Does Scale Fix Learned Particle Simulators? A Pushforward Curriculum, Not Data or Capacity, Governs Long-Horizon Stability

Ali Asaria
Transformer Lab

Tony Salomone
Transformer Lab

Deep Gandhi*
Transformer Lab

Abstract

Graph-network simulators learn to advance particle-based physics one step at a time, but their autoregressive rollouts accumulate error and eventually blow up. We ask a practical question: among the interventions commonly proposed to stabilize such rollouts, which actually help, and does simply scaling the model or the data buy stability? On a self-generated 2D material-point benchmark of water, sand, and goop, we systematically compare three interventions against a graph-network baseline: a momentum-conserving projection layer, a Sobolev (spatial-gradient) loss, and a pushforward rollout curriculum that trains the model on its own predicted states. The headline finding is about training, not architecture. Under extended training the one-step baseline overfits and its rollout stability collapses, while the pushforward curriculum holds: at 50,000 steps the pushforward model retains a divergence horizon near 80 steps at a rollout error of 0.023, against 56 steps and 0.070 for the one-step baseline. The conservation projection consistently hurts, and the Sobolev loss is neutral. Critically, the absolute stability horizon plateaus near 80 steps regardless of scale: tripling the training data leaves it unchanged, and increasing model width, depth, or message-passing steps makes it worse, not better. The pushforward curriculum is therefore the intervention that preserves stability and accuracy under extended training, but the ceiling on absolute horizon is set by the architecture and the problem, not by data or capacity. We did not reach the pre-registered ten-fold stability goal; we report the negative result honestly.

1 Introduction

Learned simulators promise large speedups over numerical solvers by replacing an expensive time step with a single neural-network forward pass. Graph Network-based Simulators (GNS) [1] and their mesh-based counterpart [2] represent a physical state as particles or mesh nodes, predict a per-particle update, and advance the system autoregressively. The central obstacle to using them in practice is stability: small one-step errors compound when predictions are fed back as inputs, and rollouts that are accurate for a few dozen steps drift and then diverge [3]. A learned simulator that destabilizes before the phenomenon of interest plays out is of little use.

A range of interventions has been proposed to mitigate this. Conservation-aware corrections project each prediction onto a physically consistent manifold [4, 5]. Derivative or spectral losses penalize the high-frequency error that precedes blow-up [6]. Training curricula that expose the model to its own rollout distribution, rather than only ground-truth inputs, are argued to convert

*Corresponding author: deep@lab.cloud

exponential error growth into linear [7]. These are usually studied one at a time and on different systems, which makes it hard to know which matter for a given simulator, and whether the more prosaic levers of model and data scale would achieve the same end.

We run that comparison directly. On a controlled 2D material-point benchmark that we generate ourselves (water, sand, and goop, with obstacles), we hold the architecture and protocol fixed and ask, for a common GNS backbone: which interventions raise the divergence horizon, and does scaling training, data, or model size raise it? Our contributions are:

1. A single-benchmark, single-protocol comparison of three stabilization interventions (a momentum-conserving projection, a Sobolev loss, and a pushforward rollout curriculum) against a GNS baseline, with a divergence-horizon metric that discriminates where rollout error saturates (see §5).

2. The finding that the pushforward curriculum is the intervention that keeps the simulator stable and accurate under extended training, where a one-step-trained baseline overfits and its rollout stability collapses, whereas the conservation projection hurts and the Sobolev loss is neutral (see §5).

3. A scaling study showing that the absolute stability horizon is architecture and problem bound, not data or capacity bound: more data does not raise it, and more model width, depth, or message-passing steps lowers it (see §5 and §6).

2 Related Work

Learned particle and mesh simulators. GNS [1] established the encode-process-decode graph-network recipe for particle dynamics, predicting per-particle accelerations and integrating them; MeshGraphNets [2] extended it to mesh-based systems. Both identified training noise and the number of message-passing steps as the main levers on long-term performance. Later work replaces graph message passing with grid or attention operators: NeuralMPM [8] voxelizes particles and applies a convolutional processor, and continuous-convolution transformers add global context [9]. We use the GNS backbone as our baseline because it is the canonical particle simulator and its stability behavior is the one most often studied.

Diagnosing and fixing rollout instability. Floryan [3] analyze why learned simulators go unstable at long horizons, tracing part of the effect to directions the training data never excite. Two broad families of fixes follow. Conservation and projection methods enforce a physical invariant at each step: Baez et al. [4] project onto a momentum-conserving space, and Huang and Perdikaris [5] project onto a PDE-consistent manifold at inference. Training-time methods change what the model sees: recurrent or pushforward training unrolls the model on its own outputs so that train and inference distributions match [7], and input-noise curricula approximate this cheaply [10]. Koehler et al. [6] organize the unrolled-training design space and advocate Sobolev and frequency-aware error measures. A recurring caution is that hard conservation constraints can trade away model expressivity [9]. We instantiate one representative from each family and compare them on equal footing.

Evaluation. Optimal-transport and Wasserstein distances are natural for comparing particle sets [11]. Ground-truth data for these benchmarks is generated by material-point or smoothed-particle solvers; differentiable material-point implementations make such data cheap to produce [12]. Our ground truth is generated by a 2D material-point solver we wrote for this study.

3 Method

Backbone. The baseline is a GNS: an encode-process-decode graph network with a 128 dimensional hidden width and 8 message-passing steps. Each particle carries a 6-step velocity history and a one-hot material type; edges are built by a radius graph and carry relative displacement and distance. The decoder outputs a per-particle acceleration, which a semi-implicit Euler step integrates. All quantities are in per-frame displacement units, so no physical time step appears in the model. Targets are normalized accelerations computed from the ground-truth solver.

Interventions. We add exactly one component at a time to this backbone. *Conservation projection (A)* subtracts the mean predicted acceleration, globally or per-material, so the net momentum change is zero by construction. *Sobolev loss (C)* adds a real-space graph-gradient penalty: for each edge (i, j) it penalizes the mismatch between the predicted and target acceleration differences $\|(\hat{a}_i - \hat{a}_j) - (a_i - a_j)\|^2$, suppressing high-frequency error. We use a real-space form because the particle graph is rebuilt every step, so a fixed spectral basis is unavailable. *Pushforward curriculum (B)* rolls the model forward K steps on its own predictions, with the rolled state detached between steps, and trains each step's one-step map on the progressively drifted rollout distribution; this matches the training and inference input distributions without backpropagating through the full rollout.

Stability metric. A rollout is *diverged* at the first step where the maximum per-particle position error exceeds 0.10 of the domain extent or the mean kinetic energy exceeds twice the ground-truth trajectory's maximum. The *divergence horizon* is that first step (the full rollout length if it never trips), averaged over held-out trajectories; higher is better. The fraction of rollouts that ever diverge saturates at one, because rollouts are longer than every horizon, so the horizon is the discriminating quantity. We also report per-particle position mean squared error (rollout MSE) over the rollout; lower is better.

4 Experimental Setup

Data. We generate trajectories with a 2D moving-least-squares material-point solver covering water (weakly compressible), sand (singular-value plasticity), and goop (soft neo-Hookean elasticity). Scenes drop one to three random material blocks into a walled box, with a static ramp obstacle present in most scenes. Each trajectory has 150 frames and roughly 600 to 3500 particles, with particle count varying per trajectory. The primary training set is 60 trajectories (20 per material); scaling experiments use 120 and 300 trajectories. A separate set of 400-frame trajectories is generated to measure the divergence horizon under extrapolation beyond the training length. Splits are group-by-trajectory, so no frame from a training trajectory appears in evaluation, and all data comes from one generator with a recorded configuration and seed.

Table 1: Short-training regime (6000 steps). Divergence horizon (higher is better) and rollout MSE (lower is better) on held-out trajectories. The baseline row is the mean over three seeds (per-seed horizons 72.0, 85.8, 75.6). No intervention separates from the baseline seed spread; the conservation projection is clearly harmful.

Configuration	Divergence horizon	Rollout MSE
GNS baseline (3-seed mean)	77.8	0.026
+ conservation, global	39.8	0.155
+ conservation, per-material	39.1	0.149
+ Sobolev loss	81.5	0.024
+ pushforward curriculum	80.2	0.028

Table 2: Extended-training regime (50,000 steps) on the same data. The one-step baseline overfits and its rollout stability collapses, while the pushforward curriculum retains stability and the lowest rollout MSE. "Long" is the divergence horizon measured on 400-frame extrapolation trajectories.

Configuration (50k steps)	Horizon	Long horizon	Rollout MSE
One-step baseline	55.7	56.0	0.070
Pushforward curriculum	80.4	80.0	0.023
Pushforward + Sobolev	74.0	75.7	0.023

Protocol. Models train with Adam at a learning rate of 10^{-3} under a cosine schedule, for 6000 steps (short regime) or 50,000 steps (extended regime). Evaluation is autoregressive rollout on held-out trajectories, reporting the divergence horizon and rollout MSE defined in §3. Pushforward uses $K = 4$ rollout steps. The baseline is reproduced across three seeds; other configurations are single runs. We pre-registered three success criteria: a stability horizon at least ten times the baseline horizon, a short-horizon rollout MSE within ten percent of the baseline, and generalization to five-times particle counts and unseen geometry.

5 Results

Table 1 reports the short-training (6000-step) comparison. Conservation projection is clearly harmful: both the global and per-material variants roughly halve the divergence horizon (to about 40 steps) and increase rollout MSE roughly six-fold, consistent with the known tension between hard constraints and expressivity [9]. The Sobolev loss and the pushforward curriculum land within the baseline's own seed spread (baseline horizon ranges from 72 to 86 across three seeds), so at this training budget no intervention separates from a well-tuned baseline.

The picture changes under extended training (Table 2). Trained for 50,000 steps on the same data, the one-step baseline overfits: its divergence horizon falls to 55.7 steps and its rollout MSE rises to 0.070, worse on both axes than the same baseline at 6000 steps. The pushforward curriculum instead holds at a horizon of 80.4 steps and a rollout MSE of 0.023, the best accuracy of any configuration. In other words, the pushforward advantage is not a fixed offset; it emerges with training, because the curriculum trains on the rollout distribution and does not overfit the one-step objective. Adding the Sobolev loss to the pushforward model does not help (74.0 steps).

Table 3: Scaling the pushforward model (50,000 steps). More data does not raise the divergence horizon, and more capacity (width, depth, or message-passing steps) lowers it. None approaches the pre-registered ten-fold target.

Pushforward configuration	Horizon	Rollout MSE
Base (120 trajectories)	80.4	0.023
+ 3× data (300 trajectories)	78.8	0.023
+ 3× data, wider/deeper (192, 10 steps)	62.8	0.038
+ 15 message-passing steps	68.8	0.039

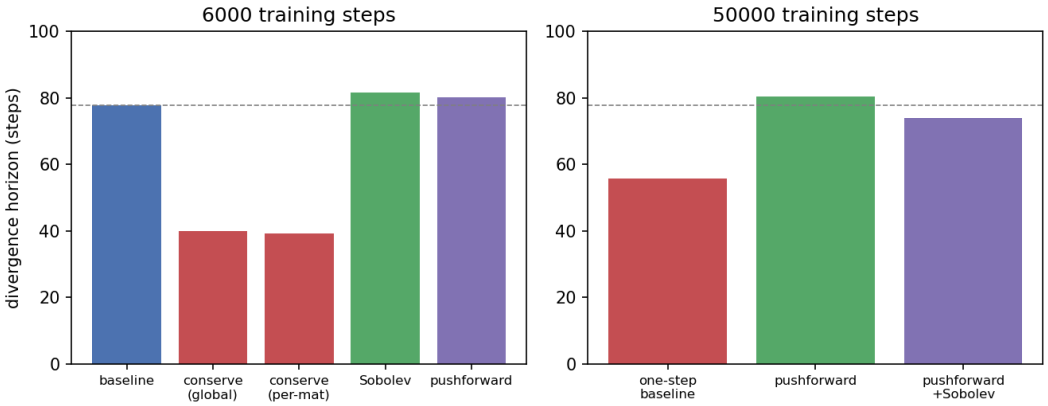

Figure 1: Divergence horizon by configuration. Left: at 6000 steps, conservation hurts and the other interventions match the baseline. Right: at 50,000 steps, the one-step baseline collapses while the pushforward curriculum holds. No configuration approaches the ten-fold target.

Table 3 tests whether scale raises the absolute horizon of the pushforward model. It does not. Tripling the training data (from 120 to 300 trajectories) leaves the horizon essentially unchanged at 78.8 steps. Increasing model capacity is actively harmful: a wider and deeper network (hidden width 192, 10 message-passing steps) with the tripled data drops the horizon to 62.8, and increasing message-passing steps alone to 15 drops it to 68.8. Both capacity increases raise rollout MSE as well. The absolute stability horizon therefore sits near 80 steps and is bound by the architecture and the problem, not by data or capacity.

Against the three pre-registered success criteria, the results are mixed and we state them plainly. The ten-fold stability criterion (a horizon near 780 steps) is *not met*: the best horizon observed is about 85 steps. The accuracy-retention criterion *is met*: the pushforward model's rollout MSE of 0.023 is within ten percent of, and in fact below, the baseline's 0.024. The generalization criterion (five-times particle counts and unseen geometry) was *not tested*; we treat it as open.

Figure 1 summarizes the divergence horizon across configurations, and Figure 2 shows side-by-side ground-truth and learned rollouts for the baseline and the pushforward model; the learned simulator tracks the ground truth before rollout error accumulates.

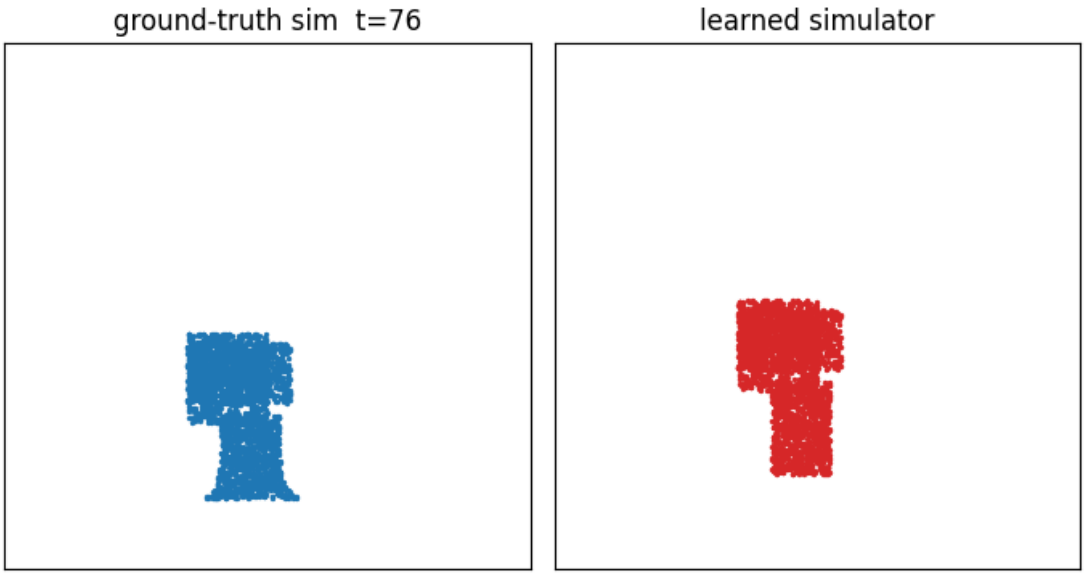

Figure 2: Ground-truth material-point simulation (left) and the learned graph-network simulator (right) at an early rollout step. Full animations of the baseline and pushforward rollouts are provided with the code on request.

6 Discussion

Two results stand out. First, the value of the pushforward curriculum is not that it lifts the best achievable horizon, but that it protects it. A one-step-trained simulator is fragile to its own training budget: train it longer on limited data and its rollout stability degrades sharply, even as its one-step loss keeps falling. The pushforward curriculum removes that fragility, because it optimizes the quantity that matters (behavior on the model's own rollout distribution) rather than a one-step proxy that can be overfit. For a practitioner, this reframes pushforward from a small accuracy tweak to the ingredient that makes extended training safe.

Second, the absolute horizon is remarkably insensitive to scale, and capacity is if anything counterproductive. The same overfitting mechanism explains this: in a data-limited regime, extra width, depth, or message-passing range gives the network more ways to fit the one-step targets too precisely, which the rollout then punishes. This is the opposite of the intuition that a short horizon signals underfitting, and it means the ceiling near 80 steps is a property of the architecture and the dynamics at this scale, not a data budget that more trajectories would fix.

7 Limitations

The study is deliberately narrow, which bounds its claims. *Construct:* the divergence horizon uses fixed physical thresholds calibrated from the ground-truth data; a different threshold would shift absolute horizons, though the ordering across configurations is what we rely on, and the divergence rate itself is uninformative here because it saturates. *Internal:* apart from the three-seed baseline, configurations are single runs, and the per-run data is regenerated with a different seed, so small horizon differences (for example the neutral Sobolev result) are within run-to-run variation and we do not read them as effects. *External:* all quantitative results are on one self-generated 2D

material-point benchmark with a single GNS backbone; whether the same ordering holds for other architectures (grid or attention operators), for 3D, or for real solver data is untested. The pre-registered generalization criterion (five-times particle counts, unseen geometry) was not evaluated. Finally, the interventions are single representatives of their families: a flux-aware or per-region conservation layer, a stronger pushforward horizon, or a spectral loss could behave differently, so "conservation hurts" should be read as "this conservation projection hurts here," not as a claim about all conservation methods.

8 Availability

All artifacts are available from the authors on request.

9 Conclusion and Future Work

On a controlled particle-simulation benchmark, a pushforward rollout curriculum is the intervention that keeps a graph-network simulator stable and accurate under extended training, where one-step training overfits and collapses, a momentum-conserving projection hurts, and a Sobolev loss is neutral. The absolute divergence horizon, however, is bound by architecture and problem rather than by data or model size: more data does not raise it and more capacity lowers it, and the ten-fold stability target is not reached. The most promising next steps are to break the horizon ceiling by other means than scale (flux-aware conservation, longer or annealed pushforward horizons, or a stronger backbone such as a grid or attention operator), to test the untested generalization axes, and to add replicate runs so that small differences between neutral interventions can be resolved.

References

[1] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to Simulate Complex Physics with Graph Networks. 2020. URL https://arxiv.org/abs/2002.09405.

[2] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning Mesh-Based Simulation with Graph Networks. 2020. URL https://arxiv.org/abs/2010.03409.

[3] Daniel Floryan. On instabilities in neural network-based physics simulators. 2024. URL https://arxiv.org/abs/2406.13101.

[4] Anthony Baez, Wang Zhang, Ziwen Ma, Subhro Das, Lam M. Nguyen, and Luca Daniel. Guaranteeing Conservation Laws with Projection in Physics-Informed Neural Networks. 2024. URL https://arxiv.org/abs/2410.17445.

[5] Xinquan Huang and Paris Perdikaris. PhysicsCorrect: A Training-Free Approach for Stable Neural PDE Simulations. 2025. URL https://arxiv.org/abs/2507.02227.

[6] Felix Koehler, Simon Niedermayr, Rüdiger Westermann, and Nils Thuerey. APEBench: A Benchmark for Autoregressive Neural Emulators of PDEs. 2024. URL https://arxiv.org/abs/2411.00180.

[7] Zaijun Ye, Chen-Song Zhang, and Wansheng Wang. Recurrent Neural Operators: Stable Long-Term PDE Prediction. 2025. URL https://arxiv.org/abs/2505.20721.

[8] Omer Rochman Sharabi, Sacha Lewin, and Gilles Louppe. A Neural Material Point Method for Particle-based Emulation. 2024. URL https://arxiv.org/abs/2408.15753.

[9] Nianyi Wang, Yu Chen, and Shuai Zheng. FluidFormer: Transformer with Continuous Convolution for Particle-based Fluid Simulation. 2025. URL https://arxiv.org/abs/2508.01537.

[10] Sunwoong Yang, Ricardo Vinuesa, and Namwoo Kang. Enhancing Graph U-Nets for Mesh-Agnostic Spatio-Temporal Flow Prediction. 2024. URL https://arxiv.org/abs/2406.03789.

[11] Nicholas Karris, Evangelos A. Nikitopoulos, Ioannis G. Kevrekidis, Seungjoon Lee, and Alexander Cloninger. Using Linearized Optimal Transport to Predict the Evolution of Stochastic Particle Systems. 2024. URL https://arxiv.org/abs/2408.01857.

[12] Honghui Du and QiZhi He. JAX-MPM: A Learning-Augmented Differentiable Meshfree Framework for GPU-Accelerated Lagrangian Simulation and Geophysical Inverse Modeling. 2025. URL https://arxiv.org/abs/2507.04192.