PhyFlow: Physics-Guided Deep Learning for Generating Interpretable 3D Flow Fields

Nikhil Muralidhar*, Jie Bu*, Ze Cao†, Neil Raj†, Naren Ramakrishnan*, Danesh Tafti† and Anuj Karpatne*
*Department of Computer Science, Virginia Tech
†Department of Mechanical Engineering, Virginia Tech
Email: {nik90, jayroxis, zec1, neilashwinraj, naren, dtafti, karpatne} @vt.edu

***Abstract*—Generating flow fields (such as pressure and velocity fields) in 3D space is a fundamental task in computational fluid dynamics (CFD), with applications across a vast spectrum of science and engineering problems. An important class of fluid flow problems in CFD is multi-phase flow, where dispersed solid particles are present in the fluid flow. Despite recent developments in deep learning (DL) for CFD applications, current state-of-the-art is still unable to model 3D flow fields, especially in multi-phase flow settings. It is with this goal that we introduce PhyFlow, a novel physics-guided deep learning architecture for modeling 3D multi-phase fluid flows, designed to mimic the popular *projection* method for solving fluid flows in CFD simulations. We demonstrate that PhyFlow generates high quality flow fields and yields a 49.61% improvement over other state-of-the-art baselines. We also test the quality of PhyFlow based fields by employing them in downstream tasks like particle drag force prediction and demonstrate state-of-the-art results, improving upon the previous best models by 9.89%. Finally, we demonstrate the consistency of PhyFlow predictions with known underlying physics governing equations. Our source code and data are available online*.**

***Index Terms*—Physics-guided ML, Deep Learning, CFD**

I. Introduction

Generating flow fields (such as pressure and velocity fields) in 3D space is a fundamental task in computational fluid dynamics (CFD), with applications across a vast spectrum of science and engineering problems. For example, CFD simulations of flow fields are used for optimizing the aerodynamic designs of automobiles and aircrafts [1], modeling the evolution of climate/weather patterns as well as disaster events like wildfires [2], and more recently, to even model blood flow in the heart to detect irregularities [3].

An important class of fluid flow problems is *multi-phase flow*, where the moving fluid contains dispersed elements of a different phase (e.g., blood flow composed of plasma with solid cells dispersed within the plasma). While multi-phase flows are widely found in nature and used heavily in a number of industrial applications (e.g., in propulsion, energy, pharmaceutical, and food processing sectors), they are non-trivial to model due to the additional complexity caused by the interaction of different phases. One problem of interest in the family of multi-phase flows is the prediction of flow fields around a random arrangement of particles, which can be used for downstream tasks such as predicting fluid forces (e.g., drag forces) acting on the particles.

*tinyurl.com/mjkcrsdw

Despite the long history of methodological advances in the field of CFD, exact physics-based solvers of flow fields are highly expensive to deploy in most practical settings. Hence, a *physics-only* solution to generating multi-phase flow fields is intractable for large systems. Recently, machine learning (ML) methods have started to gain attention for modeling highly complex problems in a wide range of scientific disciplines [4], [5], including recent applications in CFD [6]–[10].

However, current state-of-the-art methods in deep learning for CFD are still unable to model 3D flow fields (a majority of works only solve 2D flows using numerical simplifications that do not work in 3D), especially in multi-phase flow settings where it is important to generalize to unseen particle arrangements using a small number of training arrangements, which is the focus of this work. What is needed is to move beyond black-box applications of DL and principally leverage physics in the design, training, and evaluation of DL models for multi-phase flow generation problems. Our proposed physics-guided deep learning framework, termed PhyFlow, explicitly addresses this need by incorporating physics in a variety of ways. Figure 1 provides an overview of our target problem and proposed solution. Here is a summary of our main contributions:

(i) We introduce PhyFlow, a novel Physics-Guided Machine Learning (PGML) model for flow field generation that is able to generalize to unseen particle arrangements. PhyFlow consists of a novel physics-guided DL architecture that embodies the solution structure of physics-based CFD solvers as an implicit form of physics-based supervision. **(ii)** We develop a novel Physics-Guided Inverse Distance Function (IDF) that enables PhyFlow to be trained in an end-to-end fashion, going from input particle arrangements to output flow fields in a single-step framework. **(iii)** We perform rigorous comparison of PhyFlow with several state-of-the-art (SOTA) image (and flow field) generation models across various 3D flow settings. We also demonstrate SOTA results on the downstream task of predicting drag forces on particles in unseen arrangements, using our generated flow fields. **(iv)** Finally, we show that PhyFlow predictions are more physically consistent with the governing physics equations than all other baseline methods.

The rest of the paper is organized as follows: Section II discusses previous research in machine learning applications for physics problems (in particular, CFD). Section III details the

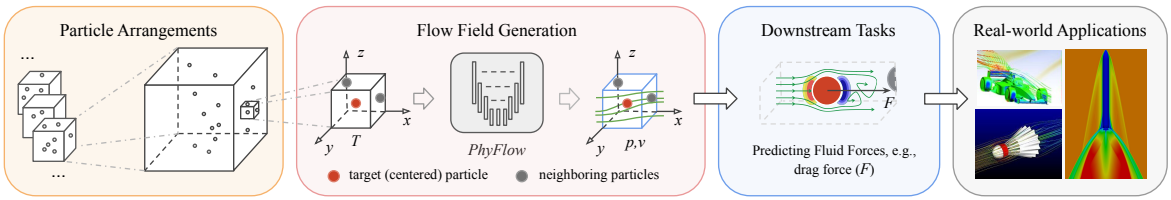

Fig. 1: Proposed Physics-Guided Deep Learning Architecture & Applications

CFD preliminaries specific to our application and section IV details the proposed PhyFlow model. Section V and section VI detail the experimental setup and describe results respectively, while section VII provides concluding remarks.

II. Related Work

There is a growing body of work on developing deep learning formulations for flow field generation [6]–[11], for different application contexts such as 2D-turbulence modeling [10] and 1D-solution of the governing physics equations of fluid flow [11]. However, a majority of these works only investigate 2D single-phase flows, using numerical simplifications such as the use of stream functions to reduce the number of variables [11], which do not apply in 3D.

A line of work on using deep learning methods in CFD applications involves upsampling high-resolution flow fields using low-resolution fields provided as inputs [8], [9]. However, since they require flow field CFD simulations (although at coarse-scales) as inputs, they are not completely free from the expensive costs of running CFD solvers during test time.

The closest line of work related to our problem is a recent approach developed by Siddani et al. [12] for 3D multi-phase flow generation. In this work, a Wasserstein GAN with gradient penalty (WGAN-GP) was used to first predict initial estimates of flow fields in a large domain around a candidate particle, followed by a second step to predict the flow fields in a focused domain around the particle of interest using a CNN network. Our proposed PhyFlow framework is different from this work on several fronts. First, we employ a physics-based architecture for predicting pressure and velocity fields in a sequential manner that resembles the solution structure of popular *projection method* based CFD simulations. Second, we use a novel physics-guided inverse distance function (IDF) representation that is able to capture local to global information around the vicinity of every particle. Third, we not only show the performance of our framework on flow field generation but also on the downstream task of drag force prediction, in contrast to previous research.

III. Background On Flow Generation

In the field of computational fluid dynamics (CFD), a common approach for generating the steady-state pressure field (p) and velocity fields along the three spatial axes (u, v, w) of incompressible fluids as a function of space (x, y, z) is by solving two basic conservation laws—mass (a.k.a divergence) and momentum conservation, described using the following equations:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \quad (1)$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} + \frac{1}{\rho}\frac{\partial p}{\partial x} - \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) = 0, \quad (2)$$

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + \frac{1}{\rho}\frac{\partial p}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) = 0, \quad (3)$$

$$u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} + \frac{1}{\rho}\frac{\partial p}{\partial z} - \frac{1}{Re}\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) = 0 \quad (4)$$

where Re is the Reynolds number and ρ is the density of the incompressible fluid. Eq. 1 corresponds to mass conservation (requiring divergence of velocity fields to be 0) while the other three equations (Eq. 2-4) correspond to momentum conservation in the x,y,z directions, respectively. Together, these four equations are referred to as the Navier-Stokes (N-S) equations [13], which form the bedrock of all CFD calculations for flow generation. While they are easy to express, finding analytical solutions to these coupled non-linear second-order partial differential equations (PDEs) is extremely challenging in most practical use-cases. As a result, a variety of numerical techniques have been developed to solve these PDEs and generate flow fields in varying input conditions, e.g., finite difference based techniques [14].

a) Projection Method: The projection method is a widely-used solution method for incompressible fluid flows [14]. It solves the momentum and mass conservation equations sequentially in three steps (i) **Intermediate Velocity Field Estimation:** first, velocity field estimates are computed from the N-S momentum equation (Eq. 2- 4) using a known pressure-field at a previous iterate or time-step. (ii) **Pressure Field Estimation:** next, the velocity field estimates are used to calculate the updated pressure field satisfying the mass conservation Eq. 1. This estimation is carried out by solving an elliptic pressure Poisson equation given the intermediate velocity field. (iii) **Velocity Field Estimate Correction:** finally, the velocity field estimates are updated based on the gradients of the updated pressure field. These three steps are iterated until convergence. We take inspiration from this popular method to develop the structure of our physics-guided PhyFlow architecture as described in Section IV-A.

b) Downstream Tasks in Multiphase Flow: Once the pressure and velocity fields have been solved, they can be used as inputs in a variety of downstream tasks in CFD applications. For example, in multiphase flow problems (which is the target use-case of this work), the flow fields can be used to estimate the interaction forces acting on particles suspended in the fluid such as the **drag force**. In what are referred to as "Particle

Resolved Simulations" or PRS, the particles are explicitly resolved in the numerical calculation on a fine 3D grid, and the velocity and pressure fields around each particle are solved before the fluid drag force on each particle can be computed. We consider PRS as the ground-truth for flow field generation and drag force prediction in this work.

IV. Proposed PhyFlow Framework

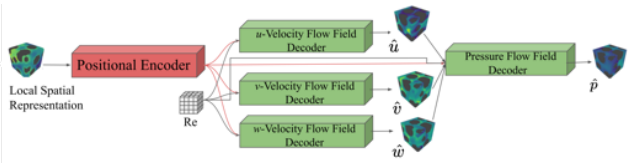

Fig. 2: Architecture of our proposed PhyFlow framework embodies the solution structure of the *projection method* used in physics-based CFD solvers.

The main goal of our proposed PhyFlow framework is to build an ML model that accepts information about the spatial arrangement of particles around a target particle as input, and generate pressure and velocity fields in the local neighborhood around a particle as outputs, which can be useful for downstream tasks such as drag force prediction in multi-phase flow contexts. By training our PhyFlow model on a limited set of PRS ground-truth simulations, we aim to achieve better generalizability on unseen particle arrangements than state-of-the-art baselines by anchoring our framework with knowledge of the physics background of flow generation in a variety of ways. In the following, we describe the key physics-guided innovations of our proposed framework.

A. Physics-Guided Neural Network Architecture

Figure 2 provides an overview of our PhyFlow framework that takes in the local spatial representation of neighboring particles centered around a candidate particle i of interest as input 3D tensors ($I_i \in \mathbb{R}^{l\times l\times l}$), and generates predictions of pressure ($\hat{p}_i \in \mathbb{R}^{l\times l\times l}$) and velocity ($\hat{u}_i \in \mathbb{R}^{l\times l\times l}, \hat{v}_i \in \mathbb{R}^{l\times l\times l}, \hat{w}_i \in \mathbb{R}^{l\times l\times l}$) fields around particle i as 3D tensor outputs. The proposed architecture has two parts. The first part, termed the *Positional Encoder*, learns a low-dimensional encoding ($e_i \in \mathbb{R}^{k\times k\times k}, k << l$) of the positions, orientations and density of particles in the local neighborhood of the candidate particle being modeled. While these positional encodings capture rich information about the arrangements of neighboring particles, flow fields also depend on the Reynolds number (Re) of the flow regime, which is a scalar dimensionless quantity represented by the ratio of inertial and viscous forces. To capture this information in our PhyFlow framework, we concatenate the positional encodings e_i of particle i with a constant 3D tensor of size $\mathbb{R}^{k\times k\times k}$ with values equal to Re at every tensor cell. The concatenated encodings are then fed as inputs into the *Flow-field Decoders* for generating pressure and velocity fields. Instead of using a black-box architecture where each component of the generated pressure and velocity fields (i.e., $\hat{p}$, $\hat{u}$, $\hat{v}$, $\hat{w}$) is generated as a separate output channel, we adopt a physics-guided architecture for generating pressure and velocity fields that leverage the solution structure of the **projection method** used in physics-based CFD solvers as described in Section III. In particular, we first resolve the velocity fields (akin to step (i) of the *projection method* i.e., Intermediate Velocity Field Estimation) around particle i ($\hat{u}_i, \hat{v}_i, \hat{w}_i$) using three separate decoders. Generated velocity fields are then concatenated with positional encodings e_i and Re and fed as inputs to another decoder that generates pressure field, $\hat{p}_i$ (akin to step (ii) of the *projection method* wherein pressure fields are estimated contingent upon intermediate velocity field estimates)[†]. Finally, the velocity fields are corrected based on errors in the predicted pressure field (similar to the *velocity field estimate correction* i.e., step (iii) of the *projection method*). In this way, the learning of pressure and velocity fields are tightly coupled such that during forward pass of our model, the velocity field predictions are used to condition the predictions of pressure field. Further, by jointly learning the decoders in an end-to-end fashion, prediction errors in the pressure field are back-propagated to correct the velocity field predictions during model training. The architecture of PhyFlow thus resembles the nature of computations occurring in commonly used CFD solvers. This is similar to recent works on embedding prior knowledge from physics in design of neural networks [6], [10], [15].

B. Physics-Guided Input Representations

Another key innovation of our approach is the way we construct the input tensor representation I_i to capture the local arrangement of particles around particle i. A simple approach for constructing I_i is to use binary masks where a voxel in I_i is 1 only if it does not intersect with a neighboring particle in a fixed-width neighborhood around particle i, as considered in [12]. However, such binary masks can only capture information about particle arrangements in the immediate local neighborhood of particle i (with fixed widths), while it is known that flow fields are considerably influenced by particles not just in the local vicinity but also outside. This is especially true for flow field values at the boundaries of the fixed-width neighborhoods. It is also known that the effect of a neighboring particle on the flow fields at a voxel decreases with the distance of the particle. We build upon these two observations well-established in CFD literature to develop a novel *Physics-Guided Inverse Distance Function* (IDF) for constructing I_i. The IDF tensor serves as the input to our PhyFlow architecture and is calculated for every voxel V_j in the tensor as follows:

$$\text{IDF}(V_j) = \sum_{i=1}^{K} \frac{1}{d(V_j, P_i)}, \tag{5}$$

where P_i represents the i^{th} nearest-neighboring particle to V_j, $d(\cdot)$ represents the Euclidean distance between P_i and V_j,

[†]An important factor in generating the pressure field is to ensure that velocity field satisfies mass conservation (Eq. 1) and we verify (see Appendix: tinyurl.com/mjkcrsdw) that PhyFlow predictions adhere to Eq. 1 well.

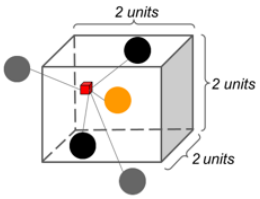

Fig. 3: The IDF tensor captures local arrangement of particles at every voxel (red) inside a fixed neighborhood (of width 2 units) centered around candidate particle i (orange). IDF value at a voxel is determined by its K-nearest particles, some of which may lie inside the fixed-width region (black), while others may lie outside (gray).

and K represents the total number of neighbors considered. A higher IDF value indicates a greater density of particles around the voxel that can influence its flow field values. As illustrated in Figure 3, the IDF tensor around a candidate particle i (orange) not only captures the effects of particles inside the fixed-width neighborhood around i (black), but also those particles that are outside yet among the top-K nearest neighbors of voxels in this neighborhood (gray). This allows us to capture local and global information of particle arrangements depending on the value of K. In our experiments, we chose $K = 15$ based on previous studies on the effect of nearest neighbor counts in multi-phase flow contexts [16].

C. Physics-Guided Model Interpretability

We propose a novel pipeline for interpreting the predictions of flow fields generated by our PhyFlow framework by quantifying the consistency of our model's predictions with respect to the underlying physics of fluid dynamics, namely, the Navier-Stokes equations.

In particular, we systematically replace any one of the four flow field values in the mass or momentum conservation equations using PhyFlow predictions, while keeping other fields fixed to ground-truth values. We then evaluate the *ablated* momentum and divergence residuals and characterize their similarity to the optimal (i.e., perfect) momentum and divergence residuals for a particular (Re,ϕ) experimental context. We define an *ablated* residual as the value obtained when terms corresponding to only one field (i.e., one of p, u, v, w) in a mass or momentum conservation equation is calculated using the corresponding predicted flow field while other terms are calculated using ground-truth flow fields.

Verifying Consistency of Predicted Pressure Field: The consistency of the pressure field is verified with respect to the momentum conservation equations. There are a set of three momentum conservation equations as defined in section III, one each for the x,y,z direction of the 3D flow. The consistency of the pressure is verified with respect to the momentum-x equation (Eq. 2) by replacing the $\frac{\partial p}{\partial x}$ term by $\frac{\partial \hat{p}}{\partial x}$ (which is the gradient of predicted flow field $\hat{p}$ w.r.t x) while all other terms are calculated using ground truth flow fields. The value of the residual obtained if $\hat{p}$ is consistent with the N-S equations should be close to 0. Similarly, the consistency of $\hat{p}$ w.r.t momentum-y (Eq. 3) and momentum-z (Eq. 4) is verified by appropriately replacing the $\frac{\partial p}{\partial y}$ in Eq. 3 by $\frac{\partial \hat{p}}{\partial y}$ and $\frac{\partial p}{\partial z}$ in Eq. 4 by $\frac{\partial \hat{p}}{\partial z}$ and in each case analyzing the *ablated* residual.

Verify Consistency of Predicted Velocity Fields: Velocity field predictions $\hat{u}$,$\hat{v}$,$\hat{w}$ (velocity predictions in x,y,z directions respectively) are used to calculate consistency w.r.t. mass conservation N-S equation (Eq. 1), by replacing appropriate terms in a similar vein to the consistency verification of the pressure field. Such *ablated* residuals (as opposed to residuals calculated using all predicted fields simultaneously) allow us to explicitly verify that each of the four predicted flow fields $\hat{p}, \hat{u}, \hat{v}, \hat{w}$ are consistent with the governing N-S equations and that they do not converge to trivial solutions. Details of calculation of ablated residuals using specialized finite-difference convolutional kernels are in the appendix[‡].

V. Experimental Setup

Dataset Description: In this paper, we employ data from a PRS simulation for a single random arrangement $\mathbf{P} = \{P_1, .., P_m\}$ of m particles in a multi-phase 3D flow setting. Each particle in our simulation domain has diameter D units with our total simulation space S spanning a cubic region of size $10D \times 10D \times 10D$. The result of the PRS simulation on S yields pressure field p and three velocity fields u, v, w (i.e., velocity in the x, y, z direction respectively) for every fluid voxel (i.e., a grid cell in S not occupied by a particle; all pressure and velocity fields inside particle cells are 0). For each $P_i \in \mathbf{P}$, we extract its *local neighborhood* to be a cube N_i of size $2D \times 2D \times 2D$ centered around P_i. N_i always fully contains P_i at its center while any other proximal particles may be fully or partially contained in N_i. Thus our dataset $\mathcal{D} = \{F_1, .., F_m\}$ consists of m local neighborhood flow fields for all particles in $\mathbf{P}$ where each $F_i = \{p_{N_i}, u_{N_i}, v_{N_i}, w_{N_i}\}$. For ease of notation, we drop the subscript and refer to these local neighborhood flow-fields as p, u, v, w for a particle. Our proposed PhyFlow model takes as input, a representation of a particle's local neighborhood (section IV-B) and returns $\hat{p}, \hat{u}, \hat{v}, \hat{w}$. Each PRS simulation is run for a particular Reynolds number (Re) and solid fraction (ϕ) case and we run 16 experiments i.e., one for each (Re,ϕ) combinations of Re=$\{$10, 50, 100, 200$\}$ and solid fractions ϕ=$\{$0.1, 0.2, 0.3, 0.35$\}$ with a total of 7260 particles and corresponding local neighborhood flow fields (p, u, v, w) obtained across all (Re,ϕ) settings. The flow field generation results are evaluated on PRS simulation run with a completely new particle arrangement (not used in training) for the same set of 16 experimental settings again with a total of 7260 particles and corresponding local neighborhood flow fields (p, u, v, w).

Baselines: We compare PhyFlow with SOTA image-to-image translation pipelines. Specifically, we compare with *pix2pix* [17] and *CycleGAN* [18] based variants for 3D image-to-image translation, denoted as *Vox2Vox* and *CycleGAN*. We also include ResNet and UNet as competitive baselines that are not based on Generative Adversarial Network (GAN) formulations. As mentioned in section II, [12] is the most

[‡]tinyurl.com/mjkcrsdw

closely related work to ours, we have re-implemented the model (as their code is not publicly available) in [12] to the best of our ability.

Evaluation Metrics: The root-mean-squared-error (RMSE) evaluation metric is used to evaluate the quality of the predicted flow fields. We also evaluate the flow fields generated by PhyFlow on a down-stream task of particle drag force prediction on an unseen arrangement of particles. For this task, as is standard practice [16], [19], we employ three evaluation metrics namely (i) Mean Squared Error (MSE), (ii) Mean Relative Error (MRE), and (iii) Area Under the Relative Error Curve (AUREC). Further details about baselines and evaluation metrics are in the appendix.

VI. Results & Discussion

We now detail the performance of PhyFlow by characterizing the quality of the generated flow fields. We do this by reporting the overall RMSE of the predicted flow fields with respect to the corresponding ground truth flow fields produced using the PRS simulation. Further, to demonstrate the effectiveness of PhyFlow , we test it on the down-stream task of predicting the drag-force on every candidate particle using the generated flow fields. In all our experiments, we specifically test for PhyFlow 's ability to extrapolate to unseen particle arrangements. Finally, to enable interpretability of PhyFlow predictions, we demonstrate consistency of predicted flow fields with the optimal physical behavior according to the Navier-Stokes equations.

A. Model Performance Comparison

Table I showcases the comparison of PhyFlow with other SOTA image generation and image-to-image translation architectures as well as related flow field generation work conducted by [12]. We report the average RMSE across the four generated fields $\hat{u}, \hat{v}, \hat{w}, \hat{p}$ for all particles in each of sixteen different experimental settings. In the table, we notice that PhyFlow outperforms all other models in 12 out of 16 settings. PhyFlow achieves an average performance improvement of **49.61%** over other models across all (Re,ϕ) settings. The proposed full PhyFlow model outperforms all other models in a large majority (12 out of 16) of the experimental settings. PhyFlow yields an average performance improvement of **10.01%** over U-Net (next best model) prediction performance across all (Re,ϕ) cases. Also noticeable from the table is that all models perform better for higher ϕ cases (i.e., 0.3,0.35) than for cases of lower ϕ. This may be partially attributed to availability of greater volume of training examples in higher ϕ cases. To further corroborate our claim of superior flow field generation quality, we have included qualitative results and also evaluated each component of our PhyFlow model through rigorous ablation analyses (see appendix[§]).

B. Consistency with Physical Governing Equations

Multi-phase 3D fluid flow is governed by Navier-Stokes mass and momentum conservation equations. Since our effort

[§]tinyurl.com/mjkcrsdw

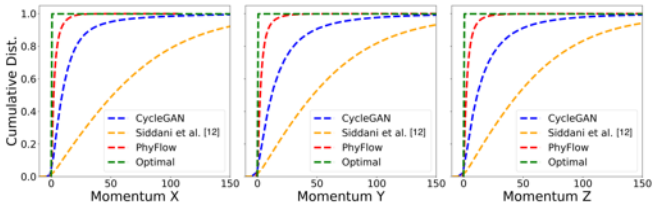

Fig. 4: PhyFlow (red) momentum characterization using *ablated* residuals shows good agreement (compared to other models) with the optimal physical residual values (green) in (Re = 10,ϕ = 0.1) experimental setting.

with development of PhyFlow has been to extensively leverage knowledge of the underlying physics toward design and development of the machine learning pipeline for flow field generation, it is only natural that we also analyze the degree of consistency of the predictions with the same governing equations. We now characterize the consistency of the predicted flow fields $\hat{p}, \hat{u}, \hat{v}, \hat{w}$ using *ablated* residuals (section IV-C). Figure 4 characterizes the degree of physical consistency of the predicted pressure $\hat{p}$ with each of the three momentum equations along the x,y,z axes, respectively. In each of the three plots (showcasing ablated residuals for $\frac{\delta \hat{p}}{\delta x}, \frac{\delta \hat{p}}{\delta y}, \frac{\delta \hat{p}}{\delta z}$ respectively), we notice that the predicted pressure by PhyFlow (red dotted line) results in residuals close to the optimal (green dotted line) residuals for the (Re=10,ϕ=0.1) experiment. We also compare the state-of-the-art CycleGAN model as well as the flow field generation model proposed by Siddani et al. [12] and find that both these models learn significantly less physically consistent representations for the pressure field than PhyFlow . We, note that even in CFD simulations, resolving the pressure field is a more expensive (complex) task. We can show that PhyFlow is able to achieve significantly more physically consistent pressure field predictions than other baselines. Similar results (see appendix) are observed for other (Re,ϕ) experiments.

C. Drag Force Prediction Results

A good indication of effective representation learning frameworks is the performance of the learned representation on downstream tasks. It is well known that the drag force of a particle in multi-phase flows may be calculated as a function of the pressure and velocity fields in the vicinity of the particle. We hence develop a pipeline (consisting of a very simple, shallow CNN consisting of 2 convolutional and 2 fully-connected layers) to ingest the flow-field predictions of PhyFlow and predict the drag forces of target particles in the context of unseen particle arrangements. We notice in table II that PhyFlow + CNN model yields the highest AUREC values compared with other state-of-the-art models specifically designed for the drag force prediction task. The only alternative to achieve a higher AUREC on drag force prediction is to run a full PRS simulation (depicted in the table as *PRS(Ground Truth)*) on the new particle arrangement and feed these ground truth flow field representations to the CNN model. However, running PRS simulation for every new particle arrangement is extremely expensive and hence PhyFlow provides a cheap,

TABLE I: Comparison of mean RMSE of generated voxels across 16 experimental settings in the context of predicting flow fields on unseen particle arrangements. We notice PhyFlow significantly outperforms (**49.61**% performance improvement) other models in 12 out of 16 experimental settings.

Re	10				50				100				200			
ϕ	0.10	0.20	0.30	0.35	0.10	0.20	0.30	0.35	0.10	0.20	0.30	0.35	0.10	0.20	0.30	0.35
CycleGAN	0.606	0.435	0.345	0.334	0.899	0.732	0.476	0.452	1.164	0.817	0.566	0.537	1.491	0.907	0.675	0.621
ResNet	0.624	0.447	0.334	0.317	0.943	0.771	0.475	0.446	1.221	0.878	0.561	0.521	1.532	0.967	0.661	0.593
Vox2vox	0.612	0.439	0.344	0.335	0.918	0.756	0.497	0.481	1.184	0.854	0.579	0.557	1.511	0.943	0.680	0.633
U-Net	0.526	**0.321**	0.197	0.191	0.742	**0.579**	0.250	0.241	1.016	**0.635**	0.299	0.275	1.330	**0.688**	0.387	0.340
WGAN-GP	0.821	0.684	0.659	0.637	0.966	0.774	0.654	0.640	1.135	0.850	0.684	0.662	1.369	0.930	0.746	0.702
Siddani [12]	0.690	0.588	0.575	0.573	0.858	0.724	0.581	0.591	1.042	0.808	0.619	0.607	1.289	0.889	0.690	0.645
Phyflow	**0.447**	**0.320**	**0.158**	**0.162**	**0.638**	0.622	**0.191**	**0.199**	**0.922**	0.692	**0.239**	**0.227**	**1.242**	0.752	**0.333**	**0.295**

TABLE II: Results on the downstream task of particle drag force prediction. PhyFlow +CNN achieves SOTA results. Only method that performs better is one that uses ground-truth PRS simulations of flow fields on the test cases.

Model	MSE	MRE	AUREC
DNN (He et al. [16])	38.53	23.12	0.78241
PhyNet (Muralidhar et al. [19])	56.42	26.34	0.75079
Mean	31.42	19.45	0.81828
PhyFlow + CNN	**9.19**	**11.31**	**0.89922**
PRS(Ground-Truth) + CNN	1.42	4.37	0.96794

scalable alternative for drag force prediction on new particle arrangements. Specifically, the average wall clock time of PRS simulation for one (Re,ϕ) setting is 72 hours running on 128 CPU cores whereas our PhyFlow + CNN model requires about 20 hours to train on all 16 (Re,ϕ) settings using 2 Nvidia P100 Tesla GPUs. Inference on new particle configurations using PhyFlow + CNN is inexpensive and takes negligible computation time i.e., less than 1 second per particle drag force prediction / flow field generation. Also note, in Table II, the current real-world methodology for drag force calculation is to assign each particle with the *mean* drag force calculated using the corresponding (Re,ϕ) of the experimental setting [16]. PhyFlow + CNN setup yields a **9.89**% improvement in AUREC over the *Mean* baseline model.

VII. Conclusions and Future Work

We introduced PhyFlow , a one-step end-to-end physics-guided deep learning model for fluid flow field generation of unseen particle arrangements in 3D multi-phase flows. Through several experiments, we demonstrated the superior representation learning capability of PhyFlow , which outperforms all SOTA image generation and image-to-image translation pipelines with average performance improvement of **49.61**%. We also demonstrated the effectiveness of PhyFlow generated flow fields on a down-stream task of particle drag force prediction and achieved SOTA results beating the next best drag force prediction model by **9.89**%. In the future, we shall extend PhyFlow to turbulent flows and to other extrapolation contexts (i.e., unseen Re,ϕ).

References

[1] J. D. Anderson and J. Wendt, *Computational fluid dynamics*. Springer, 1995, vol. 206.

[2] J. Baptiste Filippi *et al.*, "Coupled atmosphere-wildland fire modelling," *Journal of Advances in Modeling Earth Systems*, vol. 1, no. 4, 2009.

[3] B. Melka *et al.*, "Multiphase simulation of blood flow within main thoracic arteries of 8-year-old child with coarctation of the aorta," *Heat and Mass Transfer*, vol. 54, no. 8, pp. 2405–2413, 2018.

[4] M. Raissi and G. E. Karniadakis, "Hidden physics models: Machine learning of nonlinear partial differential equations," *Journal of Computational Physics*, vol. 357, 2018.

[5] R. Ramakrishnan *et al.*, "Big data meets quantum chemistry approximations: the δ-machine learning approach," *Journal of chemical theory and computation*, vol. 11, no. 5, pp. 2087–2096, 2015.

[6] D. Kochkov *et al.*, "Machine learning accelerated computational fluid dynamics," *arXiv preprint arXiv:2102.01010*, 2021.

[7] Y. Zhu *et al.*, "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data," *Journal of Computational Physics*, vol. 394, p. 56–81, Oct 2019. [Online]. Available: http://dx.doi.org/10.1016/j.jcp.2019.05.024

[8] A. Subramaniam *et al.*, "Turbulence enrichment using physics-informed generative adversarial networks," 2020.

[9] H. Gao, L. Sun, and J.-X. Wang, "Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels," 2020.

[10] R. Wang *et al.*, "Towards physics-informed deep learning for turbulent flow prediction," in *ACM SIGKDD*, 2020, pp. 1457–1466.

[11] M. Raissi *et al.*, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.

[12] B. Siddani *et al.*, "Machine learning for physics-informed generation of dispersed multiphase flow using generative adversarial networks," *arXiv preprint arXiv:2005.05363*, 2020.

[13] A. J. Chorin, "Numerical solution of the navier-stokes equations," *Mathematics of computation*, vol. 22, no. 104, pp. 745–762, 1968.

[14] J. H. Ferziger, M. Perić, and R. L. Street, *Computational methods for fluid dynamics*. Springer, 2002, vol. 3.

[15] L. Li *et al.*, "Kohn-sham equations as regularizer: Building prior knowledge into machine-learned physics," *Physical Review Letters*, vol. 126, no. 3, p. 036401, 2021.

[16] L. He and D. K. Tafti, "A supervised machine learning approach for predicting variable drag forces on spherical particles in suspension," *Powder technology*, vol. 345, pp. 379–389, 2019.

[17] P. Isola *et al.*, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.

[18] J.-Y. Zhu *et al.*, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV 2017*, 2017.

[19] N. Muralidhar *et al.*, "Phynet: Physics guided neural networks for particle drag force prediction in assembly," in *SDM 2020*. SIAM, 2020, pp. 559–567.

[20] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, 2019.

[21] A. Yew, "Brown university mathematics apma 0160, lecture notes: Numerical differentiation: finite differences," 2011, uRL: https://www.dam.brown.edu/people/alcyew/handouts/numdiff.pdf. Last visited on 2021/02/08.

[22] B. Xu *et al.*, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.