



This is the accepted manuscript made available via CHORUS. The article has been published as:

Physics-informed machine learning with smoothed particle hydrodynamics: Hierarchy of reduced Lagrangian models of turbulence

Michael Woodward, Yifeng Tian, Criston Hyett, Chris Fryer, Mikhail Stepanov, Daniel Livescu, and Michael Chertkov

Phys. Rev. Fluids **8**, 054602 — Published 5 May 2023

DOI: [10.1103/PhysRevFluids.8.054602](https://doi.org/10.1103/PhysRevFluids.8.054602)

Physics Informed Machine Learning with Smoothed Particle Hydrodynamics: Hierarchy of Reduced Lagrangian Models of Turbulence

Michael Woodward^{1,2,3}, Yifeng Tian³, Criston Hyett^{1,2,3}, Chris Fryer³,
Mikhail Stepanov^{1,2}, Daniel Livescu³, and Michael Chertkov^{1,2}

¹*Graduate Interdisciplinary Program in Applied Mathematics, UArizona, Tucson, AZ 85721*

²*Department of Mathematics, UArizona, Tucson, AZ 85721 and*

³*Computer, Computational and Statistical Sciences Division, LANL, Los Alamos, NM 87544*

(*{mwoodward,chertkov}@math.arizona.edu)

(Dated: April 14, 2023)

Building efficient, accurate and generalizable reduced order models of developed turbulence remains a major challenge. This manuscript approaches this problem by developing a hierarchy of parameterized reduced Lagrangian models for turbulent flows, and investigates the effects of enforcing physical structure through Smoothed Particle Hydrodynamics (SPH) versus relying on neural networks (NN)s as universal function approximators. Starting from Neural Network (NN) parameterizations of a Lagrangian acceleration operator, this hierarchy of models gradually incorporates a weakly compressible and parameterized SPH framework, which enforces physical symmetries, such as Galilean, rotational and translational invariances. Within this hierarchy, two new parameterized smoothing kernels are developed in order to increase the flexibility of the learn-able SPH simulators. For each model we experiment with different loss functions which are minimized using gradient based optimization, where efficient computations of gradients are obtained by using Automatic Differentiation (AD) and Sensitivity Analysis (SA). Each model within the hierarchy is trained on two data sets associated with weakly compressible Homogeneous Isotropic Turbulence (HIT): (1) a validation set using weakly compressible SPH; and (2) a high fidelity set from Direct Numerical Simulations (DNS). Numerical evidence shows that encoding more SPH structure improves generalizability to different turbulent Mach numbers and time shifts, and that including the novel parameterized smoothing kernels improves the accuracy of SPH at the resolved scales.

I. INTRODUCTION

Understanding and predicting turbulent flows is crucial for many engineering and scientific fields, and remains a great unresolved challenge of classical physics [1]. Turbulent flows are characterized by strong coupling across a broad range of scales and obtaining accurate solutions over all relevant scales currently requires computationally intensive numerical methods, and is often prohibitive in applications [2]. This motivates the development of reducing the computational cost by only simulating large scale structures instead of the full set of relevant scales [3]. Many data-driven techniques have emerged (or matured) over the last decades, such as the Proper Orthogonal Decomposition [4], Dynamics Mode Decomposition [5, 6], Mori-Zwanzig [7–9], and also many other Machine Learning for Turbulence techniques [10–15]. The main challenge of developing reduced models for turbulence, is in generalizing to flows not seen in training. This motivates our focus in this work on developing *Reduced Lagrangian based Simulators* which encode physical conservation laws independent of the resolution using Smoothed Particle Hydrodynamics (SPH).

SPH [16–18] has been widely applied to weakly- and strongly compressible turbulence in astrophysics and many engineering applications [19]. It is one of a small set of approaches based on a Lagrangian construction: the fluid quantities follow the flow using particles as opposed to the Eulerian approach which computes flow quantities at fixed locations on a computational mesh. This mesh-free, Lagrangian approximation of Navier-Stokes (NS) is appealing because it naturally unmask correlations at the resolved scale from sweeping by larger scale eddies [20, 21] making SPH use-

ful for understanding advection dominated flows, mixing and dispersion, and turbulent transport. One of the main advantages of SPH is that the conservation of mass, energy, and momentum can be enforced within a discrete formulation, ensuring conservation independent of the resolution. Furthermore, the mesh-free nature of SPH is advantageous for highly compressible flows as the Lagrangian particles naturally resolve the variable density regions. Recently [22, 23], SPH has been connected to Large Eddy Simulations (LES) as a method to coarse grain the Navier-Stokes equations. Furthermore, developing approximation-optimal SPH models (and simulators) for turbulent flows is an ongoing area of research [24], to which this paper contributes by developing two new parameterized smoothing kernels and a physics informed machine learning framework for estimating the parameters of a weakly-compressible SPH formulation fit to Direct Numerical Simulation (DNS) data.

Numerical simulators were recently blended with modern machine learning tools [25–35] out of which a promising field, coined Physics Informed Machine Learning (PIML), is emerging (and being re-discovered [36]). As set, some eight years ago at the first Los Alamos National Laboratory workshop with this name [37], PIML was meant to pivot the mixed community of machine learning researchers on the one hand and scientists and engineers on the other, to discover physical phenomena/models from data. Early work incorporating scientific domain knowledge (e.g. from physics in the form of differential equations) and computational scheme within machine learning algorithms dates back to the 1990s [36]. However, in the context of modern machine learning and specifically deep learning, interest in this area has been revived [38–41]. In part, this is due to the increased computational

power afforded by parallelism across both Central, Vector or Tensor Processing Units, along with notable achievements across disciplines (such as scientific applications [42], data-compression algorithms [43], computer vision [44], and natural language processing [45]). The main computational utilities and strategies of the Physics Informed Machine Learning (PIML) includes embedding physical structure into learn-able models, applying Neural Networks (NNs) as function approximators [46], differential programming using automatic differentiation [40, 47], and optimization tools to minimize a loss function.

In this manuscript we focus specifically on building PIML scheme(s) which take advantage of the Lagrangian formulation. Other works have pursued similar directions; one of the earliest contributions to this field was made in [41] where SPH related models were used alongside regression forests (a classical machine learning technique). In [48] evolutionary algorithms were applied to optimize parameters in SPH flows. Differentiable programming techniques were utilized in [27] for Lagrangian robotic control of flows and in [33] where a continuous Convolutional NN approach was developed. NNs were used in [35] to train a closure model for Lagrangian dynamics of the velocity gradient tensor in homogeneous isotropic turbulence on the Direct Numerical Simulation (DNS) data. It was shown in [49–52] that skillful embedding of NN helps to improve Particle Image Velocimetry techniques to map a Lagrangian representation of the flow into its Eulerian counterpart. Most recent approach of our team [23], which is most related to this work, consisted in developing a fully differentiable, NN-based and Lagrangian Large Eddy Simulator trained on the mix of Eulerian and Lagrangian high-fidelity DNS data. It was shown in [23] that the simulator is capable to fit and generalize with respect to Mach numbers, delayed times and advanced turbulence statistics.

We continue the thread of [23] and develop a hierarchy of learn-able, NN-enforced, Lagrangian simulators to examine reduced order physics at coarse grained scales within the inertial range of homogeneous isotropic turbulent flows. However, in this work we focus on constraining the models using the SPH framework. Although including SPH structure will enforce physical constraints, a priori it is not known if this will improve its ability to generalize to different turbulent flows at these scales, since enforcing constraints decreases the expressiveness as compared to the NN based models. Thus, to explore these effects, a hierarchy of parameterized Lagrangian and SPH based fluid models are developed that gradually includes more of the SPH framework, ranging from a Neural Ordinary Differential Equation (ODE) based model [39] to a weakly compressible parameterized SPH formulation. Each model is trained on two sets of the ground truth data: (a) a synthetic weakly compressible SPH simulator, and (b) Eulerian and Lagrangian data from a high-fidelity DNS (similar to the one used in [23]). Given our focus in this work is on resolving only the large scale structures within the inertial range, the ground truth data is properly coarse-grained, where each model is trained on three different resolutions ($N = 12^3, 16^3, 20^3$). An efficient gradient descent is developed using modern optimizers (e.g Adam by [53]), and mix-

ing automatic differentiation (both forward and reverse mode) with the local sensitivity analyses (such as forward and adjoint based methods).

A formulation of this hierarchy along with the learning framework is given in Section IV, where a brief background of the weakly compressible SPH framework used in this work is provided in Section II. In Section V, we first validate the methodology and learning algorithm on the synthetic ground truth SPH data, in which we show the ability to recover parameters within the SPH model as well as to learn the equation of state using NNs (embedded in the SPH framework). Furthermore, in Section V, each model is trained on high-fidelity weakly compressible (i.e. low Mach number) DNS data using field and statistical based loss functions. Then a detailed analysis of each model is carried out, from which we show that adding SPH informed structure not only increases the interpretability of the models, but improves generalizability over varying Mach numbers and time scales with respect to both statistical and field based quantitative comparisons. We observe that NNs (considered as universal function approximators [46]) embedded within this structure, such as those used in approximating the Equation of State, also improves this generalizability over the standard weakly compressible SPH using the cubic smoothing kernel. Moreover, we show that the new proposed smoothing kernels (introduced in Section IV), which when included in the fully informed SPH based model, performs best at generalizing to different DNS flows.

II. SMOOTHED PARTICLE HYDRODYNAMICS

One of the most prominent particle-based Lagrangian methods for obtaining approximate numerical solutions of the equations of fluid dynamics is Smoothed Particle Hydrodynamics (SPH) [54]. Originally introduced independently by [55] and [16] for astrophysical flows, however, over the following decades, SPH has found a much wider range of applications including computer graphics, free-surface flows, fluid-structure interaction, bio-engineering, compressible flows, galaxies' formation and collapse, high velocity impacts, geological flows, magnetohydrodynamics, and turbulence [19, 24]. Below, we give a brief formulation of a weakly compressible SPH framework and in Section III we use this SPH structure as the basis of our Physics informed machine learning models.

Essentially, SPH is a discrete approximation to a continuous flow field by using a series of discrete particles as interpolation points (using an integral interpolation with smoothing kernel W). Using the SPH formalism the Partial Differential Equations (PDEs) of fluid dynamics can be approximated by a system of ODEs for each particle (indexed by i), $\forall i \in \{1, 2, \dots, N\}$:

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i, \quad (1)$$

$$\frac{d\mathbf{v}_i}{dt} = - \sum_{j \neq i}^N m_j \left(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2} + \Pi_{ij} \right) \nabla_i W_{ij} + \mathbf{f}_{ext}. \quad (2)$$

Where $W_{ij} = W(\|r_i - r_j\|, h)$, P_i , and ρ_i represents the smoothing kernel, pressure and density respectively at particle i . Π_{ij} is an artificial viscosity term [18] used to approximate the viscous terms. Density and pressure (for the weakly compressible formulation) are computed by $\rho_i = \sum_j m_j W(\|r_i - r_j\|, h)$, and $P(\rho) = \frac{c^2 \rho_0}{\gamma} \left[\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right]$.

Briefly, Eq. (2) is an approximation of Euler's equations for weakly compressible flows, with an added artificial viscosity term Π_{ij} . In this manuscript, a deterministic external forcing $\mathbf{f}_{ext}^i = (\theta_{inj}/ke(t))\mathbf{v}_i$ consistent with DNS (as seen in [56]) provides the energy injection mechanism, where ke is the kinetic energy and θ_{inj} is an energy injection rate parameter. We also utilize an artificial viscosity Π_{ij} described in Eq. (A5), which approximates in aggregate the contributions from the bulk and shear viscosity (α), a Nueman-Richtmyer viscosity for handling shocks (β) [18, 57], as well as the effective, eddy viscosity effect of turbulent advection from the under-resolved scales, i.e. scales smaller than the mean-particle distance (see Appendix A in the appendix for a more detailed discussion).

Fig. (1) shows consecutive snapshots of an exemplary multi-particle SPH flow in three dimensional space, where coloration is added for visualization purposes. We use a standard set of parameters for weakly-compressible flows (see [58]) $\alpha = 1.0$ (bulk-shear viscosity), $\beta = 2\alpha$ (Nueman-Richtmyer viscosity), $c = 10$, $\gamma = 7.0$ with energy injection rate $\theta = 0.5$, and deterministic external forcing, (see Appendix A for more details on the parameters and equations such as the equation of state and artificial viscosity). In order to validate the learning algorithm, an inverse problem is solved on "synthetic" SPH data; given a sequence of snapshots of SPH particle flows, estimate the parameters of the SPH model that best fits the SPH flow data over a predefined time scale. The results of this inverse problem can be found in Section V (and Appendix D).

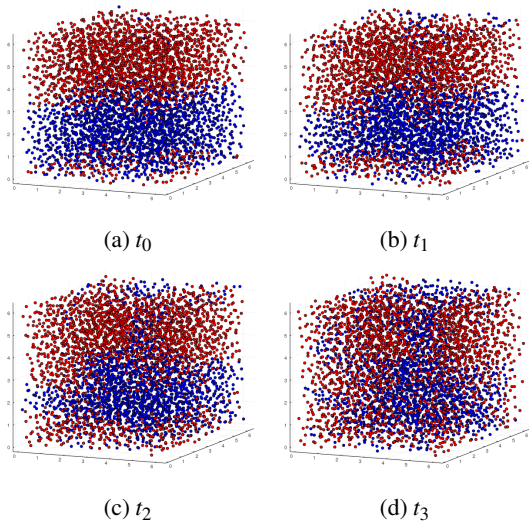


FIG. 1: SPH particles advancing in time driven by external forcing \mathbf{f}_{ext} used as training data to validate the learning algorithm, where coloration is added for visualization purposes and $t_{i+1} - t_i \approx 50\Delta t$.

III. HIERARCHY OF REDUCED LAGRANGIAN MODELS

We develop a hierarchy of parameterized Lagrangian models at coarse grained scales that gradually includes the SPH framework. The motivation for this is twofold: (1) systematically analyze the effect of including more physical structure, and (2) a priori, we do not know which of the following Lagrangian models (i.e. what level of SPH framework vs. NN parameterizations) will best fit the DNS ground truth data, as well as generalize to different flow regimes not seen in training. The NNs used in this work are Multilayer Perceptrons (MLPs) with hyperbolic tangent activation functions which serve as universal function approximators [46] and are embedded within the ODE structure evolving particles in the Lagrangian frame. It was found through hyper-parameter tuning that 2 hidden layers were sufficient for each model using a NN. Defining $\mathbf{X} := \{\mathbf{X}_i := [r_i, v_i]^T, \forall i \in \{1, \dots, N\}\}$, each of the parameterized Lagrangian models take on the general form

$$\forall i: d\mathbf{X}_i/dt = \mathcal{F}_i(\mathbf{X}(t, \theta), \theta) := (v_i, \mathbf{F}_i(\mathbf{X}, \theta))^T. \quad (3)$$

where the acceleration operator \mathbf{F} is uniquely parameterized in the following.

- **NODE:** In this least informed (and most flexible) Neural ODE [39] based Lagrangian model, the entire acceleration operator is approximated by a NN, with the exception of \mathbf{f}_{ext} . Note that no pairwise interaction between particles is assumed, which increases the flexibility of this model over SPH based parameterizations along with the generic NN structure used to approximate the acceleration of particles. This model is most related to the work done by Chen et al. [39] where we make an additional modification by considering the interaction of particles to be within a local cloud (using a cell linked list algorithm [59]). We assume that velocities, $v_i(t)$, and coordinates, $r_i(t)$, of N particles evolve in time according to

$$\frac{dv_i}{dt} = \mathbf{NN}_\theta(\eta_r(r_{ij}), \eta_v(v_{ij}) | \forall j: \|r_{ij}\| \leq 2h) + \mathbf{f}_{ext}(\theta_{inj}), \quad (4)$$

where $r_{ij} = r_i - r_j$, $v_{ij} = v_i - v_j$, $\eta_r(r_{ij})$, and $\eta_v(v_{ij})$ are min-max normalizations, $\mathbf{NN}_\theta: \mathbb{R}^{2dm} \rightarrow \mathbb{R}^l \rightarrow \mathbb{R}^l \rightarrow \mathbb{R}^d$ ($d = 2, 3$ is the space dimension, m is the fixed number of particles that are closest to the i -th particle in each cloud, and l is the height, or number of nodes, of the hidden layer). Although this NN is approximating a function which is interpretable (acceleration), the individual parameters of the NN are not. With some tuning, it was found that $m \in \{20, 21, \dots, 30\}$ and $l \in \{5, 6, \dots, 12\}$ generally produce the best fit. θ and θ_{inj} are the trainable parameters.

- **NN summand:** In the direction of including more of the SPH based physical structure, pairwise interaction is assumed represented via a sum over the i -th particle neighborhood (again using a cell linked list algorithm), where the summand term is approximated by a NN. Here, Eq. (2) is modeled by the Lagrangian based ODE

$$\frac{dv_i}{dt} = \sum_j \mathbf{NN}_\theta(\eta_r(r_{ij}), \eta_v(v_{ij})) + \mathbf{f}_{ext}(\theta_{inj}), \quad (5)$$

where $\mathbf{NN}_\theta : \mathbb{R}^{2d} \rightarrow \mathbb{R}^l \rightarrow \mathbb{R}^l \rightarrow \mathbb{R}^d$, $\eta_r(\mathbf{r}_{ij})$, and $\eta_v(\mathbf{v}_{ij})$ are min-max normalizations. With some tuning, $l \in \{5, 6, \dots\}$ generally produce the best fit. θ and θ_{inj} are the trainable parameters.

• **Rotationally Invariant NN:** In this formulation, built on the top of the NN summand, we use a neural network of the form $\mathbf{NN}_\theta : \mathbb{R}^4 \rightarrow \mathbb{R}^l \rightarrow \mathbb{R}^l \rightarrow \mathbb{R}$ to approximate the pair-wise part of the acceleration term in Eq. (2), where the rotational invariance is hard coded by construction (about a rotationally invariant basis expansion using the difference vector \mathbf{r}_{ij})

$$\frac{d\mathbf{v}_i}{dt} = \sum_j^N \mathbf{NN}_\theta \left(\frac{P_i}{\rho_i^2}, \frac{P_j}{\rho_j^2}, \mathbf{r}_{ij} \cdot \mathbf{v}_{ij}, \|\mathbf{r}_{ij}\|_2 \right) \mathbf{r}_{ij} \quad (6)$$

$$+ \mathbf{f}_{ext}(\theta_{inj}).$$

With some tuning, $l \in \{5, 6, \dots\}$ generally produce the best fit. θ and θ_{inj} are the trainable parameters.

• **∇P -NN:** In this model, we embed a neural network \mathbf{NN} within the SPH framework to approximate the gradient of pressure contribution (i.e ∇P - term in SPH Eq. (2)) and explicitly include the artificial viscosity term Π :

$$\frac{d\mathbf{v}_i}{dt} = - \sum_j^N m_j \left(\mathbf{NN}_\theta(\mathbf{r}_{ij}) + \Pi_{ij} \right) \nabla W_{ij} \quad (7)$$

$$+ \mathbf{f}_{ext}(\theta_{inj}),$$

where $\mathbf{NN}_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^l \rightarrow \mathbb{R}^l \rightarrow \mathbb{R}$. $l \in \{5, 6, \dots\}$ generally produce the best fit. θ , α , β (from $\Pi(\alpha, \beta)$) and θ_{inj} are the trainable parameters.

• **EoS NN:** Approximating the Equation of State (EoS) with a NN in the weakly compressible SPH formulation:

$$\frac{d\mathbf{v}_i}{dt} = - \sum_j^N m_j \left(\frac{P_{nn\theta}(\rho_i)}{\rho_i^2} + \frac{P_{nn\theta}(\rho_j)}{\rho_j^2} + \Pi_{ij} \right) \nabla W_{ij} \quad (8)$$

$$+ \mathbf{f}_{ext}(\theta_{inj}).$$

where $P_{nn\theta}(\rho) : \mathbb{R} \rightarrow \mathbb{R}^l \rightarrow \mathbb{R}^l \rightarrow \mathbb{R}$. $l \in \{8, 9, \dots, 12\}$ generally produce the best fit. θ , α , β (from $\Pi(\alpha, \beta)$) and θ_{inj} are the trainable parameters.

• **SPH-Informed: Fixed smoothing Kernel** In this formulation, the entire weakly compressible SPH structure is used, and the physically interpretable parameters $\alpha, \beta, \gamma, c, p_0, \theta_{inj}$ from $P(\rho)(c, \gamma, p_0)$ and $\Pi(\alpha, \beta)$ are learned

$$\frac{d\mathbf{v}_i}{dt} = - \sum_j^N m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij} \right) \nabla W_{ij} \quad (9)$$

$$+ \mathbf{f}_{ext}(\theta_{inj}).$$

• **SPH-informed: Including Parameterized W** In this formulation, the entire weakly compressible SPH structure is used along with a novel parameterized smoothing kernel (described below), and the physically interpretable parameters $\alpha, \beta, \gamma, c, p_0, a, b, \theta_{inj}$ from $P(\rho)(c, \gamma, p_0)$, $\Pi(\alpha, \beta)$, and $W(a, b)$ are learned (a, b are parameters defined below for a parameterized smoothing kernel).

$$\frac{d\mathbf{v}_i}{dt} = - \sum_j^N m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij} \right) \nabla W_{ij}(a, b) \quad (10)$$

$$+ \mathbf{f}_{ext}(\theta_{inj}).$$

Let us emphasize that, as more of the SPH based structure is added into the learning algorithm, the learned models become more interpretable; i.e. the learned parameters are associated with the actual physical quantities.

The choice of smoothing kernels is important, and effects the consistency and accuracy of results [54], where bell-shaped, symmetric, monotonic kernels are the most popular [60], however there is disagreement on the best smoothing kernels to use [61]. In this work, we introduce two new smoothing kernels to increase the flexibility of the SPH model as well as to allow the optimization framework to "discover" the best shaped kernel for our application (see Appendix A for more details). Our parameterized smoothing kernel of the first type is

$$W_1(r; h, a, b) = \begin{cases} \sigma_1(a, b)(1 - (r/(2h))^a)^b, & 0 \leq r < 2h \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

Here the parameters a, b control the shape of the kernel, h sets the spatial scale of the kernel, and constant, σ , dependent on the parameters a, b and the problem dimensionality, is chosen to guarantee normalization, $\int d\mathbf{r} W_1(r; h, a, b) = 1$, thus

$$\sigma_1(a, b) = \frac{3\Gamma(b + 3/a + 1)}{32\pi h^3 \Gamma(b + 1) \Gamma(3/a + 1)}.$$

Our parameterized smoothing kernel of the second type, which is introduced to make the second derivative at the origin, $r = 0$, smooth and thus to examine the effects of the kernel smoothness on the quality of the trained models, is as follows:

$$W_2(r; h, a, b) = \frac{1}{h^D \sigma_2(a, b)} \left(\sqrt{a^2 + 1} - \sqrt{a^2 + (r/(2h))^2} \right)$$

$$\times (1 - (r/(2h))^2)^2$$

$$\times \begin{cases} (1 + b(r/(2h))^2), & 0 \leq r < 2h \\ 0, & \text{otherwise} \end{cases} ; \quad (12)$$

$$\sigma_2(a, b) = 4\pi \left((\sqrt{1 + a^2} (32(87 + 22b) + 21a^2(-48(5 + b) + a^2(-380 + 96b + 5a^2(-30 + (46 + 21a^2)b)))) / 80640 \right.$$

$$\left. + (a^4(-32 + 16a^2(-2 + b) + 7a^6b + 10a^4(-1 + 2b)) \right.$$

$$\left. (\log(a^2) - 2\log(1 + \sqrt{1 + a^2})) / 512 \right),$$

where as with the smoothing kernel of the first type σ_2 is introduced to enforce normalization of the kernel.

IV. MIXED MODE GRADIENT BASED OPTIMIZATION: EFFICIENT PARAMETER ESTIMATION

We develop a mixed mode method, mixing local Sensitivity Analysis (SA) with forward and backwards Automatic Differentiation (AD) for efficiently computing the gradients of the loss functions (discussed in Section B).

SA is a classical technique found in many applications, such as gradient-based optimization, optimal control, parameter identification, model diagnostics [62, 63]. There are other

ways that gradients can be propagated through numerical simulators, such as using differentiable programming [40], which allows for a simple and flexible implementation of gradient based learning algorithms, however this can require higher memory costs due to storing large computational graphs when computing gradients using reverse mode [39]. The two main local SA methods are the direct, or forward, method and the adjoint method (see Section B 1 d for more details). Like forward mode AD, the forward SA method is more efficient when the number of parameters is much less than the dimension of the system and the adjoint method is more efficient when the number of parameters is much larger than the dimension of the system [63]. Therefore, in regards to this work, when the number of particles N is large, the forward SA method will be more efficient for the models described above (since the NNs are relatively small compared to the dimension of the system when N is large). When differentiating functions within the local SA frameworks we mix forward mode and reverse mode AD [47, 64] (depending on the input and output dimension of each function to be differentiated, which can include NNs) for improved efficiency over the fully differentiable programming technique.

We consider loss functions of the form, $L(\mathbf{X}, \boldsymbol{\theta}) = \int_0^{t_f} \Psi(\mathbf{X}, \boldsymbol{\theta}, t) dt$, where \mathbf{X} and $\boldsymbol{\theta}$ are, respectively, the matrix of states \mathbf{X} (defined above) and the vector of parameters. Ψ is some measure of performance at time t . Since our overall goal involves learning Lagrangian and SPH based models for turbulence applications, it is the underlying statistical features and large scale field structures we want our models to learn and generalize with. Thus, two different loss functions are considered; (1) a field based loss L_f which tries to minimize the difference between the large scale structures found in the Eulerian velocity fields, and (2) a statistical based loss L_{kl} which tries to capture the small scale statistical characteristics of turbulent flows using well known single particle statistics [65]. In the experiments below, first only the L_f is used, then a combination of the L_f and L_{kl} are used with gradient descent in order to first guide the model parameters in order to reproduce large scale structures with L_f , then later refine the model parameters with respect to the small scale features inherent in the velocity increment statistics by minimizing L_{kl} .

The field based loss L_f is introduced by setting $\Psi(\mathbf{X}, \boldsymbol{\theta}, t) = \|\mathbf{V}^f(t) - \hat{\mathbf{V}}^f(t)\|^2/N_f$, where $\mathbf{V}_i^f = \sum_{j=1}^{N_f} (m_j/\rho_j) \mathbf{v}_j W_{ij}(\|\mathbf{r}_i^f - \mathbf{r}_j\|, h)$. This uses the same SPH smoothing approximation to interpolate the particle velocity onto a predefined mesh \mathbf{r}^f (with N_f grid points). The statistical based loss function L_{kl} , using a Kullback–Leibler (KL) divergence, is also introduced by setting $\Psi = \int_{-\infty}^{\infty} P_{gt}(t, \mathbf{z}_{gt}, \mathbf{x}) \log(P_{gt}(t, \mathbf{z}_{gt}, \mathbf{x})/P_{pr}(t, \mathbf{z}_{pr}(\boldsymbol{\theta}), \mathbf{x})) d\mathbf{x}$, where $\mathbf{z}_{gt} = \mathbf{z}_{gt}(t)$, and $\mathbf{z}_{pr}(\boldsymbol{\theta}) = \mathbf{z}_{pr}(\boldsymbol{\theta}, t)$ represent single particle statistical objects over time of the ground truth and predicted data, respectively. For example, we use the velocity increment, $\mathbf{z}^i(t) = (\delta u_i, \delta v_i, \delta w_i)$, where $\delta u_i(t) = u_i(t) - u_i(0)$ and \mathbf{z} ranges over all particles. Here $P(t, \mathbf{z}(t), \mathbf{x})$ is a continuous probability distribution (in \mathbf{x}) constructed from data $\mathbf{z}(t)$ using Kernel Density Estimation (KDE), to obtain smooth

and differentiable distributions from data [66]), that is $P(\tau, \mathbf{z}, \mathbf{x}) = (Nh_{kde})^{-1} \sum_{i=1}^N K((z_i - \mathbf{x})/h_{kde})$ (where h_{kde} is a smoothing parameter selected in this work as $h_{kde} = 0.9$ based on Silverman’s rule [66]).

The gradient $\partial_{\boldsymbol{\theta}} L = \int_0^{t_f} \partial_{\mathbf{X}} \Psi(\mathbf{X}, \boldsymbol{\theta}, t) d_{\boldsymbol{\theta}} \mathbf{X}(\boldsymbol{\theta}, t) + \partial_{\boldsymbol{\theta}} \Psi(\mathbf{X}, \boldsymbol{\theta}, t) dt$, is computed with the forward SA equation by simultaneously integrating the states (Eq. 3) along with the sensitivities $\mathbf{S}_i^k := d\mathbf{X}_i/d\boldsymbol{\theta}^k$ according to

$$\forall i: \quad \frac{d\mathbf{S}_i^k}{dt} = \frac{\partial \mathcal{F}_i(\mathbf{X}(t), \boldsymbol{\theta})}{\partial \mathbf{X}_i} \mathbf{S}_i^k + \frac{\partial \mathcal{F}_i(\mathbf{X}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}^k}. \quad (13)$$

Where a mixed mode AD is used to compute the derivatives of $\partial \mathcal{F}_i/\partial \boldsymbol{\theta}$ and within $\partial \mathcal{F}_i/\partial \mathbf{X}_i$ (see Appendix B for more details).

V. RESULTS: TRAINING AND EVALUATING MODELS

First, the methodology is validated on "synthetic" SPH data, by training each model in the hierarchy on the SPH data and testing their ability to interpolate and generalize. For example, in Fig. 2 we see the ability for NNs embedded within the SPH framework to learn the equation of state. Fig. 2 provides further validation of the mixed mode learning algorithm for performing parameter estimation by learning the parameterized SPH "physics-informed" from SPH data (see Appendix D such as Fig. 23 for more details). Next, we analyze the hierarchy of models trained on weakly compressible (low Mach number) Eulerian based DNS data where the models are evolved on a coarse grained scale using $N = 12^3, 16^3, 20^3$ particles.

A. DNS data

Our “ground truth” Lagrangian data are generated by tracking simultaneously multiple particles advected by velocity which we extract from the Eulerian DNS (solving the Navier-Stokes equations) of weakly compressible, thus low Mach number, stationary Homogeneous Isotropic Turbulence. Our numerical implementation of the Eulerian DNS is on a 256^3 mesh over the three dimensional box, $\Omega = [0, 2\pi]^3$. We use sixth-order compact finite differences for spatial discretization and the 4th-order Runge-Kutta scheme for time advancement, also imposing triply-periodic boundary conditions over the box. The velocity field is initialized with 3D Gaussian spectral density enforcing zero mean condition for all components.

A large-scale linear quasi-solenoidal forcing term is applied to the simulation at wavenumber $|k| < 2$ to prevent turbulence from decaying [56]. The forcing method allows the specification of the Kolmogorov scale at the onset and ensures that it remains close to the specified value. The simulations presented here have $\eta/\Delta x = 0.8$, where Δx is the grid spacing. Compared to a standard (well-resolved) spectral simulation with $\eta k_{max} = 1.5$, where k_{max} is the maximum resolved

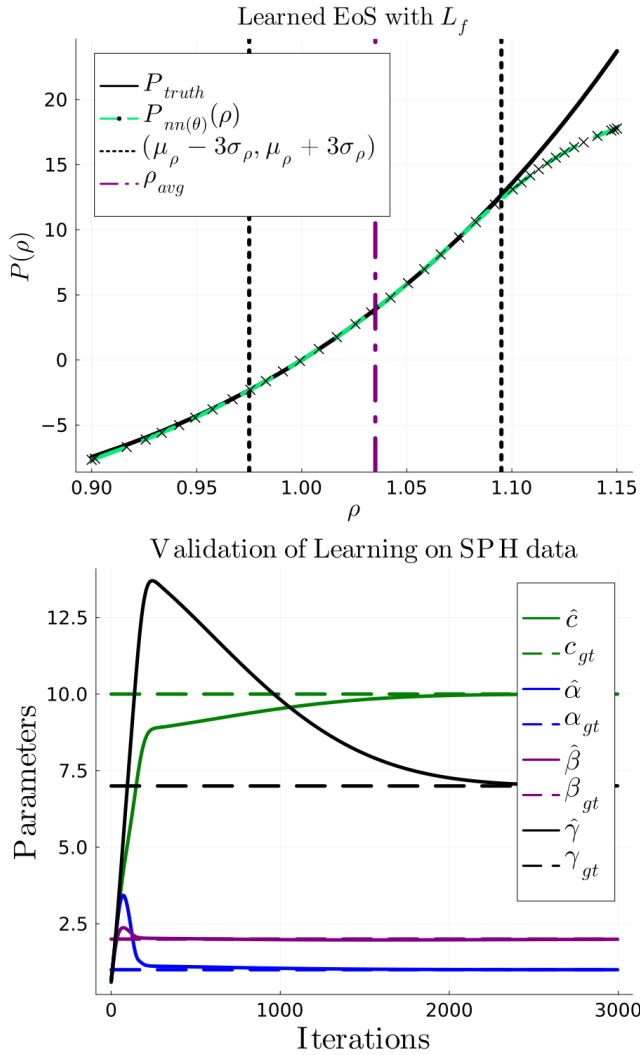


FIG. 2: Using a NN embedded within the SPH framework to approximate Eq. 8 using L_f . We see that $P(\rho)$ is well approximated, where P_{truth} is the ground truth solution and P_{nn} is the NN approximation. We note that the underlying ground truth data has 99.73% of density values within the black dotted vertical lines (within 3 standard deviations), and so the NN does well at approximating the ground truth within the data seen in training, however it fails to capture the global shape of the function. Furthermore, we see that the parameters that were used to generate the synthetic SPH training data are learned where the initial guesses are uniformly distributed about $(0,1)$.

wavenumber, which has $\eta/\Delta x = 1.5/\pi$, the contraction factor is ≈ 0.6 [56, 67] and the maximum differentiation error at the grid (Nyquist) scale is less than 3.5%. Compared to a spectral method with $\eta k_{max} = 1$, which has $\eta/\Delta x = 1/\pi$, the contraction factor is ≈ 0.4 [56, 67] and the maximum differentiation error at the Nyquist scale is less than 0.2%. The initial temperature field is set to be uniform and the initial pressure field is calculated by solving the Poisson equation. More details about the numerical method and setup can be found in Refs.

[56, 68]. The simulation is conducted until the turbulence becomes statistically stationary, which is verified based on the evolution of the kinetic energy and dissipation [56, 68].

Once a statistically-steady state of HIT is achieved, we apply a Gaussian filter to smooth the spatio-temporal Eulerian data for velocity at the resolved scale, d , and then inject the filtered flow with 16^3 non-inertial Lagrangian fluid particles. We use a Gaussian filter, which is commonly used in LES, with a filtering length scale of the order or larger than the scale d that can be resolved for the particles. In dimensionless units, where the energy containing scale, L , which is also the size of the box, is $L = 2\pi$, the smallest scale d we can resolve with this number of particles is $\pi/8$, i.e. 16 times smaller than the size of the domain.

The particles are placed in the computational domain, $[0, 2\pi]^3$, where the initial condition is set as an SPH equilibrium solution (particles are evolved according to SPH with no external forcing until particles reach an equilibrium position [69]), and then we follow trajectories of the passively advected particles for time, τ , which is of the order of (or longer) than the turbulence turnover time of an eddy of size comparable to the resolved scale, d , i.e. $\tau = O(d^{2/3}/\varepsilon^{1/3})$, where ε is the estimate of the energy flux transferred downscale within the inertial range of turbulence. Note that d is bounded from above by the size of the box, i.e. $L = 2\pi$ in the dimensionless units of our DNS setting, and from below by the Kolmogorov (viscous) scale, $\eta = O(v^{3/4}/\varepsilon^{1/4})$, where v is the (kinematic) viscosity coefficient.

In this work, we consider three turbulence cases for training and testing the model with comparable Reynolds numbers, $Re_\lambda \approx 80$ and turbulent Mach numbers, $M_t = 0.04, 0.08$, and 0.16 , as shown in Table I [23]. The Taylor Reynolds number is calculated from the turbulence Reynolds number, Re_t , using the isotropic turbulence formula $Re_\lambda = \sqrt{20/3} Re_t$ [70], where $Re_t = k_t^2/(v\varepsilon)$, with k_t the turbulent kinetic energy based on the filtered velocity. The turbulent Mach number in DNS is defined as $M_t = (2k_t)^{1/2}/c_s$. In the limit of low Mach number, for single component flows, density can be expanded as $\rho \approx \rho_0 + \rho_1$, where $\rho_1 \sim M_t^2 \rho_0$ [71], so that M_t^2 is proportional to the fluid density deviation from the uniform distribution. Note that any particle-based modeling of turbulence requires introducing, discussing, and analyzing compressibility simply because any distribution of particles translates into fluid density which is always spatially non-uniform, even if slightly. Therefore, even if we model fully incompressible turbulence, we should still introduce an effective turbulent Mach number when discussing a particle based approximation, which for the weakly compressible limit can be approximated as $M_t^2 \sim |\rho - \rho_0|/\rho_0$.

For the $M_t = 0.08$ case and over each resolution set ($N = 12^3, 16^3, 20^3$), training takes place on the order of the Kolmogorov timescale t_η . When measuring the performance of each model, generalization errors are computed over different turbulent Mach numbers as well as over different time scales (for $M_t = 0.08$) up to the eddy turn over time. Furthermore, when DNS data is used in training, the equation of state used in Eqs. 10 and 9 is $P(\rho; c, \gamma, p_0) = c^2 \rho^\gamma + p_0$, to be consistent with the DNS formulation used, where the background

TABLE I: DNS Cases for training and validation of models.

Case number	1	2	3
Turbulent Mach number M_t	0.08	0.16	0.04
Taylor Reynolds number Re_λ	80	80	80
Kolmogorov timescale t_η	2.3	1.2	4.7
Usage	training	validation	validation

pressure term p_0 is added as a correction for SPH (the pressure gradient term in the SPH framework is not invariant to changes in this background pressure term, see Appendix A).

B. Training and Evaluating Models

Parameter estimation (i.e training) is performed on each model on the order of the Kolmogorov time using the mixed mode gradient based optimization with the field based loss function (and statistical based loss Section C using velocity increment) until convergence (see Fig. 3). Once all the models are trained, they are used to make forward predictions over larger time scales as seen in Fig. 4 (on the order of the eddy turn over time) and over different turbulent Mach numbers as seen in Table I. The future state predictions are evaluated with respect to loss functions as a performance measure, and detailed statistical comparisons are given.

The shapes of the novel learned parameterized smoothing kernels are reported in Fig 5. We use the field based loss normalized with respect to the total kinetic energy from DNS as a quantitative measure comparing each model over larger time scales and different turbulent Mach numbers as seen in Figs. 6, 8, 7. The errors in translational and rotational symmetries are recorded in Table II, which shows that as more SPH based structure is included, conservation of linear and angular momenta is enforced. Furthermore, the single particle statistics, acceleration pdfs, and energy spectrum (as seen in Figs. 9, 10, 11, and 12 respectively) are used to evaluate the statistical performance of each model as external diagnostics (not used in training). The results in these figures show that the SPH informed model using the novel parameterized smoothing kernel W_2 (Eq. (10)) performs best at generalizing, with respect to the statistical and field based performance evaluations, to larger time scales and different turbulent Mach numbers, as well as enforces physical symmetries and improves interpretability over the less informed models.

Discussing in slightly more detail, Fig. 6 clearly shows that there is a gradual improvement in generalizability as more physics informed SPH based structure is included in the model. Although the most generalizable model is the fully parameterized SPH informed model using the new parameterized smoothing kernel $W_2(a, b)$, there is a close match between P_{nn} (using a NN to approximate the equation of state within the SPH framework Eq. 8) and $(\nabla P)_{nn}$ (using a NN to approximate the SPH based pressure gradient Eq. 7) and the SPH- $W_2(a, b)$ model. Thus, within the SPH-informed models, using a NN embedded within the SPH framework (such as in approximating the Equation of State and pressure gradient term) can actually improve generalizability over the unin-

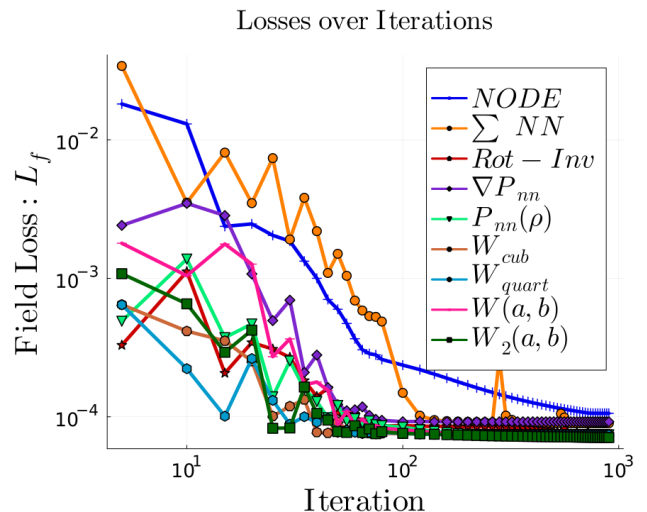


FIG. 3: Losses converging over iterations when trained on DNS up to the Kolmogorov time scale with $M_t = 0.08$.

The physics informed SPH based models achieve the lowest losses. Each model is seen to interpolate onto the DNS field data as seen in the u - component of velocity snapshots in Figure 4 within in the training window.

formed models and the standard SPH-informed model when using the classical cubic or quartic smoothing kernels. However, relying on a NN to approximate the full acceleration operator without including any conservation laws, although still being able to interpolate, does not generalize nearly as well as the SPH-informed models.

In Figs. 9 and 10, the acceleration statistics comparing smoothing kernels and each model respectively are reported for one eddy turn over time with $M_t = 0.08$ (as seen in training) and $M_t = 0.16$, and $M_t = 0.04$. In this figure, we see a clear improvement in generalization with respect to the acceleration statistics as more SPH structure is used and improving the accuracy of the SPH framework using the novel parameterized smoothing kernels. A similar trend is observed in the single particle statistics as seen in Fig. 11. However, even the SPH informed models seem to struggle to provide a close match to DNS acceleration when generalizing to $M_t = 0.04$ flow, indicating a limitation to this approach. However, we should note that the only parameter that was changed in making forward state predictions for different turbulent Mach numbers was θ_{inj} , in order to be consistent across models and with the DNS external forcing. Finally, in Fig. 12, the energy spectrum shows some of the key statistical differences in the model predictions, that the less informed models, although able to interpolate on the training window like the other models, predict more energy to be distributed in the small scales and less in the large scales as compared to DNS (and as seen in the volume plots of the u -component of velocity field Fig. 4); whereas the SPH informed models provide a closer match to the large to small scale energy cascade. For further analysis of the smoothing kernels and experiments using a combination of L_f and L_{kl} see Appendix C.

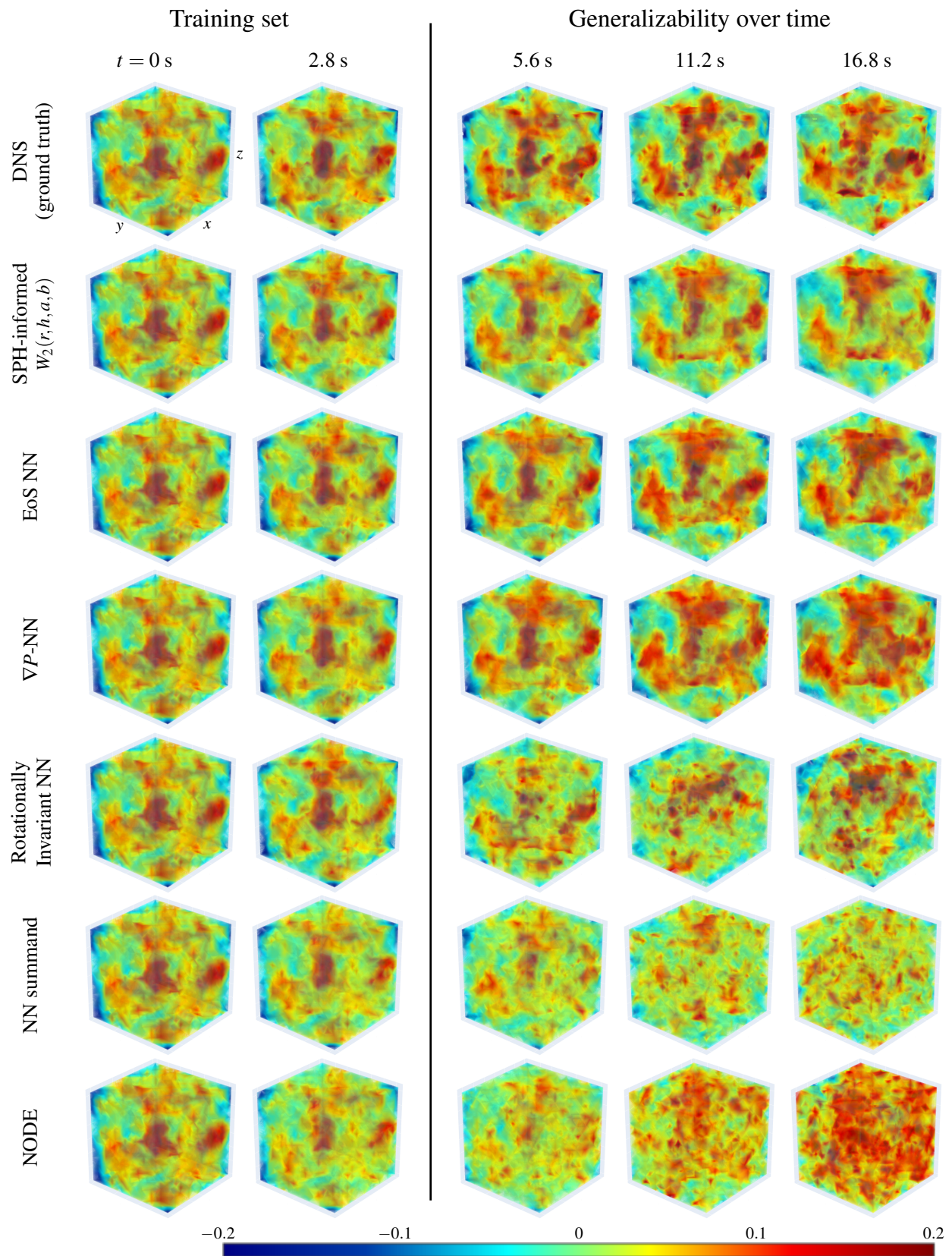


FIG. 4: Volume plots of snapshots over time comparing Eulerian u velocity component of coarse grained DNS data (at the same resolution of the models and at $M_t = 0.08$) to the predictions made with the trained models. This qualitatively shows that as more SPH structure is included, the better is the ability of the model to generalize all the way to the longest (physically relevant) time scale, t_{eddy} (which is the turnover time scale of the largest eddy of the flow) even when it is trained on the shortest relevant time scale t_η . We see the large scale structures present in the u velocity are best captured with the SPH informed models, and predictions degrade as more reliance is put on using a Neural Network to parameterize the acceleration operator [72].

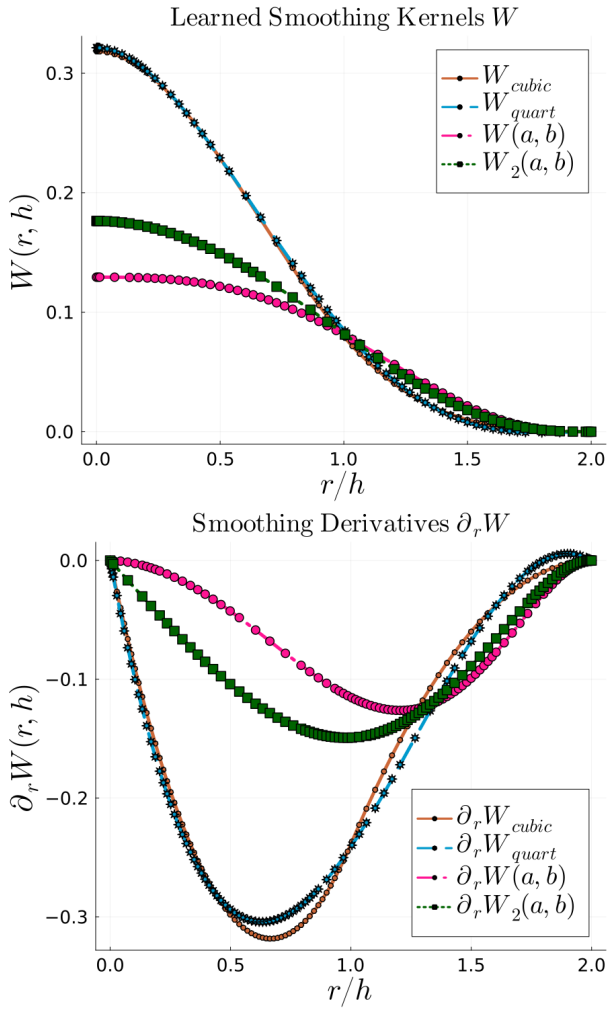
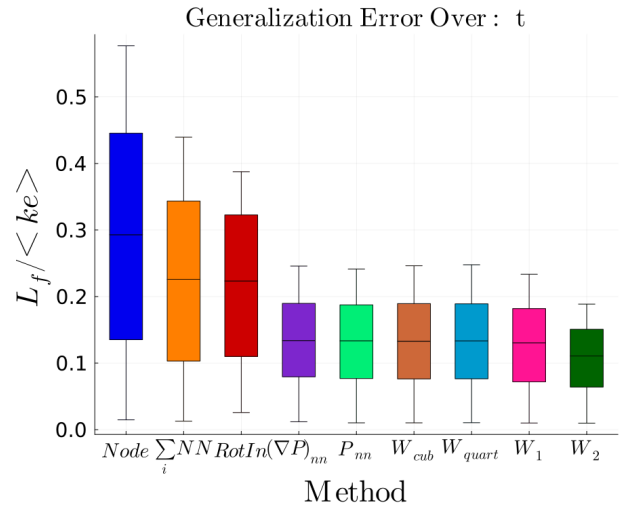


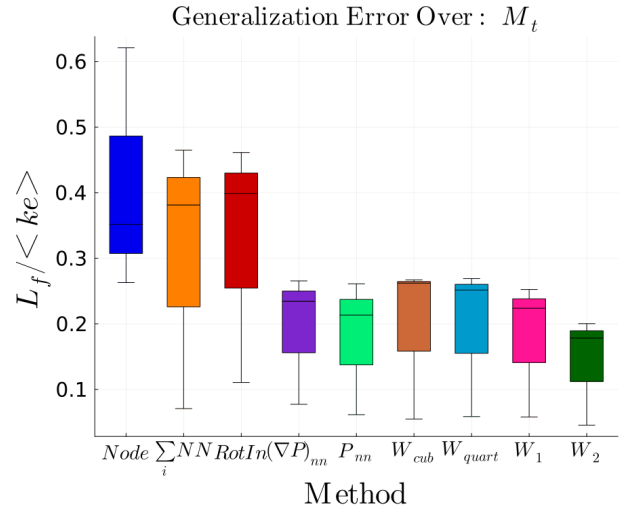
FIG. 5: Comparing learned smoothing kernels $W_1(r; h, a, b)$ and $W_2(r; h, a, b)$ with standard cubic Eq. (A1) and quartic Eq. (A2) kernels. Notice the shape of the learned parameterized smoothing kernels includes a relatively larger contribution from particles farther away and shallower gradients for nearby particles. See Table IV for the learned parameters.

TABLE II: Rotational and Translational Invariance errors in trained models

Model	Rotational	Translational
NODE	3.5×10^{-4}	4.1×10^{-4}
NN summand	3.4×10^{-5}	7.9×10^{-33}
Rot-Invariant NN	1.5×10^{-32}	3.3×10^{-32}
∇P - NN	2.6×10^{-6}	4.5×10^{-32}
EoS NN	4.1×10^{-32}	1.2×10^{-31}
SPH-informed: W_{cubic}	3.5×10^{-32}	8.5×10^{-32}
SPH-informed: $W_{quartic}$	2.1×10^{-31}	5.4×10^{-31}
SPH-informed: $W(a, b)$	2.4×10^{-32}	1.1×10^{-31}
SPH-informed: $W_2(a, b)$	3.4×10^{-32}	1.2×10^{-31}

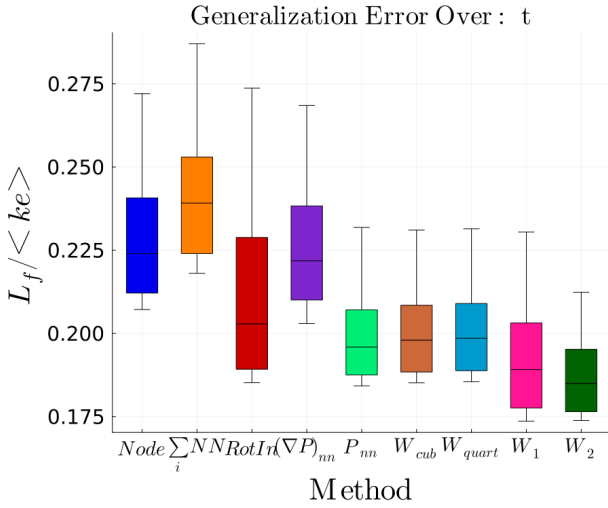


(a) Normalized L_f over time



(b) Normalized L_f over turbulent Mach numbers M_t

FIG. 6: Measuring the generalization error using the field based loss L_f normalized with respect to the total kinetic energy $\langle ke \rangle$ from DNS. (a) The generalization error over t is computed over 20 different time scales ranging from the Kolmogorov time to the eddy turn over time. (b) Generalization error over M_t is computed using 3 different turbulent Mach numbers M_t , 0.04, 0.08, and 0.16, integrated up to the eddy turn over time scale. We see that the SPH-informed model with the parameterized smoothing kernel $W_2(a, b)$ performs best at generalizing with time and turbulent Mach numbers. Furthermore, the NNs embedded within the SPH structure, namely $(\nabla P)_{nn}$ and P_{nn} , showing improvements over the standard SPH model.



(a) Coarser resolution models over time

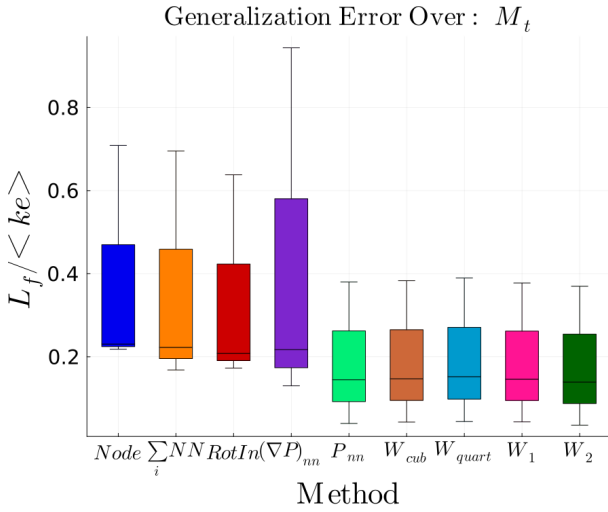
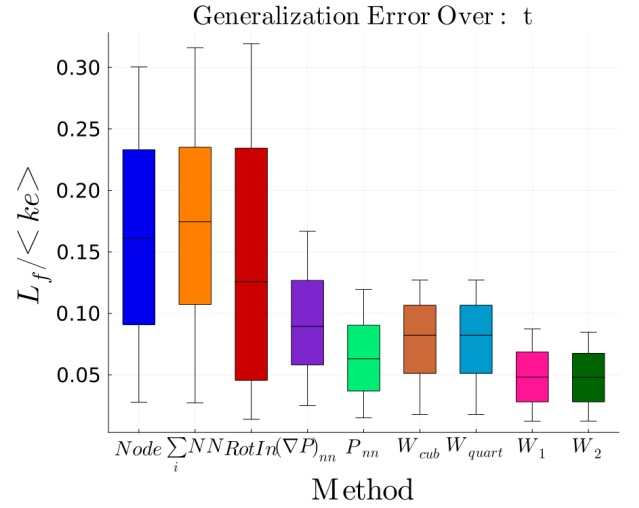
(b) Coarser resolution models over turbulent Mach numbers M_t

FIG. 7: Predictive performances of coarser resolution models with $N = 12^3$ particles. Measuring the generalization error using the field based loss L_f normalized with respect to the total kinetic energy $\langle ke \rangle$ from DNS. (a) The generalization error over t is computed over 20 different time scales ranging from the Kolmogorov to the eddy turn over time. (b) Generalization error over M_t is computed using 3 different turbulent Mach numbers M_t , 0.04, 0.08, and 0.16, integrated up to the eddy turn over time scale. We see that the SPH-informed model with the parameterized smoothing kernel $W_2(a, b)$ performs best at generalizing with respect to time and turbulent Mach numbers.



(a) Finer resolution models over time

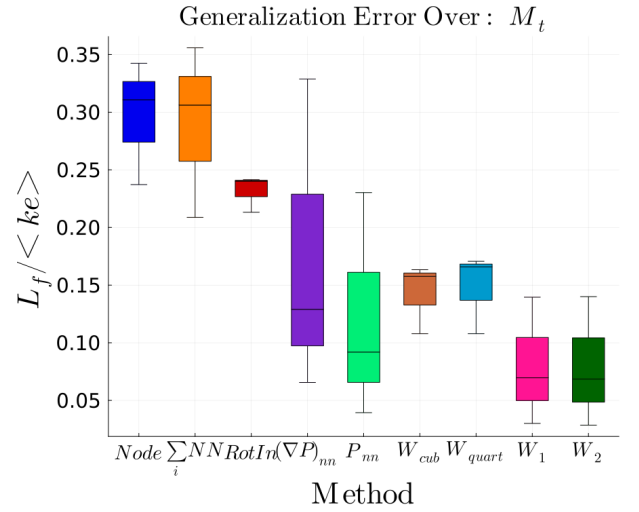
(b) Finer resolution models over turbulent Mach numbers M_t

FIG. 8: Predictive performances of finer resolution models with $N = 20^3$ particles. Measuring the generalization error using the field based loss L_f normalized with respect to the total kinetic energy $\langle ke \rangle$ from DNS. (a) The generalization error over t is computed over 20 different time scales ranging from the Kolmogorov time to the eddy turn over time. (b) Generalization error over M_t is computed using 3 different turbulent Mach numbers M_t , 0.04, 0.08, and 0.16, integrated up to the eddy turn over time scale. We see that the SPH-informed model with the parameterized smoothing kernel $W_2(a, b)$ performs best at generalizing with respect to time and turbulent Mach numbers. We see that the improvements by using the SPH-informed model with parameterized smoothing kernel $W_2(a, b)$ become greater at the finer scale resolutions. We also note that the DNS data are not fully barotropic, like in the weakly compressible SPH framework. Thus, for the training M_t , P_{nn} is more flexible and learns a better fit for equation of state than what is used in the standard SPH model. However, this model may suffer from over-fitting as seen in (b).

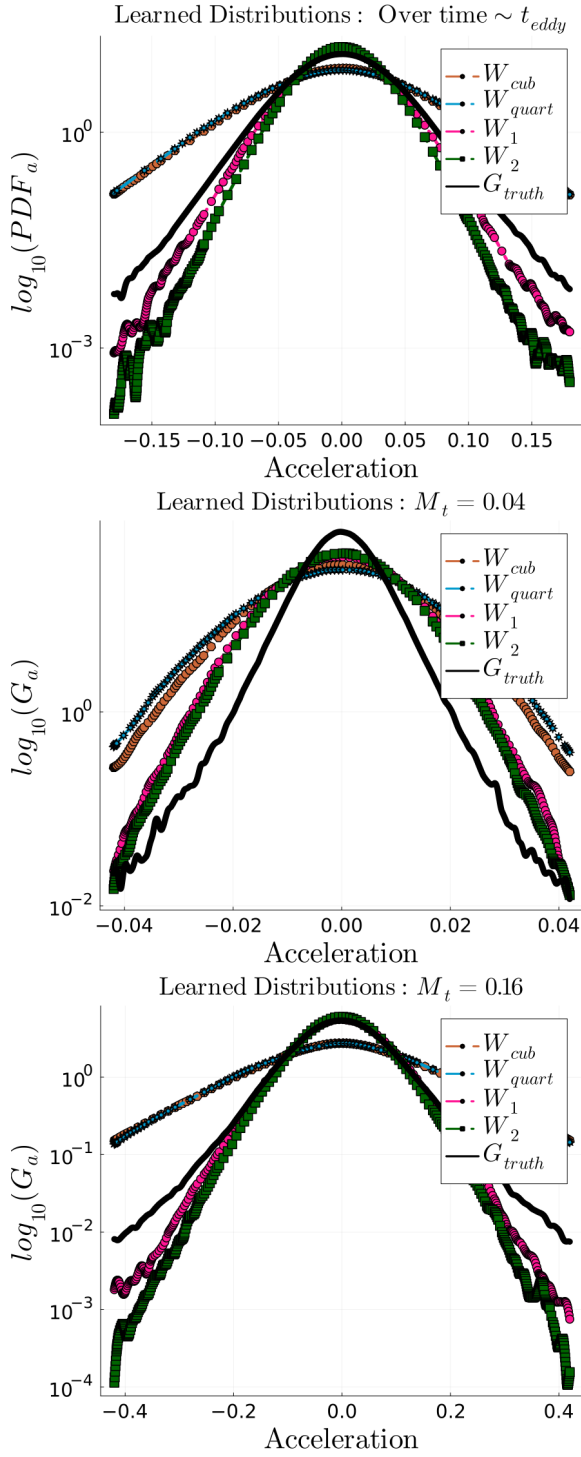


FIG. 9: Comparing acceleration statistics over time, and turbulent Mach numbers of the trained SPH based models showing the learned parameterized smoothing kernels are a closer match to DNS as compared to the cubic and quartic smoothing kernels over longer time scales as well as at different Mach numbers $M_t = 0.04$ and $M_t = 0.16$. However, each model misses the intermittent behavior seen in the long tails from DNS, and performs poorly on the $M_t = 0.04$ indicating limitations of this approach.

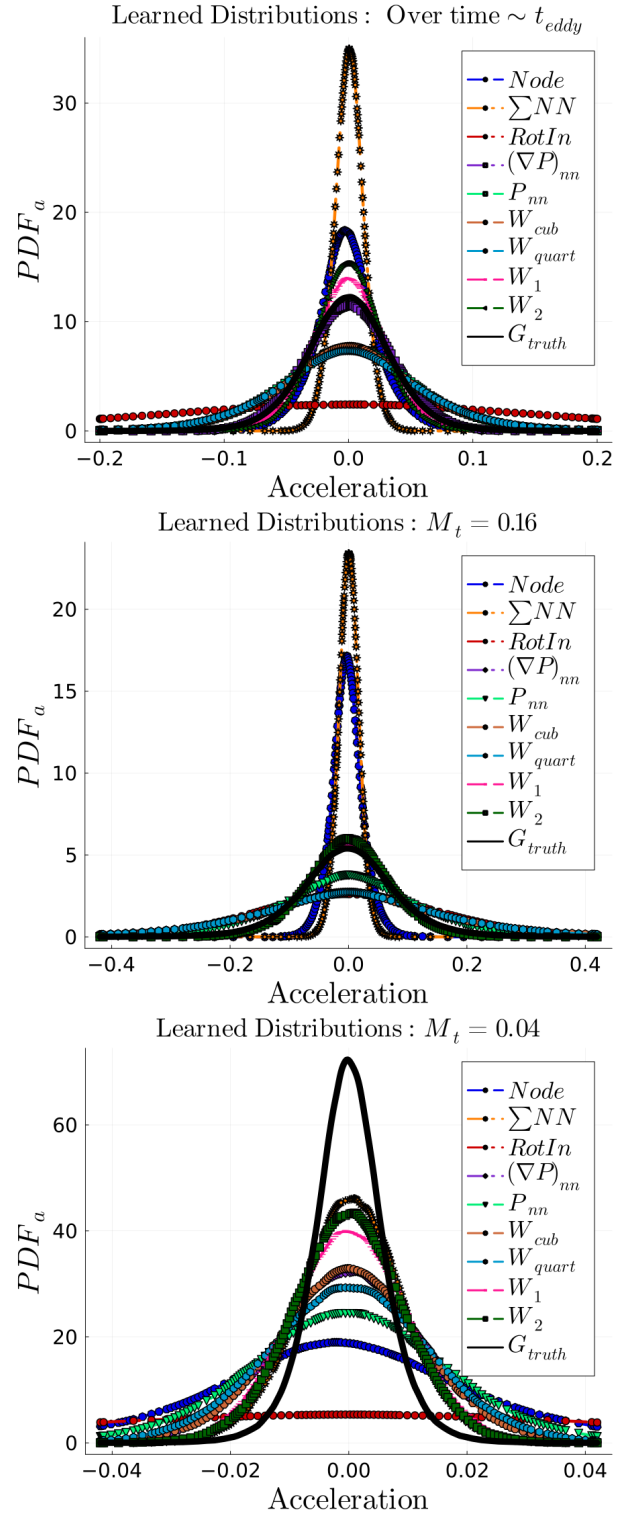


FIG. 10: Comparing acceleration statistics over time, and turbulent Mach numbers of all trained models showing the learned parameterized smoothing kernels are a closer match to DNS as compared to the cubic and quartic smoothing kernels on over longer time scales as well as at different Mach numbers $M_t = 0.04$ and $M_t = 0.16$. Here the smoothing kernels as compared in Fig. 9 are reported again to compare all the models

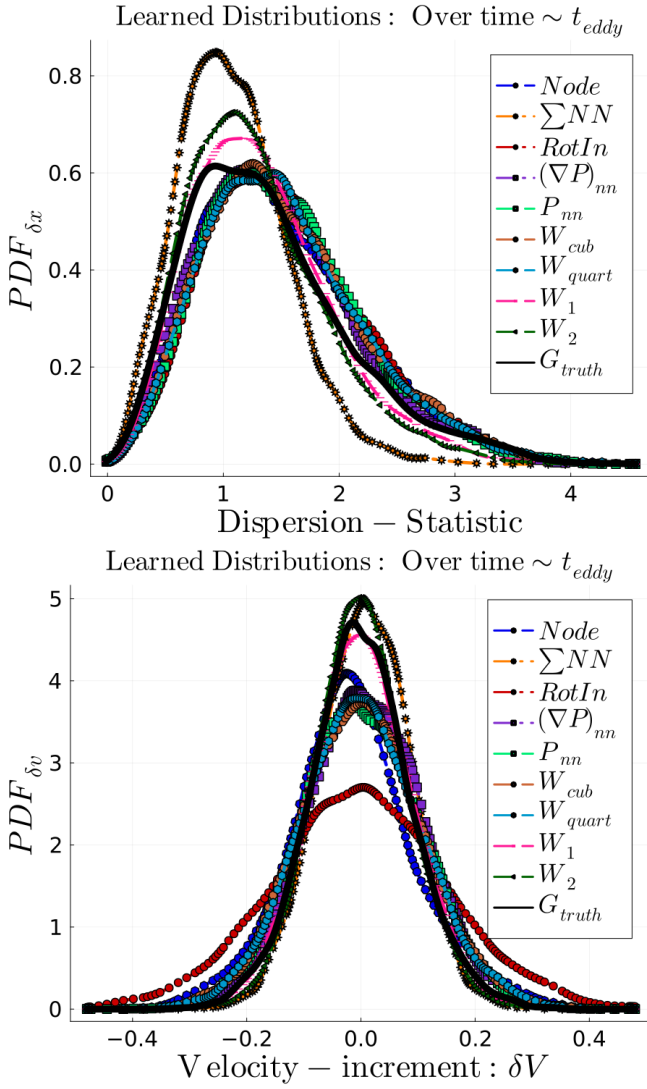


FIG. 11: A diagnostic check comparing single particle statistics on larger time scales (roughly 20 times longer than seen in training), with the new parameterized smoothing kernels SPH-informed model having the best fit. In this case, each is not seen in training, so represent an external diagnostic, however, in the Appendix C we see the effects of including the velocity increment statistical based loss.

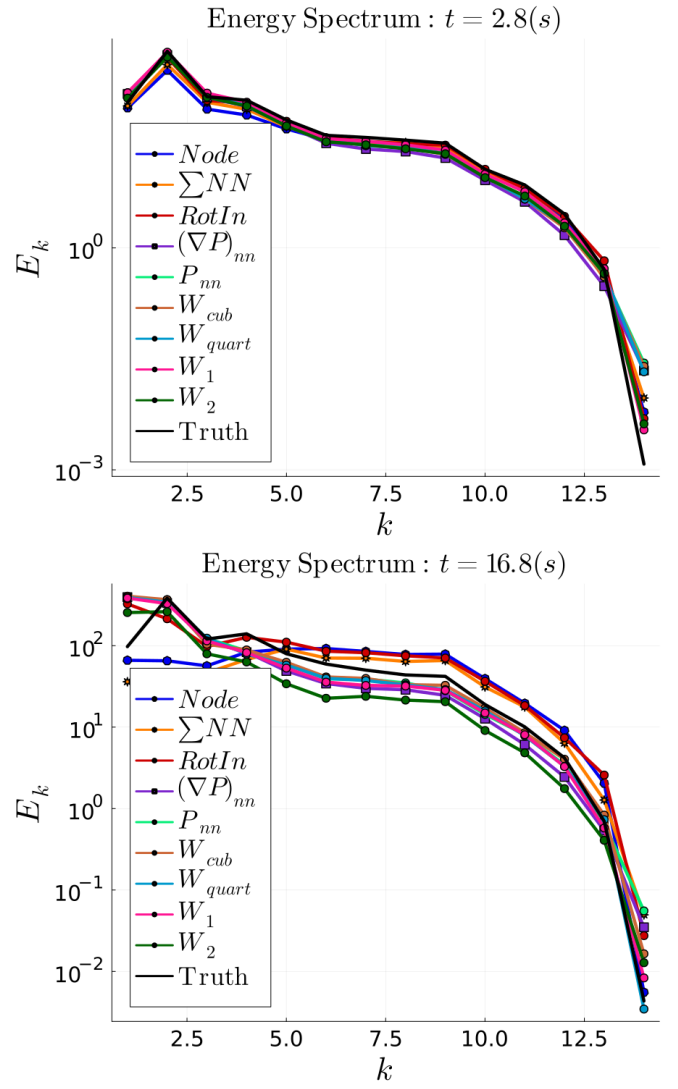


FIG. 12: Comparing energy spectrum over time; $t = 2.8s \sim t_\eta$ is on the Kolmogorov time scale, and $t = 16.8s \sim t_{eddy}$ is at the scale of the eddy turn over time. We see that the SPH based parameterized models captures the energy cascade seen in DNS on both the training time scale and in generalizing to larger time scales. Furthermore, as seen qualitatively seen in Fig. 4, the less informed models do not capture the cascade and show more energy in the small scale structures and less in the large scales. However, the fully parameterized SPH-informed models show a dissipation rate from the large to small scales to be larger than DNS.

Table III Shows a comparison of run-times and memory used for training each model on an Intel Core i9-10900X CPU, applying one step of the training algorithm (see Section B) using 2D models with 1024 particles. In the numerical experiments presented in this manuscript convergence was typically observed between 300-1000 iterations for both 2D and 3D training sets. Thus, in summary of training the 2D models, NODE required up to 1.3-4.3 CPU hours, NN-Sum required up to 3.5-11.5 CPU hours, Rot-Inv up to 2.4-8.1 CPU hours, ∇P - NN required up to 2.5-8.1 CPU hours, EoS NN required up to 0.7-2.5 CPU hours, and SPH-informed models required up to 0.5-1.7 CPU hours. Scaling up to 3D flows with 4096 particles required up to 240 CPU hours for the most expensive NN-Sum model to achieve convergence.

TABLE III: Comparing (2D) models: Run-times and Memory, $t \sim t_\lambda$: One iteration

Model with FSA	Run-time (s)	Memory (GiB)
NODE	15.50	9.62
NN summand	41.60	15.61
Rotationally Invariant NN	29.16	9.70
∇P - NN	29.37	9.51
EoS NN	8.87	5.19
SPH-informed	6.03	3.52

VI. CONCLUSIONS

Combining SPH based modeling, deep learning, automatic differentiation, and local sensitivity analysis, we have developed a learn-able hierarchy of parameterized "physics-explainable" Lagrangian models, and trained each model on both a validation set using weakly compressible SPH data and a high fidelity DNS data set (at three different resolutions) in order to find which model minimizes generalization error over larger time scales and different turbulent Mach numbers. We proposed two new parameterized smoothing kernels, which, once trained on DNS data, improve the accuracy of the SPH predictions compared to DNS, with the second kernel, $W_2(r; h, a, b)$ (which is smooth at the origin), performing best.

Starting from a Neural ODE based model, we showed that incrementally adding more physical structure into the Lagrangian models using SPH has several important benefits:

- Improves Generalizability: as seen in Section D 2 and Section V, where we test the ability of the models to predict flows under different conditions not seen in training. The general trend emerged: as more physics informed SPH based structure was embedded in the model, the lower the generalization errors became (both with respect to the loss function used in training and with the external statistical diagnostics). Furthermore, using NNs embedded within the SPH framework to approximate unknown functions can improve generalizability over the standard formulation, but the fully parameterized SPH including the novel parameterized smoothing kernels outperformed the rest.

- Learn new Smoothing Kernels: a key ingredient in the construction of the SPH method relies on the properties of the smoothing kernel. Two novel parameterized and learn-able smoothing kernels – non-smooth and smooth at the origin, respectively – were developed. We showed that introducing the parameterization freedom in the kernels increases flexibility of the physics informed SPH based models and improves generalizability.
- Enforces Physical Symmetries: the parameterized framework automatically enforces the Galileilian invariance and allows to keep conservation of linear and angular momenta (translational and rotational invariances) under control across the scales of coarse-graining, see Table II.
- Improves Interpretability: as the learned parameters become physically meaningful – in a critical contrast to parameters of the NN models which are physics-agnostic. For example, and as seen in Section C, our approach is capable to reconstruct with minimal error (counted against the ground truth) physical parameters, associated with the equation of state, artificial shear and bulk viscosity and external forcing.
- More Efficient to Train: as seen in Table III, where we compare different models within the generalized SPH hierarchy.
- Robust with respect to different levels of coarse-graining: the reported results hold under different resolutions within the inertial range of scales, namely at $N = 12^3, 16^3, 20^3$ as seen in Fig. 7 and Fig. 8.

In future works, we plan to go beyond the weakly compressible SPH Lagrangian modeling discussed so far and include compressible effects such as shocks. Specifically, we aim to further investigate SPH as a reduced-order Lagrangian model of highly compressible turbulent flows, and further investigate the ability to improve and optimize the SPH framework using the two new parameterized smoothing kernels proposed in this work, parameterized artificial viscosities and regularization terms.

The Julia source code of our parameterized Lagrangian simulators, the gradient based learning algorithm, sensitivity calculations for each model in the above hierarchy, and the post processing tools can be found in <https://github.com/mwoodward-cpu/LearningSPH>.

Acknowledgments

This work has been authored by employees of Triad National Security, LLC which operates Los Alamos National Laboratory (LANL) under Contract No. 89233218CNA000001 with the U.S. Department of Energy/National Nuclear Security Administration. We acknowledge support from LANL's Laboratory Directed Research and development (LDRD) program, project number

20190058DR and computational resources from LANL's Institutional Computing (IC) program.

The work of Michael Woodward on the project was supported by an Office of Science Graduate Student Research (SCGSR) Program of the Department of Energy (DOE) fel-

lowship, administered by the Oak Ridge Institute for Science and Education (ORISE) for the DOE. ORISE is managed by the Oak Ridge Associated Universities (ORAU) under contract number DE-SC0014664.

-
- [1] Frisch U. *Turbulence: The Legacy of AN Kolmogorov*. Cambridge University Press; 1995.
- [2] Pope SB. *Turbulent flows*. Cambridge: Cambridge Univ. Press; 2011.
- [3] Sagaut P. *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media; 2006.
- [4] Philip Holmes JL. *Turbulence Coherent Structures, Dynamical Systems and Symmetry*. Cambridge; 2012.
- [5] Schmid PJ. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*. 2010;656:5–28.
- [6] Kutz JN, Brunton SL, Brunton BW, Proctor JL. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Philadelphia, PA, USA: SIAM-Society for Industrial and Applied Mathematics; 2016.
- [7] Lin YT, Tian Y, Livescu D, Anghel M. Data-driven learning for the Mori-Zwanzig formalism: a generalization of the Koopman learning framework. *SIAM Journal on Applied Dynamical Systems*. 2021;20:2558–2601.
- [8] Tian Y, Lin YT, Anghel M, Livescu D. Data-driven learning of Mori-Zwanzig operators for isotropic turbulence. *Physics of Fluids*. 2021;33(12):125118.
- [9] Woodward M, Tian Y, Mohan AT, Lin YT, Hader C, Livescu D, et al. In: *Data-Driven Mori-Zwanzig: Approaching a Reduced Order Model for Hypersonic Boundary Layer Transition*. .
- [10] Chen W, Wang Q, Hesthaven JS, Zhang C. Physics-informed machine learning for reduced-order modeling of nonlinear problems. *Journal of Computational Physics*. 2021;446:110666.
- [11] Mohan AT, Nagarajan K, Livescu D. Learning Stable Galerkin Models of Turbulence with Differentiable Programming. arXiv; 2021.
- [12] Lin YT, Tian Y, Livescu D. Regression-based projection for learning Mori-Zwanzig operators. arXiv; 2022.
- [13] Tian Y, Chertkov M, Woodward M, Stepanov M, Fryer C, Hyett CM, et al. A11.00008 : Machine Learning Lagrangian Large Eddy Simulations with Smoothed Particle Hydrodynamics. 74th Annual Meeting of the APS Division of Fluid Dynamics, Session A11: Turbulence: Modeling & Simulations I: Data-Driven and Machine Learning Approaches. 2021;.
- [14] Hyett CM, Tian Y, Chertkov M, Woodward M, Livescu D, Stepanov M. E31.00009 : Machine Learning Statistical Evolution of the Coarse-Grained Velocity Gradient Tensor. 74th Annual Meeting of the APS Division of Fluid Dynamics, Session E31: Nonlinear Dynamics: Model Reduction & Turbulence I. 2021;.
- [15] Portwood GD, Nadiga BT, Saenz JA, Livescu D. Interpreting neural network models of residual scalar flux. *Journal of Fluid Mechanics*. 2021;907:A23.
- [16] Gingold RA, Monaghan JJ. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*. 1977 Nov;181:375–389.
- [17] Monaghan JJ. Smoothed Particle Hydrodynamics. *Annual Review of Astronomy and Astrophysics*. 1992;30(1):543–574.
- [18] Monaghan JJ. Smoothed Particle Hydrodynamics and Its Diverse Applications. *Annual Review of Fluid Mechanics*. 2012;44(1):323–346.
- [19] Shadloo MS, Oger G, Le Touzé D. Smoothed particle hydrodynamics method for fluid flows, towards industrial applications: Motivations, current state, and challenges. *Computers & Fluids*. 2016;136:11–34.
- [20] Kraichnan RH. Kolmogorov's Hypotheses and Eulerian Turbulence Theory. *The Physics of Fluids*. 1964;7(11):1723–1734.
- [21] Kraichnan RH. Lagrangian-History Closure Approximation for Turbulence. *The Physics of Fluids*. 1965;8(4):575–598.
- [22] Chola K. SPH Consistent With Explicit LES. arXiv; 2018.
- [23] Tian Y, Woodward M, Stepanov M, Fryer C, Hyett C, Livescu D, et al. Lagrangian Large Eddy Simulations via Physics Informed Machine Learning. arXiv; 2022.
- [24] Lind SJ, Rogers BD, Stansby PK. Review of smoothed particle hydrodynamics: towards converged Lagrangian flow modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*. 2020;476(2241):20190801.
- [25] Ling J, Kurzawski A, Templeton J. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*. 2016;807:155–166.
- [26] King R, Hennigh O, Mohan A, Chertkov M. From Deep to Physics-Informed Learning of Turbulence: Diagnostics. arXiv:181007785. 2018;.
- [27] Schenck C, Fox D. SPNets: Differentiable Fluid Dynamics for Deep Neural Networks. In: Billard A, Dragan A, Peters J, Morimoto J, editors. *Proceedings of The 2nd Conference on Robot Learning*. vol. 87 of *Proceedings of Machine Learning Research*. PMLR; 2018. p. 317–335.
- [28] Mohan AT, Gaitonde DV. A Deep Learning based Approach to Reduced Order Modeling for Turbulent Flow Control using LSTM Neural Networks. <https://arxiv.org/abs/191105180v1>. 2018;.
- [29] Duraisamy K, Iaccarino G, Xiao H. Turbulence Modeling in the Age of Data. *Annual Review of Fluid Mechanics*. 2019;51(1):357–377.
- [30] Fonda E, Pandey A, Schumacher J, Sreenivasan KR. Deep learning in turbulent convection networks. *Proceedings of the National Academy of Sciences*. 2019;116(18):8667–8672.
- [31] Mohan AT, Lubbers N, Livescu D, Chertkov M. Embedding Hard Physical Constraints in Neural Network Coarse-Graining of 3D Turbulence. arXiv:200200021. 2020;.
- [32] Maulik R, Mohan A, Lusch B, Madireddy S, Balaprakash P, Livescu D. Time-series learning of latent-space dynamics for reduced-order model closure. *Physica D: Nonlinear Phenomena*. 2020 Apr;405:132368.
- [33] Ummenhofer B, Prantl L, Thuerey N, Koltun V. Lagrangian Fluid Simulation with Continuous Convolutions. In: *International Conference on Learning Representations*; 2020. .
- [34] Mohan AT, Tretiak D, Chertkov M, Livescu D. Spatio-temporal deep learning models of 3D turbulence with physics informed diagnostics. *Journal of Turbulence*. 2020;21(9-10):484–524.
- [35] Tian Y, Livescu D, Chertkov M. Physics-informed machine learning of the Lagrangian dynamics of velocity gradient tensor.

- Phys Rev Fluids. 2021 Sep;6:094607.
- [36] Lagaris IE, Likas A, Fotiadis DI. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*. 1998;9(5):987–1000.
- [37] CNLS at LANL. Physics Informed Machine Learning (PIML) workshop; 2016, 2018, 2020.
- [38] Raissi M, Perdikaris P, Karniadakis GE. Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. arXiv:171110566. 2017;.
- [39] Chen RTQ, Rubanova Y, Bettencourt J, Duvenaud D. Neural Ordinary Differential Equations. arXiv:180607366. 2019;.
- [40] Rackauckas C, Ma Y, Martensen J, Warner C, Zubov K, Supekar R, et al.. Universal Differential Equations for Scientific Machine Learning; 2020.
- [41] Ladický L, Jeong S, Solenthaler B, Pollefeys M, Gross M. Data-Driven Fluid Simulations Using Regression Forests. *ACM Trans Graph*. 2015 Oct;34(6).
- [42] Zhai X, Yin Y, Pellegrino JW, Haudek KC, Shi L. Applying machine learning in science assessment: a systematic review. *Studies in Science Education*. 2020;56(1):111–151.
- [43] Wang Y, Yao H, Zhao S. Auto-encoder based dimensionality reduction. *Neurocomputing*. 2016;184:232–242. RoLoD: Robust Local Descriptors for Computer Vision 2014.
- [44] Serre T. Deep Learning: The Good, the Bad, and the Ugly. *Annual Review of Vision Science*. 2019;5(1):399–426. PMID: 31394043.
- [45] Deng L, Liu Y. Deep Learning in Natural Language Processing. Springer; 2018.
- [46] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. *Neural Networks*. 1989;2(5):359–366.
- [47] Bücker M. Automatic Differentiation Applications, Theory and Implementations. Springer; 2006.
- [48] de Anda-Suárez J, Carpio M, Jeyakumar S, Soberanes H, Mosiño J, Cruz Reyes L. In: Optimization of the Parameters of Smoothed Particle Hydrodynamics Method, Using Evolutionary Algorithms. Springer; 2018. p. 153–167.
- [49] Rabault J, Kolaas J, Jensen A. Performing particle image velocimetry using artificial neural networks: a proof-of-concept. *Measurement Science and Technology*. 2017;28(12):125301.
- [50] Lee Y, Yang H, Yin Z. PIV-DCNN: cascaded deep convolutional neural networks for particle image velocimetry. *Experiments in Fluids*. 2017;58(12):171.
- [51] Cai S, Liang J, Gao Q, Xu C, Wei R. Particle image velocimetry based on a deep learning motion estimator. *IEEE Transactions on Instrumentation and Measurement*. 2019;.
- [52] Stulov N, Chertkov M. Neural Particle Image Velocimetry. arXiv:210111950. 2021;.
- [53] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. arXiv; 2014.
- [54] Monaghan JJ. Smoothed particle hydrodynamics. *Reports on Progress in Physics*. 2005 Aug;68(8):1703–1759.
- [55] Lucy LB. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*. 1977 Dec;82:1013–1024.
- [56] Petersen MR, Livescu D. Forcing for statistically stationary compressible isotropic turbulence. *Physics of Fluids*. 2010;22(11):116101.
- [57] Morris JP, Fox PJ, Zhu Y. Modeling Low Reynolds Number Incompressible Flows Using SPH. *Journal of Computational Physics*. 1997;136(1):214–226.
- [58] Cossins PJ. Smoothed Particle Hydrodynamics. arXiv:10071245. 2010;.
- [59] Domínguez JM, Crespo AJC, Gómez-Gesteira M, Marongiu JC. Neighbour lists in smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*. 2011;67(12):2026–2042.
- [60] Fulk DA, Quinn DW. An Analysis of 1-D Smoothed Particle Hydrodynamics Kernels. *Journal of Computational Physics*. 1996;126(1):165–180.
- [61] Liu MB, Liu GR. Smoothed Particle Hydrodynamics (SPH): an Overview and Recent Developments. *Archives of Computational Methods in Engineering*. 2010;17:25–76.
- [62] Donello M, Carpenter M, Babaee H. Computing Sensitivities in Evolutionary Systems: A Real-Time Reduced Order Modeling Strategy. arXiv:201214028. 2020;.
- [63] Zhang H, Sandu A. FATODE: A Library for Forward, Adjoint, and Tangent Linear Integration of ODEs. *SIAM J Sci Comput*. 2014;36.
- [64] Ma Y, Dixit V, Innes M, Guo X, Rackauckas C. A Comparison of Automatic Differentiation and Continuous Sensitivity Analysis for Derivatives of Differential Equation Solutions. arXiv:181201892. 2021;.
- [65] Yeung PK, Bogras MS. Relative dispersion in isotropic turbulence. Part 1. Direct numerical simulations and Reynolds-number dependence. *Journal of Fluid Mechanics*. 2004;503:93–124.
- [66] Chen YC. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*. 2017;1(1):161–187.
- [67] Baltzer JR, Livescu D. Variable-density effects in incompressible non-buoyant shear-driven turbulent mixing layers. *Journal of Fluid Mechanics*. 2020;900:A16.
- [68] Ryu J, Livescu D. Turbulence structure behind the shock in canonical shock–vortical turbulence interaction. *Journal of Fluid Mechanics*. 2014;756.
- [69] Diehl S, Rockefeller G, Fryer CL, Riethmiller D, Statler TS. Generating Optimal Initial Conditions for Smoothed Particle Hydrodynamics Simulations. arXiv; 2012.
- [70] Tennekes H, Lumley JL. A first course in turbulence. MIT PRESS; 1978.
- [71] Livescu D. Turbulence with large thermal and compositional density variations. *Annual Review of Fluid Mechanics*. 2020;52:309–341.
- [72] See Supplemental Material at [URL] for simulations;.
- [73] Liu MB, Liu GR. Smoothed Particle Hydrodynamics (SPH): an Overview and Recent Developments. *Archives of Computational Methods in Engineering*. 2010;17(1):25–76.
- [74] Price DJ. Smoothed particle hydrodynamics and magnetohydrodynamics. *Journal of Computational Physics*. 2012 Feb;231(3):759–794.
- [75] Monaghan JJ. SPH and Riemann Solvers. *Journal of Computational Physics*. 1997;136(2):298–307.
- [76] Monaghan JJ, Gingold RA. Shock simulation by the particle method SPH. *Journal of Computational Physics*. 1983;52(2):374–389.
- [77] Lind SJ, Xu R, Stansby PK, Rogers BD. Incompressible smoothed particle hydrodynamics for free-surface flows: A generalised diffusion-based algorithm for stability and validations for impulsive flows and propagating waves. *Journal of Computational Physics*. 2012;231(4):1499–1523.
- [78] Marrone S, Antuono M, Colagrossi A, Colicchio G, Le Touzé D, Graziani G. δ -SPH model for simulating violent impact flows. *Computer Methods in Applied Mechanics and Engineering*. 2011;200(13):1526–1542.
- [79] Nouri AG, Babaee H, Givi P, Chelliah HK, Livescu D. Skeletal Model Reduction with Forced Optimally Time Dependent Modes. *Combustion and Flame*. 2022;235:111684.
- [80] Revels J, Lubin M, Papamarkou T. Forward-Mode Automatic Differentiation in Julia. arXiv:160707892 [csMS]. 2016;.

- [81] Innes M. Don't Unroll Adjoint: Differentiating SSA-Form Programs. CoRR. 2018;abs/1810.07951.
 [82] Bradley A. PDE Constrained Optimization and the Adjoint Method. csstanford.edu. 2019;.

Appendix A: SPH (in more details)

1. Basic formulation

In this section we give a summary of SPH, most of which can be found in [17, 18, 58, 69]. SPH is a discrete approximation to a continuous flow field by using a series of discrete particles. Starting with the trivial identity

$$A(\mathbf{r}) = \int_V A(\mathbf{r}') \delta(\mathbf{r} - \mathbf{r}') d\mathbf{r}',$$

where A is any scalar or tensor field. Using the smoothing kernel W (for interpolation onto smooth "blobs" of fluid) and after a Taylor expansion it can be shown that (according to symmetry of smoothing kernel [58])

$$A(\mathbf{r}) = \int_V A(\mathbf{r}') W(|\mathbf{r} - \mathbf{r}'|, h) d\mathbf{r}' + \mathcal{O}(h^2).$$

where W is constrained to behave similar to the delta function,

$$\int_V W(\mathbf{r}, h) d\mathbf{r} = 1, \quad \lim_{h \rightarrow 0} W(\mathbf{r}, h) = \delta(\mathbf{r}).$$

The choice of smoothing kernels is important, and effects the consistency and accuracy of results [54], where bell-shaped, symmetric, monotonic kernels are the most popular [60], however there is still disagreement on the best smoothing kernels to use (but generally should satisfy the conditions outlined in [61]). Commonly used are the B-spline smoothing kernels with a finite support (approximating a Gaussian kernel). The cubic smoothing kernel used in this work has the following form:

$$w(q) = \sigma \begin{cases} \frac{1}{4}(2-q)^3 - (1-q)^3 & 0 \leq q < 1 \\ \frac{1}{4}(2-q)^3 & 1 \leq q \leq 2 \\ 0 & 2 \leq q \end{cases} \quad (\text{A1})$$

where, $W(|\mathbf{r} - \mathbf{r}_j|, h) = h^{-d} w(q)$ with $q = |\mathbf{r} - \mathbf{r}_j|/h$, and $\sigma = \sigma(d) = [1/\pi$ if $d=3$, $10/7\pi$ if $d=2$] is a normalizing constant to satisfy the integral constraint on W (see [18, 58, 60] for more details). The finite support allows one to use neighborhood list algorithms discussed below to utilize computational advantages (only requires a local cloud of interacting particles instead of all the particles in the computational domain that would be required with a Gaussian kernel). The quartic smoothing kernel (see [73]) used in this work is

$$w(q) = \sigma \begin{cases} (2/3 - 9/8q^2 + 19/24q^3 - 5/32q^4) & 0 \leq q < 2 \\ 0 & 2 \leq q \end{cases} \quad (\text{A2})$$

Where $\sigma = 315/(208\pi h^3)$ for $d=3$.

In Section IV, two new parameterized smoothing kernels are introduced, where the shapes are described by the algebraic equations Eqs. 11 and 12. This is done in order to increase the flexibility of the SPH model as well as to allow the optimization framework to "discover" the best shaped kernel for our application. Both kernels are introduced to cover a wide range of possible shapes, not necessarily bell-shaped (see Fig. 13 below), although both satisfy the conditions for approximating the delta function. The main difference between the two is that $W_2(a, b)$ has a smoother second derivative and may reduce the likelihood of particle pairing as compared to $W_1(a, b)$, however, we postpone a deeper analysis of this comparison until future work, in which we will analyze the effects to changing kernels with standardized compressible flows (such as the Sod shock and Sedov blast wave [74]).

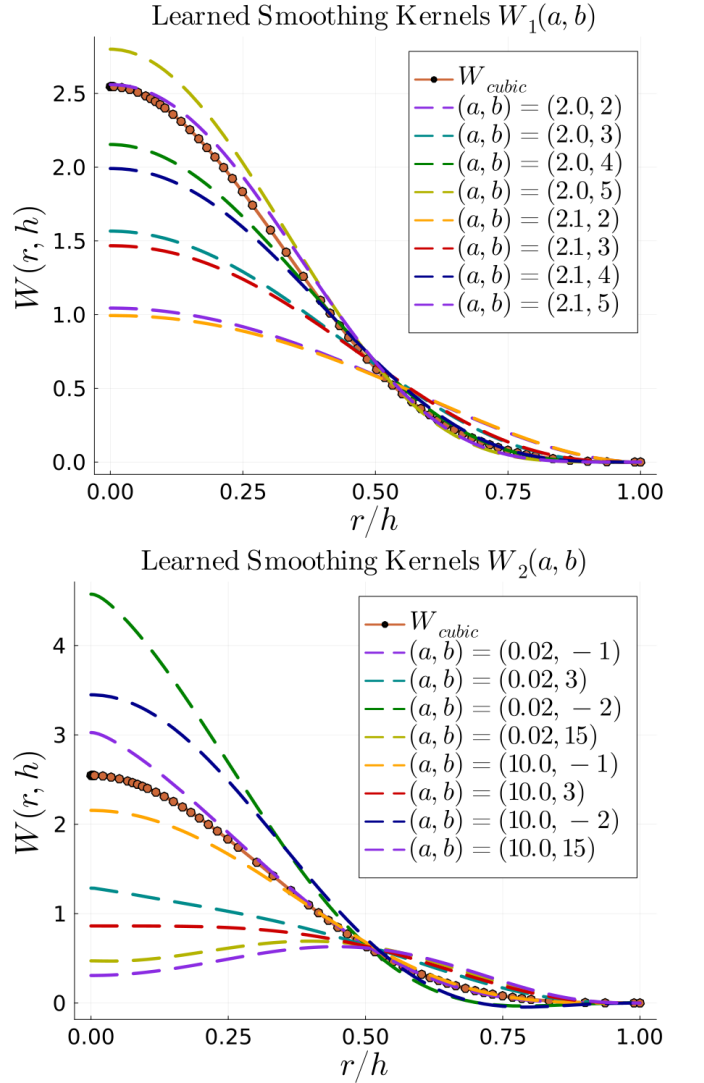


FIG. 13: Varying shapes for the two new parameterized smoothing kernels as compared to the cubic and Wendland kernel for $d=3$. The optimization framework is applied in order to discover the best fit shape (see 5).

SPH can be formulated through approximating integral interpolants of any scalar or tensor field A by a series of discrete particles

$$\begin{aligned} \langle A(\mathbf{r}) \rangle &= \int_V A(\mathbf{r}') W(|\mathbf{r} - \mathbf{r}'|, h) d\mathbf{r}' \\ &\approx \sum_i m_i \frac{A(\mathbf{r}_i)}{\rho(\mathbf{r}_i)} W(|\mathbf{r} - \mathbf{r}_i|, h), \end{aligned} \quad (\text{A3})$$

(i.e a convolution of A with W) where $d\mathbf{r}'$ denotes a volume element and $W(r, h)$ is the smoothing kernel. Each particle represents a continuous "blob" of fluid and carries the fluid quantities in the Lagrangian frame (such as pressure P_i , density ρ_i , velocity \mathbf{v}_i , etc.)

The convenience of this method becomes apparent when the differential operators are approximated (see [73] for a more detailed derivation). Using the integral interpolation,

$$\langle \nabla_r A(\mathbf{r}) \rangle = \int_V \nabla_r A(\mathbf{r}') W(|\mathbf{r} - \mathbf{r}'|, h) d\mathbf{r}'.$$

Now, using the particle approximation

$$\langle \nabla_r A(\mathbf{r}) \rangle \approx \sum_i m_i \frac{A(\mathbf{r}_i)}{\rho(\mathbf{r}_i)} \nabla_r W(|\mathbf{r} - \mathbf{r}_i|, h),$$

where we see that in this direct approach to approximate the gradient operator we only need to know the gradient of the smoothing kernel (which is usually fixed beforehand). Multiple methods have been proposed and different methods are best suited for different problems. Similar approximations hold for taking the divergence or curl of a vector field [16]. The most common "symmetrized" approximation of the gradient operator is derived from the following identity,

$$\nabla_r A(\mathbf{r}_i) = \rho \left(\frac{A(\mathbf{r}_i)}{\rho^2} \nabla_r \rho - \nabla_r \left(\frac{A(\mathbf{r}_i)}{\rho} \right) \right),$$

and is approximated with particles as

$$\nabla_r A_i \approx \rho_i \sum_j^N m_j \left(\frac{A_j}{\rho_j^2} - \frac{A_i}{\rho_i^2} \right) \nabla_{r_i} W_{ij}. \quad (\text{A4})$$

2. Approximation of Flow Equations

The above integral interpolant approximations using series of particles can be used to discretize the equations of motion (as seen in Eq. (2) with more details found in [16, 18]). Each particle carries a mass m_i and velocity \mathbf{v}_i , and other properties (such as pressure, density etc.). We can use Eq. (A3) to estimate the density everywhere by

$$\rho(\mathbf{r}_i) = \sum_j m_j W(|\mathbf{r}_i - \mathbf{r}_j|, h),$$

where although the summation is over all particles, because the smoothing kernel has compact support the summation

only needs to occur over the smoothing radius (here $2h$ as seen in Eq. (A1)). Another popular way to approximate the density is through using the continuity equation and approximating the divergence of the velocity field in different ways [17]. In what follows we use the notation $A_i = A(\mathbf{r}_i)$. Using the gradient approximation defined above (Eq. (A4)), the pressure gradient could be estimated by using

$$\rho_i \nabla_r P_i = \sum_j m_j (P_j - P_i) \nabla_{r_i} W_{ij},$$

where $W_{ij} = W(|\mathbf{r}_i - \mathbf{r}_j|, h)$. However, in this form the momentum equation $d_i \mathbf{v} = -\frac{1}{\rho} \nabla_r P$ does not conserve linear and angular momentum [17]. To improve this, a symmetrization is often done to the pressure gradient term by rewriting $\frac{\nabla_r P}{\rho} = \partial_r \left(\frac{P}{\rho} \right) + \frac{P}{\rho^2} \nabla_r \rho$. This results in a momentum equation for particle i discretized as

$$\frac{d\mathbf{v}_i}{dt} = - \sum_j m_j \left(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2} \right) \nabla_{r_i} W_{ij},$$

which produces a symmetric central force between pairs of particles and as a result linear and angular momentum are conserved [17], however is not invariant to changes in background pressure p_0 . Next, including an artificial viscosity term and external forcing, the full set of ODEs approximating PDEs governing fluid motion (namely Euler's equations with an added artificial viscosity Π) is

$$\begin{aligned} \frac{d\mathbf{r}_i}{dt} &= \mathbf{v}_i \quad \forall i \in \{1, 2, \dots, N\} \\ \frac{d\mathbf{v}_i}{dt} &= - \sum_{j \neq i}^N m_j \left(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2} + \Pi_{ij} \right) \nabla_{r_i} W_{ij} + \mathbf{f}_{ext}. \end{aligned}$$

In this work, we start by using the weakly compressible formulation by assuming a barotropic fluid, where equation of state (EoS) is given by

$$P(\rho) = \frac{c^2 \rho_0}{\gamma} \left[\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right],$$

as in [18], where ρ_0 is the initial reference density, and $\gamma = 7$ is used. In future work, we plan on including the energy equation to extend these methods for highly compressible applications.

There are many different forms of artificial viscosity that have been proposed [75]. In this work, we use the popular formulation of Π_{ij} that approximates the contribution from the bulk and shear viscosity along with an approximation of Nueman-Richtmyer viscosity for handling shocks [18, 57]:

$$\Pi_{ij} = \begin{cases} \frac{-\alpha c_{ij} \mu_{ij} + \beta \mu_{ij}^2}{\rho_{ij}}, & \mathbf{v}_{ij} \cdot \mathbf{r}_{ij} < 0, \\ 0, & \text{otherwise} \end{cases}, \quad (\text{A5})$$

where $c_{ij} = 0.5(c_i + c_j)$ and $c_i = \sqrt{dP(\rho_i)/d\rho}$ represents the speed of sound of particle i and

$$\mu_{ij} = \frac{h\mathbf{v}_{ij} \cdot \mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^2 + \varepsilon h^2}, \quad \rho_{ij} = 0.5(\rho_i + \rho_j),$$

This artificial viscosity term was constructed in the standard way following [75, 76]: The linear term involving the speed of sound was based on the viscosity of an ideal gas. This term scales linearly with the velocity divergence, is negative to enforce $\Pi_{ij} > 0$, and should be present only for convergent flows ($\mathbf{v}_{ij} \cdot \mathbf{r}_{ij} < 0$). The quadratic term including $(\mathbf{v}_{ij} \cdot \mathbf{r}_{ij})^2$ is used to prevent penetration in high Mach number collisions by producing an artificial pressure roughly proportional to $\rho|\mathbf{v}|^2$ and approximates the von Neumann-Richtmyer viscosity (and should also only be present for convergent flows). There are several advantages to this formulation of Π_{ij} ; mainly it is Galilean and rotationally invariant, thus conserves total linear and angular momentum. A more detailed derivation is found in [58] (along with other formulations of artificial viscosities).

In practice the summation Eq. (2) over all particles is carried out through a neighborhood list algorithm (such as the cell linked list algorithm with a computational cost that scales as $\mathcal{O}(N)$ [59]). We also note that Eq. (2) can also be derived from Euler-Lagrange equations after defining a Lagrangian, see [58] (for respective analysis of the inviscid case, when the artificial viscosity term is neglected), then an artificial viscosity Π_{ij} term can be incorporated by using the SPH discretizations, see [54] for details.

Although the above SPH framework is the most common and contains the same elements as most formulations [18, 54, 73, 74, 77, 78], additional terms can be included to address certain applications and situations. For example, a particle regularization can be included in an attempt to correct issues occurring in certain applications involving particle instabilities and anisotropic distributions which can decrease the accuracy of SPH. In the works of Lind et al. [77], a particle shifting is introduced for incompressible SPH solvers to reduce noise in the pressure field by using Fick's law of diffusion to shift particles in a manner that prevents highly anisotropic distributions. In Marrone et al. [78], the authors introduce a density diffusion term for simulating violent impact flows. However, in this work, we do not incorporate any of these additional particle type regularization terms (on top of the artificial viscosity term which also acts to regularize the particle distributions) for two main reasons: (1) it is not expected that these additional particle regularization terms will have a significant effect for the resolutions considered in this study (as it was found that the artificial viscosity term used was sufficient), and also because (2) the primary focus of this manuscript is to develop a hierarchy of parameterized reduced Lagrangian models for turbulent flows, and to investigate the effects of enforcing physical structure within a Lagrangian framework through SPH versus relying on neural networks (NN)s as universal function approximators.

3. External Forcing

In order to approach a stationary homogeneous and isotropic turbulent flow, a deterministic forcing is used (for simplifying the learning algorithms), which is commonly used in CFD literature, (e.g. as in [56] which is what is used in generating the ground truth DNS data) for analyzing stationary homogeneous and isotropic turbulence. Then,

$$\mathbf{f}_{ext}^i = \frac{\theta_{inj}}{KE} \mathbf{v}_i, \quad KE = \frac{0.5}{N} \sum_{k=1}^N \rho_k (u_k^2 + v_k^2 + w_k^2)$$

is the kinetic energy computed at each time step, θ_{inj} represents the rate of energy injected into the flow.

4. Numerical Algorithm for Forward Solving SPH

We use a Velocity Verlet (leap frog) numerical scheme for generating the SPH ground truth data, and for making prediction steps required in our gradient based optimization described in Eq. (IV). Using the notation,

$$\mathbf{X} = \{(\mathbf{r}_i, \mathbf{v}_i) | \forall i \in \{1, \dots, N\}\}, \quad \boldsymbol{\rho} = \{\rho_i | \forall i \in \{1, \dots, N\}\},$$

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i, \quad \frac{d\mathbf{v}_i}{dt} = \mathbf{F}_i(\boldsymbol{\rho}, \mathbf{X}),$$

we proceed according to the following algorithm

- 1: Compute ρ^k using Eq. (A3),
- 2: Compute $F_i^k(\rho^k, \mathbf{X}^k)$ using Eq. (2),
- 3: $\mathbf{v}_i^{k+\frac{1}{2}} = \mathbf{v}_i^k + \frac{\Delta t}{2} F_i^k$,
- 4: $\mathbf{r}_i^{k+1} = \mathbf{r}_i^k + \Delta t \mathbf{v}_i^{k+\frac{1}{2}}$,
- 5: Compute ρ^{k+1} using Eq. (A3),
- 6: Compute $F_i^{k+\frac{1}{2}}(\rho^{k+1}, \mathbf{X}^{k+\frac{1}{2}})$ using Eq. (2),
- 7: $\mathbf{v}_i^{k+1} = \mathbf{v}_i^{k+\frac{1}{2}} + \frac{\Delta t}{2} F_i^{k+\frac{1}{2}}$,

repeated for each time step, $k \in \{0, \Delta t, \dots, T\}$, where the time step, Δt , is chosen according to the Courant-Friedrichs-Lewy (CFL) condition, $\Delta t \leq 0.4h/c$. This algorithm has the following physical interpretation: it prevents spatial information transfer through the code at a rate greater than the local speed of sound (small in the almost incompressible case considered in this manuscript).

Appendix B: Methods

In this section, we provide further details into the methods of this work, including loss functions, sensitivity analysis, and the learning algorithm.

1. Loss functions

In this section, we consider three different loss functions: trajectory based (Lagrangian), field based Eulerian, and La-

grangian statistics based, described in the following three subsections. Since our overall goal involves learning Lagrangian and SPH based models for turbulence applications, it is the underlying statistical features and large scale field structures we want our models to learn and generalize with. This is discussed further in Section D and Appendix V, where we compare the statistical and field based generalizability of each model within the hierarchy.

a. Trajectory Based Loss Function

A naive loss function to consider is the Mean Squared Error (*MSE*) of the difference in the Lagrangian particles positions and velocities, as they evolve in time,

$$L_{tr}(\boldsymbol{\theta}) = \text{MSE}(\mathbf{X}, \hat{\mathbf{X}}(\boldsymbol{\theta})) = \|\mathbf{X} - \hat{\mathbf{X}}(\boldsymbol{\theta})\|^2/N,$$

where \mathbf{X} and $\hat{\mathbf{X}}$ are the particle states – the ground truth and the predicted, respectively. Minimizing this loss function will result in discovering optimal parameters such that the predicted trajectories gives the best possible match (within the model) for each of the particles. However, the Lagrangian tracer particles are non-inertial, whereas the SPH particles have mass, thus the trajectory based loss function is not consistent with the data.

b. Field Based Loss Function

The field based loss function tries to minimize the difference between the large scale structures found in the Eulerian velocity fields,

$$L_f(\boldsymbol{\theta}) = \text{MSE}(\mathbf{V}^f, \hat{\mathbf{V}}^f) = \|\mathbf{V}^f - \hat{\mathbf{V}}^f\|^2/N_f,$$

where $\mathbf{V}_i^f = \sum_{j=1}^{N_f} (m_j/\rho_j) \mathbf{v}_j W_{ij}(\|\mathbf{r}_i^f - \mathbf{r}_j\|, h)$ uses the same SPH smoothing approximation to interpolate the particle velocity onto a predefined mesh \mathbf{r}^f (with N_f grid points). Lets recall that SPH is, by itself, an approximation for the velocity field, therefore providing a strong additional motivation for using the field based loss function.

c. Lagrangian Statistics Based Loss Function

In order to approach learning Lagrangian models that capture the statistical nature of turbulent flows, one can use well established statistical tools/objects, such as single particle statistics [65]. In this direction, consider the time integrated Kullback–Leibler divergence (KL) as a loss function

$$L_{kl}(\boldsymbol{\theta}) = \int_{t=0}^{t_f} \int_{-\infty}^{\infty} P_{gt}(t, \mathbf{z}_{gt}, \mathbf{x}) \log \left(\frac{P_{gt}(t, \mathbf{z}_{gt}, \mathbf{x})}{P_{pr}(t, \mathbf{z}_{pr}(\boldsymbol{\theta}), \mathbf{x})} \right) d\mathbf{x} dt,$$

where $\mathbf{z}_{gt} = \mathbf{z}_{gt}(t)$, and $\mathbf{z}_{pr}(\boldsymbol{\theta}) = \mathbf{z}_{pr}(\boldsymbol{\theta}, t)$ represent single particle statistical objects over time of the ground truth and predicted data, respectively. For example, we can use the

velocity increment, $\mathbf{z}^i(t) = (\delta u_i, \delta v_i, \delta w_i)$, where $\delta u_i(t) = u_i(t) - u_i(0)$ and \mathbf{z} ranges over all particles. Here $P(t, \mathbf{z}(t), \mathbf{x})$ is a continuous probability distribution (in \mathbf{x}) constructed from data $\mathbf{z}(t)$ using Kernel Density Estimation (KDE), to obtain smooth and differentiable distributions from data, as discussed in [66], that is

$$P(\tau, \mathbf{z}, \mathbf{x}) = (Nh)^{-1} \sum_{i=1}^N K((z_i - \mathbf{x})/h),$$

where K is the smoothing kernel (chosen to be the normalized Gaussian in this work). In the experiments below, a combination of the L_f and L_{kl} are used with gradient descent in order to first guide the model parameters in order to reproduce large scale structures with L_f , then later refine the model parameters with respect to the small scale features inherent in the velocity increment statistics by minimizing L_{kl} .

d. Forward and Adjoint based Methods Supplemental

Let us, first, introduce some useful notations for our parameterized SPH informed models; $\mathbf{X}_i = (\mathbf{r}_i, \mathbf{v}_i)^T$, $\mathbf{X} = \{\mathbf{X}_i | i = 1, \dots, N\}$, where, $\mathbf{r}_i = (x_i, y_i, z_i)$, and, $\mathbf{v}_i = (u_i, v_i, w_i)$, are the position and velocity of particle i respectively. Also, $\boldsymbol{\theta} = [\theta^1, \dots, \theta^p]^T$ where p is the number of model parameters. Now, each parameterized Lagrangian model in the hierarchy can be stated in the ODE form

$$\forall i: \quad d\mathbf{X}_i/dt = \mathcal{F}_i(\mathbf{X}(t, \boldsymbol{\theta}), \boldsymbol{\theta}) = (\mathbf{v}_i, \mathbf{F}_i(\mathbf{X}, \boldsymbol{\theta}))^T, \quad (\text{B1})$$

where $\mathbf{F}_i(\mathbf{X}, \boldsymbol{\theta}) = \dot{\mathbf{v}}_i$ is the parameterized acceleration operator as defined in the above hierarchy. Forward and Adjoint based Sensitivity Analyses (FSA, ASA), analogous to forward and reverse mode AD respectively, can be used to compute the gradient of the loss function (see Section B 1), as seen in Eq.(B2)

$$\partial_{\boldsymbol{\theta}} L = \int_0^{t_f} \partial_{\mathbf{X}} \Psi(\mathbf{X}, \boldsymbol{\theta}, t) d_{\boldsymbol{\theta}} \mathbf{X}(\boldsymbol{\theta}, t) + \partial_{\boldsymbol{\theta}} \Psi(\mathbf{X}, \boldsymbol{\theta}, t) dt, \quad (\text{B2})$$

Briefly, FSA computes the sensitivity equations (SE) for $\mathbf{S}_i^k = d\mathbf{X}_i/d\theta^k$, by simultaneously integrating a system of ODEs: ($\forall i, \forall k = 1, \dots, p$:)

$$\frac{d\mathbf{S}_i^k}{dt} = \frac{\partial \mathcal{F}_i(\mathbf{X}(t), \boldsymbol{\theta})}{\partial \mathbf{X}_i} \mathbf{S}_i^k + \frac{\partial \mathcal{F}_i(\mathbf{X}(t), \boldsymbol{\theta})}{\partial \theta^k} \quad (\text{B3})$$

which has a computational cost that scales linearly with the number of parameters (for derivations of FSA and ASA see Appendix B). \mathbf{S}_i^k is computed by solving the IVP Eq. (13) with the initial condition $\mathbf{S}_i^k(0) = 0$ (assuming $\mathbf{X}(0)$ does not depend on $\boldsymbol{\theta}$). After solving for \mathbf{S}_i^k , the gradient is computed directly from Eq. (B2) which is used with standard optimization tools (e.g. Adam algorithm of [53]) to update the model parameters iteratively to minimize the loss (see Appendix B). As opposed to the FSA method, the ASA approach avoids needing to compute $d_{\boldsymbol{\theta}} \mathbf{X}$ by instead numerically solving a system of equations for the adjoint equation

(AE) Eq. (B9) backwards in time, according to Section B 2 a. Once λ is computed through integration backwards in time, the gradient of the loss function can be computed according to, $\partial_{\theta}L = -\int_0^{t_f} \lambda^T \partial \mathcal{F} / \partial \theta dt$.

The computational cost of solving the AE is independent of the number of parameters, however, for high dimensional time dependent systems, the forward-backward workflow of solving the AE imposes a significant storage cost since the AE must be solved backwards in time [62, 79]. Through hyper-parameter tuning we found that the number of parameters $\mathcal{O}(1) \lesssim p \lesssim \mathcal{O}(9 * 10^2)$ for the best fit models within the hierarchy, and so for $N \gtrsim 1000$, the dimension of the system is much larger than p so FSA is more efficient, and therefore FSA forms our main structure to compute the gradient. The gradient is computed over all parameters found from using SA over all particles; alternatively stochastic gradient descent (SGD) could be used, however, for each batch all the particle would need to be simulated forward in time (since each particle is interacting with all neighboring particles) requiring a full forward solve for each batch, defeating any computational advantages of SGD.

e. Mixed Mode AD

In both FSA and ASA described above, the gradient of \mathcal{F}_i with respect to the parameters, $\partial \mathcal{F}_i(\mathbf{X}(\tau), \boldsymbol{\theta}) / \partial \boldsymbol{\theta}$, and the Jacobian matrix, $\{\partial \mathcal{F}_i(\mathbf{X}(\tau), \boldsymbol{\theta}) / \partial \mathbf{X}_j | \nabla_i, j\}$, need to be computed. In this manuscript, we accomplish this with a mixed mode approach, i.e. mixing forward and reverse mode AD within the FSA framework, where the choice is based on efficiency. This is determined by the input and output dimensions of the function being differentiated. Depending on the above model used, several functions need to be differentiated and a mixture of forward and reverse mode can be implemented within the FSA system for optimizing efficiency. For example, when computing $\partial \mathcal{F}_i(\mathbf{X}(\tau), \boldsymbol{\theta}) / \partial \theta^\alpha$, with AD where, $\mathcal{F}_i(\boldsymbol{\theta}) : \mathbb{R}^p \rightarrow \mathbb{R}^{2d}$, if $p \gg 2d$, then reverse mode AD is more efficient than forward mode [47].

2. Forward Sensitivity Analysis

In general, the loss functions in this work can be defined as

$$L(\mathbf{X}, \boldsymbol{\theta}) = \int_0^{t_f} \Psi(\mathbf{X}, \boldsymbol{\theta}, t) dt.$$

The forward SA (FSA) approach simultaneously integrates the state variables along with their sensitivities (with respect to parameters) forward in time to compute the gradient of L ;

$$d_{\theta}L = \int_0^{t_f} \partial_{\mathbf{X}} \Psi(\mathbf{X}, \boldsymbol{\theta}, t) d_{\theta} \mathbf{X}(\boldsymbol{\theta}, t) + \partial_{\boldsymbol{\theta}} \Psi(\mathbf{X}, \boldsymbol{\theta}, t) dt. \quad (\text{B4})$$

Where, through using the chain rule, we see that the sensitivities of the state variables with respect to the model parameters ($d_{\theta} \mathbf{X}$) are required to compute the gradient of the loss. Assuming that the initial conditions of the state variables do not

depend on the parameters, then $\partial \mathbf{X}(0) / \partial \theta^\alpha = 0$. Now, define the sensitivities as $\mathbf{S}_i^k := d\mathbf{X}_i / d\theta^k$. Then, from Eq. (3) we derive

$$\frac{d\mathbf{S}_i^k}{dt} = \frac{d\mathcal{F}_i(\mathbf{X}(t), \boldsymbol{\theta})}{d\theta^k}, \quad (\text{B5})$$

then resulting, after applying the chain rule, in

$$\frac{d\mathbf{S}_i^k}{dt} = \frac{\partial \mathcal{F}_i(\mathbf{X}(t), \boldsymbol{\theta})}{\partial \mathbf{X}_i} \mathbf{S}_i^k + \frac{\partial \mathcal{F}_i(\mathbf{X}(t), \boldsymbol{\theta})}{\partial \theta^\alpha}. \quad (\text{B6})$$

Since the initial condition $\mathbf{X}(0)$ does not depend on $\boldsymbol{\theta}$, then $\mathbf{S}_i^k(0) = 0$. Now, computing the gradient of the loss function reduces to solving a forward in time Initial Value Problem (IVP) by integrating simultaneously the state variables \mathbf{X}_i defined in the main text, and sensitivities \mathbf{S}_i^k , defined in Eq. (B6).

In order to integrate Eq. (B6) the gradient of \mathcal{F}_i with respect to the parameters, both $\partial \mathcal{F}_i(\mathbf{X}(\tau), \boldsymbol{\theta}) / \partial \theta^\alpha$ and the Jacobian matrix, $\partial \mathcal{F}_i(\mathbf{X}(\tau), \boldsymbol{\theta}) / \partial \mathbf{X}_i$ need to be computed. In this work, this is done with a mixed mode approach. $\partial \mathcal{F}_i(\mathbf{X}(\tau), \boldsymbol{\theta}) / \partial \theta^\alpha$, with $\mathcal{F}_i(\boldsymbol{\theta}) : \mathbb{R}^k \rightarrow \mathbb{R}^{2d}$, is computed with AD (the choice of forward or reverse mode is determined by the dimension of the input and output space), where k is the number of parameters and d is the dimension. For example, if $k \gg 2d$ (as is the case when NNs are used), reverse mode AD is more efficient than forward mode [64]. The Jacobian matrix is computed and obtained through mixing symbolic differentiation packages (or analytically deriving by hand), as well as mixing AD. For example, when there are NNs used for the parameterization of the right hand side, then according to expression for the Jacobian from the main text, AD derivatives will need to be computed on different functions each with potentially different dimensions of input and output space. The AD packages used in this work were both ForwardDiff.jl [80] for forward mode and Zygote.jl [81] for reverse mode. The Jacobian matrix for 2D problems is

$$\frac{\partial \mathcal{F}_i(\mathbf{X}(t), \boldsymbol{\theta})}{\partial \mathbf{X}_i} = \begin{pmatrix} [0]_2 & I_2 \\ \partial \mathbf{F}_i(\mathbf{X}(t), \boldsymbol{\theta}) \\ \partial \mathbf{X}_i \end{pmatrix}$$

$$= \begin{pmatrix} [0]_2 & I_2 \\ \frac{\partial F_i^x}{\partial x_i^1} & \frac{\partial F_i^x}{\partial x_i^2} & \frac{\partial F_i^x}{\partial x_i^3} & \frac{\partial F_i^x}{\partial x_i^4} \\ \frac{\partial F_i^y}{\partial x_i^1} & \frac{\partial F_i^y}{\partial x_i^2} & \frac{\partial F_i^y}{\partial x_i^3} & \frac{\partial F_i^y}{\partial x_i^4} \end{pmatrix}.$$

where, the individual derivatives $\frac{\partial F_i^y}{\partial x_i^d}$ are computed with AD (Reverse or Forward mode depending on model chosen from Section III, and a similar formulation is carried out in 3D).

a. Adjoint Method

This section provides an outline of the Adjoint SA (ASA) method used in this work (for more details, see [82])

[62]). However, we note that the main results of the text did not require using the adjoint method because the FSA was found to be more efficient (since a relatively small number of parameters were needed compared to the number of particles as discussed in Section IV), but we include this as a reference to the associated source code that has the option of using ASA in case the number of parameters becomes large enough ($p \gg 2 * D * N$). Again, the goal is to compute the gradient of the loss function. This is a continuous time dependent formulation, where the goal is to minimize a loss function $L(\mathbf{X}(\boldsymbol{\theta}, t), \boldsymbol{\theta})$ which is integrated over time, $L(\mathbf{X}, \boldsymbol{\theta}) = \int_0^{t_f} \Psi(\mathbf{X}, \boldsymbol{\theta}, t) dt$, subject to the physical structure constraints (ODE or PDE), $H(\mathbf{X}, \dot{\mathbf{X}}, \boldsymbol{\theta}, t) = 0$, and the dependence of the initial condition, $g(\mathbf{X}(0), \boldsymbol{\theta}) = 0$, on parameters. Here, H is the explicit ODE form obtained through the SPH discretization equations Eq. (3) (discretization of PDE flow equations),

$$H(\mathbf{X}, \dot{\mathbf{X}}, \boldsymbol{\theta}, t) = \dot{\mathbf{X}}(t) - \mathcal{F}(\mathbf{X}(t), \boldsymbol{\theta}). \quad (\text{B7})$$

A gradient based optimization algorithm requires that the gradient of the loss function,

$$d_{\boldsymbol{\theta}} L(\mathbf{X}, \boldsymbol{\theta}) = \int_0^{t_f} \partial_{\mathbf{X}} \Psi(\mathbf{X}, \boldsymbol{\theta}, t) d_{\boldsymbol{\theta}} \mathbf{X}(\boldsymbol{\theta}, t) + \partial_{\boldsymbol{\theta}} \Psi(\mathbf{X}, \boldsymbol{\theta}, t) dt,$$

be computed. The main difference in the FSA and ASA approach is that in the ASA calculating $d_{\boldsymbol{\theta}} \mathbf{X}$ is not required (which avoids integrating the additional k ODEs as in FSA). Instead, the adjoint method develops a second ODE (size of which is independent of k) in the adjoint variable $\boldsymbol{\lambda}$ as a function of time (which is then integrated backwards in time).

The following provides a Lagrange multiplier approach to deriving this ODE in $\boldsymbol{\lambda}$. First define

$$\begin{aligned} \mathcal{L} = & \int_0^{t_f} (\Psi(\mathbf{X}, \boldsymbol{\theta}, t) + \boldsymbol{\lambda}^T(t) H(\mathbf{X}, \dot{\mathbf{X}}, \boldsymbol{\theta}, t)) dt \\ & + \boldsymbol{\mu}^T g(\mathbf{X}(0), \boldsymbol{\theta}), \end{aligned} \quad (\text{B8})$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the Lagrange multipliers. Now, since H and g are zero everywhere, we may choose the values of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ arbitrarily; also as a consequence of H and g being everywhere zero, we note that $\nabla_{\boldsymbol{\theta}} \mathcal{L} = \nabla_{\boldsymbol{\theta}} L$.

Now, taking the gradient of Eq. (B8), and simplifying

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L} = & \int_0^{t_f} (\partial_{\mathbf{X}} \Psi d_{\boldsymbol{\theta}} \mathbf{X} + \partial_{\boldsymbol{\theta}} \Psi + \boldsymbol{\lambda}^T (\partial_{\mathbf{X}} H d_{\boldsymbol{\theta}} \mathbf{X} \\ & + \partial_{\dot{\mathbf{X}}} H d_{\boldsymbol{\theta}} \dot{\mathbf{X}} + \partial_{\boldsymbol{\theta}} H)) dt \\ & + \boldsymbol{\mu}^T (\partial_{\mathbf{X}(0)} g d_{\boldsymbol{\theta}} \mathbf{X}(0) + \partial_{\boldsymbol{\theta}} g). \end{aligned}$$

Integrating the equation by parts, and eliminating, $d_{\boldsymbol{\theta}} \dot{\mathbf{X}}$, we arrive at

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L} = & \int_0^{t_f} \left[(\partial_{\mathbf{X}} \Psi + \boldsymbol{\lambda}^T (\partial_{\mathbf{X}} H - d_t \partial_{\dot{\mathbf{X}}} H) - \dot{\boldsymbol{\lambda}}^T \partial_{\dot{\mathbf{X}}} H) d_{\boldsymbol{\theta}} \mathbf{X} \right. \\ & \left. + \partial_{\boldsymbol{\theta}} \Psi + \boldsymbol{\lambda}^T \partial_{\boldsymbol{\theta}} H \right] dt + \boldsymbol{\lambda}^T \partial_{\dot{\mathbf{X}}} H d_{\boldsymbol{\theta}} \mathbf{X} \Big|_{t_f} \\ & + (-\boldsymbol{\lambda}^T \partial_{\dot{\mathbf{X}}} H + \boldsymbol{\mu}^T g) \Big|_0 d_{\boldsymbol{\theta}} \mathbf{X}(0) + \boldsymbol{\mu}^T \partial_{\boldsymbol{\theta}} g. \end{aligned}$$

Since the choice of, $\boldsymbol{\lambda}^T$, and, $\boldsymbol{\mu}$, is arbitrary, we set, $\boldsymbol{\lambda}^T(T) = 0$, and, $\boldsymbol{\mu}^T = (\boldsymbol{\lambda}^T \partial_{\dot{\mathbf{X}}} H)|_0 (g|_{\mathbf{X}(0)})^{-1}$, in order to avoid needing to compute, $d_{\boldsymbol{\theta}} \mathbf{X}(T)$, and thus canceling the second to the last term in the latest (inline) expression. Now, assuming that the initial values of the state variables, $\mathbf{X}(0)$, do not depend on the parameters, then $d_{\boldsymbol{\theta}} \mathbf{X}(0) = 0$, and, $g = 0$. Furthermore, we use a loss function Ψ that does not depend on $\boldsymbol{\theta}$ explicitly, so that, $\partial_{\boldsymbol{\theta}} \Psi = 0$. And finally, we can avoid computing $d_{\boldsymbol{\theta}} \mathbf{X}$ at all other times $t > 0$ by setting

$$\partial_{\mathbf{X}} \Psi + \boldsymbol{\lambda}^T (\partial_{\mathbf{X}} H - d_t \partial_{\dot{\mathbf{X}}} H) - \dot{\boldsymbol{\lambda}}^T \partial_{\dot{\mathbf{X}}} H = 0.$$

The resulting equation for the time derivative of $\boldsymbol{\lambda}$ can be restated as the following Adjoint Equation (AE)

$$\dot{\boldsymbol{\lambda}}^T = \partial_{\mathbf{X}} \Psi - \boldsymbol{\lambda}^T \frac{\partial \mathcal{F}}{\partial \mathbf{X}}, \quad \boldsymbol{\lambda}(t_f) = 0, \quad (\text{B9})$$

where we also used that according to Eq. (B7), $\partial_{\mathbf{X}} H = -\partial_{\mathbf{X}} \mathcal{F}$ and $\partial_{\dot{\mathbf{X}}} H = I_{2d}$.

Combining all of the above, we see that the simplified equation for the gradient of the loss function is

$$\nabla_{\boldsymbol{\theta}} L = \nabla_{\boldsymbol{\theta}} \mathcal{L} = \int_0^{t_f} \boldsymbol{\lambda}^T \partial_{\boldsymbol{\theta}} H dt.$$

Which, according to Eq. (B7), becomes

$$\nabla_{\boldsymbol{\theta}} L = - \int_0^{t_f} \boldsymbol{\lambda}^T \frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}} dt.$$

Therefore, in order to compute the gradient, the IVP expression Eq. (B9) needs to be integrated backwards in time for $\boldsymbol{\lambda}(t)$, starting from $\boldsymbol{\lambda}(t_f) = 0$. Similar to the FSA formulation found above, both $\partial_{\mathbf{X}} \mathcal{F}$ and $\partial_{\boldsymbol{\theta}} \mathcal{F}$, are computed with a mixture of forward and reverse mode AD tools, depending on the dimension of the input and output dimension of the functions to be differentiated.

3. Learning Algorithm

In what follows, we combine all of the computational tools and techniques introduced so far to outline the mixed mode learning algorithm.

Algorithm 1 Mixed Mode Learning Algorithm

-
- 1: Given Ground Truth: $\{\mathbf{X}(t_0), \mathbf{X}(t_1), \dots, \mathbf{X}(t_f)\}$ SPH data
 - 2: Select model: $d_t \mathbf{X}(t) = \mathcal{F}(\mathbf{X}(t), \boldsymbol{\theta})$ from hierarchy
 - 3:
 - 4: Select SA method (Forward or Adjoint)
 - 5:
 - 6: Choose Loss methods (combination of Trajectory, Field, or Probabilistic)
 - 7:
 - 8: Choose optimizer: e.g $Opt = \{RMSprop, ADAM\}$
 - 9:
-
- 10:
 - 11: **for** $k \leftarrow 1, \dots, n$ **do**
 - 12: Prediction step: Verlet integration of model $\hat{\mathbf{X}} = Verlet(\mathcal{F}, \boldsymbol{\theta}_k, \mathbf{X}(t_0), t_0, t_f)$
 - 13:
 - 14: Simultaneously compute $\frac{\partial \mathcal{F}}{\partial \hat{\mathbf{X}}_i}, \frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}}$ with mixed mode AD.
 - 15:
 - 16: Simultaneously integrate system of ODEs for sensitivities \mathbf{S}_i^α ;
 - 17:
 - 18:
$$d_t \mathbf{S}_i = \frac{\partial \mathcal{F}_i(\hat{\mathbf{X}}(t), \boldsymbol{\theta}_k)}{\partial \hat{\mathbf{X}}_i} \mathbf{S}_i + \frac{\partial \mathcal{F}_i(\hat{\mathbf{X}}(t), \boldsymbol{\theta}_k)}{\partial \boldsymbol{\theta}_k}$$
 - 19: Compute ∇L
 - 20: **end for**
 - 21: update $\boldsymbol{\theta}$ using optimizer;
 - 22:
 - 23: $\boldsymbol{\theta}_{k+1} = Opt(\boldsymbol{\theta}_k)$
-

Appendix C: Evaluating models: additional results

In this section, we include additional results of the trained models, for example using the combination of L_f and L_{kl} .

1. Comparing Smoothing Kernels

In Fig. 5 we compare four different smoothing kernels included when learning the fully parameterized SPH model; two standard and fixed smoothing kernels, namely the cubic Eq. (A1) and quartic Eq. (A2) splines (which follow the Gaussian bell shape); and two novel parameterized smoothing kernels that are learned from DNS data when training the fully parameterized SPH-informed model Eq. (10). We provide numerical evidence that shows including these novel parameterized smoothing kernels in training improves the ability of the SPH informed model to solve the interpolation problem and generalize to flows not seen in training as in Fig. 14. This result seems to suggest that an improvement to the SPH construction can be made, at least in the weakly compressible setting at the resolved scales as seen in this work, by considering smoothing kernels which are not of the classical bell shape,

but rather of the more flexible and parameterized form proposed in Eq. (11) and Eq. (12). Future work could be done in this direction to further analyze and compare these new smoothing kernels with regards to convergence and consistency. We further compare these smoothing kernels from the trained SPH models by making forward predictions and evaluating the acceleration statistics Fig. 9 and energy spectrum Fig. 15 and single particle statistics Fig. 16; from which we can conclude that the new parameterized smoothing kernels, especially W_2 , has the best generalizability, i.e improves the accuracy of SPH over the standard cubic and quartic smoothing kernels to match DNS at different time scales and turbulent Mach numbers with respect to field based and statistical measures. The learned parameters are tabulated in Fig. IV.

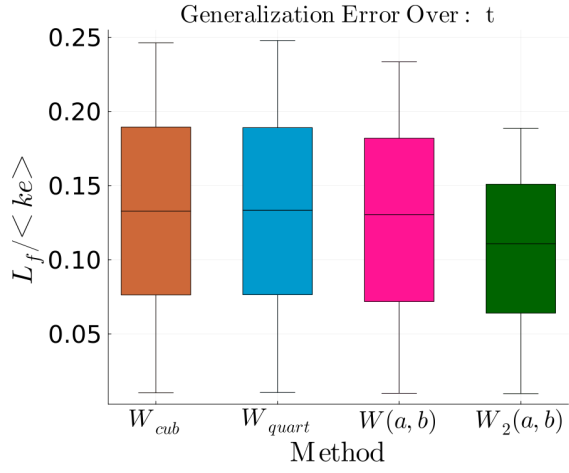
The learned parameters found from training the SPH based models are found in Table IV.

TABLE IV: Learned SPH parameters

Model	\hat{c}	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\gamma}$	$\hat{\rho}_0$	$\hat{\theta}$	\hat{a}	\hat{b}
SPH-informed: W_{cubic}	0.55	0.19	-0.29	1.69	-0.3	0.0013		
SPH-informed: $W_{quartic}$	0.53	0.19	-0.25	1.82	-0.28	0.0013		
SPH-informed: $W(a, b)$	0.48	0.21	0.62	2.05	-0.24	0.0009	2.82	3.05
SPH-informed: $W_2(a, b)$	0.50	0.19	0.44	2.05	-0.26	0.0006	1.43	0.53

2. Generalizability: Training with Statistical Loss

In this section we show numerical evidence of the generalizability of the models when using the L_{kl} defined above. This is not reported in the main text as there are only slight differences seen in the results. However, the largest difference is seen in Fig. 19 when compared to Fig. 6, namely, the generalization error in the rotational invariant model (when measured with $L_f / \langle ke \rangle$) has improved by roughly 50%. The first part of training is guided by L_f , then switched to L_{kl} in order to first guide the parameters to reproduce large scale structures, then converge the loss using the L_{kl} in order to learn the small scale characteristics seen in the velocity increment statistics. Furthermore, this was done in order to drive the distributions close enough to avoid large sensitivities of the KL divergence with respect to the predicted and target distributions. In what is shown below, we see that by using the statistics based loss, the models perform better at generalizing with respect to the single particle statistics (as seen in Fig. 20 when only L_f was used. However, now the energy spectrum of each model performs worse at generalizing to longer time scales. This could be due to the fact that the models are overfit to reproduce the velocity increment of the DNS particles and degrade with respect to other measures. This suggests that the field based loss is a better choice to capture the energy spectrum, and the L_{kl} is better to improve generalizability with respect to single particle statistics. The main conclusion between experimenting with both loss formulations is that the SPH based parameterization using W_2 performs best with respect to both statistical and field based measures at generalizing over longer time scales and different Mach numbers.



(a) Over time

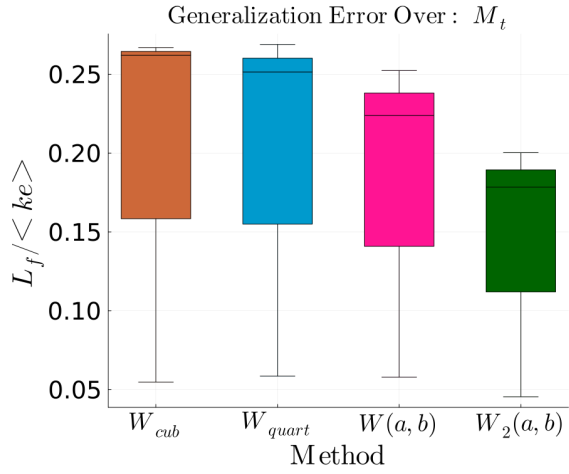
(b) Over M_t

FIG. 14: (a) Measuring (using the L_f normalized with the average kinetic energy) the generalization error over (a) 20 longer time scales up to the eddy turn over time and (b) over 3 different $M_t \in \{0.04, 0.08, 0.16\}$ at the eddy turn over time scale of each fully parameterized SPH informed models, comparing two new parameterized smoothing kernels with two classic smoothing kernels. This shows that the new parameterized smoothing kernels outperform in their ability to generalize to other DNS flows as compared to the the cubic and quartic at this spatial resolution using $N = 4096$ particles.

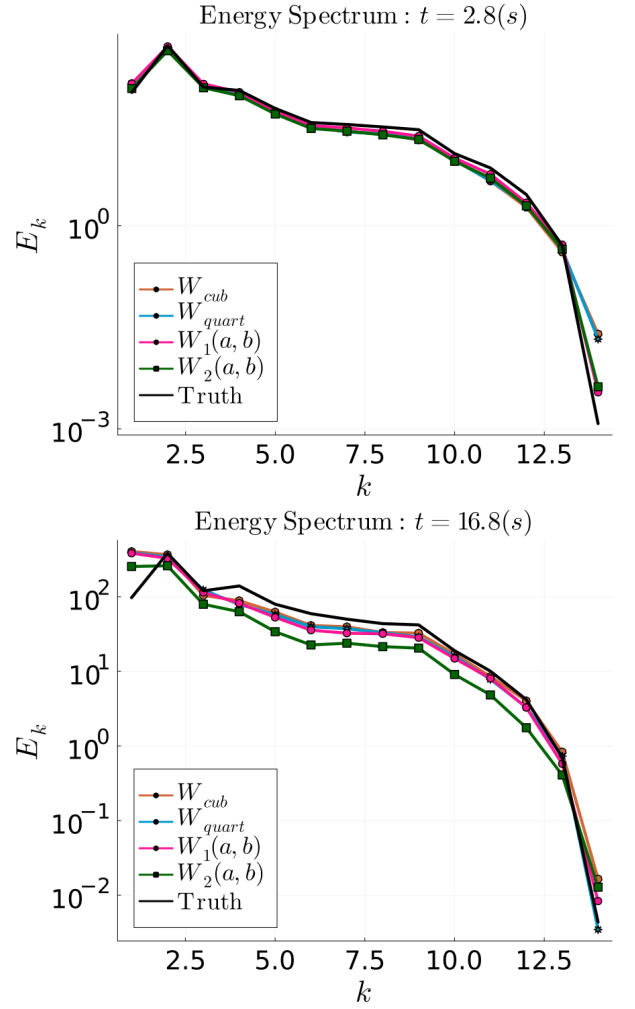


FIG. 15: Plotting Energy spectrum over different time scales and comparing each smoothing kernel. Each model does well at capturing energy spectrum on the Kolmogorov time scale, however, on the eddy turn over time, the quartic and $W(a,b)$ seem to perform marginally better.

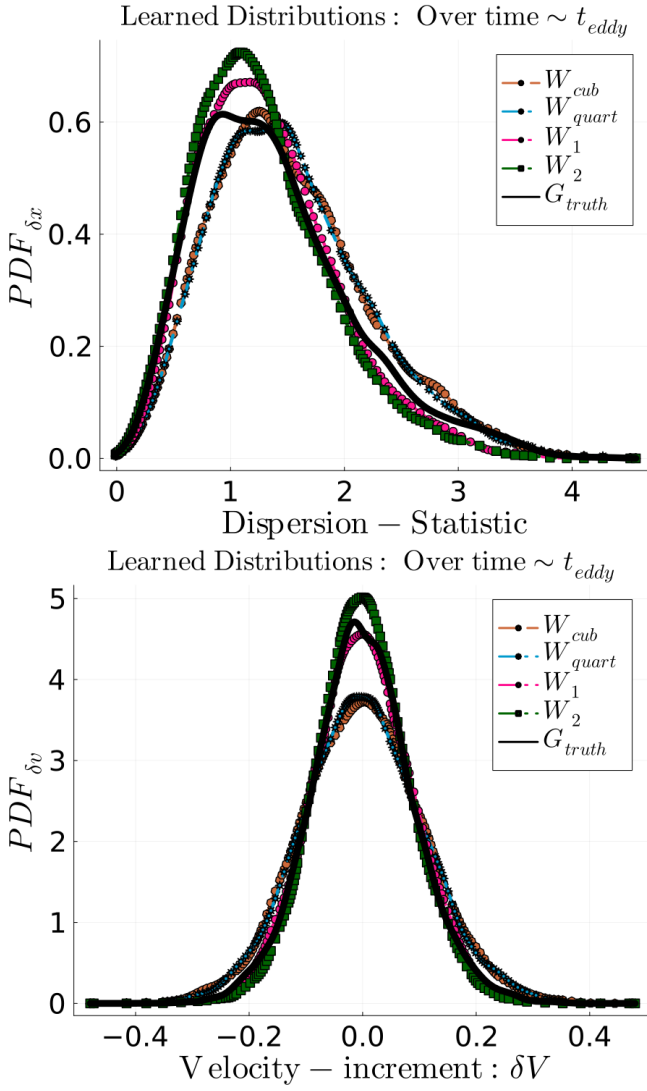


FIG. 16: A diagnostic check comparing single particle statistics on larger time scales (roughly 20 times longer than seen in training), with the new parameterized smoothing kernels SPH-informed model having the best fit. Each is not seen in training, so represent an external diagnostic.

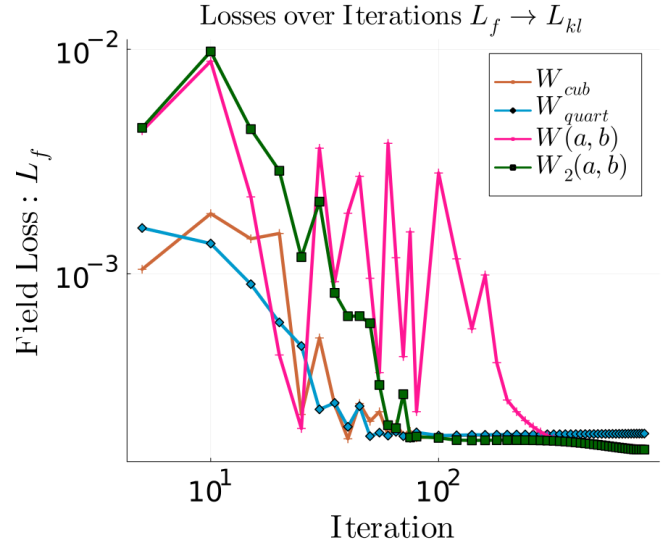


FIG. 17: The field based loss converging, using L_f to guide large scale structures then switching to L_{kl} to capture small scale statistical characteristics found the velocity increment distribution.

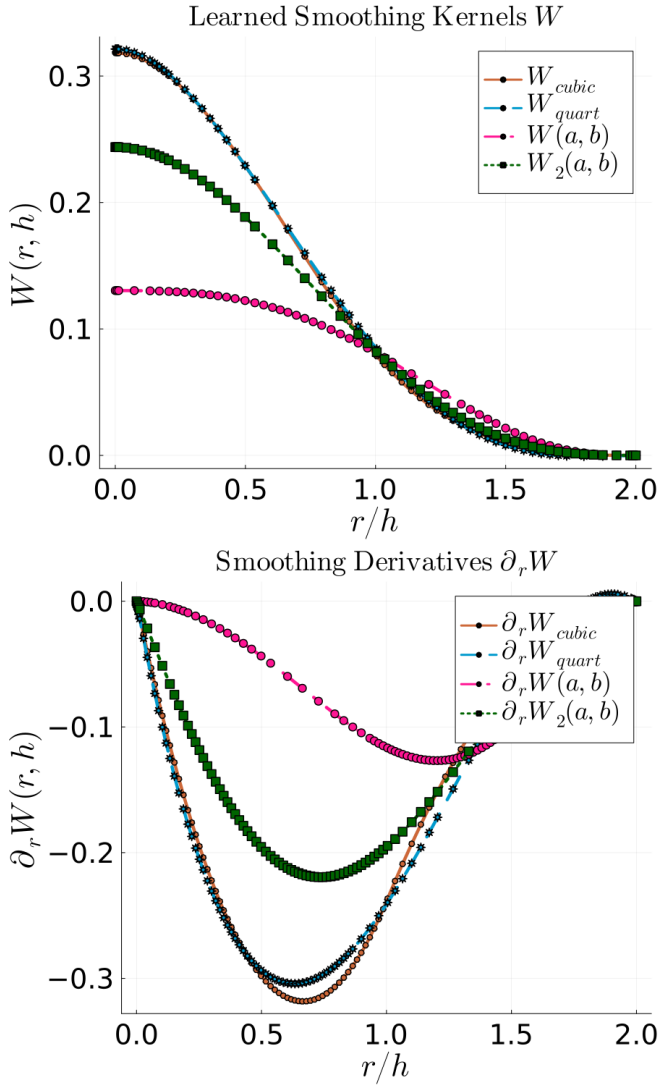


FIG. 18: Learned parameterized smoothing kernels compared to cubic and quartic smoothing kernels.

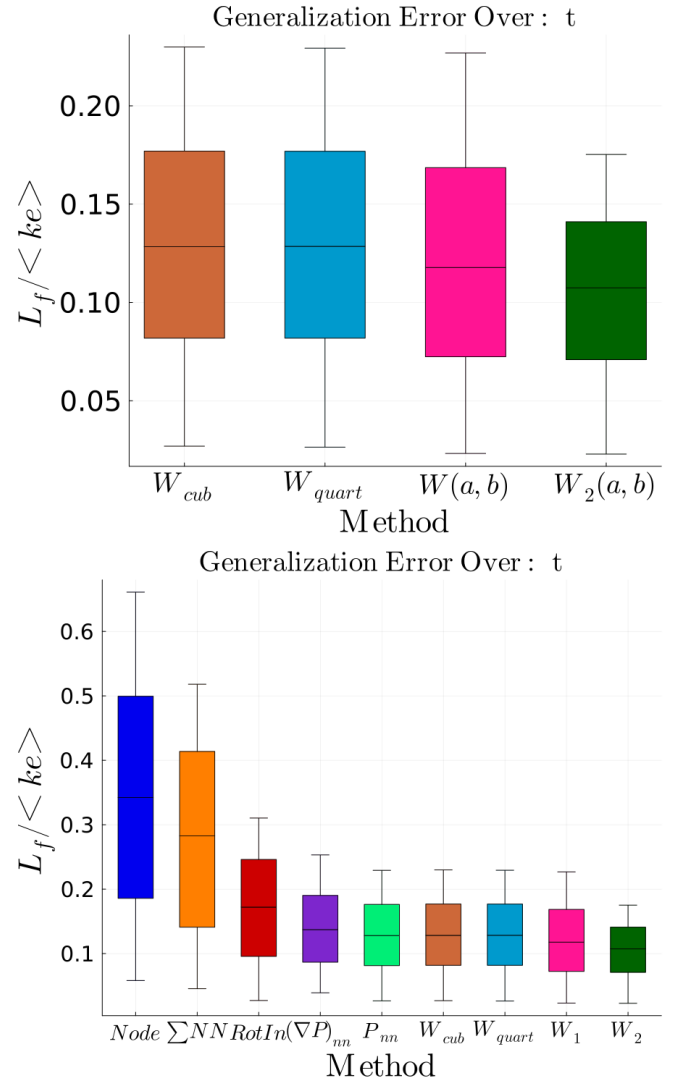


FIG. 19: Measuring the relative error of L_f as a percentage of total kinetic energy. The generalization error over M_t is computed over 3 different turbulent Mach numbers $M_t \in \{0.04, 0.08, 0.16\}$ and on the order of the eddy turn over time scale. We see that the parameterized smoothing kernel W_2 outperforms under this field based measure.

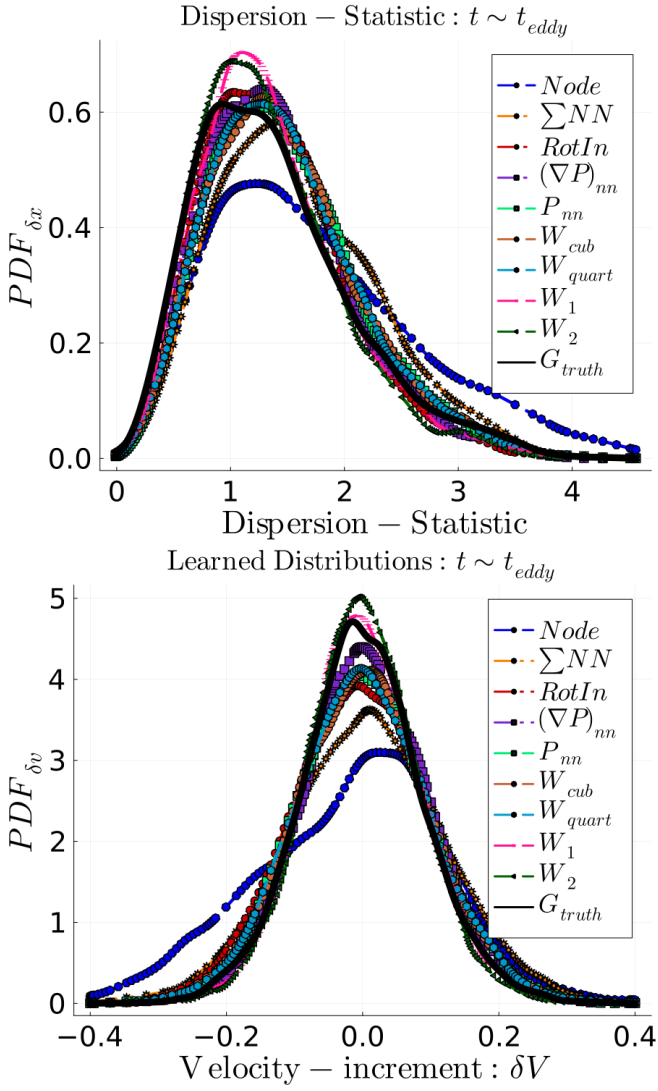


FIG. 20: Single particle statistics on larger time scales (roughly 20 times longer than seen in training), with the new parameterized smoothing kernels having the best fit. In this case, the velocity increment distribution is used in learning with L_{kl} .

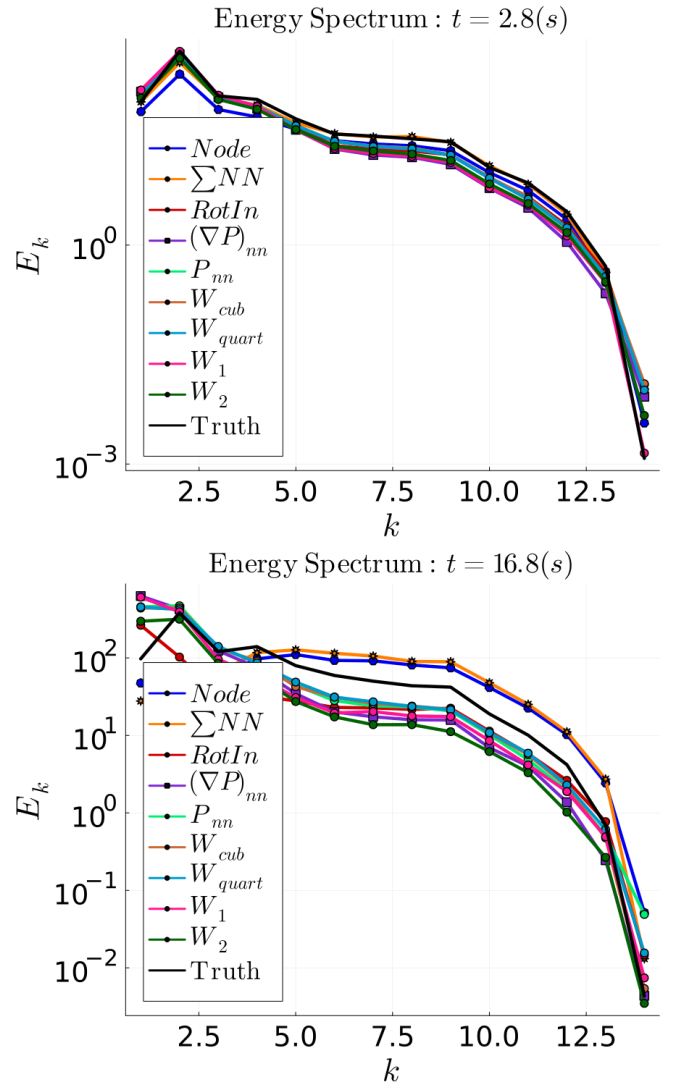


FIG. 21: Comparing energy spectrum over time; $t = 2.8s \sim t_\eta$ is the Kolmogorov time scale, and $t = 16.8s \sim t_{eddy}$ is at the scale of the eddy turn over time. We see that when, using L_f then L_{kl} , as more SPH structure is included in the models the better it is at capturing energy contained in the large scales, however the less informed models overestimate the energy contained in small scales and the SPH-informed models underestimate the energy contained in the small scales.

Appendix D: Validating methodology: Training and Evaluating Models on SPH data

In this Section, we test the learning algorithm by training the hierarchy of models on "synthetic" SPH data, and see if the learned parameters of the fully parameterized SPH model converge to the true parameters, as well as test if the other models can interpolate onto the flow. Furthermore, we demonstrate the effects of incrementally embedding physical structure into the Lagrangian models. We show the ability of the mixed mode FSA + AD method to learn the parameterized Lagrangian and SPH based simulators on SPH flow data (as seen in Section II) with the field based loss and a mixture of field based and statistical based loss functions. We show that each model is capable of solving the interpolation problem, which we evaluate using several quantitative and qualitative diagnostics, but as more physical structure is embedded in the model, the better is the generalizability to flows not seen in training and the better it conserves linear and angular momentum.

Each parameterized Lagrangian model within the hierarchy (see Section III) is trained under equivalent conditions: (a) on the same SPH samples (see Section II) of fixed temporal duration (which we choose to be equal to the time scale required for a pair of neighboring particles to separate by the distance which is on average a factor of $O(1)$ larger than the pair's initial separation and henceforth denoted t_λ); (b) with the same loss function $L_{kl} + L_f$; (c) with the FSA method; and (d) with a deterministic forcing f_{ext} (Section A 3 with constant rate of energy injection).

1. Interpolation of Models: SPH data

In Fig. 22 (and Fig. 23) we see that the physics informed parameterized SPH simulator is learn-able; the estimated physical parameters $\hat{\alpha}, \hat{\beta}, \hat{c}, \hat{\gamma}$, and energy injection rate $\hat{\theta}_{inj}$ converging to the true values where the initial guess for each parameter is uniformly distributed about $(0, 1)$. Furthermore, these figures show that the Weakly compressible SPH flow is more sensitive to the parameters α and β , which control the strength of the artificial viscosity term, compared to parameters c and γ , which control the local slope and shape of the EoS. This is due to the weakly compressible nature of the flow being less sensitive to changes in the speed of sound. Fig. 2 illustrates the ability of the mixed mode method applied to the EoS-NN model to learn (approximate) physically interpretable functions (namely the barotropic equation of state $P(\rho)$) using NNs embedded within an SPH model. However, we notice that the learned EoS $P_{mn}(\rho)$ begins to deviate from the $P_{truth}(\rho)$ outside the domain of densities that are seen in training.

In Fig. 24, we see that each model is capable of learning the parameters so that the underlying velocity increment distributions are approximated on the time scale on which learning takes place. Furthermore, In Fig. 25 we test the trained models on their ability to reproduce the distribution associated with the single particle dispersion statistic [65] (showing how far the particle disperses from its initial condition on a

set time scale). Here it is important to note that the dispersion statistic was not enforced in the L_{kl} loss, and so Fig. 24 illustrates an external diagnostic test. From this we find that without enforcing the dispersion statistic in the loss function, the less informed models (namely NODE, and NN summand) fail to reproduce the true dispersion statistics on both time scales (t_λ , and t_{eddy}).

These Figures show that the Lagrangian models introduced in Section III are learn-able (i.e. interpolated on the SPH training set). We arrive at these results applying the methods described in Section B 1 d to a mixture of the loss functions $L_{kl} + L_f$ introduced in Section B 1.

2. Generalizability (extrapolation capability) on SPH flow data

When the training is complete (i.e. when the loss function reaches its minimum, and conditions are set as described in Section D) we validate extrapolation capability of the models on a validation set, and test data set. The validation data set corresponds to data generated from the same initial conditions but are longer in duration (up to the eddy turn over time). The test set corresponds to flow derived from the setting corresponding to stronger turbulent Mach numbers (which we control by increasing intensity of the injection term, f_{ext} , while keeping the integral, i.e. energy injection scale, constant, thus increasing the turbulent mach number, M_t).

TABLE V: Errors in Rotational and Translational Symmetries on trained models

Trained Model	Rotational Error	Translational Error
NODE	4.59	6.57
NN Sum	5.42	1.51
Rot Inv	1.21×10^{-27}	4.22×10^{-27}
∇P	1.20×10^{-2}	1.48×10^{-27}
EoS NN	3.49×10^{-28}	1.18×10^{-27}
Phys Inf	2.31×10^{-27}	7.88×10^{-27}

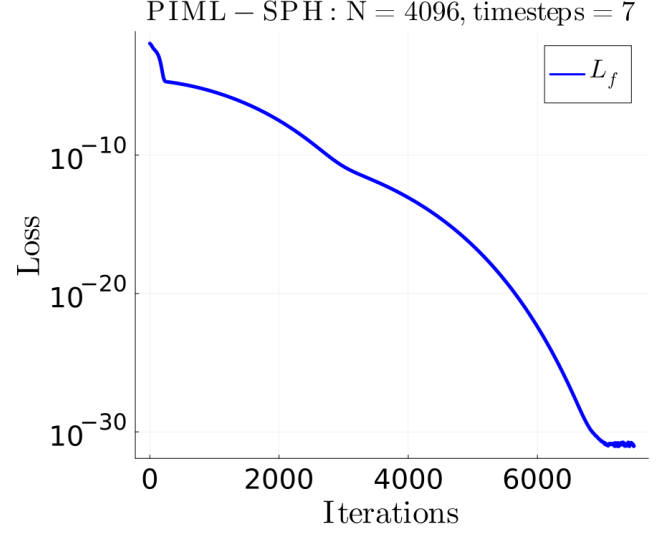
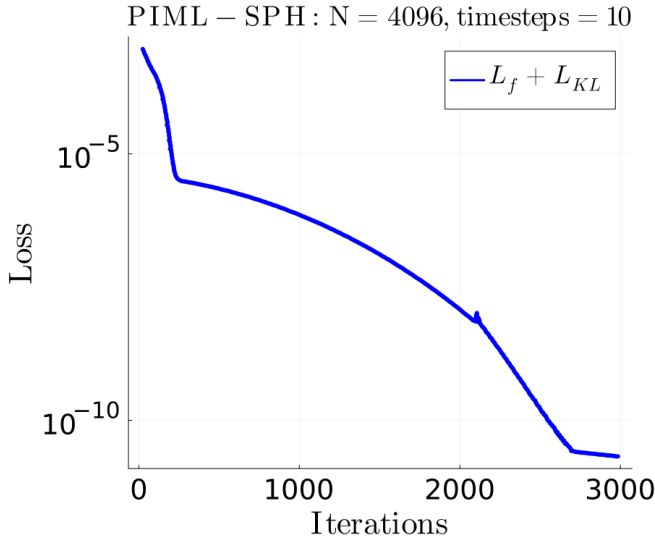
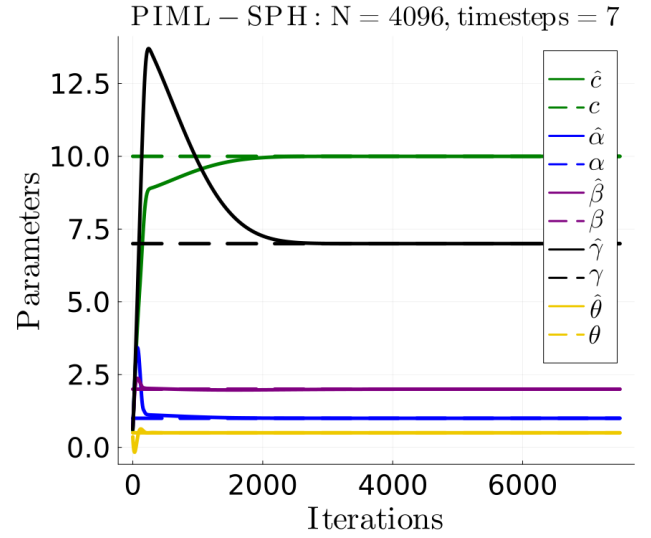
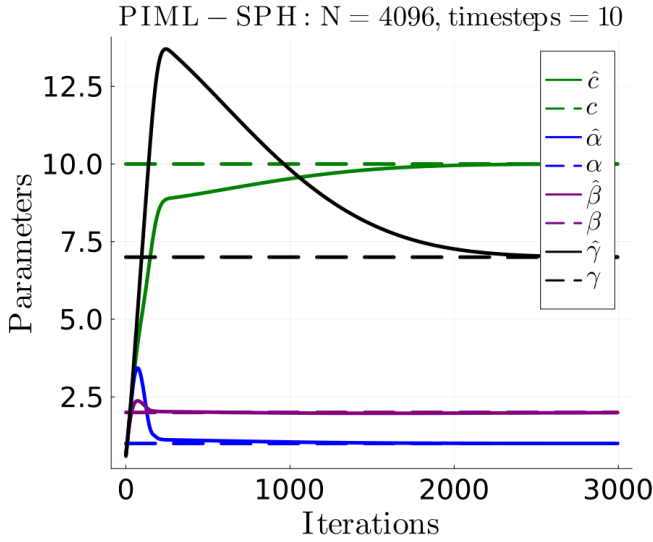


FIG. 22: Solving an inverse problem for the fully physics informed model on 3D SPH flow with deterministic external forcing on 4096 particles over the SPH based parameters. The solid lines show the SPH model parameters (initially chosen to be uniformly distributed about $(0,1)$) converging to the dashed lines representing the ground truth parameters. Here the $L_{kl} + L_f$ loss function (see Section B 1) is used (where L_f is used in pre-training up to 2100 iterations as seen in the small increase in the loss as the L_{kl} is added in).

FIG. 23: Solving the inverse problems for 3D SPH flow using L_f with a linear deterministic external forcing on 4096 particles over physical parameters including the rate of energy injection θ_{inj} . Notice the field based loss (using MSE) converging to machine precision, validating the learning algorithm and its stability. We also note that the smoothness of the convergence of the loss is most likely due to the relatively small number of parameters in this constrained optimization problem leading to a smooth loss landscape with an identifiable global optimum within a reasonable neighborhood of initial parameters. For example, when NNs were embedded within the ODE structure, the loss functions were observed to have some fluctuation along a downward trend towards convergence.

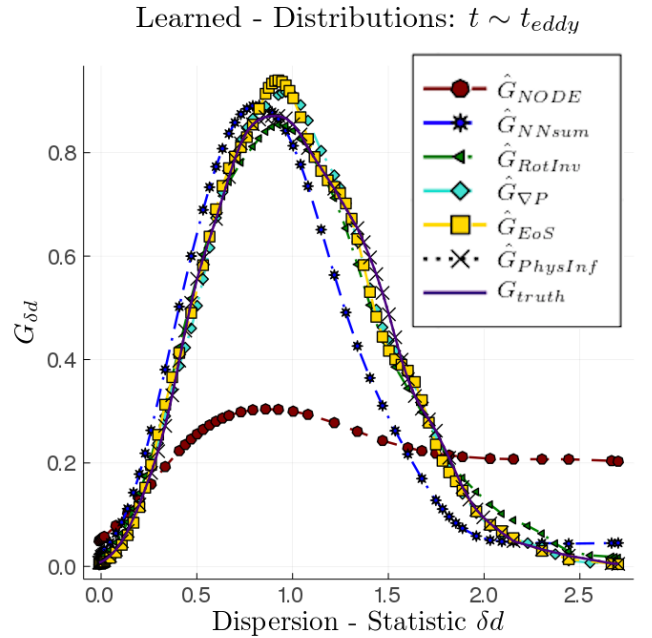
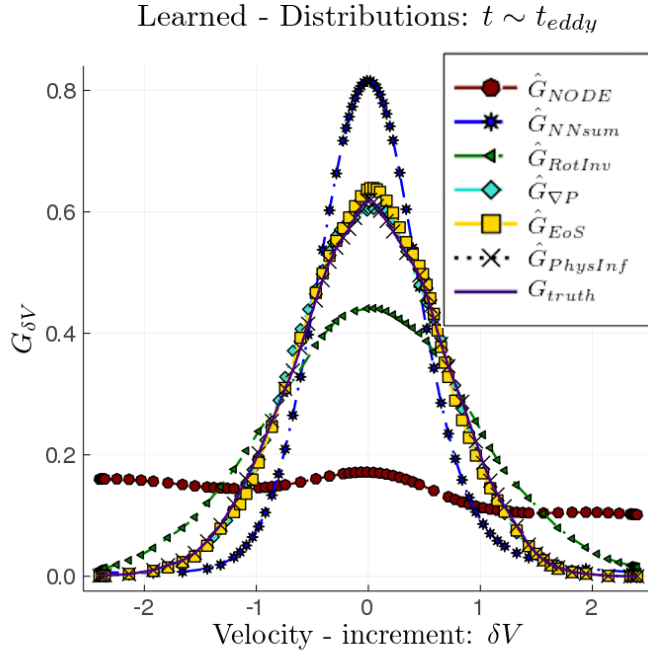
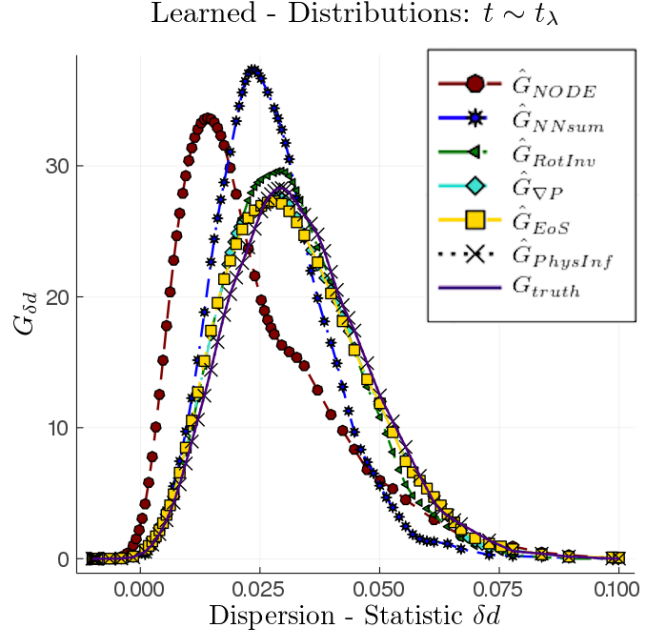
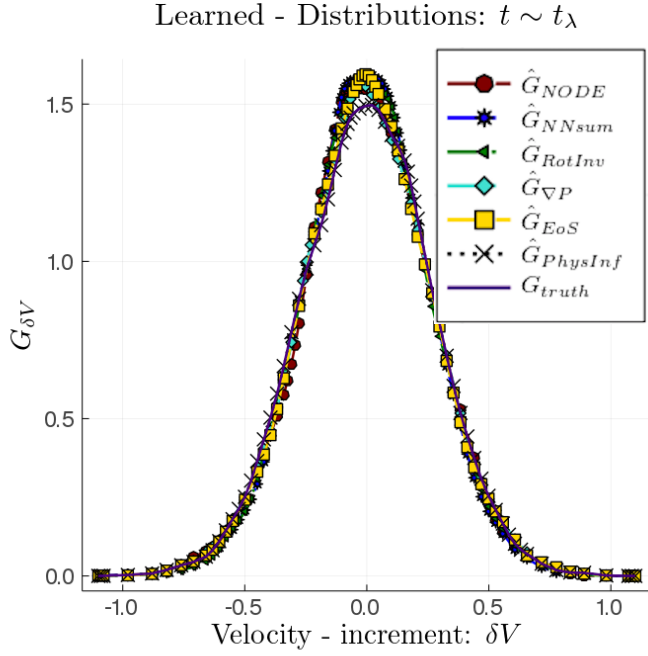


FIG. 24: The velocity increment distribution learned over each model in the hierarchy Section III on the time scale t_λ using the $L_{kl} + L_f$ loss function and the FSA method. We see that, on the learned time scale, all the models do well at capturing the small scale Lagrangian velocity increment statistics, however, only the more physics informed models do well at generalizing to larger time scales not seen in training.

FIG. 25: The single particle dispersion statistic distribution being used as a diagnostic check on each model in the hierarchy Section III on the learned time scale and on the eddy turn over time scale

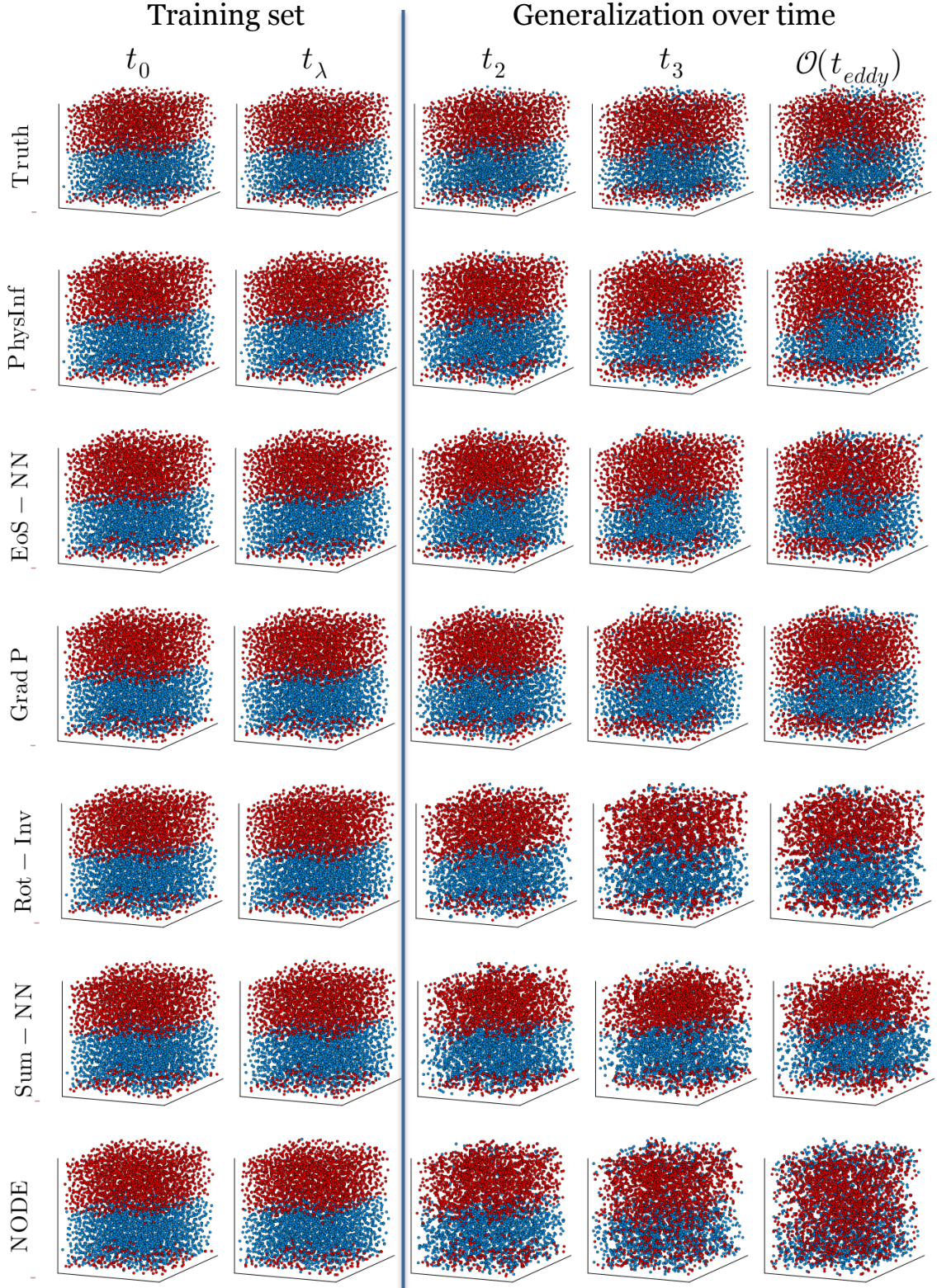


FIG. 26: Snapshots of SPH particles evolving, comparing the ground truth SPH data to the learned models, where learning is done with the $L_{KL} + L_f$ loss function. Even though the training is occurring on the shortest (physically relevant) time scale t_λ the more physics informed models are able to generalize to much longer time scales – all the way to the longest (physically relevant) time scale, t_{eddy} (which is the turnover time scale of the largest eddy of the flow).