Neural SPH: Improved Neural Modeling of Lagrangian Fluid Dynamics

Artur P. Toshev [1] **Jonas A. Erbesdobler** [1] **Nikolaus A. Adams** [1 2] **Johannes Brandstetter** [3 4]

Abstract

Smoothed particle hydrodynamics (SPH) is omnipresent in modern engineering and scientific disciplines. SPH is a class of Lagrangian schemes that discretize fluid dynamics via finite material points that are tracked through the evolving velocity field. Due to the particle-like nature of the simulation, graph neural networks (GNNs) have emerged as appealing and successful surrogates. However, the practical utility of such GNN-based simulators relies on their ability to faithfully model physics, providing accurate and stable predictions over long time horizons – which is a notoriously hard problem. In this work, we identify particle clustering originating from tensile instabilities as one of the primary pitfalls. Based on these insights, we enhance both training and rollout inference of state-of-the-art GNN-based simulators with varying components from standard SPH solvers, including pressure, viscous, and external force components. All Neural SPH-enhanced simulators achieve better performance than the baseline GNNs, often by orders of magnitude in terms of rollout error, allowing for significantly longer rollouts and significantly better physics modeling. Code available under https://github.com/tumaer/neuralsph.

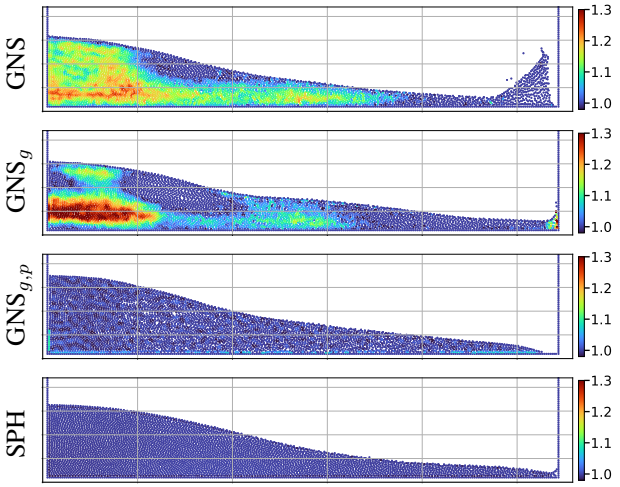

Figure 1. Neural SPH improves Lagrangian fluid dynamics, showcased by physics modeling of the 2D dam break example after 80 rollout steps. Different models exhibit different physics behaviors. From top to bottom: GNS (Sanchez-Gonzalez et al., 2020), GNS with corrected force only (GNS$_g$), full SPH enhanced GNS (GNS$_{g,p}$), and the ground truth SPH simulation. The colors correspond to the density deviation from the reference density; the system is considered physical within 0.98-1.02.

1. Introduction

In the sciences, considerable efforts have led to the development of highly complex mathematical models of our world, with many naturally formulated as partial differential equations (PDEs). Over the past years, deep neural network-based PDE surrogates have gained significant momentum as a more computationally efficient solution methodology (Thuerey et al., 2021; Brunton & Kutz, 2023), transforming amongst others computational fluid dynamics (Guo et al., 2016; Kochkov et al., 2021; Li et al., 2021; Gupta & Brandstetter, 2022; Alkin et al., 2024), weather forecasting (Rasp & Thuerey, 2021; Weyn et al., 2020; Sønderby et al., 2020; Pathak et al., 2022; Lam et al., 2022; Nguyen et al., 2023; Bodnar et al., 2024), and molecular modeling (Gasteiger et al., 2021; Batzner et al., 2022; Batatia et al., 2022; Zeni et al., 2023; Merchant et al., 2023).

In computational fluid dynamics (CFD), we broadly categorize numerical simulation methods into two distinct families: particle-based and grid-based, better known as Lagrangian and Eulerian discretization schemes. In Eulerian schemes, space is discretized, i.e., fixed finite nodes or control volumes lead to grid-based or mesh-based models. In Lagrangian schemes, the discretization happens on finite material points, commonly known as particles, which

[1]Chair of Aerodynamics and Fluid Mechanics, School of Engineering and Design, Technical University of Munich, Garching, Germany [2]Munich Institute of Integrated Materials, Energy and Process Engineering, Technical University of Munich, Germany [3]ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Johannes Kepler University, Linz, Austria [4]NXAI GmbH, Austria. Correspondence to: Artur P. Toshev <artur.toshev@tum.de>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

dynamically move with the local deformation of the continuum. One of the most prominent Lagrangian discretization schemes is smoothed particle hydrodynamics (SPH), originally proposed by Lucy (1977) and Gingold & Monaghan (1977) for applications in astrophysics. In contrast to grid- and mesh-based approaches, SPH approximates the field properties using radial kernel interpolations over adjacent particles. The strength of the SPH method is that it does not require connectivity constraints, e.g., meshes, which is particularly useful for simulating systems with large deformations. Since its foundation, SPH has been greatly extended and is the preferred method to simulate problems with (a) free surfaces (Marrone et al., 2011; Violeau & Rogers, 2016), (b) complex boundaries (Adami et al., 2012), (c) multi-phase flows (Hu & Adams, 2007), and (d) fluid-structure interactions (Antoci et al., 2007).

In deep learning, graph neural networks (GNNs) (Scarselli et al., 2008; Kipf & Welling, 2017) are an obvious fit to model particle-based dynamics. Often, predicted accelerations at the nodes are numerically integrated to model the time evolution of the particles or the mesh, i.e., dynamics are updated in a hybrid neural-numerical fashion (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2020; Mayr et al., 2023). Most recent applications of GNN-based simulators involve Lagrangian fluid simulations (Toshev et al., 2023a; 2024a; Winchenbach & Thuerey, 2024). One downside of these simulators is the risk of non-physical instabilities during rollout, which affects the neural and numerical components.

It is known that already standard SPH schemes exhibit tensile instability, i.e., numerical errors leading to particle clumping and void regions when negative pressure occurs within what should be an incompressible fluid (Price, 2012). This has led to the development of improved SPH schemes explicitly targeting regularity of particle distribution (Adami et al., 2013; Zhang et al., 2017b). A review of SPH literature indicates that even methods seeking to improve other properties, like reducing artificial dissipation (Zhang et al., 2017a) or handling violent water flows (Marrone et al., 2011), may also improve the particle distribution.

In this work, we present a large-scale analysis of Lagrangian physical modeling capabilities of various GNN-based simulators, i.e., a non-equivariant and an equivariant one. We identify a shared pitfall, i.e., particle clustering effects that are similar to those known from SPH schemes. Particle clustering in GNN-based simulators limits stable rollouts and accurate physics modeling. Based on these insights, we draw inspiration from numerical SPH solvers and enhance both training and inference of state-of-the-art GNN-based simulators with varying components from standard SPH solvers, including (i) pressure, (ii) viscous, and (iii) external force components – all implemented in JAX (Bradbury et al., 2018). Methodologically, our main contributions are two: (a) novel external force treatment during training, and (b) an additional SPH relaxation routine during inference.

We demonstrate the efficacy of Neural SPH-enhanced Lagrangian simulators by achieving better performance on seven diverse 2D and 3D Lagrangian datasets – sometimes by orders of magnitude in terms of rollout error – than the baseline GNN, allowing for significantly better physical modeling capabilities. We note that the introduced Neural SPH techniques may apply to a wide range of physics scenarios beyond GNNs and SPH. Our source code is available at https://github.com/tumaer/neuralsph.

2. Simulating Lagrangian dynamics

Smoothed particle hydrodynamics. Smoothed particle hydrodynamics (SPH) approximates the incompressible Navier-Stokes equations (NSE) by the so-called weakly compressible NSE. This is necessary because the density of the fluid is defined by radial kernel summation $\rho_i = \sum_j m_j W(r_{ij}|h)$, where m_j represents the mass of the adjacent particles j, and W the radial interpolation kernel with smoothing length h that operates on the scalar distance r_{ij}. This summation may violate strict incompressibility. However, the weak compressibility assumption typically allows for up to $\sim 1\%$ density deviation (Monaghan, 2005). This $\sim 1\%$ is also enforced for the weakly compressible SPH method, while evolving density and momentum:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\rho) = -\rho\left(\nabla \cdot \mathbf{u}\right), \tag{1}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{u}) = \underbrace{-\frac{1}{\rho}\nabla p}_{\text{pressure}} + \underbrace{\frac{\nu}{V_{ref}L_{ref}}\nabla^2\mathbf{u}}_{\text{viscosity}} + \underbrace{\mathbf{g}}_{\text{ext. force}}. \tag{2}$$

Herein, ρ is the density, $\mathbf{u}$ the velocity vector, p the pressure, $\mathbf{g}$ the external force, ν the viscosity, and U_{ref}, L_{ref} the reference velocity and length scale. Without loss of generality, we consider $U_{ref} = 1$, $L_{ref} = 1$. We note that either density summation with kernel averaging, or density evolution (Eq. (1)) is used to compute the density, and as we explain later, the former is the preferred and the latter the more general approach. To evolve the system in time, the above equation(s) are integrated in time by, e.g., semi-implicit Euler (see Appendix F). However, solving these equations with standard SPH methods may still produce artifacts, most notably when particle clumping exceeds the 1% density-fluctuation requirement (Adami et al., 2013).

SPH particle redistribution. The term responsible for a homogeneous particle distribution in the SPH method is the pressure gradient term $\frac{1}{\rho}\nabla p$ in the momentum equation Eq. (2). In weakly compressible SPH, the pressure is computed from density through the equation of state

$$p(\rho) = p_{ref}\left(\frac{\rho}{\rho_{ref}} - 1\right). \tag{3}$$

Thus, for a reliable approximation of the density ρ, the pressure term ensures a repulsive force of scale p_{ref} whenever the density exceeds the given reference value ρ_{ref}, where typically $\rho_{ref} = 1$. However, the pressure term is not necessarily sufficient for producing a good particle distribution, as we can see in the bottom part of Fig. 9 in Toshev et al. (2024a). For this reason, more advanced SPH schemes have been developed, distinguishing between the physical velocity field and the velocity by which particles are shifted (Adami et al., 2013; Zhang et al., 2017b). These schemes are related to Arbitrary Lagrangian-Eulerian methods (Hirt et al., 1974) instead of being fully Lagrangian.

Challenges of density computation at free surfaces. Accurately computing the density at free surfaces is a difficult task for SPH methods. In the standard SPH formulation, the density at each particle is calculated by a kernel-weighted summation of the mass of adjacent particles (Gingold & Monaghan, 1977). However, particles at free surfaces have low density when using density summation, which leads to incorrect pressure values (Monaghan, 1994). The low-density inconsistency can be corrected for by globally and locally conservative least-squares interpolation (Dilts, 2000), adaptive kernel estimation procedure (Sigalotti et al., 2006), or by initializing the simulation by first evolving particles with a heavily damped version of the momentum conservation (Becker & Teschner, 2007). However, most SPH methods for free surface flows resort to the continuity equation to represent the rate of change in density (Monaghan, 1994; Bonet & Lok, 1999). In this density evolution formulation, density derivatives are integrated over time (see Eq. (1)). On top of the density evolution, density filters, such as periodic re-initialization, are applied (Gomez-Gesteira et al., 2010; Colagrossi & Landrini, 2003; Shepard, 1968).

GNN-based simulators. The formulation of the learning problem is based on LagrangeBench (Toshev et al., 2024a). We look at the task of autoregressive acceleration prediction of a Lagrangian particle system, which we then integrate twice using semi-implicit Euler integration to evolve the system over time (see Appendix F). The datasets consist of particle types per particle and particle coordinates $\mathbf{P}^{t_k}$ over $k \in (0, K)$ steps, where each frame $\mathbf{P}^t$ is made up of $n \in (1, N)$ particles $\mathbf{p}_n^t \in \mathbb{R}^d$ in dimension d. The inputs to the learned surrogate are state vectors $\mathbf{X}^{t_{k-H}:t_k}$, with history size H, each of which contains the past velocities $\mathbf{U}_k = [\mathbf{u}_{k,1}, ..., \mathbf{u}_{k,N}]$ inferred using the finite difference approximation of past coordinates, as well as optional features like external force vector $\mathbf{g}$, e.g., gravity.

We use the default configuration files from LagrangeBench for training, including random walk noise (Pfaff et al., 2020) and the pushforward trick (Brandstetter et al., 2022b). These default configurations provide the baseline models, on top of which we add our methods.

Pathological particle clustering during long rollouts for GNN-based simulators. The entering point to our analysis is the realization that simulated rollouts of a learned Graph Network-based Simulators (GNS) (Sanchez-Gonzalez et al., 2020) severely violate the 1% compressibility requirement present in weakly compressible SPH methods – see top part of Fig. 1. This figure shows compression of as much as $1.4 \cdot \rho_{ref}$ in the left part, which is not only unphysical regarding the density itself but might also lead to unphysical dynamics in the sense of periodic compressions and expansions later in the rollout, see Section 4. The violation – although much worse – resembles pressure inaccuracies in classical numerical SPH solvers.

To qualitatively understand clustering, in Fig. 2, we plot the histogram of the per-particle number of neighbors corresponding to the left graphic of the 2D lid-driven cavity from Fig. 5, which also has regions with high particle density. In Fig. 2, we see a pronounced increase in the number of particles with 8-10 neighbors, indicating clustering artifacts.

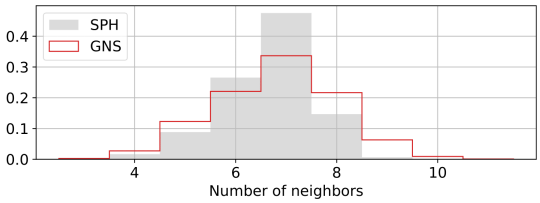

Figure 2. Number of neighbors mismatch due to particle clustering. Histogram of the number of neighbors of the 2D lid-driven cavity experiment after 400 rollout steps (average over all test rollouts).

The problem of external forces. We observe that in roughly 8 out of 25 dam break test trajectories at step 80, the front of the wave spreads out as if a virtual wall exists way in front of the actual wall – see Fig. 1 and Appendix A. Such behavior has been discussed in literature (Klimesch et al., 2022), and the current consensus is that the GNN-based simulators learn to infer the dynamics from velocity correlations. Thus, when the velocity reaches a given threshold, it has learned to model the presence of a wall. In the following, we demonstrate that by forcing the network to predict a target acceleration that excludes the external force part, the overall dynamics become more physical, and significantly fewer artifacts occur.

3. Neural SPH

In this section, we introduce Neural SPH, which improves both training and rollout inference of temporally coarsened GNN-based simulators. Neural SPH comprises a routine to correct for induced modeling errors due to external forces, and inference-time refinement steps of the system state based on SPH relaxation methods.

Correction of external forces. In the learning problem formulation by Toshev et al. (2024a), the GNN-based simulators receive as node inputs a time sequence of the H most recent historic velocities stacked to $\mathbf{u}_{k-H:k} = [\mathbf{u}_{k-H}, ...\mathbf{u}_k]$ and an optional external force vector. Consequently, the GNN-based simulators are confronted with the underlying instantaneous force and not the effective force, i.e., the force that acts on the particles upon temporal coarsening. We make two observations:

1. The impact of the external force $\mathbf{g}$ is already included in the dynamics given by the past velocities $\mathbf{u}_{k-H:k}$. Thus, providing a constant force vector, i.e., gravitational force, as model input might be necessary when training equivariant models, but as Sanchez-Gonzalez et al. (2020) show in their appendix C2, the GNS model does not improve when external force information is added. However, in the general case of systems with spatially varying forces, having force vectors as inputs is crucial. An example is the reverse Poiseuille flow, which has a positive force in x direction when $y > 1$ and a negative force when $y < 1$ (see Appendix D).

2. By predicting the full acceleration $\mathbf{a}$, the GNN-based simulators are forced to model gravity implicitly. One might argue that gravity is just a bias term in the last decoder layer, and thus, a GNN-based simulator should be able to model gravitational effects quite easily. However, we observe that for a GNS model trained on dam break (see Fig. 1 top part), the bias term in the last layer is more than an order of magnitude smaller than the respective gravitational acceleration.

Especially the latter observation indicates that GNN-based simulators indeed mainly learn velocity correlations as suggested by Klimesch et al. (2022). Referring to the structure of Eq. (2), and motivated by operator splitting, we suggest to bracket terms on the right-hand side of this equation as $[...] + \mathbf{g}$. If considering temporal coarsening of GNN-based simulators over M SPH steps, and given that the dataset is generated by running an SPH simulation with a constant time step Δt_{SPH}, the steps over which the GNN-based simulator integrates are $M\Delta t_{SPH}$. In the case of a constant force $\mathbf{g}$, this leads to an effective external force after M SPH steps of $\mathbf{g}_M^{FD} = (M\Delta t_{SPH})^2\mathbf{g}$, as by double integration of acceleration to positions with a finite difference time step $\Delta t^{FD} = 1$, see Appendix F. Thus, when removing the accumulated external force from the full acceleration, i.e.,

$$\mathbf{a} = \texttt{GNN}(\mathbf{X}^{t_{k-H-1}:t_k}, \mathbf{g}) + \mathbf{g}_M^{FD} \,, \tag{4}$$

the model is forced to disentangle the interactions between external forces and internal dynamics, i.e., the other two terms on the right-hand side of Eq. (2). We attain a powerful formulation of the learning problem since the dynamics are controlled more explicitly, as shown in Fig. 1 and in Figs. 6 to 9 of Appendix A.

However, if the force $\mathbf{g}$ varies over space or time, it cannot be independently integrated over M time steps. In this case, modeling the correct effective external force requires (i) precise information on the forces that act on a given particle over each of the M steps we want to coarse-grain over, and (ii) taking the average over these contributions, i.e., $\mathbf{g}_M^{FD} = (M\Delta t_{SPH})^2 \frac{1}{M}\sum_{m=1}^{M} \mathbf{g}_m$. Since we typically do not have access to such information, we propose a convolution-based solution. In the case of a spatially varying but constant in time force field, we use the standard deviation of velocities over the dataset σ_u as a proxy of how much a particle moves perpendicularly to the force field, as this perpendicular motion is what we want to smoothen for. We then convolve the force function with a Gaussian distribution $\mathcal{N}(0, \sigma_u^2)$ with the standard deviation σ_u and thus smoothen the force function to account for the effective force exerted on a particle that moves across regions with variable forcing.

This convolution can be implemented in two ways: (i) If the function is simple enough, i.e., an analytical solution exists, we can use it directly. (ii) Alternatively, we may evaluate the instantaneous external force at the current particle coordinates and then apply an SPH kernel convolution, which is very similar to a convolution with a Gaussian, except that it has compact support. Applying a kernel $W(r|h)$ with $h = \sigma_u$ enables us to effectively smoothen any given force function. As a side remark, applying a convolution with an SPH kernel $W(\cdot|h)$ of a particular h over the mass of each adjacent particle is exactly what density summation does.

Correction of particle distribution via SPH relaxation. In order to correct the pathological particle clustering of learned GNN-based simulators, we add an intermediate step during the rollout of a learned Lagrangian solver, namely an *SPH relaxation step*. The idea is that if the learned solver pushes the system to an unphysical particle configuration, we can reduce density fluctuations by running an SPH relaxation simulation of up to 5 steps. By SPH relaxation, we refer to the process of taking the point cloud right after the temporal update of the learned model, and then – solely based on the particle coordinates – applying an SPH update with the assumption of zero initial velocities (Litvinov et al., 2015; Fan et al., 2024). We can apply SPH relaxation using the **pressure term** in Eq. (2) or the **viscous term** in Eq. (2). One update step of relaxation corresponds to

$$\mathbf{a} = \alpha\frac{-1}{\rho}\nabla p + \alpha\beta\nabla^2\mathbf{u} \,, \tag{5}$$

$$\mathbf{p} = \mathbf{p} + \mathbf{a} \,, \tag{6}$$

where we hide the time step and the pre-factors in the hyperparameters α and β. Adding and fine-tuning these hy-

perparameters is essential for various reasons: (a) in SPH, it proves challenging to identify a reference velocity, which is needed for determining the time step size; (b) adhering to the Courant-Friedrichs-Lewy (CFL) condition (Courant et al., 1928) would most certainly result in smaller time steps, and most importantly, (c) the step size is implicitly determined by how much the GNN-based simulator distorts the system. This largest distortion depends on many factors, such as temporal coarsening steps M and the choice of the GNN-based simulator. We propose fine-tuning these hyperparameters as shown later in this section.

Correction of density at walls and free surfaces. Recall that also existing SPH methods encounter challenges when predicting the density at free surfaces. On the one hand, density summation, which is the preferred method for density computation due to implicit mass conservation, is not directly applicable to free surfaces since it encounters density inconsistencies. On the other hand, density-transport equations abandon exact mass conservation.

For GNN-based simulators, we propose a novel way of estimating the density of a system at free surfaces. Our approach combines the SPH requirement that density fluctuations should not exceed $\sim 1\%$ – which we round up to 2% – with density summation. We extend density summation by (a) setting all values $< 0.98\rho_{ref}$ to ρ_{ref}, and (b) clipping all values $> 1.02\rho_{ref}$, i.e. setting them to $1.02\rho_{ref}$. Modification (a) guarantees that particles at free surfaces are set to the reference condition, preventing surface instabilities. Modification (b) truncates large outliers akin to gradient clipping when training a neural network, stabilizing the relaxation dynamics. Our approach is closely related to cavitation modeling, where it is common to use tensile instability control (TIC) (Sun et al., 2018) to avoid negative pressure values that increase the particle disorder and eventually lead to the occurrence of particle clustering and clumping (Lyu et al., 2022). The main idea of TIC is to change the pressure gradient formulation according to the particle location, e.g., at a free surface, and the sign of its pressure value (Sun et al., 2018). With this novel density computation routine, we can easily work with wall discretizations consisting of one wall layer, whereas standard SPH typically requires three or more wall layers (Adami et al., 2012). To complete the discussion on wall boundaries, we use the generalized wall boundary condition approach by Adami et al. (2012) to enforce the impermeability of the walls.

SPH Relaxation parameter tuning. We propose a three-step parameter-tuning process for the SPH relaxation parameters (see Appendix G.2 for examples):

1. Tune α while number of relaxation steps $l = 1$ and $\beta = 0$. Typically, $\alpha \in (0.005, 0.05)$.
2. Tune l with optimal α and $\beta = 0$. Typically, $l \in (1, 5)$.
3. Tune β with optimal α and l. Typically, $\beta \in (0.1, 1)$.

The measures we use while tuning are the position MSE, Sinkhorn divergence, kinetic energy MSE, MAE of density deviation from the reference ρ_{ref}, Dirichlet energy (Zhou & Schölkopf, 2005) of the density field, and Chamfer distance, see Appendix G for more details.

Related work. We want to stress that except for the proposed treatment of external forces, our method does not require retraining the GNN-based simulator. This differentiates our work from an orthogonal line of research, which has experienced a surge in recent years, namely using differentiable solvers as part of the machine learning model (Um et al., 2020). On the spectrum of classical numerical solvers to black-box end-to-end ML models, one also finds the class of hybrid models, which are ML models utilizing algorithmic ideas from classical solvers (Toshev et al., 2023b; Lienen & Günnemann, 2022; Karlbauer et al., 2022; Kochkov et al., 2021; Li & Farimani, 2022; Brandstetter et al., 2022b). Yet, all of these approaches construct a neural network that needs to be trained, whereas our SPH relaxation happens only during inference.

Conceptually closest to our work is the recent PDE-Refiner model class (Lippe et al., 2024). PDE-Refiner draws inspiration from diffusion models to apply a small number of refinement steps on learned Eulerian solvers. The refinement steps substantially improve the modeling of high frequency components, which yields more stable long-term predictions and better physics modeling, at the cost of increased inference time and a dedicated training routine. We point out that because PDE-Refiner is designed for Eulerian systems, it does not have the notion of dynamic particle coordinates underlying Lagrangian methods. Thus, extending PDE-Refiner to the Lagrangian description is not trivial, as one could choose to refine the accelerations or velocities or directly the particle coordinates, and such investigations are beyond the scope of this work. Furthermore, for particle systems, we do not have efficient ways to accurately evaluate high spatial frequencies over point clouds akin to the FFT on grids, and additionally, the physical setup of our problems does not involve high spatial frequencies.

4. Experiments

Our analyses are based on the datasets of Toshev & Adams (2024), accompanying the LagrangeBench paper (Toshev et al., 2024a). These datasets represent challenging coarse-grained temporal dynamics and contain long trajectories, i.e., up to thousands of steps. We test the performance difference of two popular GNN-based simulators when: (i) external forces are removed from the model target ($\square_g$), (ii) an SPH relaxation with pressure term is applied ($\square_p$), and (iii) an SPH relaxation with viscous term is applied ($\square_\nu$).

GNN-based simulators. The Graph Network-based Simulator (GNS) model (Sanchez-Gonzalez et al., 2020) is a popular learned surrogate for physical particle-based simulations and our main model. The architecture is kept simple, based on the encoder-processor-decoder principle, where the processor consists of multiple graph network blocks (Battaglia et al., 2018). Our second model, the Steerable E(3)-equivariant Graph Neural Network (SEGNN) (Brandstetter et al., 2022a) is a general implementation of an E(3) equivariant GNN, where layers are directly conditioned on steerable attributes for both nodes and edges. The main building block is the steerable MLP, i.e., a stack of learnable linear Clebsch-Gordan tensor products interleaved with gated non-linearities (Weiler et al., 2018). SEGNN layers are message-passing layers (Gilmer et al., 2017) where steerable MLPs replace the traditional non-equivariant MLPs for both message and node update functions. These two models were chosen as they present the current state-of-the-art surrogates for Lagrangian fluid dynamics (Toshev et al., 2024a), and also because they are representative of two fundamentally different classes of GNNs: non-equivariant (GNS) and equivariant (SEGNN).

Implementation of SPH relaxation. In our experience, it suffices to perform the relaxation operation for 1-5 iterations (l), depending on the problem. We summarize the used hyperparameters in Table 3 and Appendix B. Given that the learned surrogate is trained on every 100th SPH step, these additional SPH relaxation steps only marginally increase the rollout time – by a factor of 1.05-1.15 per relaxation step for a 10-layer 128-dimensional GNS model simulating the 2D RPF case, see Table 4 and Appendix E. In the same table, we observe an increase in runtime for 3D RPF and GNS-10-128 of roughly 1.4x per relaxation step, but we believe that this comes from the much more compute-intense neighbor search, which is reevaluated at every relaxation step. However, as the relaxation does not need to be implemented in a differentiable framework (we currently adopt JAX-SPH (Toshev et al., 2024b)), more efficient implementations, e.g. in C++, can significantly reduce these runtimes. For more compute-intense models like SEGNN the slowdown factor reduces, as the relaxation has a fixed computational cost independent of the particular GNN model.

Most of the computational overhead of the relaxation is due to its neighbor list, which has significantly more edges than the default neighbor list of the GNN-based simulators. The GNN graph generation uses the default radial cutoff distance from LagrangeBench, which corresponds to roughly 1.5 average particle distances. In contrast, the SPH relaxation uses the Quintic spline kernel with a cutoff of 3 average particle distances, i.e., the SPH relaxation operates on 2^d more edges, with dimension $d \in \{2, 3\}$. Therefore, our approach can be regarded as a multiscale approach, similar to the learned multi-scale interatomic potential presented by (Fu et al., 2023a). The difference is that in our approach, only the part using the smaller cutoff is a neural network, and the longer-range interactions simply stabilize the system in terms of better density distributions.

Training with SPH relaxation. An appealing idea is to use the SPH relaxation as a regularization during training, in the hope that we can omit running relaxations at inference time. We tried various ways of implementing this idea, but none of them improved rollout performance, see Appendix H.

Overview of results. Our results on 400-step rollouts using the GNS model are summarized in Table 1 and are averaged over all test trajectories and over the trajectory length. See Table 2 for the SEGNN results. As error measures, we use (a) the mean-squared error of positions (MSE_{400}), (b) the Sinkhorn divergence, which quantifies the conservation of the particle distribution, and (c) the kinetic energy error (MSE_{Ekin}) as a global measure of the physical behavior. The viscous term is shown only for reverse Poiseuille flow because it did not improve the performance on the other datasets. We note that by splitting the test sets into sequences of length 400, we obtain only 12-25 test trajectories, leading to noisy performance estimates. We discuss the necessity for larger datasets later in this section. For various parameter ablations, the evolution of error metrics with error bounds, and three more error metrics (density MAE, Dirichlet energy, and Chamfer distance), see Appendix G. Overall, all Neural SPH-enhanced simulators achieve better performance than the baseline GNNs, often by orders of magnitude, allowing for significantly longer rollouts and significantly better physics modeling.

Note on error thresholds. We note that upon tuning the parameters of our method, it either improves performance or converges to the baseline, with the latter being what mainly happens to RPF 3D according to Appendix B. We hypothesize that the baseline already produces very good particle distributions, and there is little potential for improvement. It thus seems necessary to define a threshold of when a learned simulator performs *well enough* in the sense of the requirements of the downstream task of interest. We refer to physical thresholds like the *chemical accuracy* in computational chemistry or the *energy and forces within threshold* measure used in the Open Catalyst project (Chanussot et al., 2021), both of which are designed to quantify whether a computational model is useful for practical applications. We stress the importance and leave the derivations of such thresholds for Lagrangian fluid simulations to future work.

4.1. External Force Treatment

In this section, we study the influence of the proposed external force treatment without combining it with the SPH relaxation. As only the dam break and reverse Poiseuille flow datasets have external force features, we focus on them.

	Model	MSE_{400}	Sinkhorn	MSE_{Ekin}
2D TGV	GNS	$5.3e-4$	$5.4e-7$	$5.6e-7$
	GNS_p	$4.8e-4$	$1.7e-8$	$4.8e-7$
2D RPF	GNS	$2.7e-2$	$3.6e-7$	$4.3e-3$
	GNS_g	$2.7e-2$	$2.7e-7$	$3.7e-4$
	$\text{GNS}_{g,p}$	$2.7e-2$	$2.9e-8$	$4.1e-4$
	$\text{GNS}_{g,p,\nu}$	$2.7e-2$	$3.0e-8$	$1.4e-4$
2D LDC	GNS	$3.3e-2$	$3.1e-4$	$1.1e-4$
	GNS_p	$1.6e-2$	$2.8e-7$	$1.2e-6$
2D DAM	GNS	$1.9e-1$	$3.8e-2$	$4.6e-2$
	GNS_g	$8.0e-2$	$1.3e-2$	$9.4e-3$
	GNS_p	$9.7e-2$	$7.1e-3$	$5.8e-3$
	$\text{GNS}_{g,p}$	$8.4e-2$	$7.5e-3$	$2.1e-3$
3D TGV	GNS	$4.8e-2$	$4.1e-6$	$3.6e-2$
	GNS_p	$4.6e-2$	$9.0e-7$	$4.2e-2$
3D RPF	GNS	$2.3e-2$	$4.4e-7$	$1.7e-5$
	GNS_g	$2.3e-2$	$4.4e-7$	$4.1e-5$
	GNS_p	$2.3e-2$	$1.0e-7$	$1.5e-5$
	$\text{GNS}_{g,p}$	$2.3e-2$	$1.3e-7$	$4.1e-5$
3D LDC	GNS	$3.2e-2$	$2.0e-5$	$1.3e-7$
	GNS_p	$3.2e-2$	$1.1e-6$	$2.9e-8$

Table 1. Performance measures averaged over a rollout of 400-steps. An additional subscript g indicates that external forces are removed from the model outputs, subscript p indicates that the SPH relaxation has a pressure term, and subscript ν that the viscosity term is added to the SPH relaxation. The numbers in the table are averaged over all test trajectories. MSE_{400} corresponds to: MSE_{120} for 2D TGV, MSE_{55} for 3D TGV, and MSE_{395} for 2D DAM, as these are the full trajectory lengths excluding initial history size $H = 5$.

Dam break (DAM). We saw a major performance boost on dam break when removing external forces from the target (GNS_g), see Table 1 and Appendix G.1. This simple modification of the training objective improves all considered measures by at least a factor of 2 and by as much as a factor of 5 on a rollout of the full dam break trajectory, i.e., 400 steps. For up to 20-step rollouts, GNS_g training does not improve the position error, which is in accordance with Sanchez-Gonzalez et al. (2020) and their Fig. C1. However, as the simulation length goes beyond 50 steps, numerical errors quickly accumulate and lead to artifacts like the one visible in the top part of Fig. 1. This particular failure mode in the front part of the dam break wave develops by first compressing the fluid to as much as $1.5\rho_{ref}$, and then the smallest instability in the tip causes particles to detach from the free surface. From there on, GNS starts acting as if the right wall has already been reached and fails to model the double wave structure from the reference solution, see Appendix A.

Force smoothing in reverse Poiseuille Flow. The external force of the reverse Poiseuille flow datasets is provided as a function corresponding to the instantaneous force, but when we train towards the effective dynamics over multiple original solver steps, we need to adjust this force. In particular, when predicting the dynamics over $M = 100$ temporal coarse-graining steps provided by LagrangeBench, an RPF particle might jump back and forth across the boundary separating the left- and right-ward forcing. Thus, it is not possible to infer the aggregated external force directly only knowing the particle coordinates at step M. We, therefore, apply a convolution of a Gaussian function with the force function. Since the forcing in RPF is a step function, this specific convolution has an analytical solution, i.e., the error function $\text{erf}(\cdot)$. We use $\text{erf}(\cdot)$ as a replacement for the original force function. See Appendix D for more details and visualization of the force before and after the convolution.

Reverse Poiseuille flow (RPF). See Fig. 3 for a subset of our ablation results on RPF 2D with GNS-10-128, or the full results on RPF 2D/3D and GNS/SEGNN in Appendix G.3. When removing external forces from the target of the GNS model (GNS_g), we observed that using the original, i.e., not smoothed, force leads to highly unstable dynamics in the shearing region, which causes the failure of the dynamics after less than 50 steps, see $\text{GNS}_{g_{raw}}$ in Figs. 27 and 28. When switching to the smoothed force function, the system becomes much more stable to perturbations and significantly improves the kinetic energy error. It is important to note that the kinetic energy is paramount to RPF, as this physical system is characterized by constant kinetic energy up to small fluctuations.

Looking at the 20-step position MSE reported in LagrangeBench, the GNS_g training leads to worse performance, roughly by a factor of 1.5 (see the beginning of the evolution in Fig. 3). This is important to note because we trade off worse short-term behavior in favor of better long-rollout performance, with the latter being the practical use-case we target. In this context, the LagrangeBench datasets pre-define a split of 50/25/25, which is far from sufficient if we want stable error estimates on rollouts of 400-step length, as also discussed, e.g., in Fu et al. (2023b).

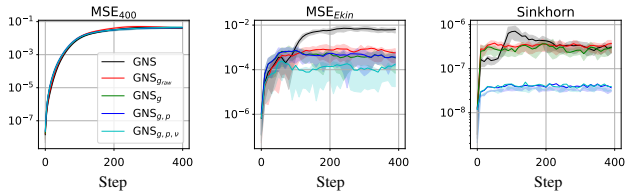

Figure 3. Ablations on RPF 2D with GNS-10-128 over the simulation length. Adapted from Fig. 26 in Appendix G.3.

4.2. SPH Relaxation

This section presents the results of our SPH relaxation on its own, and also in combination with the proposed external force treatment. We divide the discussion based on common characteristics of the datasets into periodic boundary cases, cases with wall boundaries, and free surface problems.

4.2.1. PERIODIC BOUNDARIES

Taylor-Green vortex (TGV). We did not expect the SPH relaxation to be very beneficial to the Taylor-Green vortex cases because (a) the trajectories are rather short with 125 and 60 steps in the 2D and 3D cases, respectively, and also (b) TGV represents a decaying problem, making it less prone to clustering in later stages of the trajectory. But according to Table 1, we get a consistent improvement of the position error MSE_{400} of $\sim 5\%$ and significant Sinkhorn divergence improvements on the 2D and 3D datasets.

Viscous term. In addition to external force subtraction, we found it beneficial to use the pressure (p) and viscous (ν) terms during relaxation, termed $\text{GNS}_{p,\nu}$. Viscosity, which manifests itself in shearing forces, in general, refers to the idea that if two fluid elements are close to each other but move in opposite directions, then they should both decelerate. Thus, to apply viscosity, we need to again approximate velocities by finite differences between consecutive positions of particles.

Reverse Poiseuille flow (RPF). In Figs. 4 and 10, we show histograms over velocity magnitudes to quantify how the different RPF correction terms impact the dynamics. Firstly, the original GNS model loses its high-velocity components over time, resembling a diffusion process, which makes it more stable with respect to perturbations, but, at the same time, leads to wrong kinetic energy. Secondly, simply changing the training objective by removing the external force (see GNS_g) already mitigates the problem of missing high velocities. And by adding the viscous term, which is especially relevant in the shearing region, to the pressure gradient term, we almost perfectly recover the target velocity distribution. See Fig. 3 and Appendix G.3 for further details.

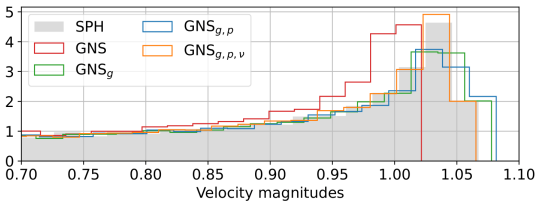

Figure 4. Velocity magnitudes histogram of 2D reverse Poiseuille flow after 400 rollout steps (averaged over all rollouts). Our $\text{GNS}_{g,p,\nu}$ matches the ground truth distribution of SPH.

4.2.2. WALL BOUNDARIES

A typical failure mode of learned solvers is that one or multiple particles penetrate what should be a solid wall, see top left part of Fig. 5 for LDC 2D and top part of Fig. 8 in Appendix A for DAM 2D. We solve this problem nearly completely with our SPH relaxations.

Relaxation at wall boundaries. The only part we have not discussed yet is how to ensure that particles do not escape the computational domain by passing through the walls. We use the simple and effective approach laid out in the generalized wall boundary condition paper by Adami et al. (2012). The idea of this approach is to enforce the impermeability of the walls by setting the pressure of the dummy wall particles to the average pressure of their adjacent fluid neighbors, see Eq. (27) in Adami et al. (2012), and, thus, constructing a setting of zero pressure gradients normal to the walls. With this boundary condition implementation, we obtain the following one-step relaxation algorithm: 1. density computation for fluid particles, 2. pressure computation for fluid particles through the equation of state, 3. computation of pressure of wall particles via weighted summation over the pressure of adjacent fluid particles, and 4. evaluation of the pressure gradient term, which gives the forces used to integrate the momentum equation Eq. (5) through Eq. (6).

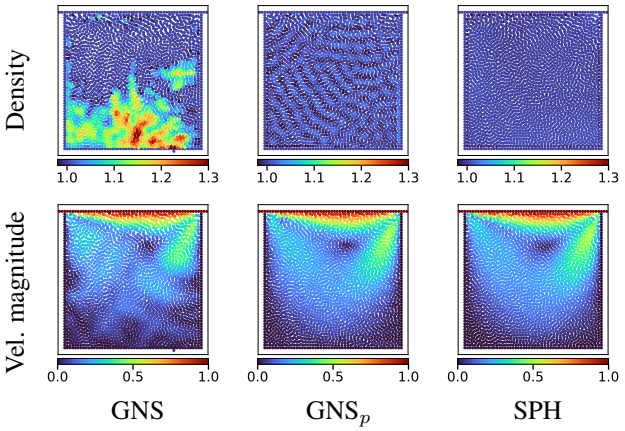

Figure 5. Density and velocity magnitude of 2D lid-driven cavity after 400 rollout steps (left to right): GNS, GNS_p, SPH. The colors in the first row correspond to the density deviation from the reference density; the system is considered physical within 0.98-1.02.

Lid-driven cavity (LDC). In the lid-driven cavity example, we observe that the learned model pushes particles away from the fast-moving lid into the lower half of the domain, which has profound consequences. On the one hand, the pressure at the bottom increases to an extent such that one or more particles gradually pass through the bottom wall. On the other hand, since too few particles reside close to the lid, the shearing forces are underrepresented, yielding a loss of kinetic energy, i.e., dynamics are lost. We fix both these issues with an SPH relaxation, forcing particles to be homogeneously distributed within the domain, see Figs. 5 and 8. See Appendix G.2 for various hyperparameter sensitivity ablations on LDC 2D/3D and GNS/SEGNN. While tuning the parameters is crucial, once tuned, they seem to work fairly reliably.

4.2.3. FREE SURFACES

A major difference between dam break and the other datasets we benchmark is that in dam break we not only care about the particle distribution within the fluid, but also about the volume filled with fluid. The latter is the focus of this section, and it is reflected in the MSE_{400} and Sinkhorn divergence measures, but not in MSE_{Ekin}.

Dam break (DAM). Interestingly, by either our external force treatment or the SPH relaxation, we seem to fix the problem of the fan-like spreading of the wavefront. We interpret this as a confirmation that the reason for this failure mode is the high compression at the tip. However, fixing the high compression levels in the bulk fluid requires our SPH relaxation, which we run with as few as three steps. The $\text{GNS}_{g,p}$ setup then recovers the correct dynamics with a significantly higher precision as measured by the Sinkhorn divergence, but also the kinetic energy MSE, indicating that the fluid also evolves more physically. Regarding the fluid surface, if we carefully look at the height of the fluid in Figs. 6 to 9, we see that the $\text{GNS}_{g,p}$ case very closely resembles the ground truth. See Appendix G.1 for ablations.

4.3. SEGNN Results

We applied the same external force treatment and SPH relaxations to the SEGNN model (Brandstetter et al., 2022a) without further tuning of the Neural SPH hyperparameters (see Appendix B) and summarize the results in Table 2. This comparison is useful not only for better comparability but also to show that proper SPH relaxation often depends more on the dataset than on the model – for example, moving the external force out of the 2D RPF case results in a 40 times lower kinetic energy error. However, in some cases, the GNS and SEGNN models behave quite differently. In most cases, SEGNN performs on par with GNS on long trajectories, with the notable SEGNN blowups on LDC 2D, DAM 2D, and RPF 3D. In particular, when we change the treatment of the external force in dam break without applying additional wall boundary conditions, we observe many particles falling through the bottom wall around step 200. Adding the relaxation with wall boundary conditions solves this problem, but investigating the qualitative differences between GNS and SEGNN would be an interesting future work. See Appendix G for our hyperparameter ablations.

	Model	MSE_{400}	Sinkhorn	MSE_{Ekin}
2D TGV	SEGNN	$4.0e-4$	$4.4e-7$	$3.9e-7$
	SEGNN_p	$3.8e-4$	$1.5e-8$	$2.8e-7$
2D RPF	SEGNN	$2.7e-2$	$3.3e-7$	$4.3e-3$
	SEGNN_g	$2.8e-2$	$3.3e-7$	$1.2e-4$
	$\text{SEGNN}_{g,p}$	$2.8e-2$	$3.5e-8$	$1.6e-4$
	$\text{SEGNN}_{g,p,\nu}$	$2.8e-2$	$3.8e-8$	$7.3e-4$
2D LDC	SEGNN	$7.6e-2$	$2.3e-3$	$9.1e+0$
	SEGNN_p	$1.8e-2$	$5.8e-7$	$1.6e-5$
2D DAM	SEGNN	$1.5e-1$	$3.4e-2$	$1.9e-2$
	SEGNN_g	$1.6e-1$	$2.1e-2$	$1.9e+1$
	SEGNN_p	$1.2e-1$	$9.4e-3$	$1.2e-2$
	$\text{SEGNN}_{g,p}$	$8.6e-2$	$4.9e-3$	$2.6e-3$
3D TGV	SEGNN	$4.2e-2$	$6.1e-6$	$2.4e-2$
	SEGNN_p	$4.1e-2$	$6.0e-7$	$2.7e-2$
3D RPF	SEGNN	$1.2e-1$	$1.0e-4$	$1.5e+3$
	SEGNN_p	$2.6e-2$	$1.3e-5$	$1.8e-2$
	SEGNN_g	$2.7e-2$	$2.6e-6$	$9.5e-3$
	$\text{SEGNN}_{g,p}$	$2.6e-2$	$7.9e-7$	$5.7e-3$
3D LDC	SEGNN	$3.3e-2$	$2.3e-5$	$1.7e-7$
	SEGNN_p	$3.3e-2$	$2.0e-6$	$1.8e-7$

Table 2. SEGNN-10-64 results. Same structure as Table 1.

5. Concluding Remarks

We introduce Neural SPH, a framework for improved training and inference of GNN-based simulators for Lagrangian fluid dynamics simulations. We demonstrate the utility of our toolkit on seven diverse 2D and 3D datasets and on two state-of-the-art GNN-based simulators, GNS and SEGNN. We identify particle clustering originating from tensile instabilities as one of the primary pitfalls of GNN-based simulators. Through the proposed external force treatment and SPH relaxation step, distribution-induced errors are minimized, leading to more robust and physically consistent dynamics. Compared to other methods, Neural SPH does not require a differentiable solver and increases the inference time only by a fixed and rather small amount.

Limitations and future work. We observe that tuning the hyperparameters of the particle relaxation is crucial since redistributing the particles inherently translates to modified velocity histories, which directly enter the next autoregressive update step. Thus, the learned solver may become unstable by bringing the past velocities out-of-distribution. Although using the proposed hyperparameter tuning recipe leads to a fairly stable inference routine of the learned solvers, further improving this recipe might be beneficial. Another potential limitation concerns the handling of external forces, namely, that information on the timestep and coarsening level of the dataset is required. Finally, and related to the parameter tuning, we point out the necessity of defining physical thresholds akin to the *energy and force within threshold* by (Chanussot et al., 2021), to identify whether our Neural SPH improvements are needed in the first place. Our work shows what is possible by integrating machine learning models with established simulation routines like enforcing boundary conditions or improving particle spreading, but one can extend this idea by adding arbitrarily many terms from the enormous body of literature on classical numerics. We point out that the proposed alternation of learned and classical solver terms is a framework, applicable to any combination of compatible methods, extending beyond GNNs and Lagrangian systems.

Impact Statement

Smoothed particle hydrodynamics plays a crucial role in computational fluid dynamics. Examples can be found in aerodynamics, astrophysics, or plasma physics. Given the widespread application of computational fluid dynamics, obtaining shortcuts or alternatives for computationally expensive simulations is essential for advancing scientific research, and has direct or indirect implications for reducing our carbon footprint. However, it is important to note that relying on simulations always necessitates thorough cross-checks and monitoring, especially when employing a "learning to simulate" methodology.

Acknowledgements

The authors thank Fabian Thiery, Christopher Zöller, and Steffen Schmidt for helpful discussions on SPH at free surfaces.

Author Contributions

A.T. conceived the ideas of SPH relaxation and the proposed external force treatment, implemented them, ran the experiments, and wrote the first version of the manuscript. J.E. contributed the Dirichlet energy metric and wrote the literature review on density summation at free surfaces. N.A. and J.B. supervised the project from conception to design of experiments and analysis of the results. All authors contributed to the manuscript.

References

Adami, S., Hu, X., and Adams, N. A. A generalized wall boundary condition for smoothed particle hydrodynamics. *Journal of Computational Physics*, 231(21):7057–7075, 2012.

Adami, S., Hu, X., and Adams, N. A. A transport-velocity formulation for smoothed particle hydrodynamics. *Journal of Computational Physics*, 241:292–307, 2013.

Alkin, B., Fürst, A., Schmid, S., Gruber, L., Holzleitner, M., and Brandstetter, J. Universal physics transformers. *arXiv preprint arXiv:2402.12365*, 2024.

Antoci, C., Gallati, M., and Sibilla, S. Numerical simulation of fluid–structure interaction by sph. *Computers & structures*, 85(11-14):879–890, 2007.

Batatia, I., Kovacs, D. P., Simm, G., Ortner, C., and Csányi, G. Mace: Higher order equivariant message passing neural networks for fast and accurate force fields. *Advances in Neural Information Processing Systems*, 35: 11423–11436, 2022.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):2453, 2022.

Becker, M. and Teschner, M. Weakly compressible sph for free surface flows. pp. 1–8. Eurographics Association, 2007.

Bodnar, C., Bruinsma, W., Lucic, A., Stanley, M., Brandstetter, J., Garvan, P., Riechert, M., Weyn, J., Dong, H., Vaughan, A., Gupta, J., Tambiratnam, K., Archibald, A., Heider, E., Welling, M., Turner, R., and Perdikaris, P. Aurora: A foundation model of the atmosphere. Technical Report MSR-TR-2024-16, Microsoft Research AI for Science, May 2024.

Bonet, J. and Lok, T.-S. Variational and momentum preservation aspects of smooth particle hydrodynamic formulations. *Computer methods in applied mechanics and engineering*, 180:97–115, 1999.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018.

Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E. J., and Welling, M. Geometric and physical quantities improve e(3) equivariant message passing. In *ICLR*, 2022a.

Brandstetter, J., Worrall, D. E., and Welling, M. Message passing neural PDE solvers. In *ICLR*, 2022b.

Brunton, S. L. and Kutz, J. N. Machine Learning for Partial Differential Equations. *arXiv preprint arXiv:2303.17078*, March 2023.

Cai, C. and Wang, Y. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

Chanussot, L., Das, A., Goyal, S., Lavril, T., Shuaibi, M., Riviere, M., Tran, K., Heras-Domingo, J., Ho, C., Hu, W., et al. Open catalyst 2020 (oc20) dataset and community challenges. *Acs Catalysis*, 11(10):6059–6072, 2021. doi: 10.1021/acscatal.0c04525.

Colagrossi, A. and Landrini, M. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of computational physics*, 191(2):448–475, 2003.

Courant, R., Friedrichs, K., and Lewy, H. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische annalen*, 100(1):32–74, 1928.

Di Giovanni, F., Rowbottom, J., Chamberlain, B. P., Markovich, T., and Bronstein, M. M. Understanding convolution on graphs via energies. *arXiv preprint arXiv:2206.10991*, 2022.

Diening, L., Harjulehto, P., Hästö, P., and Ruzicka, M. *Lebesgue and Sobolev Spaces with Variable Exponents*, volume 1, chapter 13. Springer Berlin, 2011.

Dilts, G. A. Moving least-squares particle hydrodynamics ii: conservation and boundaries. *International Journal for Numerical Methods in Engineering*, 48:1503–1524, 2000.

Fan, Y., Li, X., Zhang, S., Hu, X., and Adams, N. A. Analysis of the particle relaxation method for generating uniform particle distributions in smoothed particle hydrodynamics. 2024. doi: 10.13140/RG.2.2.29175.80806.

Fu, X., Musaelian, A., Johansson, A., Jaakkola, T., and Kozinsky, B. Learning interatomic potentials at multiple scales. *arXiv preprint arXiv:2310.13756*, 2023a.

Fu, X., Wu, Z., Wang, W., Xie, T., Keten, S., Gomez-Bombarelli, R., and Jaakkola, T. S. Forces are not enough: Benchmark and critical evaluation for machine learning force fields with molecular simulations. *Transactions on Machine Learning Research*, 2023b. ISSN 2835-8856. Survey Certification.

Gasteiger, J., Becker, F., and Günnemann, S. Gemnet: Universal directional graph neural networks for molecules. *NeurIPS*, 34:6790–6802, 2021.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, pp. 1263–1272. PMLR, 2017.

Gingold, R. A. and Monaghan, J. J. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

Gomez-Gesteira, M., Rogers, B. D., Dalrymple, R. A., and Crespo, A. J. State-of-the-artofclassicalsphforfree-surfaceflows. *Journal of Hydraulic Research*, 48:6–27, 2010.

Guo, X., Li, W., and Iorio, F. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 481–490, 2016.

Gupta, J. K. and Brandstetter, J. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

Hirt, C. W., Amsden, A. A., and Cook, J. An arbitrary lagrangian-eulerian computing method for all flow speeds. *Journal of computational physics*, 14(3):227–253, 1974.

Hu, X. and Adams, N. A. An incompressible multi-phase sph method. *Journal of computational physics*, 227(1): 264–278, 2007.

Karlbauer, M., Praditia, T., Otte, S., Oladyshkin, S., Nowak, W., and Butz, M. V. Composing partial differential equations with physics-aware neural networks. In *International Conference on Machine Learning*, pp. 10773–10801. PMLR, 2022.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Klimesch, J., Holl, P., and Thuerey, N. Simulating liquids with graph networks. *arXiv preprint arXiv:2203.07895*, 2022.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Pritzel, A., Ravuri, S., Ewalds, T., Alet, F., Eaton-Rosen, Z., et al. GraphCast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*, 2022.

Li, Z. and Farimani, A. B. Graph neural network-accelerated lagrangian fluid simulation. *Computers & Graphics*, 103: 201–211, 2022.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *ICLR*, 2021.

Lienen, M. and Günnemann, S. Learning the dynamics of physical systems from sparse observations with finite element networks. In *International Conference on Learning Representations (ICLR)*, 2022.

Lippe, P., Veeling, B., Perdikaris, P., Turner, R., and Brandstetter, J. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36, 2024.

Litvinov, S., Hu, X., and Adams, N. A. Towards consistence and convergence of conservative sph approximations. *Journal of Computational Physics*, 301:394–401, 2015.

Lucy, L. B. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal, vol. 82, Dec. 1977, p. 1013-1024.*, 82:1013–1024, 1977.

Lyu, H.-G., Sun, P., Colagrossi, A., and Zhang, A.-M. Towards sph simulations of cavitating flows with an eosb cavitation model. *Acta Mechanica Sinica*, 39, 07 2022. doi: 10.1007/s10409-022-22158-x.

Marrone, S., Antuono, M., Colagrossi, A., Colicchio, G., Le Touzé, D., and Graziani, G. δ-sph model for simulating violent impact flows. *Computer Methods in Applied Mechanics and Engineering*, 200(13-16):1526–1542, 2011.

Mayr, A., Lehner, S., Mayrhofer, A., Kloss, C., Hochreiter, S., and Brandstetter, J. Boundary graph neural networks for 3d simulations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9099–9107, 2023.

Merchant, A., Batzner, S., Schoenholz, S. S., Aykol, M., Cheon, G., and Cubuk, E. D. Scaling deep learning for materials discovery. *Nature*, pp. 1–6, 2023.

Monaghan, J. J. Simulating free surface flows with sph. *Journal of computational physics*, 110(2):399–406, 1994.

Monaghan, J. J. Smoothed particle hydrodynamics. *Reports on progress in physics*, 68(8):1703, 2005.

Nguyen, T., Brandstetter, J., Kapoor, A., Gupta, J. K., and Grover, A. ClimaX: A foundation model for weather and climate. *arXiv preprint arXiv:2301.10343*, 2023.

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., Hassanzadeh, P., Kashinath, K., and Anandkumar, A. FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators. *arXiv preprint arXiv:2202.11214*, 2022.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Price, D. J. Smoothed particle hydrodynamics and magnetohydrodynamics. *Journal of Computational Physics*, 231 (3):759–794, 2012.

Rasp, S. and Thuerey, N. Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for weatherbench. *Journal of Advances in Modeling Earth Systems*, 13(2): e2020MS002405, 2021.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pp. 8459–8468. PMLR, 2020.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Shepard, D. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, pp. 517—-524. Association for Computing Machinery, 1968.

Sigalotti, L. D. G., Daza, J., and Donoso, A. Modelling free surface flows with smoothed particle hydrodynamics. *Condensed Matter Physics*, 9:359–366, 2006.

Sønderby, C. K., Espeholt, L., Heek, J., Dehghani, M., Oliver, A., Salimans, T., Agrawal, S., Hickey, J., and Kalchbrenner, N. Metnet: A neural weather model for precipitation forecasting. *arXiv preprint arXiv:2003.12140*, 2020.

Sun, P., Colagrossi, A., Marrone, S., Antuono, M., and Zhang, A. Multi-resolution delta-plus-sph with tensile instability control: Towards high reynolds number flows. *Computer Physics Communications*, 224:63–80, 2018.

Taheri, A. Minimizing the dirichlet energy over a space of measure preserving maps. *Topological Methods in Nonlinear Analysis*, 33:170–204, 2009.

Thuerey, N., Holl, P., Mueller, M., Schnell, P., Trost, F., and Um, K. Physics-based Deep Learning. *arXiv preprint arXiv:2109.05237*, 2021.

Toshev, A., Galletti, G., Fritz, F., Adami, S., and Adams, N. Lagrangebench: A lagrangian fluid mechanics benchmarking suite. *Advances in Neural Information Processing Systems*, 36, 2024a.

Toshev, A. P. and Adams, N. A. Lagrangebench datasets, January 2024. URL https://doi.org/10.5281/zenodo.10491868.

Toshev, A. P., Galletti, G., Brandstetter, J., Adami, S., and Adams, N. A. Learning lagrangian fluid mechanics with e(3)-equivariant graph neural networks. In Nielsen, F. and Barbaresco, F. (eds.), *Geometric Science of Information*, pp. 332–341, Cham, 2023a. Springer Nature Switzerland. ISBN 978-3-031-38299-4.

Toshev, A. P., Paehler, L., Panizza, A., and Adams, N. A. On the relationships between graph neural networks for the simulation of physical systems and classical numerical methods. *arXiv preprint arXiv:2304.00146*, 2023b.

Toshev, A. P., Ramachandran, H., Erbesdobler, J. A., Galletti, G., Brandstetter, J., and Adams, N. A. Jax-sph: A differentiable smoothed particle hydrodynamics framework. *arXiv preprint arXiv:2403.04750*, 2024b.

Um, K., Brand, R., Fei, Y., Holl, P., and Thuerey, N. Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers. *Advances in Neural Information Processing Systems*, 2020.

Violeau, D. and Rogers, B. D. Smoothed particle hydrodynamics (sph) for free-surface flows: past, present and future. *Journal of Hydraulic Research*, 54(1):1–26, 2016.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. S. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *NeurIPS*, volume 31. Curran Associates, Inc., 2018.

Weyn, J. A., Durran, D. R., and Caruana, R. Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Journal of Advances in Modeling Earth Systems*, 12(9): e2020MS002109, 2020.

Winchenbach, R. and Thuerey, N. Symmetric basis convolutions for learning lagrangian fluid mechanics. *arXiv preprint arXiv:2403.16680*, 2024.

Zeni, C., Pinsler, R., Zügner, D., Fowler, A., Horton, M., Fu, X., Shysheya, S., Crabbé, J., Sun, L., Smith, J., et al. Mattergen: a generative model for inorganic materials design. *arXiv preprint arXiv:2312.03687*, 2023.

Zhang, C., Hu, X., and Adams, N. A. A weakly compressible sph method based on a low-dissipation riemann solver. *Journal of Computational Physics*, 335:605–620, 2017a.

Zhang, C., Hu, X. Y., and Adams, N. A. A generalized transport-velocity formulation for smoothed particle hydrodynamics. *Journal of Computational Physics*, 337: 216–232, 2017b.

Zhou, D. and Schölkopf, B. Regularization on discrete spaces. In Kropatsch, W. G., Sablatnig, R., and Hanbury, A. (eds.), *Pattern Recognition*, pp. 361–368, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

A. Dam Break Plots

In this section, we show some more examples of dam break trajectories. Roughly one-third of GNS trajectories have the same artifacts at step 80 as test trajectory 0 (see Figs. 6 and 7). Roughly half of the GNS trajectories show large amounts of particles leaving the box on the right at step 80 (see Fig. 8). Only a few GNS simulations behave better at step 80 (see Fig. 9).

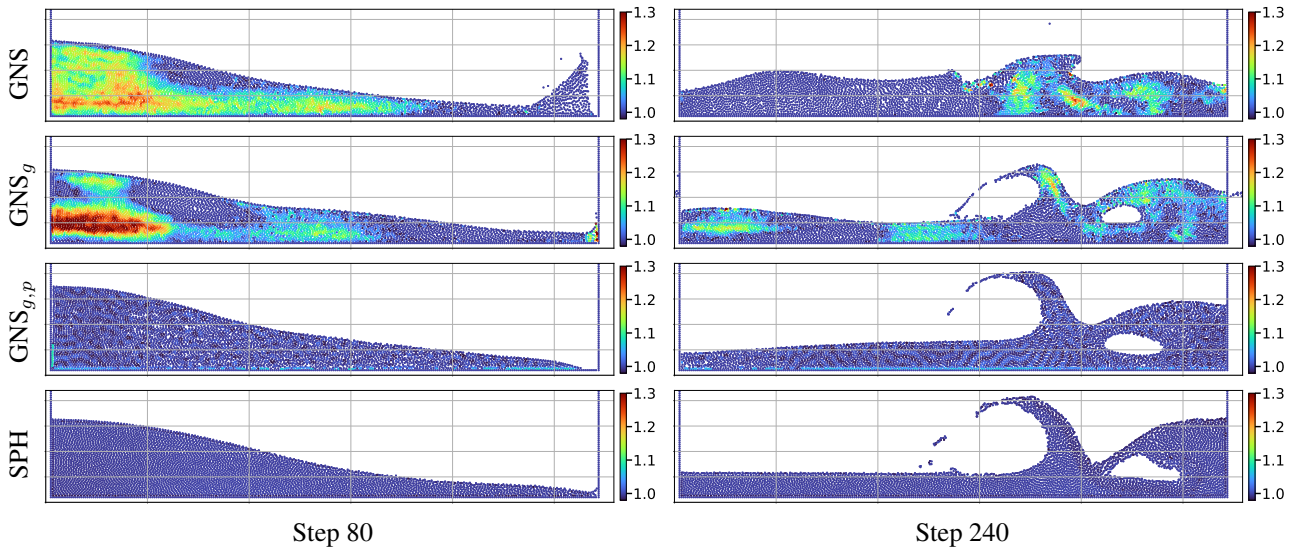

Figure 6. Dam break steps 80 and 240 of test rollout 0. Extends Fig. 1.

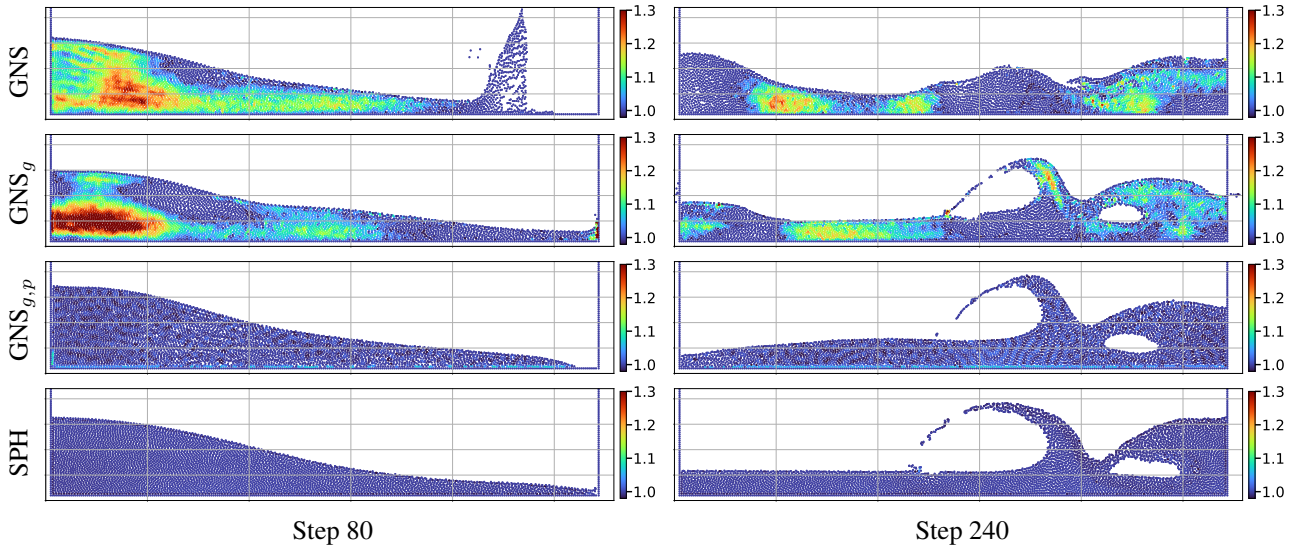

Figure 7. Dam break steps 80 and 240 of test rollout 13.

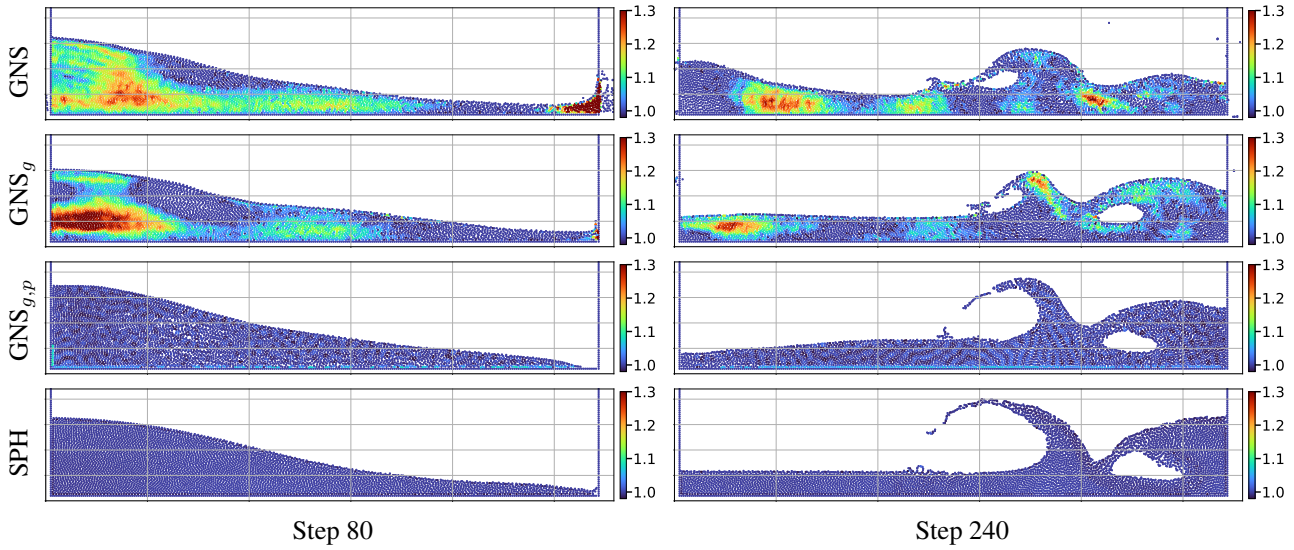

Figure 8. Dam break steps 80 and 240 of test rollout 14.

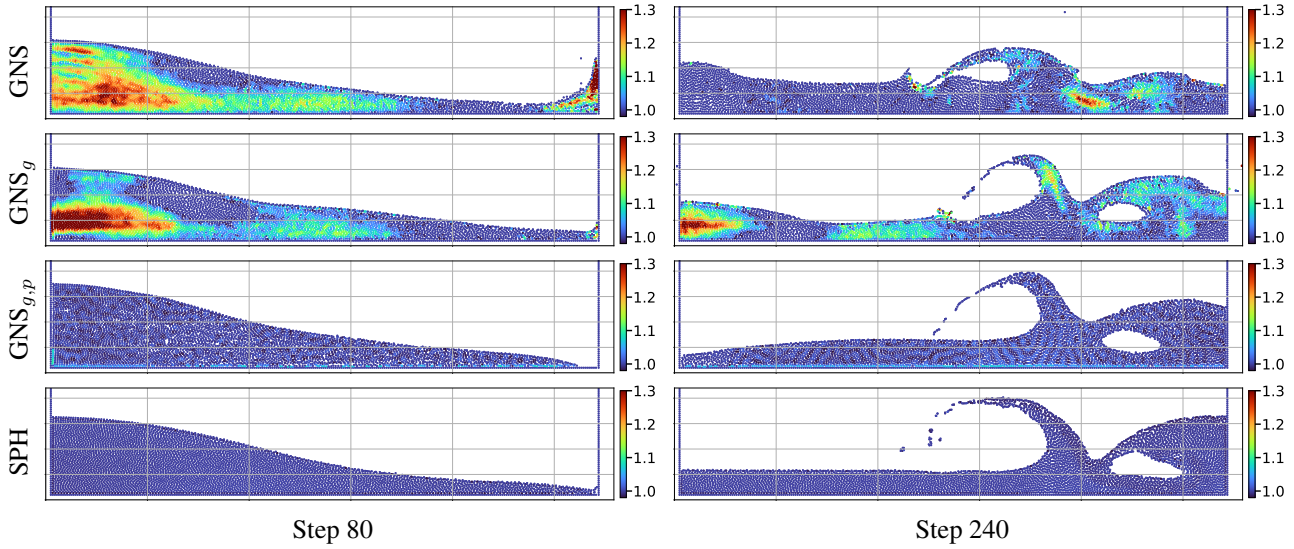

Figure 9. Dam break steps 80 and 240 of test rollout 15.

B. Hyperparameters of GNS model

Dataset	loops	α	β
2D TGV	5	0.02	–
2D RPF	3	0.02	0.2
2D LDC	5	0.03	–
2D DAM	3	0.03	–
3D TGV	1	0.01	–
3D RPF	1	0.005	–
3D LDC	1	0.02	–

Table 3. SPH relaxation hyperparameters used in our experiments. These hyperparameters were tuned on the GNS-10-128 model.

C. RPF 2D Plots

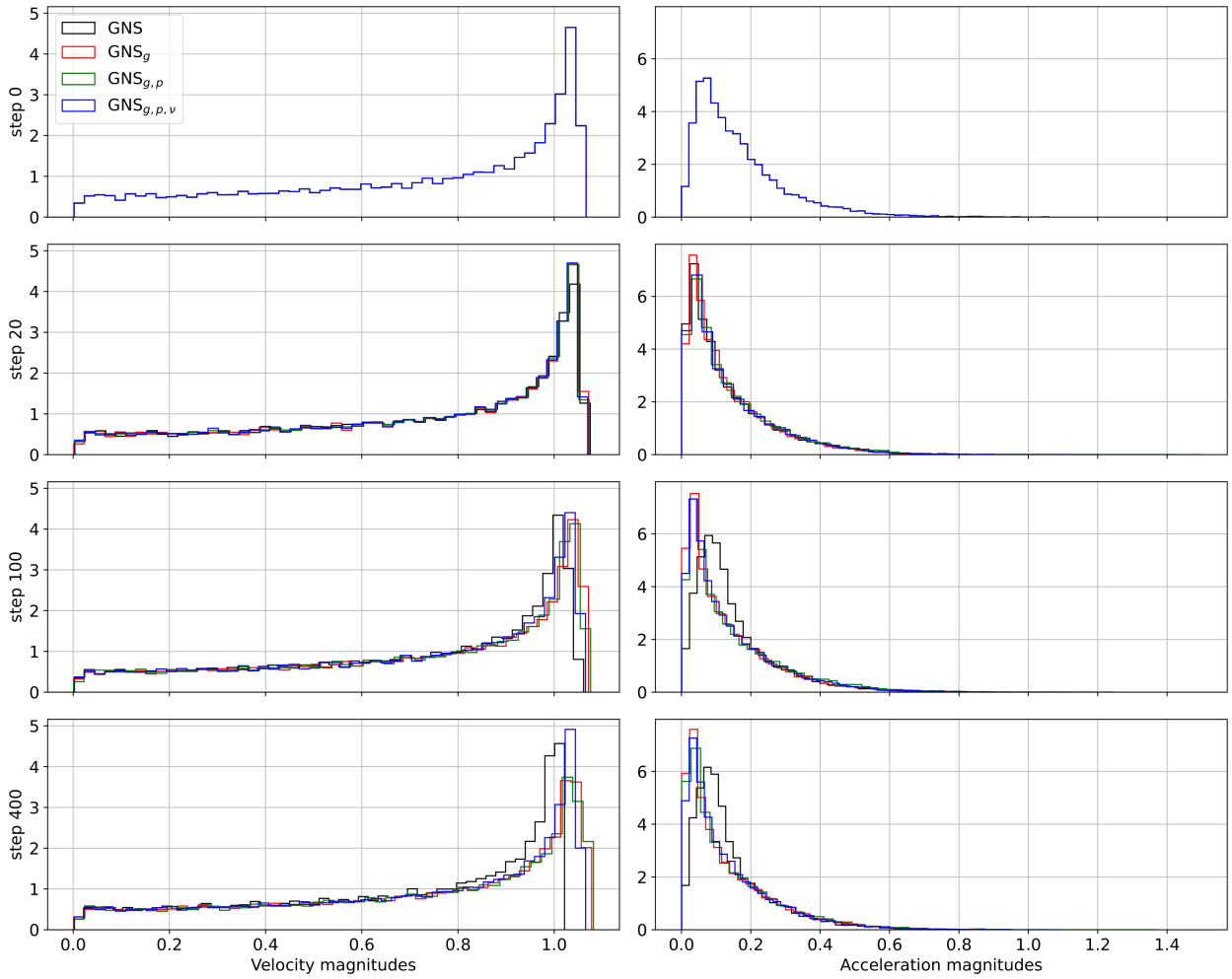

Figure 10. Velocity and acceleration magnitude histogram of 2D reverse Poiseuille flow after 400 rollout steps (average over all rollouts). Extends Fig. 4.

D. Forcing of Reverse Poiseuille Flow

The forcing step function of the reverse Poiseuille flow (RPF) is given by:

$$f(x,y,z) = \begin{cases} [-1,0,0]\,, & \text{if } y > 1 \\ [1,0,0] & \text{otherwise}\,. \end{cases} \tag{7}$$

For the two-dimensional case, the z value can be ignored. We use the analytical solution of the convolution of the forcing step function with a Gaussian kernel of width that corresponds to the standard deviation of the velocities over the dataset. In this special case, the convolution has an analytical solution given by the error function erf. For the jump in the middle, we obtain the solution

$$f_{smooth}(x,y,z) = [-\text{erf}\left(\frac{y-1}{\sqrt{2}\sigma}\right), 0, 0]\,. \tag{8}$$

We use the finite difference approximation between consecutive coordinate frames to approximate the standard deviation of the velocity. For 2D RPF, the velocity standard deviation is $[0.036, 0.00069]$, and for 3D RPF $[0.074, 0.0014, 0.0011]$.

We first convert these two standard deviation vectors to their isotropic versions, assuming that the velocity components are independent Gaussian random variables, i.e., using the quadratic mean. This leads to $\sigma_{2D} = 0.025$ and $\sigma_{3D} = 0.043$. We round the numbers and use the values $\sigma_{2D} = 0.025$ and $\sigma_{3D} = 0.05$ in our experiments. The result of this smoothing procedure can be seen in Fig. 11.

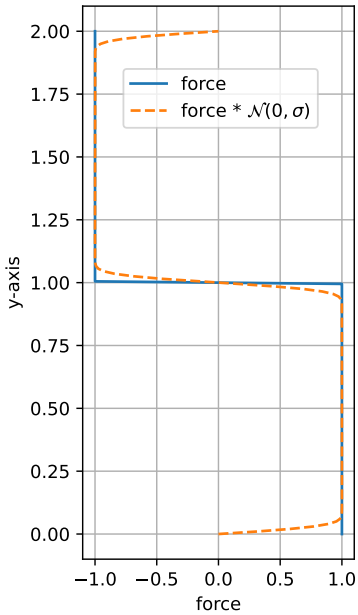

Figure 11. Forcing step function of the 2D reverse Poiseuille flow before (blue) and after convolution with normal distribution $\mathcal{N}(0, 0.025^2)$ (orange).

E. Inference Speed

We measured the inference speed of GNS-10-128 and SEGNN-10-64 on the 2D and 3D reverse Poiseuille flow datasets with 0, 1, 3, or 5 relaxation steps l and summarize the results in Table 4. This table provides more quantitative results to the discussion on inference speed in Section 4.

Dataset	t_{gt} [ms]	Model	$t_{l=0}$ [ms]	$t_{l=1}$ [ms]	$t_{l=3}$ [ms]	$t_{l=5}$ [ms]
2D RPF	43.0	GNS	10.7	11.0	13.3	14.4
		SEGNN	24.9	25.9	28.4	30.4
3D RPF	424	GNS	23.8	32.5	50.4	68.0
		SEGNN	97.9	106	124	141

Table 4. Timing experiments on RPF datasets with GNS-10-128 model. With t_{gt}, we denote the time the ground truth SPH solver takes to simulate 100 steps, as the LagrangeBench datasets consist of every 100th solver state. We took the values $t_{gt} = 43.0$ and $t_{gt} = 424$ from Table 4 in Toshev et al. (2024a). Timing runs are averaged over 10k forward calls to the model and consecutive position relaxations.

F. Temporal Coarsening

Semi-implicit Euler:

$$\mathbf{u}_1 = \mathbf{u}_0 + \Delta t \mathbf{a}_0 \tag{9}$$
$$\mathbf{p}_1 = \mathbf{p}_0 + \Delta t \mathbf{u}_1 \tag{10}$$
$$= \mathbf{p}_0 + \Delta t \mathbf{u}_0 + \Delta t^2 \mathbf{a}_0 \tag{11}$$
$$\mathbf{u}_2 = \mathbf{u}_1 + \Delta t \mathbf{a}_1 \tag{12}$$
$$= \mathbf{u}_0 + \Delta t(\mathbf{a}_0 + \mathbf{a}_1) \tag{13}$$
$$\mathbf{p}_2 = \mathbf{p}_1 + \Delta t \mathbf{u}_2 \tag{14}$$
$$= (\mathbf{p}_0 + \Delta t \mathbf{u}_0 + \Delta t^2 \mathbf{a}_0) + \Delta t(\mathbf{u}_0 + \Delta t(\mathbf{a}_0 + \mathbf{a}_1)) \tag{15}$$
$$= \mathbf{p}_0 + \Delta t 2\mathbf{u}_0 + \Delta t^2 (2\mathbf{a}_0 + \mathbf{a}_1) \tag{16}$$
$$\vdots$$
$$\mathbf{u}_M = \mathbf{u}_0 + \Delta t \sum_{m=0}^{M-1} \mathbf{a}_m \tag{17}$$
$$\mathbf{p}_M = \mathbf{p}_0 + M \Delta t \mathbf{u}_0 + \Delta t^2 \sum_{m=0}^{M-1} (M - m) \mathbf{a}_m \,. \tag{18}$$

If $\mathbf{a}_m$ is a constant number, we can simplify the last part to:

$$\mathbf{u}_M = \mathbf{u}_0 + M \Delta t \mathbf{a} \tag{19}$$
$$\mathbf{p}_M = \mathbf{p}_0 + M \Delta t \mathbf{u}_0 + 0.5 M(M+1) \Delta t^2 \mathbf{a} \,. \tag{20}$$

If we now compute the target effective acceleration by finite differences of positions, we end up with

$$\mathbf{u}_0^{FD} = (\mathbf{p}_0 - \mathbf{p}_{-M}) / \Delta t^{FD} \tag{21}$$
$$\mathbf{u}_M^{FD} = (\mathbf{p}_M - \mathbf{p}_0) / \Delta t^{FD} \tag{22}$$
$$\mathbf{a}_0^{FD} = (\mathbf{u}_M^{FD} - \mathbf{u}_0^{FD}) / \Delta t^{FD} = (\mathbf{p}_M - 2\mathbf{p}_0 + \mathbf{p}_{-M}) / \Delta t^{FD^2}. \tag{23}$$

By substituting the semi-implicit Euler rule after M steps into this finite differences approximation and setting $\Delta t^{FD} = 1$ for simplicity, we get an effective acceleration of

$$\mathbf{a}_{iM}^{FD} = \mathbf{p}_{(i+1)M} - 2\mathbf{p}_{iM} + \mathbf{p}_{(i-1)M} \tag{24}$$
$$\begin{aligned} &= M(\Delta t \mathbf{u}_0((i+1) - 2i + (i-1)) \\ &+ 0.5 \Delta t^2 \mathbf{a}(((i+1)^2 M + (i+1)) - 2(i^2 M + i) + ((i-1)^2 M + (i-1)))) \end{aligned} \tag{25}$$
$$= M\left(0 + 0.5 \Delta t^2 \mathbf{a}(2M)\right) \tag{26}$$
$$= (M \Delta t)^2 \mathbf{a} \,. \tag{27}$$

G. Ablations

We extend the results from the main paper by running multiple ablation studies mainly focusing on (a) the individual and combined impact of SPH relaxation and force treatment on the example of **dam break**, (b) the sensitivity of the parameters governing the proposed SPH relaxation on the example of **lid-driven cavity**, and (c) the impact of smoothing the external force function on the example of the **reverse Poiseuille flow** datasets. We believe that this exhaustive analysis of the hyperparameters is essential for practitioners who would consider using our proposed methods. To increase the value of the analysis we add (A) the evolution of the metrics over the simulation length, (B) error bars representing the 0.25 and 0.75 quantiles over the test trajectories, and (C) three more metrics compared to the main paper. The six metrics we use are:

1. MSE_{400} – position MSE over 400 steps.

2. MSE_{Ekin} – kinetic energy MSE between the predicted and ground truth frames.

3. Sinkhorn – Sinkhorn divergence between the particle distribution of predicted and ground truth frames. Measures how much effort it would take to move the particle mass between the two states. Scales as $\mathcal{O}(N^2)$ with the number of particles N and is more compute intense than the model inference on all our datasets.

4. MAE_ρ – density MAE error measuring the deviation of the density from the reference density ρ_{ref}. In all our experiments $\rho_{ref} = 1.0$.

5. Dirichlet – Dirichlet energy (Zhou & Schölkopf, 2005) of density field $E_D(\rho) = \frac{1}{2}\int \|\nabla\rho\|_2^2 \, dx$, based on Taheri (2009); Diening et al. (2011). It measures both high-frequency (e.g. clustering) and low-frequency (e.g. instabilities) density fluctuations. Lower is better and means less steep gradients (Cai & Wang, 2020; Di Giovanni et al., 2022).

6. Chamfer – symmetric Chamfer distance $d_{CD}(X,Y) = \sum_{x\in X} \min_{y\in Y} ||x-y||_2^2 + \sum_{y\in Y} \min_{x\in X} ||x-y||_2^2$ between predicted and ground truth frames. Similar to Sinkhorn, but only considers nearest neighbors, and thus much more compute efficient.

For all these measures applies: lower is better, and 0.0 is best.

G.1. Dam Break

We compare the impact of our external force treatment ($\square_g$), our SPH relaxation with parameters from Table 3 ($\square_p$), and combination of both ($\square_{g,p}$) on the dam break dataset using the GNS (Fig. 12) and SEGNN (Fig. 13). On the MSE_{Ekin}, we see that only through the combination of our force treatment and SPH relaxation we achieve significant performance boosts with both the GNS and SEGNN models.

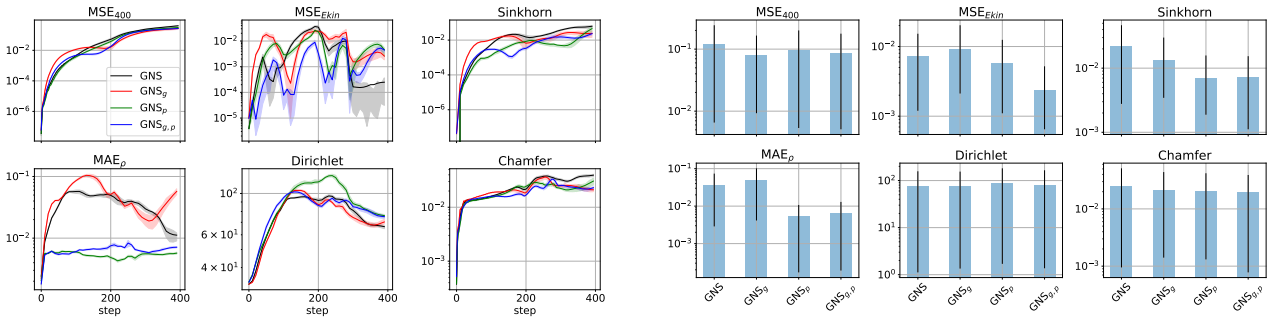

Figure 12. Ablations on DAM 2D with GNS-10-128 over the simulation length (left) and the average thereof (right).

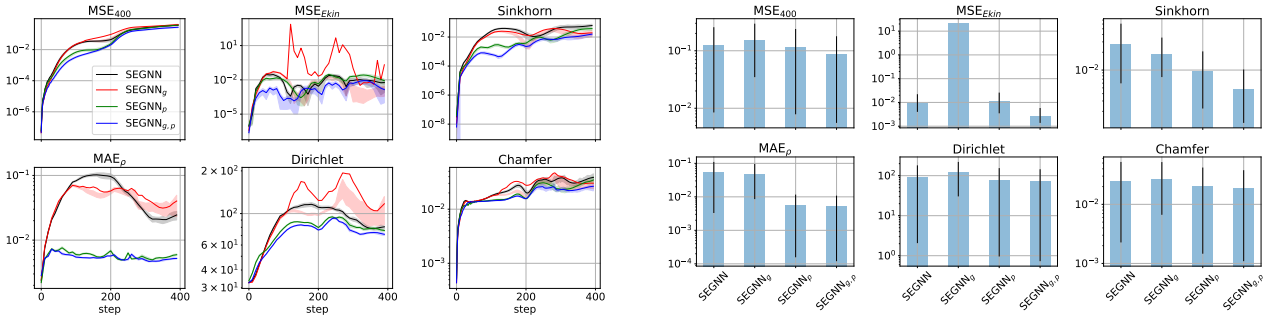

Figure 13. Ablations on DAM 2D with SEGNN-10-64 over the simulation length (left) and the average thereof (right).

G.2. Lid-Driven Cavity

We investigate the influence of the relaxation hyperparameters α and β from Eq. (5) and the number of relaxation steps/loops. The evolution of the six error measures over the 400 steps is shown on the left, and the average for each hyperparameter configuration is shown on the right. Intervals indicate the 0.25 and 0.75 quantiles over the 12 test trajectories (left) and the average of those values over the 400 steps (right).

G.2.1. LDC 2D WITH GNS

Based on Fig. 14, we choose $\alpha = 0.03$ as beyond this value, the Dirichlet energy starts increasing, indicating instabilities. In Fig. 15, we see on MSE_{400} and MSE_{Ekin} that beyond 5 iterations the accuracy drops, so we choose $l = 5$ loops. In Fig. 16, we do not see performance gains using the viscous term, so we decide not to use it.

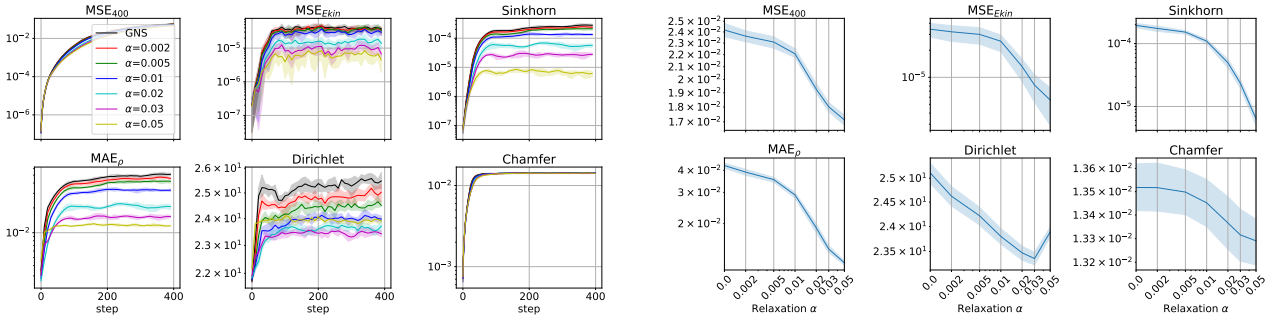

Figure 14. Ablations on LDC 2D with GNS-10-128 ($l = 1$) regarding relaxation parameter α.

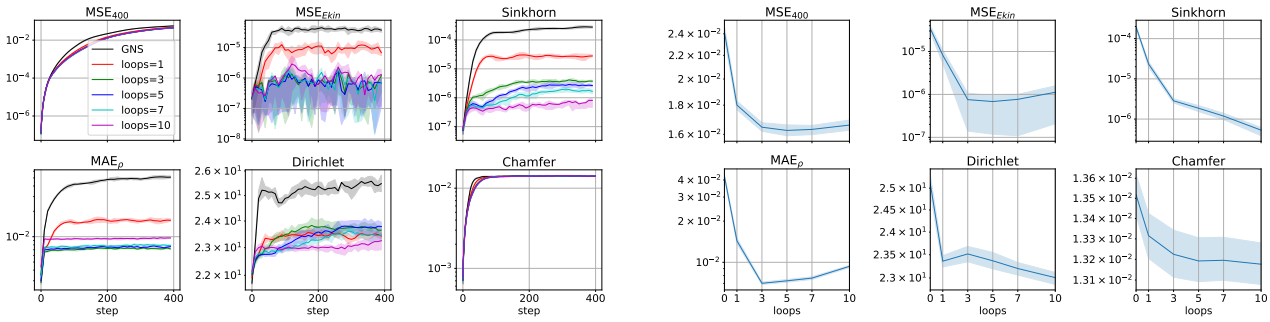

Figure 15. Ablations on LDC 2D with GNS-10-128 ($\alpha = 0.03$) regarding the number of relaxation steps/loops.

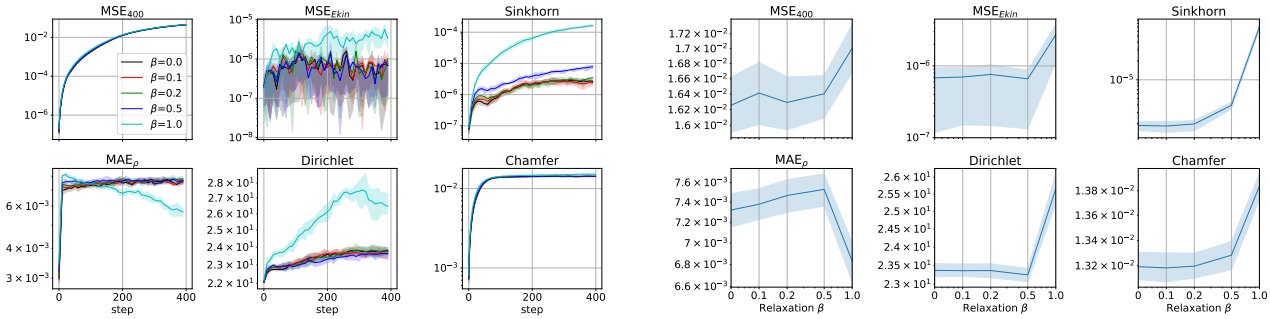

Figure 16. Ablations on LDC 2D with GNS-10-128 ($\alpha = 0.03$, $l = 5$) regarding relaxation parameter β.

G.2.2. LDC 2D WITH SEGNN

We again stress that the relaxation hyperparameters were optimized on GNS and we only ablate their influence on the performance of SEGNN. But we indeed observe similar behavior between GNS and SEGNN. We do stress the dramatic improvement in performance upon 5 and more relaxation steps visible in Fig. 18. In contrast to GNS, we do see positive impact of the viscous term on SEGNN, and would recommend using $\beta = 0.5$, see Fig. 19.

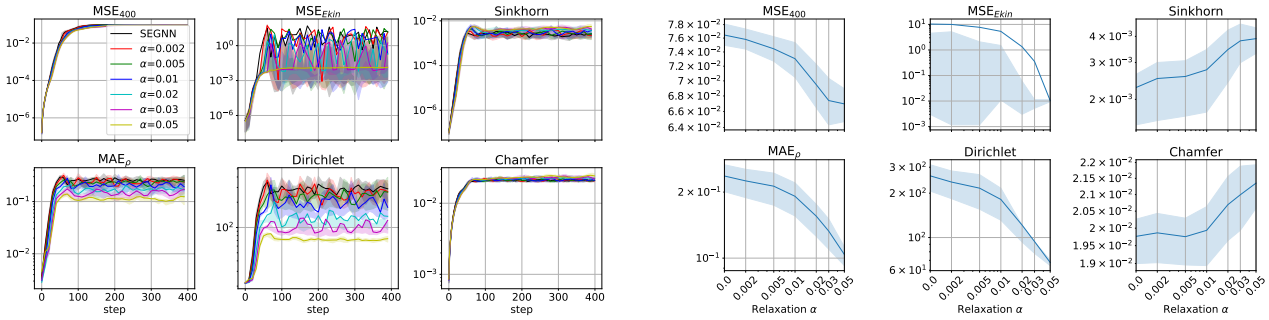

Figure 17. Ablations on LDC 2D with SEGNN-10-64 ($l = 1$) regarding relaxation parameter α.

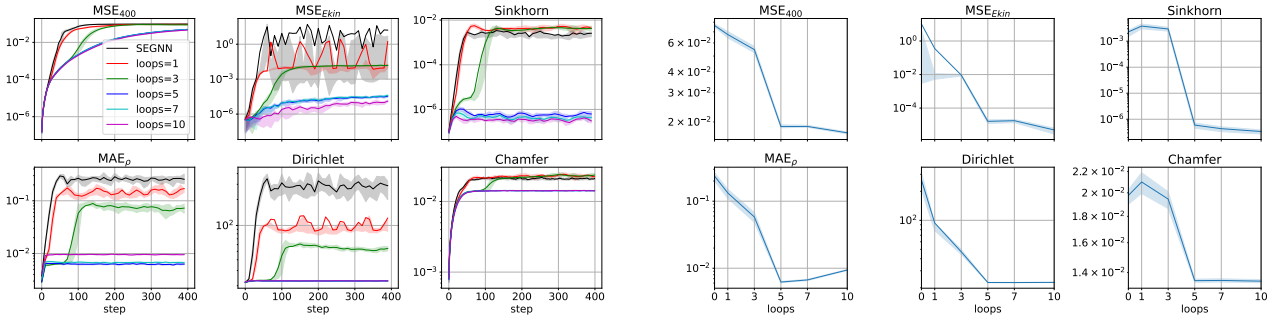

Figure 18. Ablations on LDC 2D with SEGNN-10-64 ($\alpha = 0.03$) regarding the number of relaxation steps/loops.

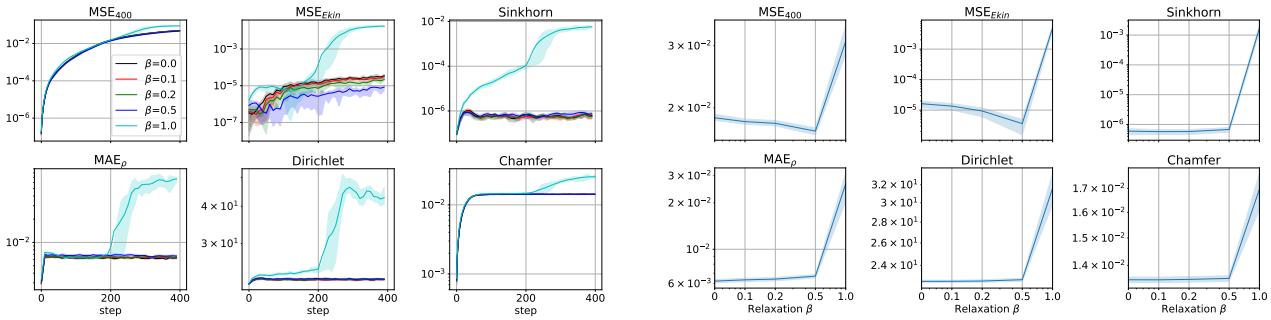

Figure 19. Ablations on LDC 2D with SEGNN-10-64 ($\alpha = 0.03$, $l = 5$) regarding relaxation parameter β.

G.2.3. LDC 3D WITH GNS

These plots agree with our choice of hyperparameters from Table 3 and show the sensitivity with respect to the relaxation parameters.

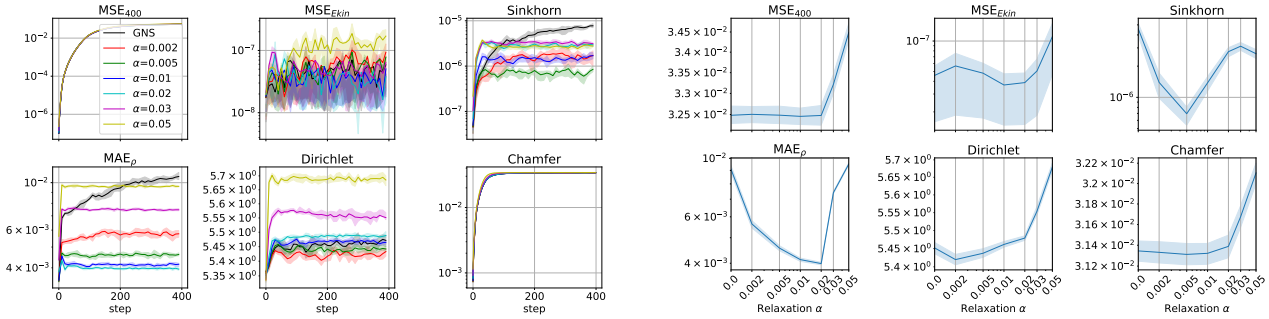

Figure 20. Ablations on LDC 3D with GNS-10-128 ($l = 1$) regarding relaxation parameter α.

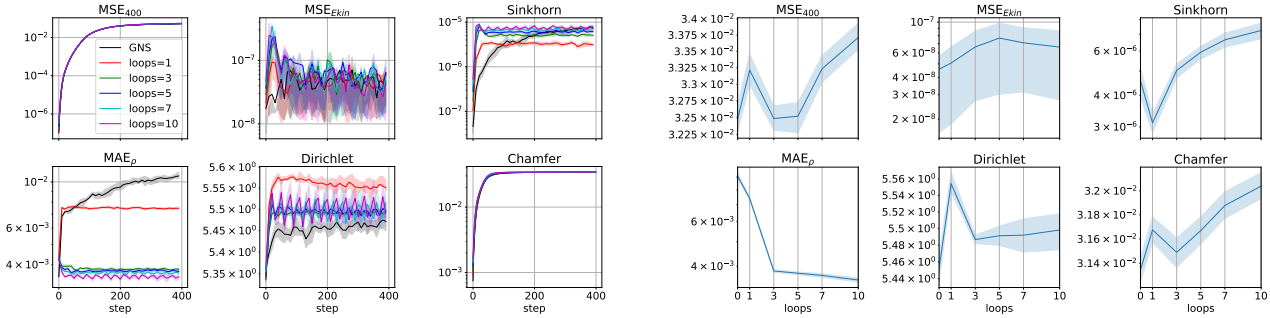

Figure 21. Ablations on LDC 3D with GNS-10-128 ($\alpha = 0.02$) regarding the number of relaxation steps/loops.

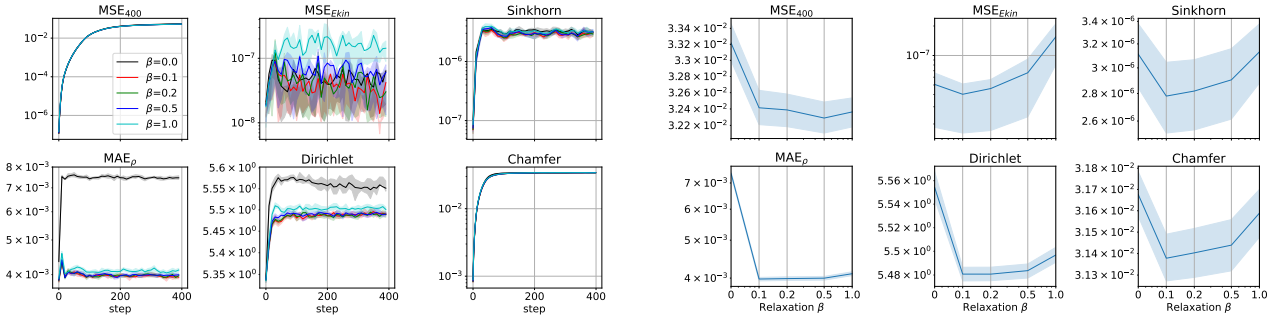

Figure 22. Ablations on LDC 3D with GNS-10-128 ($\alpha = 0.02$, $l = 1$) regarding relaxation parameter β.

G.2.4. LDC 3D WITH SEGNN

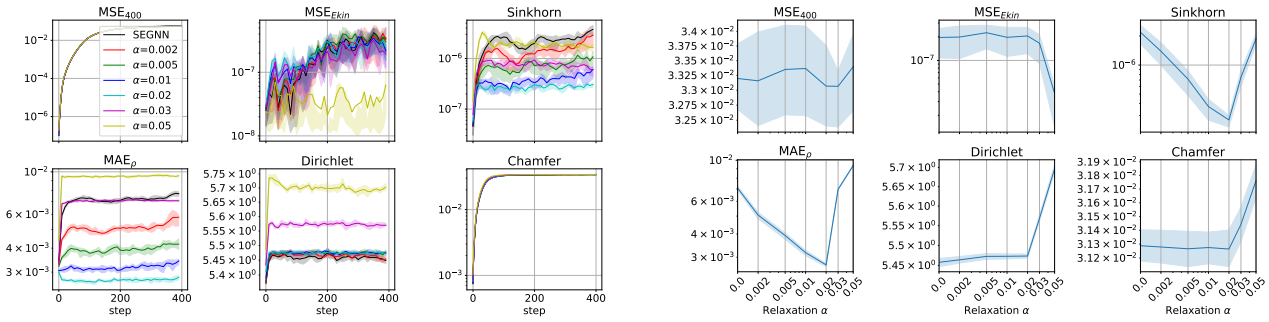

Figure 23. Ablations on LDC 3D with SEGNN-10-64 ($l = 1$) regarding relaxation parameter α.

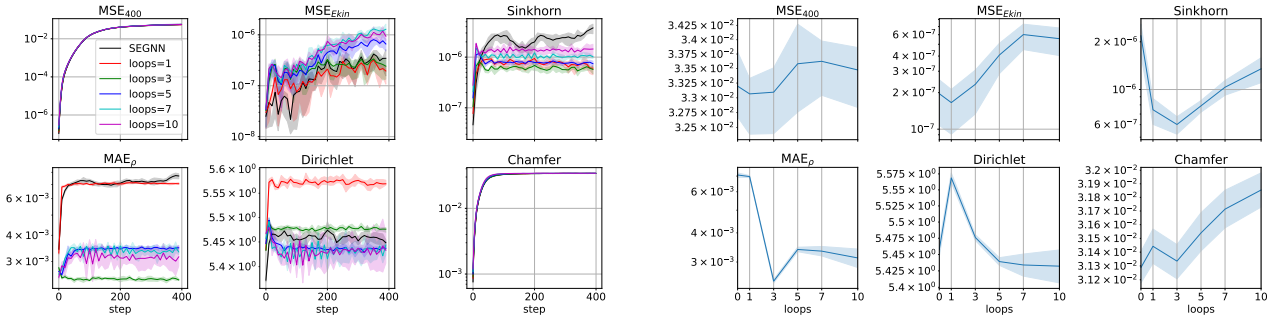

Figure 24. Ablations on LDC 3D with SEGNN-10-64 ($\alpha = 0.02$) regarding the number of relaxation steps/loops.

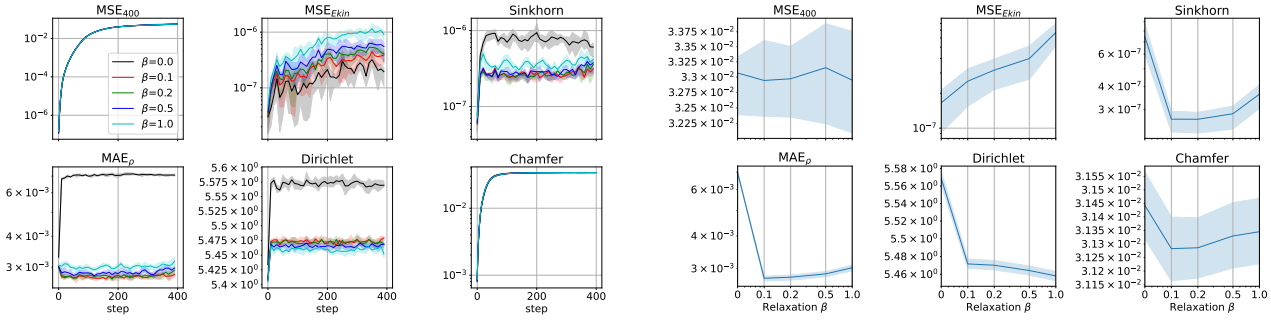

Figure 25. Ablations on LDC 3D with SEGNN-10-64 ($\alpha = 0.02$, $l = 1$) regarding relaxation parameter β.

G.3. Reverse Poiseuille Flow

We compare all variants of RPF model from the main paper with the case of not smoothing the external force, denoted $\square_{g_{raw}}$. The main message with regard to excluding the external force from the training target (all methods with $\square_g$) is that not smoothing the force function when it has discontinuities leads to highly unstable models, see MSE_{Ekin} in Figs. 27 and 28. It is probably a matter of too few test trajectories that we do not observe such blow-ups in Figs. 26 and 29.

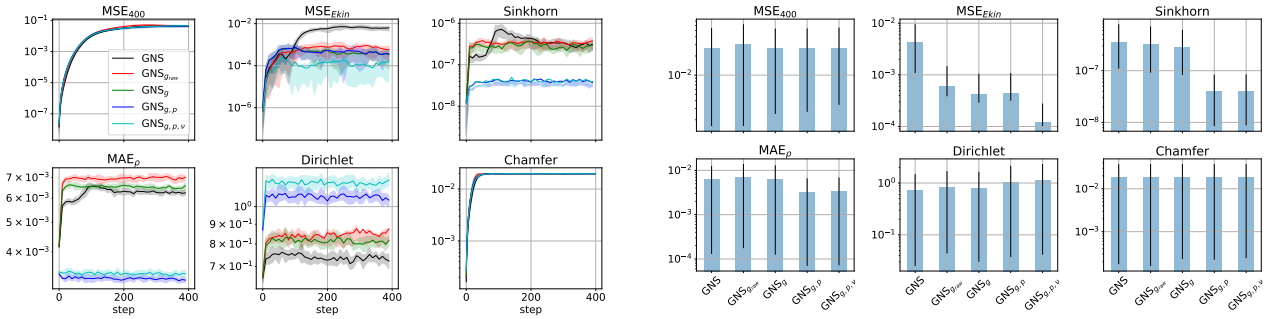

Figure 26. Ablations on RPF 2D with GNS-10-128 over the simulation length (left) and the average thereof (right).

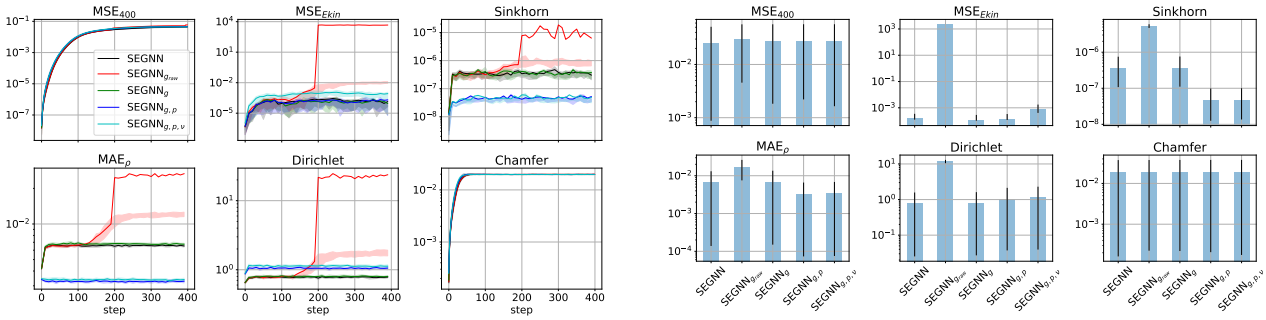

Figure 27. Ablations on RPF 2D with SEGNN-10-64 over the simulation length (left) and the average thereof (right).

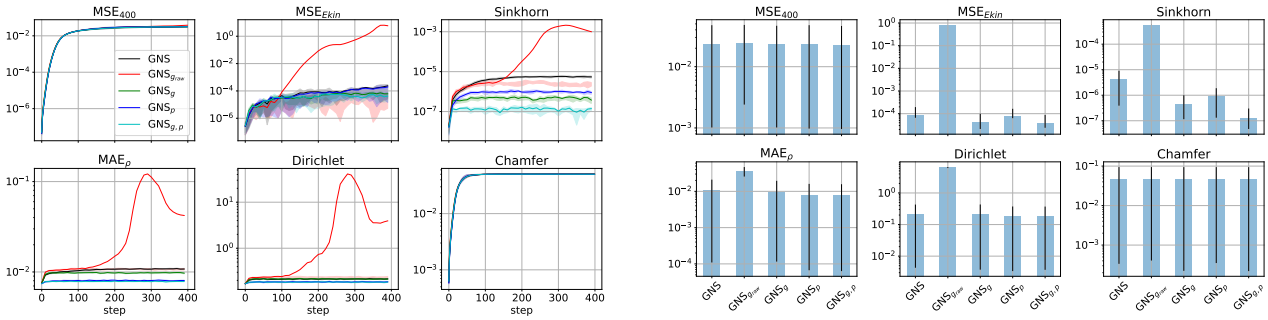

Figure 28. Ablations on RPF 3D with GNS-10-128 over the simulation length (left) and the average thereof (right).

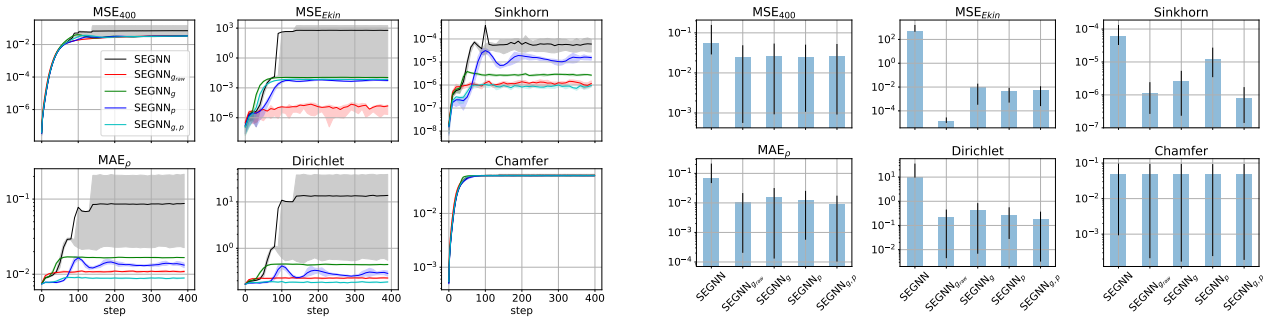

Figure 29. Ablations on RPF 3D with SEGNN-10-64 over the simulation length (left) and the average thereof (right).

H. Training with Relaxations

We also explored to idea of incorporating the SPH relaxation during training, hoping that the learned model can be regularized toward predicting better particle distributions, which could make the SPH relaxation during inference unnecessary. We explored two degrees of freedom when training a GNS-10-128 model on the 2D LDC dataset: (a) dependence on the relaxation parameter α, and (b) performance when trained with relaxation but evaluated with or without it.

Basic setup. We remind the reader that according to Table 3, the optimal relaxation parameters on 2D LDC are $\alpha = 0.03$ and 5 relaxation steps, but from the ablation in Fig. 14, we see that even one relaxation step significantly improves the dynamics. Thus, for simplicity, we use $\alpha = 0.03$ with 1 relaxation step for our training with relaxation. We implemented this training scheme by adding the relaxation to every forward call of the model, i.e. when pushforward is applied, the relaxation is executed at every pushforward step.

Training with "negative" relaxation. One highly appealing idea is to train the model with what we call "negative" relaxation, i.e. flipping the sign of the relaxation term by setting α to a negative value, by which the model would learn to over-correct unfavorable distributions. However, the results for $\alpha < 0$ in Fig. 30 are rather discouraging.

Training and inference with relaxation. Similar to subtracting the external force from the learning target, which we discussed in length and seems very useful, we investigated how the model would perform when it can predict an even worse particle distribution, which is then corrected through a relaxation both during training and inference, see $\alpha > 0$ in Fig. 31. But also here, we get worse results than only applying relaxation during inference. In addition, training with relaxation requires separate retraining until α is tuned, which is not the case with our inference time relaxation.

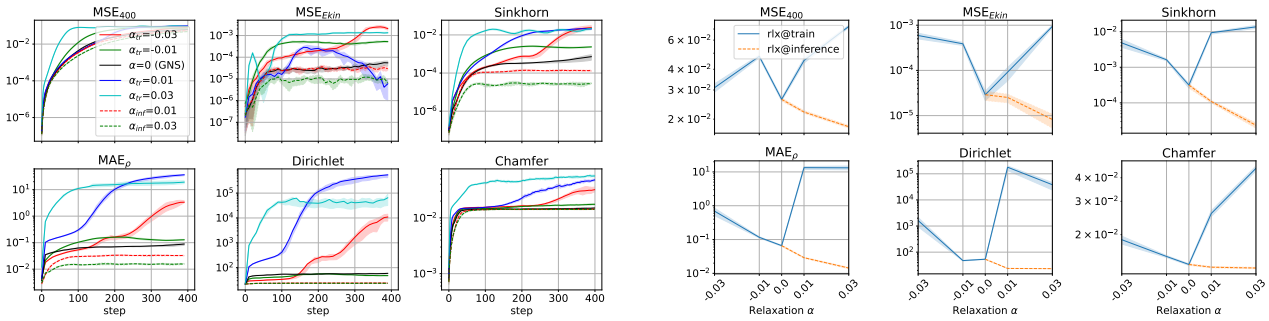

Figure 30. GNS-10-128 trained on 2D LDC with relaxation, and but evaluated **without** relaxation. We denote with α_{tr} that the model has experienced relaxation only during training and with α_{inf} only during inference. Metrics over the simulation length (left) and the average thereof (right).

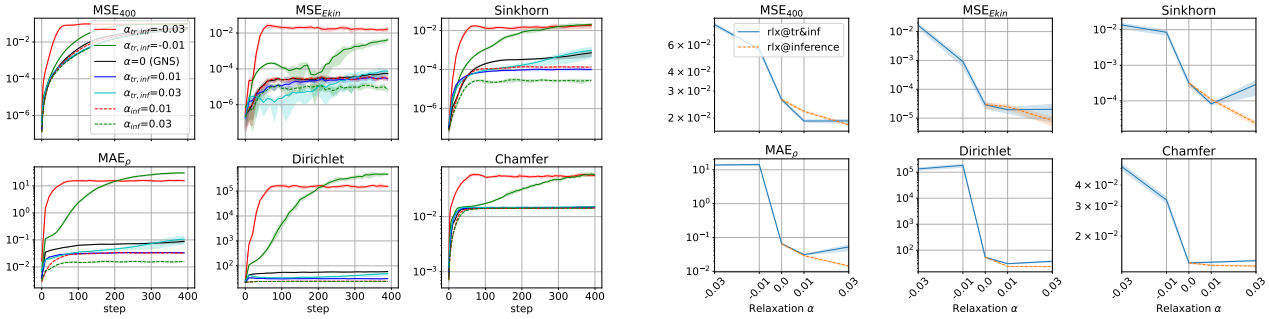

Figure 31. GNS-10-128 trained on 2D LDC with relaxation, and also evaluated **with** relaxation. We denote with $\alpha_{\text{tr,inf}}$ that the model has experienced relaxation both during training and inference and with α_{inf} only during inference. Metrics over the simulation length (left) and the average thereof (right).