PREDICT-PROJECT TRANSFORMERS FOR STABLE LARGE-SCALE FLUID MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Simulating free-surface flows with particles is difficult. Particle connectivity changes at every step, and phenomena such as wave propagation emerge only when thousands of particles are coordinated across the whole domain and over hundreds of steps. We find that current learned simulators cannot reproduce these phenomena on core validation scenarios within SPH community. Graph networks are realistic locally but their reach is limited by the receptive range of message passing, while latent-space transformers capture global motion but are bottlenecked by oversmoothing. We also find that no public particle datasets are large enough to clearly measure these failure modes. To address this, we present the Predict-Project Transformer (P2T), which models near-field interactions with message passing and applies a far-field correction with latent attention. On two new datasets built from the SPH community's validation cases, a wave tank and a dam break, P2T is the only model that reproduces the propagating wave and the dam-break impact while keeping particle density close to the reference solver's. Ablations show that its configuration is crucial for rollout stability, and a pre-trained P2T adapts to new tanks, seas and breakers from a handful of simulations. We release the model along with five large-scale datasets.

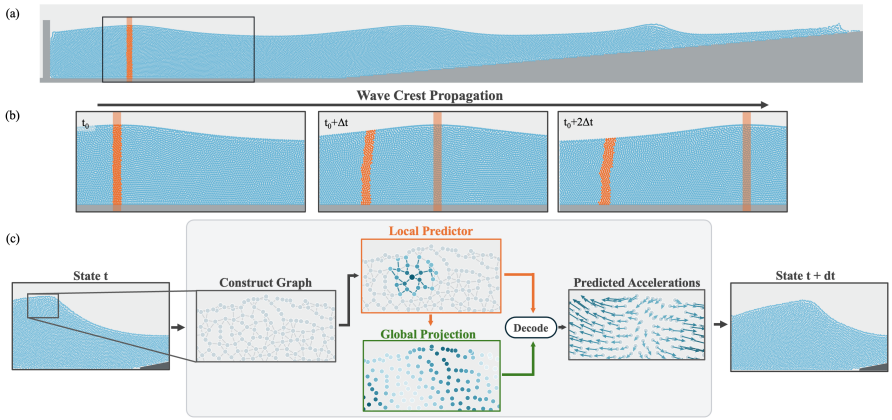

Figure 1: **(a) Waves emerge from near and far-range propagation.** A piston wavemaker drives regular waves onto a dissipative beach in a 9.4 m tank (19,618 particles). **(b)** The boxed region of (a) at three times: tagged particles (orange) oscillate with small amplitude as the crest passes through them, waves require substantial spatiotemporal coordination. **(c) P2T architecture** A message-passing *predictor* on the radius graph learns local interactions, an attention-based *projection* adds a low-rank global correction computed from the predictor's output, and a decoder combines both outputs into accelerations.

1 INTRODUCTION

Free-surface flows are central to marine and coastal engineering, where waves propagate, break, and splash onto structures such as floating platforms and wave energy converters (Violeau & Rogers,

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.

2016). Smoothed-particle hydrodynamics (SPH) (Domínguez et al., 2022) is the standard tool for high-fidelity simulation of these flows. However, a single run can take hours, which limits SPH to late-stage verification of designs that are already mature. Learned simulators promise to make high-fidelity simulation fast enough to use throughout the design process and to drive design optimization directly (Allen et al., 2022). Because they learn from data, they could also be trained on physical experiments (Hu et al., 2026a), narrowing the gap between simulated and real ocean dynamics before a design is deployed.

These flows carry important dynamics on two scales (Figure 1). Globally, waves emerge from the whole sea state and bring large loading forces onto structures, while impact forces such as splashing arise from fine-grained interaction with rigid structures. The same split appears inside the solver. Because water is nearly incompressible, pressure adjusts across the whole tank within a single step, while every other force acts between neighboring particles. Projection methods (Chorin, 1968; Cummins & Rudman, 1999) therefore advance the flow in two steps, a local *predictor* that applies every force except pressure, followed by a global *projection* that solves an elliptic PDE for the pressure.

We argue that existing learned simulators represent only one of these two steps. Graph networks aggregate information over neighbors like the predictor, and become learnable through message passing (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021) or spatial convolution (Ummenhofer et al., 2020), but their receptive field grows only with depth. Latent-space transformers learn the low-rank global maps needed for pressure projection and for steady-state prediction (Yang et al., 2026), but every interaction between two particles passes through a small set of latent tokens, so they cannot represent which particles are neighbors. At large scales, neither reproduces the emergent phenomena that SPH practitioners validate against. Graph networks lose the wave, and latent-space transformers keep the wave but let particles drift through one another.

We present the Predict-Project Transformer (P2T), which composes a message-passing predictor with an attention-based projector following the structure of projection methods (Figure 1c). Unlike previous multi-scale architectures, P2T is computationally tractable for very large-scale fluid simulation while maintaining accurate local predictions and stable global predictions.

To address a lack of public large-scale particle simulation datasets, especially for phenomena that are important to the SPH community such as wave propagation, we generate five datasets for our experiments: two used as training sets and three used as transfer learning tests. Our benchmark datasets are created from core validation cases from the SPH community and on average contain >4x as many particles as any comparable public dataset we are aware of. Our contributions are:

- We provide an analysis of learned simulators through the structure of two-scale projection methods, with applications in determining useful properties of a neural simulator as well as predicting failure modes of many existing simulators.
- We propose the Predict-Project transformer (P2T) which is the only model that accurately reproduces free surface flows and global phenomena like dam impact, and additionally maintains local density close to the reference simulator's.
- We show that a pre-trained P2T can be fine-tuned for new wave tanks, seas and breakers from a handful of simulations and a small number of additional optimizer steps.
- We release five large-scale datasets that exhibit important phenomena for validating SPH simulators.

2 Related work

Graph-based neural physics simulators. Graph-based neural physics simulators were originally introduced to speed up analytic fluid simulations by representing fluids as graph structures where each graph node represented a fluid particle (Sanchez-Gonzalez et al., 2020). Subsequent work extended these ideas to mesh-based simulation (Pfaff et al., 2021) and introduced alternative message passing mechanisms such as continuous convolutions or equivariant layers (Ummenhofer et al., 2020; Toshev et al., 2023). Although these approaches show strong generalization properties while also outperforming classic methods in speed, they only represent local interactions between particles.

Latent-token transformers Transformer layers can address the interaction locality problem by representing message passing on a fully-connected graph (Joshi, 2025; Kim et al., 2022). However, the computational complexity of attention makes scaling to tens of thousands of particles infeasible.

Current physics transformers address this problem via a *compress-process-reconstruct* paradigm where models compress particles to a few latent tokens, process via global attention, and reconstruct the output for each particle (Yang et al., 2026). Compression can be achieved by transforming to a set of predefined (e.g. spectral) basis vectors (Chen et al., 2024; Li et al., 2021), or it can also be learned. Methods such as Alkin et al. (2024) subsample a small number of "super nodes" which aggregate information from nearby particles into a compressed space while other methods use explicitly low-rank matrix transforms (Wu et al., 2024; Luo et al., 2025; Hu et al., 2026b). Reconstruct be handed with cross attention (Yang et al., 2026) or learned interpolation (Viswanath et al., 2026). This bottleneck leads to low-rank predictions that fit the global structure needed to solve many common PDEs (Boullé & Townsend, 2024), but as we show here, also results in inaccurate predictions of local phenomena like particle density.

Multiscale and hybrid networks. To get the best of both worlds, multiple multiscale and hybrid networks have been proposed.Fortunato et al. (2022) rely on a graph network representation, and simultaneously pass messages on a solver-generated coarse and fine mesh. Later work proposes new ways to generate coarse meshes through subsampling (Li et al., 2020b), breadth-first search (Cao et al., 2023), and graph partitioning (Lei et al., 2025).

These models rely on a set of precomputed hierarchies that are generated and cached before training, leading to poor generalization to new hierarchies. For example, Cao et al. (2023, App. A.4.2) reports up to $2.2\times$ higher rollout error when graph construction differs between training and test. This is a particular problem for fluid simulation because particle connectivity changes substantially over a simulation. Consequently, existing methods are usually limited to fixed meshes. Furthermore, hierarchal models struggle with error accumulation along partition boundaries and existing solutions are only applicable on regular grids (Mukhopadhyay et al., 2026). Those models which dynamically recompute connectivity have a quadratic runtime in particle count (Wang et al., 2026), making them impractical for large scale simulations. Unlike prior work, our approach recomputes connectivity dynamically while retaining linear runtime in particle count, eliminating errors at partition boundaries and working for any connectivity topology (not just grids).

3 Background

3.1 SPH projection methods

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain and $u : [0, T] \times \Omega \to \mathbb{R}^d$ and $p : [0, T] \times \Omega \to \mathbb{R}$ the velocity and pressure of an incompressible fluid, governed by the momentum equation

$$\partial_t u + (u \cdot \nabla)u = -\tfrac{1}{\rho}\nabla p + \nu\nabla^2 u + g$$

subject to the constraint $\nabla \cdot u = 0$. In particle simulations, Ω is discretized into a set of N particles at positions x^n with velocities u^n. Here we use superscript notation to denote discrete step index at time $t = n\,\Delta t$. Neural simulators learn effective accelerations a^n such that u^{n+1} can be recovered from Euler integration. We simplify our math throughout this paper by working with time in units equal to step size, meaning $\Delta t = 1$ and $a = u^{n+1} - u^n$.

The incompressible SPH (ISPH) method computes this map in two stages. A *predictor* applies every force except pressure at the current particle positions and gives a tentative velocity,

$$u^* = u^n + \mathcal{A}(u^n, x^n), \qquad \mathcal{A}(u^n, x^n) := -(u^n \cdot \nabla)u^n + \nu\nabla^2 u^n + g, \tag{1}$$

which in general violates the incompressibility constraint. A *projection* then removes the violation,

$$u^{n+1} = \mathcal{P}u^* := u^* - \nabla\phi, \qquad \nabla^2\phi = \nabla \cdot u^*, \tag{2}$$

where ϕ solves the pressure Poisson equation with the boundary conditions of the tank (Section B). The predictor and projection operator work on complementary scales. $\mathcal{A}$ is locally computed as a weighted sum over neighbors (Monaghan, 1994) while $\mathcal{P}$ acts to globally couple all particles. Writing $\mathcal{A}_h$ for the discrete SPH predictor and $\mathbf{G}_h, \mathbf{D}_h, \mathbf{L}_h$ for the sparse $N \times N$ matrices that apply

the SPH gradient, divergence, and Laplacian stencils respectively (Cummins & Rudman, 1999), the ISPH update on particles is

$$u^{n+1} = \left(\boldsymbol{I} - \mathbf{G}_h \mathbf{L}_h^{-1} \mathbf{D}_h\right)\left(u^n + \mathcal{A}_h(u^n, x^n)\right). \tag{3}$$

Equation (3) separates near and distant interactions. $\mathbf{G}_h$, $\mathbf{D}_h$ and the discrete predictor $\mathcal{A}_h$ each sum over local neighbors, while $\mathbf{L}_h^{-1}$, the discrete Green's function, couples every particle to every other, so every influence of a distant particle on the step passes through the solver (Section B). [1]

3.2 Translation to neural simulators

Differentiating Eq. (3) with respect to u^n gives the Jacobian of one simulation step as a product of two matrices,

$$\boldsymbol{K}_{\text{step}} = (\boldsymbol{I} + \mathbf{K}_{\text{far}})\,\mathbf{K}_{\text{near}}, \qquad \mathbf{K}_{\text{near}} = \boldsymbol{I} + \frac{\partial \mathcal{A}_h}{\partial u^n}, \qquad \mathbf{K}_{\text{far}} = -\mathbf{G}_h \mathbf{L}_h^{-1} \mathbf{D}_h. \tag{4}$$

Here $\mathbf{K}_{\text{near}}$ is the identity plus the Jacobian of a neighbor sum, making it banded with nonzero entries only between neighbors within a fixed radius. On the other hand, $\mathbf{K}_{\text{far}}$ is dense because $\mathbf{L}_h^{-1}$ is. Note that $\mathbf{K}_{\text{far}}$ acts on the *output* of $\mathbf{K}_{\text{near}}$, reflecting the fact that the source of the Poisson problem in Eq. (2) is the divergence of the tentative velocity u^*. Furthermore, the identity in the first factor passes that output directly to the result like a skip connection, since the projection acts as a correction to it.

Following Yang et al. (2026), we call Jacobians of this kind *interaction maps*, $N \times N$ matrices whose (i, j) entry says how much particle j influences particle i. This definition applies to network components as well as to the ISPH update step. Its sparsity determines the component's receptive field and it can be measured on trained models using backpropagation (Section I).

The terms in Eq. (4) describe how to computationally separate local and global interactions. Learned simulators need a *local predictor* whose interaction map is banded and a *global projector* whose interaction map is dense and applied to the predictor's output before being joined with a skip connection. We call the two roles a model's *near field* and the *far field*, and their arrangement a model's *configuration*.

4 Predict-Project Transformers

Based on the structure of Eq. (4), we present the Predict-Project Transformer (P2T). P2T achieves linear runtime scaling in particle count by using a message-passing network for local interactions and a latent-space transformer for global coupling (Figure 1). We first present the architecture then expand on each component to explain how P2T can learn fully connected interactions in linear time.

Architecture. Given the particle state at step n, we first build a graph over nearby particles, $\mathcal{E}_r = \{(i, j) : \|x_i - x_j\| < r\}$, and encode particle and edge features with MLPs into $\boldsymbol{H} \in \mathbb{R}^{N \times c}$ and $\boldsymbol{E}$ respectively. Input node features concatenate the particle's type (fluid, wall, or piston for the wave tank wavemaker) with the velocity history of that particle v_i over 5 history steps. Input edge features for edges connecting particle i and j are the relative displacement between the particles. P2T then learns to predict per-particle accelerations $\mathbf{a}_i$ in three stages

$$\mathbf{h}^{\text{near}} = \text{MP}_L\big(\boldsymbol{H}, \boldsymbol{E};\, \mathcal{E}_r\big), \tag{5}$$

$$\mathbf{h}^{\text{far}} = \text{LatentAttn}_M\big(\text{Enc}([\mathbf{h}^{\text{near}}, \mathbf{x}])\big), \tag{6}$$

$$\mathbf{a}_i = \text{Decode}\big(\mathbf{h}_i^{\text{near}} + \mathbf{h}_i^{\text{far}}\big), \qquad i = 1, \ldots, N, \tag{7}$$

Here MP_L represents L rounds of graph-based message passing, LatentAttn_M is a stack of latent-attention blocks operating on $M \ll N$ tokens, and Enc and Decode are MLPs applied pointwise. The three lines correspond to Eq. (3)'s predictor, projector, and sum respectively.

[1] Our data come from a weakly compressible solver that has no projection step, but at the low Mach numbers reached in our datasets, one dataset step spans hundreds of solver steps, and the resulting map between states has the same structure (Appendix C). Equation (3) is therefore the map a network has to represent.

Message passing as the local predictor (MP_L) We use the graph network processor of Sanchez-Gonzalez et al. (2020), which most closely mimics Eq. (3) by summing particle interactions from connected neighbors. Each round updates the edges and then the nodes with two MLPs f_E and f_V,

$$\mathbf{e}'_{ij} = \mathbf{e}_{ij} + f_E\big([\mathbf{e}_{ij}, \mathbf{h}_i, \mathbf{h}_j]\big), \qquad \mathbf{h}'_i = \mathbf{h}_i + f_V\Big(\big[\mathbf{h}_i, \textstyle\sum_j \mathbf{e}'_{ji}\big]\Big), \tag{8}$$

where the summation is computed over all neighbors in $\mathcal{E}_r$.

After L rounds, a particle has received information from particles within L hops, so the predictor's interaction map is banded, like $\mathbf{K}_{\mathrm{near}}$. Its range grows linearly with depth and connectivity radius r. This presents a challenge for pure message-passing networks on large-scale simulations. For example, the water in the wave tanks is hundreds of neighborhoods across (Sanchez-Gonzalez et al., 2020), so the pressure that couples the whole tank within one step is out of reach of message passing alone.

Latent attention ($\mathrm{LatentAttn}_M$) as the global projector The projection couples every particle to every other in a single step, and attention does the same in a single layer, so we use it for the far field. At tens of thousands of particles, full attention is too expensive, so we apply it through a bottleneck of $M \ll N$ tokens, similarly to latent-space transformers. In the compress–process–reconstruct template of Yang et al. (2026), one block updates hidden states $\boldsymbol{H}$ following

$$\boldsymbol{Z} = \underbrace{\boldsymbol{V}(\boldsymbol{H})^\top \boldsymbol{H}}_{\text{compress to } M \text{ tokens}}, \qquad \boldsymbol{Z}' = \underbrace{\mathrm{Attn}(\boldsymbol{Z})}_{\text{process}}, \qquad \boldsymbol{H}' = \boldsymbol{H} + \underbrace{\boldsymbol{U}(\boldsymbol{H})\,\boldsymbol{Z}'}_{\text{reconstruct particles}}, \tag{9}$$

where $\boldsymbol{V}(\boldsymbol{H})^\top$ and $\boldsymbol{U}(\boldsymbol{H})$ are input-dependent weights, computed per particle, that map particles to tokens and back, and Attn is a transformer layer on the tokens. Writing $\boldsymbol{A}$ for the linearization of Attn, the cross-particle part of the block is $\boldsymbol{U}\boldsymbol{A}\boldsymbol{V}^\top$. Transolver, LinearNO and LRSA all take this form and perform it in linear time with different methods for computing the weights (Wu et al., 2024; Hu et al., 2026b; Yang et al., 2026). We choose to use Transolver's Physics-Attention for the far field since Transolver is widely used and its slicing mechanism allows for token interpretability.

We use the bottleneck block for the far field because of its latent coupling mechanism. Every path from one particle to another passes through the M tokens, meaning that two particles can only interact through M shared channels, and the block's interaction map is diagonal plus a matrix of rank at most M (Proposition 1, Appendix A). This matches the structure of the projection in Eq. (3), since the Green's function of an elliptic problem is low-rank between well-separated regions and full-rank only near the diagonal (Bebendorf & Hackbusch, 2003; Boullé & Townsend, 2024). Everything banded therefore stays with message passing, including the pressure a particle feels from its own neighbors.

Configuration We configure the two components following the ISPH step. In Eq. (3) the projection acts on the output of the predictor, and the predictor's output also reaches the result directly. In P2T, the projector acts on the predictor's output and the decoder receives the sum of the two. Our ablation results also explore two alternative configurations that are present in the literature (Table 5). In the *parallel* configuration both components receive the encoded input and their outputs are summed, as in multiscale graph networks across levels. In the *no-skip* configuration the components are composed but everything passes through the latent tokens, as in UPT and GIOROM (Alkin et al., 2024; Viswanath et al., 2026), so the near field is never recomputed after the bottleneck.

5 Experiments and Results

Datasets. We train on both next-state prediction and steady-state prediction benchmarks. For next-state prediction, we contribute two datasets: the wave tank and the dam break (Table 6). They are the two standard validation cases of the SPH community, and their dynamics are very different. In the wave tank, a piston drives regular waves onto a beach. Particles keep the same neighbors for most of the rollout, while the wave travels the length of the tank. In the dam break, a column of water collapses. Neighbor sets change at every step, and the quantity of interest, the impact on the far wall, is a single global event. Both are generated with DualSPHysics (Domínguez et al., 2022), a weakly compressible SPH solver, at about 20,000 particles and 195 steps per simulation,

Table 1: **Rollout position RMSE (mm) after n autoregressive rollout steps.** We report the mean RMSE over test set simulations and 3 seeds. Best in **bold**, second best underlined. "div." marks diverged rollouts (Section G). Standard deviations are in the appendix tables 9 and 10.

	Wave tank					Dam break				
Model	n=1	20	50	100	190	n=1	20	50	100	190
GNS	0.212	7.56	35.0	107.5	188.9	**2.47**	**223**	822	989	1108
Transolver	0.345	9.62	42.6	70.7	131.7	4.84	318	991	1053	1033
UPT	0.377	11.0	20.7	51.7	115.7	30.4	860	1697	div.	div.
Erwin	0.321	18.4	36.0	71.2	172.0	5.29	1364	div.	div.	div.
P2T (ours)	**0.202**	**4.92**	**15.4**	**34.4**	**81.3**	2.95	260	**790**	**852**	**964**

with 400 simulations for training and 40 each for validation and test. For transfer, we generate three further wave-tank datasets, a shorter tank, irregular seas and plunging breakers (Section 5.1). Additional details for all five original datasets are in Appendix F. For steady-state prediction, we use the standard ShapeNet-Car dataset (Umetani & Bickel, 2018; Wu et al., 2024) in which the target is the air flow around a car rather than the next state of a rollout. It is the largest of the standard steady-state benchmarks, at about 32,000 mesh points per car.

Baselines. We compare against one model from each family in Table 5. GNS (Sanchez-Gonzalez et al., 2020) is a standard message-passing simulator and equivalent to P2T's predictor step. Transolver (Wu et al., 2024) is the most widely used latent-space transformer and equivalent to P2T's projector if used alone. UPT (Alkin et al., 2024) is the most widely used latent transformer-based model for particle data, and Erwin (Zhdanov et al., 2025) is a multiscale model that has been tested on the closest data to ours. Implementation details are in Appendix G.

We also compare to GIOROM (Viswanath et al., 2026) in Section H.4.

Metrics. We measure rollout accuracy as the RMSE of fluid particle positions at steps 1 to 190. Additionally, we report metrics used by the SPH community for validation (Rota Roselli et al., 2019; Lobovskỳ et al., 2014). These include and the free-surface error on the wave tank as well as the peak deflection error on the dam break (a global measure), which tell us whether the phenomenon of interest is reproduced. Furthermore, to compare local accuracy we compare how far the SPH density deviates from the reference simulator's (a local measure). Metric definitions, hyperparameters, and training details are in Appendix G. All models are trained with three seeds and we report full statistics in the Appendix.

5.1 Results

Our main findings are that P2T is the only model that is accurate at both the particle scale and the system scale, that this depends on combining the predictor and projector the way the ISPH step does rather than on having both components, that P2T outperforms global-only models even on the steady-state tasks they were designed for, and that a pre-trained P2T adapts to new tanks, seas and breakers from only a handful of simulations.

P2T captures global and local dynamics P2T is the only model that is accurate at both the particle scale and the system scale (Tables 1 and 2). On the wave tank dataset it has the lowest position error at every horizon, its density stays closest to the reference simulator's, and its free-surface error is the lowest of any model (Figure 3). On the dam break dataset it has the lowest position error from step 50 onward and predicts the peak run-up on the far wall to within a tenth of its height. P2T is the only model that recreates the impact, the deflection up the wall, and the air cavity that forms when the fluid falls back (Figure 4).

Graph networks capture local dynamics but not global ones. For example, GNS has low position RMSE for a single step of autoregressive prediction, but falls apart after many steps (Table 1). Similarly, its density deviation is the closest of any baseline to the reference simulator's on the dam break dataset (Table 2). However, the global phenomenon (the wave) lies beyond its receptive field

Table 2: **Physical metrics on held-out rollouts** (mean ± std over simulations and 3 seeds). Lower is better for the system-scale columns, closer to the reference for density deviation. Best learned model in bold, second best underlined. UPT and Erwin diverge mid-way through the dam break, so their density deviation is not reported; peak deflection occurs before step 50, so it is.

	Wave tank		Dam break	
Model	Free-surface err. $e_{\text{rms}}\downarrow$ *system-scale*	Density dev. $\overline{\delta\rho}$ *particle-scale*	Peak defl. err. (%) $\downarrow$ *system-scale*	Density dev. $\overline{\delta\rho}$ *particle-scale*
Ref.	0	0.050	0	0.146
GNS	0.467 ± 0.095	$\underline{0.086} \pm 0.021$	$\underline{22.5} \pm 1.3$	$\mathbf{0.277} \pm 0.018$
Transolver	0.604 ± 0.327	0.447 ± 0.149	24.2 ± 19.7	41.8 ± 45.0
UPT	0.454 ± 0.093	0.172 ± 0.019	119.4 ± 115.2	–
Erwin	$\underline{0.319} \pm 0.071$	0.374 ± 0.097	37.3 ± 23.7	–
P2T (ours)	$\mathbf{0.196} \pm 0.014$	$\mathbf{0.068} \pm 0.002$	$\mathbf{8.3} \pm 2.5$	$\underline{0.417} \pm 0.090$

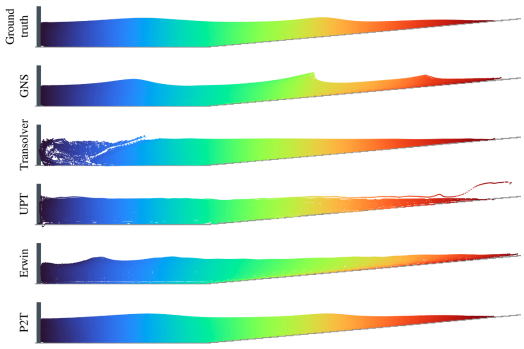

Figure 2: **Wave-tank rollouts**: Rollout snapshot at $t = 5.0s$. GNS's wave starts thinner and crashes before the beach, the latent-space transformers keep the bulk motion but their particles lose their spacing, and P2T keeps both. Full in appendix

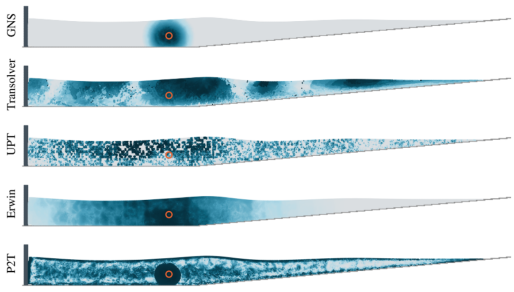

Figure 3: **Interaction maps of trained models.** For one target particle (ringed), color gives the percentile rank of every other particle's influence on the target's predicted acceleration (grey: exactly zero). Computed on ground-truth input. More targets and color scales are in Appendix I

and is therefore not well modeled (Figure 3). On the dam break it under-predicts the run-up, since the height of the run-up is set by the pressure of the whole column (Figure 12).

Latent-space transformers capture global dynamics but not local ones. Transolver, UPT and Erwin track the bulk motion of the wave better than GNS at long horizons, but their particles drift through one another (Proposition 1). Their density deviates from the reference by several times more than GNS's. Transolver has the worst free-surface error of any model and loses the wave first (Figures 3 and 9). On dam break, UPT and Erwin diverge before step 100.

The interaction maps of the trained models confirm that P2T has learned the structure of Eq. (4) (Figure 5). Its map combines a near field that looks like GNS's, with about half of its influence within a few neighborhoods of the target, and a far field that follows the free surface near the target rather than covering the whole tank. Transolver's map is nearly the same for every target, and the maps of UPT and Erwin show the supernode sampling and ball partition of their architectures rather than the physics. P2T has learned both fields of the step.

Following the ISPH step faithfully is important In the ISPH step, the projector acts on the output of the predictor, and the predictor's output is also passed directly to the final solver (Eq. (3)). We ablate each of these on the wave tank against the parallel and no-skip configurations of Section 4, with the same components and the same parameter budget as P2T (Table 3). The parallel configuration is nearly three times worse than P2T at 190 steps and the no-skip configuration nearly nine times worse. Every alternative method in Table 5 combines its components in one of these two

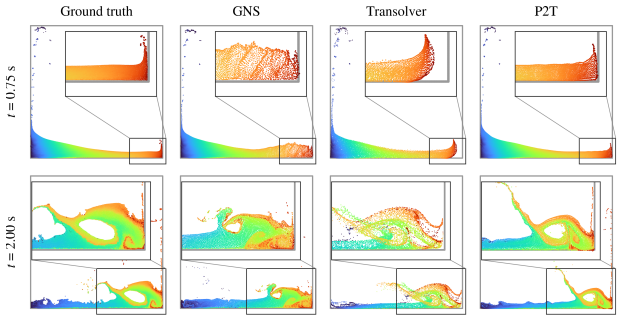

Figure 4: **Dam-break rollouts** at impact ($t = 0.75$ s, top) and after the surge falls back ($t = 2.00$ s, bottom), particles colored by initial position. Only P2T reproduces the run-up on the wall and the air cavity that forms when it collapses.

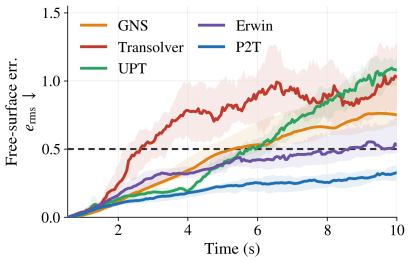

Figure 5: **Free-surface error over the wave-tank rollout** (e_{rms}, median and IQR over test simulations). Dotted: a model's error passes half the wave height, at 5.3, 2.65, 5.75 and 8.65 s for GNS, Transolver, UPT and Erwin, and never for P2T.

Table 3: **Ablations on the wave tank** (rollout position RMSE in mm, mean ± std over 3 seeds). *Parallel* gives the encoded input to both components and decodes their sum. *No skip* removes the skip connection.

Variant	n=1	20	50	100	190
P2T	**0.20**	**4.9 ± 0.8**	**15.4 ± 3.9**	**34.4 ± 8.1**	**81.3 ± 37.7**
Parallel	0.22	8.5 ± 3.6	27.9 ± 9.2	80.2 ± 28.1	229 ± 125
No skip	0.30	19.5 ± 5.2	69.3 ± 16.3	243 ± 152	699 ± 627

Table 4: **Comparison on ShapeNet-Car. Bes** results are marked. First section uses results reportedfrom Wu et al. (2024). We trained GNS and P2T.

Model	Vol ↓	Surf ↓	ρ_D ↑	
GNO	0.0383	0.0815	0.0172	0.9834
GNOT	0.0329	0.0798	0.0178	0.9833
GINO	0.0386	0.0810	0.0184	0.9826
Transolver	<u>0.0207</u>	<u>0.0745</u>	**0.0103**	<u>0.9935</u>
GNS	0.0378	0.0837	0.0195	0.9786
P2T (Ours)	**0.0196**	**0.0713**	<u>0.0107</u>	**0.9940**

ways, which suggests that all of them should be worse than P2T for the same reason, even though we do not explicitly test all of them here.

P2T outperforms global-only approaches on steady-state prediction tasks Latent-space transformers were designed for steady-state tasks such as ShapeNet-Car and generally outperform graph networks on them, but P2T outperforms both (Table 4). Error distributions are shown in Figure 13; Transolver's errors are generally higher frequency relative to GNS.

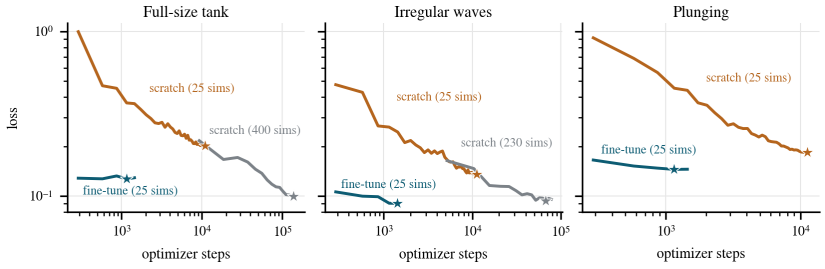

Figure 6: **Fine-tuning a P2T pre-trained on the 4.8 m short tank** (validation loss solid, training loss dashed, star at the best validation). **Left:** the 9.4 m wave tank, which doubles the tank length. **Middle:** plunging breakers, with wave heights of $[0.20, 0.30]$ m against $[0.05, 0.15]$ m in pre-training and a steeper beach. **Right:** irregular JONSWAP seas, a different forcing. Fine-tune curves start at step 0 on the target after 138,750 pre-training steps.

P2T adapts to new tanks, seas, and breakers We pre-train P2T on a short tank with a steeper beach and fine-tune it on 25 simulations of each of three new scenarios, the full-length wave tank, irregular seas from a JONSWAP spectrum (Hasselmann et al., 1973), and plunging breakers (Figure 6, Section F). We compare against training from scratch on the same 25 simulations and on 400 simulations.

A pre-trained P2T adapts to each new domain from a handful of simulations (Figure 6) substantially better than training from scratch. Within **a few hundred optimizer steps**, fine-tuning P2T outperforms the best possible model obtained by training from scratch on the same data after 10000 optimizer steps for any generalization scenario. Even if training from scratch using 400 simulations, fine-tuning P2T on just 25 simulations performs comparably with far fewer optimization steps. The representation that P2T learns on one tank therefore transfers to new tanks, seas and breakers.

6 Discussion and Conclusion

In this paper we introduce P2T, a hybrid model inspired by SPH solvers that can predict large-spatial-scale fluid phenomena like fluid propagation in wave tanks. Alongside it we introduce two datasets, a wave tank and a dam break, which are the standard validation cases of the SPH community. The wave tank requires a level of spatial and temporal coordination which, to our knowledge, is not required by any comparable dataset. On both, P2T is the only model that reproduces the propagating wave and the impact on the far wall, and on the wave tank its particle density is the closest of any model to the reference solver's. Graph networks maintain locally realistic dynamics, but their limited receptive field produces inaccurate wave propagation. Latent-space transformers track waves, but their bottlenecks lose neighbor information (Proposition 1), resulting in particles that drift through one another. Beyond these two rollout tasks, P2T also outperforms Transolver on ShapeNet-Car, and its runtime scales linearly, supporting hundred-thousand-particle simulations (Section H.7).

We attribute P2T's accuracy at both scales to the configuration of its two components rather than to the components themselves. Ablating against commonly used configurations shows that P2T's configuration is crucial for rollout stability (Table 3). Furthermore, interaction maps confirm that P2T learns the two fields matching the ISPH step (Figure 5).

In SPH, the interactions between neighboring particles remain the same across scenarios, while only the global coupling changes with the geometry and the forcing, so a model configured this way should have little to relearn. We find that P2T indeed fine-tunes to domains far outside its pre-training regime from a handful of simulations (Section 5.1). This transferability suggests a route to simulators trained on physical experiments (Hu et al., 2026a), narrowing the sim-to-real gap. A model pre-trained on solver data could be fine-tuned on sparse measured trajectories or even video.

Limitations. We emphasize that a graph network remains the better choice when the domain is small. P2T's advantage grows with domain size (Figure 7), and on public benchmarks a few tens of connectivity radii across, GNS is as accurate or better (Section H.2), since most of the domain is already within the predictor's reach. On the dam break, P2T's gain is mainly at system scale, though it performs second best on density deviation. All of our experiments are two-dimensional. Three-dimensional tanks are within reach computationally, since P2T's cost is linear in the number of particles (Section H.7), but we have not tested them. Our analysis assumes the low-Mach limit (Section C), is reasonable for calm waves but could break in cases with extreme fluid-structure interactions.

Future work. Any bottleneck block can serve as the projector, so newer operators such as LinearNO or LRSA (Hu et al., 2026b; Yang et al., 2026) can be slotted in for Transolver's Physics-Attention without changing the rest of P2T. Inference-time SPH corrections (Toshev et al., 2024), which apply to every model here, could be added to further improve performance. More broadly, the analysis of Section 3 applies to any stiff system whose interactions are local except for a global constraint, such as cloth and elastic materials, and we expect the same configuration to carry over. We will release the model, the five datasets and the code that generates them, and we hope they give the marine-engineering and machine-learning communities common ground on which to build simulators that are useful in the real world.

Ethics Statement

We hope that this work accelerates the development of engineering-viable fluid simulators, and can aid in developing renewable marine energy devices.

Reproducibility statement

We will provide all datasets and code needed to generate the datasets and reproduce the experiments on publication.

AI use statement

We used generative AI tools to help revise our drafts and check citations. We also used generative AI as a starting point for making figures of our experimental data, but the code was manually reviewed and verified to accurately represent the data. No experiment was designed or run, and no data were created by generative AI. We take responsibility for the final content of this work.

References

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, pp. 25152–25194, 2024.

Kelsey R. Allen, Tatiana Lopez-Guevara, Kimberly L. Stachenfeld, Alvaro Sanchez-Gonzalez, Peter Battaglia, Jessica B. Hamrick, and Tobias Pfaff. Inverse design for fluid-structure interactions using graph network simulators. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, volume 35, pp. 13759–13774, 2022. URL https://arxiv.org/abs/2202.00728.

Mario Bebendorf and Wolfgang Hackbusch. Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients. *Numerische Mathematik*, 95:1–28, 2003. doi: 10.1007/s00211-002-0445-6.

Nicolas Boullé and Alex Townsend. A mathematical guide to operator learning. In *Numerical Analysis Meets Machine Learning*, volume 25 of *Handbook of Numerical Analysis*, pp. 83–125. Elsevier, 2024. doi: 10.1016/bs.hna.2024.05.003.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations (ICLR)*, 2022.

Yadi Cao, Menglei Chai, Minchen Li, and Chenfanfu Jiang. Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 3541–3558, 2023. URL https://proceedings.mlr.press/v202/cao23a.html.

Gengxiang Chen, Xu Liu, Qinglu Meng, Lu Chen, Changqing Liu, and Yingguang Li. Learning neural operators on Riemannian manifolds. *National Science Open*, 3(6):20240001, 2024. doi: 10.1360/nso/20240001.

Alexandre Joel Chorin. Numerical solution of the Navier–Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968. doi: 10.1090/S0025-5718-1968-0242392-2.

A. J. C. Crespo, J. M. Domínguez, B. D. Rogers, M. Gómez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, and O. García-Feal. DualSPHysics: Open-source parallel CFD solver based on smoothed particle hydrodynamics (SPH). *Computer Physics Communications*, 187: 204–216, 2015. doi: 10.1016/j.cpc.2014.10.004.

Sharen J. Cummins and Murray Rudman. An SPH projection method. *Journal of Computational Physics*, 152(2):584–607, 1999. doi: 10.1006/jcph.1999.6246.

J. M. Domínguez, G. Fourtakas, C. Altomare, R. B. Canelas, A. Tafuni, O. García-Feal, I. Martínez-Estévez, A. Mokos, R. Vacondio, A. J. C. Crespo, B. D. Rogers, P. K. Stansby, and M. Gómez-Gesteira. DualSPHysics: from fluid dynamics to multiphysics problems. *Computational Particle Mechanics*, 9(5):867–895, 2022. doi: 10.1007/s40571-021-00404-2.

Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. Multiscale meshgraphnets. In *ICML 2022 2nd AI for Science Workshop*, 2022. URL https://openreview.net/forum?id=G3TRIsmMhhf.

K. Hasselmann, T. P. Barnett, E. Bouws, H. Carlson, D. E. Cartwright, K. Enke, J. A. Ewing, H. Gienapp, D. E. Hasselmann, P. Kruseman, A. Meerburg, P. Müller, D. J. Olbers, K. Richter, W. Sell, and H. Walden. Measurements of wind-wave growth and swell decay during the Joint North Sea Wave Project (JONSWAP). *Ergänzungsheft zur Deutschen Hydrographischen Zeitschrift, Reihe A*, (12):1–95, 1973.

Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *European Conference on Computer Vision*, pp. 289–305. Springer, 2024.

Peiyan Hu, Haodong Feng, Hongyuan Liu, Tongtong Yan, Wenhao Deng, Tianrun Gao, Rong Zheng, Haoren Zheng, Chenglei Yu, Chuanrui Wang, Kaiwen Li, Zhi-Ming Ma, Dezhi Zhou, Xingcai Lu, Dixia Fan, and Tailin Wu. RealPDEBench: A benchmark for complex physical systems with real-world data. In *International Conference on Learning Representations (ICLR)*, 2026a. URL https://arxiv.org/abs/2601.01829.

Wenjie Hu, Sidun Liu, Peng Qiao, Zhenglun Sun, and Yong Dou. Transolver is a linear transformer: Revisiting physics-attention through the lens of linear attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 408–416, 2026b. doi: 10.1609/aaai.v40i1.37003. URL https://arxiv.org/abs/2511.06294.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2022.

Chaitanya K. Joshi. Transformers are graph neural networks. *arXiv preprint arXiv:2506.22084*, 2025. URL https://arxiv.org/abs/2506.22084.

Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, 2022. URL https://arxiv.org/abs/2207.02505.

Bo Lei, Victor M Castillo, and Yeping Hu. M4GN: Mesh-based multi-segment hierarchical graph network for dynamic simulations. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=R3vDbqWa1v.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020a. URL https://arxiv.org/abs/2003.03485.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, volume 33, pp. 6755–6766, 2020b. URL https://arxiv.org/abs/2006.09535.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.08895.

Libor Lobovskỳ, Elkin Botia-Vera, Filippo Castellana, Jordi Mas-Soler, and Antonio Souto-Iglesias. Experimental investigation of dynamic pressure loads during dam break. *Journal of Fluids and Structures*, 48:407–434, 2014.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41432–41449, 2025. URL https://arxiv.org/abs/2502.02414.

Joseph J. Monaghan. Simulating free surface flows with SPH. *Journal of Computational Physics*, 110(2):399–406, 1994. doi: 10.1006/jcph.1994.1034.

Payel Mukhopadhyay, Michael McCabe, Ruben Ohana, and Miles Cranmer. Overtone: Cyclic patch modulation for clean, efficient, and flexible physics emulators. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2507.09264.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03409.

Riccardo Angelini Rota Roselli, Giuliano Vernengo, Stefano Brizzolara, and Roberto Guercio. SPH simulation of periodic wave breaking in the surf zone – a detailed fluid dynamic validation. *Ocean Engineering*, 176:20–39, 2019. doi: 10.1016/j.oceaneng.2019.02.013.

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 (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8459–8468, 2020.

Stefan A. Sauter and Christoph Schwab. *Boundary Element Methods*, volume 39 of *Springer Series in Computational Mathematics*. Springer, 2010. doi: 10.1007/978-3-540-68093-2.

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 36 (NeurIPS), Datasets and Benchmarks Track*, 2023.

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*, volume 235 of *Proceedings of Machine Learning Research*, 2024. URL https://arxiv.org/abs/2402.06275.

Nobuyuki Umetani and Bernd Bickel. Learning three-dimensional flow for interactive aerodynamic design. *ACM Transactions on Graphics*, 37(4):89:1–89:10, 2018. doi: 10.1145/3197517.3201325.

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

Damien Violeau and Benedict D. Rogers. Smoothed particle hydrodynamics (SPH) for free-surface flows: Past, present and future. *Journal of Hydraulic Research*, 54(1):1–26, 2016. doi: 10.1080/00221686.2015.1119209.

Hrishikesh Viswanath, Yue Chang, Aleksey Panas, Julius Berner, Peter Yichen Chen, and Aniket Bera. Learning lagrangian interaction dynamics with sampling-based model order reduction. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=vXCQA1EzaG.

Caoliwen Wang, Minghao Guo, Siyuan Chen, Heng Zhang, Mengdi Wang, Xingyu Ni, Hanson Sun, Kunyi Wang, Zherong Pan, Kui Wu, Lingjie Liu, Yin Yang, Chenfanfu Jiang, Taku Komura, Wojciech Matusik, and Peter Yichen Chen. WorldParticle: Unified world simulation of Lagrangian particle dynamics via transformer. *arXiv preprint arXiv:2605.15305*, 2026. URL https://arxiv.org/abs/2605.15305.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53681–53705, 2024. URL https://arxiv.org/abs/2402.02366.

Zherui Yang, Haiyang Xin, Tao Du, and Ligang Liu. Simple yet effective: Low-rank spatial attention for neural operators. *arXiv preprint arXiv:2604.03582*, 2026. URL https://arxiv.org/abs/2604.03582.

Maksim Zhdanov, Max Welling, and Jan-Willem van de Meent. Erwin: A tree-based hierarchical transformer for large-scale physical systems. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78023–78040, 2025. URL https://arxiv.org/abs/2502.17019.

A Proof of Proposition 1

This appendix states and proves the claim of Section 4 that a bottleneck block is diagonal plus low-rank. The key observation is that every path from one particle's features to another's passes through the M tokens, so the off-diagonal part of the Jacobian factors through an M-dimensional space regardless of the nonlinearities on either side.

Proposition 1 (Bottleneck blocks are diagonal plus low-rank)**.** *For the block of Eq.* (9)*, with* $\boldsymbol{Z} = \boldsymbol{Z}(\boldsymbol{H}) \in \mathbb{R}^{M\times c}$ *any differentiable function of* $\boldsymbol{H}$*,* Attn *any differentiable function of* $\boldsymbol{Z}$*, and row* i *of* $\boldsymbol{U}$ *a function of* $\mathbf{h}_i$ *alone, every interaction map is* $\boldsymbol{K} = \boldsymbol{\Lambda} + \boldsymbol{U}\boldsymbol{B}$ *with* $\boldsymbol{\Lambda}$ *diagonal and* $\boldsymbol{B} \in \mathbb{R}^{M\times N}$*, that is, diagonal plus rank at most* M*.*

Proof. Write $\boldsymbol{Z}' = \mathrm{Attn}(\boldsymbol{Z}) \in \mathbb{R}^{M\times c}$ for the processed tokens, so that $h'_{i,a} = h_{i,a} + \sum_{m=1}^{M} U_{im} Z'_{m,a}$. Fix a channel pair (a, b), write $\boldsymbol{K}^{(a,b)} = [\partial h'_{i,a}/\partial h_{j,b}]_{ij}$ for its interaction map, and take $j \neq i$. The residual term and, by hypothesis, row i of $\boldsymbol{U}$ do not depend on $\mathbf{h}_j$, and $\boldsymbol{Z}'$ depends on $\boldsymbol{H}$ only through $\boldsymbol{Z}$, so by the chain rule

$$\frac{\partial h'_{i,a}}{\partial h_{j,b}} = \sum_{m=1}^{M} U_{im}\, B_{m,j}, \qquad B_{m,j} := \frac{\partial Z'_{m,a}}{\partial h_{j,b}} = \sum_{m'=1}^{M}\sum_{b'=1}^{c} \frac{\partial \mathrm{Attn}_{m,a}}{\partial Z_{m',b'}}\, \frac{\partial Z_{m',b'}}{\partial h_{j,b}}, \qquad j \neq i, \tag{10}$$

where U_{im} depends on i alone and $B_{m,j}$ on j alone. Hence $\boldsymbol{K}^{(a,b)}$ agrees off the diagonal with $\boldsymbol{U}\boldsymbol{B}$, $\boldsymbol{B} \in \mathbb{R}^{M\times N}$, i.e. $\boldsymbol{K}^{(a,b)} = \boldsymbol{\Lambda} + \boldsymbol{U}\boldsymbol{B}$ for a diagonal $\boldsymbol{\Lambda}$, and $\mathrm{rank}(\boldsymbol{U}\boldsymbol{B}) \leq M$. □

If row i of $\boldsymbol{U}$ also depends on the tokens $\boldsymbol{Z}'$, as in the cross-attention decoders of UPT and LRSA, the same argument applies with $\boldsymbol{Z}$ in place of $\boldsymbol{Z}'$, because every dependence of $h'_{i,a}$ on $\mathbf{h}_j$ still passes through the Mc entries of $\boldsymbol{Z}$, so the off-diagonal rank is at most Mc. Which case each model falls into is worked out in Section D.

B Next-state versus steady-state tasks

This appendix makes the distinction between the two kinds of task concrete on the ISPH step of Cummins & Rudman (1999). For steady-state tasks, the target is often the solution operator of an elliptic problem. For next-state tasks, the kernel instead appears between two banded operators.

One ISPH step. ISPH (Cummins & Rudman, 1999) is Eqs. (1) and (2) on particles. The predictor performs a local kernel sum, the pressure Poisson equation is solved with Neumann conditions on the walls and $\phi = 0$ on the free surface, and the projection subtracts the SPH gradient of ϕ. Writing $G(x, y)$ for the Green's function of the Laplacian with these boundary conditions, the solve is

$$\phi(x) = \int_{\Omega} G(x, y)\, (\nabla \cdot u^*)(y)\, dy. \tag{11}$$

On particles, row i of the pressure correction is $\sum_j \sum_{k,l} (\mathbf{G}_h)_{ik} (\mathbf{L}_h^{-1})_{kl} (\mathbf{D}_h)_{lj}\, u^*_j$, and since $\mathbf{G}_h$ and $\mathbf{D}_h$ are nonzero only between neighbors, a distant particle j can affect i only through an entry of $\mathbf{L}_h^{-1}$. This is the sense in which the solve is the only global factor of the step.

Steady-state tasks. At a steady state $u^{n+1} = u^n$, meaning the pressure correction cancels $\mathcal{A}$ exactly and the only nontrivial computation in the step is solving Eq. (11). This framework applies to many steady-state tasks. In the Darcy flow dataset (Li et al., 2021), for example, the target is the pressure u that solves $-\nabla \cdot (a\nabla u) = f$ for a given permeability field a and forcing function f. The solution operator is given by the Green's representation formula (Sauter & Schwab, 2010, Thm. 3.1.6),

$$u(x) = \int_\Omega G_a(x, y)\, f(y)\, dy. \tag{12}$$

which many neural operators such as GKN, MGNO, Transolver and LRSA are built to learn (Li et al., 2020a;b; Wu et al., 2024; Yang et al., 2026).

These operators often exhibit rapidly decaying singular values, allowing efficient low-rank representations (Boullé & Townsend, 2024). However, the residual error of such models concentrates around areas such as boundary layers and shocks where the solution changes rapidly (Yang et al., 2026; Wu et al., 2024).

Next-state tasks. In ISPH, a similar term appears but with local factors. In Eq. (4) the far field is $\mathbf{K}_{\text{far}} = -\mathbf{G}_h \mathbf{L}_h^{-1} \mathbf{D}_h$, composing the solve in between the divergence stencil $\mathbf{D}_h$ and gradient stencil $\mathbf{G}_h$. $\mathbf{L}_h^{-1}$ is a discrete Green's function which also lends well to low-rank approximation. In contrast, $\mathbf{G}_h$ and $\mathbf{D}_h$ interact over neighbors, which amplify high frequencies rather than damping them. On particles their interaction maps are banded matrices whose singular values do not decay and thus cannot be approximated well with low-rank approaches.

C Weakly compressible SPH at large time steps

Though our data are generated with a weakly compressible SPH solver, we argue that the local-global split is quite fitting at the large time steps that DualSPHysics by default writes outputs at (and thus at the time steps that our data are sampled at).

In weakly compressible SPH (WCSPH), pressure arises from a stiff equation of state and propagates acoustically (Monaghan, 1994). Information travels at two speeds, with acoustic (pressure) and advective (material-based) operating locally but at different speeds.

Over the s solver steps that make up one dataset step the two speeds result in drastically different propagation ranges. In the wave tank ($s \approx 576$, $c_s = 38.6$ m/s, $h = 0.0170$ m) sound travels 1.93 m per model step, or 114 smoothing lengths, and reflects within the domain, while material moves 0.33 smoothing lengths on average. In the dam break ($s \approx 1{,}710$, $c_s = 68$ m/s) sound travels 3.4 m and material 2.5 smoothing lengths.

While the ISPH-based theory is idealized for the sake of a cleaner framework, the idealizations mostly hold and are a faithful representation of real-world fluid dynamics. Density variations are remain small, with a mean relative deviation of 0.05% in the wave tank and 0.146% in the dam break. The acoustic horizon that defines $\mathbf{K}_{\text{far}}$ is finite but covers most of the domain.

D Existing simulators in the template

Table 5 writes each model family within Eq. (4), based on Yang et al. (2026, App. B) for the far field. Throughout, $\boldsymbol{H} \in \mathbb{R}^{N \times c}$ is the matrix of particle features.

GNS and MeshGraphNets (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021). L rounds of message passing on the radius graph give $\mathbf{K}_{\text{near}} = \prod_{\ell=1}^{L} \boldsymbol{K}^{(\ell)}$ so $\mathbf{K}_{\text{near}}$ is supported on L-hop neighborhoods. The model has a max receptive field of radius Lr, far field interactions can only be learned within this range.

Transolver (Wu et al., 2024). Transolver instead uses physics attention among M slices. Each particle is assigned to the slices by weights $\mathbf{s}_i = \text{softmax}_M(\boldsymbol{W}\mathbf{h}_i/\tau)$ computed from its own features, stacked into $\boldsymbol{S} \in \mathbb{R}^{N \times M}$. The block compresses, $\boldsymbol{Z} = \text{diag}(\boldsymbol{S}^\top \mathbf{1})^{-1} \boldsymbol{S}^\top \boldsymbol{H}$, mixes with

Table 5: **Existing simulators fill one of the two roles in Eq. (4), and P2T fills both in the configuration of the ISPH step.** Mc replaces M when the reconstruction weights depend on the tokens. Derivations are in Appendix D. **This table could be simplified to does the network have a non-trivial near field, does the network have a far field. Right now far field is confusing because $\mathbf{K}_{\text{far}}$ is dense, but also low rank and $\boldsymbol{U}\boldsymbol{A}\boldsymbol{V}^\top$ is dense and low rank**

	Near field	Far field	Configuration
Projection step Eq. (3)	$\mathbf{K}_{\text{near}}$	$\mathbf{K}_{\text{far}}$	sequential, skip
Message passing (GNS, MeshGraphNets)	$\mathbf{K}_{\text{near}}$, reach Lr	none	–
Multiscale attention (Erwin)	attention in balls	attention on coarsened balls	parallel, fixed partition
Latent attention (Transolver, LinearNO, LRSA)	none	$\boldsymbol{U}\boldsymbol{A}\boldsymbol{V}^\top$, rank $\leq M$	parallel, on raw input
Encode–process–decode (UPT, GIOROM)	$\mathbf{K}_{\text{near}}$, limited to 1-2 hops	$\boldsymbol{U}\boldsymbol{A}^T\boldsymbol{V}^\top$, rank $\leq Mc$, no diagonal	sequential, no skip
P2T (ours)	$\mathbf{K}_{\text{near}}$	$\boldsymbol{U}\boldsymbol{A}\boldsymbol{V}^\top$ stack, rank $\leq L_{\text{att}}M$	sequential, skip

attention among the slices, and reconstructs with the same weights, $\boldsymbol{H}' = \boldsymbol{H} + \boldsymbol{S}\boldsymbol{Z}'$. This corresponds to $\boldsymbol{U} = \boldsymbol{S}$ and $\boldsymbol{V}^\top = \operatorname{diag}(\boldsymbol{S}^\top\mathbf{1})^{-1}\boldsymbol{S}^\top$, with $\boldsymbol{U} \propto \boldsymbol{V}$, and the feed-forward path adds only diagonal terms. The near field is empty, since the only path from one particle to another runs through the M slices.

LinearNO (Hu et al., 2026b) and LRSA (Yang et al., 2026). LinearNO keeps the slicing used by Transolver but decouples $\boldsymbol{U}$ from $\boldsymbol{V}$ and simplifies the token mixing. LRSA replaces the slicing with cross-attention from M learned queries to particles, so its reconstruction weights depend on the tokens and its rank is bounded by Mc (Section A).

UPT (Alkin et al., 2024). UPT applies its bottleneck once per rollout rather than once per layer. A single message-passing layer aggregates particles onto supernodes, $\tilde{\boldsymbol{H}} = \boldsymbol{K}^{(1)}\boldsymbol{H}$, a Perceiver-style cross-attention (Jaegle et al., 2022) pools them into M latent tokens, a transformer advances the tokens, and finally a cross-attention decoder reads the state at any query position from tokens. One step is therefore $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{A}\boldsymbol{V}^\top\boldsymbol{K}^{(1)}$, in the order of the ISPH step, but nothing reaches the decoder except through the tokens, so the configuration is $\mathrm{Decode}(\mathrm{Far}(\mathrm{Near}(\boldsymbol{H})))$ with no skip connection. When UPT advances T steps in latent space between one encoding and one decoding, the map is $\boldsymbol{U}\boldsymbol{A}^T\boldsymbol{V}^\top\boldsymbol{K}^{(1)}$.

Erwin (Zhdanov et al., 2025). Erwin applies attention within the balls of a ball tree and moves between levels by tree coarsening and refinement. The interaction map at each level is block-diagonal, and the near field is a partition rather than a band. This extends in the coarser levels to the far field whose rank is set by the number of balls. However, these partitions lead to errors across ball boundaries visible in Figure 5.

GIOROM (Viswanath et al., 2026). GIOROM samples a subset of the particles, evolves them with a GNOT processor, and reconstructs the remaining particles using learned interpolation from nearby sampled points. Every output is therefore a function of the sampled points alone. The rank of the interaction map cannot exceed their number. A reconstructed particle never depends on the particles beside it, so the near field is defined by interpolation kernel rather local interactions. The configuration is sequential with no skip connection, the no-skip row of Table 3. The discrete nature of its sampling pattern leads is visible in Figure 22.

P2T. In P2T, the near field is composed of L message passing steps as $\mathbf{K}_{\text{near}} = \prod_{\ell=1}^{L}\boldsymbol{K}^{(\ell)}$. The far field is composed of a stack of L_{att} Transolver blocks applied to $\mathbf{K}_{\text{near}}\boldsymbol{H}$, resulting in a map that is diagonal plus rank at most $L_{\text{att}}M$. The decoder acts on the sum of the two, so the far field sees the near field's output and the near field keeps a skip connection to the output. To our knowledge, P2T is the only network with the components matching Eq. (4): a banded near field, a low-rank far field applied to its output, and a skip connection for the predictor, appear together only here.

E ARCHITECTURE DETAILS

E.1 MESSAGE-PASSING PREDICTOR

Each round of message passing consists of an edge update and a node update, each with a residual connection (Sanchez-Gonzalez et al., 2020). The edge update uses an MLP $f_E^{(\ell)}$. The hidden state $\mathbf{e}_{ij}$ of edge (i, j) is updated from itself and the hidden states of nodes i and j,

$$\mathbf{e}'_{ij} = \mathbf{e}_{ij} + f_E^{(\ell)}\big([\mathbf{e}_{ij}, \mathbf{h}_i, \mathbf{h}_j]\big). \tag{13}$$

The node update uses an MLP $f_V^{(\ell)}$ on the node's state and the sum of its incoming updated edge states,

$$\mathbf{h}'_i = \mathbf{h}_i + f_V^{(\ell)}\Big(\big[\mathbf{h}_i, \textstyle\sum_j \mathbf{e}'_{ji}\big]\Big). \tag{14}$$

Since only fluid particles are accelerated, the node update is not applied to boundary particles, which we found to slightly improve rollouts over updating all particles. After L rounds, $\mathbf{h}^{\text{near}} = \boldsymbol{H}^{(L)}$.

E.2 SLICE-ATTENTION PROJECTOR

The projector is a stack of Transolver blocks (Wu et al., 2024). Its input is $\mathbf{y}_i^0 = \mathrm{Enc}([\mathbf{h}_i^{\text{near}}, x_i])$, an MLP of the predictor's output and the absolute position, following the original implementation. Each block first assigns particle i a membership over M slices,

$$\mathbf{s}_i = \mathrm{softmax}\left(\boldsymbol{W}\mathbf{y}_i/\tau\right) \in \mathbb{R}^M, \tag{15}$$

where $\boldsymbol{W} \in \mathbb{R}^{M\times c}$ is learned and τ is a learnable temperature. The softmax over slices normalizes each particle's membership and makes slices compete for particles, as in slot attention (Locatello et al., 2020). Slice states are membership-weighted means of particle features, $\mathbf{z}_m = \sum_i S_{im}\mathbf{y}_i / \sum_i S_{im}$. Standard multi-head attention among the M slice states gives $\boldsymbol{Z}'$, and particles are updated by $\mathbf{y}'_i = \boldsymbol{Z}'^{\top}\mathbf{s}_i$, i.e. $\boldsymbol{Y}' = \boldsymbol{S}\boldsymbol{Z}'$ (Physics-Attention). Blocks are stacked with pre-norm residuals,

$$\hat{\boldsymbol{Y}}^{(\ell)} = \mathrm{PhysAttn}^{(\ell)}(\mathrm{LN}(\boldsymbol{Y}^{(\ell)}))+\boldsymbol{Y}^{(\ell)}, \qquad \boldsymbol{Y}^{(\ell+1)} = \mathrm{FFN}^{(\ell)}(\mathrm{LN}(\hat{\boldsymbol{Y}}^{(\ell)}))+\hat{\boldsymbol{Y}}^{(\ell)}, \qquad \ell = 0, \ldots, L_{\text{att}}-1, \tag{16}$$

and $\mathbf{h}^{\text{far}} = \mathrm{LN}(\boldsymbol{Y}^{(L_{\text{att}})})\boldsymbol{W}_{\text{out}}^{\top}$. We note that any $\boldsymbol{U}\boldsymbol{A}\boldsymbol{V}^{\top}$ operator (LinearNO, LRSA, a Perceiver block) can replace this block without changing the rest of P2T.

E.3 TRANSLATION INVARIANCE

Graph networks owe part of their generalization capabilities to translation invariance. In contrast, P2T's projector receives absolute-positions as input. We also tried to restore translation invariance by only giving block node features which discards relative-distance information that the edge features carry. However, we found that this drastically degraded performance and thus use absolution positions throughout the paper. A promising second option for restoring translation invariance is to use a two-dimensional rotary embedding (Heo et al., 2024) evaluated at continuous particle positions rather than integer grid indices.

F DATASETS

Wave tank. A 9.4 m tank, about 3.5 wavelengths long, with 0.40 m still-water depth, a piston wavemaker and a 1:11 beach. Wave height and period are sampled uniformly in the ranges $[0.05, 0.15]$ m and $[1, 2]$ s. The wave crosses the tank in about 5 s, so a 195-step rollout contains 3–5 wave cycles at the beach.

Dam break. A 4×4 m box in which a water column of random position, width and height collapses. The surge reaches the far wall after 0.71 ± 0.06 s, and the peak deflection event used for measuring system-scale recreation. Initial column x position, column width, and column heigh are sample uniformly in the ranges $[0, 1.5]$ m, $[0.5, 1.0]$ m, and $[1.0, 2.0]$ m. Particles displace up to several smoothing lengths per model step, so neighbor sets change at every step.

Table 6: **Benchmarks.** Steps per sample are at the model time step. Our two benchmarks are Lagrangian, feature test time-dependent and independent force scenarios, and larger than the public Lagrangian sets we are aware of.

Dataset	Physics	Target	Train / val / test	Nodes	Len.
ShapeNet-Car	RANS k–ε	steady	789 / 100 / –	32,186	–
CylinderFlow	incompressible NS, fixed mesh	next state	1,000 / 100 / 100	1,885	600
DAM 2D	WCSPH, violent free surface	next state	50 / 25 / 25	5,740	401
Wave tank (ours)	WCSPH, free surface waves	next state	400 / 40 / 40	19,618	200
Dam break (ours)	WCSPH, violent free surface	next state	400 / 40 / 40	21,202	200

Transfer sets. We also generate three wave tank variants used for finetuning experiments (Table 7). The short tank, the pre-training source, samples from the same wave height and period distribution as the full-length tank. Its main difference is its steeper beach. The plunging set extrapolates in wave height ($H \in [0.20, 0.30]$ m from $[0.05, 0.15]$ m), with a 6.0 m tank and varied beach slope to exaggerate plunging. The irregular set changes the forcing to a JONSWAP spectrum with parameters significant wave height, peak period, and peak factor sampled uniformly from $[0.08, 1.12]$ m, $[1.0, 1.85]$ s, and $[1, 6]$

For pretraining we train on the short tank because its dynamics are simplest and then finetune on the original (full) tank, irregular waves, and plunging waves.

Generation. All SPH datasets are generated with DualSPHysics v5.4 (Domínguez et al., 2022; Crespo et al., 2015), two-dimensional weakly compressible SPH at a particle spacing of $dp = 0.01$ m. Each simulation spans a 10 s window sampled every 0.05 s, and we release 195 steps per simulation, of which the first 5 supply the velocity history and 195 are predicted. Simulations are stored as $(T, N, 3)$ arrays of position and particle type (fluid, wall or piston). Table 7 summarizes the geometry and the sampled parameters of each set.

Table 7: Geometry and sampled parameters of the SPH datasets.

	Wave tank	Dam break	Wave (irregular)	Wave (short)	Wave (plunging)
Forcing	piston, regular	none	piston, JONSWAP	piston, regular	piston, regular
Train/val/test size	400/40/40	400/40/40	230/15/15	400/40/40	25/25/–
Particles	19,618	21,202	19,618	9,990	11,503
Sim length (steps)	200	200	200	200	200
Domain (m)	9.4 long, 0.40 deep	4.0 × 4.0 box	9.4 long, 0.40 deep	4.8 long, 0.40 deep	6.0 long, 0.40 deep
Beach slope	1:11	–	1:11	1:5.6	1:5.0-1:7.7
Parameters sampled	H, T, phase	column x, width, height	H_s, T_p, γ	H, T (as wave tank)	$\xi_0 \in [0.55, 1.15]$
Wave height (m)	0.100 ± 0.029	column 1.0–2.0 high	0.100 ± 0.011	0.099 ± 0.029	0.247 ± 0.030
Period (s)	1.53 ± 0.29	–	1.41 ± 0.24	1.53 ± 0.28	1.85 ± 0.17
Max speed (m/s)	1.43 ± 0.36	8.69 ± 2.03	0.85 ± 0.44	1.62 ± 0.32	3.36 ± 0.36

G TRAINING AND EVALUATION DETAILS

This appendix gives the training protocol, the model sizes and the metric definitions behind Tables 1 to 3.

Training. All next-state models predict the normalized acceleration of fluid particles from the current positions, a history of 5 velocities and per-particle types (water, wall, piston), use symplectic Euler integration, and are trained to predict acceleration with a one-step mean-squared loss. We add random-walk noise of standard deviation 8.0×10^{-4} m following (Sanchez-Gonzalez et al., 2020). We use AdamW with weight decay 0.01, a cosine learning-rate schedule from 10^{-4} to 10^{-6} over 60 epochs with a warmup epoch, and gradient-norm clipping at 1.0 on H100 GPUs.

Furthermore, on the dam break dataset, particles can move much further than our connectivity radius every step. To make the problem well-defined for graph networks and help training in general, we additionally add an input feature to each particle which is its distance from the nearest wall similar

to Sanchez-Gonzalez et al. (2020) scaled and clipped to 1.0 at 25 cm. The this threshold was chosen to match the 99th percentile one-step displacement.

Rollouts To allow for us to flexibly introduce new time-dependent forces, models are only trained to predict the motion of fluid particles. At rollout time the wall and piston particles are set to their ground-truth positions at every step. This allows us to update the motion of the wavemaker at each time step since every simulation has its own wavemaker amplitude and period. Without this, simulators could theoretically still learn to compute the period of the wavemaker from finite differences in the first frame. However, this would encode an assumption for regular waves that would prohibit generalization to new wave types in the transfer datasets.

Model hyperparameters For next-state prediction P2T, Transolver and GNS are parameter-matched at 2.9M parameters on the wave tank and 8.1M on the dam break. For next-state tasks, P2T allocates 75% of its parameters to message passing and 25% to Physics Attention (not including encoder and decoder parameters) for next-state tasks. For steady-state tasks which require more global computation, we instead use a $25/75$ allocation split. We additionally perform a sweep across allocation splits in Section H.1 and Table 8 gives the widths of the two components at each split. The predictor uses 15 message passing layers on wave tank and 10 on dam break. The projector uses 10 stacked Transolver blocks each with $M = 32$ slices and 8 heads.

We chose to keep UPT and Erwin close to their published configurations since they are generally much larger than our model, with hyperparameters chosen by performing a hyperparameter sweep on the validation set. Furthermore, UPT normally advances several steps in latent space between encoding and decoding, which requires the overall shape of the fluid to persist across those steps. On the free-surface flows here we found encoding and decoding at every step to be more accurate, so all UPT results use a latent rollout step size of one.

Table 8: **Model sizes.** Hidden widths of the message-passing predictor (H_g) and the Transolver projector (H_t) at each allocation α.

	Wave tank		Dam break	
Model	H_g / H_t	Params	H_g / H_t	Params
Transolver (0%)	– / 232	2.87M	– / 384	7.82M
P2T (25%)	64 / 200	2.89M	128 / 336	8.07M
P2T (50%)	88 / 160	2.78M	184 / 272	8.13M
P2T (75%)	112 / 112	2.92M	224 / 192	8.10M
GNS (100%)	128 / –	2.88M	256 / –	7.84M

Noise injection. Several methods improve rollout stability at training time, through data augmentation (Sanchez-Gonzalez et al., 2020) or the pushforward trick (Brandstetter et al., 2022). We use the noise injection of GNS for every model, since it prevents overfitting even on the small transfer sets (Section 5.1). Most baselines do not use it by default, so we trained each with and without it and found it helpful for every model.

Metrics. All metrics are computed on held-out rollouts of 190 steps started from the ground-truth initial steps. Reported numbers on wave tank and dam break are mean and standard deviation over 3 seeds. While we follow Rota Roselli et al. (2019) and Lobovskỳ et al. (2014) for validation, which use a few real experimental runs over hundreds of wave cycles, we adapt them for evaluating hundreds of simulations over 3–5 cycles each.

Rollout position RMSE. At rollout step n, $\mathrm{RMSE}_n = \left(\frac{1}{N_f}\sum_{i\in\text{fluid}} \|x_i^n - \hat{x}_i^n\|^2\right)^{1/2}$ in mm, with $\hat{x}$ the model rollout and N_f the number of fluid particles. A rollout is marked as diverged (div.) once its RMSE exceeds 2,000 mm.

Free-surface error (wave tank). At every step we extract the free-surface elevation $\eta(x)$ of the model and of the reference in 2 cm columns from $x = 0.3$ m to the still-water shoreline at $x = 6.98$ m. The error is the spatial RMS of $\eta - \hat{\eta}$ over the columns, averaged over the rollout and normalized by the

wave height measured at the beach toe in the reference run,

$$e_{\mathrm{rms}} = \frac{1}{H_{\mathrm{toe}}} \left\langle \left(\overline{(\eta - \hat{\eta})^2} \right)^{1/2} \right\rangle_t, \tag{17}$$

so that 0 is perfect and 1 is an error of one wave height. Figure 3 shows the time series before averaging, and a rollout has lost the wave when it exceeds 0.5 for three consecutive steps.

Peak deflection error (dam break). We track the run-up against the far wall as $h_{10}(t)$, the mean height of the highest 10% of fluid particles, in the model and in the reference, and report the relative error of its peak, $100 \times \left| \max_t h_{10} / \max_t \hat{h}_{10} - 1 \right|$ in percent.

Density deviation. We evaluate how well rollouts match DualSPHysics's density deviations by using it the two-dimensional Wendland C2 kernel with $q = r/h$ and support $2h$,

$$W(r) = \frac{7}{4\pi h^2} \Big(1 - \frac{q}{2}\Big)^4 (2q+1) \ \text{ for } q < 2, \qquad h = 0.0170\,\mathrm{m}\ \text{(wave tank)},\ 0.0141\,\mathrm{m}\ \text{(dam break)}, \tag{18}$$

on every fluid particle i, summing over the fluid particles $\mathcal{W}$ and the boundary particles $\mathcal{B}$ within its support. We then normalize by the value a particle takes in a perfect lattice of spacing dp,

$$\hat{\rho}_i = \frac{dp^2}{\rho_{\mathrm{lat}}} \Big[W(0) + \sum_{j \in \mathcal{W} \setminus \{i\}} W(r_{ij}) + \sum_{b \in \mathcal{B}} W(r_{ib}) \Big], \qquad \rho_{\mathrm{lat}} = dp^2 \sum_{(m_1, m_2) \in \mathbb{Z}^2} W\big(dp \sqrt{m_1^2 + m_2^2}\big), \tag{19}$$

so that a perfectly packed particle has $\hat{\rho}_i = 1$. Free-surface particles have missing neighbors by construction, so we restrict to the interior set $\mathcal{I}(t)$ of fluid particles with at least 8 fluid neighbors within $2h$, and report the RMS deviation from 1 averaged over a rollout of T steps,

$$\delta\rho(t) = \Big(\frac{1}{|\mathcal{I}(t)|} \sum_{i \in \mathcal{I}(t)} \big(\hat{\rho}_i(t) - 1\big)^2 \Big)^{1/2}, \qquad \overline{\delta\rho} = \frac{1}{T} \sum_{n=1}^{T} \delta\rho(t_n). \tag{20}$$

Weakly compressible SPH simulators naturally exhibit a small amount of density deviation so Table 2 reports each model's deviation next to the ground truth's, and closer is better in either direction. A larger value means particles have bunched or spread, and a smaller one means the model has over-regularized them. In general, we observe the former.

H Full results

H.1 Full rollout results including P2T skew

Tables 9 and 10 give Table 1 with standard deviations over simulations and seeds, and add the two other allocations of P2T's parameter budget (Table 8).

Table 9: Rollout position RMSE (mm) on the wave tank test set for all baselines and the three P2T allocations. Global skew, balanced and local skew put 25%, 50% and 75% of the parameters in the predictor.

Model	n=1	20	50	100	190
GNS	0.2121 ± 0.0289	7.561 ± 2.678	35.03 ± 13.12	107.5 ± 32.9	188.9 ± 87.8
Transolver	0.3445 ± 0.0572	9.622 ± 2.563	42.60 ± 21.00	70.73 ± 14.88	131.7 ± 12.9
UPT	0.3766 ± 0.0812	10.98 ± 3.42	20.69 ± 5.83	51.68 ± 12.75	115.7 ± 41.1
Erwin	0.3214 ± 0.0082	18.43 ± 3.98	36.01 ± 5.30	71.22 ± 28.60	172.0 ± 65.2
P2T (global skew)	0.2174 ± 0.0114	5.589 ± 1.573	17.77 ± 4.43	51.74 ± 16.04	135.0 ± 46.2
P2T (balanced)	0.2033 ± 0.0135	4.961 ± 1.257	22.98 ± 4.74	40.74 ± 11.06	121.1 ± 39.0
P2T (local skew)	**0.2021 ± 0.0106**	**4.916 ± 0.786**	**15.44 ± 3.91**	**34.41 ± 8.05**	**81.26 ± 37.7**

H.2 Comparison on public benchmarks

Table 11 reports P2T, GNS and Transolver on two public time-dependent benchmarks, CylinderFlow (Pfaff et al., 2021) and DAM 2D from LagrangeBench (Toshev et al., 2023). Both domains are a few

Table 10: Rollout position RMSE (mm) on the dam break test set. "div." marks diverged rollouts).

Model	n=1	20	50	100	190
GNS	**2.467 ± 0.145**	**222.5 ± 22.11**	822.2 ± 6.888	989.4 ± 29.94	1108 ± 55.72
Transolver	4.843 ± 2.195	317.7 ± 214.5	991.0 ± 86.24	1053.2 ± 196.3	1032.7 ± 175.7
UPT	30.38 ± 0.54	859.9 ± 106	1697 ± 249	div.	div.
Erwin	5.285 ± 2.447	1364 ± 452	div.	div.	div.
P2T (global skew)	2.960 ± 0.217	247.8 ± 18.2	833.0 ± 33.7	980.5 ± 39.0	1040 ± 72.11
P2T (balanced)	2.983 ± 0.339	325.2 ± 29.8	826.3 ± 51.8	880.5 ± 41.3	986.5 ± 64.3
P2T (local skew)	2.949 ± 0.037	260.2 ± 26.1	**790.3 ± 15.9**	**852.2 ± 19.5**	**964.1 ± 54.1**

tens of connectivity radii across, so most of the domain is already within the predictor's reach and there is little far field for the projector to add. GNS is the more accurate model on CylinderFlow, the smallest domain, and the two perform similarly on DAM 2D. Figure 7 plots P2T's error relative to GNS against domain size for the four time-dependent benchmarks.

Table 11: Rollout RMSE after $\boldsymbol{n}$ autoregressive steps on two public benchmarks. CylinderFlow is mesh-based, so the error is normalized velocity RMSE ($\times 10^{-3}$). DAM 2D is particle-based, so the error is position RMSE (mm). Best in **bold**, second best underlined.

	CylinderFlow					**DAM 2D (LagrangeBench)**				
Model	n=1	20	50	100	200	n=1	20	50	100	200
GNS	**2.19**	**3.14**	**4.75**	**7.67**	**16.10**	**0.1422**	12.59	**54.53**	141.44	252.00
Transolver	23.28	84.29	121.57	150.07	174.28	0.2807	20.86	118.92	284.11	556.97
P2T (ours)	3.31	5.71	9.36	18.75	40.35	0.1857	**10.30**	54.96	**138.06**	**244.89**

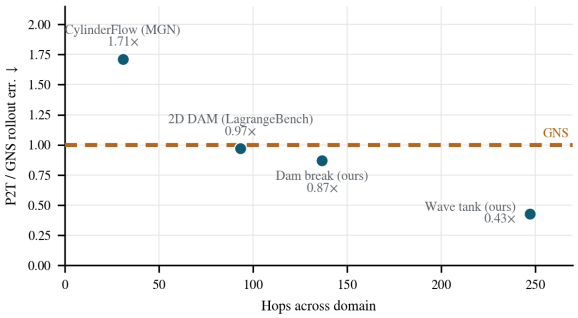

Figure 7: P2T's final-step rollout RMSE relative to GNS against average simulation size in connectivity radii. P2T's advantage over GNS scales with how large the simulation is.

H.3 WAVE TANK ROLLOUTS

Figure 8 extends Figure 2 to $t = 10\,\mathrm{s}$, the end of the rollout, and Figure 9 shows the same rollouts close up at the leading crest and at the piston.

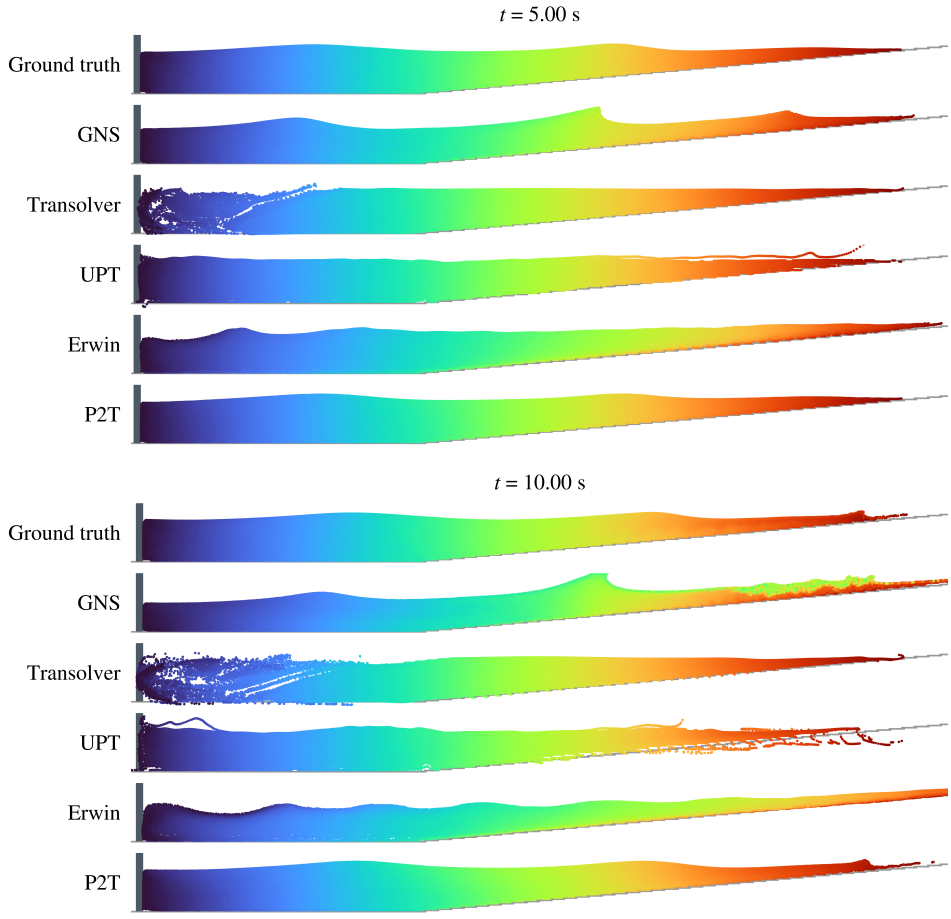

Figure 8: Wave-tank rollouts at $t = 5$ and $10\,\mathrm{s}$ for every model, particles colored by initial position.

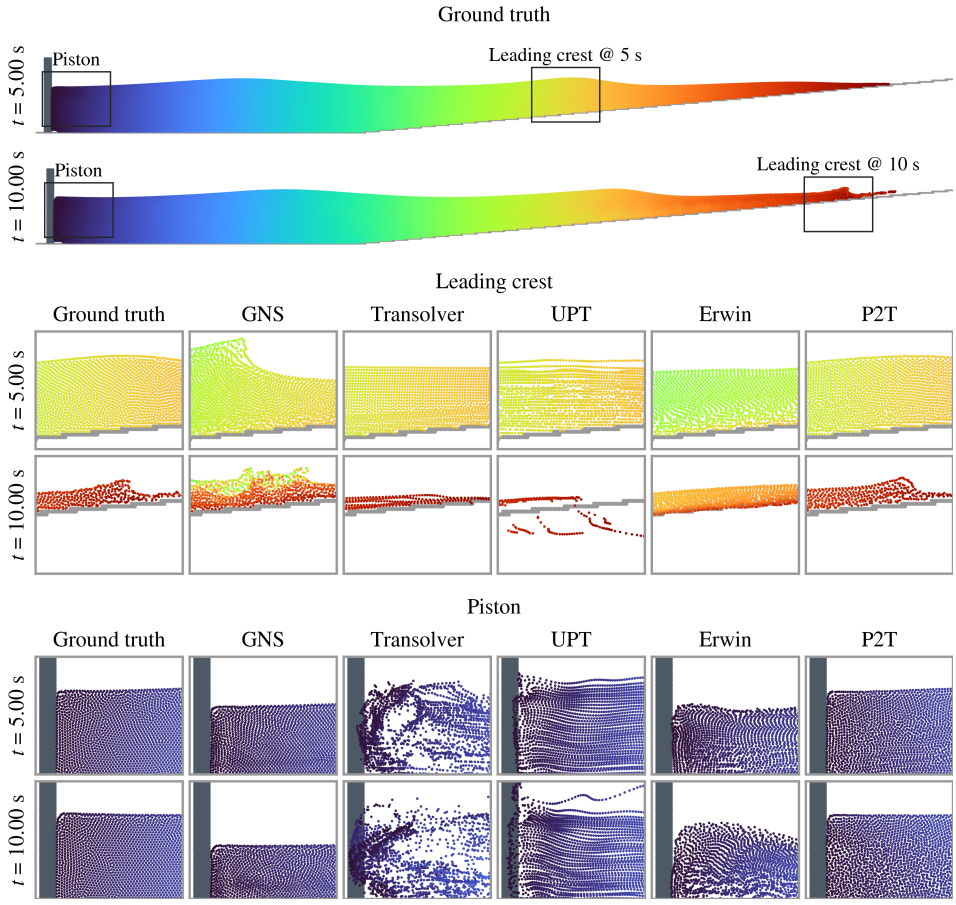

Figure 9: Close-ups of Figure 8 at the leading crest and at the piston (indicated by the boxes in the top rows). GNS's crest breaks early, Transolver's and UPT's particles lose their spacing, Erwin's crest flattens, and P2T keeps both the crest and the spacing.

H.4 Comparison to GIOROM

GIOROM (Viswanath et al., 2026) evolves a sampled subset of the particles with a 2-layer graph-network encoder, a GNOT processor and a 2-layer graph-network decoder. They reconstruct the rest of the particles using learned interpolation from the subset. We trained it on both datasets, extending its grid to our larger spatial domain while keeping its original spacing. Table 12 gives rollout errors and Figures 10 and 11 the rollouts. GIOROM is the most accurate model at one step on the wave tank, but its error grows fastest of all models and its simulation quickly becomes unrealistic. This is most likely due to compounding error at interpolation boundaries seen in its interaction maps (Figure 22, Section I).

Table 12: Rollout position RMSE (mm) of GIOROM and P2T after n autoregressive steps, mean over the test set. P2T is averaged over 3 seeds, GIOROM over a single run. Best in **bold**, "div." marks diverged rollouts.

	Wave tank					**Dam break**				
Model	n=1	20	50	100	190	n=1	20	50	100	190
GIOROM	**0.190**	8.97	27.6	119.5	726.7	5.71	311	1739	div.	div.
P2T (ours)	0.202	**4.92**	**15.4**	**34.4**	**81.3**	**2.95**	**260**	**790**	**852**	**964**

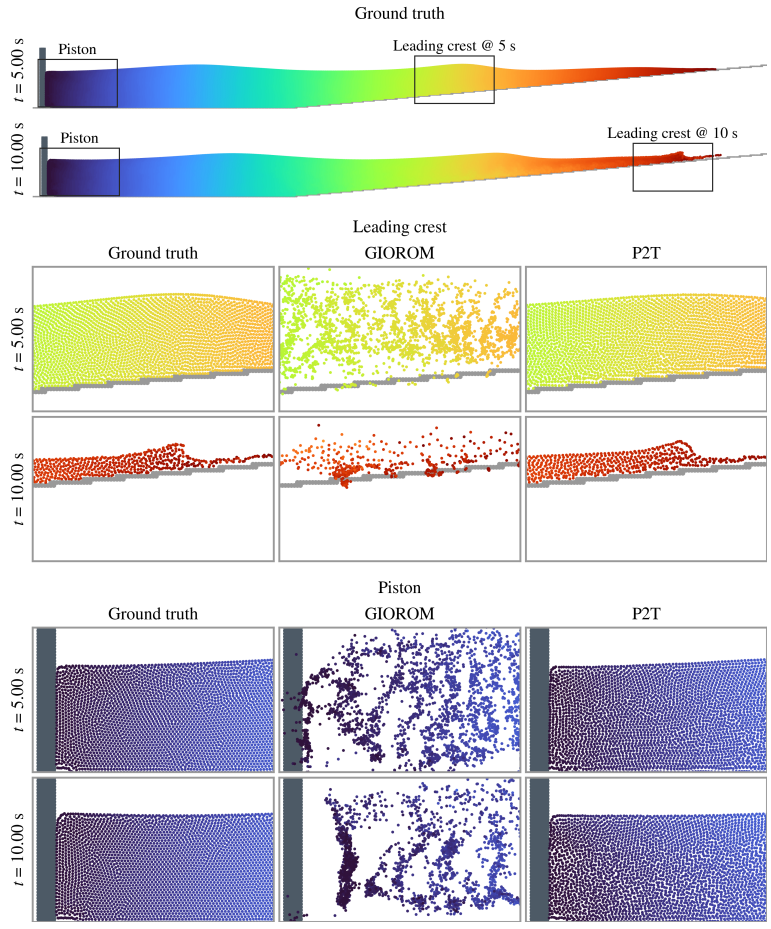

Figure 10: Wave-tank rollouts of GIOROM and P2T at $t = 5$ and $10\,\mathrm{s}$, full tank (top) and close-ups at the leading crest and the piston. GIOROM's particles scatter periodically following latent grid spacing.

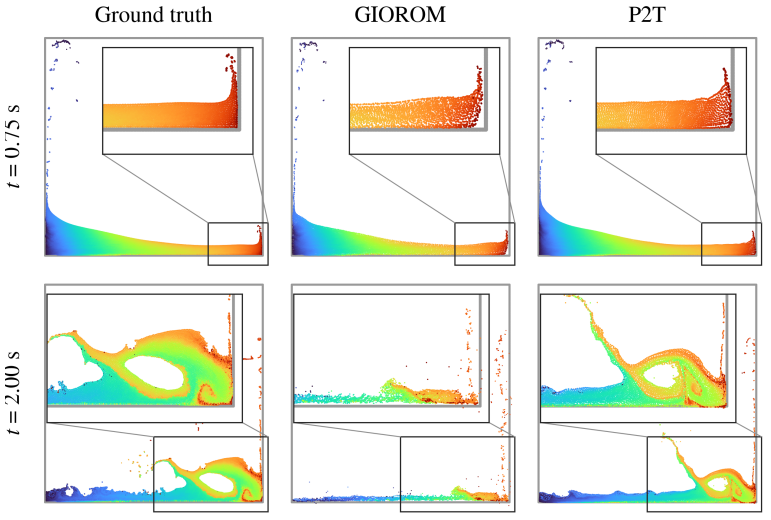

Figure 11: Dam-break rollouts of GIOROM and P2T at impact ($t = 0.75\,\mathrm{s}$) and after the surge falls back ($2.00\,\mathrm{s}$), with insets at the far wall. GIOROM's surge thins and scatters at the wall and fails to produce an air cavity.

H.5 DAM BREAK ROLLOUTS ACROSS MODELS

Figure 12 extends Figure 4 to four times for every model. Prediction starts after the first five steps, which supply the velocity history.

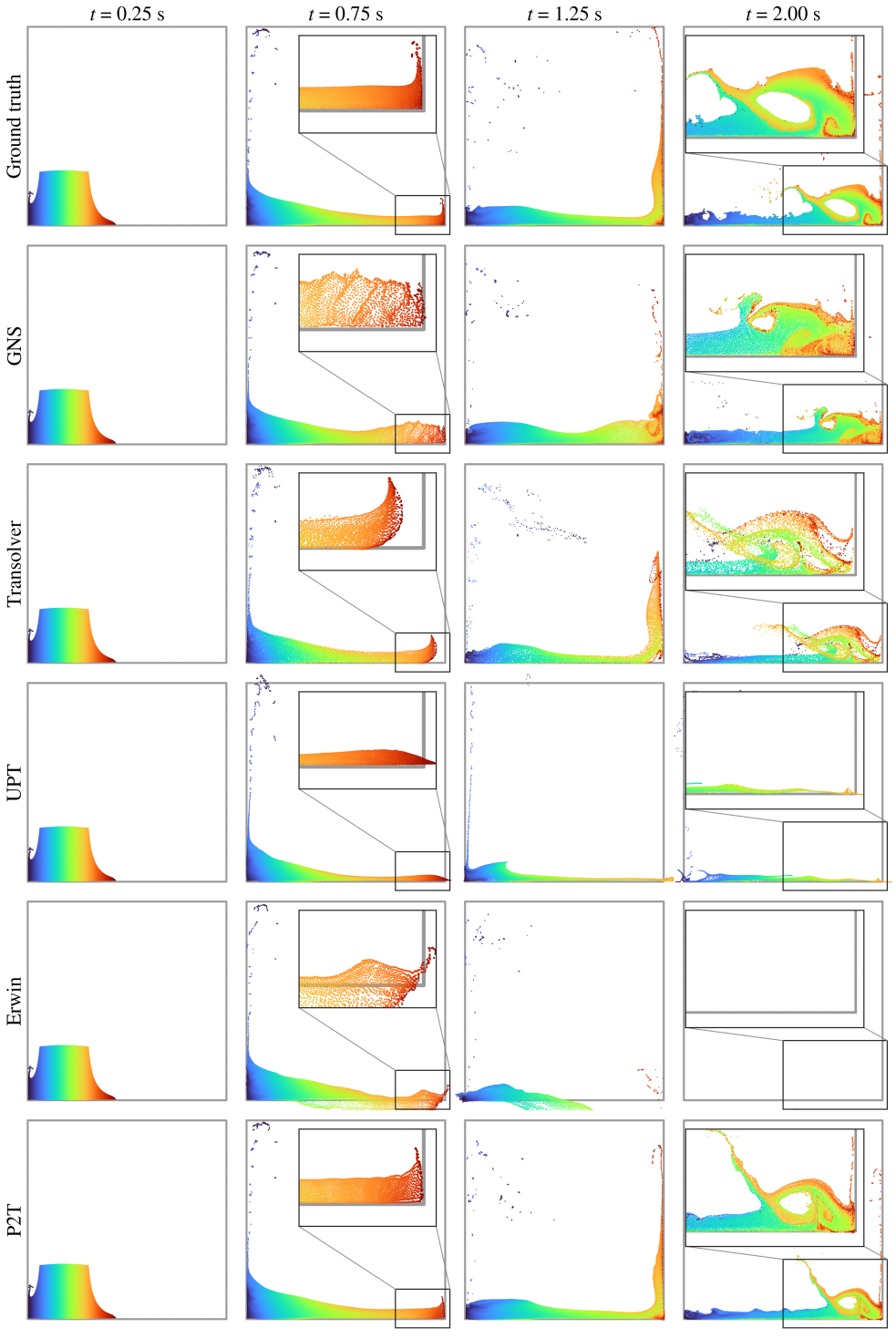

Figure 12: Dam-break rollouts at four times, particles colored by initial x-position: the collapsing column ($t = 0.25$ s), impact on the far wall (0.75 s), run-up (1.25 s) and the fall-back that traps an air cavity (2.00 s). P2T is the only model that reproduces all four stages.

H.6 Error maps on ShapeNet-Car

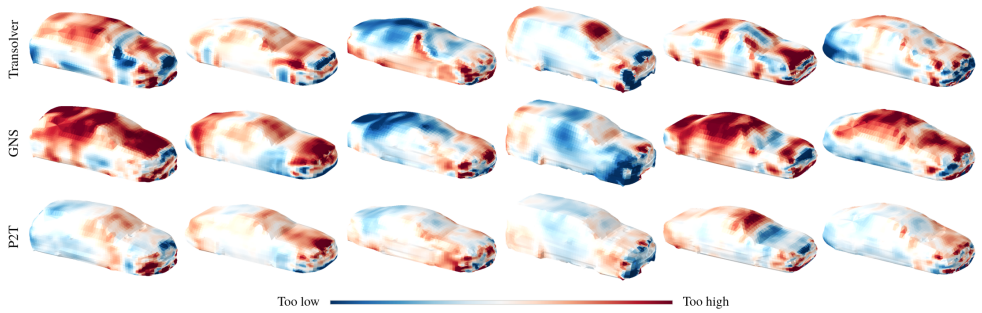

Figure 13: Surface-pressure error on ShapeNet-Car for six test cars, for Transolver, P2T and GNS. GNS's errors are large and smooth over whole panels, Transolver's are smaller but high-frequency, and P2T's are the smallest.

H.7 Runtime scaling

Figure 14 measures inference time per step and peak GPU memory as the wave tank is lengthened from $1\times$ to $16\times$ at fixed particle density, from 20k to 584k particles. Models scale linearly in the number of particles, P2T costs no more than GNS at any size, and DualSPHysics on the same GPU is $18\times$ slower than P2T at $1\times$ and about $2\times$ at $16\times$. Erwin runs out of memory at $16\times$.

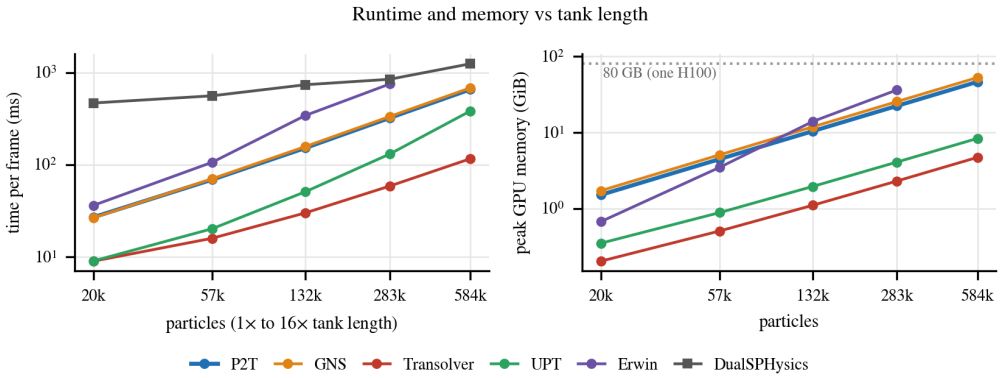

Figure 14: Time per step (left) and peak GPU memory (right) on one H100 for wave tanks of $1\times$ to $16\times$ the original length at fixed particle density. Erwin is out of memory at $16\times$.

I ADDITIONAL INTERACTION MAPS

Figures 15 and 16 show the step of Figure 5 with linear and logarithmic color scales, and Figures 17 to 22 show, for each model, several target particles at different positions and depths. Transolver's map is nearly identical across targets, while P2T's follows the target.

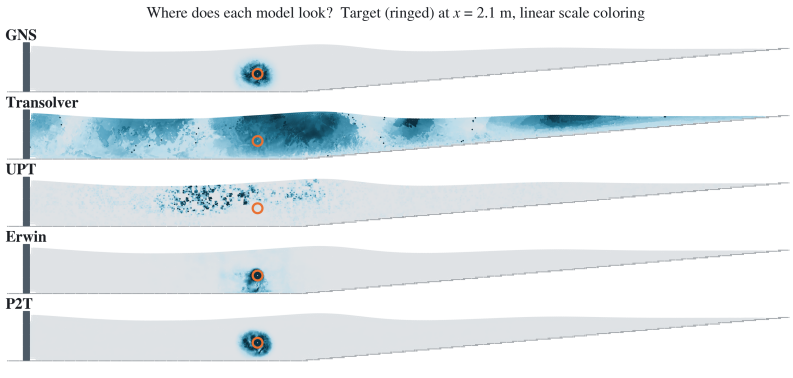

Figure 15: As Figure 5, linear color scale.

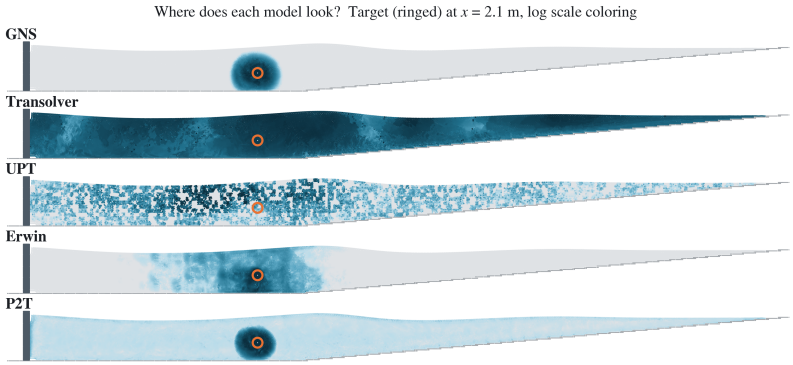

Figure 16: As Figure 5, logarithmic color scale.



Figure 17: GNS: interaction maps (rank coloring) for several target particles.

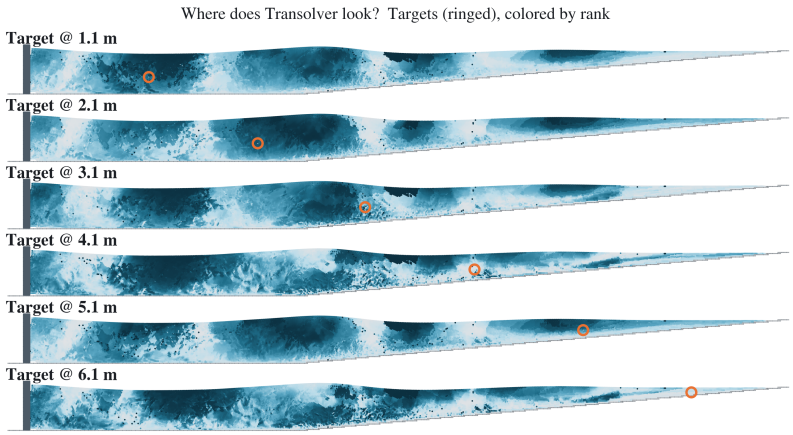

Figure 18: Transolver: interaction maps (rank coloring) for several target particles.

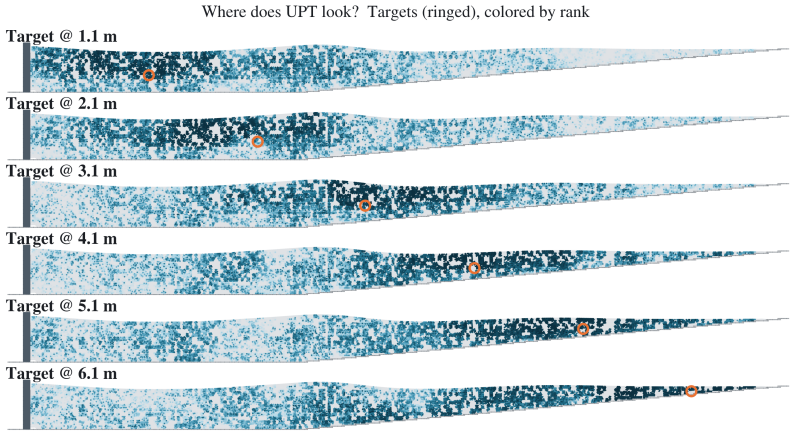

Figure 19: UPT: interaction maps (rank coloring) for several target particles.

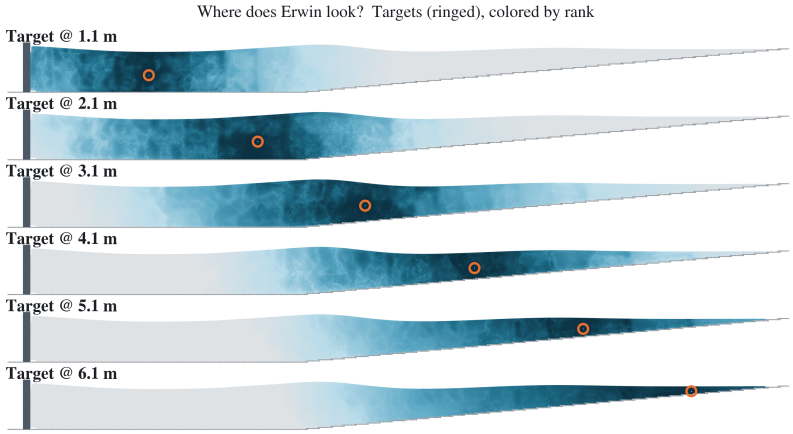

Figure 20: Erwin: interaction maps (rank coloring) for several target particles.

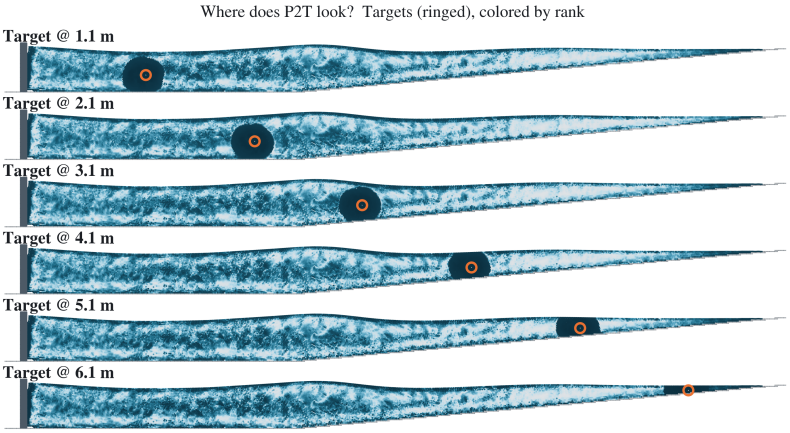

Figure 21: P2T: interaction maps (rank coloring) for several target particles.

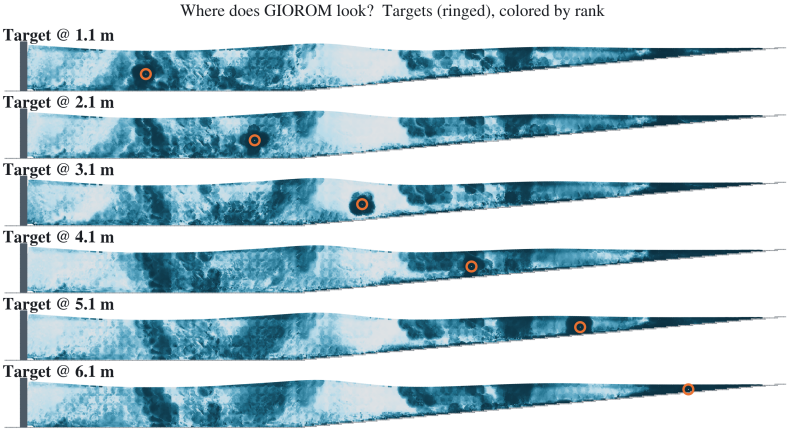

Figure 22: GIOROM: interaction maps (rank coloring) for several target particles.