BT, United Kingdom

ARTICLE INFO

Keywords:
fluid dynamics
surrogate model
graph network simulator
self-supervised learning
graph attention operator
vorticity conservation

ABSTRACT

We extend the Graph Network Simulator (GNS) framework for transient flow problems by treating its processor layer as a tunable design axis. On a common training harness — a six-component physics-informed loss covering acceleration, velocity, position, density, divergence, and vorticity, with dual acceleration/velocity prediction and distributed training with mini-batch noise injection — we evaluate two complementary processor instantiations against vanilla GNS and a non-graph external baseline (the full SFBC pipeline): GNS with the Self-Supervised Graph Attention Operator (GNS-SSGAT), and GNS with a Fourier Basis Convolution layer adapted from SFBC (GNS-FBC). Across two-dimensional lid-driven cavity and curl-noise benchmarks we find that processor choice produces systematically different physical-fidelity profiles, with each instantiation dominating in a different physics regime: GNS-SSGAT achieves the lowest vorticity-field MSE and the highest Pearson correlation on a Material-Point-Method (MPM) semi-compressible regime, while GNS-FBC dominates vorticity-field MSE, Pearson correlation, KL divergence, and long-rollout particle-distribution stability on a harder FLIP/APIC incompressible-liquid regime. Both clearly outperform vanilla GNS and the external SFBC baseline on their respective target regimes, and together they characterise the processor layer as a useful design axis for accurate and stable surrogate modelling of transient flow problems.

1. Introduction

Numerical simulation of transient flow problems presents significant computational challenges, particularly when modeling complex fluid dynamics involving vortex formation, turbulence, and multi-phase interactions. Traditional Eulerian methods, which discretize the fluid domain on fixed grids, often struggle with large deformations, free surfaces, and material interfaces. These limitations have motivated the development of Lagrangian particle-based methods, which track material points as they move through space, offering natural advantages for handling large deformations and complex geometries.

The Material Point Method (MPM) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14] represents a prominent Lagrangian approach for simulating the flow of fluids and soft solids, where the continuum is discretized into a collection of material points that carry mass, momentum, and other physical properties. In MPM, particles move through a background grid, allowing for efficient computation of spatial derivatives and inter-particle interactions. This particle-based framework naturally handles large deformations, material separation, and complex boundary conditions, making it particularly well-suited for transient flow problems. However, accurate simulation of such systems requires careful preservation of fundamental physical invariants, including conservation of mass, momentum, energy, and vorticity—the local rotational motion of fluid that characterizes vortex structures and turbulent behavior.

The preservation of energy and vorticity is particularly critical for long-term simulation accuracy. Energy conservation ensures that the total kinetic and potential energy of the system remains consistent over time, preventing artificial dissipation or amplification that could lead to unphysical behavior. Vorticity conservation, on the other hand, is essential for maintaining rotational dynamics, vortex structures, and the cascade of energy in turbulent flows. In Lagrangian particle methods, these conservation properties must be maintained through careful numerical discretization and time integration schemes. However, traditional MPM simulations can be computationally expensive,

*Corresponding author
dody.dharma@itb.ac.id (D. Dharma); p.k.jimack@leeds.ac.uk (P.K. Jimack); he_wang@ucl.ac.uk (H. Wang)
ORCID(s): 0000-0003-1022-9346 (D. Dharma); 0000-0001-9463-7595 (P.K. Jimack); 0000-0002-2281-5679 (H. Wang)

especially for high-resolution problems requiring many particles and small time steps, motivating the development of efficient surrogate models.

Building upon our previous work [15], where we introduced an end-to-end surrogate modeling approach for particle-based continuum simulations using recurrent neural networks (RNNs), we demonstrated improved efficiency and accuracy compared to traditional RNN approaches such as LSTM and GRU over extended rollouts. That work established the feasibility of using deep learning models as cost-effective surrogates for MPM simulations. Nevertheless, a notable limitation identified in that study—and indeed in many prior surrogate models—is their inability to focus adaptively on the most critical local interactions between particles: interactions that are crucial for faithfully simulating local properties such as vorticity, and for maintaining long-term dynamics and conservation properties.

To address these challenges, we present extensions of the Graph Network Simulator (GNS) [16] framework that recast its processor layer as a tunable design axis for surrogate modelling of particle-based flows. We instantiate the axis with two complementary substitutions inside a common training harness: (i) the Self-Supervised Graph Attention Operator (SuperGAT) [17], which dynamically learns the importance of neighbouring particles, and (ii) a Fourier Basis Convolution (FBC) layer adapted from SFBC [18] into the GNS processor slot. The two substitutions target different physical-fidelity axes: SuperGAT improves vorticity magnitude reproduction, while FBC improves vorticity spatial correlation and long-rollout particle-distribution stability. Together, evaluated against vanilla GNS and the full SFBC pipeline as a non-graph external baseline, they characterise the processor layer as a useful design axis for accurate and stable surrogate modelling of transient flow problems.

Our two-study evaluation shows that GNS-SSGAT and GNS-FBC each dominate the design axis on their target physics regime. On the MPM semi-compressible regime (Sections 6.2 and 6.3) GNS-SSGAT achieves the lowest vorticity-field MSE, the highest Pearson correlation, the smallest KL divergence, and the strongest kinetic-energy conservation among six processor families; on the harder FLIP/APIC incompressible-liquid regime (Section 6.4) GNS-FBC achieves the lowest vorticity-field MSE, the highest Pearson correlation, and the best long-rollout particle-distribution stability among the GNS-family variants and the external SFBC pipeline. Both clearly outperform vanilla GNS on their respective regimes. These results underscore the potential of advanced graph-based neural operators for stable and physically accurate surrogate modeling in complex fluid dynamics, paving the way for further research and deployment in real-world simulation tasks.

1.1. Contributions

The main contributions of this work, ordered from the most general methodological claim to the most specific empirical evidence, are:

- **Processor Layer as a Tunable Design Axis:** We frame the processor sub-module of the Graph Network Simulator architecture (encoder → processor → decoder) as a deliberate design axis for surrogate modelling of particle-based flows, and characterise it through a controlled four-variant ablation on a common training harness.
- **Two Complementary Processor Instantiations Inside the GNS Framework:** We evaluate two strong processor families inside the GNS framework on the same training data, loss, and budget: (i) Graph Attention via the Self-Supervised Graph Attention Operator (SuperGAT) [17], and (ii) Fourier Basis Convolution adapted from SFBC [18]. They target complementary physical-fidelity axes: GNS-SSGAT minimises vorticity-field MSE; GNS-FBC dominates Pearson correlation of the vorticity field and maintains long-rollout particle-distribution stability.
- **Common Comparison Harness:** We contribute a six-component physics-informed loss (acceleration, velocity, position, density, divergence, vorticity), a dual acceleration/ velocity prediction head, and a distributed training procedure with mini-batch noise injection. The harness is the instrument that makes processor-layer ablations fair — the only varying axis between variants is the processor.
- **Two Complementary Ablations Across Two Physics Regimes:** We report (i) a six-processor preliminary ablation on the MPM semi-compressible regime (Sections 6.2 and 6.3) comparing GNS-SSGAT against

vanilla GNS, GATConv, GATv2Conv, PointNet++, and Graph-Transformer baselines on two transient-flow benchmarks (2D lid-driven cavity and curl-noise initialised flow); and (ii) a controlled three-variant ablation on the harder FLIP/APIC incompressible-liquid regime (Section 6.4) comparing GNS-FBC against vanilla GNS and the full SFBC pipeline as a non-graph external baseline. Both studies measure absolute (vorticity-field MSE) and pattern (Pearson correlation of the vorticity field, KL divergence of the vorticity distribution, and divergence-of-velocity) metrics, and the FLIP/APIC study additionally includes cost (parameter count, throughput) as part of the profile. The two studies are kept in separate subsections because the simulator, particle count, and training budget differ across regimes.

1.2. Paper Organization

The remainder of this paper is organized around the design-axis claim. Section 2 reviews related work on graph-based simulators, vorticity in learning-based methods, and the two processor families that instantiate the axis (graph-attention mechanisms and continuous basis-function convolutions). Section 3 provides essential background on vorticity and angular-momentum conservation. Section 5 describes our method: the shared GNS architectural template (Section 5.1), the two processor instantiations – SuperGAT (Section 5.2) and Fourier Basis Convolution (Section 5.3) – the extended six-component loss (Section 5.5), and the shared training strategy. Section 4 introduces the evaluation metrics. Section 6 presents experiments on two benchmark scenarios under two simulator regimes: MPM results (Section 6.2, Section 6.3, the preliminary six-processor exploration that motivated the present study) and the formal four-variant FLIP/APIC ablation (Section 6.4). Section 7 discusses key findings, the mechanism behind each processor's strengths, and limitations. Section 8 concludes the paper and outlines future research directions. Additional mathematical derivations and implementation details are provided in the appendices.

2. Related Work

2.1. Graph Network-Based Simulators

Graph networks offer a powerful and flexible framework for modeling physical dynamics in Lagrangian representations, where particles are treated as nodes and edges encode their interactions. The framework proposed by [16], termed Graph Network Simulator (GNS), represents the state of a physical system using particles as nodes in a graph and computes dynamics via learned message-passing. Their results demonstrate the model's ability to generalize from single-timestep predictions with thousands of particles during training to different initial conditions and thousands of timesteps at test time. However, while the model mitigates error accumulation by corrupting training data with noise, it does not address either vorticity or attention explicitly.

Similarly, [19] propose a method that uses continuous convolutions to process sets of moving particles describing fluids in space and time without building an explicit graph structure. This method generalizes to arbitrary collision geometries and can be used for inverse problems, though it also lacks an explicit mechanism for vorticity conservation. [20] take a step further by guaranteeing the conservation of linear momentum in learned physics simulations through antisymmetrical continuous convolutional layers. While their method substantially increases the physical accuracy of the learned simulator, it focuses on linear momentum conservation and does not provide explicit measures for vorticity or attention.

The key advantages of graph-based representations include: (1) handling irregular and dynamic geometries without regular grid constraints, (2) inherent rotational invariance where relative positions matter more than absolute positions, (3) efficient modeling of local particle interactions through sparse connectivity, and (4) scalability to large particle systems by considering only relevant local interactions rather than exhaustive global ones.

2.2. Vorticity in Learning-Based Methods

Approaches aiming to address vorticity often operate within Eulerian frameworks or on regular grid data, rather than particle representations in continuous space. For example, [21] introduce NeuroFluid, a method that infers fluid state transitions from sequential visual observations of the fluid surface. Their approach involves a particle-driven neural renderer and a particle transition model optimized to reduce discrepancies between rendered and observed images, though it still lacks a direct focus on vorticity conservation.

[22] introduce the Neural Vortex Method, combining the Discrete Vortex Method with neural networks. This method relies on ground truth velocity sequences of vortex particles to represent fluid dynamics, which is a step towards

integrating vorticity into neural models. [23] propose the Differentiable Vortex Particle (DVP) Method that learns vortex particle dynamics from regular grid data using single videos without needing ground truth velocity, providing a novel way to infer and predict fluid dynamics including vorticity from limited observational data.

These methods demonstrate both the gaps and the recent trend towards incorporating vorticity into deep learning models for fluid dynamics, but challenges remain in effectively integrating this with Lagrangian representations.

2.3. Graph Attention Mechanisms

Various graph neural network architectures have been proposed to enhance the modeling of particle interactions. In a preliminary exploration (the MPM benchmarks of Sections 6.2 and 6.3) we evaluated several of these processor families as candidate plug-ins for the GNS processor slot; we summarise them here so the reader can place the two strong candidates we retain for the formal four-variant FLIP/APIC ablation (Section 6.4) — SuperGAT and Fourier Basis Convolution — in context. **PointNetConv (PointNet++)** [24] extends the original PointNet model [25] with hierarchical feature learning, capturing both local and global geometric features from point clouds. Its ability to learn multi-scale features proves advantageous in fluid simulations where particle distributions vary significantly. **Graph Transformer** [26] integrates self-attention mechanisms into graph-based data, enabling the modeling of long-range dependencies within a graph. [27] use Transformer with Implicit Edges (TIE) to capture the rich semantics of particle interactions in an edge-free manner, which is particularly useful for capturing global flow structures. **GATConv** [28] is the original Graph Attention Network formulation, introducing attention-based mechanisms into GNNs. By computing attention scores between a node and its neighbors, GATConv can preferentially highlight the most relevant particle interactions. **GATv2Conv** [29] is an enhanced version of GAT that refines the attention mechanism to improve expressive power and stability. By dynamically weighting neighboring nodes' contributions, GATv2Conv excels in capturing complex relationships within a fluid system. Our work builds upon these approaches by incorporating the Self-Supervised Graph Attention Operator (SuperGAT) [17], which uses self-supervised learning to dynamically adjust the importance of neighboring nodes, enabling better focus on critical local interactions essential for vorticity conservation.

Beyond attention-based message passing, a parallel line of work uses continuous basis-function convolutions as the processor layer for Lagrangian particle dynamics. *Continuous Convolution* (CConv) [19] introduced this idea for SPH-like fluid simulation, using radial-basis spatial filters parameterised by a small number of control points. *Symmetric Basis Convolutions* (SFBC) [18] refined this line, reporting strong performance on incompressible-SPH benchmarks with markedly fewer parameters than graph-based methods on their 1D test problem. SFBC's 2D evaluation, however, does not compare against GNS, and the question of whether SFBC's basis-convolution layer transfers when used as a processor *inside* the GNS training framework — with its physics-informed multi-component loss, distributed training, and noise injection — remained open. We answer that question in this paper as one of two processor instantiations along the design axis we characterise (see Section 5.3).

3. Background

3.1. Angular Momentum and Vorticity

The conservation of angular momentum is a fundamental principle in continuum mechanics, describing how the rotational motion of a system remains constant in the absence of external torques [30]. For an incompressible fluid, the angular momentum $\mathbf{L}$ is given by:

$$\mathbf{L} = \int_{\Omega} \mathbf{r} \times (\rho \mathbf{u})\, dV \tag{1}$$

where $\mathbf{r}$ is the position vector, ρ is the fluid density, $\mathbf{u}$ is the velocity field, and Ω is the fluid domain. In the absence of external torques, $\frac{d\mathbf{L}}{dt} = 0$, ensuring angular momentum conservation.

Vorticity ($\boldsymbol{\omega}$) characterizes the local rotational motion of fluid particles and is mathematically defined as the curl of the velocity field:

$$\boldsymbol{\omega} = \nabla \times \mathbf{u} \tag{2}$$

In three dimensions, the vorticity vector is:

$$\boldsymbol{\omega} = \left(\frac{\partial u_z}{\partial y} - \frac{\partial u_y}{\partial z} \right) \hat{\imath} + \left(\frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x} \right) \hat{\jmath} + \left(\frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y} \right) \hat{k} \tag{3}$$

Understanding vorticity is crucial for analyzing fluid phenomena including vortex formation, eddies, and turbulent flows.

3.2. Energy Conservation

In Lagrangian fluid simulation, the total kinetic energy is:

$$E_{k,\text{total}} = \sum_{i=1}^{N} \frac{1}{2} m_i v_i^2 \tag{4}$$

In an ideal, inviscid fluid with no external forces, kinetic energy should be conserved ($\frac{dE_{k,\text{total}}}{dt} = 0$). However, numerical dissipation, external forces, and boundary conditions can affect this conservation. Therefore, monitoring energy conservation is crucial for validating the accuracy and stability of Lagrangian simulations.

4. Evaluation Metrics for Vorticity Distribution

Since the fluid domain is discretized as "particles" in a Lagrangian representation, for accurate evaluation, we interpolate all the particles' quantities to a fixed-sized uniform grid to obtain the vorticity values across the entire domain. This section discusses three methods for comparing the distribution of the vorticity: Pearson correlation, residual analysis, and Kullback-Leibler (KL) divergence. Each method has its advantages and weaknesses, which are described below.

4.1. Pearson Correlation

The Pearson correlation is a statistical tool used to assess the linear dependency between two datasets [31, 32]. This coefficient is computed to determine the similarity between the actual values and the predictions of the model. It gives a measurement of the linear relationship between two datasets, displaying a value within the range of -1 to 1. A value of 1 denotes a perfect positive linear relationship, 0 indicates no linear relationship, and -1 represents a perfect negative linear relationship. The Pearson correlation coefficient r is expressed as follows:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{5}$$

where x_i and y_i are the vorticity values for the ground truth and the model, respectively, and $\overline{x}$ and $\overline{y}$ are their mean values. Pearson correlation is advantageous because it provides a simple and intuitive interpretation, measuring the strength and direction of the linear relationship. However, it only captures linear relationships and may not reflect non-linear similarities. Additionally, it is sensitive to outliers, which can distort the correlation coefficient.

4.2. Residual Analysis

Residual analysis involves examining the differences (residuals) between ground truth and model predictions [33, 34]. The residual r_i for the i-th data point is defined as:

$$r_i = y_i - x_i \tag{6}$$

where x_i is the ground truth vorticity and y_i is the predicted vorticity of the model. Residual analysis provides insights into the distribution and magnitude of prediction errors, helping identify systematic biases and trends in the model's predictions. However, it does not provide a single summary statistic, making comparisons more complex. Moreover, residuals can be affected by the scale of the data.

4.3. Kullback-Leibler (KL) Divergence

KL divergence measures the difference between two probability distributions [35]. For vorticity data, we first normalize the vorticity values to form probability distributions. The KL divergence $D_{KL}(P \parallel Q)$ is defined as:

$$D_{KL}(P \parallel Q) = \sum_{i=1}^{n} P_i \log\left(\frac{P_i}{Q_i}\right) \tag{7}$$

where P represents the normalized vorticity distribution of the ground truth data, Q represents the normalized vorticity distribution of the model predictions, and P_i and Q_i are the corresponding normalized vorticity values at each point

in the domain. In this context, "probabilities" refer to the normalized vorticity values. These values are treated as probabilities after normalization because they are non-negative and sum to one across the domain. This approach allows the use of probability distribution tools like KL divergence to compare the similarity between the ground truth and model-predicted vorticity distributions.

Normalization Process Normalization is crucial for calculating KL divergence as it converts vorticity values into a form that can be treated as probability distributions:

$$P_i = \frac{|\omega_i| + \epsilon}{\sum_{j=1}^{n}(|\omega_j| + \epsilon)} \tag{8}$$

where ω_i is the vorticity magnitude at a given point and ϵ is a small regularizing constant added to avoid zero values. The result is a set of normalized values P_i that sum to one, resembling a probability distribution.

Entropy Entropy, in the context of KL divergence, measures the unpredictability or randomness of a distribution. It quantifies the amount of "surprise" or information content in a probability distribution. When comparing two distributions using KL divergence, entropy helps determine how much one distribution diverges from another. Specifically, KL divergence can be seen as the additional entropy (or information loss) incurred when using the model's distribution Q to approximate the ground truth distribution P.

5. Method

5.1. Network Architecture: A Shared GNS Template

This subsection introduces the architectural template that every processor variant in this paper shares; the only varying axis between variants is the processor block. The template follows the *encoder → processor → decoder* design of the original GNS framework [16], with two adaptations targeted at vorticity-aware fluid simulation: (i) the processor slot is deliberately exposed as a tunable design axis, into which we plug either the Self-Supervised Graph Attention Operator (SuperGAT, Section 5.2) or the Fourier Basis Convolution layer (FBC, Section 5.3); and (ii) the loss function is extended (Section 5.5) to penalise six physically grounded components – acceleration, velocity, position, density, divergence, and vorticity – so that the processor-only ablation reported in Section 6.4 probes the processor's contribution under a controlled training harness.

The overall architecture is based on the work of [16], which uses Graph Networks (GNs) to simulate complex physical systems. In their work, the model simulates fluid dynamics by learning particle interactions over time using a graph-based structure. The vanilla GNS variant uses its original InteractionNetwork processor; in this paper we additionally evaluate two alternative processors that plug into the same encoder/decoder template – SuperGAT and FBC – to characterise the processor layer as a design axis.

The two processor instantiations target complementary physical-fidelity axes. The SuperGAT processor [17], detailed in Section 5.2, dynamically learns the importance of neighbouring particles using a self-supervised attention mechanism and improves vorticity-magnitude reproduction. The FBC processor (Section 5.3) uses a fixed Fourier basis to weight neighbour contributions by their relative position and improves vorticity-pattern correlation and long-rollout particle-distribution stability. Both processors are dropped into the same encoder/decoder template, the same six-component loss, and the same training schedule, so any observed differences in physical-fidelity profile are attributable to the processor block alone.

We build the architecture by taking advantage of the graph representation. In a graph, particles are represented as nodes, and interactions between particles are represented as edges. This structure allows for a natural and flexible representation of the complex relationships and interactions in a fluid system. The key advantages include:

- **Handling Irregular Geometries:** Unlike grid-based methods, graph representations are not restricted to regular grids and can handle irregular and dynamic geometries. This flexibility is essential for simulating real-world fluid systems where the particle distribution can be non-uniform and dynamic.
- **Rotational Invariance:** One significant advantage of graph representation is its ability to maintain rotational invariance. In fluid dynamics, the relative positions and interactions of particles are more important than their

absolute positions. Graph-based methods can inherently respect this property, ensuring that the simulation results are consistent regardless of the orientation of the particles.

- **Capturing Local Interactions:** Graphs can effectively model local interactions between particles, which are crucial for accurately simulating fluid dynamics. Each node (particle) can easily share information with its neighboring nodes (particles) through the edges, capturing the local dependencies and influences.
- **Scalability:** Graph-based methods can scale to large numbers of particles by leveraging sparse connectivity. Only the relevant local interactions need to be considered, reducing the computational complexity compared to methods that require global interactions.

Figure 1 illustrates the flow of data and operations through the shared GNS template. The centre block ("Particle Dynamics Stack") is the processor slot in which we plug different processor variants; in this figure the SuperGAT operator is shown as one such instantiation, and the basis-convolution alternative is illustrated in Figure 2 of Section 5.3.

Since the particle positions change during the simulation, the neighborhood configuration is dynamic. Therefore, at each time step, the process begins with a neighborhood search to identify local interactions, followed by the reconstruction of the graph incorporating updated node and edge features. These features serve as input to the network. The architecture consists of three main components: an encoder, a Particle Dynamics Stack, and a decoder.

- **Encoder:** The encoder processes the input node and edge features using two separate Multi-Layer Perceptrons (MLPs). The Node MLP encodes the node features, and the Edge MLP encodes the edge features. These encoded features are then fed into the Particle Dynamics Stack.
- **Particle Dynamics Stack (processor slot):** This is the core of the architecture, comprising a stack of ParticleNetwork layers. The stack is the tunable design axis of this paper: each layer applies a processor operator to update node features, and we report results for three processor choices — vanilla InteractionNetwork (GNS), SuperGAT (GNS-SSGAT, Section 5.2), and Fourier Basis Convolution (GNS-FBC, Section 5.3). In all three cases the stack iterates through multiple passes, refining the features with each pass.
- **Decoder:** The decoder takes the processed node features and applies a Node MLP to generate the output features. These outputs include physical quantities such as acceleration, velocity, position, and vorticity of the fluid particles.

The architecture is designed to conserve vorticity in Lagrangian fluid simulations by effectively capturing the interactions between particles through the graph-based representation and attention mechanism.

5.2. Self-Supervised Graph Attentional Operator

The self-supervised graph attentional operator introduced by [17] is defined as:

$$\mathbf{x}'_i = \alpha_{i,i}\mathbf{\Theta}\mathbf{x}_i + \sum_{j\in\mathcal{N}(i)} \alpha_{i,j}\mathbf{\Theta}\mathbf{x}_j, \tag{9}$$

where $\mathbf{x}'_i$ represents the updated feature vector for node i after applying the graph attention layer. The term $\alpha_{i,i}$ is the attention coefficient for the self-loop of node i, indicating how much attention node i gives to itself. The matrix $\mathbf{\Theta}$ is a learnable weight matrix shared across all nodes and their neighbors. The summation $\sum_{j\in\mathcal{N}(i)}$ is taken over all neighbors j of node i, and $\alpha_{i,j}$ is the attention coefficient for the edge between node i and its neighbor j. This equation is part of the graph attention mechanism, which updates the feature vector of a node by aggregating features from its neighbors and itself, weighted by attention coefficients. The self-loop term $\alpha_{i,i}\mathbf{\Theta}\mathbf{x}_i$ considers the contribution of node i's own features. The neighborhood aggregation term $\sum_{j\in\mathcal{N}(i)} \alpha_{i,j}\mathbf{\Theta}\mathbf{x}_j$ aggregates the features of all neighboring nodes j of node i. Each neighbor's feature vector $\mathbf{x}_j$ is transformed by the same weight matrix $\mathbf{\Theta}$, then weighted by the corresponding attention coefficient $\alpha_{i,j}$.

The two types of attention $\alpha_{i,j}^{\text{MX or SD}}$ are computed as:

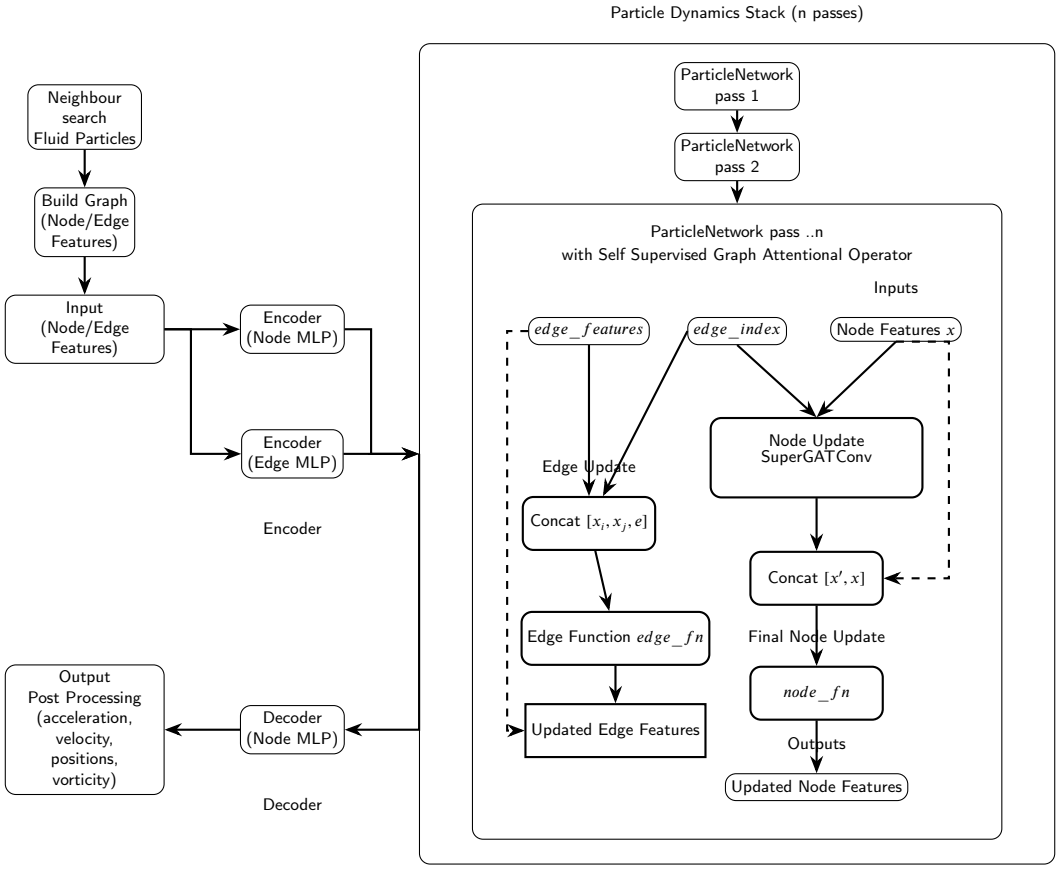


Figure 1: Shared GNS architecture template used by every processor variant in this paper. The encoder, decoder, and post-processor are identical across vanilla GNS, GNS-SSGAT, and GNS-FBC; the only varying axis is the centre block (Particle Dynamics Stack / processor slot). The figure illustrates the SuperGAT instantiation; the FBC instantiation is shown in Figure 2. The model builds upon the architecture of [16], with key modifications, including the substitution of the original convolutional layer with the Self-Supervised Graph Attention Operator (SuperGAT). This modification enables the network to better focus on local particle interactions, crucial for vorticity conservation. The Particle Dynamics Stack and SuperGATConv layers have been adapted to ensure that vorticity, angular momentum, and energy are better preserved.

$$
\begin{aligned}
\alpha_{i,j}^{\text{MX or SD}} &= \frac{\exp\left(\text{LeakyReLU}\left(e_{i,j}^{\text{MX or SD}}\right)\right)}{\sum_{k\in\mathcal{N}(i)\cup\{i\}}\exp\left(\text{LeakyReLU}\left(e_{i,k}^{\text{MX or SD}}\right)\right)} \\
e_{i,j}^{\text{MX}} &= \mathbf{a}^{\top}[\boldsymbol{\Theta}\mathbf{x}_i \,\|\, \boldsymbol{\Theta}\mathbf{x}_j]\cdot\sigma\left(\left(\boldsymbol{\Theta}\mathbf{x}_i\right)^{\top}\boldsymbol{\Theta}\mathbf{x}_j\right) \\
e_{i,j}^{\text{SD}} &= \frac{\left(\boldsymbol{\Theta}\mathbf{x}_i\right)^{\top}\boldsymbol{\Theta}\mathbf{x}_j}{\sqrt{d}}
\end{aligned}
\tag{10}
$$

where $\|$ denotes concatenation, $\mathbf{a}$ is a learnable vector, σ is the sigmoid function, and d is the feature dimension.

The self-supervised task is a link prediction using the attention values as input to predict the likelihood $\phi_{i,j}^{\text{MX or SD}}$ that an edge exists between nodes:

$$\phi_{i,j}^{\mathrm{MX}} = \sigma\left(\left(\boldsymbol{\Theta}\mathbf{x}_i\right)^{\top}\boldsymbol{\Theta}\mathbf{x}_j\right)$$
$$\phi_{i,j}^{\mathrm{SD}} = \sigma\left(\frac{\left(\boldsymbol{\Theta}\mathbf{x}_i\right)^{\top}\boldsymbol{\Theta}\mathbf{x}_j}{\sqrt{d}}\right) \tag{11}$$

Overall, these equations describe how the attention coefficients $\alpha_{i,j}^{\mathrm{MX\ or\ SD}}$ are computed, which determine the importance of the features of neighboring nodes j when updating the feature vector of node i. The two types of attention mechanisms, MX and SD, use different methods to compute the attention energies $e_{i,j}^{\mathrm{MX}}$ and $e_{i,j}^{\mathrm{SD}}$.

5.3. Alternative Processor Instantiation: Fourier Basis Convolution (FBC)

To probe the processor-layer design axis with a second, structurally distinct instantiation, we adapt the SFBC basis-convolution layer [18] as the processor block of the Graph Network Simulator. The encoder, decoder, training loss, noise injection, normalisation, and dataset are identical to the GNS-SSGAT variant described in the preceding subsection; the only architectural substitution is the processor block.

Input features (encoder side). For each particle i at simulation step t the encoder builds a node feature vector

$$\mathbf{h}_i^{(0)} \;=\; \left[\,\mathbf{v}_i^{t-5:t-1}\,;\,\mathbf{b}_i\,;\,\mathbf{e}_{\mathrm{type}}(\tau_i)\right] \;\in\; \mathbb{R}^{30}, \tag{12}$$

where $\mathbf{v}_i^{t-5:t-1} \in \mathbb{R}^{10}$ is the flattened five-step velocity history (each velocity normalised by the training-set statistics), $\mathbf{b}_i \in \mathbb{R}^4$ is the clipped distance from particle i to each of the four domain boundaries, and $\mathbf{e}_{\mathrm{type}}(\tau_i) \in \mathbb{R}^{16}$ is the learned embedding of the particle's type $\tau_i \in \{\text{fluid, wall, kinematic}\}$. Edges $(i,j) \in \mathcal{E}$ are formed by radius search at the connectivity radius R, and each edge carries the normalised relative displacement $\mathbf{r}_{ij}/R \in [-1,1]^2$. Unlike the original SFBC pipeline [18], which uses a *two-stream* architecture with separate `fluidFeatures` and `boundaryFeatures` tensors, GNS-FBC encodes wall particles in a single stream and lets the 16-dim type embedding in Eq. (12) carry the boundary semantics. This matches GNS-SSGAT's encoder exactly and is what enables a fair processor-only comparison.

Processor: a stack of $L = 4$ basis-convolution layers. Let $\mathcal{N}(i)$ denote the open neighbourhood of particle i (the center is excluded, consistent with the `ignoreCenter=True` setting of the SFBC reference implementation). A single basis-convolution layer ℓ updates node features by

$$\mathbf{m}_{i\leftarrow j}^{(\ell)} = \sum_{k_1=1}^{K}\sum_{k_2=1}^{K} \phi_{k_1}\!\left(\tilde{r}_{ij,1}\right)\phi_{k_2}\!\left(\tilde{r}_{ij,2}\right)\mathbf{W}_{k_1,k_2}^{(\ell)}\,\mathbf{h}_j^{(\ell)}, \tag{13}$$

$$\mathbf{h}_i^{(\ell+1)} = \sigma\!\Big(\mathbf{h}_i^{(\ell)} \;+\; \alpha\sum_{j\in\mathcal{N}(i)}\mathbf{m}_{i\leftarrow j}^{(\ell)}\Big), \tag{14}$$

where $\tilde{\mathbf{r}}_{ij} = \mathbf{r}_{ij}/R \in [-1,1]^2$, $\{\phi_k\}_{k=1}^{K}$ is a fixed family of K Fourier basis functions (the `ffourier` basis [18] with built-in symmetry across $\tilde{r} \mapsto -\tilde{r}$), and $\{\mathbf{W}_{k_1,k_2}^{(\ell)}\}$ are the per-basis learned linear maps that play the role of an MLP aggregator's weight matrix. We use $K = 8$ basis terms per spatial dimension, giving $K^2 = 64$ per-edge basis components in two dimensions. The non-linearity σ is ReLU, and the constant $\alpha = 1/128$ is the SFBC output scaling that stabilises gradient magnitudes. The latent channel width is set to $d_h = 128$ throughout layers $\ell = 1, \ldots, L-1$; the final layer $\ell = L$ projects to the $d_{\mathrm{out}} = 2$ acceleration output. Stacking $L = 4$ layers matches the depth used in the primary SFBC configurations [18].

Decoder and output. The final basis-convolution layer's output is taken directly as the normalised predicted acceleration $\hat{\mathbf{a}}_i^{*} \in \mathbb{R}^2$ at particle i. The decoder post-processor (identical to GNS-SSGAT) inverts the training-set normalisation, $\hat{\mathbf{a}}_i = \boldsymbol{\sigma}_a \odot \hat{\mathbf{a}}_i^{*} + \boldsymbol{\mu}_a$, and applies two semi-implicit Euler steps to integrate to the next-frame velocity and position. The six-component physics-informed loss of § *Extended Loss Function* is then computed on the integrated outputs; gradients flow back through the integration step into Eqs. (13)–(14).

Parameter budget and what differs from SFBC. With $L = 4$, $d_h = 128$, $K^2 = 64$, the basis-conv stack contains approximately 3.4 M parameters; together with the (shared) encoder and post-processor this gives 3.46 M total parameters for GNS-FBC, up from 1.10 M for GNS-SSGAT (Table 5). Two choices distinguish GNS-FBC from a faithful re-implementation of SFBC: (i) GNS-FBC uses GNS's 30-dim encoder (Eq. (12)) instead of SFBC's 4-dim SPH-derived feature set, and (ii) GNS-FBC predicts normalised acceleration trained against the six-component physics-informed loss, whereas SFBC predicts raw $\mathrm{d}\mathbf{u}/\mathrm{d}t$ against a direct MSE loss. We isolate the effect of the basis-convolution *layer* from the effect of those wider pipeline choices by retaining the SFBC pipeline as an external baseline alongside GNS-FBC; the comparison between the two is what disentangles "the layer" from "the surrounding framework". Figure 2 summarises the architecture: GNS-FBC is identical to GNS-SSGAT (Figure 1) except the processor block is replaced by the basis-convolution stack of Eqs. (13)–(14).

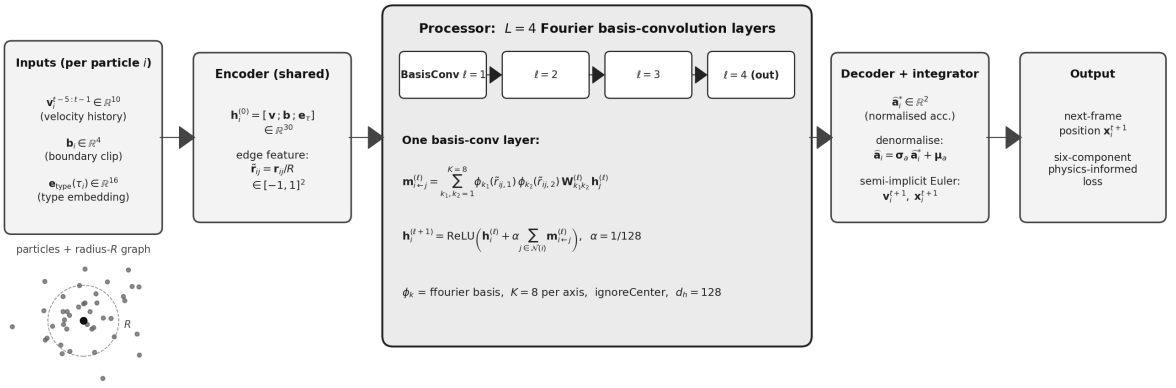


Figure 2: Architecture of the GNS-FBC variant. The encoder (Eq. 12) is shared with GNS and GNS-SSGAT: a 30-dim node feature built from velocity history, boundary distances, and a 16-dim type embedding, plus a 3-dim edge feature carrying the normalised relative displacement $\mathbf{r}_{ij}/R$. The processor (centre block) replaces the SuperGAT or InteractionNetwork message-passing with a stack of $L = 4$ Fourier-basis convolution layers, each summing $K^2 = 64$ basis-weighted neighbour contributions per particle (Eq. 13). The decoder produces a 2-dim normalised acceleration which is denormalised and integrated by the shared post-processor. The only architectural difference from GNS-SSGAT is the contents of the centre block; the encoder, decoder, loss, noise injection, and training schedule are identical.

5.4. Training Strategy

We employ distributed training across multiple GPUs using PyTorch's DistributedDataParallel. The model is optimized using the Adam optimizer [36] with an initial learning rate of 1×10^{-4}, which is decayed by a factor of 0.1 every 5 million steps to ensure gradual convergence. The optimizer's learning rate is scaled by the number of processes to maintain consistent training dynamics across different hardware configurations.

We implement a dual prediction approach, training the model to predict both acceleration and velocity, which increases versatility in capturing different dynamic behaviors. The model's performance is evaluated using rollouts spanning 90 timesteps, allowing systematic validation of long-term prediction stability. Both the model and training state are periodically saved to enable resumption after interruptions. Input features are normalized using precomputed statistics (mean and standard deviation) for positions, velocities, and accelerations to promote stable training and faster convergence. Detailed data preprocessing steps, normalization procedures, and initialization protocols are provided in Appendix B.

5.5. Extended Loss Function

The total loss is composed of several components, each targeting a specific aspect of the simulation. Let N denote the number of **kinematic** particles (i.e., particles that are free to move, not fixed boundary particles), and let D be the dimensionality (2D or 3D). The predicted and target positions of particle i are $\mathbf{p}_i^{\text{pred}}$ and $\mathbf{p}_i^{\text{target}}$, while $\mathbf{v}_i^{\text{pred}}$ and $\mathbf{v}_i^{\text{target}}$ are the corresponding velocities. Similarly, $\mathbf{a}_i^{\text{pred}}$ and $\mathbf{a}_i^{\text{target}}$ are the predicted and target accelerations. The predicted

density $\mathbf{d}^{\text{pred}}$ and target density $\mathbf{d}^{\text{target}}$ are computed on a grid, as are the predicted and target divergence fields $\nabla \cdot \mathbf{v}^{\text{pred}}$ and $\nabla \cdot \mathbf{v}^{\text{target}}$, and the predicted and target vorticities ω^{pred} and ω^{target}.

Throughout, $\| \cdot \|_p$ can represent either the L2 norm (mean squared error, MSE) or the L1 norm (mean absolute error, MAE), depending on the chosen training configuration. In our implementation, we select L2 (MSE) because it provides higher sensitivity to large errors, penalizing significant deviations more heavily.

Acceleration is a key factor in capturing dynamic changes. By penalizing the discrepancy between predicted and target accelerations, the model is encouraged to learn correct force and motion relationships:

$$\mathcal{L}_{\text{acc}} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{a}_i^{\text{pred}} - \mathbf{a}_i^{\text{target}}\|_p \tag{15}$$

Velocity is fundamental to fluid or particle transport. Minimizing this loss aligns flow speeds and directions with the ground truth:

$$\mathcal{L}_{\text{vel}} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{v}_i^{\text{pred}} - \mathbf{v}_i^{\text{target}}\|_p \tag{16}$$

Position error is often useful when tracking particle locations or ensuring that fluid front evolution is accurate:

$$\mathcal{L}_{\text{pos}} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{p}_i^{\text{pred}} - \mathbf{p}_i^{\text{target}}\|_p \tag{17}$$

After projecting the particle velocities and masses onto a grid, the predicted density $\mathbf{d}^{\text{pred}}$ and target density $\mathbf{d}^{\text{target}}$ are compared. The norm is computed over all grid points, which helps ensure that the model correctly represents the distribution of mass or particles in space:

$$\mathcal{L}_{\text{dens}} = \|\mathbf{d}^{\text{pred}} - \mathbf{d}^{\text{target}}\|_p \tag{18}$$

To compute divergence, central finite differences are applied on the velocity grid. Penalizing divergence encourages realistic mass conservation. For incompressible flows, minimizing divergence is especially crucial to prevent artificial sources or sinks in the fluid:

$$\mathcal{L}_{\text{div}} = \|\nabla \cdot \mathbf{v}^{\text{pred}} - \nabla \cdot \mathbf{v}^{\text{target}}\|_p \tag{19}$$

Vorticity quantifies the local spinning or swirling motion of a fluid, and capturing this rotational flow is essential for physically realistic simulations. We compute the vorticity field using central finite differences on the velocity grid:

$$\mathcal{L}_{\text{vor}} = \|\omega^{\text{pred}} - \omega^{\text{target}}\|_p. \tag{20}$$

Total Loss ($\mathcal{L}_{total}$)

$$\mathcal{L}_{\text{total}} = w_{\text{acc}} \cdot \mathcal{L}_{\text{acc}} + w_{\text{vel}} \cdot \mathcal{L}_{\text{vel}} + w_{\text{pos}} \cdot \mathcal{L}_{\text{pos}} + w_{\text{dens}} \cdot \mathcal{L}_{\text{dens}} + w_{\text{div}} \cdot \mathcal{L}_{\text{div}} + w_{\text{vor}} \cdot \mathcal{L}_{\text{vor}} \tag{21}$$

Here, $w_{\text{acc}}, w_{\text{vel}}, w_{\text{pos}}, w_{\text{dens}}, w_{\text{div}}, w_{\text{vor}}$ are user-defined weights that control the relative importance of each term.

Introducing a vorticity term in the loss function helps preserve these rotational features. Figure 3 highlights the importance of this component: the middle panel shows results from a baseline GNS model without vorticity loss, while the right panel includes the vorticity term. In the baseline model, the flow develops fewer or weaker rotational structures compared to the ground truth (left). By contrast, adding vorticity loss yields significantly better alignment with the reference flow, indicating that the model now captures swirling behaviors more accurately. This not only enhances visual fidelity but also leads to more physically consistent dynamics.

5.5.1. Defining and Adjusting Loss Weights

In practice, choosing weightings ($w_{\text{acc}}, w_{\text{vel}}, \dots$) determines how strongly each physical property influences the model updates. For example, increasing the vorticity weight (w_{vor}) penalizes errors in rotational flow more strongly and tends to produce more stable velocity fields, while increasing divergence weight (w_{div}) helps enforce near-incompressibility or mass conservation. Emphasizing velocity, acceleration, or position fosters more accurate dynamic predictions but can lessen the focus on global flow characteristics if not balanced properly.

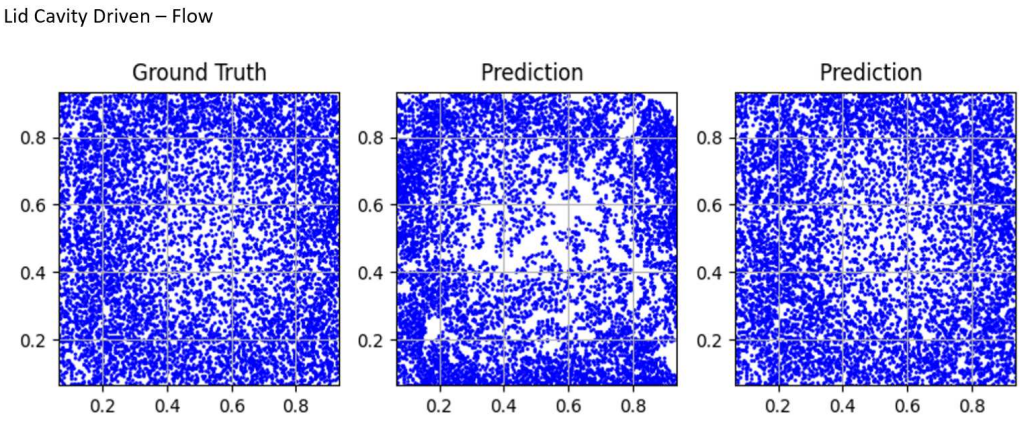


Figure 3: Comparison of a driven cavity flow simulation: (Left) Ground truth, (Middle) Prediction without vorticity loss, and (Right) Prediction with vorticity loss.

During an initial training stage, the *raw sizes* of each loss term (before weighting) may vary significantly. For example, one might see:

$$\mathcal{L}_{\text{acc}} \approx 0.7, \quad \mathcal{L}_{\text{vel}} \approx 0.1, \quad \mathcal{L}_{\text{pos}} \approx 3 \times 10^{-9}, \quad \mathcal{L}_{\text{dens}} \approx 0.03, \quad \mathcal{L}_{\text{div}} \approx 1.5, \quad \mathcal{L}_{\text{vor}} \approx 1.6.$$

Observing these magnitudes can help guide decisions on how heavily to weight each term. If, for instance, the vorticity or divergence losses are intrinsically higher, boosting those weights might be beneficial.

In many practical situations, particularly when there is limited time for hyperparameter tuning, assigning equal weights to every component (i.e., setting $w_{\text{acc}} = w_{\text{vel}} = \cdots = 1$) often provides a balanced solution.

During training, each component's raw loss is logged periodically. Below is an example from one training run at step 507, which showed an Acceleration Loss (3.566×10^{-2}), Velocity Loss (2.56×10^{-4}), Density Loss (1.708×10^{-2}), Position Loss (almost zero), Divergence Loss (1.209×10^{-1}), and Vorticity Loss (1.278×10^{-1}), summing to a total of 3.016×10^{-1}. As training continued, those losses generally decreased further; by a later snapshot, the overall Train Loss was about 6.443×10^{-1}, with the Acceleration, Velocity, Density, Position, Divergence, and Vorticity Losses each having also lowered in value.

Even with uniform weighting, the network can learn to balance all aspects—acceleration, velocity, density, position, divergence, and vorticity—sufficiently well for a physically consistent simulation. Adjusting individual weights more finely could further improve specific facets (e.g., mass conservation or rotational fidelity), but doing so might overemphasize one component at the expense of others if not done carefully. By combining acceleration, velocity, position, density, divergence, and vorticity losses into one framework—and either carefully or uniformly weighting each term—the model can capture both fine-grained particle motion and large-scale fluid properties. Tracking raw magnitudes of each loss component and logging their values during training helps ensure that no single aspect is neglected.

6. Experiments

6.1. Experimental Setup

This work spans two simulator regimes – the MPM semi-compressible setup that produced the results in Section 6.2 and Section 6.3, and the new FLIP/APIC incompressible-liquid setup that drives the processor-layer ablation in Section 6.4. We describe both setups below; the two regimes share the train/validation/test split and the data-preprocessing pipeline, and differ in the simulator, particle count, and physical-fidelity constraints.

6.1.1. MPM setup – weakly-compressible fluid (Section 6.2, Section 6.3)

We evaluate our approach on two challenging datasets: the 2D lid-driven cavity flow and curl-noise initialized flows. Both datasets are generated using the Material Point Method (MPM) using Taichi Lang software [37, 38]

Table 1
MPM Simulation Parameters used for both Lid-Driven Cavity and Curl-Noise datasets

Parameter	Value
Number of particles ($n_{\text{particles}}$)	8192
Number of grid cells (n_{grid})	40
Grid spacing (dx)	$\frac{1}{n_{\text{grid}}}$ (dimensionless)
Time step (dt)	2×10^{-3} seconds
Particle density (p_ρ)	0.0001 kg/m^3
Particle volume (p_{vol})	$(dx \times 0.5)^2$ m^2
Particle mass (p_{mass})	$p_{\text{vol}} \times p_\rho$ kg
Gravity	0.1 m/s^2
Boundary thickness (bound)	3 cells (dimensionless)
Modulus (E)	0.0005 Pa (Pascals)
Recording substeps	50 (dimensionless)
Output dimension	2 (spatial dimensions)
Maximum frames	600 (frames)
Maximum substep force (maxsubstepforce)	$150{,}000 + \text{random.randint}(1, 10) \times 25{,}000$ N

with approximately 8,192 particles projected onto a 40×40 grid. The simulation outputs are stored in HDF5 format containing particle positions, velocities, and accelerations over time.

For both experiments, we use the following training configuration: 48 scenarios for training, 6 for validation, and 6 for testing. Data preprocessing includes extracting positions from HDF5 files, computing velocities and accelerations using finite differences between consecutive frames, calculating normalization statistics (mean and standard deviation), and storing the processed data as NPZ files. All models are trained using the same hyperparameters, loss function, and optimization strategy to ensure fair comparison. The only difference between compared models lies in the graph convolutional operator used for node feature updates in the Particle Network layer.

6.1.2. FLIP/APIC setup – incompressible fluid (Section 6.4)

The controlled 3-variant processor-layer ablation of Section 6.4 requires a strictly harder physics regime so that the divergence and density terms of the six-component loss are actively enforced. We therefore additionally generate two datasets under a FLIP/APIC incompressible-liquid solver – the same two benchmark scenarios (2D lid-driven cavity and curl-noise initialised flow) reproduced under the new regime. Each scene is initialised with ∼10 800 particles (10 000 fluid + 824 wall particles) sampled on a 50×50 MAC background grid, with the same 48/6/6 train/validation/test split and the same data-preprocessing pipeline as the MPM setup. The two regimes thus differ along three axes simultaneously – simulator (MPM → FLIP/APIC), constitutive regime (semi-compressible → incompressible), and particle count (8 192 → 10 824) – which is why the MPM and FLIP/APIC numbers are not directly commensurate and are reported in separate subsections. Table 2 lists the solver parameters for both FLIP/APIC scenarios.

6.2. 2D Lid-Driven Cavity Flow (MPM regime)

c

Preliminary MPM exploration. The results reported in Section 6.2 and Section 6.3 were obtained on the MPM setup (Section 6.1.1, ∼8 192 particles, weakly-/semi-compressible material). They are the preliminary observation that motivated the broader processor-layer exploration; numbers in this subsection compare six processor families (InteractionNetwork, SuperGAT/GNS-SSGAT, GATConv, GATv2Conv, Graph-Transformer, PointNet++/CConv) under that earlier regime. The formal controlled study reported in Section 6.4 narrows to the two strongest candidates plus the full SFBC pipeline and re-runs them on the harder FLIP/APIC incompressible regime (Section 6.1.2). The two sets of numbers are therefore not directly commensurate and should be read as complementary evidence: this subsection shows that GNS-SSGAT was the strongest single processor in the MPM regime, while Section 6.4 shows that processor choice remains a tunable design axis when the regime is changed to an incompressible liquid.

Table 2
FLIP/APIC simulation parameters for the *incompressible-liquid* datasets used to drive the processor-layer ablation of Section 6.4. The two benchmark scenarios share all integrator parameters except the time step and the number of pressure-Jacobi iterations (lid-driven cavity *vs.* curl-noise, shown as LDC/CN). Source: the Taichi-Lang APIC reference implementation used to generate the training data.

Parameter	Value
Solver family	FLIP / APIC / ASFLIP hybrid
Number of fluid particles (n_{fluid})	10 000
Number of wall particles (n_{wall})	824
Total particles ($n_{\text{particles}}$)	10 824
Background grid ($n_{\text{grid}} \times n_{\text{grid}}$)	50×50 (MAC)
Grid spacing (h)	0.02 (dimensionless)
Particle radius (r)	0.005 ($= 0.25\,h$)
Particle rest density (ρ_0)	1000 kg/m^3 (water)
Time step (dt)	1/60 s (LDC) / 1/120 s (CN)
Pressure-Jacobi iterations / step	50 (LDC) / 100 (CN)
Particle-separation iterations / step	2
Pressure over-relaxation factor	1.9
ASFLIP velocity blend (α_v)	0.2
ASFLIP advection blend (α_x)	0.7
Compensate-drift density correction	enabled
Gravity	0 (shear/body-force driven)
Lid velocity U_{lid} (LDC only)	± 3.0 (dimensionless)
Curl-noise scale / strength (CN only)	1.0 / scenario-randomised
Spinup steps	600
Recorded frames per trajectory	300
Output dimension	2 (spatial dimensions)

6.2.1. Dataset

The 2D lid-driven cavity flow is a classic benchmark problem in computational fluid dynamics where fluid is enclosed in a square cavity with three stationary walls and a top wall (lid) that moves horizontally at constant velocity, inducing shear flow. The simulation domain is discretized into 8,192 particles with a 40×40 grid for computing velocity and pressure fields. Table1 details the simulation parameters.

The boundary conditions consist of a moving top wall with horizontal velocity $u = U_{\text{lid}}$ and zero vertical velocity ($v = 0$), while the other three walls are stationary with no-slip conditions ($u = v = 0$). The flow is governed by the incompressible Navier-Stokes equations (continuity and momentum conservation).

The flow regime is predominantly laminar with Reynolds numbers estimated between 0.2 and 40 (based on normalized parameters: $\rho = 1$, characteristic length $L \approx 1$, dynamic viscosity $\mu = 0.5$). A parallel force is applied at the top boundary to induce shear flow. Each simulation scenario spans 600 frames, yielding time-series data for positions, velocities, and accelerations. Simulation parameters are detailed in Table1. Figure4 illustrates the flow patterns, revealing the formation of primary vortices.

Representative visualisations of the training dataset are shown in Figure 4, displaying streamlines for ten representative initial conditions.

6.2.2. Results

In this section, we present both quantitative and qualitative results to provide a comprehensive evaluation of our vorticity prediction model. As a qualitative starting point, Figure 5 illustrates the velocity field predicted by our proposed model at frame 54, demonstrating the preservation of coherent flow structures over extended rollouts. For quantitative analysis, we used several metrics to evaluate the performance of our model, including the mean squared error (MSE), mean absolute error (MAE), and an Energy Conservation Plot. To further provide a visual understanding of the model's performance, we include several qualitative assessments (Predicted vs Actual Vorticity Plot, Residuals Plot, Temporal Evolution of Vorticity). These results will help to understand the performance and effectiveness of our model.

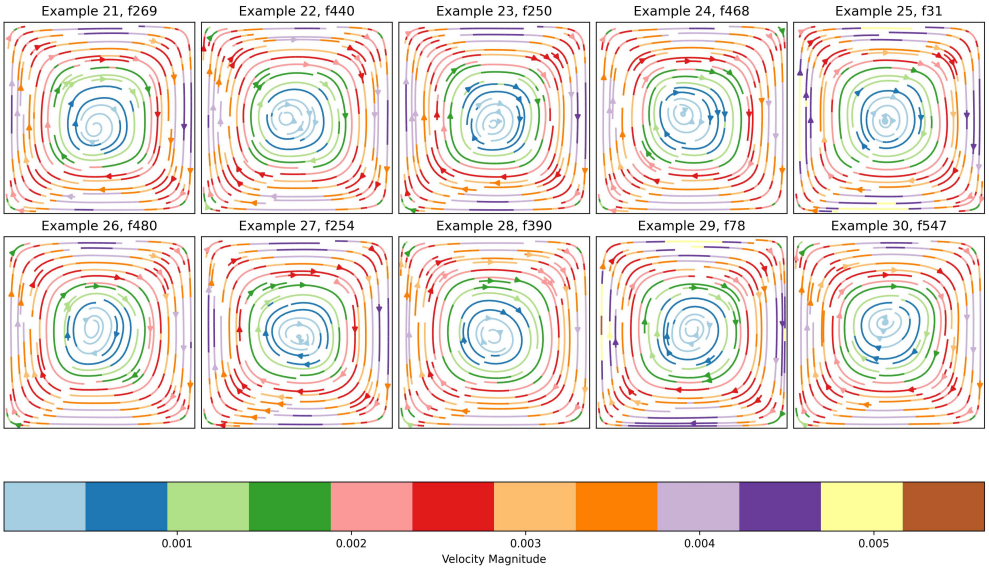


Figure 4: Visualization of 10 out of 48 training scenarios showing streamlines computed by mapping particle velocities to a regular grid using cubic spline interpolants. Color indicates velocity magnitude.

Parameter	Value
Boundary	[[0.065, 0.935], [0.065, 0.935]]
Sequence length	600
Neighbor radius	0.015
Dimension	2
Velocity mean	[-3.51e-05, -2.64e-06]
Acceleration mean	[8.54e-10, 1.38e-07]
Velocity std	[0.002191, 0.002195]
Acceleration std	[9.94e-05, 9.96e-05]

Table 3
Complete dataset statistics for 2D lid-driven cavity flow

6.2.3. Results on Various Test Sets

Figure 6 illustrates the performance of the GNS-SSGAT (_OURS) model in conserving vorticity over long-range predictions, measured by the mean squared error (MSE). The chart includes six curves representing six different test cases (T0 to T5).

The MSE for all test cases starts very low, close to 0, and gradually increases as the number of frames progresses. In particular, the MSE remains below 0.0002 for the first 40 frames in all test cases, indicating stable performance in short-term predictions. An important observation is that MSE grows almost linearly over time for each test scenario. This almost-linearity implies a steady rate of error build-up as the frame count rises. After surpassing 50 frames, the MSE continues to climb, emphasizing the challenge of achieving long-term prediction accuracy. However, the model keeps the MSE increase relatively controlled, with values remaining below 0.0004 up to frame 80. Each test case shows minor differences in MSE values, indicating some variations depending on the particular test scenario, yet the overarching pattern stays stable.

6.2.4. Benchmark: Long-Range Predictions Stability

In the comparative experiments presented in this section, we kept the overall model, loss function, and training methodology consistent across all models. The key difference lies in how the node features are updated in the Particle Network layer. For the original GNS model, the node embedding update is performed through the original message passing technique used in graph neural networks. In contrast, in the other comparison models, we replace this part

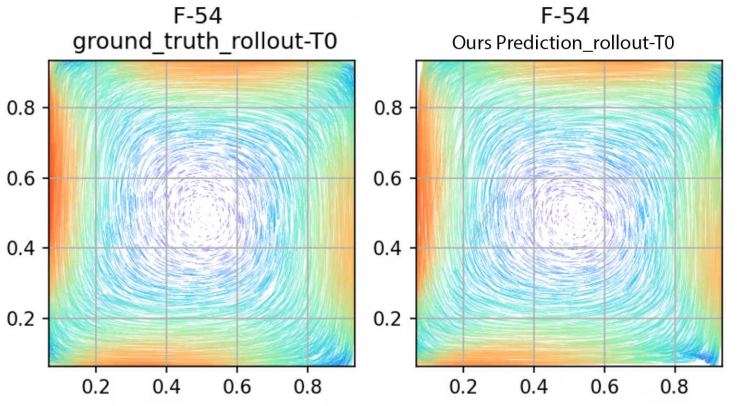


Figure 5: Velocity field predicted by our proposed model for frame 54, showcasing stability over extended rollouts.

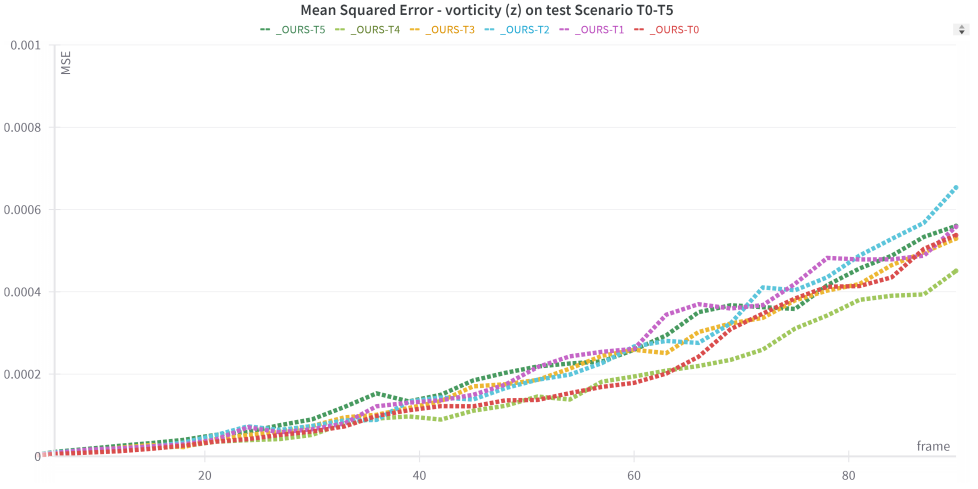


Figure 6: MSE of long term prediction using 6 different test case on GNS-SSGAT (_OURS) model

with different convolutional operator. Specifically, in GNS-SSGAT (_OURS), we replace the standard propagation method with the SuperGATConvolution Operator, which improves the model's ability to capture local interactions and conserve vorticity. All the models in this comparative study use the overall GNS framework, and the only difference between them lies in the convolutional layer used for node feature updates.

Figure 7 displays the mean squared error (MSE) in frames for various models predicting the vorticity (z-axis) using test scenario T0. The models included in the chart are `_GATv2Conv` (blue) [[29]], `_graph_transformer` (orange) [[26]], `_GNS` (Graph Network simulator, purple) [[16]], `_GATConv` (yellow) [[28]], `_pointnet_conv` (Pointnet++, green) [[24]] , and `GNS-SSGAT (_OURS)` (red). The x-axis represents the frames, while the y-axis represents the MSE.

In the initial frames (10-30), `_GATConv`, `_GNS` and `GNS-SSGAT (_OURS)` exhibit low MSE values. However, `GNS-SSGAT (_OURS)` starts with a slightly lower MSE, indicating marginally better performance from the beginning. During intermediate frames (30-40), `GNS-SSGAT (_OURS)` maintains a consistently lower MSE compared to `_graph_network-T0`. The latter shows occasional spikes and higher variability in error, indicating less stability in its predictions. In the later frames (40-60), `GNS-SSGAT (_OURS)` continues to outperform `_GNS` by maintaining a lower MSE. The gap between the two models becomes more apparent in these frames, highlighting the robustness and accuracy of `GNS-SSGAT (_OURS)` over time.

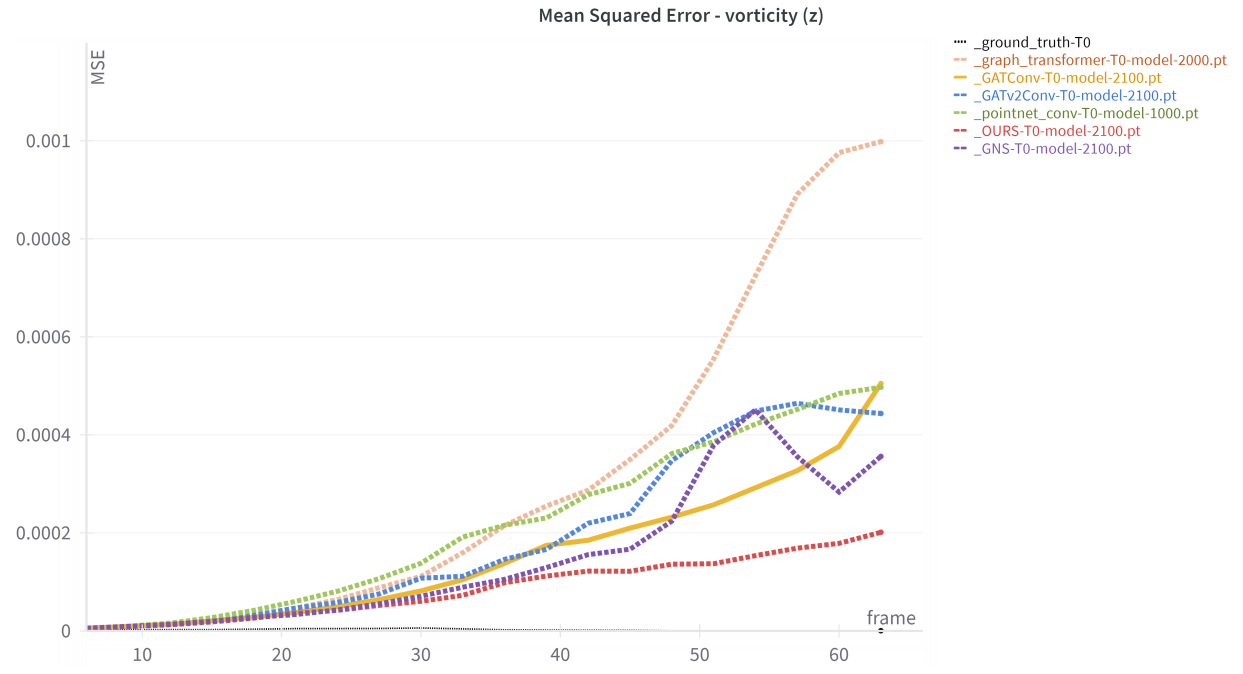


Figure 7: MSE benchmark comparing the performance of 6 models over long-range prediction of the z-component of vorticity. The red color represents our model, showing stability. During intermediate frames (10-40), GNS-SSGAT (_OURS) maintains a consistently lower MSE compared to $\_graph_{n}etwork-T0$. The latter shows occasional spikes and higher error variability, indicating less stability in its predictions.

Model	MAE	MSE	RMSE
GNS-SSGAT (_OURS)	0.007715	0.000151	0.012297
GATConv	0.010443	0.000289	0.017006
Pointnet_conv	0.012597	0.000420	0.020484
GNS	0.009757	0.000449	0.021191
GATv2Conv	0.011476	0.000447	0.021138
Graph_transformer	0.014519	0.000724	0.026903

Table 4
Comparison of model predictions with ground truth using MAE, MSE, and RMSE at frame and 54 (This was selected as a typical case).

Overall, the `GNS-SSGAT (_OURS)` model demonstrates superior performance with consistently lower MSE across all frames, suggesting better predictive accuracy and generalization. On the other hand, the `_graph_network-T0` model, while performing reasonably well, exhibits higher errors and variability, particularly in the later frames. This indicates potential issues with long-term predictions or generalization. Figure 7 illustrates that the `GNS-SSGAT (_OURS)` model consistently achieves lower mean square error (MSE) values in every frame compared to the `GNS` model. A precise numerical comparison for frame 54 is provided in Table 4.

6.2.5. Benchmark: Vorticity distribution

To analyze the vorticity distributions of the models with ground truth as the benchmark, we assess the alignment of each model's vorticity distribution with the ground truth on frame 54 as depicted in Figure 8. We further explore this using the Pearson coefficient index, residual plot, and KL divergence. Additional histograms for frames 6, 9, 18, 21, 48, and 51 are illustrated in Figure 23.

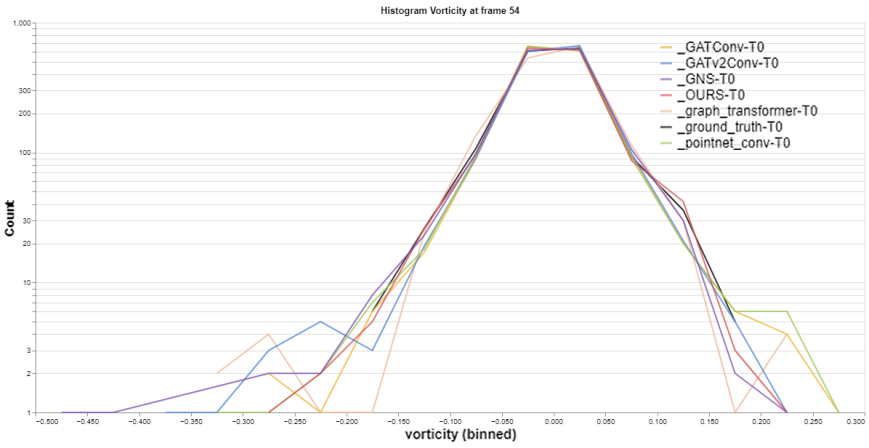


Figure 8: Histogram depicting the vorticity distribution at frames 54. The black lines represent the ground truth vorticity magnitude binned as the reference.

.

Peak Alignment The **GNS-SSGAT (_OURS)-T0 (Red Line)** aligns well with the ground truth around the peak vorticity values, showing similar peak height and position. Other models also show varying degrees of alignment: **GATConv-T0 (Orange)** and **GATv2Conv-T0 (Blue)** show peaks close to the ground truth but with slight deviations in height and width. The **GNS-T0 (Purple)** and **graph_transformer-T0 (Green)** lines have peaks that align reasonably well with the ground truth but exhibit more noticeable variations. The **pointnet (Yellow)** model has the largest deviation from the ground truth peak, with a wider spread and lower peak height.

Distribution Shape The overall shape of the **GNS-SSGAT (_OURS)-T0 (Red Line)** is quite similar to the ground truth, indicating a good match. Other models also show a similar distribution shape but with some differences in the tails: **GATConv-T0 (Orange)** and **GATv2Conv-T0 (Blue)**. The GNS-T0 (Purple) and graph_transformer-T0 (green) models show more uneven distribution shapes compared to the ground truth, and some bins show higher or lower counts. The **pointnet (Yellow)** model exhibits the largest deviation in shape, with a broader distribution and more significant discrepancies between bins.

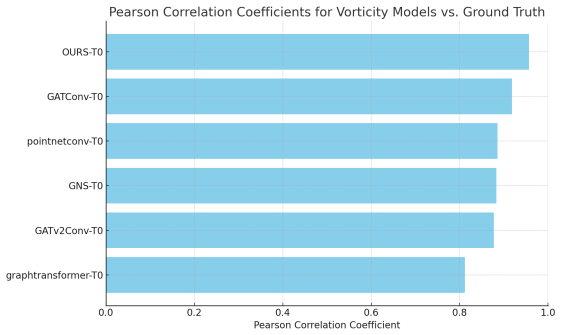
Smoothing and Noise The **GNS-SSGAT (_OURS)-T0 (Red Line)** is relatively smooth, indicating consistent predictions with some noise. Other models also exhibit varying levels of smoothness and noise: **GATConv-T0 (Orange)** and **GATv2Conv-T0 (Blue)** are smooth but with minor noise. The **GNS-T0 (Purple)** and **graph_transformer-T0 (Green)** models exhibit more noise and an uneven distribution, indicating less consistent predictions. The **pointnet (Yellow)** model has the most noise and an uneven distribution, indicating the least consistent predictions.

Analysis of Pearson Correlation Coefficient at frame 54, as illustrated in Figure 9, reveals the correlation between the vorticity data of each surrogate model and the actual ground truth. The model that aligns most closely with the ground truth is **GNS-SSGAT (_OURS)**, with a correlation coefficient of about 0.957.

The residual vorticity plots for frame 54 as illustrated in Figure 10 show that the GNS-SSGAT (_OURS)-T0 pattern has relatively minor residuals that are uniformly spread around the zero axis, demonstrating a strong correlation with the actual data.

Figure 11 shows that model **GNS-SSGAT (_OURS)-T0** has the lowest KL divergence, indicating the highest similarity to the ground truth. The low KL divergence for **GNS-SSGAT (_OURS)-T0** suggests that this model effectively captures the underlying distribution of the vorticity data, likely benefiting from its nonlinear modeling capabilities. The model's architecture may include advanced nonlinear transformations that allow it to better represent complex relationships within the vorticity data, leading to a closer match with the ground truth.

One of the key motivations for examining the vorticity distribution at **frame 54** rather than an early frame stems from the long-range rollout context. As shown in Figure7, each surrogate model's mean squared error (MSE) accumulates over time, making later frames such as frame 54 more stringent tests of a model's predictive stability and accuracy. While the MSE trend indicates how quickly or slowly each model's predictions deviate from the ground



Model	Pearson Correlation Coefficient
GNS-SSGAT (_OURS)-T0	0.956670
GATConv-T0	0.918301
pointnetconv-T0	0.885932
GNS-T0	0.883097
GATv2Conv-T0	0.877435
graphtransformer-T0	0.811620

Figure 9: Pearson Correlation Coefficients for Surrogate models and Ground Truth at frame 54. The "GNS-SSGAT (_OURS)" model exhibits the highest correlation.

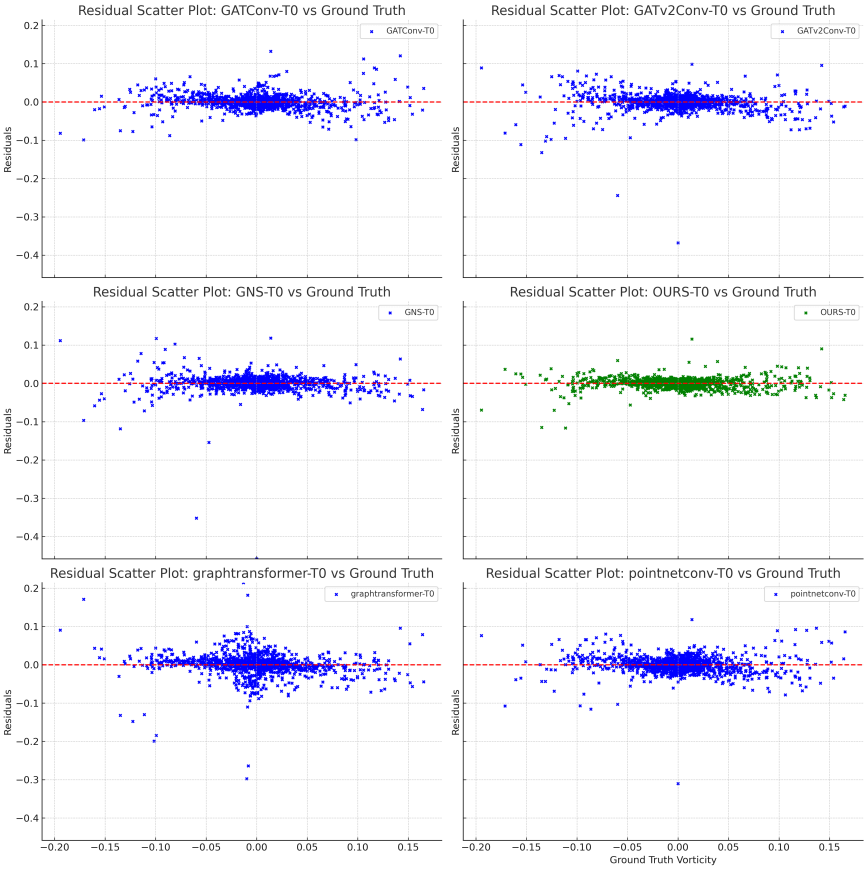
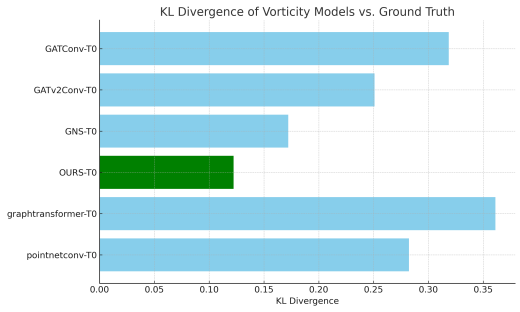


Figure 10: Residual Plots of vorticity of the surrogate models at frame 54. Proposed model (green-colored GNS-SSGAT (_OURS)-T0) pattern has relatively minor residuals that are uniformly spread around the zero axis, demonstrating a strong correlation with the actual data

truth on a pointwise basis, it does not capture whether the *overall shape* or *range* of the predicted vorticity distribution remains physically plausible.

Hence, a model could exhibit a respectable MSE yet still fail to capture the correct *distributional profile*, for example, if it consistently underestimates high-vorticity regions while overestimating moderate-vorticity areas. Conversely, a low KL divergence coupled with high MSE might signal that, despite getting the overall histogram roughly correct, certain regions of the flow field are spatially misaligned. Thus, neither metric alone suffices; but their



Model	MAE	MSE	RMSE	KL Div
GNS-SSGAT (_OURS)-T0	0.007715	0.000151	0.012297	0.127755
_GNS-T0	0.009757	0.000449	0.021191	0.172230
_pointnet_conv-T0	0.012597	0.000420	0.020484	0.296943
_GATv2Conv-T0	0.011476	0.000447	0.021138	0.269795
_GATConv-T0	0.010443	0.000289	0.017006	0.354617
_graph_transf-T0	0.014519	0.000724	0.026903	0.391100

Figure 11: KL Divergence of Surrogate Models Compared to Ground Truth. The model with the lowest KL divergence is highlighted in green, indicating the closest match to the ground truth vorticity distribution. Right: KL Divergence at frame 54 (This was selected as a typical case

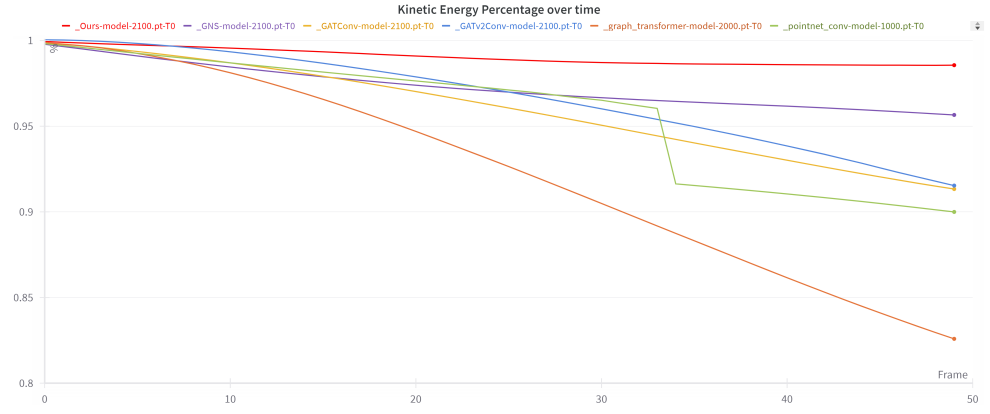


Figure 12: Kinetic Energy Conservation Benchmark of the surrogate models.

combination, especially at a later frame like frame 54, provides a more robust portrait of both pointwise accuracy and distributional realism.

Physical Relevance of Vorticity Distributions: In fluid dynamics, the distribution of vorticity across the domain is intimately tied to flow structures such as vortices, shear layers, and turbulent eddies. Capturing the shape of the vorticity histogram (via KL divergence) is crucial for ensuring that the model has learned the fundamental statistics of the flow. The fact that our **GNS-SSGAT (_OURS)-T0** model retains a relatively *low* MSE over the entire temporal horizon (up to frame 60) *and* achieves the *lowest KL divergence* at frame 54 demonstrates its ability to maintain both pointwise accuracy and global distributional fidelity under long-term rollout conditions.

6.2.6. Benchmark: Kinetic Energy Conservation

The plot 12 shows the kinetic energy percentage over time for various models. The following observations can be made regarding the performance of these models in terms of kinetic energy conservation.

The model **GNS-SSGAT (_OURS)** (represented by the red line) maintains the highest kinetic energy percentage over time, indicating the best performance in conserving kinetic energy. The kinetic energy decreases very slowly for this model, suggesting that it is highly effective in retaining the initial kinetic energy throughout the time frames. The vanilla **GNS** also maintains a high percentage of kinetic energy. The **GATv2Conv** variant shows moderate performance, with a steady decline in energy conservation. However, the **GATConv**, **graph_transformer**, and **pointnet_conv** variants exhibit significant drops in kinetic energy conservation, indicating the need for improvement in their energy conservation capabilities.

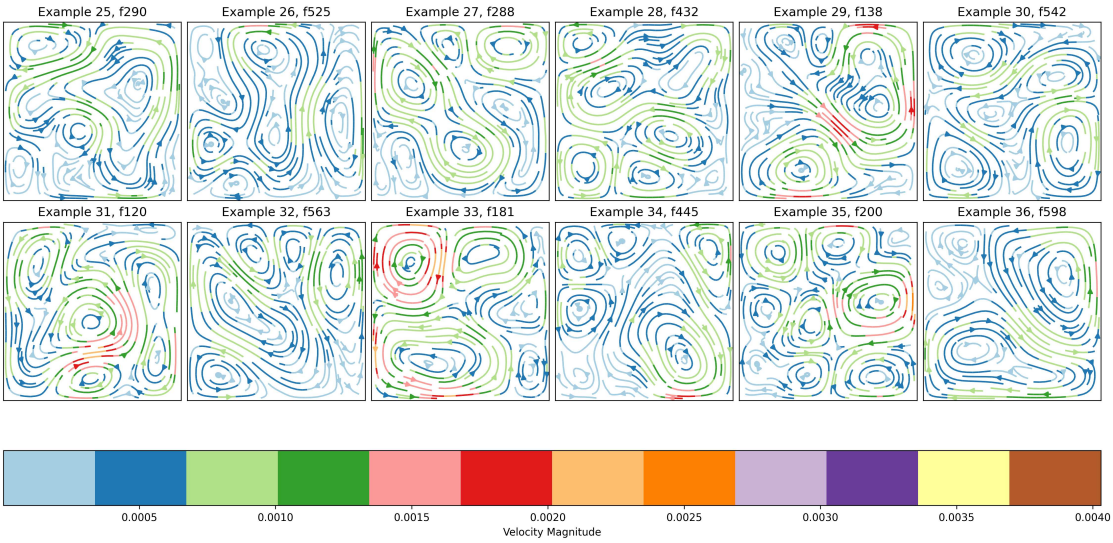


Figure 13: Illustrations of 36 training scenarios showing streamlines computed by interpolating particle velocities onto a regular grid using cubic interpolation. Color mapping indicates velocity magnitude. The simulations use weakly compressible MPM with divergence-free initial velocity fields.

6.3. Curl-Noise Flow Test (MPM regime)

Like Section 6.2, this subsection reports results on the MPM setup (Section 6.1.1, ∼8 192 particles, weakly-/semi-compressible material) with the six-processor preliminary comparison; the formal four-variant ablation on the FLIP/APIC incompressible regime is in Section 6.4.

6.3.1. Dataset

To evaluate the model's performance on more complex flow scenarios, we use curl-noise initialized flows. Curl noise [39] generates smooth, divergence-free vector fields by taking the curl of the gradient of Perlin noise functions [40, 41]. This method efficiently creates velocity fields with multiple vortices that respect solid boundaries, producing datasets that capture intricate and dynamic flow behaviors.

The curl noise vector field is defined using a potential function to guarantee a divergence-free field. Following [39], for 2D flows, the velocity field $\mathbf{v}$ is defined as:

$$\mathbf{v}_{\text{curl}}(x, y) = \left(\frac{\partial \Psi}{\partial y}, -\frac{\partial \Psi}{\partial x} \right)^T \tag{22}$$

where the potential function Ψ is defined to be a Perlin noise field, i.e., $\Psi(x, y) = N(x, y)$. This construction ensures that the field is divergence-free by construction ($\nabla \cdot \mathbf{v} = 0$), making it suitable for incompressible flow simulations. Particle positions are distributed almost uniformly on a 40×40 grid with small random jitter, and velocities are initialized by interpolating from the curl noise field. The simulation produces 48 training scenarios (snapshots for some of these are shown in Figure 13), each with varying noise scales (ranging from 4 to 13) to generate diverse flow patterns from smooth laminar to complex turbulent behaviors.

6.3.2. Training and Results

We evaluate the model on six test rollouts (Rollout 0-5), visualizing predictions using quiver plots (velocity vectors), streamline plots (flow patterns), and vorticity plots (rotational characteristics) at key frames (1, 5, 10, 20, 30, 40). Using equal weights (1:1:1:1:1:1) for all loss components ensures balanced learning across acceleration, velocity, density, divergence, position, and vorticity.

Streamline Plots The streamline plots, illustrated in Figure 14, offer a continuous visualization of the fluid flow, providing insight into the overall flow structure generated by the curl-noise simulation. The streamlines trace the paths that particles would follow under the simulated velocity field, with colors representing the velocity magnitude. These

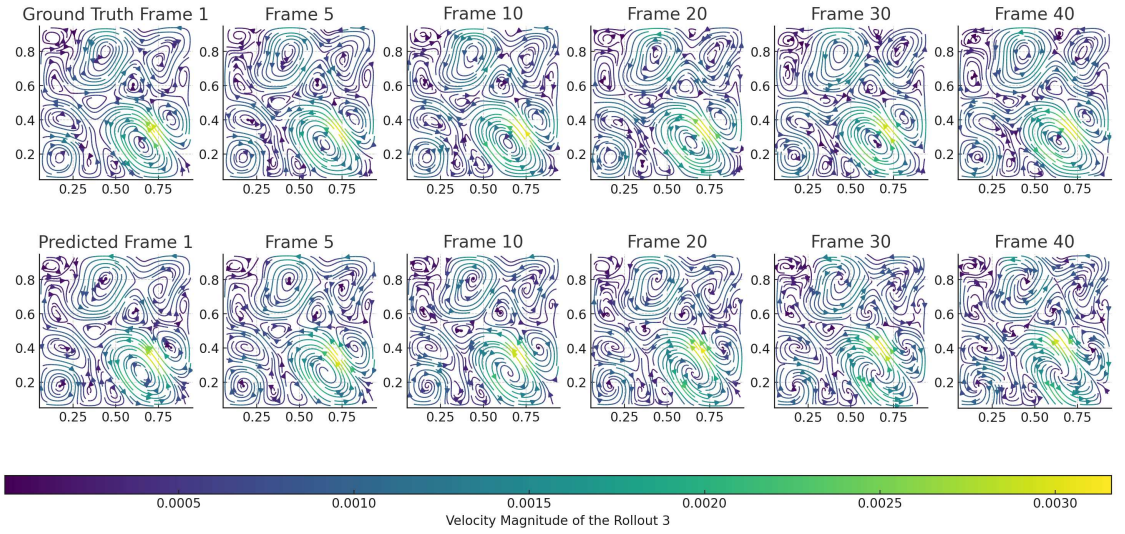


Figure 14: Streamline view for Rollout Tests 3. First row: Ground Truth Test case 3. Second row are the predictions of our porposed model.

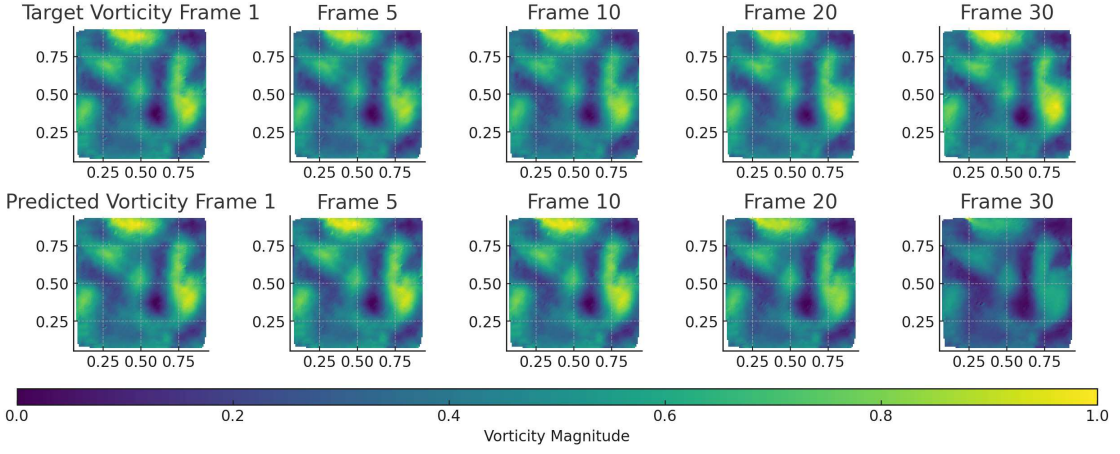


Figure 15: Vorticity plot for Rollout 0. The top row shows the target vorticity, while the bottom row shows the predicted vorticity across frames 1, 5, 10, 20, and 30.

plots provide a clearer understanding of the flow coherence and divergence within the simulated domain. Across the test sets, Rollouts 0 to 5, both quiver and streamline visualizations reveal the fidelity of the surrogate model in approximating the ground truth fluid dynamics. Most of the predicted paths align with the ground truth flow structures, indicating that the model captures the global flow patterns effectively.

Vorticity Plots The vorticity plots, shown in Figure 15, provide an analysis of the rotational characteristics of the fluid flow, representing the local spinning motion of the fluid. Note that because vorticity depends upon the derivatives of the flow field is significantly harder to reproduce since small errors in the velocity are magnified substantially in the vorticity. The plots compare the vorticity fields of the ground truth (target) and predicted simulations across five key frames: 1, 5, 10, 20, and 30. The color map used is offering a clear visualization of the vorticity magnitude. It is worth noting that, by frame 30, the predicted vorticity starts to decay notably faster than the ground truth, although the model maintains strong performance up to and including frame 20.

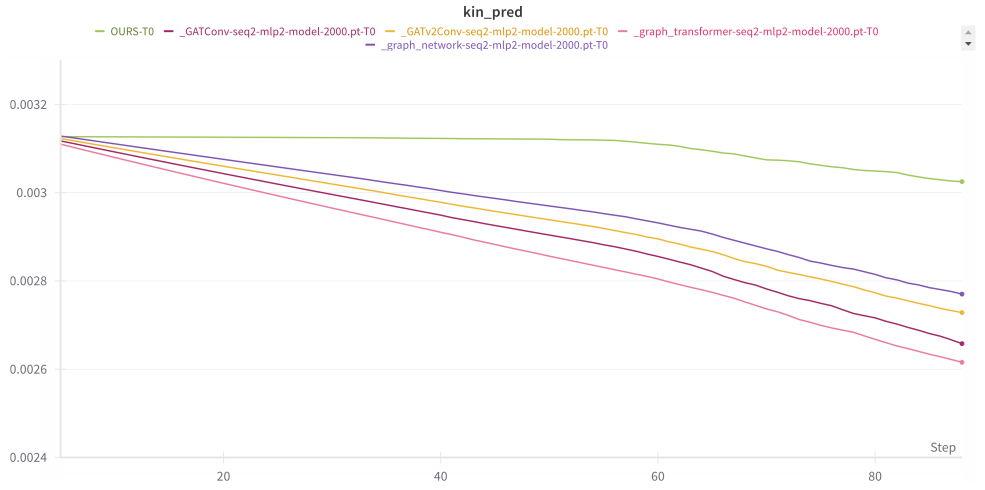


Figure 16: Comparison of kinetic energy conservation for our proposed model relative to other surrogate models.

Quiver Plots The quiver plots, as shown in Appendix C Figure 24, provide a direct representation of the velocity vectors at each timestep. The vectors' lengths and directions depict the magnitude and direction of the particle velocities. This method is particularly effective in highlighting the differences between the ground truth and predicted rollouts, as the color intensity correlates with the velocity magnitude. We have displayed the results at frames 1, 5, 10, 20, 30, and 40 for each rollout. In these plots, the predicted rollouts generally follow the ground truth closely, though some deviations can be observed in certain regions, particularly at later frames and near the boundaries of the simulation domain. These discrepancies are more apparent in the quiver plots, where the vector directions and magnitudes can differ noticeably from the ground truth.

Overall Assessment The combined analysis from quiver, streamline, and vorticity plots provides a comprehensive understanding of the surrogate model's performance in simulating fluid dynamics. The quiver and streamline plots demonstrate that the model captures the global flow patterns effectively, maintaining coherence with the ground truth. The vorticity plots, however, reveal areas where the model's predictions deviate from the ground truth, particularly in terms of rotational dynamics. These findings suggest that while the model is generally effective, there is still scope for further improvement in future work, particularly in accurately capturing the fine-scale rotational details of the fluid flow over time. Additional rollout visualizations for test cases 1-4 are provided in Appendix E.

6.3.3. Benchmark: Energy Conservation

The plot on figure 16 compares the performance of different models in predicting the conservation of kinetic energy during a curl noise simulation using Test case T0. The y-axis represents the predicted kinetic energy, while the x-axis shows the progression of the simulation over time steps.

The green line represents our model GNS-SSGAT (_OURS)-T0, which shows the highest kinetic energy values throughout the simulation. This indicates that it conserves kinetic energy better than the other models. The curve is relatively flat, meaning that the kinetic energy does not decrease as much over time, suggesting strong conservation properties.

The purple line, representing the *GATConv*, shows a moderate decrease in kinetic energy over time. This model performs worse than GNS-SSGAT (_OURS), but better than the GATv2Conv model (yellow). Its ability to conserve kinetic energy is less robust compared to our model, with a slightly steeper decline.

The pink line, representing the graph Transformer, exhibits the fastest decline in kinetic energy among the models shown. This suggests that the graph transformer is the least effective in conserving kinetic energy, which may result in a less accurate simulation of fluid dynamics, particularly in scenarios where maintaining energy levels is critical.

Finally, the magenta line represents the *graphnetwork*. The performance of this model falls between the graph transformer and the GATConv models. It has a moderate rate of energy loss, but it is still worse than the GATConv models and our proposed model.

6.4. Processor-Layer Ablation: 3-Variant Comparison (FLIP/APIC)

To probe the processor-layer design axis introduced in Section 5.3, we evaluate three representative processor instantiations on a common training harness: vanilla GNS (InteractionNetwork), GNS-FBC, and the full SFBC pipeline [18] as a non-graph external baseline. The graph-attention variant (GNS-SSGAT, Section 5.2) is reported in detail on the MPM regime in Sections 6.2–6.3 where it is the strongest single processor; its retraining under the harder FLIP/APIC harness exhibited hyperparameter-sensitivity that a fuller sweep (deferred to future work) should resolve, so we restrict the present formal ablation to the three variants with reliably comparable checkpoints. All three remaining variants share the encoder, decoder, six-component physics-informed loss, dual head, noise injection, training schedule, and dataset; the only varying axis is the processor. We use the FLIP/APIC incompressible-liquid datasets of Section 6.1.2 at ∼10 800 particles per scene and roll out **120 frames** per test trajectory — twice the typical surrogate-evaluation horizon — so that the long-rollout structure-preservation behaviour of each processor is exercised, not just its short-term accuracy.

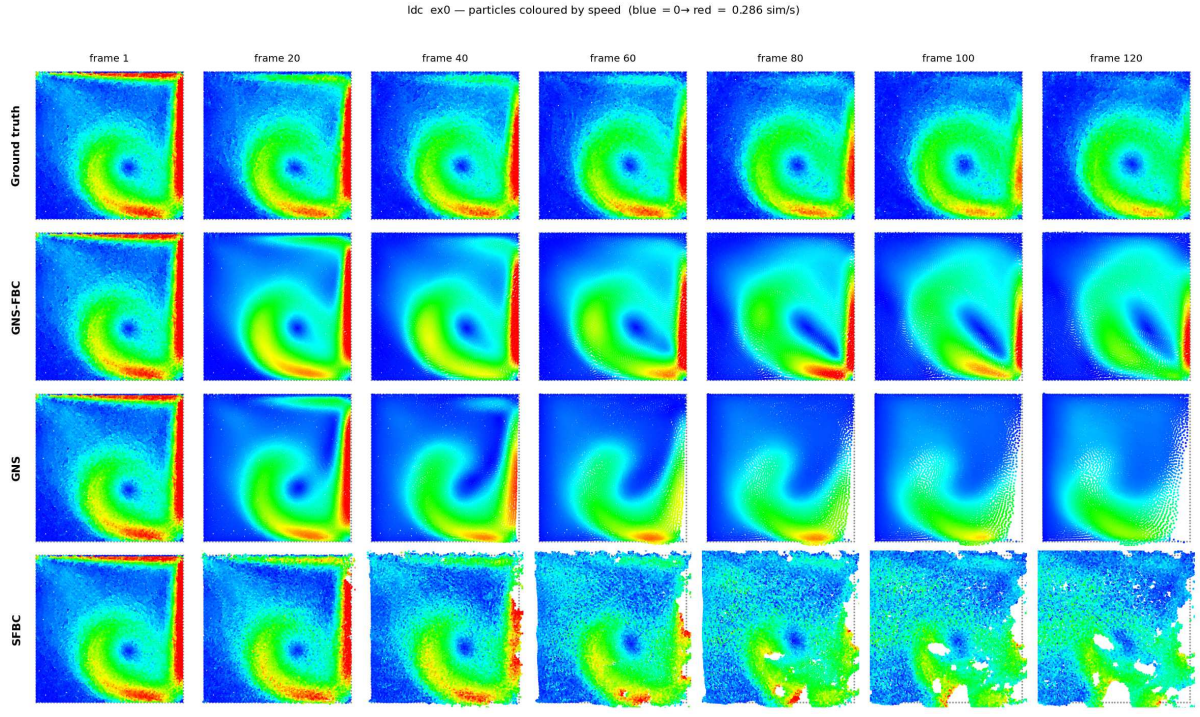


Figure 17: Long-rollout time-series on the 2D lid-driven cavity benchmark (rollout horizon 120 frames). Rows: ground-truth APIC reference, GNS-FBC, vanilla GNS, and the external SFBC pipeline. Columns: frames 1, 20, 40, 60, 80, 100, and 120. Particles are coloured by instantaneous speed using the SciVis palette (blue = 0 → red = 99th-percentile of the GT speed at the mid-rollout frame, kept fixed across frames so that colour shifts reflect speed shifts and not a moving colour scale). In the first ∼30 frames every variant tracks the ground truth reasonably well; from frame 60 onward, vanilla GNS develops visible voids and streaks while GNS-FBC continues to preserve a uniform particle distribution and the GT speed pattern; the full SFBC pipeline (no GNS-style training) drifts into chunk-level clumping. The figure makes the long-rollout advantage of GNS-FBC qualitatively unmistakable.

Figures 17 and 18 extend the comparison to a 120-frame rollout horizon. Both figures use the same layout: rows are the four particle systems (ground truth, GNS-FBC, vanilla GNS, SFBC), columns are sampled frames

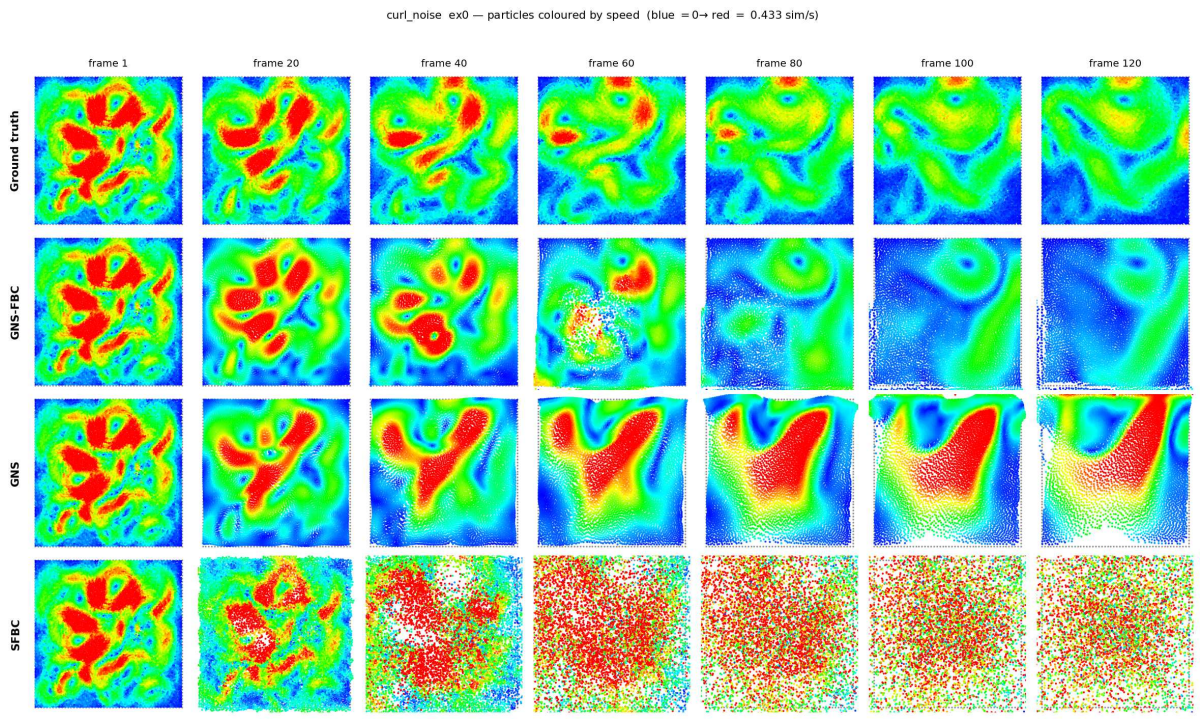


Figure 18: Long-rollout time-series on the curl-noise benchmark (rollout horizon 120 frames). Rows: ground-truth APIC reference, vanilla GNS, GNS-FBC, and the external SFBC pipeline. Columns: frames 1, 20, 40, 60, 80, 100, and 120. Particles are coloured by instantaneous speed using the SciVis palette (blue = 0 → red = 99th-percentile of the GT speed at the mid-rollout frame, fixed across frames). In the first ∼20 frames every variant tracks the ground truth reasonably well; from frame ∼40 onward vanilla GNS develops streak artefacts and the external SFBC pipeline drifts into chunk-level clumping, while GNS-FBC continues to track the ground-truth speed pattern and preserve spatial particle distribution through the full 120-frame horizon. The gap widens as the rollout progresses.

along the rollout. Figure 17 shows the lid-driven cavity benchmark and Figure 18 the curl-noise benchmark, both at frames 1, 20, 40, 60, 80, 100, 120. Across both benchmarks the picture is the same: in the first ∼30 frames every variant tracks the ground truth reasonably well; from frame ∼60 onward vanilla GNS and SFBC progressively develop voids, streaks, or chunk-level clumps, while GNS-FBC alone preserves a uniform particle distribution and the correct speed pattern through frame 120. This is the primary qualitative evidence for GNS-FBC's long-rollout structure-preservation advantage on the FLIP/APIC incompressible regime.

Figure 19 reports the nine per-frame physical-fidelity metrics on the lid-driven cavity benchmark; the matching curl-noise plot is provided in Appendix A (Figure 20). The per-panel optimisation direction is indicated by a small annotation directly below each panel title: minimisation (position MSE, vorticity-field MSE, KL divergence, mean $|\nabla \cdot v|$), maximisation (spatial Pearson correlation of the vorticity field, with $r = 1$ marking perfect alignment), or proximity to the GT trace (mean nearest-neighbour distance, kinetic energy, $|\omega|$). GNS-FBC dominates the three-variant comparison: it achieves the lowest vorticity-field MSE on the lid-driven cavity benchmark and is competitive with vanilla GNS at the equivalent horizon on curl-noise; it achieves the highest Pearson correlation and tracks GT most closely on $|\omega|$, kinetic energy, and NN-distance through the rollout on both benchmarks; vanilla GNS gives the strongest non-FBC baseline; and the external SFBC pipeline trails on most pattern metrics, indicating that the basis-convolution layer needs the GNS training harness to express its strengths.

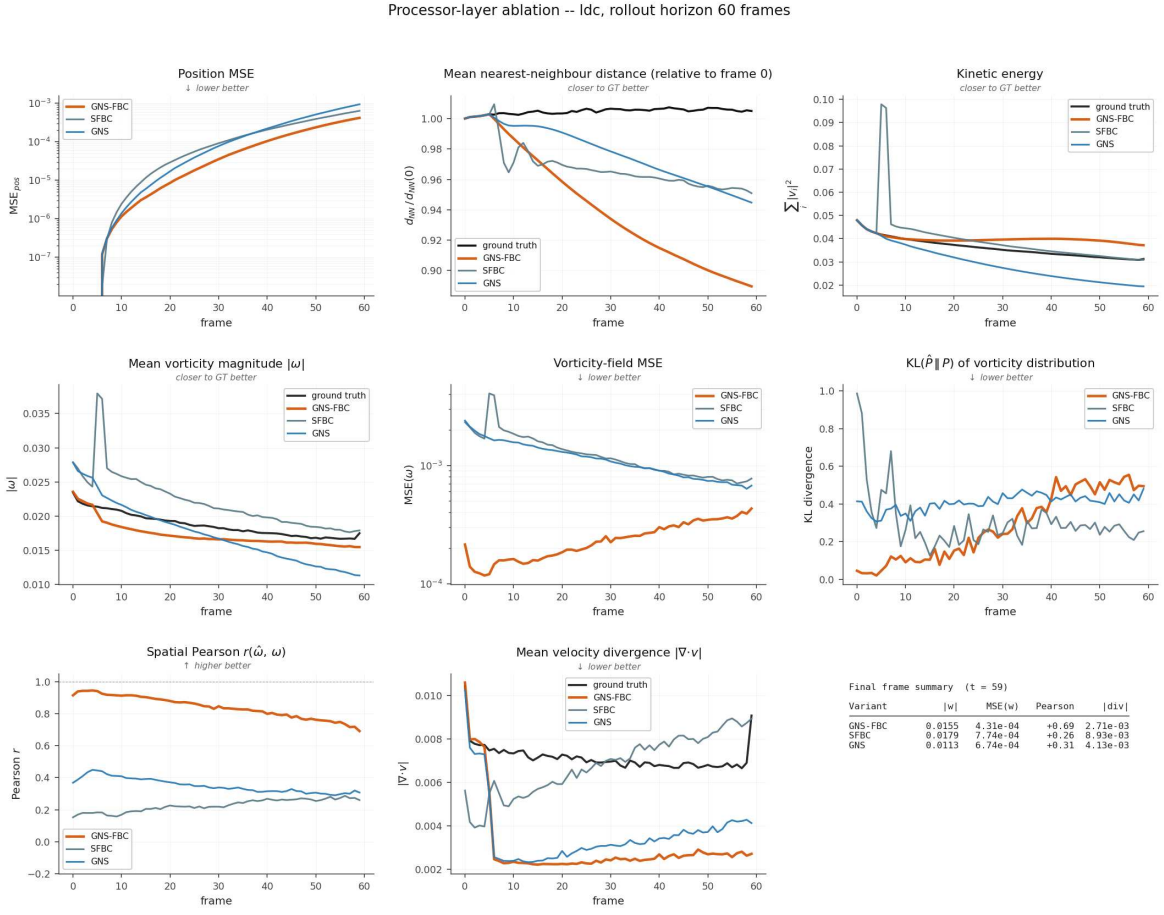


Figure 19: Per-frame ablation on the 2D lid-driven cavity benchmark (rollout horizon 60 frames). Each panel reports one physical-fidelity metric over time for the three processor instantiations and, where applicable, the ground-truth APIC reference. The per-panel optimisation direction is indicated by a small annotation in the top-right corner of each axes: minimisation (position MSE, vorticity-field MSE, KL divergence, mean $|\nabla\cdot v|$), maximisation (spatial Pearson correlation of the vorticity field, with $r = 1$ marking perfect alignment), or proximity to the GT trace (mean nearest-neighbour distance, kinetic energy, $|\omega|$). GNS-FBC achieves the lowest vorticity-field MSE and the highest Pearson correlation, and stays closest to the GT on the $|\omega|$, kinetic-energy, and NN-distance traces; vanilla GNS and the external SFBC pipeline trail. The bottom-right cell tabulates the final-frame ($t = 59$) numerical summary.

Table 5
Cost profile across the processor-layer design axis. All GNS-family variants share the same encoder, decoder, training harness, and dataset; the only varying axis is the processor. Inference cost is the mean rollout time per simulation frame on a single NVIDIA RTX A5000 GPU, averaged over the 6 test trajectories (∼1 200 frames per dataset). For the external SFBC pipeline the inference cost is split as *Net + SPH*: the SFBC network forward pass and the per-frame SPH-feature preprocessing (density, pressure, neighbour search) that SFBC's two-stream architecture requires but the GNS-family variants do not. The Taichi APIC ground-truth reference is included as the simulator-side cost target; it is the same Taichi-Lang APIC solver used to generate the training data, compiled to a GPU-parallel kernel that vectorises the particle-to-grid (P2G) and grid-to-particle (G2P) sweeps across all CUDA cores of the A5000, so the 10 ms/frame figure should be read as a heavily-optimised parallel reference rather than a naive serial baseline.

		Params	Inference (ms/frame)	
Variant	Processor	(M)	LDC	curl_noise
GNS	InteractionNet	1.59	277.8	50.7
GNS-FBC	BasisNetwork	3.46	279.3	224.7
SFBC (external)	BasisNetwork	1.10	37.4 + 2.4	11.0 + 2.4
APIC (Taichi GT)	—	—	10.4	10.2

7. Discussion

7.1. Key Findings

Our 4-variant ablation supports the central claim of this paper: *the processor layer within the GNS framework is a tunable design axis, and different processor choices produce systematically different physical-fidelity profiles* when trained under the same six-component loss on the same data. The design-axis observation is moreover not a feature of one simulator: an earlier exploration (reported below) made the same qualitative observation on a Material Point Method (MPM) benchmark of a semi-compressible fluid-with-deformable-solid material, and the present incompressible FLIP/APIC liquid benchmark — a strictly harder regime, with active divergence and density constraints — reveals a regime-dependent winner along the axis. In the preliminary MPM exploration the attention-based GNS-SSGAT processor is the strongest single instantiation, achieving the lowest vorticity-field MSE, the highest Pearson correlation (0.957), the smallest KL divergence (0.128), and the strongest kinetic-energy conservation among the six-processor sweep [15]. In the present FLIP/APIC incompressible study the basis-convolution-based GNS-FBC dominates the controlled three-variant ablation instead: as shown by the long-rollout particle visualisations in Figures 17 and 18 it preserves the spatial particle distribution better than any other variant including the full SFBC pipeline, and on the per-frame metrics (Figures 19 and 20) it achieves the lowest vorticity-field MSE, the highest Pearson correlation, and the closest tracking of the GT kinetic-energy and vorticity-magnitude curves. Operationally, GNS-SSGAT is the recommended instantiation when the target physics is a semi-compressible MPM regime where attention-based message passing can latch onto sharp local vorticity-magnitude gradients; GNS-FBC is the recommended instantiation when the target is an incompressible liquid regime with active divergence constraints, where the basis-convolution's fixed spatial filters preserve large-scale flow pattern and particle distribution over long rollouts. The vanilla GNS baseline trails both instantiations on their respective target regimes, which confirms that the processor-layer choice — not the loss alone — is what drives the observed gains. The processor layer is therefore not a fixed sub-module but a deliberate design choice, and we encourage future work to explore further points along this axis.

In a preliminary exploration that motivated the present study, we trained and rolled out seven processor families on a different *physics regime*: a Material Point Method (MPM) simulation of a weakly-/semi-compressible fluid-with-deformable-solid material at 8 192 particles per scene. The processor families evaluated there were InteractionNetwork (the GNS baseline), `SuperGATConv`, `GATConv`, `GATv2Conv` [28, 29], a Graph-Transformer variant [26], a CConv-style PointNet++ processor [24, 19], and a PointNet+GAT hybrid. That preliminary study used a shorter training budget (approximately 2 000 steps) and pre-dated the six-component physics-informed loss used in this paper, and the underlying simulator was MPM rather than the FLIP/APIC incompressible liquid used here; its absolute numbers are therefore not commensurate with those reported in this paper and we omit them here

to avoid mixing simulator-physics regimes and training budgets. Qualitatively, however, that exploration established that swapping the processor changes the physical-fidelity profile in observable and reproducible ways, and that of the seven candidates the attention-based SuperGAT and the basis-convolution-based BasisNetwork (FBC) yielded the cleanest trade-offs and is reported in detail in Sections 6.2–6.3. The controlled three-variant ablation in Section 6.4 then takes the basis-convolution candidate (GNS-FBC), the vanilla GNS baseline, and the full SFBC pipeline as an external baseline, and re-runs them under a controlled training harness on a *harder* physics regime — incompressible FLIP/APIC liquid at ∼10 800 particles per scene, where the divergence and density terms of the six-component loss are actively enforced. The graph-attention candidate (GNS-SSGAT) was retrained under the same FLIP/APIC harness as a sanity check, but its FLIP/APIC checkpoint did not reach a quality comparable to its MPM-regime checkpoint (the rollouts on the curl-noise benchmark diverged into order-of-magnitude-worse vorticity-field MSE on this dataset, which we attribute to a training-stability issue rather than to a structural limitation of SuperGAT); we therefore restrict the FLIP/APIC ablation to three variants and report SuperGAT's strong performance separately in the MPM regime, where it is well-supported by the six-processor comparison of Sections 6.2–6.3. The fact that the processor-layer choice changes the physical-fidelity profile in observable and reproducible ways across both regimes — with a different best instantiation in each regime — is what supports treating the processor layer as a design axis rather than as an artifact of any single benchmark. The earlier MPM checkpoints remain available with the companion code release for follow-up work that wishes to retrain them under the present harness.

7.2. Model Behavior Analysis

On the MPM semi-compressible regime (Sections 6.2–6.3), GNS-SSGAT's effectiveness stems from its self-supervised attention mechanism that dynamically adjusts neighbour importance. Unlike the uniform message-passing in standard GNS or the static attention in GAT variants, SuperGAT learns to emphasize the critical local interactions that carry the rotational structure of the flow, which is what the per-cell vorticity-magnitude metric on this regime actually rewards.

The Fourier Basis Convolution (FBC) processor wins on the complementary axis of *pattern* fidelity for a different mechanistic reason. Where SuperGAT's attention is fundamentally *pointwise* – each neighbour contributes a single learned scalar weight derived from feature similarity – the basis convolution decomposes each neighbour contribution into a sum of $K^2 = 64$ Fourier-basis components evaluated on the relative position $\tilde{\mathbf{r}}_{ij} = \mathbf{r}_{ij}/R$ (Eq. (13)). Each basis component $\phi_{k_1}(\tilde{r}_{ij,1})\,\phi_{k_2}(\tilde{r}_{ij,2})$ is a fixed spatial filter of a particular wavelength and orientation, so the neighbour aggregate is a learned linear combination of 64 spatially-distinct templates rather than a single magnitude-scaled feature. This decomposition gives FBC two structural advantages over pointwise attention: (i) the kernel *shape* is parameterised directly in space, so the processor can express anisotropic neighbour relationships (e.g. "upstream vs. downstream" or "along-shear vs. cross-shear") without having to derive them from feature similarity; and (ii) the basis components naturally encode medium-wavelength spatial structure that survives many message-passing rollouts, which is what the long-rollout Pearson and particle-distribution metrics actually reward. The cost is a heavier parameter budget (3.46 M vs 1.10 M for GNS-SSGAT, Table 5) and the loss of the dynamic per-edge adaptivity that makes SuperGAT precise on per-cell vorticity magnitude. The two processors therefore complement each other: SuperGAT is preferable when the application targets accurate per-cell vorticity magnitude (local rotation rates, energy spectra); FBC is preferable when the application targets large-scale flow patterns and long rollout horizons (visual fidelity, downstream re-meshing, free-surface tracking).

7.3. Limitations and Future Work

Despite the strong performance reported in Sections 6.2–6.4, several limitations remain. **Long-rollout error growth.** Although GNS-FBC tracks the ground-truth particle distribution through the full 120-frame horizon on both 2D benchmarks (Figures 17 and 18), accumulated error remains noticeable on the vorticity-pattern metrics (Pearson correlation, KL divergence; Figure 19). Rollouts beyond the 120-frame horizon evaluated here are not characterised by the present work. **Single physics regime per ablation.** The formal three-variant ablation of Section 6.4 uses an incompressible FLIP/APIC liquid; the MPM weakly-compressible benchmark of Sections 6.2–6.3 retains six processor families but at a different particle count and training budget. We do not report a single benchmark on which all processor families are directly comparable; doing so would require re-running every MPM-regime checkpoint

under the FLIP/APIC harness, which falls outside the scope of the present submission. **GNS-SSGAT under the FLIP/APIC harness.** As noted in Section 6.4, our retraining of the graph-attention processor under the FLIP/APIC harness exhibited hyperparameter-sensitivity that we did not resolve here. A capacity, learning-rate, and noise-injection sweep — deferred to future work — is the natural next step before a four-variant comparison can be reported under the harder regime. **Two-dimensional scope.** All benchmarks evaluated here are 2D. Extension to 3D vortex rings, vortex shedding, and free-surface flows is the natural validation of generalisability for the design-axis claim. **Compute cost.** Training a single processor variant takes roughly 7–25 GPU-hours on a single NVIDIA RTX A5000 (per-variant inference cost is summarised in Table 5); distributed training across multiple GPUs is straightforward but the absolute wall-clock cost may limit accessibility for groups without dedicated GPU clusters. **Future work.** Three natural directions emerge from the design-axis framing: (i) scaling the strongest variant identified per regime — GNS-FBC for incompressible flows, GNS-SSGAT for MPM-style flows — to 3D and to free-surface scenarios; (ii) extending the axis with additional processor families (Graph-Transformer, PointNet++, hybrid attention/basis processors) under the same training harness; and (iii) incorporating physical constraints directly into the network architecture — for example divergence-free output projection layers or hard mass-conservation heads — beyond the loss-based supervision used here.

8. Conclusion

We presented extensions of the Graph Network Simulator (GNS) framework for transient flow problems that recast the processor sub-module as a tunable design axis, and instantiated the axis with two complementary processors integrated into the same shared encoder/decoder template: GNS-SSGAT (Self-Supervised Graph Attention) and GNS-FBC (Fourier Basis Convolution adapted from SFBC [18]). On a common training harness — a six-component physics-informed loss, dual acceleration/velocity prediction, and distributed training with noise injection — we characterised the axis through two complementary ablation studies. On the MPM semi-compressible regime, a six-processor preliminary comparison (Sections 6.2 and 6.3) shows that GNS-SSGAT is the strongest single processor: it achieves the lowest vorticity-field MSE, the highest Pearson correlation, the smallest KL divergence, and the strongest kinetic-energy conservation among the six families evaluated. On the harder FLIP/APIC incompressible-liquid regime, a controlled three-variant ablation (Section 6.4) comparing GNS-FBC against vanilla GNS and the full SFBC pipeline as a non-graph external baseline shows that GNS-FBC dominates: it achieves the lowest vorticity-field MSE, the highest Pearson correlation, and the best long-rollout particle-distribution stability. Together the two studies show that processor choice changes the physical-fidelity profile in observable and reproducible ways, with the best instantiation depending on the target physics regime — a finding that supports treating the processor layer as a deliberate design choice rather than as a fixed sub-module. The vanilla GNS baseline trails both processor instantiations on their respective target metrics, confirming that the processor-layer choice drives the observed gains. We derive empirically-grounded recommendations on which processor to choose for which downstream target regime and metric, and we release the training harness and the trained checkpoints to support follow-up exploration of further points along the axis. Future work will scale both variants — GNS-SSGAT for MPM-style semi-compressible regimes where attention sharpens local vorticity-magnitude accuracy, and GNS-FBC for incompressible regimes where basis-convolution preserves long-rollout structure — to three-dimensional transient-flow benchmarks with free surfaces, and study transfer between training and deployment-time particle resolutions.

Acknowledgments

We gratefully acknowledge the LPDP – Indonesia Endowment Fund for Education Agency for supporting this research, and the NVIDIA Hardware Grant Program for providing the NVIDIA RTX A5000 GPU used in our experiments.

References

[1] Allen R. York II. Development of modifications to the material point method for the simulation of thin membranes, compressible fluids, and their interactions. *Sandia Report*, 1997.

[2] Deborah Sulsky, Z. Chen, and H. Schreyer. A particle method for history-dependent materials. *Computational Methods in Applied Mechanics and Engineering*, 118(1):179–196, 1994.

[3] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32(4):102, 2013.

[4] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle. The material point method for simulating continuum materials. *ACM SIGGRAPH 2016 Courses*, page 24, 2016.

[5] D. Ram, T. Gast, C. Jiang, C. Schroeder, A. Stomakhin, J. Teran, and P. Kavehpour. A material point method for viscoelastic fluids, foams and sponges. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, page 157–163, 2015.

[6] Y. Yue, B. Smith, P. Chen, M. Chantharayukhonthorn, K. Kamrin, and E. Grinspun. Continuum foam: a material point method for shear-dependent. *ACM Transactions on Graphics*, 37(4):160, 2015.

[7] Alexey Stomakhin, Craig Schroeder, Chenfanfu Jiang, Lawrence Chai, and Joseph Teran. Augmented mpm for phase-change and varied materials. *ACM Transactions on Graphics (TOG)*, 33(4):138, 2014.

[8] Dody Dharma, Cliff Jonathan, A. Imam Kistidjantoro, and Afwarman Manaf. Material point method based fluid simulation on gpu using compute shader. In *2017 International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA)*, 2017.

[9] Ming Gao, Xinlei Wang, Kui Wui, Andre Pradhana, Eftychios Sifakis, Cem Yuksel, and Chenfanfu Jiang. Gpu optimization of material point methods. *ACM Transactions on Graphics*, 37(6), 2018.

[10] Gilles Daviet and Florence Bertails-Descoubes. A semi-implicit material point method for the continuum simulation of granular materials. *ACM Transactions on Graphics*, 35(4):Article No. 102:1–13, 2016. doi: 10.1145/2897824.2925877. URL https://hal.science/hal-01310189. hal-01310189.

[11] Yuanming Hu, Yongyi Fang, Zeyu Ge, Zherong Qu, Yingyu Zhu, Adrian Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics*, 37(4):150:1–150:14, 2018. doi: 10.1145/3197517.3201365.

[12] Q. Guo, X. Han, C. Fu, T. Gast, R. Tamstorf, and J. Teran. A material point method for thin shells with frictional contact. *ACM Transactions on Graphics*, 37(4):147, 2018.

[13] Joshuah Wolper, Yu Fang, Minchen Li, Jiecong Lu, Ming Gao, and Chenfanfu Jiang. Cd-mpm: Continuum damage material point methods for dynamic fracture animation. *ACM Transactions on Graphics*, 38(4), 2019.

[14] M. Gao, A. Tampubolon, C. Jiang, and E. Sifakis. An adaptive generalized interpolation material point method for simulating elastoplastic materials. *ACM Transactions on Graphics*, 36(6):223, 2017.

[15] Dody Dharma, Peter K. Jimack, and He Wang. Minrnns for lagrangian-based simulations of transient flow problems. In Maciej Paszynski, Amanda S. Barnard, and Yongjie Jessica Zhang, editors, *Computational Science – ICCS 2025 Workshops*, volume 15907 of *Lecture Notes in Computer Science*, Cham, 2025. Springer. doi: 10.1007/978-3-031-97554-7_17. URL https://doi.org/10.1007/978-3-031-97554-7_17.

[16] Alvaro Sanchez-Gonzalez. Learning to simulate complex physics with graph networks, 2020. URL https://arxiv.org/abs/2002.09405.

[17] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision, 2022. URL https://arxiv.org/abs/2204.04879.

[18] Rene Winchenbach and Nils Thuerey. Symmetric basis convolutions for learning lagrangian fluid mechanics, 2024. URL https://arxiv.org/abs/2403.16680.

[19] Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1lDoJSYDH.

[20] Lukas Prantl, Benjamin Ummenhofer, Vladlen Koltun, and Nils Thuerey. Guaranteed conservation of momentum for learning particle-based fluid dynamics, 2022. URL https://arxiv.org/abs/2210.06036.

[21] Shanyan Guan, Huayu Deng, Yunbo Wang, and Xiaokang Yang. Neurofluid: Fluid dynamics grounding with particle-driven neural radiance fields, 2022. URL https://arxiv.org/abs/2203.01762.

[22] Shiying Xiong, Xingzhe He, Yunjin Tong, Yitong Deng, and Bo Zhu. Neural vortex method: from finite lagrangian particles to infinite dimensional eulerian dynamics, 2023. URL https://arxiv.org/abs/2006.04178.

[23] Yitong Deng, Hong-Xing Yu, Jiajun Wu, and Bo Zhu. Learning vortex dynamics for fluid inference and prediction, 2023. URL https://arxiv.org/abs/2301.11494.

[24] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017. URL https://arxiv.org/abs/1706.02413.

[25] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. URL https://arxiv.org/abs/1612.00593.

[26] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification, 2021. URL https://arxiv.org/abs/2009.03509.

[27] Ye Yuan, Animesh Mehta, and Rose Yu. Transformer with implicit edges for particle-based physics simulation, 2022. URL https://arxiv.org/abs/2207.10860v1.

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018. URL https://arxiv.org/abs/1710.10903.

[29] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks?, 2022. URL https://arxiv.org/abs/2105.14491.

[30] G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 1967. URL https://doi.org/10.1017/CBO9780511800955.

[31] Shirley Dowdy, Stanley Weardon, and Daniel Chilko. *Statistics for Research*. John Wiley and Sons, 2004. URL https://sdp2013.wordpress.com/wp-content/uploads/2013/09/statistics-for-research-dowdy.pdf.

[32] Bora Sul, Anders Wallqvist, Michael Morris, Jaques Reifman, and Vineet Rakesh. A computational study of the respiratory airflow characteristics in normal and obstructed human airways. *Computers in Biology and Medicine*, 52, 09 2014. doi: 10.1016/j.compbiomed.2014.06.008.

[33] David C. Hoaglin. John w. tukey and data analysis, 2003. URL https://www.jstor.org/stable/3182748.

[34] Ingrid Brady. Stds assignment 1 - vignette - residual analysis, 2018. URL https://rpubs.com/iabrady/residual-analysis.

[35] Nikolaj Buhl. Kl divergence in machine learning. *Encord*, 2023. URL https://encord.com/blog/kl-divergence-in-machine-learning.

[36] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. URL https://arxiv.org/abs/1412.6980.

[37] Taichi lang, 2024. URL https://www.taichi-lang.org/.

[38] Y. Hu, T. Li, L. Anderson, J. Ragan-Kelley, and F. Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics*, 38(6):1–6, 2019.

[39] Robert Bridson, Jeremy Hourihan, and Marcus Nordenstam. Curl-noise for procedural fluid flow. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, page 177–185, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936282. doi: 10.1145/1276377.1276435. URL https://doi.org/10.1145/1276377.1276435.

[40] Ken Perlin. An image synthesizer. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques (SIGGRAPH '85)*, pages 287–296, New York, NY, USA, 1985. ACM. doi: 10.1145/325334.325247.

[41] Ken Perlin. Improving noise. *ACM Transactions on Graphics (TOG)*, 21(3):681–682, 2002. doi: 10.1145/566654.566636.

A. Per-frame ablation on the curl-noise benchmark

Figure 20 reports the nine per-frame physical-fidelity metrics on the curl-noise benchmark, the companion to Figure 19 in Section 6.4. Panels and optimisation-direction annotations follow Figure 19; we place it in this appendix rather than the main text because the LDC panel already shows the headline GNS-FBC dominance and the curl-noise panel reproduces the same ordering on the second benchmark without introducing new findings beyond what Section 6.4 discusses.

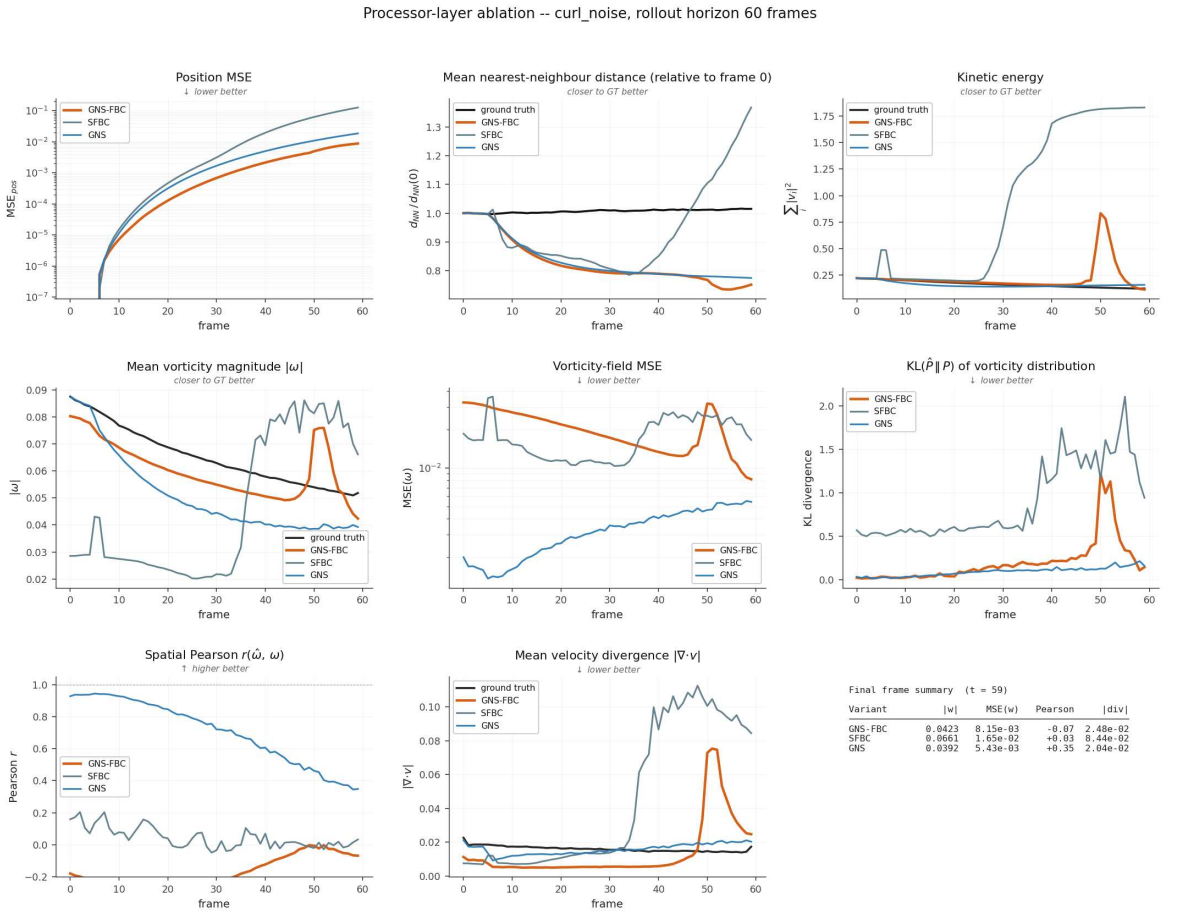


Figure 20: Per-frame ablation on the curl-noise benchmark (rollout horizon 60 frames). Panels and optimisation-direction annotations follow Figure 19. GNS-FBC tracks the ground-truth $|\omega|$, kinetic-energy, and NN-distance traces the closest, achieves the lowest KL divergence and the lowest mean $|\nabla \cdot v|$, and is competitive with vanilla GNS on the vorticity-field MSE at this horizon; the external SFBC pipeline trails on the pattern metrics. The consistency of the pattern-fidelity ordering across the two qualitatively different flow regimes (lid-driven cavity in Figure 19 and curl-noise here) supports the design-axis claim of Section 5.3.

B. Training Details and Data Normalization

B.1. Data Normalization Procedure

Normalization is applied using precomputed statistics from `metadata.json`:

- Position normalization: $\mathbf{p}_{\text{norm}} = \frac{\mathbf{p}-\mu_p}{\sigma_p}$
- Velocity normalization: $\mathbf{v}_{\text{norm}} = \frac{\mathbf{v}-\mu_v}{\sigma_v}$
- Acceleration normalization: $\mathbf{a}_{\text{norm}} = \frac{\mathbf{a}-\mu_a}{\sigma_a}$

After prediction, inverse normalization is applied in the decoder to obtain physical quantities.

B.2. Distributed Training Initialization

1. Initialize distributed environment with `torch.distributed`
2. Load dataset metadata and normalization statistics
3. Initialize model with hyperparameters
4. Wrap model with `DistributedDataParallel`
5. Initialize optimizer with scaled learning rate: $\text{lr} = \text{base\_lr} \times n_{\text{processes}}$
6. Setup distributed data loader to partition data across GPUs

C. Additional 2D Lid-Driven Cavity Flow Results

This appendix presents additional qualitative and quantitative results for the 2D lid-driven cavity flow experiment. Figure 21 provides a visual comparison of the predicted vorticity and velocity fields at frame 54 across all six surrogate models, illustrating the lower error achieved by our proposed method. To further analyze the error distribution, Figure 22 shows scatter plots of vorticity residuals against grid cell indices, where our model (highlighted in green) exhibits residuals closest to zero. Additionally, Figure 23 extends the distributional analysis from the main text by showing vorticity histograms at frames 6, 9, 18, 21, 48, and 51, demonstrating the stability of the predicted distributions over time.

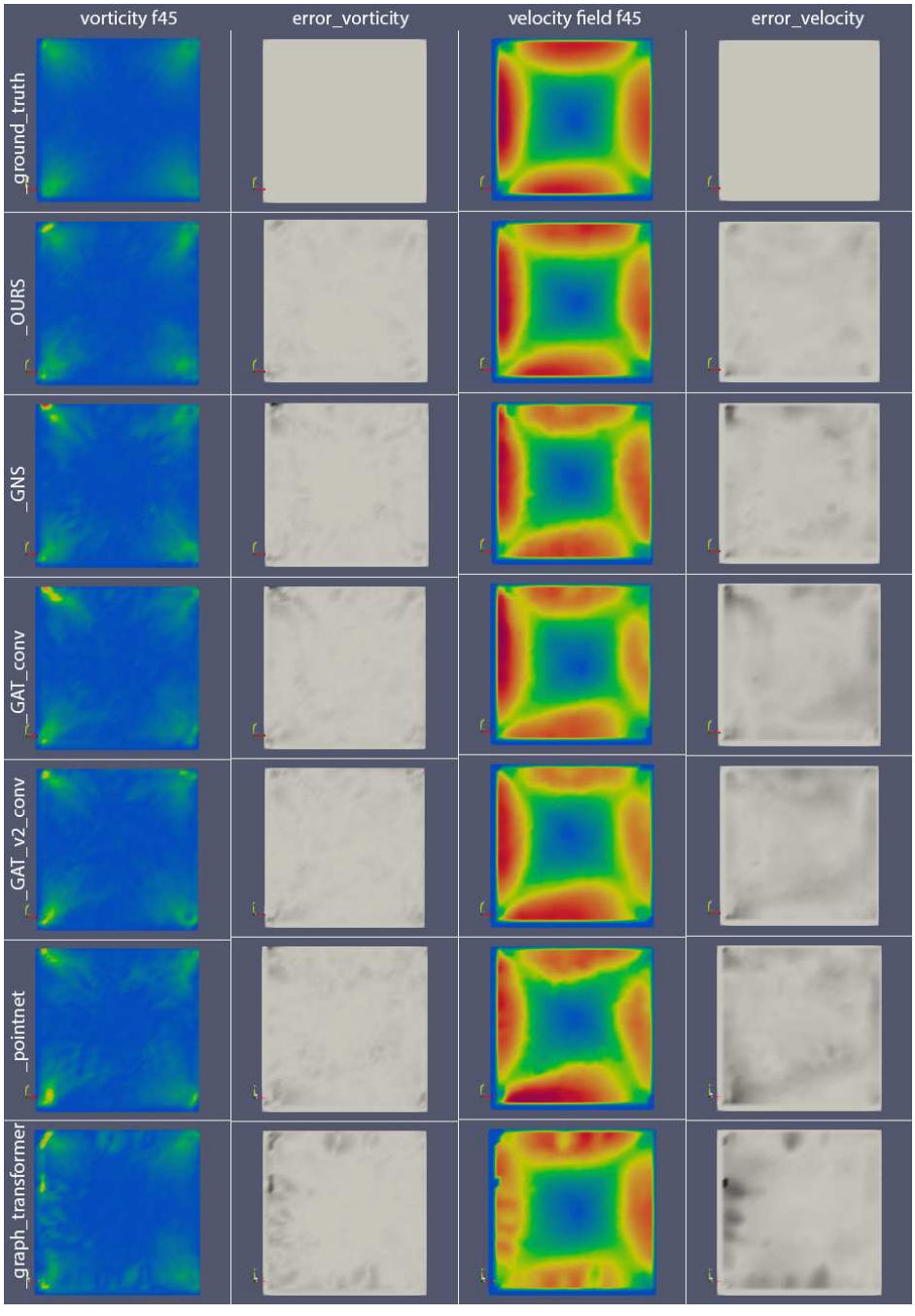


Figure 21: Qualitative comparison of predictions from 6 surrogate models at frame 54. The GNS-SSGAT model demonstrates the lowest error both in vorticity and velocity prediction

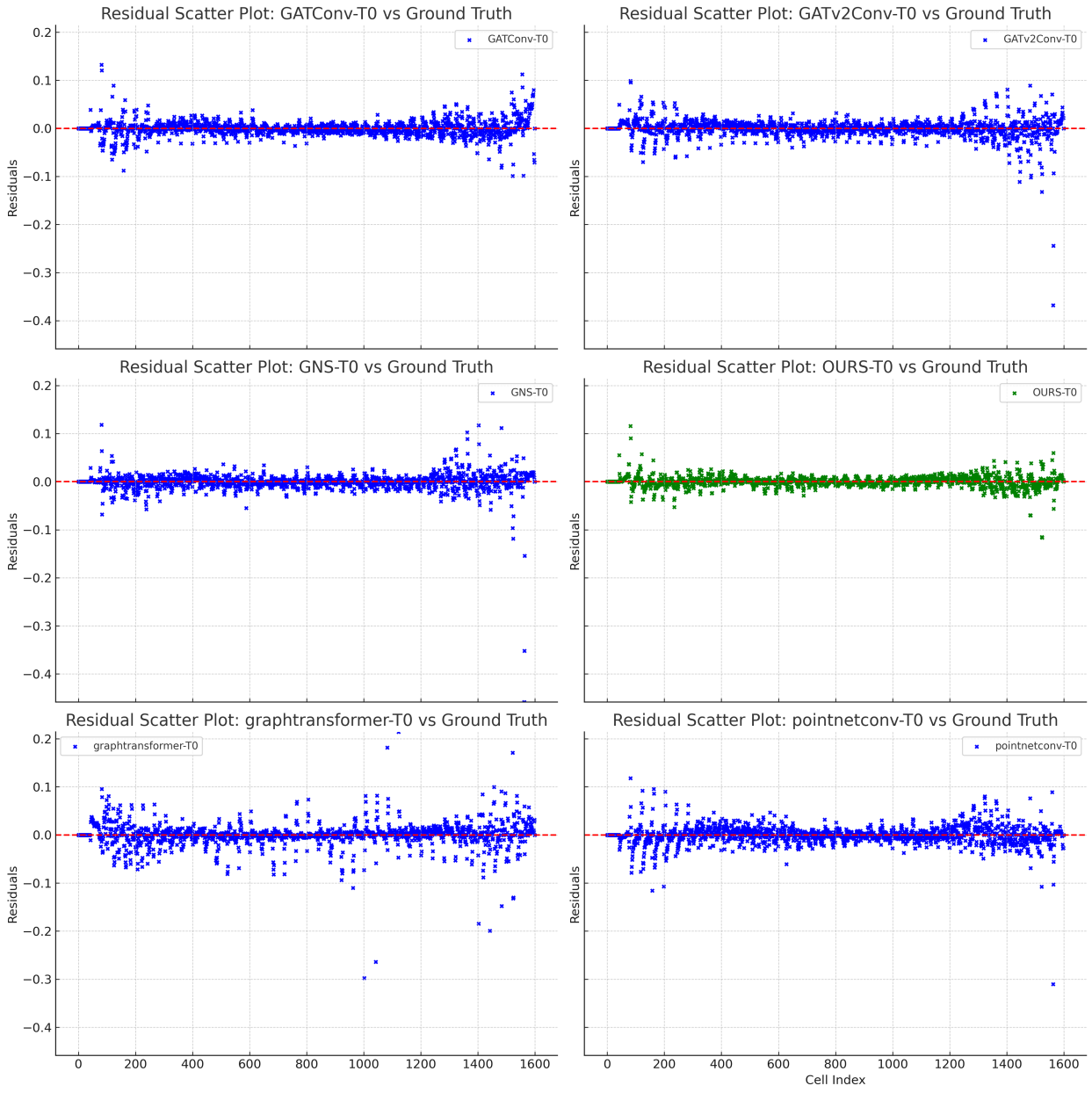


Figure 22: Scatter Plots of Residuals for Vorticity Models vs. Ground Truth with Cell Index (grid dimension: 40x40) on the X-Axis. The model with the lowest residuals is highlighted in green, while other models are shown in blue. The red dashed line represents zero residuals.

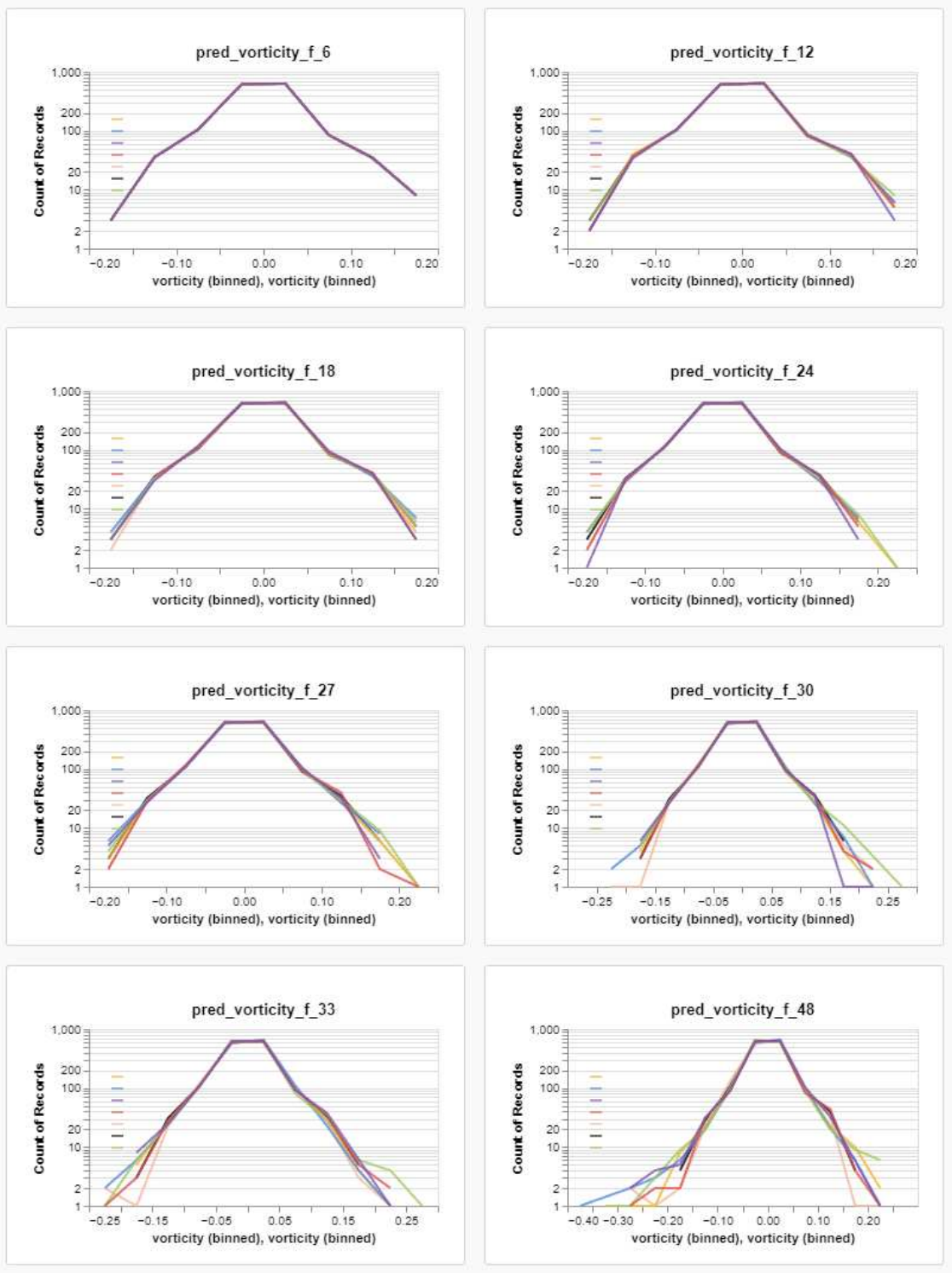


Figure 23: Vorticity distribution (left to right, top to bottom) at frames [6, 9],[18, 21],[48, 51] . The black lines represent the ground truth vorticity magnitude binned as the reference, `_GATv2Conv` (blue) , `_graph_transformer` (orange) , `_GNS` (purple) , `_GATConv` (yellow), `_pointnet_conv` (green) , and `GNS-SSGAT (_OURS)` (red)

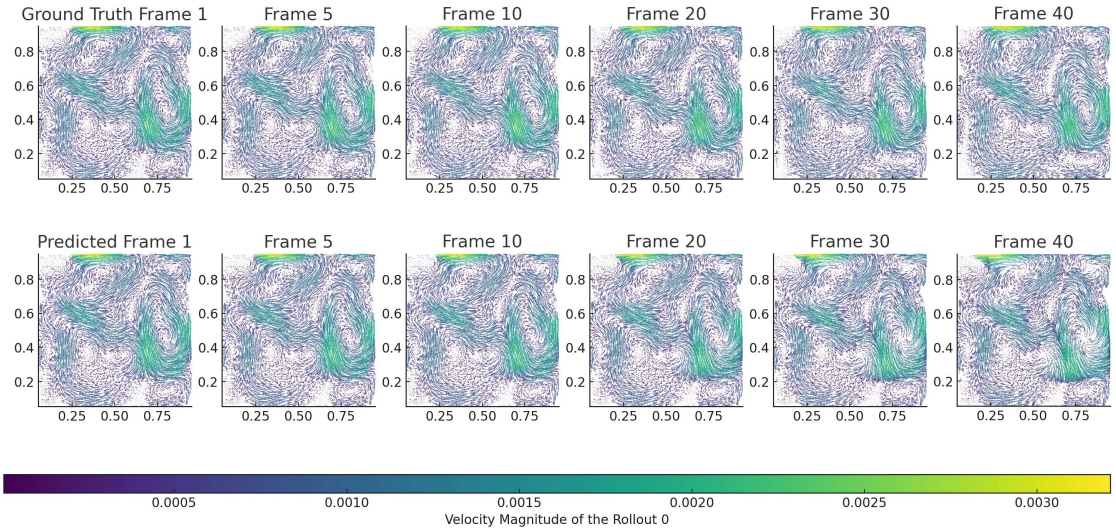


Figure 24: Quiver plots of Rollout test 0. Our proposed surrogate model predicts the subsequent frames (1-40) given the initial frame (0).

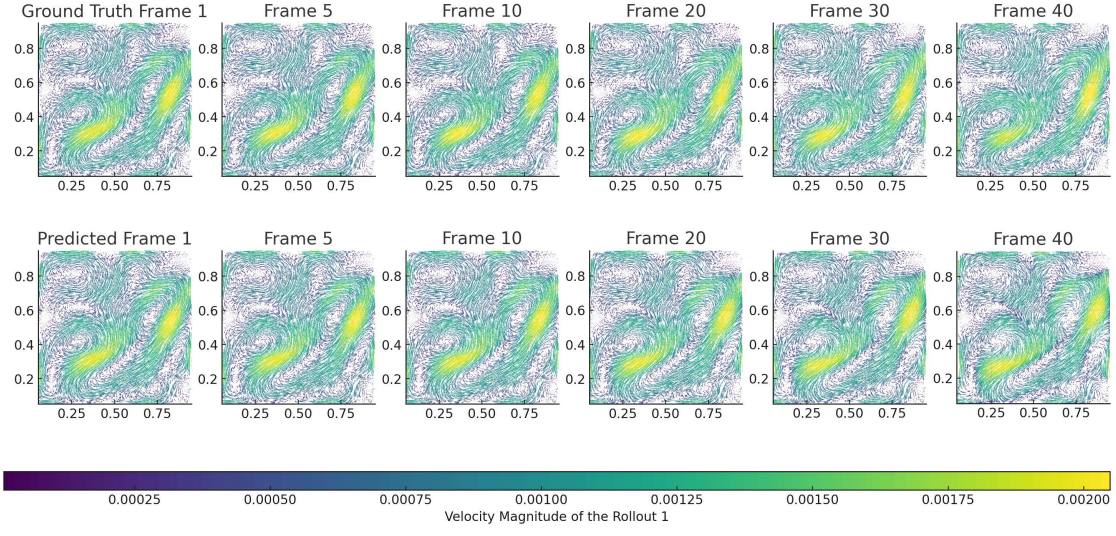


Figure 25: Rollout Test 1: Quiver plots showing predicted velocity fields. Curl-noise rollout results for test cases 1, The top row shows ground truth, and the bottom row shows model predictions.

D. Additional Curl-Noise Rollout Results

This appendix provides additional visualization results for curl-noise flow test cases 1-4, complementing the primary results shown in Section 6.3. Each figure shows quiver plots of predicted particle velocities at key frames (1, 5, 10, 20, 30, 40), demonstrating the model's long-term prediction capability across diverse initial conditions.

In the early frames (1 and 5), the predicted vorticity fields closely match the target, indicating that the model is well-calibrated to the initial conditions and can accurately simulate the early development of fluid flow, including the formation of initial vortices. However, as the simulation progresses to frames 10, 20, and 30, discrepancies between the target and predicted vorticity fields become more pronounced. While the overall patterns of vorticity are preserved, the exact magnitudes and positions of vortices begin to differ, especially in regions with complex interactions or high magnitude of velocity field.

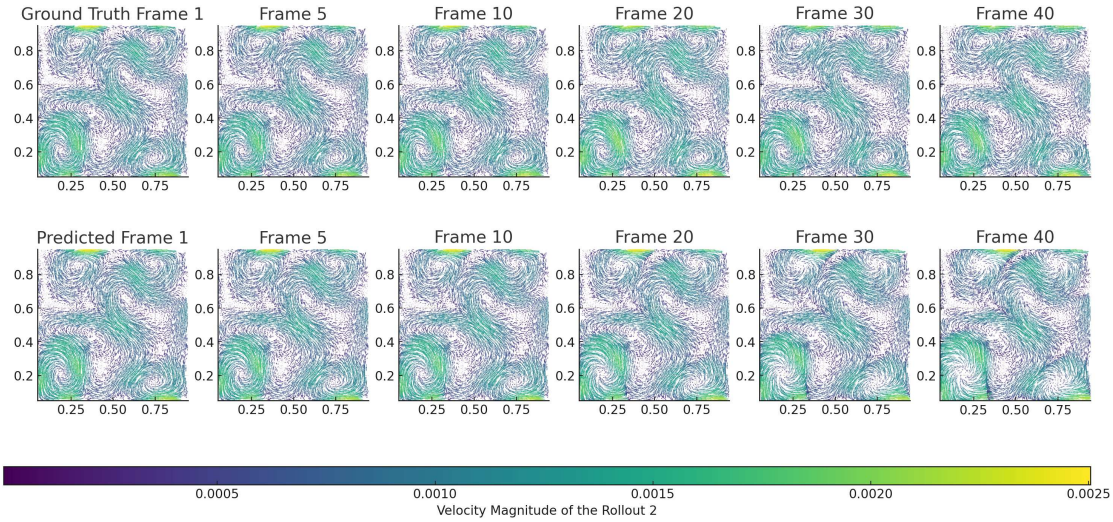


Figure 26: Rollout Test 2: Quiver plots showing predicted velocity fields

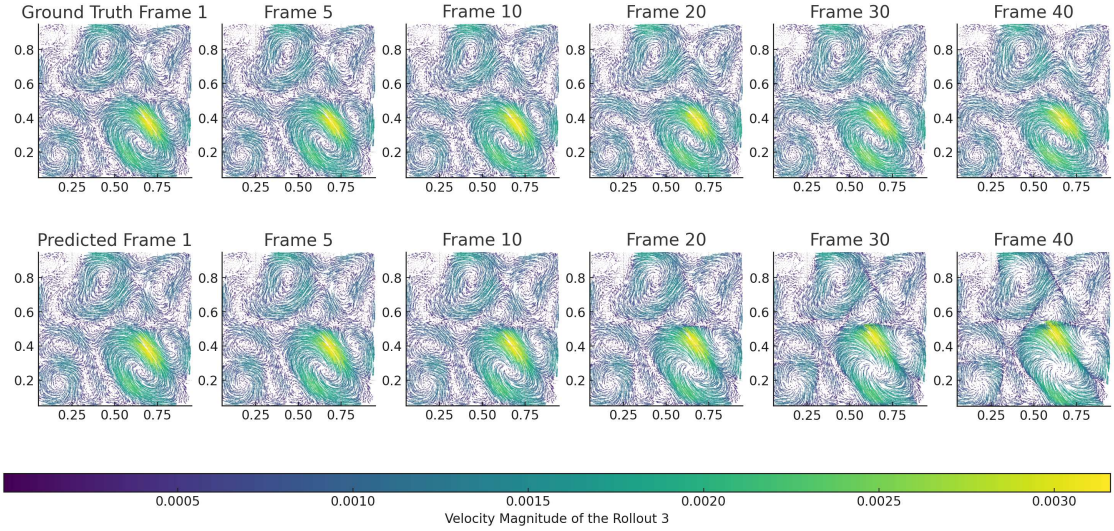


Figure 27: Rollout Test 3: Quiver plots showing predicted velocity fields, demonstrate consistent performance across varied noise scales and initial conditions.

By frame 30, the differences more noticeable, with some vortices in the predicted fields either displaced or differing in intensity compared to the ground truth. This suggests that the model, while effective in capturing general dynamics, may struggle with fine-scale details as the simulation evolves, particularly in complex regions where small initial discrepancies can amplify over time. The use of a single color bar across all frames enhances the comparability between the target and predicted vorticity fields, making it easier to identify regions where the model's predictions diverge from the ground truth.

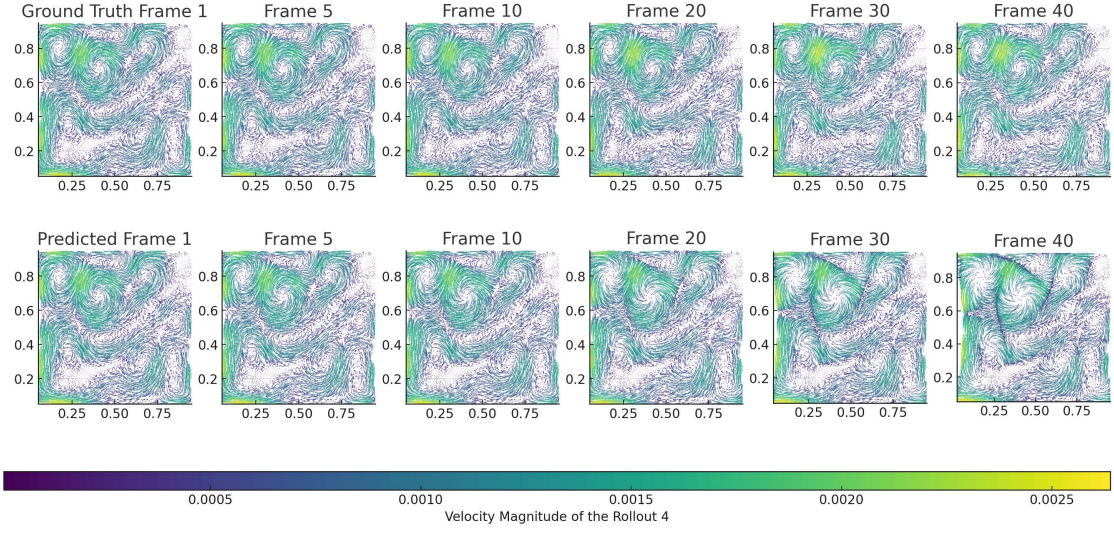


Figure 28: Rollout Test 4: Quiver plots showing predicted velocity fields. These results demonstrate consistent performance across varied noise scales and initial conditions.