Towards Improving Simulations of Flows around Spherical Particles Using Genetic Programming

Julia Reuter*, Manoj Cendrollu†, Fabien Evrard†, Sanaz Mostaghim*, Berend van Wachem†
Otto-von-Guericke-University Magdeburg, Germany
* *Faculty of Computer Science*
† *Faculty of Process Engineering*
{julia.reuter, manoj.cendrollu, fabien.evrard, sanaz.mostaghim, berend.vanwachem}@ovgu.de

***Abstract*—The simulation of particle-laden flows is a crucial task in fluid dynamics, requiring high computational cost owing to the complex interactions between numerous particles. Typically, the flow velocity is described with the equations proposed by Stokes. While there is an analytical solution for the Stokes flows around a single spherical particle, the Stokes flows around many particles are still unsolved. In this paper, we study Genetic Programming (GP) for symbolic regressions to explore the potentials of multi-objective GP in recovering analytical expressions for two and, in the future, N particles. We propose a new GP approach containing building blocks to scale up the problem and provide a new benchmark with 22 cases for this application. To identify the strengths and limitations of GP, we generate fully resolved training data from simulations. We compare the results of our algorithm to the superimposition method and a multi-layer perceptron as two baseline methods. The results show that GP can find comparable and sometimes better solutions with smaller failure rates than the two baseline methods. In addition, the produced solutions by GP are explainable and certain function patterns inline with physical laws can be identified across the benchmark problems.**

***Index Terms*—Genetic Programming, Stokes Flow, Explainability**

I. Introduction

Particle-laden flows can be encountered in many natural and engineering processes, such as the flow of blood cells in plasma or the fluidization of biomass particles in furnaces. Simulating the complex interactions between the particles and the fluid is a non-trivial task. Computational simulations to predict this flow behavior often utilize numerical approaches to approximate the analytical solution, which are expensive in time and computational power. Advanced machine learning (ML) approaches such as artificial neural networks (ANN) can be applied successfully to such complex problems to reduce the computation time. However, they lack explainability of the found relation between input and output, which is an important feature in engineering problems. Genetic Programming (GP), on the other hand, is a suitable supervised learning method which produces human-readable equations and gives insights into the underlying relations between input and output variables. Albeit explainable, out-of-the-box GP approaches are not sufficient to solve such complex problems; expert knowledge is required to tailor the algorithm to a specific use-case. In this paper, we use GP combined with domain knowledge to solve the Stokes flow around two spheres, which is an important step towards improving the simulations of particle-laden flows.

The fluid flow around a single fixed rigid sphere is governed by a set of non-linear partial differential equations, the Navier-Stokes (NS) equations, and its nature depends upon the Reynolds number, Re, a dimensionless quantity characterizing the ratio of inertial effects over viscous effects within the fluid. Owing to the non-linearity of the NS equations, there is no general analytical solution to this fluid-dynamics problem. However, when $\mathrm{Re} \to 0$ (also referred to as the Stokes limit, or Stokes flow) the NS equations can be approximately linearized, and an analytical solution to the steady-state flow over a single sphere of radius a, subject to the far-field velocity $\mathbf{u}_\infty$, can be derived [1]. Recent research from Zille et al. shows that GP can solve the Stokes flow around a single sphere, i.e. finding the known analytical solution to the problem [2]. The next step towards scaling up to N particles is to study the flow around two spherical particles, which is the goal of this paper. Here, we address the configuration of two inline particles. Since there is no analytical solution for flows around more than one sphere, the problem complexity tremendously increases compared to the flow around a single sphere.

In this paper, we propose a novel multi-objective GP approach to predict the Stokes flow around two spherical particles with the goal to incorporate expert knowledge. We conduct a performance study of the proposed approach on a benchmark dataset, as well as a comparison to two baseline methods from the areas of fluid dynamics and machine learning. In addition, we provide a comprehensive analysis of the resulting GP equations to examine their structure and interpretability. Since, to our best knowledge, this paper is the first to use GP for predicting the Stokes flow around two spheres, there is no benchmark data available. Therefore, we additionally provide a benchmark for the Stokes flow around two inline spheres using a high-resolution simulation environment. In our experiments, we investigate the potentials and limitations of GP to solve the complex two particles Stokes flow problem. The results serve as a starting point for the application of GP to more complex fluid dynamics problems, such as the flow around higher numbers of particles or particles with varying radii.

II. Simulation of Stokes flow

At low Reynolds numbers, the fluid flow is highly dominated by the viscous forces and the inertial forces play a very negligible role. One of the fundamental results of such flows is Stokes solution for the flow past a spherical particle. This analytical Stokes solution was considered in a related paper [2] to generate the training data for GP. In this paper, we aim to further explore the potential of GP to recover the analytical expressions for 2 (and later N) particles. However, there isn't any general Stokes solution to generate the training data for an array of particles. To deal with such flows and to further include the effects of neighboring particles, we would have to look beyond Stokes solution for more sophisticated approaches. The theory of fundamental solutions allows us to model such a collection of particles as external sources of concentrated point forces. The resulting solution called as *Stokeslet* represents the velocity due to these point forces.

Cortez in his work *regularized Stokeslet* [3] considers the forces to be applied over a volume using a radially symmetric smooth function ϕ rather than being concentrated at points. Following this idea, the momentum equation could be rewritten as

$$-\eta\Delta\mathbf{u} + \nabla p = \mathbf{f}\phi(\mathbf{x} - \mathbf{x_0}) \tag{1}$$

where $\mathbf{x_0}$ is the location of point force and η refers to the dynamic viscosity of the fluid. On further simplifications and then introducing the regularized Green's function G_δ which is a solution to $\Delta G_\delta = \phi$, and B_δ the solution to the biharmonic equation $\Delta B_\delta = G_\delta$, we arrive at the *regularized Stokeslet* velocity in three dimensions given by

$$\mathbf{u}(\mathbf{x}) = \frac{1}{\eta}((\mathbf{f}\cdot\nabla)\nabla B_\delta - \mathbf{f}G_\delta) \tag{2}$$

Explicit expressions could be derived for B_δ and G_δ based on the kernel used. Evrard et al. in their work [4] follow a similar approach to derive the regularized Stokeslet using a kernel based on the family of Wendland functions. The detailed derivation of regularized velocity could be found in [4] and the final expression is given by Equation (3).The radius of regularization support δ characterizes the size of the kernel.

The above expression, also referred to as *regularized Stokeslet* solution, is used to compute the disturbances in the velocity field due to a point force. Each of these Stokeslet solutions satisfies the governing equations of the Stokes flow. Owing to the linearity of the Stokes flow, a linear combination of these solutions also satisfies the governing equations. Therefore, the flow around a sphere could be simply approximated by populating its surface with a distribution of regularized Stokeslets. The force associated with each of the Stokeslets is chosen so as to ensure that the velocity of the fluid on the surface of the sphere matches that of the sphere. This would lead to a system of equations given by

$$\mathbf{u}(\mathbf{x_i}) = \sum_{j=1}^{N} M_{ij}(\mathbf{x_1}, ...\mathbf{x_N})\mathbf{f_j} \tag{4}$$

The resulting forces from the above system of equations can be used in Equation (3) to estimate the velocity field at any point in the domain.

As a simple numerical experiment, we compare the velocity field around a fixed isolated sphere from Equation (3) with the analytical Stokes solution. To begin with, we populate the surface of the sphere with N Stokeslets using the concept *generalized spiral set* as proposed in [5]. The velocity at each of these forcing points is chosen so as to enforce a no-slip boundary condition on the surface of the sphere. The radius of regularization kernel support is chosen as

$$\delta = 2\sqrt{\frac{\text{surface area of the sphere}}{N}} \tag{5}$$

The *Rooted Mean Squared Error* (RMSE) of the velocity field differences $(\mathbf{u}-\mathbf{u_{exact}})$ for a sphere of radius 0.25 at different values of N is shown in Figure 1. It can be noticed that the accuracy of regularized Stokeslet velocity field improves with increase in number of Stokeslets.

As the results from the regularized Stokeslets were found to be in good agreement with the analytical solution for the flow around a single sphere, we further extend this approach to compute the flow field around two in-line spheres separated by a distance as illustrated in the Figure 2. This serves as training data for GP to recover analytical expressions for flow around two spheres.

III. Genetic Programming

A. Background

Stemming from the family of Evolutionary Algorithms (EA), GP applies the principles of evolution to a population of potential solutions to a problem. Commonly, GP individuals are represented as trees. Problem-dependent function and terminal sets are required to construct a GP tree. While the terminal set $\mathcal{T}$ contains all training features as well as additional constants, the function set $\mathcal{F}$ consists of functions that can be applied to the terminals. For each function, the number of input arguments needs to be defined. This allows for problem-dependent functions beyond basic operators like addition or division, which makes GP a very flexible machine learning approach.

Initially, a random population of individuals is created. These solutions are refined iteratively applying the evolutionary principles of selection, crossover and mutation. Special mechanisms are necessary to perform crossover and mutation on a GP tree. In the crossover procedure, subtrees or branches are exchanged between individuals from the current population to create new children. For mutation, only smaller changes in an individual are conducted, such as replacing a terminal node by a different feature. The quality of a solution/model is evaluated using a problem-dependent fitness function. For regression problems, the fitness of an individual is usually a distance measure between the target values and the produced output. The best solutions of a population survive to the next generation, where the process is repeated for a certain number of generations until a stopping criterion is reached.

$$\mathbf{u} = \frac{1}{120\pi\eta\delta^8} \begin{cases} \mathbf{f}(-81r^7 + 400r^6\delta - 735r^5\delta^2 + 540r^4\delta^3 - 168r^2\delta^5 + 60\delta^7) \\ +(\mathbf{f}\cdot\mathbf{x})\mathbf{x}(63r^5 - 300r^4\delta + 525r^3\delta^2 - 360r^2\delta^3 + 84\delta^5), & \text{if } 0 \leq r \leq \delta \\ \\ \mathbf{f}(15r^{-1}\delta^8 + r^{-3}\delta^{10}) + (\mathbf{f}\cdot\mathbf{x})\mathbf{x}(15r^{-3}\delta^8 - 3r^{-5}\delta^{10}), & \text{if } r > \delta \end{cases} \quad (3)$$

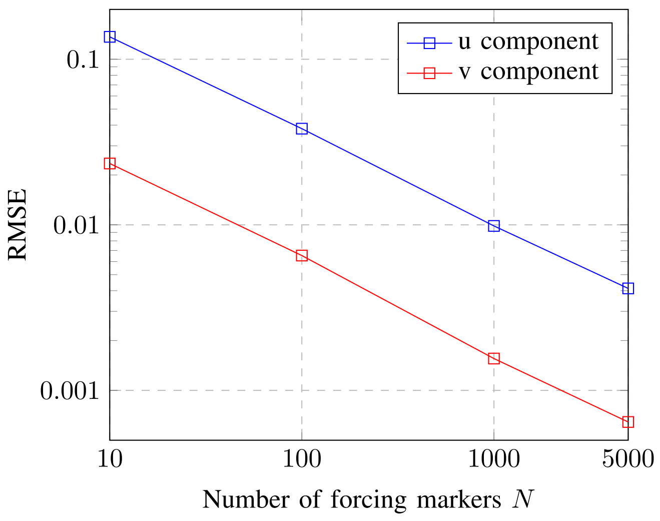

Fig. 1. RMS error of velocity field on the 2D plane ($Z = 0$)

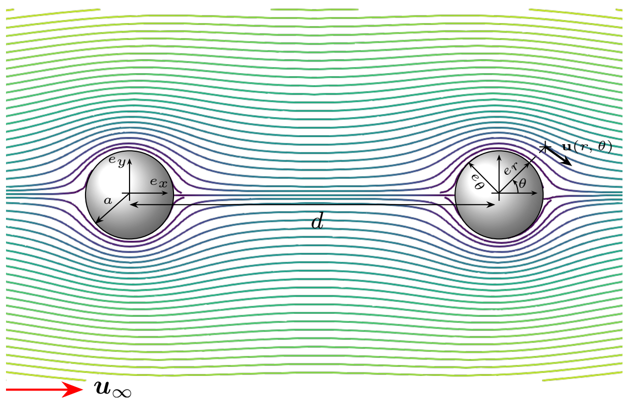

Fig. 2. Streamlines of the Stokes flow around two spheres with radius a, separated by a distance d

The basic GP regression algorithm optimizes for a single objective, i.e. minimizes the distance between the desired and the actual output. As Zille et al. showed, it can be beneficial to include additional objectives to achieve better solutions in the final population [2]. Such multi-objective optimization problems (MOOP) can be formulated as

$$\begin{aligned} \min \quad & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x}))^T \\ s.t. \quad & \mathbf{x} \in \Omega \end{aligned} \quad (6)$$

A MOOP maps the search space Ω to the objective space $\mathcal{M}$ of dimension m. In GP, Ω refers to the set of all possible models that can be created using the provided terminals $\mathcal{T}$ and functions $\mathcal{F}$. The solution to such MOOP is usually a set of Pareto-optimal solutions. The concept of Pareto-dominance is used to attain such solutions. A solution $\mathbf{x}_1$ is said to dominate the solution $\mathbf{x}_2$, if the following conditions are met [6]: (1) The solution $\mathbf{x}_1$ is no worse than $\mathbf{x}_2$ in all objectives, i.e. $f_j(\mathbf{x}_1) \leq f_j(\mathbf{x}_2)$ for all $j = 1 \dots m$. (2) The solution $\mathbf{x}_1$ is strictly better than $\mathbf{x}_2$ in at least one objective, i.e. $f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2)$ for at least one $j = 1 \dots m$. The Pareto-optimal solutions are not dominated by any other solution in Ω.

B. Related Work

The main advantage of GP over other ML methods is the potential to create human-readable equations. Especially in the engineering or physics domain, these equations must be conformal with physical laws to be explainable and add actual value. Different approaches in GP have emerged to guarantee the development of physically meaningful individuals: more restricting approaches such as grammar-based [7] and strongly typed [8] GP limit the programs to only create physically meaningful solutions. Penalizing non-physical operations as proposed in [9] is a softer concept to avoid conflicts with physical laws: For each operation that violates physical laws, a dimension penalty is computed and aggregated throughout an individual. This penalty value is an additional objective to be minimized by the algorithm. In this way, non-meaningful individuals are guided towards being conformal with physical laws. In addition to the dimension penalty, recent work has shown that a rank-based correlation coefficient as a further objective contributes positively to the evolutionary process [2].

To approach a problem as complex as ours with GP, it can be of great use to incorporate expert knowledge in the design of the algorithm. One way is to define tailored building blocks that can be used alongside the standard functions. Finding the optimal building blocks itself is a challenge that has a considerable impact on the success of GP [10] [11].

In the broader scientific field of fluid dynamics, ML was used to predict important model coefficients of flows, e.g. GP models [12], [13], [14] and ANN models [15]. The resolution of these problems is much coarser and not on a particle level as in our work. Research concerning the specific field of the simulation of particle-laden flows uses scaling methods to approximate the impact of neighboring particles [16], [17]. Recent improvements in simulations of particle-laden flows are mainly driven by new assumptions regarding particle-particle interactions [18], [19], [20]. To improve these simulations with ML, Balachandar et al. attempted to derive a high-resolution flow model using an ANN, which was shown to suffer from severe overfitting [19]. To date, there is no GP approach known to the authors to improve simulation of particle-laden flows on a particle level. A basic study on the Stokes flow around a single sphere was conducted by Zille et al. [2].

IV. Proposed Methods

A. Overall Algorithm

Based on the recent research for predicting the Stokes flow around a single sphere [2], our proposed GP algorithm is outlined in Algorithm 1. Preliminary tests have shown that a multi-phase GP advances the evolutionary process in the best case, and does not deteriorate the results in the worst case. Therefore, we enhance the standard GP procedure by a two phase evolution with k generations of crossover and mutation to generate offspring, followed by k generations of mutation only. This helps to refine solutions that have a suitable structure but require small changes on the terminal or function level. We use the $(\mu+\lambda)$ reproduction scheme, which allows parents and children to survive to the next generation.

Algorithm 1 Proposed Genetic Programming Algorithm

```
Input: Training Data $X$, Terminals $\mathcal{T}$, Functions $\mathcal{F}$, phase
    generations $k$, crossover probability $p_c$, mutation proba-
    bility $p_m$
Output: Set of non-dominated solutions $A$
 1: $A \leftarrow$ Empty Pareto-dominance based archive
 2: pop $\leftarrow$ Random initial population of solutions
 3: evaluate(pop)
 4: $A \leftarrow$ updateArchive(pop)
 5: repeat
 6:   for $k$ generations do
 7:     parents $\leftarrow$ select(pop)
 8:     offspring $\leftarrow$ reproduce(parents, $p_c$, $p_m$)
 9:     evaluate(offspring)
10:     pop $\leftarrow$ updatePopulation($\cup$(pop, offspring))
11:     $A \leftarrow$ updateArchive(offspring)
12:   end for
13:   for $k$ generations do
14:     parents $\leftarrow$ select(pop)
15:     offspring $\leftarrow$ reproduce(parents, $p_c = 0$, $p_m = 1$)
16:     evaluate(offspring)
17:     pop $\leftarrow$ updatePopulation($\cup$(pop, offspring))
18:     $A \leftarrow$ updateArchive(offspring)
19:   end for
20: until stopping criterion is reached
21: return $A$
```

B. Terminal and Function Sets

Table I displays the Terminals and Functions provided to the GP algorithm. As terminals, we select the features from the training data as well as a few constants.

TABLE I
Terminal and Function Sets

Terminals	Training Features plus Constants: 1.0, 2.0, 0.5, 0.25
Functions	$+$, $-$, $\times$, $/$ (protected), $u_0(x,y)$, $v_0(x,y)$, $u_d(x,y)$, $v_d(x,y)$

The available functions are basic arithmetic operators, including a protected division to avoid division by zero. In this case, an infinity value is returned. Preliminary experiments showed that a standard GP setting with only arithmetic operators performed poorly on the given problem. The need to incorporate domain knowledge in our approach arose. We comprise the solutions for the flow past a single particle as GP building block, namely the velocities in u- and v-direction: u_0 and v_0 for the disturbance around the left particle, and u_d and v_d for the right particle, with a distance d between them.

Algorithm 2 Computation of flow disturbances u_i, v_i past a single spherical particle at center position $x_{c,i}, y_{c,i}$

```
Input: $x$ and $y$ coordinates
Output: $u_i$ and $v_i$, $i \in \{0,1,2,3,4,5,6,7,8,9,10,20\}$
1: $x_{rel}, y_{rel} \leftarrow$ compute the relative position of $x, y$ from
    particle center $x_{c,i}, y_{c,i}$
2: $r, \theta \leftarrow$ convert $x_{rel}, y_{rel}$ to polar coordinates
3: $U_r, U_\theta \leftarrow$ compute Equations 7 and 8
4: $u, v \leftarrow$ convert $U_r, U_\theta$ to global Cartesian coordinates
5: return $u - u_\infty, v$
```

Given a particle i, u_i and v_i take in the position x and y, at which the flow is to be predicted (Algorithm 2). The computation of the relative position to the center of the particle i is required for the subsequent calculations in polar coordinates. The functions return the disturbance around a single spherical particle using the following equations:

$$U_r(r,\theta,u_\infty,a) = u_\infty * \cos(\theta) * (1 + \frac{a^3}{2r^3} - \frac{3a}{2r}) \tag{7}$$

$$U_\theta(r,\theta,u_\infty,a) = -u_\infty * \sin(\theta) * (1 + \frac{a^3}{4r^3} - \frac{3a}{4r}) \tag{8}$$

The disturbance describes the difference in the undisturbed flow velocity due to the addition of a particle. It is computed in u direction by $u - u_\infty$, whereas the disturbance in v is not modified, since $v_\infty = 0$.

C. Objectives

We follow the example of [2] and incorporate multiple objectives in our approach. Our first objective f_1 mainly determines the quality of a solution. We use the *RMSE*, which penalizes larger errors more and is a common fitness measure in GP:

$$f_1 = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (\hat{y}_i - y_i)^2} \tag{9}$$

Furthermore, we employ a rank-based correlation coefficient using the Spearman correlation as a second objective f_2. The goal is to keep individuals in the evolutionary process that are not yet numerically accurate, but produce a high correlation with the target data. Thus, promising individuals get the chance to be refined towards more accurate solutions. To minimize the objective, we use $f_2 = 1 - |\rho|$, where ρ is the Spearman correlation coefficient ranging from -1 to 1. The absolute value is taken to also keep inversely proportional solutions.

To account for the compliance with physical laws, which is an essential feature of our paper, we employ a third objective f_3 that penalizes individuals that execute non-physical operations. For each unit-violating operation, a penalty of 1.0 is added and aggregated throughout an individual. Additionally, to guarantee that the final unit of an individual corresponds to our target unit `meters/second` = $\mathtt{m}^1 \cdot \mathtt{s}^{-1}$, we add the distances between the exponents of the SI-base units of which the final unit is composed. For example, the distance of $\mathtt{m}^2 \cdot \mathtt{s}^3$ to the target unit is 5.

V. Experimental Design

A. Proposed Benchmark

To identify the potentials and limitations of GP on the two-particle problem, we propose a new benchmark dataset with varying distances between the two particles (see Figure 2). The closer the particles are, the more they influence each other and vice versa. Following this idea, the bigger the distance between the two particles, the better the approximation of the flow around two particles using the analytical solution for one particle. We expect GP to perform well on a large distance $d >= 10$ and want to identify limitations of the approach by a stepwise reduction of the distance. To this end, training data for $d = [20, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$ is generated. Since the flow velocity is described by two components, u and v, separate benchmarks are created. The combinations of eleven distances and two velocity components makes a total of 22 benchmarks.

Each dataset contains the following features:

- u, v: target velocities in u or v direction
- x, y: positions for which the flow is to be predicted
- d: distance between the two spheres
- a: radius of the spheres
- u_∞: undisturbed flow velocity

The datasets are generated using the regularized Stokeslet simulation as described in Section II. We distribute $N = 5000$ Stokeslets over the surface of a sphere following the generalized spiral set approach. The parameters are set to $a = 0.25$ and $u_\infty = 1.0$. The origin of the coordinate frame is located at the center of the left sphere. The right sphere has the center coordinates $(d, 0)$ accordingly. Due to the symmetric nature of the problem, the z-dimension can be omitted, and a dataset is generated from a plane through the centers of both spheres, where $z = 0$. To cover all relevant flow disturbances, the training data comprises $[x_{c,0} - 4a, x_{c,d} + 4a]$ in x-direction and $[y_{c,0} - 4a, y_{c,0} + 4a]$ in y-direction. We employ a resolution of 200 data points in x- and 50 data points in y-direction, which makes a total of 10,000 instances minus the data points that lie inside the spheres, for which no flow can be predicted.

B. Baseline Methods

We use two baseline methods to assess the performance of the proposed approach. First, we employ the superimposition method (SIP), which crudely computes the flow field at any point by summing up the individual disturbances caused by each particle in the domain. Similar to our approach, it uses the analytical solution of the flow around one particle to compute the disturbances caused by M particles:

$$u(x, y) = u_\infty + \sum_{i=1}^{M} u_{\text{disturbance},i}(x, y) \tag{10}$$

$$v(x, y) = \sum_{i=1}^{M} v_{\text{disturbance},i}(x, y) \tag{11}$$

Since the disturbances in u and v direction are also contained in the function set of our proposed GP, we can directly compare the two methods. Due to the deterministic nature of the superimposition method, we execute it only once.

Additionally, we incorporate a multi-layer perceptron (MLP) as baseline method from the ML field. To this end, we use the Scikit-Learn implementation of the MLP regressor. Compared to the standard parameter setting, a larger hidden layer size of 200 and number of iteration of 500 is proposed, to adapt to the considerably high complexity of the problem. We perform 31 runs to achieve stable results of the algorithm.

C. GP Parameter settings

Certain algorithmic parameters need to be defined for GP to yield optimal results (see Table II). We employ a similar parameter setting as in related research for the Stokes flow around one particle [2]. To limit the individual lengths and avoid bloating, a length limit (i.e. number of nodes in GP tree) of 30 is applied. The dataset is split with a ratio of 70% training data and 30% test data. For each benchmark instance, 31 repetitions of the GP algorithm are performed. All algorithms are implemented using the `deap`-framework version 1.3.1 [21] and the `pint` package [1] version 0.16.1.

D. Quality Criteria

During the evolutionary process, the algorithm optimizes three objectives *RMSE* f_1, correlation f_2 and dimension penalty f_3. Additionally, we record the determination coefficient *R*^{2} as well as the *Mean Absolute Error MAE* during the training process. Since the algorithm optimizes for multiple objectives, a single optimal solution cannot be identified anymore. Thus, we use the following procedure to determine the final solution of a run: To guarantee the explainability of the solutions, we only consider solutions that comply with physical laws in our final evaluation, i.e. that have a dimension penalty $f_3 = 0$. From those solutions, the one with the lowest f_1 is the designated output of the run. To verify if the proposed and the baseline methods achieve results that are significantly different, a Friedman's Test is performed. If the equality hypothesis is rejected, the Holm-Bonferroni test for

[1]https://github.com/hgrecco/pint

TABLE II
GP Algorithmic Parameters

Parameters	Settings
μ	2,000
λ	2,000
Number of Evaluations	600,000
Generations per Phase k	20
Selection Mechanism	NSGA-II selection
Initialization Method	Half Full, Half Grow
Crossover Probability p_c	0.7
Mutation Probability p_m	0.3
Crossover	One-point, Leaf-biased ($p_{leaf} = 0.9$) (chosen at random)
Mutation	Uniform, Insert, Shrink, Node Replacement (chosen at random)
Mutation ($p_m = 1.0$)	Shrink (1/3), Node Replacement (2/3)
Max. Tree Length	30
Max. Init Depth	4
Min. Init Depth	1
Max. Mutation Depth	2
Min. Mutation Depth	0
Objectives	f_1, f_2, f_3
Train-Test Ratio	0.7 train, 0.3 test
Runs	31

adapted p-values is conducted for pairwise comparison of the methods. A significance level $\alpha = 0.05$ is used for both tests.

VI. Results and Analysis

The results of the proposed algorithm and the baseline methods on the test data are shown in Table III. The most expressive quality criteria are R^2 and *RMSE*. Because of space reasons, we leave out *MAE* and only present *RMSE*, which is stricter than *MAE* in assessing the quality of a solution. The values of objectives f_2 and f_3 are omitted in the table, as they are mainly used to improve the training and explainability of the solutions. For each measure, the value of the best solution, the mean over the 31 runs and the standard deviation to determine the stability of an algorithm are shown. The baseline methods undergo a statistical comparison (SC) to GP using the Holm-Bonferroni test, where (=) means no significant differences to GP, (-) indicates significantly worse and (+) significantly better results than GP.

A. Convergence Behavior

The convergence behavior of GP for selected distances in u direction is displayed in Figure 3. The behavior in v direction is similar to the ones shown here. Each line represents the mean *RMSE* over the best (lowest *RMSE* and a dimension penalty of $f_3 = 0$) solutions for 31 runs at different times.

The tree plots indicate, that no overfitting to the training data occurred, as training and test error decrease simultaneously and no increase in the test error in later stages of the training can be identified. To the contrary, the gap between training and test error is almost negligible. The convergence speed for $d = 10$ is notably higher than for $d = 1$ and $d = 5$, which indicates that good solutions were easily found. The curve for $d = 10$ resembles almost perfect convergence behavior with a

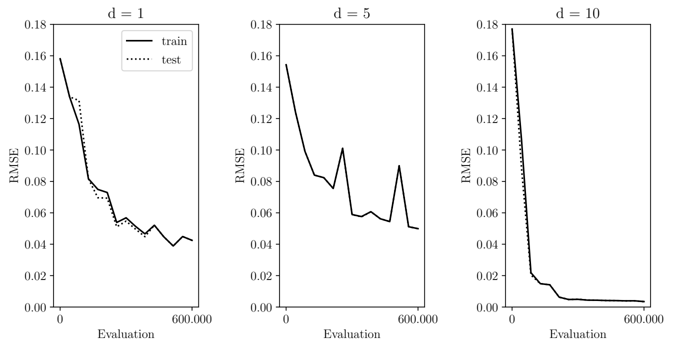

Fig. 3. Convergence behavior of GP on training and test data for selected distances d in u direction

steep decline in the first quarter of the training time and a flat curve in the remainder. To the contrary, the learning curves for $d = 1$ and $d = 5$ do not show a perfect convergence, and it can be expected that *RMSE* can be further reduced with more evaluations. Additionally, two spikes in the error curve for $d = 5$ can be identified. This indicates that the problem for $d = 5$ is especially complex for the algorithm to solve given the provided terminal and function sets.

B. Effects of Different Distances between Particles

For all runs of each benchmark instance, at least one individual with a dimension penalty of $f_3 = 0$ was found. Thus, physically meaningful solutions can be identified for the given problem. The columns *Best* in Table III indicate, that GP found at least one solution for all benchmark instances that is better than SIP and MLP. This observation is especially interesting for the fluid dynamics experts, for whom a single optimal solution over multiple runs is already beneficial.

Figure 4 displays the *RMSE* distributions over the 31 runs for GP and MLP as well as the deterministic result of SIP. As expected, the failure rate and error magnitude decreases with increasing distances between the two particles. In u direction from $d = 20$ to $d = 6$, GP is stable in producing better or equally good solutions than SIP. For $d < 6$, the error spread increases and stable results cannot be guaranteed. This is also reflected in the statistical comparison of *RMSE* values in Table III. Similarly, the mean R^2 values for GP are greater than 0.9 for $d >= 6$, with a drop for $d = [5, 4, 3]$. Interestingly, GP finds comparatively good solutions for small distances $d = 1$ and $d = 2$ compared to SIP. This can be explained by the fact that SIP is a deterministic approach and the same equation is used over all benchmark sets. GP, on the other hand, is trained separately on each benchmark set and can therefore adapt to the changing flow pattern for different distances between the particles. MLP always performs equal or worse than GP.

In v direction, the GP algorithms outperforms SIP for all distances in terms of *RMSE*. While the statistical comparison of R^2 indicates statistically better results to the favor of SIP for some benchmark instances, the real values of R^2 of 0.99 do not reflect large differences between the two methods. Again, the MLP is surpassed by GP in almost all benchmarks. The

TABLE III
RESULTS OF EXPERIMENTS FOR THE u AND v COMPONENTS OF THE FLOW

		u						v					
		R^2			*RMSE*			R^2			*RMSE*		
d	Method	Best	Mean ± Std	SC	Best	Mean ± Std	SC	Best	Mean ± Std	SC	Best	Mean ± Std	SC
1	GP	0.98898	0.93829 ± 0.09497		0.01550	0.04124 ± 0.02642		0.99358	0.92330 ± 0.06684		0.00483	0.01703 ± 0.00921	
	MLP	0.97489	0.94277 ± 0.01702	=	0.03001	0.04483 ± 0.00734	=	0.95736	0.93300 ± 0.01272	=	0.01425	0.01763 ± 0.00162	=
	SIP	0.64865	0.64865 ± 0.00000	-	0.18567	0.18567 ± 0.00000	-	0.92054	0.92054 ± 0.00000	=	0.02633	0.02633 ± 0.00000	-
2	GP	0.99762	0.89169 ± 0.16221		0.00483	0.03873 ± 0.03515		0.99604	0.99324 ± 0.01196		0.00201	0.00297 ± 0.00272	
	MLP	0.93478	0.88892 ± 0.01877	=	0.04202	0.05432 ± 0.00488	-	0.90982	0.85886 ± 0.02657	-	0.01905	0.02339 ± 0.00219	-
	SIP	0.81471	0.81471 ± 0.00000	-	0.09112	0.09112 ± 0.00000	-	0.97546	0.97546 ± 0.00000	-	0.01161	0.01161 ± 0.00000	-
3	GP	0.99099	0.76842 ± 0.26984		0.00264	0.05576 ± 0.04611		0.99704	0.99370 ± 0.00451		0.00101	0.00311 ± 0.00263	
	MLP	0.91038	0.86254 ± 0.03580	=	0.04338	0.05464 ± 0.00705	=	0.87434	0.84426 ± 0.02626	-	0.02170	0.02399 ± 0.00194	-
	SIP	0.90174	0.90174 ± 0.00000	+	0.05440	0.05440 ± 0.00000	=	0.98821	0.98821 ± 0.00000	-	0.00743	0.00743 ± 0.00000	-
4	GP	0.99343	0.64824 ± 0.35060		0.00114	0.06542 ± 0.05365		0.99735	0.99525 ± 0.00208		0.00072	0.00257 ± 0.00183	
	MLP	0.92419	0.85880 ± 0.04212	+	0.03995	0.05398 ± 0.00785	=	0.85032	0.79705 ± 0.03930	-	0.02241	0.02644 ± 0.00257	-
	SIP	0.94880	0.94880 ± 0.00000	+	0.03637	0.03637 ± 0.00000	+	0.99317	0.99317 ± 0.00000	-	0.00531	0.00531 ± 0.00000	-
5	GP	0.99586	0.72624 ± 0.38381		0.00097	0.05048 ± 0.05600		0.99752	0.99668 ± 0.00109		0.00058	0.00160 ± 0.00132	
	MLP	0.90334	0.86075 ± 0.02666	=	0.04342	0.05338 ± 0.00543	=	0.80215	0.73257 ± 0.04348	-	0.02523	0.02912 ± 0.00229	-
	SIP	0.97220	0.97220 ± 0.00000	+	0.02606	0.02606 ± 0.00000	+	0.99559	0.99559 ± 0.00000	-	0.00401	0.00401 ± 0.00000	-
6	GP	0.99886	0.92884 ± 0.16668		0.00072	0.01683 ± 0.03354		0.99843	0.99705 ± 0.00065		0.00049	0.00155 ± 0.00090	
	MLP	0.90542	0.86129 ± 0.02901	=	0.04533	0.05444 ± 0.00584	-	0.83157	0.71321 ± 0.06179	-	0.02215	0.02875 ± 0.00307	-
	SIP	0.98411	0.98411 ± 0.00000	=	0.01964	0.01964 ± 0.00000	=	0.99695	0.99695 ± 0.00000	=	0.00316	0.00316 ± 0.00000	-
7	GP	0.99918	0.97034 ± 0.11585		0.00061	0.01045 ± 0.02073		0.99760	0.99711 ± 0.00033		0.00051	0.00133 ± 0.00081	
	MLP	0.91455	0.85857 ± 0.02956	-	0.04207	0.05575 ± 0.00631	-	0.78306	0.69738 ± 0.06003	-	0.02412	0.02829 ± 0.00269	-
	SIP	0.99036	0.99036 ± 0.00000	=	0.01531	0.01531 ± 0.00000	=	0.99777	0.99777 ± 0.00000	+	0.00256	0.00256 ± 0.00000	-
8	GP	0.99938	0.94054 ± 0.19538		0.00059	0.01269 ± 0.03218		0.99777	0.99736 ± 0.00022		0.00046	0.00128 ± 0.00060	
	MLP	0.90303	0.85571 ± 0.03211	-	0.04576	0.05617 ± 0.00627	-	0.75335	0.65335 ± 0.06812	-	0.02485	0.02896 ± 0.00274	-
	SIP	0.99381	0.99381 ± 0.00000	=	0.01225	0.01225 ± 0.00000	=	0.99831	0.99831 ± 0.00000	+	0.00212	0.00212 ± 0.00000	-
9	GP	0.99749	0.99165 ± 0.00682		0.00059	0.00371 ± 0.00441		0.99829	0.99744 ± 0.00024		0.00045	0.00113 ± 0.00056	
	MLP	0.91789	0.85642 ± 0.03543	-	0.04233	0.05633 ± 0.00731	-	0.75158	0.62227 ± 0.06753	-	0.02370	0.02927 ± 0.00267	-
	SIP	0.99586	0.99586 ± 0.00000	+	0.00998	0.00998 ± 0.00000	-	0.99869	0.99869 ± 0.00000	+	0.00179	0.00179 ± 0.00000	-
10	GP	0.99878	0.99253 ± 0.00686		0.00061	0.00340 ± 0.00415		0.99822	0.99747 ± 0.00016		0.00036	0.00110 ± 0.00043	
	MLP	0.89850	0.84437 ± 0.03641	-	0.04817	0.05875 ± 0.00653	-	0.69999	0.60329 ± 0.07305	-	0.02461	0.02876 ± 0.00257	-
	SIP	0.99714	0.99714 ± 0.00000	+	0.00828	0.00828 ± 0.00000	-	0.99894	0.99894 ± 0.00000	+	0.00154	0.00154 ± 0.00000	-
20	GP	0.99927	0.99865 ± 0.00026		0.00029	0.00085 ± 0.00051		0.99922	0.99921 ± 0.00001		0.00044	0.00054 ± 0.00005	
	MLP	0.88535	0.81167 ± 0.05013	-	0.04589	0.06034 ± 0.00840	-	0.61003	0.37455 ± 0.09140	-	0.02174	0.02743 ± 0.00209	-
	SIP	0.99982	0.99982 ± 0.00000	+	0.00188	0.00188 ± 0.00000	-	0.99974	0.99974 ± 0.00000	+	0.00057	0.00057 ± 0.00000	-

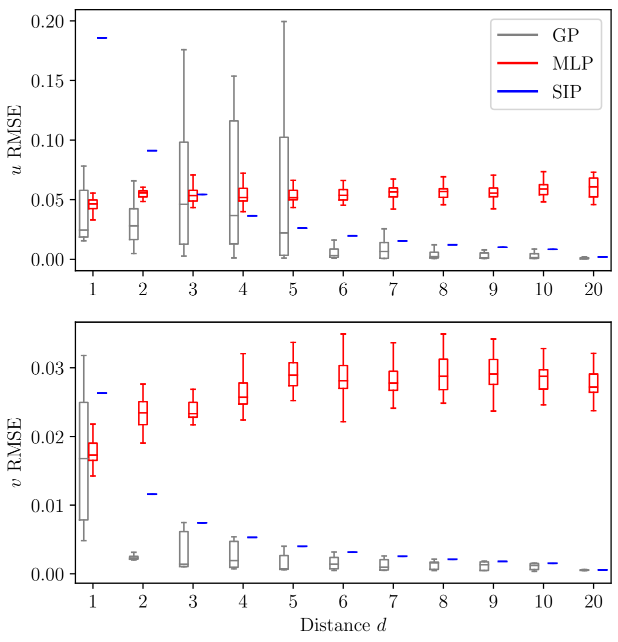

Fig. 4. *RMSE* distribution for GP, MLP and SIP in u and v direction over 31 independent runs for GP and MLP

better performance in v direction can be explained by the fact that the solution required fewer operations than in u direction, i.e. u_∞ is omitted.

The limitation of our algorithm is defined by the flow component with a weaker prediction, since both components are required to predict the flow. Thus, altogether the results suggest that the proposed algorithm delivers *stable* results with low failure rates for the datasets with $d \geq 6$. For $d < 6$, albeit not stable, the algorithm manages to find *at least one* solution within 31 runs that outperforms the SIP method. MLP is utterly outperformed in almost all cases by GP. Since we used a simple MLP implementation, adaptations on model parameters such as the number of hidden layers or layer sizes are necessary to improve the results.

C. Explainability of Individuals

A main concern of this paper is the explainability of the solutions produced by the proposed GP algorithm. Objective f_3 guaranteed the conformity with physical laws. Table IV lists the best GP solutions over 31 runs for the u component of the flow for different distances. The `sympy` package was used to simplify the equations.

The equations utilize the solutions for the disturbance around a single particle, i.e. the expert knowledge given to the algorithm. All individuals follow a similar scheme of adding one or multiple terms to the undisturbed flow velocity u_∞. Essentially, the disturbances around the two particles are aggregated. This pattern is very similar to the superimposition method, which adds the disturbances of all particles involved. However, GP found slight modifications to the superimposition method to generate better results, such as multiplying the solution around a single particle $\mathrm{u_d}$ for $d = 9$ with a factor of 0.97, $d = 7$ (factor 0.958), $d = 4$ (factor 0.923) and $d = 3$ (factor 0.9). Interestingly, instead of using such a fixed factor, other solutions include a term that depends on x and y, which produces infinitesimal values, such as $\frac{\mathrm{u_0}\left(2.25x, 2.0y\right)\mathrm{u_{20}}\left(x, y\right)}{u_\infty}$ in the solution for $d = 20$. Overall, the solutions follow a pattern similar to the superimposition method with slight modifications. The terms involved are physically meaningful and explainable, while the solutions are concise throughout all benchmark instances.

TABLE IV
BEST SOLUTIONS IN U DIRECTION

d	Solution
1	$u_\infty + 2.0\,\mathrm{u}_0\left(\frac{4.0au_\infty}{\mathrm{u}_0\,(x,y)+\mathrm{u}_1\,(x,y)}, 0.25y\right)$
2	$u_\infty + \mathrm{u}_0\,(x,y) - \mathrm{u}_0\left(y, \frac{2.5au_\infty}{0.5\,\mathrm{u}_0\,(x,y)+0.5\,\mathrm{u}_2\,(x,y)}\right) + \mathrm{u}_2\,(x,y)$
3	$u_\infty + 0.9\,\mathrm{u}_0\,(x,y) + 0.9\,\mathrm{u}_3\,(x,y)$
4	$u_\infty + 0.923\,\mathrm{u}_0\,(x,y) + 0.923\,\mathrm{u}_4\,(x,y)$
5	$u_\infty + \frac{u_\infty(\mathrm{u}_0\,(x,y)+\mathrm{u}_5\,(x,y))}{u_\infty-\mathrm{u}_5\,(-2.75a,y)}$
6	$u_\infty + \frac{(u_\infty+\mathrm{u}_6\,(-6.0a,y))(\mathrm{u}_0\,(x,y)+\mathrm{u}_6\,(x,y))}{u_\infty}$
7	$u_\infty + 0.958\,\mathrm{u}_0\,(x,y) + 0.958\,\mathrm{u}_7\,(x,y)$
8	$u_\infty + \frac{\mathrm{u}_0\,(x,y)+\mathrm{u}_8\,(x,y)}{1.0-\frac{0.75\,\mathrm{u}_8\,(a,0.25y)}{u_\infty}}$
9	$u_\infty + 0.970\,\mathrm{u}_0\,(x,y) + 0.970\,\mathrm{u}_9\,(x,y)$
10	$u_\infty + \mathrm{u}_{10}\,(x,y) + \frac{(u_\infty+0.75\,\mathrm{u}_{10}\,(x,y))\,\mathrm{u}_0\,(x,y)}{u_\infty}$
20	$u_\infty + \mathrm{u}_0\,(x,y) + \mathrm{u}_{20}\,(x,y) + \frac{\mathrm{u}_0\,(2.25x,2.0y)\,\mathrm{u}_{20}\,(x,y)}{u_\infty}$

VII. Conclusion and Future Work

This paper presented a first study on the prediction of the Stokes flow around two inline spherical particles using GP for symbolic regression. We implemented a multi-objective approach for the flow around one particle. Preliminary trials showed that GP has difficulties finding a regression function with standard GP parameter settings. To enrich the algorithm with expert knowledge, the solution for the flow around one particle was given to the function set. To identify the strengths and limitations of the GP algorithm, we proposed a new benchmark for the u and v direction of the flow, with decreasing distances between the particles. Our multi-objective approach allowed for solutions aligned with physical laws, which contributes positively to the explainability of the final solution set. All solutions are concise and physically meaningful. The algorithm successfully includes the analytical solution for one particle in the prediction for the flow around two particles. GP found at least one solution that outperforms the SIP baseline method, and achieved significantly better results for distances $d >= 6$. MLP was outperformed by GP throughout all benchmark instances. Many GP solutions multiply the solution for one particle with a factor and aggregate them. Currently, these factors are built from combinations of constants in the terminal set.

This behavior opens space for future research. Multi-gene GP is a reliable approach to aggregate partial GP solutions and identify the ideal factors for multiplication with these solutions. One could consider different linear regression techniques to tackle the problem. Better solutions are expected compared to the current approach, since it only comprises a limited number of constants in the primitive set. This study trained separate GP models to identify the limitations with decreasing distances between two particles. Eventually, a single solution applicable to all distances is desirable. To this end, we will study GP building blocks more extensively, i.e. provide additional functions such as spherical harmonics, which are frequently used in numerical simulations.

References

[1] G. G. Stokes, “On the effect of the internal friction of fluids on the motion of pendulums,” *Transactions of the Cambridge Philosophical Society*, vol. 9, p. 8, 1851.

[2] H. Zille, F. Evrard, J. Reuter, S. Mostaghim, and B. van Wachem, “Assessment of multi-objective and coevolutionary genetic programming for predicting the stokes flow around a sphere,” in *EUROGEN ’21 Proceedings*, 2021.

[3] R. Cortez, “The method of regularized stokeslets,” *SIAM Journal on Scientific Computing*, vol. 23, no. 4, pp. 1204–1225, 2001.

[4] F. Evrard, F. Denner, and B. van Wachem, “Euler-lagrange modelling of dilute particle-laden flows with arbitrary particle-size to mesh-spacing ratio,” *Journal of Computational Physics: X*, vol. 8, p. 100078, 2020.

[5] E. Rakhmanov, E. Saff, and Y. Zhou, “Minimal discrete energy on the sphere0.024614,” *Mathematical Research Letters*, vol. 1, no. 6, pp. 647–662, 1994.

[6] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. USA: Wiley, 2001.

[7] R. I. Mckay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. O’neill, “Grammar-based genetic programming: a survey,” *Genetic Programming and Evolvable Machines*, vol. 11, no. 3-4, pp. 365–396, 2010.

[8] S. Wappler and J. Wegener, “Evolutionary unit testing of object-oriented software using strongly-typed genetic programming,” in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 2006, p. 1925–1932.

[9] D. Li and J. Zhong, “Dimensionally aware multi-objective genetic programming for automatic crowd behavior modeling,” *ACM Transactions on Modeling and Computer Simulation*, vol. 30, no. 3, pp. 1–24, 2020.

[10] T. McConaghy, P. Palmers, M. Steyaert, and G. G. E. Gielen, “Trustworthy genetic programming-based synthesis of analog circuit topologies using hierarchical domain-specific building blocks,” *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 4, pp. 557–570, 2011.

[11] K. Sastry, U.-M. O’Reilly, D. Goldberg, and D. Hill, *Building-Block Supply in Genetic Programming*. Boston, MA: Springer US, 2003, pp. 137–154.

[12] H. Riahi-Madvar, M. Dehghani, A. Seifi, and V. P. Singh, “Pareto optimal multigene genetic programming for prediction of longitudinal dispersion coefficient,” *Water Resources Management*, vol. 33, no. 3, pp. 905–921, 2019.

[13] M.-Y. Liu, W.-x. Huai, Z.-H. Yang, and Y.-h. Zeng, “A genetic programming-based model for drag coefficient of emergent vegetation in open channel flows,” *Advances in Water Resources*, vol. 140, p. 103582, 2020.

[14] A. Danandeh Mehr and E. Kahya, “A pareto-optimal moving average multigene genetic programming model for daily streamflow prediction,” *Journal of Hydrology*, vol. 549, pp. 603–615, 2017.

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

[16] J. Richardson and W. Zaki, “The sedimentation of a suspension of uniform spheres under conditions of viscous flow,” *Chemical Engineering Science*, vol. 3, no. 2, pp. 65–73, 1954.

[17] S. Tenneti, R. Garg, and S. Subramaniam, “Drag law for monodisperse gas-solid systems using particle-resolved direct numerical simulation of flow past fixed assemblies of spheres,” *Int. J. Multiphase Flow*, vol. 37, pp. 1072–1092, 2011.

[18] G. Akiki, T. L. Jackson, and S. Balachandar, “Pairwise interaction extended point-particle model for a random array of monodisperse spheres,” *Journal of Fluid Mechanics*, vol. 813, p. 882–928, 2017.

[19] S. Balachandar, W. C. Moore, G. Akiki, and K. Liu, “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*, vol. 34, no. 4, pp. 401–428, 2020.

[20] G. Akiki, W. Moore, and S. Balachandar, “Pairwise-interaction extended point-particle model for particle-laden flows,” *Journal of Computational Physics*, vol. 351, pp. 329–357, 2017.

[21] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, “DEAP: Evolutionary algorithms made easy,” *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, 2012.