CoLEP: Co-Evolving Particles and Persistent Eulerian Latent Fields for Neural Fluid Simulation

Anonymous authors
Paper under double-blind review

Abstract

Graph neural networks (GNNs) have emerged as a powerful framework for learning particle-based fluid dynamics. However, GNN-based simulators suffer from rapid error accumulation over long rollouts and the high computational cost of repeated message passing. We introduce CoLEP, a co-evolving Lagrangian–Eulerian particle simulator that augments particle-carried states with a persistent latent field on a fixed Eulerian grid. At each step, current particle information updates the inherited Eulerian state, which is then fed back to guide particle prediction, allowing spatial interaction context to accumulate and be reused across time. Building on this persistent state, we introduce periodic GNN skipping (PGS), which interleaves full graph-based interaction updates with learned graph-free Fast steps while continuing to evolve both the particle and Eulerian states. Across four particle-simulation benchmarks, CoLEP consistently improves long-horizon rollout accuracy over the evaluated baselines. Eulerian state ablations further show that persistence is particularly important when graph evaluations are sparse, while PGS provides a favorable accuracy–inference-cost trade-off.

1 Introduction

Particle-based methods represent continuous media as discrete material points that move with the flow, naturally handling free surfaces, large deformations, and multiphase interactions (Monaghan, 2012; Sulsky et al., 1994; Jiang et al., 2016). Graph Neural Networks (GNNs), whose relational structure closely mirrors particle interactions, have become the dominant paradigm for learning such dynamics from data. GNS (Sanchez-Gonzalez et al., 2020) introduced a general encoder–processor–decoder architecture applicable to fluids, rigid bodies, and deformable solids. Subsequent work has strengthened physical fidelity through geometric equivariance (Satorras et al., 2021; Brandstetter et al., 2022a), momentum conservation (Prantl et al., 2022), and learned time integration (Toshev et al., 2023a). Maintaining this fidelity over long rollouts is critical to the practical utility of learned particle simulators (Toshev et al., 2024).

Achieving such long-horizon fidelity, however, remains a fundamental challenge for neural particle simulators. During autoregressive rollout, prediction errors perturb subsequent model inputs and can accumulate over time, causing simulated trajectories to drift from the reference dynamics (Sanchez-Gonzalez et al., 2020; McCabe et al., 2023). To mitigate this accumulation, a common approach is to expose the model to perturbed or self-generated states at training time. GNS (Sanchez-Gonzalez et al., 2020) injects noise into its inputs, DMCF (Prantl et al., 2022) trains on progressively longer rollouts, and related strategies include the pushforward trick (Brandstetter et al., 2022b) and multi-step unrolling (List et al., 2025). These strategies improve robustness to imperfect inputs during autoregressive rollout during training.

Beyond these training-time remedies, another line of work tackles rollout fidelity through the lens of physical representation and interaction modeling. Standard particle GNNs adopt a Lagrangian formulation, associating features with moving particles and modeling interactions via local message passing (Sanchez-Gonzalez et al., 2020). However, capturing long-range interactions necessitates stacking multiple message-passing layers, substantially increasing computational costs (Gilmer et al., 2017). Conversely, the Eulerian perspective anchors features to fixed spatial locations, enabling

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

regular grids and multiscale architectures to efficiently aggregate broader spatial context (Ronneberger et al., 2015). To exploit the computational efficiency of Eulerian representations in modeling global interactions, recent methods have begun to incorporate an Eulerian perspective into particle-based simulation. Inspired by the material point method (MPM) (Sulsky et al., 1994), NeuralMPM (Rochman-Sharabi et al., 2025) restricts learnable dynamics to a voxelized background grid before transferring updates back to particles. Similarly, CoRGI (Ji et al., 2026) augments particle-level GNN interactions with grid-aggregated spatial context. In both frameworks, the Eulerian grid acts as a transient computational medium to structure and route particle updates.

We propose CoLEP, a co-evolving Lagrangian–Eulerian particle simulator that jointly advances two coupled states, one carried by moving particles and the other maintained as a persistent latent field on a fixed Eulerian grid. At each step, current particle features are combined with the inherited grid state to update the field, which in turn guides the next particle prediction. In transient particle–grid formulations, errors in the current particle state can propagate into the grid features reconstructed from it, without an inherited Eulerian state to provide additional context. CoLEP's coupling provides a mechanism for mutual refinement, allowing current particle observations to revise the evolving field while retained field information can help correct deviations in particle predictions.

To reduce computational cost, we further introduce periodic GNN skipping (PGS), which interleaves full steps with graph-free fast steps on a fixed schedule. Fast steps preserve spatial communication through the evolving grid field, which combines inherited context with current particle-wise encodings and returns updated features to the particles. We train lightweight adapters at the grid input and particle readout to accommodate the missing GNN features, while keeping the Full model frozen. PGS thus substantially reduces repeated graph computation while preserving bidirectional particle–field coupling and the evolution of both states across all steps.

In summary, our main contributions are:

- We introduce CoLEP, a co-evolving particle–grid simulator that jointly advances particle-carried states and a persistent Eulerian latent field across autoregressive steps. The Eulerian field is continually updated from particle dynamics and fed back to particle prediction, providing spatial context beyond that reconstructed from the current particle state alone.
- Building on this persistent Eulerian state, we introduce periodic GNN skipping (PGS), which interleaves full graph-based interaction updates with a learned graph-free Fast branch while continuing to evolve both the particle and Eulerian states at every step.
- Across four particle-simulation benchmarks, CoLEP substantially improves long-horizon position and distribution accuracy over the evaluated baselines. Controlled memory interventions and matched memoryless ablations further show that persistence is particularly important when graph evaluations are sparse, while PGS provides a favorable accuracy–inference-cost trade-off.

2 Related Work

GNN-based Particle Simulation and Particle–Grid Architectures Particle-based simulators model a physical system as a set of interacting particles whose states are advanced over time. In GNN-based formulations, particles become graph nodes and message passing aggregates information from neighboring particles to model their interactions. Building on Interaction Networks (Battaglia et al., 2016) and DPI-Net (Li et al., 2019), the Graph Network-based Simulator (GNS; Sanchez-Gonzalez et al., 2020) formalized this paradigm with an encoder–processor–decoder architecture that has since been applied to fluids, rigid bodies, and elastoplastic materials. Subsequent work has enriched local interaction modeling. SEGNN (Brandstetter et al., 2022a) uses steerable equivariant message passing, while Toshev et al. (2023a) incorporate temporal physical quantities into equivariant networks for Lagrangian fluid simulation. Particle GNNs construct a neighborhood graph and perform message passing at each model evaluation; both operations are computationally expensive (Toshev et al., 2023b). For local message passing, reaching more distant particles requires more neighbors or more layers, further increasing the cost (Ji et al., 2026).

To accelerate spatial information transfer beyond local neighborhoods, several methods supplement particle GNNs with Eulerian representations. Chentanez et al. (2022) connect particles through increasingly coarse sparse-grid cells. CoRGI (Ji et al., 2026) projects particle features onto a grid,

processes them with multiscale convolutions, and interpolates them back to particles. The Hybrid Lagrangian–Eulerian Model (Li et al., 2026) aggregates particle features at fixed Eulerian nodes and uses cross-attention to return spatial context to the particle update.

Long-Horizon Rollout and Temporal State Prediction errors in autoregressive neural simulators can compound over long rollouts (Sanchez-Gonzalez et al., 2020; Brandstetter et al., 2022b; Toshev et al., 2024). Prior work improves robustness through noise injection (Sanchez-Gonzalez et al., 2020), pushforward training (Brandstetter et al., 2022b), multi-step unrolling (Rochman-Sharabi et al., 2025; List et al., 2025), and physical constraints (Prantl et al., 2022). The Hybrid Lagrangian–Eulerian Model (Li et al., 2026) also uses fixed Eulerian nodes as a spatial reference to reduce trajectory drift. Yet in this model, as in NeuralMPM (Rochman-Sharabi et al., 2025) and CoRGI, Eulerian features are constructed from particle inputs at each model call. Their spatial locations are fixed, but the learned features are not carried across calls as an evolving state and can reflect errors in the latest particle predictions. CoLEP instead evolves an Eulerian field alongside the particles, retaining spatial context from preceding steps for subsequent updates. This inherited context also motivates testing particle updates that bypass GNN message passing on selected steps.

3 Problem Formulation

In the framework of neural particle simulators, the evolution of a particle system is formulated as an autoregressive rollout. The system comprises fluid particles and prescribed obstacle or boundary particles, indexed by $\mathcal{V} = \{1, \dots, N\}$. Let $\mathcal{V}_{\text{fluid}} \subseteq \mathcal{V}$ index the fluid particles, and let $\mathcal{V}_{\text{bnd}} = \mathcal{V} \setminus \mathcal{V}_{\text{fluid}}$ index boundary particles, including obstacles and domain boundaries. Given the first H observed frames, indexed by $1, \dots, H$, autoregressive prediction begins at $t = H$. At each step $t \geq H$, we define the ordered position and velocity histories of particle i as

$$\mathbf{X}_i^t = (\mathbf{x}_i^{t-H+1}, \dots, \mathbf{x}_i^t), \qquad \mathbf{V}_i^t = (\mathbf{v}_i^{t-H+2}, \dots, \mathbf{v}_i^t),$$

where velocities are obtained by finite differences, $\mathbf{v}_i^\tau = (\mathbf{x}_i^\tau - \mathbf{x}_i^{\tau-1})/\Delta t$ for $\tau \in \{t-H+2, \dots, t\}$. Thus H position frames provide $H-1$ velocity samples. Each particle also has static attributes $\xi_i = (\rho_i, \kappa_i)$, where $\rho_i \in \{\text{fluid}, \text{obstacle}, \text{boundary}\}$ specifies the particle's role and κ_i identifies its material. We collect these histories and attributes separately for fluid and prescribed particles:

$$P^t = \left\{ \left(\mathbf{X}_i^t, \mathbf{V}_i^t, \xi_i \right) \right\}_{i \in \mathcal{V}_{\text{fluid}}}, \qquad B^t = \left\{ \left(\mathbf{X}_i^t, \mathbf{V}_i^t, \xi_i \right) \right\}_{i \in \mathcal{V}_{\text{bnd}}}.$$

The fluid state P^t is advanced through model predictions, whereas the boundary state B^t is supplied from the known boundary conditions throughout the rollout. Let $\mathbf{a}^{\text{ext},t} \in \mathbb{R}^d$ denote the known external acceleration vector acting on the fluid particles, such as gravity, where d is the spatial dimension. Starting from $\widehat{P}^H = P^H$, an autoregressive rollout of horizon T follows

$$\widehat{P}^{H+k} = f_\theta\Big(\widehat{P}^{H+k-1}, B^{H+k-1}, \mathbf{a}^{\text{ext},H+k-1} \Big), \qquad k = 1, \dots, T. \tag{1}$$

Each transition predicts the next fluid positions and updates the history window accordingly, with velocities computed by finite differences and particle attributes kept unchanged.

4 Particle–Grid Co-Evolution Framework

CoLEP couples Lagrangian particle dynamics with a persistent Eulerian latent field G^t defined over a fixed background grid. While the fluid state P^t defined in Section 3 records the recent position and velocity histories and the attributes of individual particles, G^t is updated recurrently from its previous state and current particle information, retaining spatial context across simulation steps. We therefore augment the rollout state $\widehat{P}^t$ with G^t and define the coupled transition as

$$\big(\widehat{P}^{t+1}, G^{t+1} \big) = \mathcal{T}_\theta\big(\widehat{P}^t, G^t, B^t, \mathbf{a}^{\text{ext},t} \big). \tag{2}$$

where $\mathcal{T}_\theta$ denotes the complete coupled transition and θ collects all trainable parameters of the architecture described in Section 4.1.

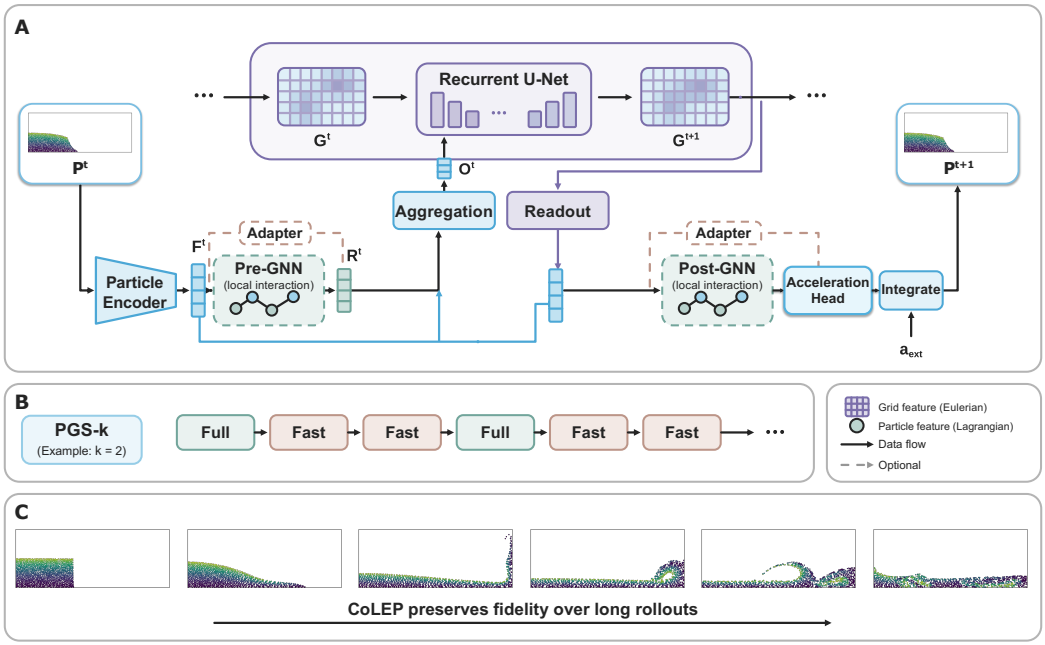


Figure 1: Overview of the CoLEP architecture and rollout workflow. At each step, particle histories are encoded into particle features $\mathbf{F}^t$, while a Pre-GNN produces local interaction residuals $\mathbf{R}^t$. Both are transferred to the Eulerian grid to form the current observation O^t, which is combined with the inherited latent field G^t and evolved by a recurrent U-Net to produce G^{t+1}. The updated field is then gathered back to the particles, fused with $\mathbf{F}^t$, refined by a Post-GNN, and decoded to advance the particle state to $\widehat{P}^{t+1}$. Thus, CoLEP jointly advances the recurrent state $(\widehat{P}^t, G^t) \to (\widehat{P}^{t+1}, G^{t+1})$.

4.1 Model Architecture

Our model first encodes each particle's state with an MLP and then uses a Pre-GNN to model local interactions. Particle-to-grid aggregation transfers the particle encodings and interaction residuals to the fixed grid, where a U-Net evolves the persistent Eulerian latent field. The updated feature is read out at particle positions and fused with their encodings; a Post-GNN then refines the fused features through local message passing before acceleration decoding and integration advance the particles.

I. Particle encoding and local interactions. At each step, we encode the particle histories and attributes in $\widehat{P}^t$ and B^t into a feature vector for each particle, using one shared encoder $\mathrm{MLP}_{\mathrm{node}}$ for fluid, obstacle, and boundary particles. For particle i, we concatenate its velocity history $\mathbf{V}_i^t$ with learned type and material embeddings. The shared encoder produces

$$\mathbf{F}_i^t = \mathrm{MLP}_{\mathrm{node}}\big([\mathbf{V}_i^t;\, T_{\mathrm{type}}[\rho_i];\, T_{\mathrm{mat}}[\kappa_i]]\big), \qquad i \in \mathcal{V}. \tag{3}$$

Here T_{type} and T_{mat} are trainable lookup tables indexed by ρ_i and κ_i, respectively, in the static attribute tuple $\xi_i = (\rho_i, \kappa_i)$ defined in Section 3. Collectively, $\mathbf{F}^t = \{\mathbf{F}_i^t\}_{i\in\mathcal{V}}$ is a feature field defined on the particles, which serves as the node input to the Pre-GNN in Fig. 1. Following LagrangeBench (Toshev et al., 2023b), we construct a radius graph E_R^t from the particles' current positions, connecting pairs within radius R. For each directed edge $j \to i$, let $\delta_{ij}^t = \mathbf{x}_j^t - \mathbf{x}_i^t$ be its relative displacement. An edge encoder maps $[\delta_{ij}^t/R;\, \|\delta_{ij}^t\|/R]$ to the initial edge feature $\boldsymbol{\phi}_{ji}^{t,(0)}$. Assume that the Pre-GNN consists of L_{pre} message-passing blocks. At block $\ell \in \{0, \dots, L_{\mathrm{pre}} - 1\}$, the edge and node update MLPs $\mathcal{E}_\ell$ and $\mathcal{U}_\ell$ compute

$$\boldsymbol{\phi}_{ji}^{t,(\ell+1)} = \boldsymbol{\phi}_{ji}^{t,(\ell)} + \mathcal{E}_\ell\Big([\mathbf{u}_i^{t,(\ell)};\, \mathbf{u}_j^{t,(\ell)};\, \boldsymbol{\phi}_{ji}^{t,(\ell)}]\Big), \quad \Delta\mathbf{R}_i^{t,(\ell)} = \mathcal{U}_\ell\Big([\mathbf{u}_i^{t,(\ell)};\, \textstyle\sum_{j:\,(j,i)\in E_R^t} \boldsymbol{\phi}_{ji}^{t,(\ell+1)}]\Big). \tag{4}$$

Here, $\mathbf{u}_i^{t,(\ell)} = \mathbf{F}_i^t + \mathbf{R}_i^{t,(\ell)}$, where $\mathbf{R}_i^{t,(\ell)}$ accumulates the residual updates to particle i from the first ℓ message-passing blocks:

$$\mathbf{R}_i^{t,(0)} = \mathbf{0}, \qquad \mathbf{R}_i^{t,(\ell+1)} = \mathbf{R}_i^{t,(\ell)} + \Delta\mathbf{R}_i^{t,(\ell)}, \quad \ell = 0, \dots, L_{\mathrm{pre}} - 1. \tag{5}$$

Then the Pre-GNN accumulates feature updates from local interactions over L_{pre} message-passing layers, yielding the interaction residual field $\mathbf{R}^t = \{\mathbf{R}_i^{t,(L_{\mathrm{pre}})}\}_{i\in\mathcal{V}}$, alongside the node encoding $\mathbf{F}^t$.

II. Particle-to-grid aggregation. To inform the Eulerian latent field, we project particle features onto the grid. Let w_{gi}^t be the interpolation weight from particle i at $\mathbf{x}_i^t$ to grid node g. The scatter operator Π^t acts on a particle feature field Z as $\Pi^t(Z)_g = \sum_{i\in\mathcal{V}} w_{gi}^t Z_i$. The interpolation weights $w_{gi}^t \geq 0$ are multilinear and satisfy $\sum_{g\in\mathcal{G}} w_{gi}^t = 1$. Utilizing a learned affine projection $\mathrm{Proj}_{\mathrm{grid}}$, the resulting grid observation is obtained by $O^t = \mathrm{Proj}_{\mathrm{grid}}([\Pi^t(\mathbf{F}^t);\, \Pi^t(\mathbf{R}^t)])$,

III. Eulerian feature update. We next update the Eulerian latent field by combining the current grid observation with the inherited field state. At the first prediction step, $t = H$, we initialize G^H from O^H, which aggregates particle features encoding the initial H observed frames. Starting from this initialization, the field advances at every step $t > H$ according to

$$G^{t+1} = \Phi_\theta\big([O^t;\, G^t]\big)\,, \tag{6}$$

where Φ_θ projects and fuses the two input fields and evolves the grid state with a multi-scale U-Net.

IV. Particle readout. We interpolate the updated grid field G^{t+1} back to the particles using Γ^t with the same weights as Π^t, giving $\Gamma^t(G^{t+1})_i = \sum_{g\in\mathcal{G}} w_{gi}^t G_g^{t+1}$. We fuse the interpolated grid features with the skip-connected particle encodings $\mathbf{F}^t$ via a learned projection $\widetilde{\mathbf{F}}^t = \mathrm{Proj}_{\mathrm{particle}}\big([\mathbf{F}^t;\, \Gamma^t(G^{t+1})]\big)$.

V. Particle state prediction. We finally decode the updated particle features into internal accelerations and integrate the total accelerations to update particle velocities and positions. The Post-GNN uses the same residual edge and node update structure as the Pre-GNN on the same radius graph E_R^t, with independent parameters and the fused particle features $\widetilde{\mathbf{F}}^t$ as node inputs. The acceleration decoder D_θ then maps the refined particle features to the internal accelerations:

$$\mathbf{F}^{t,\mathrm{post}} = \mathrm{Post}_\theta(\widetilde{\mathbf{F}}^t, E_R^t), \qquad \widehat{\mathbf{a}}^{\mathrm{int},t} = D_\theta(\mathbf{F}^{t,\mathrm{post}}). \tag{7}$$

For $i \in \mathcal{V}_{\mathrm{fluid}}$, we add the external acceleration vector $\mathbf{a}^{\mathrm{ext},t}$ defined in Section 3 to the predicted internal acceleration and apply semi-implicit integration:

$$\widehat{\mathbf{v}}_i^{t+1} = \mathbf{v}_i^t + \Delta t\big(\widehat{\mathbf{a}}_i^{\mathrm{int},t} + \mathbf{a}^{\mathrm{ext},t}\big), \qquad \widehat{\mathbf{x}}_i^{t+1} = \mathbf{x}_i^t + \Delta t\, \widehat{\mathbf{v}}_i^{t+1}. \tag{8}$$

Appending the predicted fluid positions to the fluid history, dropping the oldest frame, and recomputing velocities by finite differences produces $\widehat{P}^{t+1}$, with the particle attributes unchanged. Together with G^{t+1}, this advances the state to $t+1$, ready for the subsequent rollout step with prescribed boundary conditions B^{t+1}.

4.2 Periodic GNN Skipping

In the co-evolution framework of Section 4.1, particle–grid interactions are symmetrically mediated by two networks. The Pre-GNN extracts interaction residuals that are aggregated onto the grid. After the grid update, the grid features are interpolated back to the particles and fused with their node encodings; the Post-GNN then applies residual interaction refinements to these fused features before acceleration decoding. And running both GNNs introduces a severe bottleneck through dynamic radius-graph construction and deep message passing blocks.

This architecture offers a natural path to bypass graph overhead. Because the Eulerian grid evolves continuously with the particles, its latent memory inherently retains the recent interaction history of preceding steps. Leveraging this context alongside current node encodings enables accurate short-term prediction even when interaction updates are omitted. To exploit this property, we propose Periodic GNN Skipping (PGS), which repeats one complete update (Full) followed by k lightweight updates (Fast) that omit all graph computations while maintaining continuous particle–grid co-evolution.

Adapter-Based Fast Transitions. Directly bypassing both GNN modules disrupts the latent feature representations expected by the co-evolving model parameters θ. To compensate for these

omitted interaction residuals without altering downstream tensor dimensions, we introduce two lightweight adapters parameterized by $\phi = (\phi_g, \phi_p)$. For the grid update, we retain the direct node projection while zero-padding the uncomputed interaction channels to form the partial observation $O^t_{\text{Fast}} = \text{Proj}_{\text{grid}}([\Pi^t(\mathbf{F}^t); \mathbf{0}])$. The grid adapter then fuses this observation with the persistent Eulerian memory to substitute for the original Pre-GNN pathway $\widetilde{O}^t_{\text{Fast}} = \mathcal{A}^{\text{g}}_{\phi_g}(O^t_{\text{Fast}}, G^t)$, which is processed alongside G^t by the frozen U-Net to yield the updated grid state G^{t+1}_{Fast}. Symmetrically, the particle adapter bypasses the Post-GNN by mapping the current node encodings and interpolated grid memory directly into the frozen decoder's latent space:

$$\mathbf{F}^t_{\text{dec}} = \mathcal{A}^{\text{p}}_{\phi_p}\left([\mathbf{F}^t; \Gamma^t(G^{t+1}_{\text{Fast}})]\right), \qquad \widehat{\mathbf{a}}^{\text{int},t}_{\text{Fast}} = D_\theta(\mathbf{F}^t_{\text{dec}}), \tag{9}$$

from which particle positions are advanced via standard integration equation 8.

Lightweight Adapter Training. Freezing the pretrained co-evolving model θ, we optimize exclusively the adapters ϕ through a two-phase progressive scheme. First, using states reached by Full rollouts, we optimize the adapters to match the Full model's intermediate features and acceleration predictions from identical inputs.

We then fine-tune the adapters within autoregressive PGS-k rollouts, starting with a single-cycle Full–Fast handoff and progressing to longer mixed rollouts. This phase combines distillation with ground-truth supervision, training the adapters to produce states that subsequent frozen Full transitions can use. In practice, adaptation can be completed in a short time, as the adapters are lightweight and the pretrained backbone remains frozen throughout training. Detailed loss formulations and training schedules are provided in Appendix A.

5 Experiments

Our experiments evaluate three aspects of CoLEP: long-horizon predictive fidelity across diverse particle-dynamics regimes, the contribution of the persistent Eulerian state beyond transient particle–grid representations, and the extent to which this persistent context enables less frequent graph interaction updates for improved accuracy–inference-cost trade-offs. Dataset-specific details, complete configurations, and additional results are provided in the appendix.

5.1 Experimental Settings

Datasets. We evaluate CoLEP on four particle-simulation problems spanning complementary dynamical regimes. *DAM-2D* and *TGV-2D* follow the LagrangeBench protocols (Toshev et al., 2023b). *Water-3D* is a large-scale 3D SPH free-surface benchmark (Sanchez-Gonzalez et al., 2020). We additionally construct *MovingObstacle-2D* with PhiFlow (Holl & Thuerey, 2024), extending the dam-break setting with a translating and tilting internal obstacle to create continuously changing fluid–boundary interactions. Dataset sizes, rollout horizons, and split details are provided in Appendix B.1.

Baselines. We compare against GNS (Sanchez-Gonzalez et al., 2020), NeuralMPM (Rochman-Sharabi et al., 2025), and CoRGI (Ji et al., 2026), representing particle-based, grid-based, and hybrid particle–grid neural simulators, respectively. We use the open-source GNS implementation from LagrangeBench (Toshev et al., 2023b). NeuralMPM is evaluated only on the 2D problems, following the scope of its available implementation.

Evaluation metrics. Following LagrangeBench and CoRGI (Toshev et al., 2023b; Ji et al., 2026), we evaluate rollout quality using three complementary metrics. Position MSE measures particle-wise trajectory accuracy by comparing predicted and reference coordinates with particle identities aligned. Kinetic-energy MSE (KE-MSE) measures discrepancies in the evolution of the system's total kinetic energy, providing a complementary view of dynamical fidelity beyond position error. Sinkhorn distance compares the predicted and reference particle distributions without requiring particle-wise correspondence, and therefore captures differences in the overall spatial distribution and morphology of the particle set.

We report short-horizon accuracy over 20-step autoregressive rollouts using MSE_{20} and KE-MSE_{20}, and long-horizon accuracy over the complete rollout using MSE_{full}, $\text{KE-MSE}_{\text{full}}$, and $\text{Sinkhorn}_{\text{full}}$. Complete metric definitions and dataset-specific evaluation scopes are provided in Appendix B.2.

Table 1: Short- and full-rollout predictive accuracy across four particle-simulation benchmarks. Lower is better for all metrics. Bold denotes the best reported result within each dataset and metric.

		20-step rollout		**Full rollout**		
Dataset	Model	$\mathrm{MSE}_{20} \downarrow$	$\mathrm{KE\text{-}MSE}_{20} \downarrow$	$\mathrm{MSE}_{\mathrm{full}} \downarrow$	$\mathrm{KE\text{-}MSE}_{\mathrm{full}} \downarrow$	$\mathrm{Sinkhorn}_{\mathrm{full}} \downarrow$
DAM-2D	GNS	4.13×10^{-5}	2.27×10^{-5}	1.59×10^{-1}	4.00×10^{-3}	1.93×10^{-2}
	NeuralMPM	2.84×10^{-4}	3.48×10^{-3}	1.26×10^{-1}	8.81×10^{-3}	2.49×10^{-2}
	CoRGI	2.24×10^{-5}	6.90×10^{-5}	9.04×10^{-2}	1.08×10^{-3}	1.24×10^{-2}
	CoLEP (Ours)	$\mathbf{1.49\times10^{-5}}$	$\mathbf{4.09\times10^{-6}}$	$\mathbf{2.57\times10^{-2}}$	$\mathbf{7.67\times10^{-5}}$	$\mathbf{2.84\times10^{-4}}$
TGV-2D	GNS	7.08×10^{-6}	2.45×10^{-7}	5.75×10^{-4}	2.40×10^{-7}	1.02×10^{-6}
	NeuralMPM	2.32×10^{-5}	7.12×10^{-6}	9.74×10^{-4}	4.89×10^{-6}	9.96×10^{-6}
	CoRGI	5.94×10^{-6}	1.43×10^{-7}	4.79×10^{-4}	1.20×10^{-7}	6.13×10^{-7}
	CoLEP (Ours)	$\mathbf{3.48\times10^{-6}}$	$\mathbf{4.99\times10^{-8}}$	$\mathbf{3.17\times10^{-4}}$	$\mathbf{4.85\times10^{-8}}$	$\mathbf{2.20\times10^{-7}}$
Moving Obstacle-2D	GNS	9.25×10^{-6}	6.65×10^{-7}	1.10×10^{-2}	2.74×10^{-5}	1.30×10^{-2}
	NeuralMPM	$\mathbf{2.67\times10^{-7}}$	$\mathbf{1.20\times10^{-8}}$	2.97×10^{-3}	3.24×10^{-6}	2.13×10^{-3}
	CoRGI	7.15×10^{-6}	4.67×10^{-7}	3.14×10^{-3}	4.68×10^{-6}	2.23×10^{-3}
	CoLEP (Ours)	1.91×10^{-6}	1.53×10^{-7}	$\mathbf{7.81\times10^{-4}}$	$\mathbf{5.18\times10^{-7}}$	$\mathbf{2.49\times10^{-4}}$
Water-3D	GNS	5.26×10^{-5}	2.75×10^{-4}	7.88×10^{-2}	1.67×10^{-2}	8.13×10^{-2}
	CoRGI	3.68×10^{-5}	2.26×10^{-4}	6.58×10^{-2}	7.01×10^{-3}	5.30×10^{-2}
	CoLEP (Ours)	$\mathbf{3.50\times10^{-6}}$	$\mathbf{9.87\times10^{-6}}$	$\mathbf{1.44\times10^{-2}}$	$\mathbf{3.33\times10^{-5}}$	$\mathbf{9.11\times10^{-3}}$

Implementation details, model architectures, and training settings for all evaluated models are provided in Appendices B.3, B.4, and B.5, respectively.

5.2 Overall Predictive Accuracy

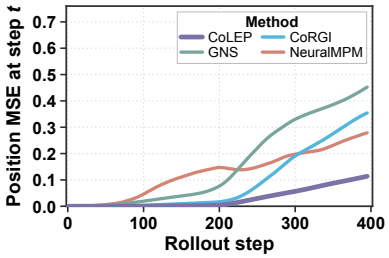


Figure 2: Instantaneous position MSE over autoregressive rollout steps on DAM-2D for CoLEP and baselines. Errors are evaluated independently at each step and are not cumulative.

Table 1 compares predictive accuracy over 20-step and complete autoregressive rollouts. Across all four benchmarks, CoLEP achieves the lowest reported full-rollout position MSE, kinetic-energy MSE, and Sinkhorn distance. Relative to the strongest reported baseline on each task, CoLEP reduces full-rollout position MSE by approximately 72% on DAM-2D, 34% on TGV-2D, 74% on MovingObstacle-2D, and 78% on Water-3D.

The results also reveal a clear distinction between short- and long-horizon accuracy. On MovingObstacle-2D, NeuralMPM achieves the lowest reported 20-step position MSE, whereas its full-rollout MSE is approximately $3.8\times$ that of CoLEP. Thus, strong short-horizon accuracy does not necessarily translate into accurate long autoregressive rollout, and the advantage of CoLEP becomes more pronounced over extended prediction horizons.

We next examine how this separation develops over the course of the rollout.

5.3 Long-Horizon Error Growth

We examine how the predictive advantage of CoLEP develops over long autoregressive rollouts using stepwise error growth and late-stage particle configurations. Additional TGV-2D results are provided in Appendix B.6.

Figure 2 shows the instantaneous position MSE over the complete 395-step DAM-2D rollout. The methods remain close early in the trajectory, but their errors progressively separate as the rollout continues. Around step 200, CoLEP maintains an MSE of approximately 0.004; by the final step, its error is 0.114, compared with 0.279–0.452 for the baselines. Since the error is evaluated

independently at each step, this widening gap reflects increasing disagreement in the predicted states rather than accumulation in the metric itself.

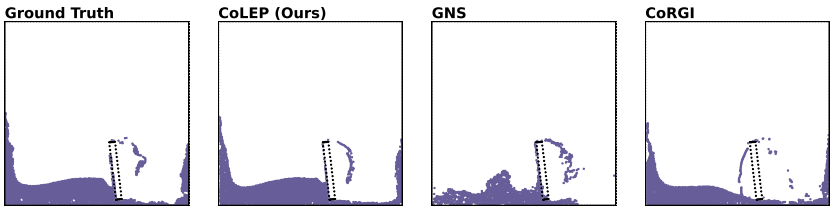
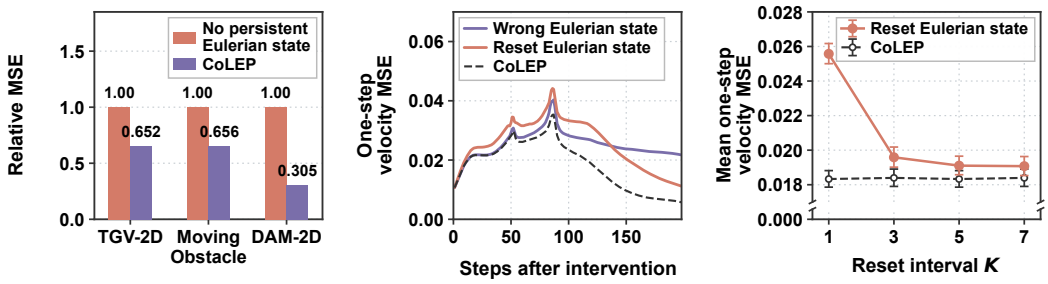


Figure 3: Comparison on the MovingObstacle-2D scenario at rollout step 240. CoLEP more closely reproduces the reference fluid bulk and free-surface geometry, whereas GNS and CoRGI exhibit larger deviations in the fluid configuration around the moving obstacle.

Figure 3 provides a complementary qualitative example on MovingObstacle-2D. At rollout step 240, CoLEP better preserves the fluid extent and free-surface geometry, while GNS and CoRGI show larger structural distortion around the moving obstacle. More details are provided in Appendix B.6.

5.4 Contribution of Persistent Eulerian Field

At each rollout step, CoLEP updates the Eulerian field using both the current particle-derived grid observation and the latent field inherited from previous steps. We examine whether this inherited state provides predictive information beyond the transient particle–grid representation, and whether the information it carries is trajectory dependent.



(a) Effect of state persistence

(b) Temporal error dynamics with $K = 1$

(c) Effect of reset interval K

Figure 4: Role of the persistent Eulerian state. (a) Full-rollout test position MSE on 3 datasets, normalized by the corresponding model without a persistent Eulerian state. (b) One-step velocity MSE on DAM-2D for CoLEP and for mismatched-state and reset interventions, both applied every step ($K = 1$). (c) Mean one-step velocity MSE on DAM-2D for CoLEP and reset interventions at different intervals K.

Effect of temporal persistence. We first compare CoLEP with no-persistent-Eulerian grid state variants that retain the same per-step particle–grid pathway but reconstruct the grid representation from the current particle state at every step. As shown in Figure 4a, removing persistence consistently degrades full-rollout prediction. Relative to the corresponding no-persistent-Eulerian-field variants, CoLEP reduces full-rollout position MSE by approximately 35% on TGV-2D, 34% on MovingObstacle-2D, and 69.5% on DAM-2D. Thus, the improvement cannot be attributed solely to per-step grid processing; carrying the Eulerian state across time provides additional predictive context.

Trajectory-dependent information in Eulerian State. We next intervene directly on the inherited state of a trained CoLEP model on DAM-2D. We either replace it with the time-aligned Eulerian field from a different trajectory or periodically reinitialize it from the current particle state.

Figure 4b and 4c show that both interventions degrade prediction. Using mismatched state increases the post-intervention velocity error by about 38%, while resetting the state at every step increases it

by about 40%. The reset penalty decreases rapidly as the interval between resets grows, indicating that useful temporal context is progressively rebuilt through continued particle–grid evolution. More results are included in Appendix B.7.

Together, these results show that the Eulerian field is not merely a transient spatial representation: its persistence carries trajectory-dependent context across rollout steps and improves subsequent particle prediction.

5.5 Persistent Eulerian State Enables Sparse GNN Evaluation

We next examine the role of persistent memory when graph interactions are evaluated less frequently. On DAM-2D, we compare Full and PGS-1 execution with and without cross-step Eulerian memory.

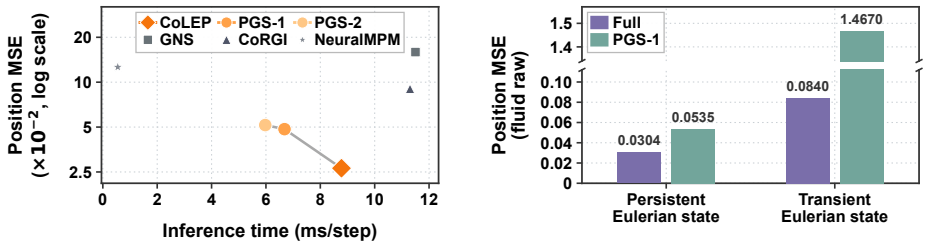


Figure 5: Accuracy–efficiency trade-offs and persistent-state comparisons on DAM-2D. (a) Full-rollout position MSE versus inference time for CoLEP, PGS variants, and baselines; points closer to the lower-left corner indicate better accuracy and efficiency. (b) Position MSE under Full and PGS-1 execution for models with a persistent Eulerian state and models with a transient Eulerian state.

Figure 5 shows that persistence is particularly important under sparse GNN evaluation. With memory, switching from Full to PGS-1 increases the full-rollout position MSE from 0.0304 to 0.0535. Without memory, the same change increases the error from 0.0840 to 1.4670. Thus, removing fresh graph interactions is far more damaging when the Eulerian state cannot carry context across steps, supporting the use of persistent memory for graph-free intermediate updates.

We then evaluate the resulting accuracy–efficiency trade-off on DAM-2D. CoLEP executes Full at every step, PGS-1 alternates Full and Fast, and PGS-2 performs two Fast steps after each Full step. Fast steps omit graph construction and neighbor message passing, while continuing to update both the particle state and Eulerian memory through the learned graph-free adapters.

Figure 5 reports full-rollout position MSE against per-step wall-clock latency. Reducing the frequency of Full evaluations lowers inference cost while gradually trading off rollout accuracy. Together, these results show that persistent Eulerian memory makes sparser graph interaction updates viable, allowing PGS to improve inference efficiency without recomputing particle interactions at every step. See Appendix B.8 for more results.

6 Conclusion

We introduced CoLEP, a co-evolving Lagrangian–Eulerian particle simulator that turns the Eulerian grid from a transient spatial representation into a persistent latent state. By accumulating and feeding back spatial context across rollout steps, CoLEP improves long-horizon prediction, while memory ablations and interventions confirm the value of this inherited state beyond per-step particle–grid processing.

Building on this persistent representation, PGS interleaves full graph-based interaction updates with learned graph-free Fast steps, enabling a favorable accuracy–inference-cost trade-off. These results suggest that interaction context need not be reconstructed from scratch at every step, but can instead be accumulated and reused through an evolving Eulerian state. The current formulation does not impose explicit conservation or stability guarantees, and PGS uses a fixed execution schedule with separately trained Fast adapters; adaptive interaction scheduling remains an interesting direction for future work.

AI USE STATEMENT

Large language models (LLMs) were used to polish the language of the manuscript, including sentence rephrasing, grammar checking, and improvements to readability, clarity, and flow. They were not used for research ideation, methodology development or experimental design. The authors reviewed the revised text and take full responsibility for the content of the manuscript.

ETHICS STATEMENT

This work studies neural particle simulation using numerical simulation datasets. It does not involve human participants or personal data. We do not anticipate specific ethical concerns arising from the methods and experiments presented in this paper.

REPRODUCIBILITY STATEMENT

We will publicly release the source code for CoLEP to support reproducibility and further research. The model architecture, implementation details, dataset descriptions, and evaluation protocols are provided in the main text and appendices.

REFERENCES

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/3147da8ab4a0437c15ef51a5cc7f2dc4-Abstract.html.

Johannes Brandstetter, Rob Hesselink, Elise van der Pol, Erik J. Bekkers, and Max Welling. Geometric and physical quantities improve $E(3)$-equivariant message passing. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=_xwr8gOBeV1.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=vSix3HPYKSU.

Nuttapong Chentanez, Stefan Jeschke, Matthias Müller, and Miles Macklin. Learning physics with a hierarchical graph network. *Computer Graphics Forum*, 41(8):283–292, 2022. doi: 10.1111/cgf.14643. URL https://doi.org/10.1111/cgf.14643.

Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouvé, and Gabriel Peyré. Interpolating between optimal transport and MMD using Sinkhorn divergences. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 2681–2690. PMLR, 2019. URL https://proceedings.mlr.press/v89/feydy19a.html.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html.

Philipp Holl and Nils Thuerey. PhiFlow: Differentiable simulations for PyTorch, TensorFlow and JAX. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 18515–18546, 2024. URL https://proceedings.mlr.press/v235/holl24a.html.

Ethan Ji, Yuanzhou Chen, Arush Ramteke, Fang Sun, Tianrun Yu, Jai Parera, Peipei Ping, Wei Wang, and Yizhou Sun. CoRGI: GNNs with convolutional residual global interactions for lagrangian simulation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 2026–2037, 2026. doi: 10.1145/3770855.3818147. URL https://doi.org/10.1145/3770855.3818147.

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, pp. 24:1–24:52, 2016. doi: 10.1145/2897826.2927348. URL https://alexey.stomakhin.com/research/mpm.html.

Ruoyan Li, Wei Wang, and Yizhou Sun. Hybrid lagrangian–eulerian model for lagrangian fluid simulation. *arXiv preprint arXiv:2608.01164*, 2026. URL https://arxiv.org/abs/2608.01164.

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2019. URL https://jiajunwu.com/papers/dpi_iclr.pdf.

Bjoern List, Li-Wei Chen, Kartik Bali, and Nils Thuerey. Differentiability in unrolled training of neural physics simulators on transient dynamics. *Computer Methods in Applied Mechanics and Engineering*, 433:117441, 2025. doi: 10.1016/j.cma.2024.117441.

Michael McCabe, Peter Harrington, Shashank Subramanian, and Jed Brown. Towards stability of autoregressive neural operators. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=RFfUUtKYOG.

J. J. Monaghan. Smoothed particle hydrodynamics and its diverse applications. *Annual Review of Fluid Mechanics*, 44:323–346, 2012. doi: 10.1146/annurev-fluid-120710-101220.

Lukas Prantl, Benjamin Ummenhofer, Vladlen Koltun, and Nils Thuerey. Guaranteed conservation of momentum for learning particle-based fluid dynamics. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/2dd7f33ffbb59b4ff987be5442a13016-Abstract-Conference.html.

Omer Rochman-Sharabi, Sacha Lewin, and Gilles Louppe. A neural material point method for particle-based emulation. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=zSK81A2hxQ.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, volume 9351 of *Lecture Notes in Computer Science*, pp. 234–241. Springer, 2015. URL https://arxiv.org/abs/1505.04597.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 8459–8468, 2020. URL https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html.

Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. $E(n)$ equivariant graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9323–9332, 2021. URL https://proceedings.mlr.press/v139/satorras21a.html.

Deborah Sulsky, Zhen Chen, and Howard L. Schreyer. A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, 118(1–2):179–196, 1994. doi: 10.1016/0045-7825(94)90112-0.

Artur P. Toshev, Gianluca Galletti, Johannes Brandstetter, Stefan Adami, and Nikolaus A. Adams. Learning lagrangian fluid mechanics with $E(3)$-equivariant graph neural networks. In *Geometric Science of Information*, volume 14072 of *Lecture Notes in Computer Science*, pp. 332–341, 2023a. doi: 10.1007/978-3-031-38299-4_35. URL https://arxiv.org/abs/2305.15603.

Artur P. Toshev, Gianluca Galletti, Fabian Fritz, Stefan Adami, and Nikolaus A. Adams. Lagrangebench: A lagrangian fluid mechanics benchmarking suite. In *Advances in Neural Information Processing Systems*, volume 36, 2023b. URL https://github.com/tumaer/lagrangebench.

Artur P. Toshev, Jonas A. Erbesdobler, Nikolaus A. Adams, and Johannes Brandstetter. Neural SPH: Improved neural modeling of lagrangian fluid dynamics. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=Pbey7LqBRl.

A Periodic GNN Skipping Details

We freeze the pretrained CoLEP parameters θ and optimize only the grid-observation and particle-readout adapters $\phi = (\phi_g, \phi_p)$ introduced in Section 4.2. On each Fast step, a detached Full evaluation provides distillation targets from the same current particle–grid state $(\widehat{P}^t, G^t)$ and the same boundary and external-force inputs.

Two-phase adaptation. We first train the adapters by single-step matching on detached states obtained from Full rollouts. This phase uses distillation only, without ground-truth supervision.

We then adapt the adapters within autoregressive PGS-k rollouts, combining distillation with ground-truth supervision. This phase begins with a single-cycle handoff: after a Full step, the model executes k Fast steps followed by another Full step. Training subsequently uses longer mixed rollouts according to the rollout-length curriculum. The initial state is prepared by a detached mixed-rollout prefix ending in Full; thereafter, both particle histories and the Eulerian state follow the selected Full/Fast transitions.

Training objectives. The matching loss compares normalized acceleration, projected grid observation, updated Eulerian state, gathered grid features, and particle-readout features:

$$\ell_{\mathrm{m}}^t = \lambda d_a^t + 0.1 d_O^t + 0.25 d_G^t + 0.25 d_g^t + 0.05 d_h^t. \tag{10}$$

Here d_a is normalized-acceleration MSE, while d_O, d_G, d_g, d_h are feature MSEs normalized by their fixed teacher scales. All particle-level terms are restricted to fluid particles. We use $\lambda = 0.5$ during single-step matching and $\lambda = 0.25$ during both the handoff and longer-rollout segments.

Single-step training minimizes ℓ_{m}^t alone. For a mixed rollout of length T, the objective is

$$\mathcal{L}_{\mathrm{roll}} = \frac{1}{T}\left(\sum_{j \in \mathcal{I}_{\mathrm{Fast}}} \ell_{\mathrm{m}}^{t+j} + \sum_{j=0}^{T-1} \ell_{\mathrm{c}}^{t+j}\right), \tag{11}$$

where $\mathcal{I}_{\mathrm{Fast}}$ contains the Fast-step indices. The corrective-acceleration loss ℓ_{c} compares the selected route's normalized acceleration with the target computed from the current predicted particle state and the ground-truth next position. It has unit weight on both Full and Fast steps. The handoff segment uses $T = k + 1$; subsequent segments use the sampled mixed-rollout length.

Gradient handling. Distillation targets are detached, whereas selected Full transitions remain differentiable with respect to their inputs, allowing losses to train preceding Fast steps without updating θ. Truncated backpropagation detaches the carried state without resetting particle histories or Eulerian-state values. Because O is computed before the adapters, d_O has no direct gradient to the current-step adapters; in mixed rollouts, it can influence earlier Fast steps through the current particle state.

Optimization settings and training budgets are provided in Section B.8.

B Experiment

B.1 Datasets and Data Generation

Public benchmarks. We use three public datasets, DAM-2D, TGV-2D, and Water-3D, and follow the experimental settings of LagrangeBench (Toshev et al., 2023b). DAM-2D and TGV-2D are from the LagrangeBench benchmark suite, while Water-3D is from the GNS dataset collection (Sanchez-Gonzalez et al., 2020).

MovingObstacle construction. We initialize two stationary liquid blocks in the upper region and lower-left corner of a two-dimensional box. Their sizes and the upper block's position are randomized. Simulations use PhiFlow on a 64×64 grid over $[0, 64]^2$, with fixed gravity $(0, -9.81)$. An internal rectangular obstacle of size 2×20 undergoes periodic translation with amplitudes $(4, 1)$ and period $3.2\,\mathrm{s}$, together with sinusoidal tilting of amplitude 8° and period $4.8\,\mathrm{s}$. Its reference center and tilt are sampled from $[24, 40] \times [12, 26]$ and $[-30^\circ, 30^\circ]$, respectively, with a randomized motion phase. Each trajectory contains 4480 fluid particles and 300 frames at $0.016\,\mathrm{s}$ intervals. We generate 1000 training, 50 validation, and 50 test trajectories. Positions are normalized to $[0, 1]^2$, and the prescribed boundary geometry is supplied throughout the rollout.

Dataset statistics and splits. Table 2 summarizes the spatial dimension, particle counts, stored trajectory lengths, full-rollout horizons, and train/validation/test splits for all four datasets. Each rollout starts from six observed frames and predicts all remaining frames autoregressively. Test trajectories are not used for within-run checkpoint selection or bandwidth/threshold tuning.

Table 2: Dataset statistics and splits. Full-rollout horizons count the autoregressive predictions remaining after the six input frames.

Dataset	Dim.	Particles per trajectory	Stored frames	Full rollout	Train / val. / test
DAM-2D	2-D	5,000 fluid + 740 wall	401	395	50 / 25 / 25
TGV-2D	2-D	2,500	126	120	100 / 50 / 50
Water-3D	3-D	4,096–13,248	801	795	1,000 / 100 / 100
MovingObstacle	2-D	4,480 fluid + 320 boundary	300	294	1,000 / 50 / 50

Particle-count ranges indicate the minimum and maximum across the corresponding dataset. Wall and boundary particles provide prescribed geometric context and are listed separately from predicted fluid particles.

B.2 Evaluation Metrics

We evaluate position error, kinetic-energy error, and particle-distribution error, following the complementary criteria in LagrangeBench (Toshev et al., 2023b). Let $\mathcal{D}_{\mathrm{test}}$ denote the evaluated test rollouts: independently initialized windows for the 20-step metrics and complete test trajectories for the full-rollout metrics. Let $\mathcal{I}_\tau$ denote the particle indices included in the position and distribution metrics for rollout τ. On DAM-2D, this set contains all $5{,}740$ original nodes, including 740 static wall nodes whose positions are supplied from the test data. On the other benchmarks, we evaluate fluid particles and exclude prescribed boundary nodes and padding. For a rollout horizon T, position MSE is defined as

$$\mathrm{MSE}_T = \frac{1}{|\mathcal{D}_{\mathrm{test}}|} \sum_{\tau \in \mathcal{D}_{\mathrm{test}}} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N_\tau d} \sum_{i \in \mathcal{I}_\tau} \left\| \mathrm{wrap}\left(\widehat{\mathbf{x}}_i^{\tau,t} - \mathbf{x}_i^{\tau,t}\right) \right\|_2^2, \tag{12}$$

where $N_\tau = |\mathcal{I}_\tau|$, d is the spatial dimension, and wrap follows the displacement convention of the reference evaluator: it applies the minimum-image convention on axes treated as periodic and is the identity otherwise. We use $T = 20$ for MSE_{20} and the complete autoregressive horizon for $\mathrm{MSE}_{\mathrm{full}}$.

For kinetic energy, let $\mathcal{V}_\tau$ denote the fluid or material particles, excluding prescribed boundary and wall nodes. Following the physical energy definition in LagrangeBench (Toshev et al., 2023b), the predicted and reference total kinetic energies are

$$\widehat{E}^{\tau,t} = \frac{1}{2} \sum_{i \in \mathcal{V}_\tau} m_i \left\| \widehat{\mathbf{v}}_i^{\tau,t} \right\|_2^2, \qquad E^{\tau,t} = \frac{1}{2} \sum_{i \in \mathcal{V}_\tau} m_i \left\| \mathbf{v}_i^{\tau,t} \right\|_2^2. \tag{13}$$

For a rollout of T predicted positions, we use the $T - 1$ velocity intervals between consecutive predicted positions; the interval from the last observed history frame to the first prediction is excluded. The kinetic-energy error is

$$\text{KE-MSE}_T = \frac{1}{|\mathcal{D}_{\mathrm{test}}|} \sum_{\tau \in \mathcal{D}_{\mathrm{test}}} \frac{1}{T-1} \sum_{t=1}^{T-1} \left(\widehat{E}^{\tau,t} - E^{\tau,t} \right)^2. \tag{14}$$

Here $\widehat{\mathbf{v}}_i^{\tau,t} = \mathrm{wrap}(\widehat{\mathbf{x}}_i^{\tau,t+1} - \widehat{\mathbf{x}}_i^{\tau,t})/\Delta t$, and reference velocities are computed analogously. The timestep Δt is the time between stored frames, including the simulation output stride. We use

$m_i = (\Delta x)^d$ under the unit-reference-density convention, with Δx taken from the dataset metadata. All reported KE-MSE values include the factor $1/2$ in the energy definition.

At rollout step t, we represent the predicted and reference particle configurations as empirical measures

$$\widehat{\mu}^{\tau,t} = \sum_{i \in \mathcal{I}_\tau} w_i \delta_{\widehat{\mathbf{x}}_i^{\tau,t}}, \qquad \mu^{\tau,t} = \sum_{i \in \mathcal{I}_\tau} w_i \delta_{\mathbf{x}_i^{\tau,t}}, \tag{15}$$

where $w_i = 1/|\mathcal{I}_\tau|$ assigns equal weight to each particle in the dataset-specific evaluation set. Let T_τ denote the full rollout horizon of trajectory τ. The full-rollout Sinkhorn metric is

$$\mathrm{Sinkhorn}_{\mathrm{full}} = \frac{1}{|\mathcal{D}_{\mathrm{test}}|} \sum_{\tau \in \mathcal{D}_{\mathrm{test}}} \frac{1}{T_\tau} \sum_{t=1}^{T_\tau} S_{\varepsilon_{\tau,t}} \left(\widehat{\mu}^{\tau,t}, \mu^{\tau,t} \right). \tag{16}$$

We use the debiased Sinkhorn divergence (Feydy et al., 2019), as implemented in the LagrangeBench evaluator (Toshev et al., 2023b):

$$\begin{aligned} S_\varepsilon(\widehat{\mu}, \mu) = \mathrm{OT}_\varepsilon(\widehat{\mu}, \mu) \\ - \tfrac{1}{2}\mathrm{OT}_\varepsilon(\widehat{\mu}, \widehat{\mu}) - \tfrac{1}{2}\mathrm{OT}_\varepsilon(\mu, \mu), \end{aligned}$$

where OT_ε denotes the entropy-regularized optimal transport cost. The ground cost is $c(\mathbf{x}, \mathbf{y}) = \| \mathrm{wrap}(\mathbf{x} - \mathbf{y}) \|_2^2$, using the same displacement convention as position MSE. For each frame, the default regularization in `ott-jax` 0.4.6 is $\varepsilon_{\tau,t} = 0.05 \ \mathrm{mean}_{i,j \in \mathcal{I}_\tau} \, c(\widehat{\mathbf{x}}_i^{\tau,t}, \mathbf{x}_j^{\tau,t})$, shared by the cross and two self-comparison problems. We use the LagrangeBench evaluation code at commit `b880a6c` with `ott-jax` 0.4.6 and a Sinkhorn convergence threshold of 10^{-4}; all predicted frames are included in the rollout average.

For methods evaluated with multiple independently trained seeds, we first average the errors for the same test trajectory:

$$\widetilde{e}_\tau^{t,(m)} = \frac{1}{|\mathcal{S}_m|} \sum_{s \in \mathcal{S}_m} e_\tau^{t,(m,s)}, \tag{17}$$

where $\mathcal{S}_m$ denotes the set of seeds available for method m. The value plotted at rollout step t is then

$$\overline{e}^{t,(m)} = \frac{1}{|\mathcal{D}_{\mathrm{test}}|} \sum_{\tau \in \mathcal{D}_{\mathrm{test}}} \widetilde{e}_\tau^{t,(m)}. \tag{18}$$

For TGV-2D, $|\mathcal{D}_{\mathrm{test}}| = 50$ and $t \in \{1, \ldots, 120\}$. Thus, each point on the curve measures the mean instantaneous coordinate error at one fixed rollout step; it is not a cumulative error. The shaded region shows a pointwise, normal-approximation 95% confidence interval across test trajectories, $\overline{e}^{t,(m)} \pm 1.96 \, s_t^{(m)} / \sqrt{|\mathcal{D}_{\mathrm{test}}|}$, where $s_t^{(m)}$ is the sample standard deviation of the seed-averaged trajectory errors $\widetilde{e}_\tau^{t,(m)}$. The lower bound is clipped at zero for the linear-scale DAM-2D plot and at a positive numerical floor for the TGV-2D plot. For DAM-2D, $|\mathcal{D}_{\mathrm{test}}| = 25$ and $t \in \{1, \ldots, 395\}$. The 5,740-node scope comprises 5,000 dynamic fluid nodes and 740 static wall nodes. NeuralMPM predicts the fluid nodes; the wall positions are restored from their static test-data coordinates. Its curve uses the validation-selected 10,000-step checkpoint, consistent with Table 1; the other methods likewise use validation-selected checkpoints. The curves are not temporally smoothed.

B.3 Implementation Details

The experiments were conducted on two servers. The first was equipped with an AMD Ryzen Threadripper PRO 3955WX CPU (16 cores) and four NVIDIA GeForce RTX 4090 GPUs (24 GB of memory each). The second was equipped with two AMD EPYC 9654 CPUs (96 cores each) and four NVIDIA A100-SXM4 GPUs (40 GB of memory each).

B.4 Network Architecture

Particle inputs and outputs. Each particle is represented by five velocity samples obtained from six consecutive position frames, together with learned particle-type and material embeddings and

dataset-dependent boundary features. The resulting input dimension is 26 for TGV-2D, 44 for DAM-2D and MovingObstacle, and 61 for Water-3D. A shared MLP maps these inputs to 128-dimensional particle encodings. The model predicts normalized accelerations, which are converted to physical units for particle integration. Only fluid particles are advanced by the learned prediction; boundary and obstacle states follow their prescribed trajectories.

Backbone configuration. The Full model uses five message-passing layers in each of the Pre-GNN and Post-GNN, with 128-dimensional node and edge features. The two processors share the current radius graph but have independent parameters.

Particle encodings and Pre-GNN interaction residuals are scattered onto the grid and projected, together with support and execution-mode channels, into a 130-channel observation. The recurrent Eulerian network combines this observation with the inherited 96-channel field and produces an updated 96-channel field. Its U-Net uses three spatial scales with channel widths $(96, 128, 192)$, using 2-D convolutions for the two-dimensional datasets and 3-D convolutions for Water-3D.

The updated field is both carried to the next step and gathered back to the particles. Each gathered 96-dimensional feature is concatenated with the original 128-dimensional particle encoding and projected to 128 dimensions. The Post-GNN refines these features before an MLP decodes them into d acceleration components, where d is the spatial dimension.

Spatial grid configuration. The Eulerian field is stored on a fixed background grid. Grid resolution, spatial spacing, interaction radius, and boundary conditions are specified for each dataset, while the network feature widths above remain unchanged. TGV-2D uses periodic boundaries in both spatial directions; the other datasets use their prescribed container and obstacle geometry.

Grid shapes are reported in tensor order, (H, W) in 2-D and (D, H, W) in 3-D, including the configured halo cells on both sides of each axis. The Eulerian field is maintained over this complete computational grid.

B.5 Model Training Settings

GNS. We use the JAX implementation of GNS provided by LagrangeBench (Toshev et al., 2023b), following its training pipeline and adapting the data interface to each benchmark.

CoRGI. We use the authors' released JAX implementation of CoRGI (Ji et al., 2026). We follow the training procedure in the released code and allocate a budget of $500{,}000$ optimizer updates.

NeuralMPM. We use the authors' official implementation of NeuralMPM (Rochman-Sharabi et al., 2025), following its training pipeline with dataset-specific adaptations to the input representation and boundary conditions.

CoLEP. We implement CoLEP in PyTorch and train it using AdamW with a cosine learning-rate schedule and gradient-norm clipping at 1. The training objective is the mean-squared error of normalized particle accelerations. As in GNS (Sanchez-Gonzalez et al., 2020) and CoRGI (Ji et al., 2026), we inject noise into the input particle histories during training. During multi-step training, predicted particle states are fed back autoregressively, and the persistent Eulerian state is carried across successive steps. We use backpropagation through time, allowing gradients to propagate through both the particle-state updates and the Eulerian-state updates within each backpropagation window. Training budgets, batch sizes, rollout schedules, and backpropagation window lengths are dataset-specific.

Ablation studies. All architectural ablations, including variants of the persistent Eulerian state, the number of GNN layers, and the Pre-GNN and Post-GNN modules, follow the same training settings as CoLEP on the corresponding dataset. We keep the optimization procedure, training budget, and data splits fixed, varying only the component under investigation.

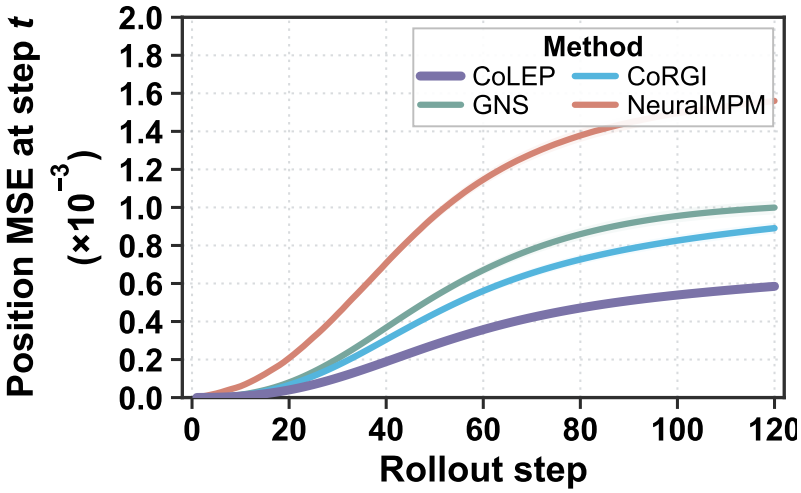


Figure 6: Instantaneous position MSE over 120 TGV-2D test rollout steps. Curves show per-step means across 50 trajectories, with shaded 95% confidence intervals.

B.6 Long-Horizon Error Growth

Instantaneous Long-Horizon Position Error. The stability curves in Figs. 2 and 6 report position error at each autoregressive rollout step. For test trajectory τ and rollout step t, the instantaneous position error is

$$e_\tau^t = \frac{1}{N_\tau d} \sum_{i=1}^{N_\tau} \left\| \mathrm{wrap}\left(\hat{\mathbf{x}}_{\tau,i}^t - \mathbf{x}_{\tau,i}^t\right) \right\|_2^2, \tag{19}$$

where N_τ is the number of nodes in the dataset-specific metric scope and d is the spatial dimension. The particle scope and displacement convention are the same as for the aggregate position metric above. For DAM-2D, the metric includes all 5,740 original fluid and static-wall nodes with box lengths $(5.586, 2.22)$.

Each curve shows the mean instantaneous error across test trajectories at each rollout step, without temporal accumulation, averaging over preceding steps, or smoothing. DAM-2D uses 25 test trajectories over 395 predicted steps; TGV-2D uses 50 test trajectories over 120 predicted steps.

B.7 Trajectory-dependent information in Eulerian State

Protocol. We first evaluate the CoLEP on DAM-2D test trajectories for 395 autoregressive steps. Every branch shares the unmodified prefix through $\tau = 197$. The *normal* branch carries each predicted grid state into the next step. A *reset* discards the inherited state and reinitializes memory from the current autoregressive particle observation; reset-once applies this operation only at step 198, while reset-K repeats it every K post-intervention steps. *Freeze* reuses the snapshot G_τ at every later step, and *wrong memory* supplies the time-aligned state of a fixed donor trajectory without changing the recipient particles. All effects below are trajectory-paired differences from normal; confidence intervals use 10,000 deterministic bootstrap resamples.

Intervention results. Table 3 reports the post-intervention velocity MSE changes for the evaluated interventions. Donor-state replacement and per-step reinitialization increase the error by 37.61% and 39.52%, respectively, consistent with the main-text results. The reset penalty decreases as the interval K increases. As an additional control, repeatedly supplying the frozen field increases the error by 95.34%.

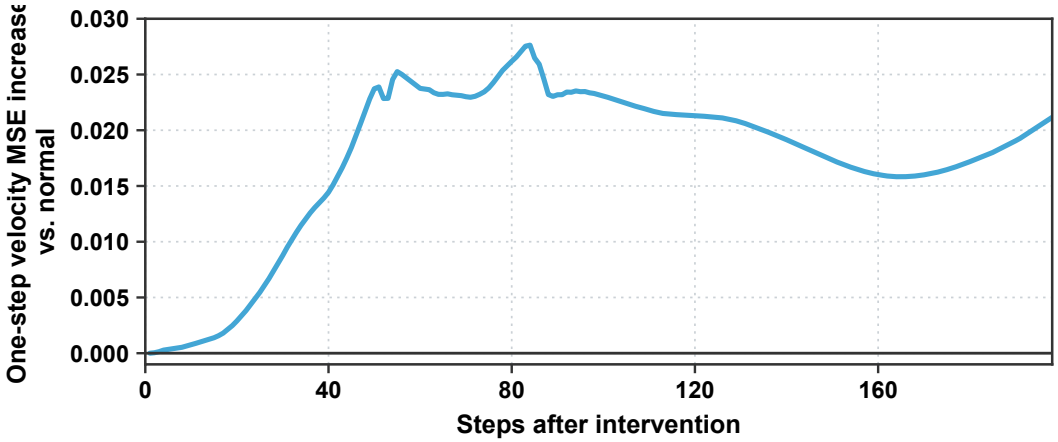


Figure 7: The instantaneous velocity error under frozen-Eulerian intervention. After the shared $\tau = 197$ prefix, every subsequent prediction reuses G_τ and discards the newly produced memory. At each step, the curve reports the cohort-mean increase over the unmodified rollout in one-step velocity MSE, computed from the predicted and reference displacements between the preceding and current states, over the same 25 DAM-2D test trajectories.

Table 3: Full DAM-2D memory-intervention results. Relative changes are measured against the normal branch over steps 198–395. "Worse" counts trajectories with positive paired MSE change.

Intervention	Relative change (%)	95% CI (%)	Worse / 25
Freeze after τ	95.34	[92.76, 97.91]	25
Reset, $K = 1$	39.52	[37.93, 41.05]	25
Reset, $K = 3$	6.47	[5.42, 7.54]	25
Reset, $K = 5$	4.22	[3.44, 4.99]	25
Reset, $K = 7$	3.69	[2.95, 4.43]	25
Reset, $K = 20$	1.48	[0.99, 1.94]	22
Wrong memory every step	37.61	[31.94, 43.92]	25

The $K = 3$ and $K = 7$ evaluations were added in a separately logged sweep and are paired with the normal branch from that sweep; all remaining rows come from the original intervention evaluation.

B.8 Persistent Eulerian State Enables Sparse GNN Evaluation

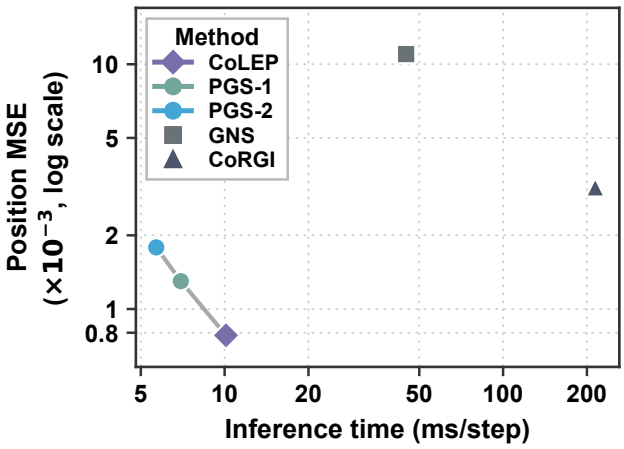


Figure 8: Accuracy–efficiency trade-off on Moving Obstacle.

In addition to the DAM-2D experiments presented in the main text, we evaluate Fast adapters on Moving Obstacle-2D using the same training protocol. For both datasets, we follow the adaptation

procedure described in Section A, optimizing only the Fast-adapter parameters ϕ while keeping the corresponding pretrained Full-model parameters θ fixed.

Optimization. We use AdamW with weight decay 10^{-8}, gradient-norm clipping at 1, and geometric learning-rate decay within each training segment. The physical batch size is one, with no gradient accumulation. Training proceeds sequentially, with each subsequent segment initialized from the preceding adapter checkpoint.

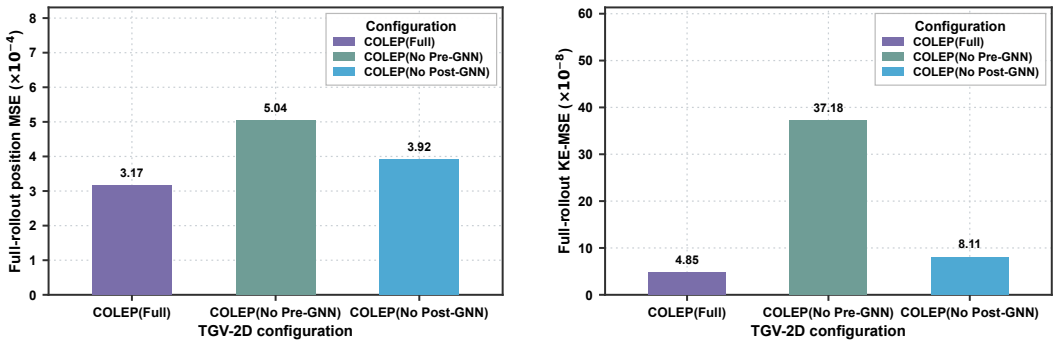
Training schedule. Both datasets use the same two-phase schedule. Single-step matching uses 600 updates. Mixed-rollout adaptation begins with 600 single-cycle handoff updates, followed by 2400 updates on longer mixed rollouts. For each dataset, the pretrained Full checkpoint remains unchanged throughout both phases.

PGS adaptation with a transient Eulerian state. For the transient Eulerian model, grid features are reconstructed from the current particles at every step, without carrying an Eulerian state across steps. On Fast steps, GNN message passing is skipped, so the particle features aggregated onto the grid consist only of the current node encodings, without the interaction residuals. We therefore apply PGS using a grid-observation adapter with a single input branch for the current observation, omitting the inherited-state branch used by the persistent Eulerian model.

C Additional Ablation Results

We show the ablation results of different components and depth metrics in the following figures.

Pre- and Post-GNN components. Figures 9a and 9b compare the full TGV-2D model with versions that omit one local interaction module. The full model has the lowest position and kinetic-energy MSE of the three. These metrics therefore support different conclusions about the observed trade-off.

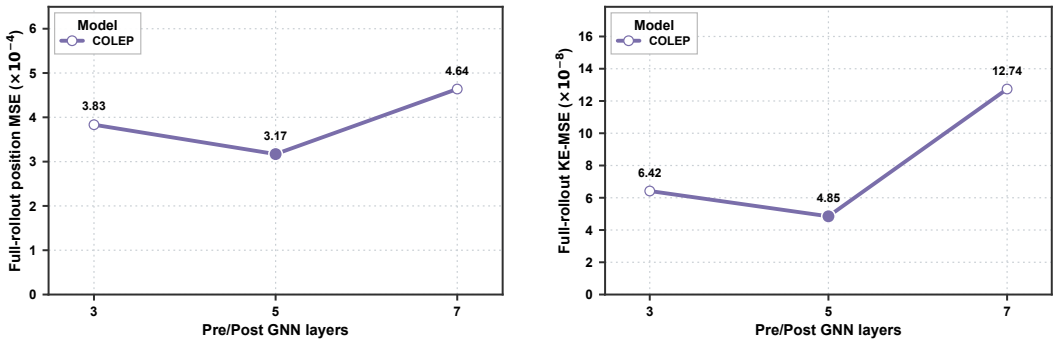


(a) Position MSE.

(b) Kinetic-energy MSE.

Figure 9: Full-rollout errors on TGV-2D for the full model and the No Pre-GNN and No Post-GNN configurations. Lower is better in both panels.

GNN depth. Figures 10a and 10b vary both the Pre- and Post-GNN depth on TGV-2D. Five layers have the lowest position and kinetic-energy MSE among the three models.



(a) Position MSE.

(b) Kinetic-energy MSE.

Figure 10: Full-rollout errors on TGV-2D with three, five, or seven layers in both the Pre- and Post-GNNs. Lower is better in both panels.