Adaptive Interaction Graphs for Particle Simulation

Aiden Zhou [1]

Abstract

Learned particle simulators based on graph neural networks achieve strong one-step accuracy, but errors compound over long horizons. An underexplored variable is the interaction graph: existing methods fix its topology via k-nearest neighbors or a static radius rule, regardless of local model confidence. We propose making this graph adaptive: a per-particle variance head, trained jointly with the acceleration head under a heteroscedastic Gaussian NLL loss, drives a trajectory in which high-uncertainty particles receive an expanded neighborhood. This is done at little extra inference cost by using the previous step's uncertainty estimate. A key discovery is that the variance head learns a meaningful notion of uncertainty: high-variance particles concentrate near complex regions, such as splash zones or free surfaces. When this signal drives graph topology, the resulting ADAPTGNS simulator achieves a strict Pareto improvement on WATERDROP and a modest gain on SAND. Given the model's stronger performance on WATERDROP, we hypothesize that adaptive graphs are most useful when complexity is concentrated in space. Our code can be found at https://github.com/aidenzhou8/AdaptGNS.

1. Introduction

Problem statement. Learned simulators based on graph neural networks (GNNs) have shown strong one-step accuracy on particle dynamics tasks, but are hurt by compounding error over many steps (Sanchez-Gonzalez et al., 2020). A key architectural choice in such systems is the **interaction graph**, which decides what particles exchange messages at each time step. Existing methods fix this topology using static geometric heuristics, such as a radius rule (Sanchez-Gonzalez et al., 2020) or a learned edge mask (Kipf et al., 2018), regardless of the model's local confidence.

[1]Yale University, New Haven, Connecticut, USA. Correspondence to: Aiden Zhou <aiden.zhou@yale.edu>.

Proceedings of the AI4Physics Workshop at the 43rd International Conference on Machine Learning (AI4Physics@ICML 2026), Seoul, South Korea. 2026. Copyright 2026 by the author(s).

We propose ADAPTGNS, an **uncertainty-adaptive interaction graph** that selectively expands edges per timestep based on the simulator's predicted uncertainty. By providing greater relational capacity where the model is least confident — and leaving the default sparse connectivity in place everywhere else — ADAPTGNS aims to make graph network simulators (GNSs) more stable, but also to preserve the speed and efficiency that motivate using a learned simulator in the first place.

Motivation and significance. Long-horizon physical simulation is key to a wide range of subjects, such as robotics, molecular dynamics, and world models. Learned simulators can provide much faster inference than classical solvers, with reported speedups of up to 5,000× over the Material Point Method (MPM) on granular flow simulations (Kumar & Vantassel, 2023). More broadly, neural surrogate models for fluid simulation have been reported to reduce simulation cost by orders of magnitude relative to conventional numerical solvers (Lino et al., 2023). However, this benefit is lost if errors accumulate over many steps. Enhancing consistency without increasing average compute per step would make learned simulators a more viable drop-in replacement.

A second motivation comes from the physical observation that **uncertainty is rarely uniform**. When two fluids crash against each other, the unpredictable dynamics happen inside a "splash zone," whereas the rest of the pool sits quietly. Our adaptive graph methodology encodes this intuition.

Learning objective. Our task is per-particle acceleration prediction at each timestep, with the predicted values integrated forward to produce a complete simulation trajectory. Errors at each step compound over many steps, and our goal is to reduce this error growth relative to baseline models. A secondary objective trains a **per-particle uncertainty head** — a scalar variance estimate σ_i — that drives the adaptive graph policy at each step. We report **MSE at step 200 (MSE@200)** as a standard measure of accuracy and **average edges per step** as a hardware-agnostic proxy for efficiency. Note that each processor layer of a GNS does $O(|E|)$ work, so compute scales with edge count.

Summary of results. ADAPTGNS achieves a strict Pareto improvement on WATERDROP (20% fewer edges and lower MSE@200 than fixed $k = 5$), but only a modest gain on

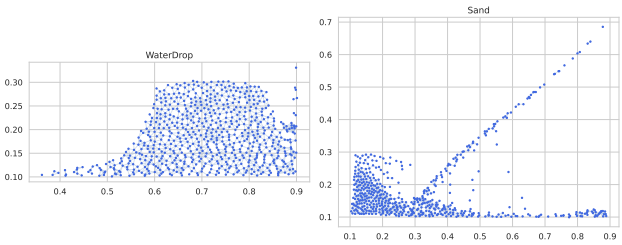

Figure 1. Interaction graph snapshots at Step 100 for WATERDROP (left) and SAND (right).

Table 1. Data splits and rollout lengths.

Dataset	Train	Val.	Test	Steps
SAND	1,000	100	30	320
WATERDROP	1,000	30	30	1,000

SAND (12% lower MSE@200 than fixed $k = 5$, using 10% fewer edges than $k = 10$).[1] The story becomes clearer once we ablate for the choice of loss function: compared to a fixed GNS **trained on NLL loss**, the adaptive policy once more achieves a strong gain (21% fewer edges and 8% lower MSE@200). We also report the optimized hyperparameters found by grid sweep on SAND and WATERDROP (σ-percentile $= 70$, radius factor $= 1.267$) and several negative results.

2. Dataset

Description. We use two distinct particle simulation datasets. SAND is taken from (Kumar & Vantassel, 2023), hosted on DesignSafe-CI (DOI: `10.17603/ds2-0phb-dg64`), and generated with the Taichi-MPM solver. WATERDROP is taken from (Sanchez-Gonzalez et al., 2020) and generated with a Smoothed Particle Hydrodynamics (SPH) solver. Note that WATERDROP trajectories are given as TFRecords, so we provide a script to convert the dataset to a standard `.npz` format. These two materials were chosen to evaluate adaptive graphs across both fluid and granular flow regimes.

Graph statistics. The edges of a standard interaction graph are constructed at each step between particles separated by at most $r = 0.015$. Table 2 reports graph statistics taken at Step 100, averaged across trajectories. Both datasets produce sparse graphs with moderate average degree.

The temporal evolution of graph statistics reveals varying dynamics: in WATERDROP, edge count and average degree rise at impact and taper as particles disperse, whereas in SAND the graph undergoes a sharp slump around Step 50 before recovering to a stable plateau as particles settle. These signatures are central to understanding the performance gap between datasets, discussed in Section 5.2.

[1] $k \approx$ the average number of neighbors per particle at the given connectivity radius r. We use this notation throughout, but be aware that r is fixed; neighbor count varies with local density.

3. Related Work

Graph network simulators. The idea of representing physical systems as graphs was introduced by (Battaglia et al., 2016), who developed the relational inductive bias that subsequent work extends upon. (Sanchez-Gonzalez et al., 2020) scaled this paradigm to large particle systems via radius-based interaction graphs and noise-augmented training. This work forms the primary baseline and codebase for our project. A central restriction shared by both works is that the interaction graph is fixed by geometric rules, which do not regard local dynamics. The phenomenon of "over-squashing" provides a theoretical motivation: fixed sparse graphs can restrict the information exchanged across bottleneck regions on a per-step basis (Topping et al., 2022). In autoregressive simulators, such representational constraints may interact with compounding error, motivating adaptive distribution of edges in complex and uncertain regions.

Adaptive predecessors. The most straightforward try to learn the graph structure itself came from (Kipf et al., 2018), whose Neural Relational Inference (NRI) derives a discrete interaction graph via a VAE encoder over particle histories. However, NRI produces a graph that is fixed at inference, and incurs significant overhead. More recently, AdaMeshNet (Seo et al., 2025) proposes per-step rewiring for mesh-based GNNs using geometric heuristics, outperforming static baselines. Our work diverges from both: we target point-cloud particle systems rather than meshes, use no separate encoder, and use the model's own predicted uncertainty σ_i as the adaptation signal.

Physics priors and non-graph alternatives. Several recent approaches look to enhance long-horizon simulation without adapting a particle interaction graph. Neural SPH augments learned Lagrangian fluid simulators with components taken from Smoothed Particle Hydrodynamics (SPH) (Toshev et al., 2024). NeuralMPM uses a hybrid particle-grid representation: particles are interpolated to a regular grid, updated with image-to-image networks, and interpolated back (Rochman-Sharabi et al., 2025). Lastly, Transformer-based particle simulators such as FluidFormer combine local convolutions with global attention to capture long-range dependencies (Wang et al., 2025). These methods address the same problem of long-horizon consistency, but through physics priors, grid-based emulation, or global attention. ADAPTGNS is orthogonal to these directions, and could in principle be integrated with stronger message functions or physics-informed losses.

Table 2. Graph statistics at Step 100, averaged over 30 test trajectories.

Dataset	Trajs	Nodes	Edges	Density	Avg Degree	Avg Clustering
WATERDROP	30	567 ± 200	1550 ± 583	1.08×10^{-2}	5.42 ± 0.22	0.494 ± 0.036
SAND	30	1270 ± 503	4324 ± 1864	7.35×10^{-3}	6.76 ± 1.01	0.500 ± 0.034

4. Methodology

Variance head. We extend a standard GNS with a **variance prediction head**: alongside the acceleration output, the network predicts a scalar uncertainty σ_i per particle. This is done via a two-layer MLP on the final node embeddings, using a softplus activation to enforce positivity:

$$\sigma_i^2 = \mathrm{softplus}(f_\theta(h_i)).$$

This change adds fewer than 20,000 parameters to the standard GNS, so the architectural overhead is insignificant relative to the base simulator.

Heteroscedastic NLL loss. Both acceleration and variance heads are trained jointly under a Gaussian NLL loss:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{\|\hat{\mathbf{a}}_i - \mathbf{a}_i\|_2^2}{2\sigma_i^2} + \frac{d}{2}\log\sigma_i^2\right], \tag{1}$$

where $d = 2$ is the acceleration dimension. NLL ensures the model is rewarded for concentrating uncertainty where error is large and punished for inflating it elsewhere.

Adaptive graph construction loop. Our core contribution is an inference-time loop that uses predicted uncertainty to modify the graph topology. A naive implementation would run a GNN forward pass, extract σ_i, rebuild the graph, and then run a second forward pass to compute the final acceleration. However, this doubles inference cost. Instead, we use a **single-pass lagged-σ** design. At step t, the graph is constructed using the previous step's uncertainty estimates $\sigma_i^{(t-1)}$. The forward pass at step t then produces both the next position and a fresh uncertainty estimate $\sigma_i^{(t)}$ for step $t+1$. Concretely, the adaptive rule is:

1. Construct the standard interaction graph.
2. Compute the per-step τ-th percentile of $\{\sigma_i\}_{i=1}^N$ and mark particles above this percentile as high uncertainty.
3. Add extra edges from a larger radius $r_{\text{large}} = \texttt{rad_factor} \cdot r$ for edges incident to one or more high-uncertainty particles.

Hyperparameters. Using a grid sweep over validation set trajectories, we selected values of $r = 0.015$, $\texttt{rad_factor} = 1.267$, and $\tau = 70$. The expanded radius roughly matches $r = 0.019$, so high-σ particles receive the same connectivity as the strongest fixed-graph baseline.

5. Experiments and Results

Baselines and Ablations. We compare ADAPTGNS to five models on both WATERDROP and SAND. The naive comparison is to a **per-particle MLP**, which measures the value of relational inductive bias. Far more reasonable models are a **fixed-graph GNS with** $k = 5$ (or $r = 0.015$), which serves as the primary comparison, and a **fixed-graph GNS with** $k = 10$ (or $r = 0.019$), which gives an upper bound on the benefit of denser interaction graphs.

We also conduct two ablation studies. **GNS (k-mid)** uses an intermediate radius ($r = 0.0176$) matched to ADAPTGNS's mean edge count, isolating whether our gains come from adaptive topology or just from having more edges on average.[2] **ADAPTGNS fixed-graph uses NLL** but with fixed $r = 0.015$, isolating the contribution of the adaptive loop from the changed loss function.

5.1. Quantitative Results

Our first question is whether the variance head reveals physical structure. If σ_i is just noise, a graph policy that depends on it is doubtful to succeed. We find evidence that σ_i acts as an accurate rank-based signal, even though it is not perfectly calibrated in absolute value.

Takeaway 1

As shown in Figure 2, the network correctly infers what "uncertainty" means in the context of physical simulation, purely as a byproduct of the learning objective. This is supported by our calculations in Table 3. On SAND, Bins 0–7 are tightly calibrated, with $|\Delta| \leq 0.06$. The total ECE is large only because the model severely underestimates its error on the hardest particles (Bins 8 and 9). Results are weaker on WATERDROP, but conform to the same trend. As our adaptive methodology thresholds at a percentile rank, this does not pose a huge obstacle.

Next, we evaluate ADAPTGNS on SAND and WATERDROP. See Table 4 for a summary of results and Figure 3 for more nuanced dynamics.

[2] Note that this ablation was not conducted for WATERDROP. On SAND, k_{mid} was constructed by interpolating in r^2 between the $k = 5$ and $k = 10$ edge counts to match ADAPTGNS's budget, giving $r \approx 0.0176$. On WATERDROP, ADAPTGNS already uses **fewer** edges than fixed $k = 5$, so no equivalent comparison exists.

Table 3. Per-decile calibration on SAND and WATERDROP.

	SAND			WATERDROP		
Bin	mean SE	$2\hat{\sigma}^2$	$\lvert\Delta\rvert$	mean SE	$2\hat{\sigma}^2$	$\lvert\Delta\rvert$
0	0.023	0.009	0.014	0.543	0.003	0.540
1	0.025	0.014	0.011	0.540	0.004	0.537
2	0.024	0.018	0.006	0.131	0.009	0.121
3	0.026	0.023	0.003	0.081	0.015	0.066
4	0.033	0.029	0.004	0.208	0.022	0.186
5	0.046	0.038	0.008	0.495	0.032	0.463
6	0.067	0.051	0.016	1.386	0.048	1.339
7	0.141	0.084	0.057	3.167	0.065	3.101
8	0.560	0.195	0.365	5.739	0.094	5.645
9	15.482	1.882	13.599	17.878	0.288	17.590
Total ECE			1.408			2.959

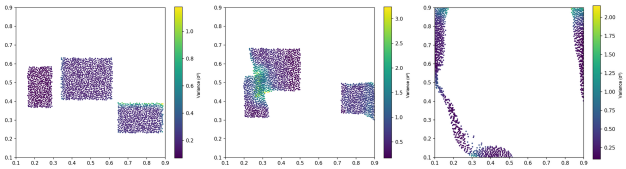

Figure 2. Per-particle uncertainty σ_i at steps 1, 10, and 50 of a SAND trajectory. High-σ particles are concentrated in impact zones, free surfaces, and regions of rapid velocity change.

SAND. Comparing ADAPTGNS to its fixed-graph counterpart (using NLL and $r = 0.015$), the adaptive policy **reduces edges by 21% and lowers MSE@200 by 8% (0.0417 vs. 0.0382)**. This is the cleanest comparison: it isolates the value of placing edges purposely rather than uniformly. Second, comparing ADAPTGNS to k_{mid} ($r = 0.0176$) at matched edge counts ($\sim$23,000 edges each), **the model using MSE wins (0.0355 vs. 0.0382)**. The benefit of MSE as an easier learning objective is made clear by the NLL model at $r = 0.0176$. Although it also uses a larger fixed radius, its MSE@200 is much worse (0.0422).

WATERDROP. The result here is a strong advance: ADAPTGNS **uses 20% fewer edges than $k = 5$ (7,821 vs. 9,741) and also achieves lower MSE@200 (0.0412 vs. 0.0414).** Even compared to $k = 10$, the tradeoff is 36% fewer edges for a 6% relative MSE@200 increase. In contrast to SAND, the decrease in performance caused by using NLL instead of MSE is close to zero.

Takeaway 2

ADAPTGNS strictly beats the same NLL model without the adaptive graph mechanism on both datasets. On WATERDROP, it also achieves a strong advance over every fixed-graph comparison. Due to its harder learning objective, ADAPTGNS does not yet defeat the best MSE-trained models on SAND.

5.2. Analysis and Discussion

Why WATERDROP and not SAND? The performance gap between datasets can be attributed to two physical properties. First, **graph density evolves differently**. Most granular trajectories end up packing particles into an increasingly dense heap. A droplet impact does the opposite, dispersing particles and keeping the graph sparse for the rest of the trajectory. Second, **uncertainty is more heterogeneous in WATERDROP**: a splash creates local complexity but is mostly surrounded by quiet, giving the variance head plenty of contrastive signal. In SAND, nearly every particle undergoes the same dynamic, as was previously observed in our exploratory data analysis.

MSE vs. NLL. A second question concerns the model's objective. The variance head requires a learning objective that integrates uncertainty, but NLL can be harder for raw acceleration accuracy than MSE. This is visible on SAND, where the best MSE baselines outperform NLL models. Hence, a future goal is to decouple acceleration / force prediction from uncertainty ranking; for instance, by using MSE loss along with a subsidiary ranking or calibration loss for the variance head.

Edge count is a noisy proxy for compute. The radius graph at step t is constructed from the model's predicted positions. Hence, a key observation is that **worse models can drift into denser configurations and end up using more edges**. This is shown by our results in Table 4 and Figure 3, particularly for the two NLL SAND models.

6. Limitations and Future Work

Several restrictions should be considered when interpreting this study, and addressed in future work.

Runtime measurement. We report average edge count because it tracks the principal cost of a GNN forward pass,

Table 4. Results on SAND (320 steps) and WATERDROP (1000 steps).

SAND

Model	Edges/step	MSE@200
MLP	N/A	0.121
$k = 5$ ($r = 0.015$)	18,682	0.0434
k_{mid} ($r = 0.0176$)	23,498	0.0355
$k = 10$ ($r = 0.019$)	25,516	**0.0285**
NLL ($r = 0.015$)	28,968	0.0417
NLL ($r = 0.0176$)	28,768	0.0422
AdaptGNS	**22,958**	0.0382

WATERDROP

Model	Edges/step	MSE@200
MLP	N/A	0.207
$k = 5$ ($r = 0.015$)	9,741	0.0414
$k = 10$ ($r = 0.019$)	12,212	**0.0388**
NLL ($r = 0.015$)	7,875	0.0417
AdaptGNS	**7,821**	0.0412

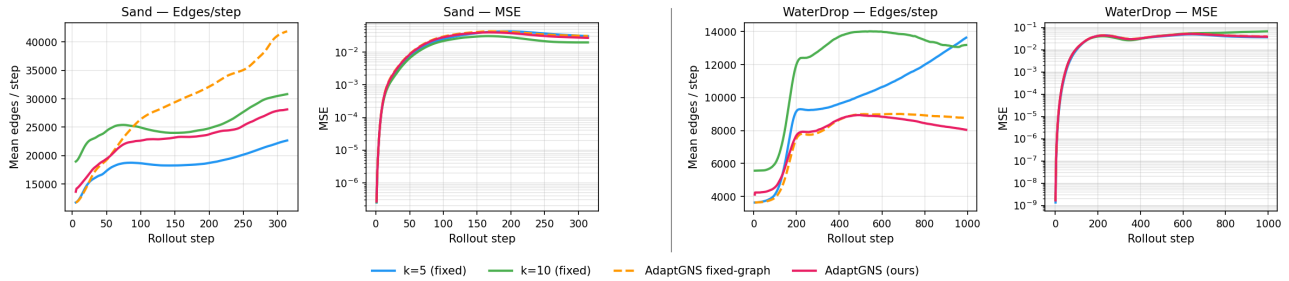

Figure 3. Per-step dynamics (edge count and MSE) on SAND and WATERDROP.

but it is not a complete runtime metric. Adaptive graph construction searches over an expanded radius for selected particles. The overhead this process creates should be quantified in future work.

Lagged uncertainty. The single-pass design uses $\sigma^{(t-1)}$ to construct the graph at step t. This avoids a second forward pass but may react too late during abrupt transformations. A next-step predictor or a physical prior based on acceleration or velocity divergence could reduce this lag.

Scope. The present experiments cover one SPH fluid dataset and one MPM granular dataset. Broader evaluation on more materials, 3D contexts, and hybrid rigid–fluid scenes is needed before drawing general conclusions.

Broader domains. Uncertainty-guided graph adaptation may also prove useful beyond particle simulation. Natural extensions include molecular dynamics, where reactive sites or conformational changes may benefit from larger local neighborhoods; mesh-based PDE solvers, where shocks or boundary layers motivate adaptive refinement; and robotics, where local interactions often dominate trajectory error.

7. Conclusion

In this paper, we explored making the interaction graph of a learned particle simulator adaptive — sparse where the model is confident, denser where it is not — without paying the cost of two GNN computations per step. We contribute a single-pass adaptive mechanism driven by a per-particle variance head, which is trained jointly with the acceleration head under NLL loss. We observe that σ_i corresponds to real physical uncertainty, and that ADAPTGNS, driven by this signal, achieves modest but notable results on the WATERDROP and SAND datasets.

Future work should target variance-aware learning objectives that preserve MSE accuracy, a next-step uncertainty predictor (solving both lag and calibration concerns) and a learnable per-particle expansion radius. We also intend to test ADAPTGNS on a broader range of physical regimes and on longer-horizon tasks, and to conduct rigorous multi-seed evaluations. These are needed to more carefully examine the general usefulness of adaptive interaction graphs for learned particle simulation.

Impact Statement

This paper presents work whose goal is to refine learned physical simulation. Potential societal consequences are tied to downstream uses of faster and more accurate simulation systems. We do not identify additional societal risks beyond those typically associated with machine learning research.

References

Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D. J., and Kavukcuoglu, K. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, 2016. URL https://arxiv.org/abs/1612.00222.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. URL https://arxiv.org/abs/1802.04687.

Kumar, K. and Vantassel, J. GNS: A generalizable graph neural network-based simulator for particulate and fluid modeling. *Journal of Open Source Software*, 8(88):5025, 2023. doi: 10.21105/joss.05025.

Lino, M., Fotiadis, S., Bharath, A. A., and Cantwell, C. D. Current and emerging deep-learning methods for the simulation of fluid dynamics. *Proceedings of the Royal Society A*, 479(2275):20230058, 2023. doi: 10.1098/rspa.2023.0058. URL https://royalsocietypublishing.org/doi/10.1098/rspa.2023.0058.

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

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. W. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/2002.09405.

Seo, S., Kim, H., Kim, J., and Park, C. Adaptive graph rewiring to mitigate over-squashing in mesh-based GNNs for fluid dynamics simulations. *arXiv preprint arXiv:2511.12709*, 2025. URL https://arxiv.org/abs/2511.12709.

Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2111.14522.

Toshev, A., Erbesdobler, J. A., Adams, N. A., and Brandstetter, J. Neural SPH: Improved neural modeling of lagrangian fluid dynamics. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48428–48452. PMLR, 2024. URL https://proceedings.mlr.press/v235/toshev24a.html.

Wang, N., Chen, Y., and Zheng, S. FluidFormer: Transformer with continuous convolution for particle-based fluid simulation. *arXiv preprint arXiv:2508.01537*, 2025. URL https://arxiv.org/abs/2508.01537.