# Graph Networks as Inductive Bias for Genetic Programming: Symbolic Models for Particle-Laden Flows

Julia Reuter[1][0000−0002−7023−7965], Hani Elmestikawy[2][0000−0003−0083−2360], Fabien Evrard[2][0000−0002−5421−1714], Sanaz Mostaghim[1][0000−0002−9917−5227], and Berend van Wachem[2][0000−0002−5399−4075]

[1] Institute for Intelligent Cooperating Systems
[2] Institute for Mechanical Process Engineering
Otto-von-Guericke-University Magdeburg, Germany
{julia.reuter | hani.elmestikawy | fabien.evrard | sanaz.mostaghim | berend.vanwachem}@ovgu.de

**Abstract.** High-resolution simulations of particle-laden flows are computationally limited to a scale of thousands of particles due to the complex interactions between particles and fluid. Some approaches to increase the number of particles in such simulations require information about the fluid-induced force on a particle, which is a major challenge in this research area. In this paper, we present an approach to develop symbolic models for the fluid-induced force. We use a graph network as inductive bias to model the underlying pairwise particle interactions. The internal parts of the network are then replaced by symbolic models using a genetic programming algorithm. We include prior problem knowledge in our algorithm. The resulting equations show an accuracy in the same order of magnitude as state-of-the-art approaches for different benchmark datasets. They are interpretable and deliver important building blocks. Our approach is a promising alternative to "black-box" models from the literature.

**Keywords:** Genetic Programming · Graph Networks · Fluid Mechanics.

## 1 Introduction

The rapid growth of computational power over the last decades has played an important role in fluid mechanics research, as it enables the direct-numerical-simulation (DNS) of flows of ever-increasing complexity. Within the field of fluid mechanics, the simulation of the so-called *particle-laden flows*, i.e., cases of numerous particles immersed and evolving within a fluid, is particularly challenging. Examples of particle-laden flows can be found in the flow of blood cells in plasma, or in the fluidization of biomass particles in furnaces. Due to the complex particle-particle and fluid-particle interactions, high-resolution simulations of such flows are only limited to micro-scale systems with thousands of particles (e.g., see [23,28]). These cannot meet the requirements of real-world applications

commonly involving billions of particles. To overcome this issue, volume-filtered approaches solve the flow on a lower resolution [2,7]. To close the information gap owing to the lower simulation resolution, they require information about the fluid-induced force acting on an individual particle $\mathsf{F}_{\text{fluid}}$, which depends on the above-mentioned interactions and consequently the number of particles.

Various approaches have been presented in the literature to approximate the value of $\mathsf{F}_{\text{fluid}}$, including empirical models [20,22], pairwise-interaction extended models [1,3] and physics-informed artificial neural networks (ANN) [25]. While empirical approaches only predict the mean force, the other methods show promising results to also predict the variations form the mean, but often lack explainability and predict $\mathsf{F}_{\text{fluid}}$ only with a certain error. The goal of this paper is to develop interpretable symbolic models for $\mathsf{F}_{\text{fluid}}$ which can capture the complex interactions for a large number of particles. Interpretable models for this problem are desirable as they allow for deeper understanding and analysis of the underlying interactions, which remain opaque in approaches up until now.

The identification of interpretable models from experimental and simulation data has recently gained importance to overcome the "black-box" nature of machine-learning algorithms such as ANNs. Genetic programming (GP) is a suitable approach to develop human-interpretable symbolic models from data. Given the problem of predicting $\mathsf{F}_{\text{fluid}}$ in particle-laden flows, symbolic models allow a better understanding of the underlying interactions. However, previous work has shown that the straightforward application of GP algorithms on DNS data, for instance to recover the velocity field of the flow around two particles, cannot cope with the complexity of the problem [19]. The complexity of GP algorithms scales exponentially with the number of input variables and functions, thus some pre-processing of the data and/or combination with other model reduction techniques is required to reduce the dimensionality of the problem [4]. In this paper, we present an approach using inductive bias to identify symbolic models for $\mathsf{F}_{\text{fluid}}$. Our approach comprises the following two steps:

1. **Inductive bias**: We first train a Graph network (GN) to predict $\mathsf{F}_{\text{fluid}}$. This step reduces the problem complexity and makes it tractable for GP.
2. **Symbolic model**: We then employ a GP algorithm to develop symbolic models, which replace the internal ANN blocks of the GN.

Since the underlying pairwise interactions between particles are unknown, we employ two different structures as interaction patterns. Our main contributions are *(i)* the supply of high-resolution DNS data of particle-laden flows at different volume-fractions, *(ii)* an extensible algorithm that combines GN and GP and allows for different underlying structures, and *(iii)* a comprehensive analysis of the resulting equations. Our experiments show promising results for $\mathsf{F}_{\text{fluid}}$. Moreover, the symbolic functions are concise and deliver meaningful building blocks.

## 2 Background and Related Work

Since the problem addressed in this paper is rather complex, we first give an overview about related research in the GP area. We then introduce basic concepts of particle-laden flows as well as related research in a separate subsection.

### 2.1 Genetic Programming in Physics Applications

Symbolic models (i.e., mathematical expressions) are interpretable and help to understand underlying patterns in data. Under certain conditions, they generalize better and have higher extrapolation capabilities compared to ANNs [10]. Despite all the advantages, identifying symbolic models from data is a non-trivial task. With increased interest in symbolic models for physics applications, efforts have been made to develop algorithms for symbolic regression, which include, but are not limited to [5,9,12,17,21,27,30]. Population-based methods making use of evolutionary algorithms have shown to perform well on these tasks (also known as GP for symbolic regression).

Applications from the physics area are not new to the GP community. Twenty years ago already, Keijzer et al. proposed a dimension penalty as additional objective to evolve equations that are conformal with physical laws [13]. Other approaches include grammar-based GP algorithms, which restrict the search space to pre-defined rules (e.g., see [14,18]), and strongly typed GP [29]. However, as pointed out by Cranmer et al. [10], high-dimensional problems are too complex to be directly approached with GP, due to the combinatorial explosion with increasing number of features and functions. Being a common underlying pattern for many physics applications, they proposed a framework for problems which can be modeled as interacting particles. Since GNs can represent this underlying structure, they first train a GN on the available data, thus induce a bias. The internal parts of the GN are then replaced by symbolic models. In this way, the problem complexity for the GP algorithm is reduced.

Two recent publications address interesting problems at the intersection of GP and fluid mechanics: Zille et al. examined the capabilities of GP algorithms to predict known equations for the flow around a single spherical particle [31]. Reuter et al. [19] extended this approach to two particles. Their work indicates that the problem is too complex to be directly approached with GP. The problem addressed in the present paper has a considerably higher complexity due to the many particles involved.

### 2.2 Machine Learning for Particle-Laden Flows

Particle-laden flows can be locally characterized based on the particle Reynolds number, Re, and the particle volume fraction, . Re is a dimensionless quantity characterizing the ratio of inertial effects over viscous effects within the fluid, whereas is the local fraction of volume occupied by the particles in the mixture.

The identification of an accurate model for the fluid forces acting on particles is a non-trivial task, as shown by the rich recent literature on the matter

[1,3,24,25]. Promising approaches in this area assume so-called pairwise interactions between particles [1,25]. It states that in a flow locally governed by Re and , the force $\mathsf{F}_{\text{fluid};i}$ acting on a particle $i$ is approximated with a sum of interactions with neighboring particles depending on their relative locations $\mathsf{r}_j$. In this context, the prediction accuracy increases only up to a number of considered neighboring particles between 20 and 30 [1,25]. Although this assumption introduces a certain error to the models, since only considering first-order interaction between particles, their predictive abilities are competitive with those of other approaches.

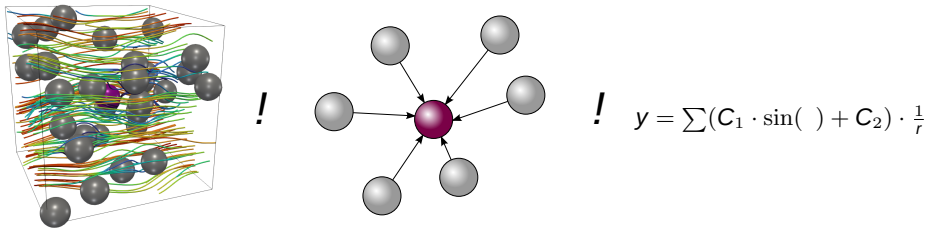
We are mainly interested in the data-driven approaches, which use data from DNS to find an accurate model for $\mathsf{F}_{\text{fluid}}$. Noteworthy publications in this area include [15] and [16], which employ multiple linear regression on expansions of spherical harmonics. Wachs et al. extract distributions of particle locations within a pre-defined neighborhood from DNS data [24]. These are used to find correlations between the force exerted on a particle and the locations of its neighboring particles. Balachandar et al. were the first to implement an artificial neural network (ANN) for force prediction [3]. The input data comprises the relative particle positions within a neighborhood, as well as Re and . The ANN severely overfits the training data. A recent publication from Wachs et al. indicates that physics-inspired neural networks (PINN) can overcome the overfitting problem [25]. A main characteristic of this approach is the parameter sharing between neural network blocks: Making use of the pairwise interaction assumption, the influence of each neighbor of a particle is calculated by a small ANN, which is shared among all neighbors. The total force on a particle is the linear superposition of the influences of its neighboring particles. Next to the neighbor locations, the predictive features include the local average velocity, which can be approximated from the particle locations. The mentioned approaches impose an underlying form on the model, which is deduced from prior knowledge about the problem. For example, the prior assumptions of the PINN model regarding basic interactions between particles are easy to understand. However, the transformations inside the ANN blocks remain opaque.

Building upon the works of Reuter et al. [19], Wachs et al. [25] and Cranmer et al. [10], our goals are: First, to identify which underlying pattern describes the data best by inducing two variants of bias through GN. Second, to overcome the "black-box" nature of GNs by replacing the network blocks with symbolic models. In this paper, we present an algorithm that fits the nature of the problem at hand.

## 3 Proposed Methods

As depicted in Fig. 1, the overall algorithm comprises two phases: In the first phase, a GN is trained on the input data. The inductive bias of the GN determines the internal structure, i.e., which particles interact with others and how the influences of multiple neighboring particles are aggregated. This surrogate model facilitates the development of symbolic models, since the general shape

**Fig. 1.** Symbolic models are generated from simulation data using a GN as surrogate model. Physical particles in the simulation translate to nodes in the GN.




of the equation is determined beforehand. Subsequently, the GP algorithm fits symbolic models to the output of the internal structures of the GN, rather than the actual target variable. The prediction of the target variable is achieved by aggregating the symbolic models, using the same aggregation scheme as previously employed in the GN.

### 3.1 Graph Networks

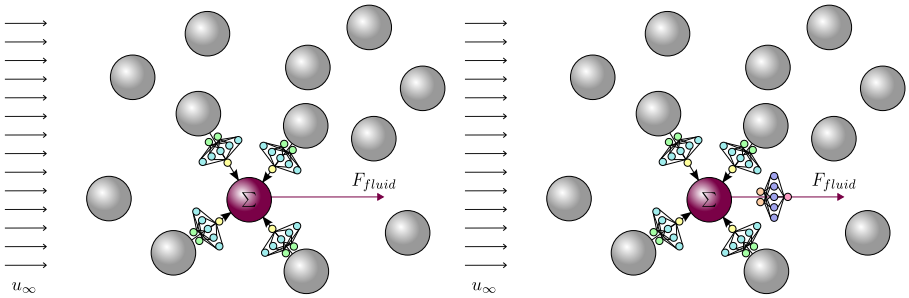
Many systems in physics or real-world applications can be represented by graphs, such as spring systems [10] or particles in a particle-laden flow. This motivates the use of GNs to model interactions between objects or particles. GNs are a subtype of graph neural networks (GNN) [6]. They contain network models for each internal structure of a graph.

A particle translates to a node in the GN, which is described by the **node model** $^{n}$. A system of $q$ particles is represented by a graph with $q$ nodes $n_i$, where $i = 1 :: q$. A node $n_i$ has incoming and/or outgoing edges from/to the nodes in its neighborhood $N_i$. The neighborhood can be defined by a number of closest nodes to $n_i$ or all nodes within a certain distance from $n_i$.

The edges of the graph are represented by the **edge model or message function** $^{e}$. The message function $m_{i;j}$ captures pairwise interactions between two nodes $n_i$ and $n_j$, where $n_j \ 2 \ N_i$. The pairwise interaction is determined by the current state of the interacting particles, so that the input to the edge model comprises the features of the two interacting particles. The node model updates the state of a particle as a function of the current state of a particle $n_i$, as well as the aggregated incoming edge messages. Both $^{e}$ and $^{n}$ use shared parameters for all pairwise interactions and node updates.

The global model $^{g}$ processes all aggregated messages and updated nodes. It computes a global property $g$ for the entire graph. The formal definition of a GN with $q$ nodes is as follows:

**Fig. 2.** GN to predict $\mathbf{F}_{\text{fluid}}$ imposed on the red particle in a particle-laden flow, given four particles in its neighborhood $\mathcal{N}_i$: $y = g(x)$ (left) and $y = f(g(x))$ (right). $u_1$ is the flow velocity.




$$m_{i,j}(t+1) = \phi^e(n_i(t), n_j(t)) \tag{1}$$

$$n_i(t+1) = \phi^n(n_i(t), \sum_{j \in \mathcal{N}_i} m_{i,j}(t)) \tag{2}$$

$$g(t+1) = \phi^g(n_i(t), \ldots, m_{i,j}(t), \ldots), \text{ where } i = 1, \ldots, q,\ j \in \mathcal{N}_i \tag{3}$$

A GN facilitates different ways of predicting a target variable, i.e., different underlying structures. Since the optimal structure of the model to predict a target variable $y$ is unknown, our framework proposes two variants aligned with the structure of the problem at hand:

1. $y = g(x) = \sum_{j \in \mathcal{N}_i} m_{i,j}$: Only the edge model $\phi^e$ is captured. The target variable is the sum of the edge messages received by a node.
2. $y = f(g(x)) = f(\sum_{j \in \mathcal{N}_i} m_{i,j})$: Both the edge model $\phi^e$ and node model $\phi^n$ are captured. The target variable is a function of the summed edge messages. Thus, the summed edge messages are an input to the node model, which predicts the target variable.

Fig. 2 depicts these two variants using the example of a particle-laden flow. It becomes apparent that the internal structure of a GN is separable, which means that we can fit separate symbolic models to the outputs of the edge and node models. This facilitates the equation fitting in the next step tremendously.

### 3.2 Genetic Programming

Genetic Programming is a population-based approach to develop symbolic models from data. The equations are represented as parse trees, which consist of basic operators, functions, features and constants, often referred to as set of primitives. New equations are formed in an evolutionary manner, by applying crossover and mutation on selected equations from the population. With growing

interest in symbolic regression for physics applications, the basic GP algorithm got enhanced with techniques from the area of machine learning. An important property is the possibility to include and fit constants in the equations, usually achieved by a regression algorithm on top of the evolution of equations. Other techniques are batch-wise training to process big datasets in a reasonable amount of time, and the use of sophisticated error functions.

**Algorithm** The proposed GP algorithm makes use of the training features, which can include raw data as well as pre-processed or transformed data to induce prior knowledge about the problem. In addition, the algorithm employs constants, which are fitted through a regression algorithm. Since no ground truth equation is available, the choice of an appropriate function set is a non-trivial task, with major influence on the result. Preliminary experiments and a coarse function tuning have shown that the set of functions and operators $\{+, \ , \sin(\ ), \cos(\ ), \tan(\ ), e^{(\circ)}, \log(\ )\}$ yields satisfactory results. The fitness function to be minimized is the commonly used mean square error (MSE).

Depending on the underlying structures imposed by the GN, the GP algorithms slightly differ:

1. $y = g(x) = \sum_{j \in \mathcal{N}_i} m_{i;j}$: The symbolic model $e^{'}$ replaces the message model $e$, and is thus fitted to the output of the message function, which was recorded during GN training. Constants in the resulting equations are then refitted to the original target variable to avoid the accumulated approximation error. To this end, we employ the Levenberg-Marquardt algorithm and use the constants found by the GP algorithm as starting values.
2. $y = f(g(x)) = f(\sum_{j \in \mathcal{N}_i} m_{i;j}) = n_i$: The first symbolic model $e^{'}$ replaces the message model $e$ and follows the same procedure as in (1). The second symbolic model $n^{'}$ replaces the node model $n$ and predicts the target variable, given the influence of the neighboring particles. Thus, it receives the summed influences $\sum_{j \in \mathcal{N}_i} m'_{i;j}$ as function input. To refit the constants, the inner function $e^{'}$ is plugged into the outer function $n$.

**Techniques for Physically Meaningful Equations** Physical laws often follow relatively simple equations. Thus, our GP algorithm aims at finding equations of low complexity. At the same time, these equations should be in line with physical laws in terms of units. While approaches like grammar-based GP or a dimension penalty as an additional objective are often effective to avoid unit violations, they are complex to implement and sometimes restrict the search space in an undesirable way. Recent research shows, that relatively simple techniques can also yield satisfactory results [9]. To this end, our algorithm employs a complexity measure, complexity-constrained function inputs as well as certain building rules for the parse trees.

*Complexity Measure:* To compute the complexity of an equation, each operation, function, feature and constant is assigned a complexity value. The total

equation complexity is the sum of the complexity values of the used primitives. The complexity values were determined by a coarse hyperparameter tuning.

Binary operators like addition, subtraction and multiplication, as well as the training features are assigned a complexity value of 1. Constants play an important role in numerous physical laws, such as the gravity constant, to name one. When the number of constants in an equation is unknown, they come with the cost of overfitting the training data if too many of them appear in the same expression. Thus, we assign a higher complexity value of 2 to constants. Unary functions such as sin( ), cos( ), tan( ), $e^{(\circ)}$ and log( ) apply a non-linear transformation to the input. Consequently, they are associated a higher complexity value of 2 compared to the basic operators. The unary operation $\frac{1}{\circ}$ is associated with a complexity of 1.

*Complexity-constrained Function Inputs:* Another technique to keep the expressions simple yet effective is to restrict the input of certain operations to a maximum allowed complexity. Our algorithm restricts the input complexity of trigonometric, logarithmic and exponential functions to 8. This means, an expression like $y = \sin(2.0 \quad x \quad 3.0)$ with an input complexity of 7 is allowed. $y = \sin(2.0 \quad x + \log(x) + 3.0)$ with an input complexity of 11 exceeds the limit.

*Building Rules:* Preliminary experiments have shown that GP algorithms sometimes tend to include multiple nested functions in expressions, for instance sin(cos(sin( ))). This behavior is to be avoided, as it can lead to the model overfitting the training data and usually has little meaning in terms of explainability. Consequently, we limit the nesting of trigonometric functions to a maximum of 1, so that sin(sin( )) is allowed, but further nesting with any trigonometric function is prohibited.

## 4 Experiment Design

We investigate the viability of the presented approach using benchmark data from the Stokes flow (i.e., $Re = 0$), with four different particle-volume fractions . For each dataset, separate models are trained for the underlying structures $y = g(x)$ and $y = f(g(x))$.

### 4.1 Data Generation: Simulation of Particle-Laden Flows

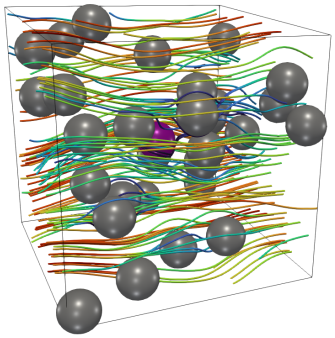
We consider the flow past a stationary array of monodisperse spherical particles in the Stokes regime (Re ! 0), at which the viscous forces dominate. In this Regime, the flow is governed by the Stokes equations as follows

$$\mathbf{u} \quad r\,p = \quad \mathbf{F}_{\text{fluid}}; \tag{4}$$

$$r \quad \mathbf{u} = 0; \tag{5}$$

where is the fluid viscosity, $\mathbf{u}$ is the fluid velocity, $p$ is the hydrostatic pressure and $\mathbf{F}_{\text{fluid}}$ is the Force acting on the fluid. There is no closed-form solution for

**Fig. 3.** Random array of stationary spherical particles at = 0.064



such equations in complex configurations that include more than a single spherical particle. However, a solution can be built from the superposition of fundamental solutions due to the linearity of the governing equations. The method of Regularized Stokeslets [8] is incorporated to construct the solution of the flow around the array of particles. A single regularized Stokeslet solves the flow driven by locally distributed force ($\mathbf{F}_{\text{fluid}} = \mathbf{g}$ ($j\mathbf{x}$ $\mathbf{x}_0 j$)) in free space, where is an isotropic regularization kernel with compact support over the length . Each particle is represented by a group of locally distributed forces to achieve the no-slip at the particle surface.

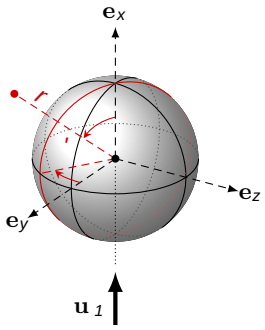
A random array of 30 spherical particles is generated in a unit cube except for one particle which is placed at the center of the cube. Each particle is represented by 300 force markers. The free stream flows in x-direction with uniform velocity $u_\infty = 1$ m/s and the fluid viscosity is = 1 kg/(m s). The force $\mathbf{F}_{\text{fluid}}$ has the three components and is computed on the particle located at the center of the cube for 500 random particle arrangements. The scope of this paper is to predict the *streamwise force component* $F_{\text{fluid}}$. We provide benchmark data for each of the following volume fractions = [0.064, 0.125, 0.216, 0.343]. Visualization of a sample case is shown in Fig. 3. Each training sample encompasses the following features:

- Relative positions $\mathbf{r}_i$ of the 29 neighboring particles
- Average fluid velocity $\mathbf{u}^f$ within the unit cube
- Streamwise force component $F_{\text{fluid}}$ exerted by the fluid on the particle of interest

### 4.2 Data Preprocessing

The raw data generated by the Stokes flow solver from Sec. 4.1 undergoes further transformations before it serves as input to the GN. Initially, the raw particle locations are represented in a three-dimensional Cartesian coordinate system. Preliminary experiments have shown that the GP algorithms perform better

**Fig. 4.** Spherical coordinate system with radius $r$, polar angle , and azimuthal angle $'$.




when locations are available in spherical coordinates. The GN performance remains similar for both configurations. Thus, we convert the particle locations to spherical coordinates $r$, and (see Fig. 4 for exact definition). We assume that it behaves this way because the particle distance $r$ plays an important role in the underlying symbolic model. Furthermore, the trigonometric functions employed in the function set of the GP algorithm are more meaningful with an angle like and as input. This could save the algorithm an intermediate step to compute a dimensionless quantity from the features in Cartesian coordinates.

To increase the number of available data samples, we augment the data by rotations around the axis of the free stream. This has the added benefit of representing symmetries around the free flow direction in our data. In this way, a total of 3,000 samples per training set is available. We split the data with a 3:1 ration into training and test sets. Since the mean force $\langle F_{\text{fluid}} \rangle$ can already be approximated from existing correlations [26], we will predict the deviation from the mean force. This can have the same order of magnitude as $\langle F_{\text{fluid}} \rangle$ itself.

### 4.3 Algorithm Settings

The features of the neighboring particles, i.e., input features to the edge model, are the relative position from the center particle in spherical coordinates $r$, , $'$. The training features of the center particle of interest comprise the local average velocity in $x$, $y$ and $z$ direction, $\bar{u}_x^f$, $\bar{u}_y^f$ and $\bar{u}_z^f$.

The edge and node models of the GN comprise two fully connected hidden layers with 30 neurons each. We use the hyperbolic tangent as nonlinearity. For both $y = g(x)$ and $y = f(g(x))$ as underlying structures, the output of the edge model is recorded during training of the GN to be used as target features of the GP algorithm. The learning rate with an initial value of 0.002 is adjusted during the training process. The model parameters are optimized by the Adam optimizer. We train the model for 5000 epochs to minimize the MSE as loss function. The GN is implemented using *PyTorch Geometric* [11].

Our GP algorithm is implemented in the *PySR* framework [9]. We run the GP algorithm for 200 iterations, with a population size of 100 individuals. The

multi-objective algorithm minimizes the MSE as well as the complexity value of an equation. The best individual from the final Pareto front is identified using a combined measure of accuracy and complexity, as implemented in [9]. The algorithm employs the problem-specific parameters as described in Sec. 3.2, and uses the standard configuration of *PySR* with regards to genetic operators and operator probabilities.

In first trials, we observed that the nested symbolic models $y = f(g(x))$ are more complex than $y = g(x)$, with a tendency to mainly using constants in the outer equations. This can be explained by the two consecutive GP runs for $f$ and $g$. To keep the comparison fair, we want to allow the algorithm for $y = g(x)$ to use more constants, by reducing the constant complexity to 1. Since the structure of an accurate equation is unknown, and fewer constants can be beneficial for generalization, we still run experiments for $y = g(x)$ and a constant complexity of 2. Considering the four benchmark datasets, this makes a total of twelve experiment instances. The training data and code for this paper are publicly available at https://github.com/juliareuter/flowinGN.

## 5 Results and Analysis

Since the algorithm comprises two steps, we applied the following procedure: The GN was trained ten times for each experiment variant. We observed similar accuracies for all runs, which are comparable to those of state-of-the-art-approaches [3,25]. We randomly selected one of the ten models as our basis model. In the next step, we employed the GP algorithm to replace this basis model with symbolic models. For statistical comparison, we perform 31 independent realizations of the GP algorithm per experiment instance. The experiments are analyzed regarding the overall algorithm performance, the explainability of the resulting equations as well as validation on unseen data.

### 5.1 Overall Algorithm Performance

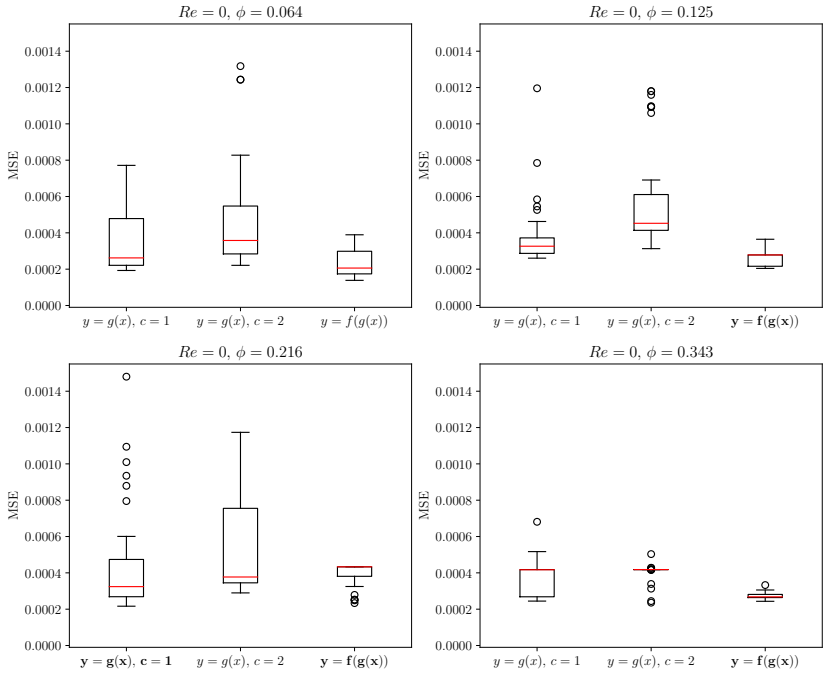
Fig. 5 displays the MSE distributions over 31 realizations for each experiment variant. We used the Holm-Bonferroni test to compare the results for each . The best variants are displayed in bold. For = 0:064, no statistically significant difference between the three variants was identified.

We can observe that all experiment variants for all achieve MSE values of similar magnitude. In general, the nested function $y = f(g(x))$ has a lower spread compared to $y = g(x)$ over 31 runs, i.e., is more reliable to achieve good results. While the medians differ, the best models found by each algorithm have almost the same error value. For most , no significant difference between the constant complexities $c = 2$ and $c = 1$ for $y = g(x)$ is observable.

### 5.2 Explainability of Equations

For a more profound analysis of the resulting equations, we select the best and/or most frequently found symbolic model for each experiment variant. The con-

**Fig. 5.** MSE for different experiment instances over 31 realizations. The variable $c$ indicates the constant complexity for $y = g(x)$. Bold experiments performed best.




stants of these equations are refitted to the original dataset, since they were trained on the outputs of the GN edge model rather than the target variable. Table 1 shows the refitted equations together with their MSE values on the test dataset. For comparison, the MSE of the GN on the same dataset is displayed.

The equations are concise across all experiment instances. It becomes obvious that the algorithm settings successfully prevented function nesting as well as complex input arguments for trigonometric, logarithmic and exponential functions. Almost all equations are physically meaningful without the use of a dimension penalty or grammar-based approach, only through including prior problem knowledge as constraints. Solely $\sin(r)$ and $\exp(\sin(\ ))$ are unusual terms. While the input comprises six features $r$, , ', $\bar{u}_x^f$, $\bar{u}_y^f$ and $\bar{u}_z^f$, mainly $r$ and are used, twice as well $\bar{u}_x^f$ comes into play.

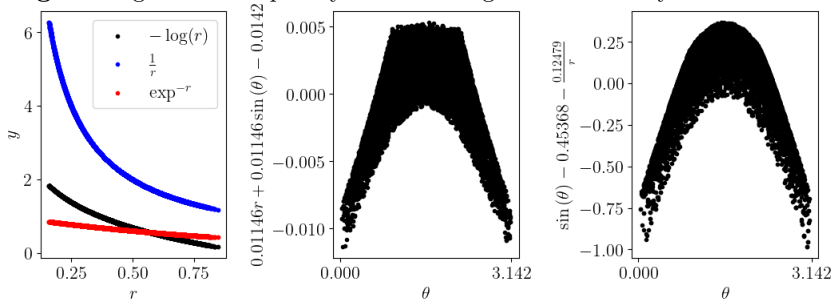
Having a look at the MSE values, the GP equations perform slightly worse than the GN. The errors of the symbolic models are in the same order of magnitude of $10^{-4}$ as the GN, but are sometimes about 1.5 times larger. The underlying structure $y = f(g(x))$ performs better for all benchmark datasets. The best equations with $y = g(x)$ as underlying structure show better performance for $c = 1$ than $c = 2$ across all benchmark instances.

For the underlying structure $y = g(x)$, we examined two complexity values for constants, of $c = 2$ and $c = 1$. The constant complexity $c = 1$ resulted in one

**Table 1.** Symbolic models with constants refitted to the original dataset.

| | Experiment | Equation | GP MSE | GN MSE |
|---|---|---|---|---|
| 0.064 | $y = g(x),\ c = 2$ | $\sum\left(0{:}01146r + 0{:}01146\sin(\ ) \quad 0{:}0142\right)\frac{1}{r}$ | 0.000209 | 0.000120 |
| | $y = g(x),\ c = 1$ | $\sum\left((0{:}03448r + 0{:}03448\sin(\ ) \quad 0{:}04238)(\ \log(r))\right)$ | 0.000188 | 0.000120 |
| | $y = f(g(x))$ | $0{:}0992\sum\left((r\ (\sin(\ ) \quad 0{:}1312) \quad 0{:}1983)(\ \log(r))\right) + v_x^f \quad 0{:}3177$ | 0.000157 | 0.000106 |
| 0.125 | $y = g(x),\ c = 2$ | $\sum\left(0{:}01397\sin(r) + 0{:}01397\sin(\ ) \quad 0{:}01724\right)\frac{1}{r}$ | 0.000284 | 0.000173 |
| | $y = g(x),\ c = 1$ | $\sum\left(0{:}00839 + (0{:}01578\sin(\ ) \quad 0{:}01644)\ \frac{1}{r}\right)$ | 0.000260 | 0.000173 |
| | $y = f(g(x))$ | $0{:}0597\sum\left(\left(\sin(\ ) \quad 0{:}45368 \quad \frac{0{:}12479}{r}\right)e^{\ r}\right) \quad 0{:}0616$ | 0.000209 | 0.000146 |
| 0.216 | $y = g(x),\ c = 2$ | $\sum v_x^f\left(\sin(\ ) \quad 0{:}57328 \quad \frac{0{:}10557}{r}\right)$ | 0.000316 | 0.000206 |
| | $y = g(x),\ c = 1$ | $\sum\left(0{:}00944 + (0{:}01932\sin(\ ) \quad 0{:}01982)\ \frac{1}{r}\right)$ | 0.000247 | 0.000206 |
| | $y = f(g(x))$ | $0{:}1166\sum\left(\left(0{:}17448\sin(\ ) \quad 0{:}08318 \quad \frac{0{:}01419}{r}\right)\frac{1}{r}\right) \quad 0{:}1602$ | 0.000248 | 0.000167 |
| 0.343 | $y = g(x),\ c = 2$ | $\sum\left((0{:}08249\sin(\ ) \quad 0{:}07348)(\ \log(r)) + 0{:}00539\right)$ | 0.000239 | 0.000191 |
| | $y = g(x),\ c = 1$ | $\sum\left((0{:}08749\sin(\ ) \quad 0{:}07348)(\ \log(r)) + 0{:}00423\right)$ | 0.000239 | 0.000191 |
| | $y = f(g(x))$ | $0{:}3904\sum\left(\left(0{:}10982e^{\sin(\ )} \quad 0{:}26635\right)(\ \log(r)) + 0{:}0165\right) \quad 0{:}0421$ | 0.000219 | 0.000197 |

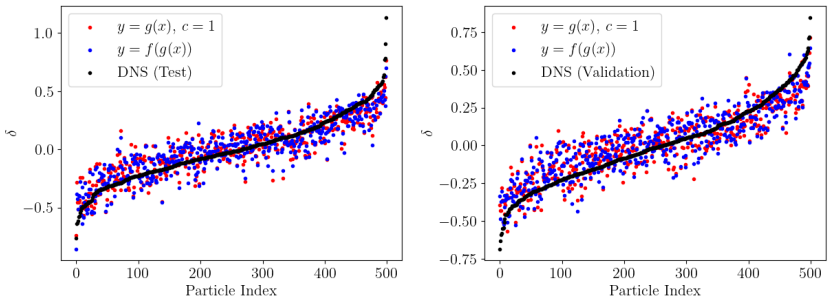
**Fig. 6.** Insights into frequently used building blocks in the symbolic models.




additional constant for $= 0{:}125$ and $= 0{:}216$. The other instances employ the same number of constants for both complexity values. Comparing the number of constants of the two underlying structures, $y = f(g(x))$ always contains one or two more constants than $y = g(x)$.

Similar patterns across equations of all experiment instances can be identified. Each equation contains a building block that accounts for the distance of a neighboring particle: The terms $\frac{1}{r}$, $\log(r)$ and $\exp(r)$ scale the influence of a neighboring particle on the particle of interest, i.e., decrease with increasing radius $r$. Furthermore, each equation contains a larger building block which includes $\sin(\ )$ and constants or other small terms. We can assume that this building block determines the influence of a particle, which is then scaled with the distance to the center particle. Fig. 6 displays the function values of some of the identified building blocks.

**Fig. 7.** Normalized predictions of the deviation from the mean force $\langle F_{fluid} \rangle$, i.e., $= \frac{F_{fluid} \quad \langle F_{fluid} \rangle}{\langle F_{fluid} \rangle}$. Particles are sorted in ascending order by their target value.




### 5.3 Validation of Symbolic Models

Due to the complex underlying relations, overfitting to training data is a common issue in machine learning for fluid mechanics. Thus, we validate the equations found by the GP algorithm on a dataset with the same values of $Re$ and , but from a different DNS realization. Fig. 7 exemplarily depicts the normalized predictions of the deviation from the mean force $\langle F_{\text{fluid}} \rangle$ for $= 0{:}216$. The left plot shows the predictions for 500 particles from the same realization as the training data, and the right plot from a different realization. The plot as well as the MSE values of 0.000247 (left) and 0.000225 (right) for $y = g(x)$ and 0.000248 (left) and 0.000226 (right) for $y = f(g(x))$ indicate that the equations identified actual underlying patterns and did not overfit the training data. The other benchmarks behave similarly, but are omitted here because of space reasons.

## 6 Conclusion and Future Work

We presented an approach to develop symbolic models for the fluid force acting on particles in particle-laden flows from simulation data. A GN serves as surrogate model, from which the symbolic models are deduced using the introduced GP algorithm. We include prior knowledge about the problem by employing a complexity measure as well as imposing constraints on the equation generation process. Furthermore, we preprocessed the data to make them manageable for the GP algorithm. Since the shape of the final model is unknown, we examined two underlying structures $y = g(x)$ and $y = f(g(x))$.

Compared to state-of-the-art approaches, the presented GN achieved similar accuracies [3,25]. The symbolic models consistently perform slightly worse than GN, although errors of both approaches are of the same order of magnitude. A validation on unseen data indicated that our models do not overfit. The underlying structure $y = f(g(x))$ performed best on the provided benchmark instances. We identified building blocks which frequently appear in equations across all

benchmark instances. The equations also reveal which features are most influential on the target variable. Altogether, our approach offers a promising, human-interpretable alternative to the hidden transformation in ANN blocks. Building upon the work of Reuter et al., we scaled up from two to thirty particles.

This study confirms the applicability of our approach to the problem at hand. In a next step, we will examine the performance of our approach on the prediction of the other two force components. While most of the equations evolved are physically meaningful, small terms such as $\sin(r)$ violate the unit system. Converting all features to non-dimensional quantities, such as dividing the distance $r$ by the particle radius, can circumvent this issue.

## References


1. Akiki, G., Moore, W., Balachandar, S.: Pairwise-interaction extended point-particle model for particle-laden flows. Journal of Computational Physics **351**, 329–357 (2017)
2. Anderson, T.B., Jackson, R.O.Y.: A fluid mechanical description of fluidized beds. I and EC Fundamentals **6**(4), 524–539 (1967)
3. Balachandar, S., Moore, W.C., Akiki, G., Liu, K.: 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 **34**(4), 401–428 (2020)
4. Beetham, S., Capecelatro, J.: Multiphase turbulence modeling using sparse regression and gene expression programming (2021), https://arxiv.org/abs/2106.10397
5. Biggio, L., Bendinelli, T., Neitz, A., Lucchi, A., Parascandolo, G.: Neural symbolic regression that scales. In: International Conference on Machine Learning. pp. 936–945 (2021)
6. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine **34**(4), 18–42 (2017)
7. Capecelatro, J., Desjardins, O.: An Euler–Lagrange strategy for simulating particle-laden flows. Journal of Computational Physics **238**, 1–31 (Apr 2013)
8. Cortez, R.: The Method of Regularized Stokeslets. SIAM Journal on Scientific Computing **23**(4), 1204–1225 (Jan 2001)
9. Cranmer, M.: Pysr: Fast & parallelized symbolic regression in python/julia (Sep 2020). https://doi.org/10.5281/zenodo.4041459
10. Cranmer, M., Sanchez-Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., Ho, S.: Discovering symbolic models from deep learning with inductive biases. NeurIPS 2020 (2020)
11. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428 (2019)
12. Kaptanoglu, A.A., de Silva, B.M., Fasel, U., Kaheman, K., Goldschmidt, A.J., Callaham, J., Delahunt, C.B., Nicolaou, Z.G., Champion, K., Loiseau, J.C., Kutz, J.N., Brunton, S.L.: Pysindy: A comprehensive python package for robust sparse system identification. Journal of Open Source Software **7**(69), 3994 (2022)
13. Keijzer, M., Babovic, V.: Dimensionally aware genetic programming. In: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation. vol. 2, pp. 1069–1076 (1999)

14. Mckay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'neill, M.: Grammar-based genetic programming: a survey. Genetic Programming and Evolvable Machines **11**(3-4), 365–396 (2010)
15. Moore, W.C., Balachandar, S.: Lagrangian investigation of pseudo-turbulence in multiphase flow using superposable wakes. Phys. Rev. Fluids **4**, 114301 (2019)
16. Moore, W., Balachandar, S., Akiki, G.: A hybrid point-particle force model that combines physical and data-driven approaches. Journal of Computational Physics **385**, 187–208 (2019)
17. Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., Ramadhan, A., Edelman, A.: Universal differential equations for scientific machine learning (2020). https://doi.org/10.48550/arXiv.2001.04385v4
18. Ratle, A., Sebag, M.: Grammar-guided genetic programming and dimensional consistency: application to non-parametric identification in mechanics. Applied Soft Computing **1**(1), 105 – 118 (2001)
19. Reuter, J., Cendrollu, M., Evrard, F., Mostaghim, S., van Wachem, B.: Towards improving simulations of flows around spherical particles using genetic programming. In: 2022 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8 (2022)
20. Richardson, J.F., Zaki, W.N.: The sedimentation of a suspension of uniform spheres under conditions of viscous flow. Chemical Engineering Science **3**(2), 65–73 (1954)
21. Ross, A.S., Li, Z., Perezhogin, P., Fernandez-Granda, C., Zanna, L.: Benchmarking of machine learning ocean subgrid parameterizations in an idealized model. Earth and Space Science Open Archive p. 43 (2022)
22. Schiller, L., Naumann, A.: über die grundlegenden Berechnungen bei der Schwerkraftaufbereitung. Zeitschrift des Vereines Deutscher Ingenieure **77**, 318–320 (1933)
23. Schneiders, L., Meinke, M., Schröder, W.: Direct particle–fluid simulation of Kolmogorov-length-scale size particles in decaying isotropic turbulence. Journal of Fluid Mechanics **819**, 188–227 (May 2017)
24. Seyed-Ahmadi, A., Wachs, A.: Microstructure-informed probability-driven point-particle model for hydrodynamic forces and torques in particle-laden flows. Journal of Fluid Mechanics **900**, A21 (2020)
25. Seyed-Ahmadi, A., Wachs, A.: Physics-inspired architecture for neural network modeling of forces and torques in particle-laden flows. Computers & Fluids **238**, 105379 (2022)
26. Tenneti, S., Garg, R., Subramaniam, S.: 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**(9), 1072–1092 (2011)
27. Udrescu, S.M., Tegmark, M.: Ai feynman: A physics-inspired method for symbolic regression. Science Advances **6**(16), eaay2631 (2020)
28. Uhlmann, M., Chouippe, A.: Clustering and preferential concentration of finite-size particles in forced homogeneous-isotropic turbulence. Journal of Fluid Mechanics **812**, 991–1023 (Feb 2017)
29. Wappler, S., Wegener, J.: Evolutionary unit testing of object-oriented software using strongly-typed genetic programming. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation. p. 1925–1932 (2006)
30. Werner, M., Junginger, A., Hennig, P., Martius, G.: Informed equation learning. arXiv preprint arXiv:2105.06331 (2021)
31. Zille, H., Evrard, F., Reuter, J., Mostaghim, S., van Wachem, B.: 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 (2021)