Proceedings of the 1st international Symposium on AI and Fluid Mechanics
Paper No S18 P1

Drag modelling for flows through assemblies of spherical particles with machine learning: A comparison of approaches

Julia Reuter[b], Hani Elmestikawy[a], Sanaz Mostaghim[b,c] and Berend van Wachem[a,*]

[a]Chair of Mechanical Process Engineering, Otto-von-Guericke-Universität, Magdeburg, 39106, Germany
[b]Chair of Computational Intelligence, Otto-von-Guericke-Universität, Magdeburg, 39106, Germany
[c]Fraunhofer Institute for Transportation and Infrastructure Systems IVI, Dresden, 01069, Germany

ARTICLE INFO

Keywords:
Drag modelling
Particle-laden flows
Graph neural networks
Symbolic regression
Machine learning

ABSTRACT

Drag forces on particles in random assemblies can be accurately estimated through particle-resolved direct numerical simulations (PR-DNS). Despite its limited applicability to relatively small assemblies, data obtained from PR-DNS has been the driving force for the development of drag closures for much more affordable simulation frameworks, such as Eulerian-Lagrangian point particle methods. Recently, more effort has been invested in the development of deterministic drag models that account for the effect of the structure of the particle assembly. Current successful deterministic models are mainly black-box neural networks which: 1) Assume pairwise superposition of the neighbours' effect on the drag, and 2) Are trained on PR-DNS data for a wide range of particle concentrations and flow regimes. To alleviate the black-box nature of neural networks, we use genetic programming (GP) to develop interpretable models. In our previous research, this has been proven successful in the Stokes regime. In the current contribution, we extend the application of GP to higher particle Reynolds number regimes. This is done by training a graph neural network (GNN) on the PR-DNS data to learn the pairwise interactions among the particles that constitute the drag variation. The significance of the input features of the GNN is assessed via a feature permutation approach. Then, the estimated pairwise interactions as extracted from the GNN are fed to a GP algorithm, which searches for symbolic expressions that fit the input data. A comparison between the trained GNN model and the resulting symbolic expressions is presented, to assess whether the symbolic expression can capture the underlying patterns learnt by the GNN. The comparison demonstrates the potential of GP in finding relatively simple symbolic models. At the same time, the accuracy of the symbolic models slightly fall behind the GNN.

1. Introduction

Understanding and simulating gas-solid flow systems relies heavily on the prediction of drag experienced by particles in particle-laden flows [5]. Although particle-resolved direct numerical simulations (PR-DNS) can accurately describe particle-fluid interactions and provide high-fidelity flow data, their excessive computational cost limits their use to small-scale systems. However, the data from PR-DNS has driven the development of drag closure models for point particle simulations, which enables more affordable simulations of relatively larger-scale systems.

Recent efforts have focused on deterministic drag models that account for particle arrangement effects, which assume pairwise superposition of neighbour effects [2]. State-of-the-art deterministic drag models are mostly data-driven and take the form of a black-box neural network providing accurate

*Corresponding author: Berend van Wachem, Email: berend.van.wachem@multiflow.org
ORCID(s): 0000-0002-5399-40750 (B. van Wachem)

drag estimation; however, they lack interpretability [59, 60, 61]. Previous work has demonstrated the potential of symbolic regression to develop interpretable drag models in the Stokes Regime. By adopting graph neural networks (GNNs) that apply the pairwise interaction by construction, symbolic regression was able to transfer these pairwise particle interactions into symbolic expressions [50, 24]. This approach not only provides transparent models, but also helps to understand the underlying physics of particle-laden flows. Building on these findings, the current work seeks to extend this framework to more complex flow regimes, mainly higher Reynolds numbers, where the interplay of inertial effects are more prominent.

This paper is organized as follows: Section 2 provides a background on the particle-laden flow problem at hand, the development of deterministic drag models, and a comprehensive overview of GNN and symbolic regression. Section 3 details the implemented machine learning framework, including the GNN architecture and the GP algorithm. Section 4 presents the results of different GNN and GP experiments, comparing the performance of the GNN and the derived symbolic models in various flow regimes. Finally, the conclusions are presented in Section 5, with a summary of the key findings.

2. Background

2.1. Simulation of particle-laden flows

The PR-DNS data are generated by solving the governing equations of the flow past a periodic arrangement of stationary monodispersed particles, randomly positioned in a cubic domain. The incompressible Newtonian fluid flow is governed by the continuity and the Navier-Stokes equations as follows:

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho \boldsymbol{\nabla} \cdot (\mathbf{u} \otimes \mathbf{u}) = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{S}_{\text{body}} + \mathbf{S}_{\text{IBM}}, \tag{2}$$

where ρ and μ and the fluid density and the fluid dynamic viscosity, respectively. $\mathbf{u}$ is the fluid velocity vector and p represents the pressure. The flow is driven by a body force, $\mathbf{S}_{\text{body}}$, in the direction of the main flow. To enforce the boundary conditions of no-slip and no-penetration at the surfaces of the particles, a momentum source, $\mathbf{S}_{\text{IBM}}$, is introduced. This source term is computed using the hybrid immersed boundary method [16].

The two key global parameters affecting the drag are the particle Reynolds number, Re_p, and the global particle volume fraction, ε_p. Re_p is defined based on the fluid properties, particle diameter d_p, and the magnitude of the fluid superficial velocity, $\|\langle \mathbf{u} \rangle\|$, as follows:

$$\text{Re}_\text{p} = \frac{\rho \|\langle \mathbf{u} \rangle\| d_\text{p}}{\mu}. \tag{3}$$

The governing equations are discretized and solved numerically using a finite volume framework [23]. The drag force acting on each particle in the arrangement can then be estimated from the discretized solution. Fig. 1 shows a sample of the velocity field from one of the considered cases. A dataset is built by varying Re_p, ε_p, and the random particle arrangement.

The considered datasets, originally generated in [68] and are openly available [67], cover a range of $\text{Re}_\text{p} \in \{0.1, 1, 10, 50, 100, 300\}$ and $\varepsilon_\text{p} \in \{0.1, 0.2 \dots 0.6\}$. In this work, the datasets are augmented with two additional values of Re_p: 5 and 200. Table 1 summarizes the mean drag, $\langle F_d \rangle$, and the standard deviation of the drag, σ_d, present in the datasets across different (Re_p, ε_p) pairs. The reported mean drag is normalized by the Stokes drag, $F_{\text{St}} = 3\pi\mu d_\text{p} \|\langle \mathbf{u} \rangle\|$. There exists an extensive body of research that has successfully developed accurate models for the mean drag as a function of Re_p and

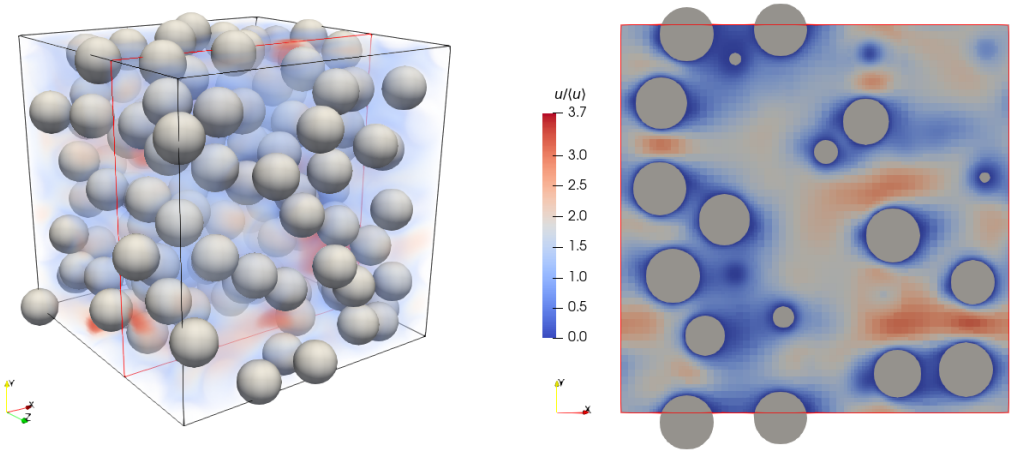


Figure 1: Sample flow field visualizations: 3D (left) and 2D (right), at $\mathrm{Re_p} = 5$ and $\varepsilon_\mathrm{p} = 0.2$. The flow is driven in the positive x direction.

Table 1
Statistics of the considered PR-DNS datasets

	$\varepsilon_\mathrm{p} = 0.1$		$\varepsilon_\mathrm{p} = 0.2$		$\varepsilon_\mathrm{p} = 0.3$	
$\mathrm{Re_p}$	$\langle F_d \rangle / F_\mathrm{St}$	$\sigma_d / \langle F_d \rangle$	$\langle F_d \rangle / F_\mathrm{St}$	$\sigma_d / \langle F_d \rangle$	$\langle F_d \rangle / F_\mathrm{St}$	$\sigma_d / \langle F_d \rangle$
0.094	2.772	22.78	5.668	18.12	10.757	20.72
0.949	2.804	23.00	5.708	18.25	10.828	20.79
4.863	3.152	25.09	6.154	19.41	11.504	21.46
9.605	3.603	26.42	6.809	20.84	12.553	22.46
47.628	6.250	26.89	11.144	25.06	19.860	26.16
95.304	8.831	26.58	15.520	26.80	27.583	27.95
188.415	13.914	26.70	23.773	28.64	41.912	29.85
281.933	18.423	26.73	31.384	28.70	55.436	29.59

	$\varepsilon_\mathrm{p} = 0.4$		$\varepsilon_\mathrm{p} = 0.5$		$\varepsilon_\mathrm{p} = 0.6$	
$\mathrm{Re_p}$	$\langle F_d \rangle / F_\mathrm{St}$	$\sigma_d / \langle F_d \rangle$	$\langle F_d \rangle / F_\mathrm{St}$	$\sigma_d / \langle F_d \rangle$	$\langle F_d \rangle / F_\mathrm{St}$	$\sigma_d / \langle F_d \rangle$
0.094	21.593	18.38	42.763	15.71	114.182	19.59
0.949	21.744	18.58	43.062	15.79	114.539	19.60
4.863	22.901	18.69	45.361	15.91	123.500	19.50
9.605	24.802	19.37	48.623	16.25	129.842	19.62
47.628	38.310	22.24	74.631	18.56	211.314	21.16
95.304	53.684	24.95	103.123	20.17	311.419	21.63
188.415	83.061	27.01	164.673	22.14	501.833	23.70
281.933	109.347	27.45	215.591	22.50	672.010	22.45

ε_p, e.g., [53, 10, 64, 62]. However, these models do not account for the variation of the drag experience by each particle around that mean value [68]. This variation, as shown in Table 1, can be as high as $\sim 30\%$ of the mean drag force [1]. The primary focus of this work is on the variation of drag forces within the same assembly of particles, i.e., at a given $(\mathrm{Re_p}, \varepsilon_\mathrm{p})$ pair, assuming the availability of a perfect mean drag model.

2.2. Physics-informed machine learning

In the recent literature, an increased interest in the integration of domain-specific knowledge and constraints into machine learning algorithms can be observed. Purely data-driven machine learning (ML) approaches tend to exhibit certain limitations, such as weak generalization on limited or noisy datasets, lack of interpretability, and difficulties with scalability. Physics-informed machine learning (PIML) aims at developing models that reflect expected behaviours and adhere to fundamental physical laws [33]. Raissi et al. [47] introduced physics-informed neural networks (PINNs), which incorporate physics-based soft constraints on the loss function of the learning algorithm, usually in the form of partial differential equations ([48]). PINNs demonstrated to be effective for a wide range of problems (e.g., [19]), but exhibited ill-conditioned behaviour for increasing problem complexities, as pointed out in [35]. Watson et al. [71] provided a comprehensive review of contemporary approaches for incorporating physical principles into neural networks, extending beyond the framework of PINNs.

PIML has been applied to the specific problem of predicting the variation around the mean drag in particle-laden flows in the past. One of the initial attempts to model variations in drag using machine learning was the work of He and Tafti [30]. In their work, a neural network is trained with the information of all neighbouring particles to predict the force variation. Subsequent work by Balachandar et al. [6] revealed limitations in this approach, as the neural network suffered from overfitting and required extensive training data to achieve reasonable accuracy. Addressing these challenges, Seyed-Ahmadi and Wachs [59] proposed a physics-inspired architecture that replaced the neural network that takes all neighbour information with a repeated neural network designed to process one neighbouring particle at a time, effectively applying pairwise interactions in the network architecture. This approach demonstrated a significant improvement in accuracy, particularly in terms of generalization to unseen data. Building on this progress, Siddani and Balachandar [60] extended the framework by incorporating trinary and higher-order interactions, achieving a slight improvement in predictive performance.

2.3. Graph neural networks

Graph neural networks (GNNs) [9, 11] are specifically designed to handle data which is encoded as graph structures. They exhibit a well motivated inductive bias for problems that can be formulated as interacting entities, as they frequently appear in science and engineering. In this context, objects translate to nodes in a graph, and their pairwise interactions are represented as edges between these nodes. Battaglia et al. [8] provide a generalized GNN framework for the applicability to a wide range of problem domains. Their proposed generalized GNN block is based on directed multi-graphs with attributes at different levels. The definition of a graph is a 3-tuple $G = (\mathbf{u}, V, E)$, which we further illustrate using the sun and planets of our solar system as an example. In this system, each mass body is modelled as a node of the graph, and their mutual gravitational interactions as edges between them. This example, studied in more detail in [38], shares similarities with our problem, but differs in terms of scale (planets vs. particles):

- $\mathbf{u}$ represents global or universal properties. A universal property in the solar system could be the gravitational constant or the overall gravitational field that influences all celestial bodies.
- $V = \{\mathbf{v}_i\}_{i=1:N^v}$ is the set of nodes, which represent celestial bodies such as the Sun and planets. The total number of nodes in the graph is N^v. $\mathbf{v}_i$ are the node features, which describe properties such as mass, position, or velocity of each celestial body in the system.
- $E = \left\{\left(\mathbf{e}_k, r_k, s_k\right)\right\}_{k=1:N^e}$ is the set of edges, describing relations between bodies. N^e represents the number of edges in the graph. $\mathbf{e}_k$ contains the edge features, such as the gravitational force acting between two bodies. An edge connects the receiver node with the index r_k to the sender node s_k, representing the direction of gravitational influence.

Using the terminology introduced in [8], a few computations are required to predict a target variable at the node level, considering influences from the connected nodes. These include two network blocks, namely an edge model ϕ^e and a node model ϕ^v, together with an aggregation function of messages on the node level, denoted by $\rho^{e\to v}$.

$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}) \qquad \textit{message computation} \tag{4}$$

$$\bar{\mathbf{e}}'_i = \rho^{e\to v}(\{\mathbf{e}'_k, r_k, s_k\}_{r_k=i,k=1:N^e}) \qquad \textit{message aggregation} \tag{5}$$

$$\mathbf{v}'_i = \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}) \qquad \textit{node update} \tag{6}$$

The node and edge models are comparably small ANN blocks, which are shared among all nodes and edges. In other words, all messages between two nodes are computed by the same edge model, and all node updates are computed by the same node network. In Eq. 4, a message is computed by the edge model, using the edge features, the features of the sending and receiving nodes as well as the universal property. This can be understood as the approximation of pairwise interactions between entities. All incoming messages of a node are aggregated in Eq. 5, requiring an aggregation function invariant to permutations of the input, such as elementwise summation or mean. The node update is learned by the node model, using the aggregated messages, the current node state and the universal property as input features.

Different applications of GNNs to systems concerning fluid mechanics have been presented in the recent literature. Sanchez-Gonzalez et al. [55] have introduced an encoder-decoder architecture based on graphs to model physical processes, including water dynamics, sand friction, and the behaviour of viscous material. Additionally, Li and Farimani [41] has developed two variations of GNNs, one emphasizing edges and the other focusing on nodes, to enhance the efficiency of Lagrangian fluid simulations. Network-based ML methods, however, typically lack interpretability, which is a significant drawback for model inference tasks, where understanding the underlying principles is as critical as achieving accurate predictions.

To overcome this issue, symbolic regression (SR) can be used to develop interpretable equations. Although the learning capability of SR is generally considered lower than that of network-based models, this method has the inherent advantage of generating mathematical expressions, thus improving the model explainability. In addition, a well-performing symbolic model can, in certain cases, generalize better and mitigate the extrapolation limitations often observed in ANNs. Cranmer et al. [18] proposed GNNs as an inductive bias for SR, aiming to overcome issues related to high-dimensional problems that can be modelled as interacting particle systems. In the first step, they approximated the pairwise interactions between particles using GNNs. Since the edge and node models are shared for all particles and interactions between them, this separable internal structure breaks originally high-dimensional problems into smaller subproblems, which are tractable for SR algorithms. In the next step, the edge and node models were approximated by an SR algorithm, operating on a reduced amount of input features compared to the original problem. This method was applied successfully to rediscover the gravity equation for international trade in [70], the collective behaviour of swarms [46], as well as the orbital mechanics of the solar system according to Newton's gravitation law [38]. We demonstrated the applicability of this two-step ML pipeline to learn symbolic expressions for the variation from the mean drag on particles in particle-laden flows in two previous studies [24, 50]. While earlier results were presented for the Stokes regime, in the present paper, we extend these findings to a broader range of $\mathrm{Re_p}$ and ϵ_p.

2.4. Symbolic regression

Symbolic regression learns symbolic expressions from data and, other than classical regression methods, does not assume a specific functional form a priori. The identification of a fitting model structure is part of the learning task, next to learning the coefficient values and selecting appropriate

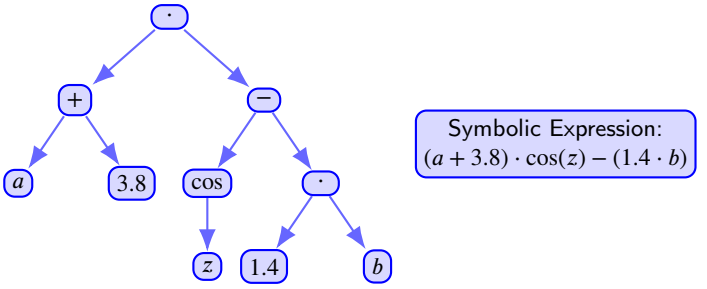


Figure 2: Representation of a symbolic expression as a syntax tree.

functions and features from the provided sets. Various techniques have been presented in the past to approach SR problems. A popular method for physics and fluids is sparse identification of nonlinear dynamical systems (SINDy), which exploits sparse regression to describe underlying principles of nonlinear dynamical systems [13, 21]. AI Feynman [66, 65] aims at identifying functions for physical systems, which often share characteristics such as symmetries and separability, and require the adherence of physical units. Recent developments in the area of SR are based on artificial neural networks (ANNs) to learn a symbolic model. Deep symbolic regression (DSR) [45] formulates SR as a reinforcement learning problem with an underlying recurrent neural network to generate equations as a sequence of tokens. Tenachi et al. [63] introduced a DSR approach for physics, incorporating unit constraints to guide the learning process. Dimensional analysis was applied as in situ constraints, which updated the probability distributions for the token selection to ensure unit-consistent operations and features. The area of ANN-based SR is currently emerging rapidly, with many new approaches frequently presented that are of relevance for science and engineering problems. This encompasses various approaches, such as the dynamic adjustment of the network architecture [40], the learning of ordinary [20] and partial differential equations [12], transformer-based architectures [32] as well as deep generative SR [31].

Some physics-inspired SR methods are constrained to particular system types or predefined equation structures, which can hinder their capacity to discover new or unexpected physical laws and limit their flexibility when applied to complex physical systems. Genetic programming (GP) is a long-standing method in the literature and can be considered a flexible approach for SR. It allows for extensive customization, enabling the algorithm to evolve a wide variety of equation forms and to adapt to different problem domains. Therefore, GP is the preferred method for SR in the approach presented in this paper. In the following, we will briefly introduce the fundamentals of GP and then focus on recent developments in this area, that are relevant for our approach.

GP is a method from the family of evolutionary algorithms, which operates on expressions that are represented as syntax trees (see Fig. 2). GP formally requires a set of terminals, denoted by $\mathcal{T}$, which are features or constants located at the leaf nodes of the tree, and a set of functions, denoted by $\mathcal{F}$, which are applied to these terminals. Being a population-based algorithm, GP evolves multiple expression simultaneously, which are combined and refined throughout evolution to converge to expressions that can accurately predict the target feature. A basic GP algorithm with an additional archive of non-dominated solutions according to Pareto dominance is outlined in Algorithm 1.

First, an initially random population of expressions is generated using $\mathcal{T}$ and $\mathcal{F}$. These are then evaluated with respect to the objective functions. Afterward, λ children are created and added to the offspring population. In the reproduction step, the evolutionary operators, namely crossover and mutation, are employed. These operate on the tree structures of the syntax tree, where crossover exchanges subtrees of two parents to create offspring, and mutation inserts small modifications into the

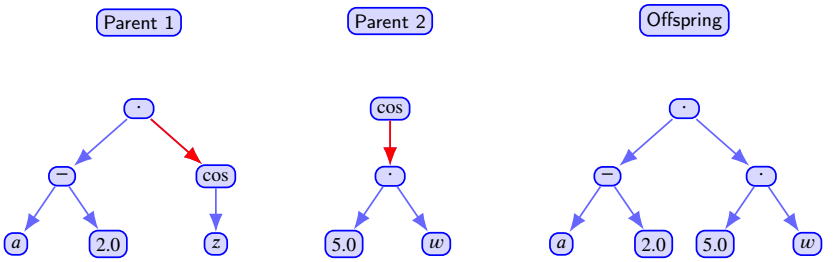
Algorithm 1: Genetic Programming Algorithm

Input : Training data X, population size μ, number of children λ, crossover probability p_c, mutation probability p_m, maximum generations G
Output: Solution population P

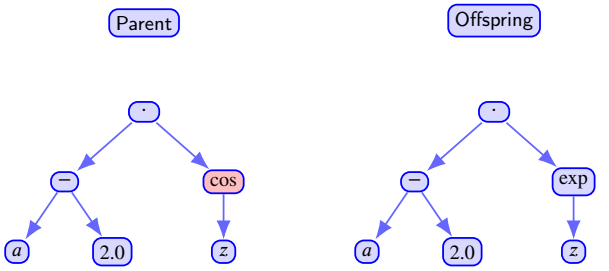
```
A ← ∅                                          // empty Pareto-dominance based archive
P ← initializePopulation(μ)                    // create random initial population
evaluate(P, X)                                 // compute objective values
for G generations do
    P_offspring ← ∅                            // create empty child population
    for i = 1 ... λ do
        genOp ← selectGenOp(p_c, p_m)          // select genetic operator
        p ← select(P)                          // select parent(s)
        o ← reproduce(p, genOp)                // create offspring
        P_offspring ← P_offspring ∪ o          // add to offspring population
    end
    evaluate(P_offspring, X)                   // evaluate offspring population
    P ← updatePopulation(P ∪ P_offspring)      // update population
    A ← updateArchive(A ∪ P_offspring)         // update non-dominated archive
end
return A                                       // return Hall of Fame
```

tree. Examples for crossover and mutation are given in Fig. 3. The offspring population is evaluated, before the parent population gets updated with the newly generated offspring. To this end, we employ a $\mu + \lambda$ reproduction scheme, where both, the parent population and all generated offsprings, are considered in the update. This process is repeated and expressions are refined iteratively until a stopping criterion, typically a maximum number of generations, is reached. Next to an error measure, the complexity of the equation is typically minimized to avoid the uncontrolled growth of equations that only have small improvements in terms of accuracy. This turns the problem into a multi-objective one, which affects the selection and update-steps of the algorithm. To this end, the multi-objective NSGA-II algorithm [22] is frequently employed, which optimizes multiple objectives simultaneously. Rather than a single optimal solution, a Pareto-optimal front of solutions, each satisfying the objectives to different degrees, is returned by the algorithm. At least one of the two steps, selection of parents and updating of the population, has to be fitness-based to ensure convergence to better solutions over time. Fitness-based in this context means that expressions with better objective values are more likely to get selected as parents, or to survive to the next generation. Finally, the algorithm returns an archive of non-dominated solutions, from which a decision maker will select one expression with a good trade-off between the objectives. This archive of the best solutions is typically referred to as the Hall of Fame (HoF).

State-of-the-art GP algorithms are frequently enhanced by components from the ML area to improve the computational efficiency and accuracy of the developed expressions. One important component concerns the fitting of new constants in an expression. This is typically implemented by an additional constant placeholder in the feature set. Before the evaluation of an equation, these placeholder constants are fitted using a multiple linear regression algorithm on top of GP. For example, the `PySR` [17] framework employs the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, and `TiSR` [42] and `Operon` [14] implement the Levenberg-Marquardt algorithm to fit new constants.



(a) One-point Crossover



(b) Point Mutation

Figure 3: Examples of crossover and mutation operation, where crossover points are marked as red arrows and mutation points as red nodes.

Similar to the area of PIML, the integration of domain-specific knowledge into GP algorithms has become increasingly important in recent research. By incorporating prior knowledge, expressions can be constrained to follow expected behaviours or patterns, while reducing the search space of possible equations and improving accuracy and interpretability. Moreover, this approach can help overcome the limitations of purely data-driven methods, such as lack of generalization and physical consistency. We will present a few notable publications on this topic subsequently, some of which will also be employed in our proposed approach.

One line of research concerns the inclusion of shape constraints in GP algorithms. Shape constraints can be useful when prior knowledge about the expected function behaviour is available, such as monotonicity, symmetry or convexity, but the data is not sufficient to fully reflect this behaviour [36]. Analysing the function behaviour using optimistic [15] and pessimistic approaches [27, 28] has been studied in the literature. Moreover, the application of appropriate constraint handling methods was investigated, such as minimizing the constraint violations as an additional objective using multi-objective optimization [15, 27, 29], or counter-example driven SR [15, 37].

Another important method to include prior knowledge is unit-aware GP, which aims at developing expressions that are conformal with physical units. This is especially relevant for problems in science and engineering, where measurements and their associated units are frequently available. Equations that, for example, add two quantities in meters and seconds are typically not plausible for such applications. Unit-aware GP propagates the units of the features through the syntax tree and detects unit violations. Again, constraint handling methods are required to address potential unit violations. These include adding a high penalty value to the primary objective for unit-violating equations [7, 43], multi-objective optimization [39, 72], as well as a repair mechanism to remove unit violations in expressions [34]. Another challenge is related to the dimensional analysis of equations that contain new constants, which typically have unknown units. However, all features and constants require units

to be considered in the dimensional analysis and propagated through the tree. Our recently proposed unit-aware GP approach helps to overcome this issue [52]. It treats the units of newly learned constants as jokers in the dimensional analysis and propagates them further through the tree.

Other methods to introduce knowledge into GP algorithms include prior assumptions about the functional structure, such as fixing parts of the function to a known sub-expression [4, 57], or seeding the initial equations with expected building blocks [56]. The pre-computation of expected combinations of functions and features and their inclusion in the feature set improved the accuracy of the final equations in [69, 72].

For the specific problem of particle-laden flows, our proposed approach builds upon our previous research on the development of symbolic models for particles in Stokes flow using GP. These include the flow past one [72] and two [49] spherical particles, as well as using GNNs as inductive bias for GP for larger particle arrangements in infinite [51] and periodic [24] flow domains. The goal in this paper is to assess the performance of the proposed framework on a broader range of Reynolds numbers and volume fractions.

3. Machine learning framework

3.1. Overall approach

With the two-step ML pipeline introduced subsequently, we aim to predict the variation from the mean drag force experienced by a particle. The mean drag is denoted by $\langle F_x \rangle$, and $\Delta F_{x,i}$ denotes the variation from the mean drag. Typically, the deviation from the mean drag on a particle is formulated as a function of the positions of a predefined number of neighbours, denoted by N_n.

$$\frac{\Delta F_{x,i}}{\langle F_x \rangle} = f_x(\varepsilon_\mathrm{p}, \mathrm{Re}_\mathrm{p}, \bar{r}_1, \theta_1, \varphi_1, \bar{r}_2, \theta_2, \varphi_2, \dots, \bar{r}_{N_n}, \theta_{N_n}, \varphi_{N_n}). \tag{7}$$

Depending on the number of neighbours, typically 20 or more, this problem formulation exhibits high-dimensional properties, which are often difficult to handle for data-driven ML methods. An initial approach using a fully connected neural network considering all features at the same time showed poor generalization capabilities [6], which can be attributed to the problem formulation. To improve the model generalization, we assume pairwise interactions between particles, which reduces the number of features for each pairwise interaction [58]. This moreover motivates the use of GNNs, as they are designed to model pairwise interactions between particles while preserving the relationships between them. Consequently, our target model aggregates the pairwise interactions, where each individual pairwise interaction is approximated by a function $\tilde{f}_x$.

$$\frac{\Delta F_{x,i}}{\langle F_x \rangle} = \sum_{j=1}^{N_\mathrm{n}} \tilde{f}_x(\varepsilon_\mathrm{p}, \mathrm{Re}_\mathrm{p}, \bar{r}_j, \theta_j, \varphi_j). \tag{8}$$

A pairwise interaction is thus a function of the volume fraction, the Reynolds number, and the relative distance of the neighbour particle from the particle of interest in spherical coordinates. The particle of interest is always located at the origin of the coordinate system.

The overall learning pipeline is displayed in Fig. 4. The GNN is fed with data from the particle-resolved simulations and learns the underlying pairwise interactions between the particle of interest and its neighbours. The edge model Φ^e is shared to this end, so that each pairwise interaction is approximated by the same model. The pairwise interactions are aggregated internally to predict the target feature $\frac{\Delta F_{x,i}}{\langle F_x \rangle}$. In the previously introduced structure of the GNN, these operations correspond to

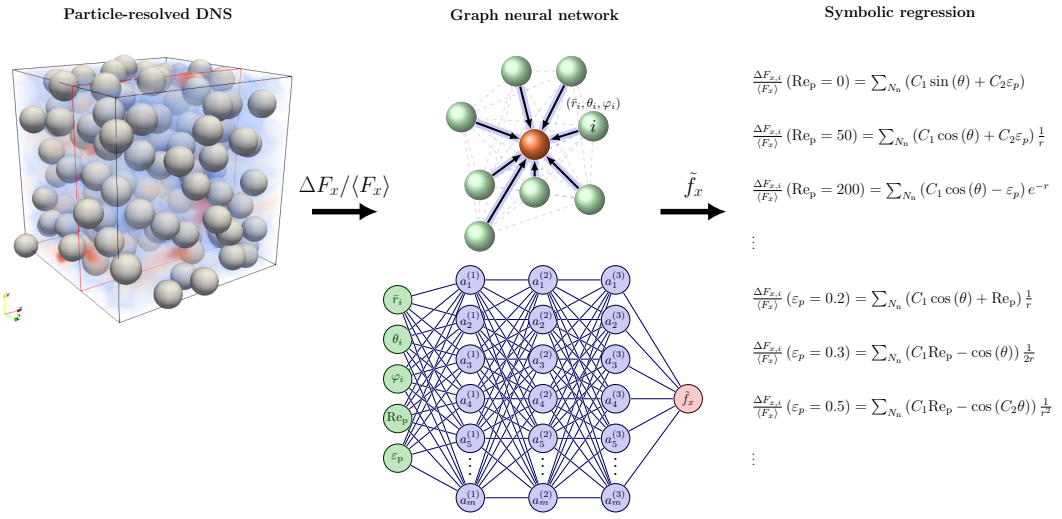


Figure 4: Symbolic models are generated from simulation data using a GNN as a surrogate model. Physical particles in the simulation translate to nodes in the GNN.

Table 2
Exemplary intermediate data products as extracted from the GNN. Each row corresponds to one pairwise interaction between a neighbor and the particle of interest. These data serve as training data for the subsequent GP algorithm, with the target variable y.

$\bar{r}_j$	θ_j	φ_j	ε_p	$\mathrm{Re_p}$	$\tilde{f}_x$
1.051813	1.998868	0.660557	0.4	188.415	0.033569
1.302048	2.127855	−1.610042	0.4	281.933	0.004412
1.857303	0.652636	2.494535	0.4	47.628	−0.011249
1.503460	2.399730	2.959001	0.3	4.863	−0.053079
2.092653	0.131259	3.052247	0.3	281.933	−0.090326
1.100217	2.058815	2.576300	0.5	95.304	0.016363
1.975268	0.748523	−2.570214	0.2	188.415	−0.015988
3.202558	2.437589	−1.476869	0.1	0.094	−0.006449
1.005887	1.695564	0.834028	0.4	95.304	0.186315
...	...	...	...	...	...

Eqs. 4 and 5. Eq. 6 is not considered in our approach, as our experiments from previous studies have indicated that the accuracy does not improve significantly by an additional node update [51, 24]. We employ a single GNN to learn the underlying interactions for all $\mathrm{Re_p}$ and ε_p. In this way, we aim to ensure that both the GNN and symbolic models maintain a consistent pattern across the entire range of input data, which is important for their generalizability and reliability. Once the training phase of the GNN is finished, a dataset with intermediate data products is generated, which includes the input and output features of the edge model Φ^e. Examples for these intermediate data products are shown in Tab. 2. A GP algorithm uses this data to replace the edge model with symbolic expressions. Since the training data contains one pairwise interaction per training sample, the number of input features decreases compared to the problem formulated without pairwise interactions. While the GNN employed samples from all available $\mathrm{Re_p}$ and ε_p, we know from previous studies that the GP algorithm has difficulties identifying a universal equation that covers the entire range of data. Therefore, we

filter the dataset and develop separate equations, once for each $\mathrm{Re_p}$ with ε_p as a variable, and once for each ε_p with $\mathrm{Re_p}$ as a variable. This reduces the space of searched equations tremendously, and simplifies the learning for GP. An important prerequisite to this end is that the GNN has to achieve a certain accuracy; otherwise, errors in the first step will be further propagated to the resulting symbolic expressions.

The GNN module of our approach generally has a high similarity with the physics-inspired neural network by Seyed-Ahmadi and Wachs [58], which approximates pairwise interactions with small network blocks. However, the use of GNNs embedded in the `PyTorch geometric` library as in our approach allows for easier customization and extension in the future, such as trinary interactions between particles or alternative neighbourhood structures.

3.2. Algorithm settings

In the following, we will describe the settings of the previously introduced ML pipeline as they are employed in our experiments. The GNN is trained using three fully connected layers, with rectified linear unit (ReLU) activations in the first two layers and a final hyperbolic tangent activation. Using ReLU in the first layers helps to avoid the problem of vanishing gradients, which is a known drawback of the hyperbolic tangent activation. ReLU moreover allows for sparse activations, since $\mathrm{ReLU}(x) = \max\{0, x\}$, which can be useful for covering diverse datasets like ours, containing various $\mathrm{Re_p}$ and ε_p. The final hyperbolic tangent activation constrains the output range, which is meaningful for our problem, as forces are naturally bounded. In our previous studies [51, 24], we trained separate GNNs for each ε_p in Stokes flow at $\mathrm{Re_p} = 0$ and employed two hidden layers to this end. Here, we introduced an additional layer to better accommodate the increased number of parameters in the data. Generally, the network size must be sufficiently large to capture the underlying interactions between particles, while remaining compact enough to ensure good generalization. We selected 30 neurons per hidden layer, which we also employed in previous studies [51, 24] and has shown to be a good compromise between model expressiveness and generalization. A dropout probability of 0.05 is applied after the second ReLU layer to prevent overfitting while maintaining stable training dynamics. Training is conducted using the Adam optimizer with an initial learning rate of 0.002 for 10,000 epochs, with a batch size of 64. Each epoch corresponds to a complete pass through the training dataset. The loss function is the mean-squared error (MSE), which is a useful loss function for regression problems. The training features include the relative locations of the neighbouring particles in spherical coordinates, $\bar{r}_j$, θ_j, and φ_j, as well as $\mathrm{Re_p}$ and ε_p. Our implementation is based on `PyTorch` [44] for general neural network operations, and `PyTorch geometric` [25] for tasks specific for GNNs. Tab. 3 summarizes these experiment settings.

The GP algorithm is applied once the training of the GNN is finished, and intermediate data products have been generated. We employ a population of 500 expressions, from which 500 children are created in each generation using genetic operators like crossover and mutation. From this combined pool of 1000 expressions, the best 500 according to the NSGA-II algorithm survive to the next generation, where this process is repeated for 200 generations. We set an upper limit of 30 for the complexity of an equation to avoid bloat and reduce the search space to a reasonable size. The optimization objectives applied to the population at the selection step include the MSE as the primary objective, supported by three additional objectives: First, the Spearman correlation, as previous studies indicated that it can help to keep solutions with high correlation but also high error in the population, and allow for further refinement in the following generations. Second, a dimension penalty is implemented, which aims to minimize operations that violate physical units. We consider the units as displayed in Tab. 4, where [−] represents dimensionless quantities and [?] is a joker unit as introduced by tuneable constants that come without an assigned unit. We want to mention that we assign a unit of [m] to the distance from the centre particle, $\bar{r}_j$, even though this feature is provided in a normalized and dimensionless format in the training features. We do this to reflect the semantics of the feature as

Table 3
Parameter settings used to train the GNN

GNN parameter	Settings
Φ^e	3 fully connected layers
Nonlinearity	ReLU — ReLU — hyperbolic tangent
Dropout probability	0.05, applied after the second ReLU layer
Number of neurons	30 per hidden layer
Initial learning rate	0.002
Training features	$\bar{r}_j$, θ_j, φ_j, Re_p, ε_p
Target feature	$\frac{\Delta F_{x,i}}{\langle F_x \rangle}$
Optimizer	Adam
Training duration	10,000 epochs
Batch size	64
Loss function	MSE
Implementation	`PyTorch` [44], `PyTorch geometric` [25]

a distance in the algorithm and avoid operations such as $\sin(\bar{r}_j)$. Such operations would be allowed if $\bar{r}$ was a dimensionless unit; however, they lack physical meaning from a fluid mechanics perspective. More details on our multi-objective unit-aware GP approach, which includes the mentioned joker units in the dimensional analysis, can be found in our previous work [52]. Lastly, we aim to minimize complexity since simpler solutions typically tend to generalize better. This objective enables the simultaneous optimization of MSE across all complexity levels during evolution. The Spearman correlation is not included in the HoF, since it is designed to store only those solutions that already perform well in terms of MSE and do not require further refinement.

Since we develop equations either for each ε_p with Re_p as a variable, or for each Re_p with ε_p as a variable, only one of the two are included in the feature set. The used set of functions comprises basic arithmetic operations, as well as trigonometric, logarithmic and exponential functions. Moreover, a power operator, denoted by $\circ^\circ$ is employed, which takes two input arguments, the basis and the exponent, where the latter can include a term consisting of multiple operations.

Custom complexity values are employed to reflect the complexity of operations in the complexity of the expression. While features are assigned a complexity of one, constants have a complexity of two to reduce the excessive use of constants in equations. Basic arithmetic operations are considered less complex than non-linear transformations. In addition, we employ measures to avoid combinations of operations that would be in agreement with pure dimensional analysis, but not meaningful from a practical perspective. To this end, we prohibit the nesting of certain functions, such as $\sin(\sin(\circ))$ and other functions. Tab. 4 outlines these functions and summarizes the previously described algorithm settings. Our GP algorithm is implemented in the `TiSR` framework, which was initially proposed for SR for thermodynamics applications. However, it is also well-suited to our problem due to its high degree of customizability and integration of state-of-the-art algorithmic components [42]. In general, various other SR frameworks could be employed for the SR step of our ML pipeline, such as `PySR` [17] or `Operon` [14]. These are, however, more challenging to extend with customized algorithmic components such as the dimensional analysis considering unknown constants, which are essential for our approach.

3.3. Data preprocessing

We have discussed some data-related preprocessing steps in the previous sections. In the following, we summarize how the original data is processed before both training the GNN and the GP algorithm. We set the number of neighbouring particles considered predicting the target variable to 30. Related

Table 4
Parameter settings used in the GP algorithm

GP parameter	Settings
Population size	500
Training duration	200 generations
Max. complexity of expression	30
Optimization objectives	MSE, Spearman correlation, dimension penalty, complexity
HoF objectives	MSE, dimension penalty, complexity
Training features	$\bar{r}_j$ [m], θ_j [−], φ_j [−], Re_p [−] or ε_p [−], tuneable constants [?]
Target feature	$\frac{\Delta F_{x,i}}{\langle F_x \rangle}$ [−]
Function set $\mathcal{F}$	$\{+, -, \cdot, \div, \circ^{(\circ)}, \sin(\circ), \cos(\circ), \circ^{(\circ)}, e^{(\circ)}, \log(\circ)\}$
Complexity values	
Features	1
Constants	2
$+, -, \cdot, \div$	1
$\circ^{(\circ)}$, $\sin(\circ)$, $\cos(\circ)$, e°, $\log(\circ)$	2
Prohibited function nesting	$\sin \to \{\sin, \cos\}$, $\cos \to \{\sin, \cos\}$, $\exp \to \{\exp\}$, $\log \to \{\log\}$, $\circ^\circ \to \{\circ^\circ\}$
Implementation	`TiSR` [42]

studies suggested diminishing returns in terms of accuracy when more neighbours are considered, so that 30 is a good compromise of accuracy and computational cost [60]. In a first step, all relative positions of neighbouring particles are transformed to a spherical coordinate system $(\bar{r}_j, \theta_j, \varphi_j)$, with the particle of interest located at the origin. θ_j represents the polar angle relative to the flow direction, while φ_j is the azimuthal angle in a plane normal to the flow direction. The distance $\bar{r}_j$ is normalized by the particle diameter to facilitate comparison and analysis across different particle sizes and configurations. Compared to Cartesian coordinates, this representation simplifies the problem, particularly for the GP algorithm, as it allows the direct application of trigonometric functions to the input features without requiring intermediate transformations.

Overall, 13,736 particle arrangements are available, which are evenly distributed over the combinations of eight Re_p and six ε_p. Initially, the dataset is split into 70% training and 30% test data. Thereby, it is ensured that the distribution of Re_p and ε_p is even among both sets to avoid over- or underrepresentation of certain configurations in the training or testing phase. Both sets are augmented by seven incremental rotations of each particle arrangement of $\frac{\pi}{4}$ around the axis of the free stream to cover each quadrant in the coordinate system. In this way, the symmetry inherent to the underlying problem is reflected in the datasets, and more data samples are available [59]. With the augmentation after the dataset split, it is ensured that the datasets are mutually exclusive, and the test set does not include samples that are rotations of samples from the training set. Preliminary experiments suggest a slight improvement in accuracy when applying a logarithmic transformation to the Reynolds number, $\log_{10}(\mathrm{Re}_\mathrm{p})$. This transformation can help rescale the feature to make it more suitable for numerical optimization by compressing large variations in Re_p.

During the training of the GNN, 80% of the training data is used for tuning the trainable parameters of the network, and 20% is used as a validation set to detect potential overfitting and trigger early stopping. Again, we make sure that these datasets, including their rotations, are mutually exclusive. The intermediate data products are generated from the samples from the validation datasets. Since 30 pairwise interactions per configuration are stored, this generates a dataset with 658,560 samples covering pairwise interactions for all Re_p and ε_p. To approximate these pairwise interactions with symbolic expressions, the datasets are filtered by ε_p and by Re_p before the training of the GP algorithm.

During the training of the GP algorithm, the training data is again split into 80% for evaluating and fitting the constants of the expression, and 20% to trigger early stopping during parameter fitting. The test set is a holdout dataset only employed to assess the final performance of the trained models.

4. Experiments and results

Using the algorithm settings from Sec. 3.2 and data preprocessing steps as described in Sec. 3.3, we conducted experiments for the combination of eight $\mathrm{Re_p}$ and six ε_p as introduced in Sec. 2.1. In the following, we will present and analyse the performance of the GNN and the derived symbolic models.

4.1. GNN results

The GNN training was repeated eleven times, to account for variations in the training process due to the stochastic nature of the parameter initialization. In this way, we aimed to ensure the reliability of the method and verify that the results do not arise from random chance. The GNN plays a crucial role in our approach by decomposing the force deviation into pairwise contributions, which reduces the problem complexity tremendously for the subsequent GP step. Furthermore, we employ a single GNN model on data spanning all $\mathrm{Re_p}$ and ε_p, which we expect to improve generalization and to approximate a unified representation of the force deviations. In the GP step, we then develop separate symbolic models: one for each $\mathrm{Re_p}$ with ε_p as a variable and one for each ε_p with $\mathrm{Re_p}$ as a variable. This separation is intended to reduce the search space for the GP algorithm, as previous experiments have shown difficulties in learning accurate equations that remain valid across multiple ε_p and $\mathrm{Re_p}$. We expect that a unified GNN representation, from which the GP training data is extracted, helps to learn symbolic models that are more consistent and structurally similar, despite being trained on separate datasets. We analyze the performance of the GNN and the symbolic models based on the coefficient of determination, R^2. We employ a computation as presented in [2, 3], where R^2 is defined as

$$R^2 = 1 - \frac{\sum_{i=1}^{N} \left[\Delta F_{x,i}^{\mathrm{true}} - \Delta F_{x,i}^{\mathrm{model}} \right]^2}{\sum_{i=1}^{N} \left[\Delta F_{x,i}^{\mathrm{true}} - \langle \Delta F_{x}^{\mathrm{true}} \rangle \right]^2}, \tag{9}$$

where $\Delta F_{x,i}^{\mathrm{model}}$ and $F_{x,i}^{\mathrm{true}}$ are the predicted and ground truth force deviations of the $i-$th particle in the dataset, respectively. The term $\langle \Delta F_{x}^{\mathrm{true}} \rangle$ represents the mean force deviation, which is expected to be approximately zero since we assume that there exists a perfect mean drag model. The value R^2 generally indicates the portion of the force deviations that are captured by the model as compared to the standard deviation of the drag force. A perfect model should achieve $R^2 = 1$.

Fig. 5 shows the distribution of R^2 values across eleven repeated runs of the GNN, disaggregated by the different values of $\mathrm{Re_p}$. The box plots represent the distributions of R^2 scores, showing the median, the interquartile range and the whiskers of each distribution. The markers indicate the specific R^2 values of the model that was selected as the "best" model and used to generate the training data for the GP algorithm: blue triangles represent the R^2 values for the training dataset, red circles represent the R^2 values for the test dataset. Overall, the R^2 values are between 0.783 and 0.861 on the training set and 0.641 and 0.799 on the test set for the different values of $\mathrm{Re_p}$. Over the different ε_p values, the highest R^2 values are 0.880 on the training and 0.794 on the test set, and the lowest values are 0.767 and 0.630 respectively. These values are comparable or better than those reported in related studies [59, 60], suggesting that the GNN approach generally achieves a reasonable predictive performance. By displaying the disaggregated data for $\mathrm{Re_p}$ and ε_p, we aim to analyze whether a specific parameter configuration is over- or underfitted by the network, despite the equal share of

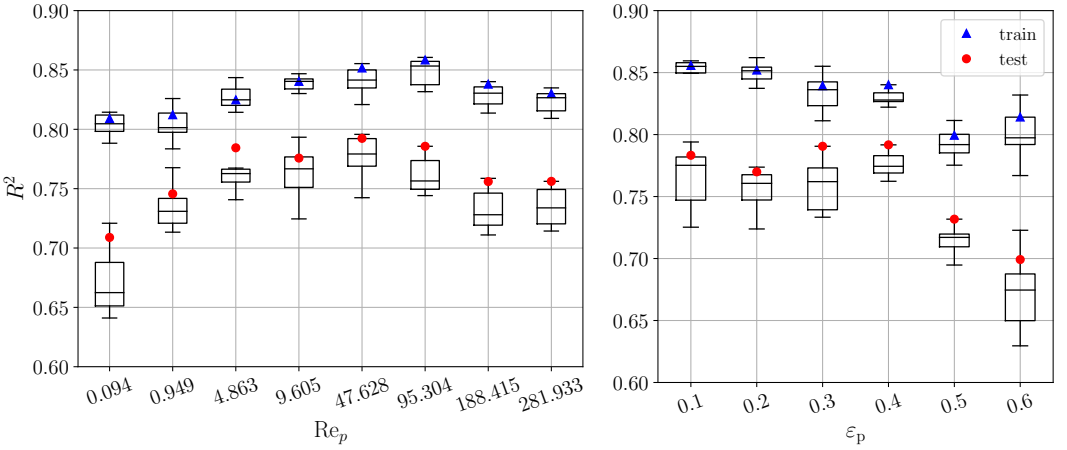


Figure 5: Distribution of R^2 over 11 repeated runs of the GNN, disaggregated by Re_p (left) and ε_p (right). The markers indicate the R^2 values of the selected model that was used to generate the training data for the GP algorithm.

Table 5
R^2 scores of the selected GNN model. The entire dataset was disaggregated, once by ε_p, and once by Re_p, to evaluate the performance for different flow regimes and configurations.

(a) R^2 per ε_p

ε_p	R^2 Train	R^2 Test
0.102	0.8558	0.7833
0.200	0.8520	0.7699
0.300	0.8394	0.7906
0.404	0.8402	0.7917
0.500	0.7994	0.7318
0.612	0.8141	0.6992

(b) R^2 per Re_p

Re_p	R^2 Train	R^2 Test
0.094	0.8091	0.7089
0.949	0.8123	0.7457
4.863	0.8249	0.7844
9.605	0.8404	0.7758
47.628	0.8516	0.7924
95.304	0.8584	0.7857
188.415	0.8380	0.7560
281.933	0.8301	0.7562

each parameter configuration in the training data. We observe that the highest R^2 scores of up to 0.799 on the test set are achieved for mid-ranged Re_p values, i.e., 4.863, 9.605, 47.628, and 95.304. At the lower and higher ends of the Re_p range, the GNN yields lower accuracies and higher distances between train and test sets, with the worst performing GNN having an R^2 of 0.641 for $\mathrm{Re}_\mathrm{p} = 0.094$. On the datasets disaggregated by ε_p, we observe similarly high R^2 scores for ε_p between 0.1 and 0.4. For the two highest ε_p, i.e., 0.5 and 0.6, the accuracy on the test set drops, indicating that the GNN struggles to generalize well in denser regimes.

In both plots in Fig. 5, we observe a difference between train and test R^2 spanning over all Re_p and ε_p values. However, during the training of the GNNs, the MSE on the validation dataset decreased along with the training MSE, indicating that the model was learning the underlying patterns in the data without overfitting excessively to the training set. The markers in Fig. 5 indicate the R^2 values of the GNN that were selected among the eleven models for the extraction of the pairwise interactions, which serve a training data for the GP algorithm. For each of the eleven models, we computed the maximum

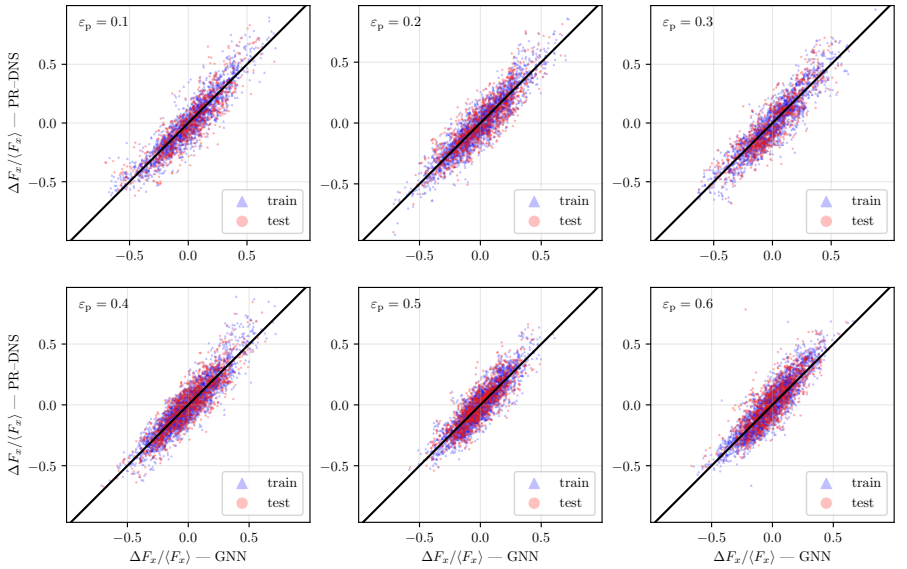


Figure 6: Correlation plots of the predicted deviation from the mean drag as predicted by the GNN (x-axis) and ground truth from the particle-resolved simulation (y-axis) for different ε_p. Training and test data are shown in blue and red markers, respectively. The diagonal line represents the perfect correlation line.

difference between train and test R^2 over the range of Re_p. The model with the lowest maximum difference was then selected. With this, we aimed to select the best generalized model among the trained GNNs, which is an important aspect to develop symbolic models with similar structure. The exact numerical R^2 values for the selected GNN model can be found in Tab. 5. To further investigate the effects of varying Re_p and ε_p on the GNN performance, we analyze the correlation plots obtained by this selected model.

Fig. 6 displays the correlation plots of the model prediction and the particle-resolved simulation data for different ε_p, and Fig. 7 for different Re_p. Each marker in the plots represents the prediction result obtained for a single sample of the dataset. The black line indicates perfect agreement between PR-DNS data and GNN predictions. The model demonstrates good correlations with the particle-resolved simulation datasets for both training and test data. For those ε_p with lower R^2 scores, i.e., $\varepsilon_p = \{0.5, 0.6\}$, we observe more markers with a higher distance to the black line. Similarly, $\mathrm{Re}_p \approx \{10, 50, 100\}$ with the highest R^2 scores in Tab. 5b exhibit markers more concentrated around the black line in Fig. 7.

As mentioned previously, one significant drawback of purely data-driven ML methods is their lack of interpretability, which limits our understanding of the underlying computations. To address this issue, various approaches have been proposed, including the analysis of feature importance. This analysis measures the impact on the model predictions or error when one or more features are removed or altered. We employ a permutation-based feature importance analysis to understand how influential each of the features are in the predictions of the GNN; with this, we moreover gain insights into the features that will most likely be important in the symbolic models, as these are derived from the GNN. In the permutation-based approach, we shuffle one feature column in the dataset at a time, while

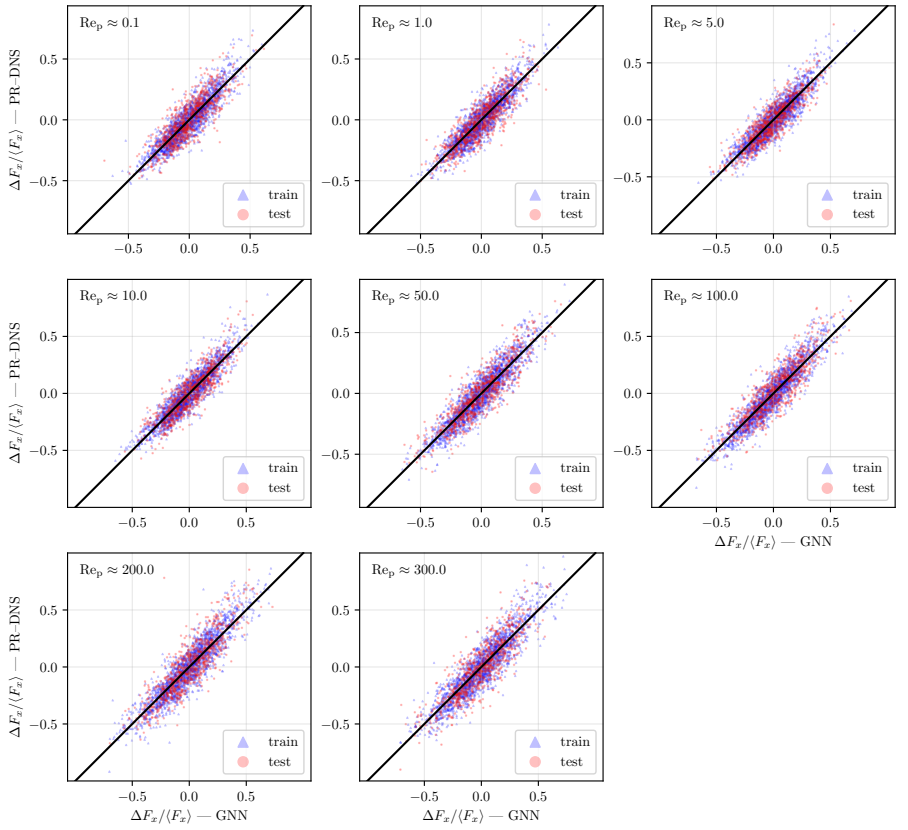


Figure 7: Correlation plots of the predicted deviation from the mean drag as predicted by the GNN (x-axis) and ground truth from the particle-resolved simulation (y-axis) for different Re_p. Training and test data are shown in blue and red markers, respectively. The diagonal line represents the perfect correlation line.

keeping the other columns unchanged. This column-wise shuffling preserves the overall distribution of feature values compared to the original dataset. However, it is possible that the permuted dataset may contain combinations of extreme values of multiple features, which were not present in the original data. Despite this limitation, this relatively simple method helps to qualitatively estimate how different features contribute to the model predictions. To avoid misleading results due to randomness in the shuffling process, we repeat the process ten times for each feature and present the mean increase in terms of MSE in Fig. 8. The datasets for this analysis contain all combinations of ϵ_p and Re_p.

We observe a high importance of $\bar{r}$ and θ, while the other three features are by orders of magnitude less important. In other words, the specific flow regime as a combination of Re_p and ϵ_p has a lower influence on the model error than the distance and polar angle of the neighbouring particles. Since we incorporate the pairwise assumption, we expect the azimuthal angle ϕ to not have any influence on the prediction due to rotational symmetry around the flow direction when considering a pair of particles at

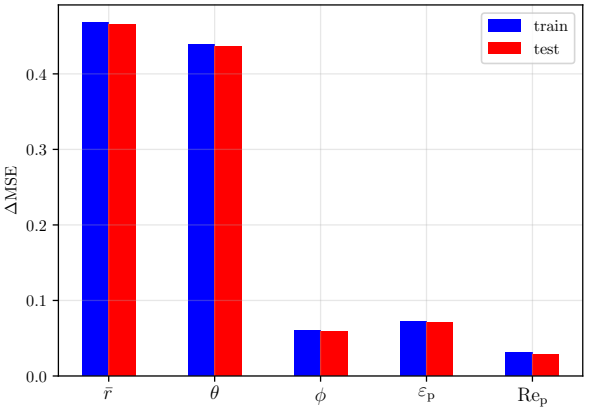


Figure 8: Feature importance analysis of the selected GNN model: the y-axis indicates how much the MSE increases, ΔMSE, when the feature on the x-axis is permuted. We report ΔMSE over ten permutations per feature to eliminate randomness in the shuffling.

a time. This expectation aligns with the equations we previously developed for the Stokes flow, where ϕ did not appear in any equation [24]. However, the MSE of the GNN slightly increases when the azimuthal angle is permuted, indicating that some influence is present. Overall, the importance values for the train and test sets are similar, which further confirms our assumption that no severe overfitting to the training data has occurred.

Considering the overall good correlation of the GNN predictions with the PD-DNS data (see Figs. 6 and 7) as well as the analysis of important features expected to be present in the equations, we extract the pairwise interactions from the selected GNN model and train our proposed GP algorithm to replace the edge model with symbolic expressions.

4.2. GP results

1 In the following, we report and examine the resulting symbolic models as developed by the GP algorithm. As already mentioned, we train the GP algorithm on the pairwise contributions estimated by the GNN. For each ε_p and Re_p, we repeated the GP algorithm eleven times to account for the non-deterministic nature of the algorithm. Each run returned a Hall-of-Fame (HoF) with non-dominated solutions each satisfying the HoF objectives, MSE, complexity, and dimension error, to varying extents. We first filter the HoFs for equations without unit violations. Then, we apply the "top x%" selection strategy similar to the one proposed in [26] to effectively reduce the number of equations from which we make a final selection. For each run, we select the three least complex equations within the top 95% in terms of R^2 on the test set. The symbolic expressions reported below represent a chosen sample of these expressions that are relatively less complex and achieve reasonable accuracy. It should be noted that all the numerical coefficients in the expressions are replaced with an arbitrary coefficient variable, C_k, to reveal any existing patterns.

Table 6 shows the results of the GP algorithm when trained with datasets at a given Re_p. By examining the expressions, it is clear that all expressions include $\bar{r}$ and θ, while φ never appears in any of them. In the majority of the expressions, $\bar{r}$ has a diminishing impact on the pairwise contribution, i.e., distant neighbours have less pairwise contribution. The exclusion of φ is expected as a direct result of incorporating the pairwise assumption in GNN, which is successfully captured by the GP algorithm. The absence of φ is also in line with feature importance analysis performed on the trained GNN. Since these expressions result from applying the GP algorithm to datasets of a given Re_p at a time, expressions cannot have Re_p as there is no variation in Re_p across the training samples. Another

Table 6
Sample of GP results when trained on datasets at a given $\mathrm{Re_p}$

$\mathrm{Re_p}$	R^2	Equation
0.094	0.5394	$\left(C_2 + e^{\sin(\theta)}\right)\left(C_0 + \varepsilon_\mathrm{p}\left(C_1 + \frac{C_3}{\bar{r}}\right)\right)$
0.949	0.6109	$C_2\left(C_0 - \sin^{C_1}(\theta)\right)\left(\varepsilon_\mathrm{p}\left(1 - \frac{C_4}{\bar{r}}\right) - \frac{C_3}{\bar{r}}\right)$
4.863	0.5262	$\left(C_0 + \varepsilon_\mathrm{p}\left(C_1 + \frac{C_3}{\bar{r}}\right) + \frac{C_1\varepsilon_\mathrm{p}^2}{\bar{r}}\right)\left(C_2 - e^{\sin(C_4+\theta)}\right)$
9.605	0.5574	$\left(C_5 + \cos\left(C_0\theta + C_2\right)\right)\left(\frac{C_1}{\bar{r}} + C_4 + C_3\varepsilon_\mathrm{p}\left(C_6 + \bar{r}\right)\right)$
47.628,	0.5865	$\left(C_2 - \sin\left(C_0\theta + C_5\right)\right)\left(\frac{C_1}{\bar{r}} - \varepsilon_\mathrm{p}\left(C_3 + \frac{C_4}{\bar{r}}\right)\right)$
95.304	0.5903	$\left(\frac{C_0}{\bar{r}} + \varepsilon_\mathrm{p}\left(C_1 + \frac{C_3}{\bar{r}}\right)\right)\left(C_2 + \sin\left(C_4 + C_5\theta\right)\right)$
188.415	0.4591	$\left(C_0 + \frac{\sin\left(C_3+C_4\theta\right)}{\bar{r}}\right)\left(C_1 + \frac{C_2}{\bar{r}}\right)$
281.933	0.4943	$\bar{r}^{C_2}\left(\frac{C_3\cos\left(C_1+\theta\right)}{\bar{r}} + C_4\right)\cos\left(C_0 + \theta\right)$

Table 7
Sample of GP results when trained on datasets at a given ε_p

ε_p	R^2	Equation
0.1	0.6359	$\left(C_1 - \sin\left(C_5 + \frac{C_0 - \theta\left(C_3+\bar{r}\right)}{\bar{r}}\right)\right)\left(C_2 + \frac{C_4}{\bar{r}}\right)$
0.2	0.6269	$\bar{r}^{C_3}\left(C_0 + \frac{C_2\theta}{C_4+\mathrm{Re_p}}\right)\left(C_1 - \cos(\theta)\right)$
0.3	0.5806	$\left(C_3 - \sin\left(C_2 + C_4\theta\right)\right)\left(\frac{C_0}{\bar{r}} + C_1\right)$
0.4	0.5226	$\left(C_1 + \frac{C_3}{\bar{r}}\right)\left(C_2 + \cos\left(C_0\theta\right)\right)$
0.5	0.5237	$\left(C_0 + \frac{C_3}{\bar{r}}\right)\left(C_4\bar{r} + \sin\left(C_1\theta + C_2\right)\right)$
0.6	0.5080	$\left(\frac{C_1}{\bar{r}} + C_4\right)\left(C_0 + \frac{C_3}{\bar{r}} - \cos\left(C_2\theta\right)\right)$

observation is that ε_p appears less frequently than expected. This can be justified by the fact that both $\bar{r}$ and ε_p encompass similar characteristics of the arrangement. For denser arrangements of particles, where ε_p is high, $\bar{r}$ of the nearest neighbours is relatively lower and vice versa.

Repeated patterns can be recognized in the GP expressions, some of which are highlighted in yellow. These patterns and sub-expressions can potentially be used to tailor a symbolic expression that mimics the behaviour of the GNN, while satisfying meaningful constraints [24]. Although the GP expressions are compact and traceable, there is still a gap in terms of predictive capabilities between the GP expression and the trained GNN. This is evident by comparing the R^2 values reported in Table 5b for the GNN and their counterparts of the symbolic models in Table 6.

In Table 7, a sample of the expressions generated by the GP algorithm when trained on datasets at a given ε_p is presented. Similar to the results in Table 6, these expressions exhibit recurring patterns. These patterns could serve as building blocks for constructing more complex and generalized symbolic models. $\bar{r}$ and θ remain prevalent in all expressions. Contrary to expectations, $\mathrm{Re_p}$ does not appear frequently in the expressions generated by the GP algorithm, although $\mathrm{Re_p}$ is allowed to vary in the considered datasets. This is, however, consistent with the feature importance study conducted on the GNN and confirms that the GP symbolic models, to some extent, replicate the behaviour of the GNN.

The R^2 values in Table 7 indicate the same gap in predictive accuracy of the GP expressions when compared against the GNN performance as reported in Table 5a. R^2 decreases as ε_p increases, which is a trend observed in the GNN performance as well. Decrease in accuracy for denser particle arrangements is expected generally for models that rely on pairwise assumption, as the assumption loses its validity in denser arrangements [3]. Overall, the GP results demonstrate the potential to uncover interpretable symbolic models for the drag variation. However, further refinement and incorporation of physical and/or mathematical constraints are necessary to limit the search space for the GP algorithm.

5. Summary and conclusions

In this paper, we extend our previous work on deterministic drag modelling using a machine learning framework to higher Reynolds number regimes and denser particle volume fractions. Using particle-resolved direct numerical simulation (PR-DNS) data, we train a graph neural network (GNN) to estimate the pairwise contribution to the drag variation. Then a genetic programming (GP) algorithm is employed to derive drag deviation models in the form of symbolic expressions. The GNN achieves a coefficient of determination, R^2, comparable to those reported in related studies, despite training a single network for all flow regimes and volume fractions. This demonstrates the robustness and predictive capabilities of the GNN, as it successfully captured the drag variation across a wide range of Reynolds numbers Re_p and volume fractions ε_p.

A feature importance analysis is conducted on the trained GNN using the method of feature permutation. This analysis indicates that the normalized inter-particle distance, $\bar{r}$, and the polar angle θ of the neighbors have more influence on the drag deviations compared to the azimuthal angle, φ. The analysis also indicates that the global flow conditions ($\mathrm{Re}_p, \varepsilon_p$), usually used in mean drag models, are relatively less impactful on the drag deviations of each individual particle. This suggests that the local arrangement of particles plays a critical role in drag variation, while the global flow conditions have a secondary effect.

The GP algorithm generated symbolic models with lower R^2 values compared to the GNN, although GP was applied on datasets of a constant Re_p and ε_p separately. When applied to the constant ε_p datasets, symbolic models showed almost no dependency on Re_p, which is in-line with the low feature importance of Re_p in GNN. This discrepancy highlights a limitation of the current approach and suggests the need for future research to better account for global input features. One potential solution is the multi-view approach [54], where separate constants in the equations for each Re and VF are fitted during the training of the GP algorithm.

This study represents the first attempt to develop symbolic models for determining the drag variations on particles in assemblies with higher Re_p and ε_p. Despite the varying flow conditions, recurring patterns and sub-expressions can be identified in the GP results. These patters and sub-expressions can serve as the foundation for developing a generalized symbolic drag deviation model that applies across different ($\mathrm{Re}_p, \varepsilon_p$).

Further refinements to the symbolic regression approach can be achieved by restricting the input features from being arguments to certain GP building blocks, thereby limiting the search space. Additionally, incorporating shape constraints, as introduced in Sec. 2.4, could ensure that the symbolic models adhere to physical limits, such as the vanishing effect of neighboring particles as the distance $\bar{r}$ approaches infinity. These constraints would not only enhance the robustness of the models, but also reduce the complexity of the search space for the GP algorithm.

In conclusion, this work demonstrates the potential of combining GNNs and symbolic regression to develop interpretable drag models for particle-laden flows. While the GNN provides high accuracy, the symbolic models offer a potential insight into the underlying physics and can be more easily integrated into simulation frameworks. Future work should focus on further refining the search space

for the GP algorithm. This study paves the way for further advancements in the development of interpretable and accurate drag models for complex fluid-particle systems.

6. Acknowledgments

The authors acknowledge the funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project number 466092867 within the Priority Programme "SPP 2331: Machine Learning in Chemical Engineering". They also acknowledge the computing time granted on the HPC-cluster Sofja at OvGU Magdeburg.

References

[1] Akiki, G., Jackson, T.L., Balachandar, S., 2016. Force variation within arrays of monodisperse spherical particles. Physical Review Fluids 1, 044202. doi:10.1103/PhysRevFluids.1.044202.

[2] Akiki, G., Jackson, T.L., Balachandar, S., 2017a. Pairwise interaction extended point-particle model for a random array of monodisperse spheres. Journal of Fluid Mechanics 813, 882–928. doi:10.1017/jfm.2016.877.

[3] Akiki, G., Moore, W.C., Balachandar, S., 2017b. Pairwise-interaction extended point-particle model for particle-laden flows. Journal of Computational Physics 351, 329–357. doi:10.1016/j.jcp.2017.07.056.

[4] Asadzadeh, M.Z., Gänser, H.P., Mücke, M., 2021. Symbolic regression based hybrid semiparametric modelling of processes: An example case of a bending process. Applications in Engineering Science 6, 100049. doi:10.1016/j.apples.2021.100049.

[5] Balachandar, S., 2020. Lagrangian and Eulerian drag models that are consistent between Euler-Lagrange and Euler-Euler (two-fluid) approaches for homogeneous systems. Physical Review Fluids 5. doi:10.1103/PhysRevFluids.5.084302.

[6] Balachandar, S., Moore, W.C., Akiki, G., Liu, K., 2020. Toward particle-resolved accuracy in Euler–Lagrange simulations of multiphase flow using machine learning and pairwise interaction extended point-particle (PIEP) approximation. Theoretical and Computational Fluid Dynamics doi:10.1007/s00162-020-00538-8.

[7] Bandaru, S., Deb, K., 2013. A Dimensionally-Aware Genetic Programming Architecture for Automated Innovization, in: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (Eds.), Evolutionary Multi-Criterion Optimization, Springer, Berlin, Heidelberg. pp. 513–527. doi:10.1007/978-3-642-37140-0_39.

[8] Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., Pascanu, R., 2018. Relational inductive biases, deep learning, and graph networks. doi:10.48550/arXiv.1806.01261.

[9] Battaglia, P.W., Pascanu, R., Lai, M., Rezende, D., Kavukcuoglu, K., 2016. Interaction Networks for Learning about Objects, Relations and Physics. doi:10.48550/arXiv.1612.00222.

[10] Beetstra, R., Van der Hoef, M., Kuipers, J., 2007. Drag force of intermediate Reynolds number flow past mono-and bidisperse arrays of spheres. AIChE journal 53, 489–501. doi:10.1002/aic.

[11] Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P., 2017. Geometric Deep Learning: Going beyond Euclidean data. IEEE Signal Processing Magazine 34, 18–42. doi:10.1109/MSP.2017.2693418.

[12] Brunton, S.L., Kutz, J.N., 2023. Machine Learning for Partial Differential Equations. doi:10.48550/arXiv.2303.17078.

[13] Brunton, S.L., Proctor, J.L., Kutz, J.N., 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences 113, 3932–3937. doi:10.1073/pnas.1517384113.

[14] Burlacu, B., Kronberger, G., Kommenda, M., 2020. Operon C++: an efficient genetic programming framework for symbolic regression, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, ACM, Cancún Mexico. pp. 1562–1570. doi:10.1145/3377929.3398099.

[15] Błądek, I., Krawiec, K., 2019. Solving symbolic regression problems with formal constraints, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, Prague Czech Republic. pp. 977–984. doi:10.1145/3321707.3321743.

[16] Chéron, V., Evrard, F., van Wachem, B., 2023. A hybrid immersed boundary method for dense particle-laden flows. Computers & Fluids , 105892doi:10.1016/j.compfluid.2023.105892.

[17] Cranmer, M., 2023. Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl. doi:10.48550/arXiv.2305.01582.

[18] Cranmer, M., Sanchez Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., Ho, S., 2020. Discovering Symbolic Models from Deep Learning with Inductive Biases, in: Advances in Neural Information Processing Systems, Curran Associates, Inc.. pp. 17429–17442. URL: https://proceedings.neurips.cc/paper/2020/hash/c9f2f917078bd2db12f23c3b413d9cba-Abstract.html.

[19] Cuomo, S., Di Cola, V.S., Giampaolo, F., Rozza, G., Raissi, M., Piccialli, F., 2022. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. Journal of Scientific Computing 92, 88. doi:10.1007/s10915-022-01939-z.

[20] d'Ascoli, S., Becker, S., Mathis, A., Schwaller, P., Kilbertus, N., 2023. ODEFormer: Symbolic Regression of Dynamical Systems with Transformers. doi:10.48550/arXiv.2310.05573.

[21] De Silva, B.M., Higdon, D.M., Brunton, S.L., Kutz, J.N., 2020. Discovery of Physics From Data: Universal Laws and Discrepancies. Frontiers in Artificial Intelligence 3, 25. doi:10.3389/frai.2020.00025.

[22] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6, 182–197. doi:10.1109/4235.996017. conference Name: IEEE Transactions on Evolutionary Computation.

[23] Denner, F., Evrard, F., van Wachem, B., 2020. Conservative finite-volume framework and pressure-based algorithm for flows of incompressible, ideal-gas and real-gas fluids at all speeds. Journal of Computational Physics 409, 109348. doi:10.1016/j.jcp.2020.109348.

[24] Elmestikawy, H., Reuter, J., Evrard, F., Mostaghim, S., van Wachem, B., 2024. Data for the paper: "Deterministic drag modelling for spherical particles in Stokes regime using data-driven approaches". doi:10.5281/zenodo.10611852.

[25] Fey, M., Lenssen, J.E., 2019. Fast Graph Representation Learning with PyTorch Geometric. doi:10.48550/arXiv.1903.02428. arXiv:1903.02428 [cs, stat].

[26] de França, F.O., 2023. Alleviating overfitting in transformation-interaction-rational symbolic regression with multi-objective optimization. Genetic Programming and Evolvable Machines 24, 13. doi:10.1007/s10710-023-09461-3.

[27] Haider, C., De Franca, F., Burlacu, B., Kronberger, G., 2022a. Shape-constrained multi-objective genetic programming for symbolic regression. Applied Soft Computing 132, 109855. doi:10.1016/j.asoc.2022.109855.

[28] Haider, C., de França, F.O., Kronberger, G., Burlacu, B., 2022b. Comparing optimistic and pessimistic constraint evaluation in shape-constrained symbolic regression, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, Boston Massachusetts. pp. 938–945. doi:10.1145/3512290.3528714.

[29] Haider, C., Kronberger, G., 2022. Shape-Constrained Symbolic Regression with NSGA-III, in: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (Eds.), Computer Aided Systems Theory – EUROCAST 2022, Springer Nature Switzerland, Cham. pp. 164–172. doi:10.1007/978-3-031-25312-6_19.

[30] He, L., Tafti, D.K., 2019. A supervised machine learning approach for predicting variable drag forces on spherical particles in suspension. Powder Technology 345, 379–389. doi:10.1016/j.powtec.2019.01.013.

[31] Holt, S., Qian, Z., van der Schaar, M., 2023. Deep Generative Symbolic Regression. doi:10.48550/arXiv.2401.00282.

[32] Kamienny, P.A., Lample, G., Charton, F., 2023. End-to-end Symbolic Regression with Transformers, in: Koyejo, S. (Ed.), 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Curran Associates, Inc.. number 35 in Advances in Neural Information Processing Systems, pp. 10269–10281.

[33] Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L., 2021. Physics-informed machine learning. Nature Reviews Physics 3, 422–440. doi:10.1038/s42254-021-00314-5.

[34] Keijzer, M., Babovic, V., 1999. Dimensionally Aware Genetic Programming, in: GECCO'99: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation. volume 2, pp. 1069–1076.

[35] Krishnapriyan, A.S., Gholami, A., Zhe, S., Kirby, R.M., Mahoney, M.W., 2021. Characterizing possible failure modes in physics-informed neural networks. doi:10.48550/arXiv.2109.01050.

[36] Kronberger, G., De Franca, F.O., Burlacu, B., Haider, C., Kommenda, M., 2022. Shape-Constrained Symbolic Regression—Improving Extrapolation with Prior Knowledge. Evolutionary Computation 30, 75–98. doi:10.1162/evco_a_00294.

[37] Kubalík, J., Derner, E., Babuška, R., 2020. Symbolic regression driven by training data and prior knowledge, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, ACM, Cancún Mexico. pp. 958–966. doi:10.1145/3377930.3390152.

[38] Lemos, P., Jeffrey, N., Cranmer, M., Ho, S., Battaglia, P., 2023. Rediscovering orbital mechanics with machine learning. Machine Learning: Science and Technology 4, 045002. doi:10.1088/2632-2153/acfa63.

[39] Li, D., Zhong, J., 2020. Dimensionally Aware Multi-Objective Genetic Programming for Automatic Crowd Behavior Modeling. ACM Transactions on Modeling and Computer Simulation 30, 19:1–19:24. doi:10.1145/3391407.

[40] Li, Y., Li, W., Yu, L., Wu, M., Liu, J., Li, W., Hao, M., Wei, S., Deng, Y., 2023. MetaSymNet: A Dynamic Symbolic Regression Network Capable of Evolving into Arbitrary Formulations. doi:10.48550/arXiv.2311.07326.

[41] Li, Z., Farimani, A.B., 2022. Graph neural network-accelerated Lagrangian fluid simulation. Computers & Graphics 103, 201–211. doi:10.1016/j.cag.2022.02.004.

[42] Martinek, V., Frotscher, O., Richter, M., Herzog, R., 2023. Introducing Thermodynamics-Informed Symbolic Regression – A Tool for Thermodynamic Equations of State Development. doi:10.48550/arXiv.2309.02805.

[43] Mei, Y., Nguyen, S., Zhang, M., 2017. Constrained Dimensionally Aware Genetic Programming for Evolving Interpretable Dispatching Rules in Dynamic Job Shop Scheduling, in: Shi, Y., Tan, K.C., Zhang, M., Tang, K., Li, X., Zhang, Q., Tan, Y., Middendorf, M., Jin, Y. (Eds.), Simulated Evolution and Learning, Springer International Publishing, Cham. pp. 435–447. doi:10.1007/978-3-319-68759-9_36.

[44] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library doi:10.48550/ARXIV.1912.01703.

[45] Petersen, B.K., Landajuela, M., Mundhenk, T.N., Santiago, C.P., Kim, S.K., Kim, J.T., 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. doi:10.48550/arXiv.1912.04871. arXiv:1912.04871 [cs, stat].

[46] Powers, S., Smith, J., Pinciroli, C., 2022. Extracting Symbolic Models of Collective Behaviors with Graph Neural Networks and Macro-Micro Evolution, in: Dorigo, M., Hamann, H., López-Ibáñez, M., García-Nieto, J., Engelbrecht, A., Pinciroli, C., Strobel, V., Camacho-Villalón, C. (Eds.), Swarm Intelligence, Springer International Publishing, Cham. pp. 142–154. doi:10.1007/978-3-031-20176-9_12.

[47] Raissi, M., Perdikaris, P., Karniadakis, G., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics 378, 686–707. doi:10.1016/j.jcp.2018.10.045.

[48] Raissi, M., Yazdani, A., Karniadakis, G.E., 2020. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science 367, 1026–1030. doi:10.1126/science.aaw4741.

[49] Reuter, J., Cendrollu, M., Evrard, F., Mostaghim, S., van Wachem, B., 2022. Towards Improving Simulations of Flows around Spherical Particles Using Genetic Programming, in: 2022 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. doi:10.1109/CEC55065.2022.9870301.

[50] Reuter, J., Elmestikawy, H., Evrard, F., Mostaghim, S., van Wachem, B., 2023a. Graph Networks as Inductive Bias for Genetic Programming: Symbolic Models for Particle-Laden Flows, Springer Nature Switzerland, Cham. volume 13986. doi:10.1007/978-3-031-29573-7_3.

[51] Reuter, J., Elmestikawy, H., Evrard, F., Mostaghim, S., van Wachem, B., 2023b. Graph Networks as Inductive Bias for Genetic Programming: Symbolic Models for Particle-Laden Flows, in: Pappa, G., Giacobini, M., Vasicek, Z. (Eds.), Genetic Programming, Springer Nature Switzerland, Cham. pp. 36–51. doi:10.1007/978-3-031-29573-7_3.

[52] Reuter, J., Martinek, V., Herzog, R., Mostaghim, S., 2024. Unit-Aware Genetic Programming for the Development of Empirical Equations, in: Affenzeller, M., Winkler, S.M., Kononova, A.V., Trautmann, H., Tušar, T., Machado, P., Bäck, T. (Eds.), Parallel Problem Solving from Nature – PPSN XVIII, Springer Nature Switzerland, Cham. pp. 168–183. doi:10.1007/978-3-031-70055-2_11.

[53] Richardson, J., Zaki, W., 1954. Sedimentation and fluidisation: Part I. Transactions of the Institution of Chemical Engineers 32, 35–53. doi:10.1016/S0263-8762(97)80006-8.

[54] Russeil, E., de França, F.O., Malanchev, K., Burlacu, B., Ishida, E., Leroux, M., Michelin, C., Moinard, G., Gangler, E., 2024. Multiview Symbolic Regression, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, Melbourne VIC Australia. pp. 961–970. doi:10.1145/3638529.3654087.

[55] Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., Battaglia, P.W., 2020. Learning to Simulate Complex Physics with Graph Networks. doi:10.48550/arXiv.2002.09405.

[56] Schmidt, M.D., Lipson, H., 2009. Incorporating expert knowledge in evolutionary search: a study of seeding methods, in: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, ACM, Montreal Québec Canada. pp. 1091–1098. doi:10.1145/1569901.1570048.

[57] Schwab, I., Senn, M., Link, N., 2012. Improving Expert Knowledge in Dynamic Process Monitoring by Symbolic Regression, in: 2012 Sixth International Conference on Genetic and Evolutionary Computing, IEEE, Kitakyushu, Japan. pp. 132–135. doi:10.1109/ICGEC.2012.105.

[58] Seyed-Ahmadi, A., Wachs, A., 2020. Microstructure-informed probability-driven point-particle model for hydrodynamic forces and torques in particle-laden flows. Journal of Fluid Mechanics 900. doi:10.1017/jfm.2020.453.

[59] Seyed-Ahmadi, A., Wachs, A., 2022. Physics-inspired architecture for neural network modeling of forces and torques in particle-laden flows. Computers & Fluids 238, 105379. doi:10.1016/j.compfluid.2022.105379.

[60] Siddani, B., Balachandar, S., 2023. Point-particle drag, lift, and torque closure models using machine learning: Hierarchical approach and interpretability. Physical Review Fluids 8, 014303. doi:10.1103/PhysRevFluids.8.014303.

[61] Siddani, B., Balachandar, S., Zhou, J., Subramaniam, S., 2024. Investigating the influence of particle distribution on force and torque statistics using hierarchical machine learning. AIChE Journal , e18339doi:10.1002/aic.18339.

[62] Tang, Y., Peters, E.A.J.F., Kuipers, J.A.M., Kriebitzsch, S.H.L., van der Hoef, M.A., 2015. A new drag correlation from fully resolved simulations of flow past monodisperse static arrays of spheres. AIChE Journal 61, 688–698. doi:10.1002/aic.14645.

[63] Tenachi, W., Ibata, R., Diakogiannis, F.I., 2023. Deep Symbolic Regression for Physics Guided by Units Constraints: Toward the Automated Discovery of Physical Laws. The Astrophysical Journal 959, 99. doi:10.3847/1538-4357/ad014c.

[64] Tenneti, S., Garg, R., Subramaniam, S., 2011. Drag law for monodisperse gas–solid systems using particle-resolved direct numerical simulation of flow past fixed assemblies of spheres. International Journal of Multiphase Flow 37, 1072–1092. doi:10.1016/j.ijmultiphaseflow.2011.05.010.

[65] Udrescu, S.M., Tan, A., Feng, J., Neto, O., Wu, T., Tegmark, M., 2020. AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity, in: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (Eds.), Advances in Neural Information Processing Systems. Curran Associates, Inc.. volume 33, pp. 4860–4871. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/33a854e247155d590883b93bca53848a-Paper.pdf.

[66] Udrescu, S.M., Tegmark, M., 2020. AI Feynman: A physics-inspired method for symbolic regression. Science Advances 6, eaay2631. doi:10.1126/sciadv.aay2631. publisher: American Association for the Advancement of Science.

[67] van Wachem, B., Chéron, V., Elmestikawy, H., 2023. Microstructure-based prediction of hydrodynamic forces in stationary particle assemblies. doi:10.5281/ZENODO.8282784.

[68] Van Wachem, B., Elmestikawy, H., Chéron, V., 2024. Microstructure-based prediction of hydrodynamic forces in stationary particle assemblies. International Journal of Multiphase Flow 175, 104815. doi:10.1016/j.ijmultiphaseflow.2024.104815.

[69] Versino, D., Tonda, A., Bronkhorst, C.A., 2017. Data driven modeling of plastic deformation. Computer Methods in Applied Mechanics and Engineering 318, 981–1004. doi:10.1016/j.cma.2017.02.016.

[70] Verstyuk, S., Douglas, M.R., 2022. Machine Learning the Gravity Equation for International Trade. SSRN Electronic Journal doi:10.2139/ssrn.4053795.

[71] Watson, J., Song, C., Weeger, O., Gruner, T., Le, A.T., Hansel, K., Hendawy, A., Arenz, O., Trojak, W., Cranmer, M., D'Eramo, C., Bülow, F., Goyal, T., Peters, J., Hoffman, M.W., 2024. Machine Learning with Physics Knowledge for Prediction: A Survey. doi:10.48550/arXiv.2408.09840.

[72] Zille, H., Evrard, F., Reuter, J., Mostaghim, S., Wachem, B., 2021. Assessment of Multi-objective and Coevolutionary Genetic Programming for Predicting the Stokes Flow around a Sphere, in: 14th International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control, p. 190. doi:10.7712/140121.7959.18341.