Received: 24 February 2021 | Revised: 27 July 2021 | Accepted: 5 August 2021

DOI: 10.1002/fld.5037

RESEARCH ARTICLE

WILEY

# Using a deep neural network to predict the motion of underresolved triangular rigid bodies in an incompressible flow

**Henry von Wahl** | **Thomas Richter**

Institute for Analysis and Numerics, Otto-von-Guericke-Universität Magdeburg, Magdeburg, Germany

**Correspondence**
Henry von Wahl, Universitätsplatz 2, 39106 Magdeburg, Germany.
Email: henry.vonwahl@ovgu.de

**Abstract**

We consider nonspherical rigid body particles in an incompressible fluid in the regime where the particles are too large to assume that they are simply transported with the fluid without back-coupling and where the particles are also too small to make fully resolved direct numerical simulations feasible. Unfitted finite element methods with ghost-penalty stabilization are well suited to fluid-structure-interaction problems as posed by this setting, due to the flexible and accurate geometry handling and for allowing topology changes in the geometry. In the computationally underresolved setting posed here, accurate computations of the forces by their boundary integral formulation are not viable. Furthermore, analytical laws are not available due to the shape of the particles. However, accurate values of the forces are essential for realistic motion of the particles. To obtain these forces accurately, we train an artificial deep neural network using data from prototypical resolved simulations. This network is then able to predict the force values based on information which can be obtained accurately in an underresolved setting. As a result, we obtain forces on very coarse and underresolved meshes which are on average an order of magnitude more accurate compared with the direct boundary-integral computation from the Navier–Stokes solution, leading to solid motion comparable to that obtained on highly resolved meshes that would substantially increase the simulation costs.

**KEYWORDS**

deep neural network, fluid–structure interaction, level set, Navier–Stokes, rigid particles, unfitted FEM

## 1 | INTRODUCTION

The aim of this work is to simulate multiple small triangular rigid bodies inside an incompressible fluid using a moving domain approach,[1-3] within an unfitted finite element method known as CutFEM.[4] Solids in flows are the subject of various applications in engineering and medicine. The homogenized behavior of these particles is well studied, but

numerical simulations on the fine scale, considering resolved particles, are rare, especially when nonspherical bodies are considered.

If the solid particles within the fluid are very small, they can be considered as point-like within the flow field, and the interaction can be represented in a discrete element framework in terms of analytical laws for the fluid mechanical coefficients, as well as by modeling the effect of the bodies on the surrounding flow. Approximative laws exist for some simple shaped nonspherical particles,[5-9] and simulations with a very large number of particles can be realized in mixed continuum mechanics/discrete element frameworks.

However, if the particles are assumed to be large, only a resolved simulation remains. When considering spherical particles, arbitrary Lagrangian-Eulerian (ALE) approaches can be used. These allow for very good accuracy on relatively coarse grids.[10,11] The ALE method is based on fixed reference domains that allow resolving the boundaries of the particles exactly. The motion of the particles is implicitly tracked by mapping the flow problem onto this reference domain. ALE approaches are highly efficient and accurate, but whenever the particles' motion (or rotation) gets too large, they suffer from a distortion of the underlying finite element mesh and call for remeshing. The ALE approach is highly efficient and very well established in the area of elastic fluid–structure interactions.[12-14] However, if a very large number of particles are considered, or if they are no longer radially symmetric, alternative discretization methods must be considered, as tracking the single particles with a finite element mesh may require a prohibitive computational effort. A prominent example is the immersed boundary method, going back to Peskin,[15] which can efficiently simulate dense particle suspensions.[16] Although the particles must not be exactly resolved, these methods still require an increased mesh resolution in the vicinity of the particles.

Finally, we note that on a theoretical level, the motion of rigid bodies of arbitrary shapes has been studied by Brenner and summarized by Happel and Brenner.[17] However, these considerations only take into account the case of creeping flows governed by the linear Stokes equations rather than the full nonlinear Navier–Stokes equations considered here.

In this work, we define a hybrid finite element/(deep) neural network framework that allows us to describe the interaction with rigid bodies of triangular shape on underresolved meshes accurately. Triangular bodies are a simple example of nonspherical shapes where no rotational symmetry can be assumed that would allow us to use efficient and accurate ALE approaches. Furthermore, since triangles involve reentrant corners, the finite element solution's approximation property suffers due to the lack of regularity of the exact solution, which poses another challenge to the coupled simulation. Both problems also arise for general polygonal objects. By choosing triangles, we can restrict ourselves to geometries that are easy to parameterize in this demonstration. In order to write the transfer of forces between flow and particles with sufficient accuracy, we will use a neural network that represents the forces acting on a particle as a function of the particular flow situation instead of directly evaluating the forces from the Navier–Stokes solution. This network is trained in a presented offline phase based on prototypical resolved simulations. A similar approach with nonspherical but very small particles in the linear Stokes regime has recently been described by Minakowska et al.[18] In principle, this procedure can be transferred to all interface capturing schemes that allow for a robust underresolved representation of structures in a background mesh. Besides different CutFEM or extended finite element approaches[4,19-21] locally adjusted or parametric finite element methods[22-24] are also an option.

Finally, we will illustrate that our neural network approach can efficiently represent the interaction of a fluid with nonspherical particles with high accuracy on coarse grids. To date, there is no efficient alternative approach. ALE methods are too costly due to the constant need for remeshing, and resolved CutFEM approaches are not competitive for their limited approximation properties that would require highly refined meshes at the particles.

## 1.1 | Motivation of approach

The approach is based in the hypothesis that while it may not be possible to evaluate transmission forces with sufficient accuracy on coarse and efficient discretizations, averaged flow features can indeed be obtained accurately on such meshes and that these features are sufficient to predict the transmission forces with the help of a neural network. To illustrate that we can accurately compute volumetric flow features, while the boundary forces are not computed accurately in underresolved CutFEM simulations, we consider the well-established benchmark *flow around a cylinder*,[25] where the laminar flow around a cylinder is considered and where the forces on the cylinder are taken as goal values for a quantitative evaluation of different discretizations, see Figure 1 for a sketch of the solution. We note that the configuration is slightly nonsymmetric, with the cylinder not being vertically centered such that a nonvanishing lift coefficient will result. Here we take the stationary "2D-1" test case and compute this once using a fitted approach together with the Babuška-Miller

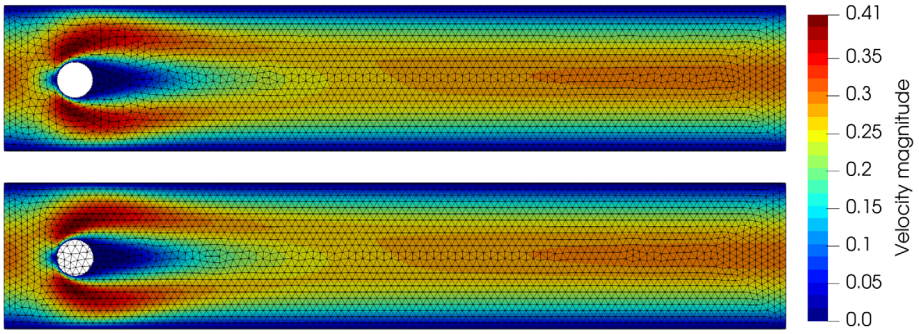


**FIGURE 1** Velocity solution to the ST-2D1 problem with $h_{\max} = 0.02$. Top: fitted FEM and bottom: CutFEM [Colour figure can be viewed at wileyonlinelibrary.com]

**TABLE 1** Benchmark quantities for the ST-2D1 problem computed using fitted FEM and CutFEM on meshes with $h_{\max} = 0.02$

| Method | $c_{\text{drag}}$ | (Err) | $c_{\text{lift}}$ | (Err) | $\Delta p$ | (Err) |
|---|---|---|---|---|---|---|
| Fitted $\texttt{TH}_3$ | 5.579521 | 0.00025% | 0.0105942 | 0.232% | 0.117162 | 0.3046% |
| CutFEM $\texttt{TH}_3$ | 5.571512 | 0.14379% | −0.0063662 | 160.0% | 0.119766 | 1.9111% |
| Ref. | 5.579535 | – | 0.0106189 | – | 0.117520 | – |

*Note*: Reference values obtained from `www.featflow.de` (accessed on February 15, 2021).

trick[26] to evaluate the drag and lift functional, and once using a CutFEM approach[27,28] together with an isoparametric mapping approach[29,30] to obtain the necessary geometry approximation of the discrete level set domain but with the direct evaluation of the boundary integral to realize the force values. For both the fitted and unfitted simulations, we use inf–sup stable Taylor–Hood elements $\mathbb{P}^k/\mathbb{P}^{k-1}$, abbreviated as $\texttt{TH}_k$. For the present computations, we take the order $k = 3$. To make the comparison as fair as possible, we consider unstructured meshes with the same mesh size in each computation.

These and all following numerical finite element computations are implemented using the finite element library `NGSolve/netgen`.[31,32] For CutFEM computations, we additionally use the add-on `ngsxfem`[33] for unfitted finite element functionality within the `NGSolve` framework.

In Figure 1, we can see the velocity solution to these computations together with the computational meshes. Note that it is nearly impossible to distinguish the two solutions visually. In Table 1, we see the benchmark quantities resulting from the two computations. Here we immediately see that while the values from the fitted computations are reasonably for such coarse meshes, the values resulting from the CutFEM computations are poor approximations and even result in the wrong sign for the lift coefficient.

Since the solutions from the fitted and unfitted simulations look indistinguishable in the bulk of the domains, it is natural to ask whether there are other quantities based on the solution near the obstacle, which we can compute accurately in both settings. To this end, we compute the average velocity in a strip of width 0.05 around the obstacle. The results from this are presented in Table 2. Here we see that while the fitted FEM solution resulted in values closer to the reference values compared with CutFEM, the difference between the accuracy of the fitted and unfitted computations are significantly smaller. Furthermore, we see that we keep multiple significant figures of accuracy in the functional value, even at values of order $10^{-5}$.

We conclude that while it is difficult to obtain accurate forces from the boundary integral formulation in an underresolved CutFEM computation, other solution features can be obtained much more accurately, even on such coarse meshes. Therefore, if we can construct a mapping from flow features near the obstacle to the forces acting on the obstacle, we should get more accurate force values in the underresolved CutFEM setting.

The remainder of this text is structured as follows. In Section 2, we describe the basic equations governing the system of a rigid body in an incompressible fluid. In Subsection 2.1, we then discuss the relevant physical parameters we will consider. Section 3 covers the details of the neural network, that is, the design, the generation of the training data, and the training results. Furthermore, we validate the accuracy of the network prediction in comparison to the direct evaluation

**TABLE 2** Average velocity in a strip of width 0.05 around the obstacle

| Method | Avg. $u_1$ | (Err) | Avg. $u_2$ | (Err) |
|---|---|---|---|---|
| Fitted $\mathrm{TH}_3$ | $4.293\,27 \cdot 10^{-3}$ | 0.0013% | $1.492\,00 \cdot 10^{-5}$ | 0.0306% |
| CutFEM $\mathrm{TH}_3$ | $4.287\,96 \cdot 10^{-3}$ | 1.2506% | $1.493\,17 \cdot 10^{-5}$ | 0.1086% |
| Ref. | $4.293\,33 \cdot 10^{-3}$ | – | $1.491\,55 \cdot 10^{-5}$ | – |

*Note*: Reference values computed using a fitted approach with $\mathrm{TH}_6$ elements on a mesh with $h_{\max} = 0.005$. We also indicate the relative error of both approaches compared with the reference solution.

from the Navier–Stokes solution in the setting of the generated training data. In Section 4, we will then apply the resulting network in different scenarios to compare the results against a resolved ALE solution, and show the method's capabilities in settings where a standard ALE discretization is not realizable In Section 5, we give some concluding remarks.

## 2 | GOVERNING EQUATIONS

Let us consider an open-bounded domain $\Omega = \mathcal{F} \dot\cup \mathcal{I} \dot\cup \mathcal{S} \subset \mathbb{R}^d$, $d \in \{2, 3\}$, divided into a $d$-dimensional fluid region $\mathcal{F}$, a $d$-dimensional solid region $\mathcal{S}$, and a $d-1$-dimensional interface region $\mathcal{I}$ coupling the two. The fluid in $\mathcal{F}$ is governed by the incompressible Navier–Stokes equations: Find a velocity and pressure $(\boldsymbol{u}, p)$ such that

$$\begin{aligned} \rho_{\mathcal{F}}\left(\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\right) - \mathrm{div}\boldsymbol{\sigma}(\boldsymbol{u}, p) &= \rho_{\mathcal{F}}\boldsymbol{f}, \\ \mathrm{div}(\boldsymbol{u}) &= 0 \end{aligned} \tag{1}$$

holds in $\mathcal{F}$, with a given fluid density $\rho_{\mathcal{F}}$, an external body force $\boldsymbol{f}$, and the Cauchy stress tensor

$$\boldsymbol{\sigma}(\boldsymbol{u}, p) = \mu_{\mathcal{F}}(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T) - \mathrm{Id}p,$$

where Id is the identity tensor, $\mu_{\mathcal{F}} = \rho_{\mathcal{F}}\nu_f$ is the fluid's dynamic viscosity and $\nu_{\mathcal{F}}$ is the kinematic viscosity. The boundary conditions which complete the system (1) will be given later. We shall consider the homogeneous form of (1), that is, we take $\boldsymbol{f} \equiv 0$.

We further divide the solid region $\mathcal{S} = \dot\bigcup_{i=1}^{N} \mathcal{B}_i$ into a finite set of distinct rigid bodies (particles) $\mathcal{B}_i$. For simplicity, we assume that they each have the same homogeneous density $\rho_{\mathcal{S}}$. Let

$$U_i^{\mathrm{total}} = U_i + \boldsymbol{\omega}_i \times \boldsymbol{r}$$

be the motion of $\mathcal{B}_i$, where $U_i$ is the particle's *velocity*, $\boldsymbol{\omega}_i$ its *angular velocity*, and $\boldsymbol{r} = \boldsymbol{x} - c_{\mathcal{B}}$ is the position vector of a point in $\overline{\mathcal{B}_i}$ relative to the body's centroid $c_{\mathcal{B}}$. These two velocities are then governed by the Newton–Euler equations

$$m_{\mathcal{B}}\partial_t U_i = \boldsymbol{F} + \boldsymbol{F}_{\mathrm{buoyant}} + \boldsymbol{F}_{\mathrm{gravity}}, \tag{2a}$$

$$\boldsymbol{I}_{\mathcal{B}}\partial_t\boldsymbol{\omega} + \boldsymbol{\omega}\times\boldsymbol{I}_{\mathcal{B}}\boldsymbol{\omega} = \boldsymbol{T} + \boldsymbol{T}_{\mathrm{buoyant}} \tag{2b}$$

with the particles mass $m_{\mathcal{B}} = \rho_{\mathcal{S}}\mathrm{vol}(\mathcal{B})$, the force and torque exerted by the fluid on the particle

$$\boldsymbol{F} = \int_{\partial\mathcal{B}_i} \boldsymbol{\sigma}(\boldsymbol{u}, p)\boldsymbol{n}dS \quad \text{and} \quad \boldsymbol{T} = \int_{\partial\mathcal{B}_i} \boldsymbol{r}\times\boldsymbol{\sigma}(\boldsymbol{u}, p)\boldsymbol{n}dS$$

and the particles moment of inertia tensor defined by

$$\boldsymbol{I}_{\mathcal{B}} = \rho_{\mathcal{S}}\int_{\mathcal{B}_i} (\|\boldsymbol{r}\|^2\mathrm{Id}_3 - \boldsymbol{r}\otimes\boldsymbol{r})d\boldsymbol{x}.$$

**TABLE 3** Fluid and solid material properties of a 1:4 water–glycerine mixture and quartz

| Parameter | $\mu_{\mathcal{F}}$ | $\rho_{\mathcal{F}}$ | $\nu_{\mathcal{F}}$ | $\rho_{\mathcal{S}}$ |
|---|---|---|---|---|
| Value | $8.5679 \cdot 10^{-2}\,\mathrm{N\,s\,m^{-2}}$ | $1.2167 \cdot 10^{3}\,\mathrm{kg\,m^{-3}}$ | $7.0419 \cdot 10^{-5}\,\mathrm{m^2\,s^{-1}}$ | $2.65 \cdot 10^{3}\,\mathrm{kg\,m^{-3}}$ |

The buoyancy force and torque are

$$\boldsymbol{F}_{\text{buoyant}} = -m_{\mathcal{F}}\boldsymbol{g} \quad \text{and} \quad \boldsymbol{T}_{\text{buoyant}} = -m_{\mathcal{F}}\boldsymbol{r}_{bo} \times \boldsymbol{g},$$

where $m_{\mathcal{F}} = \rho_{\mathcal{F}}\text{vol}(\mathcal{B}_i)$ is the mass of the displaced fluid and $\boldsymbol{r}_{bo}$ is the vector from the center of mass to the center of buoyancy. The center of buoyancy is defined as the centroid of the displaced fluid volume. Since the particles are completely submersed and both the fluid and solid have a constant density, the center of mass and center of buoyancy coincide, such that $\boldsymbol{r}_{bo} = 0$ and therefore $\boldsymbol{T}_{\text{buoyant}} = 0$. Note that equivalently, the buoyancy effects can be included in the system by including the forces due to gravity on the right-hand side of (1). However, since we do not consider pressure-robust discretizations[34] here, including buoyancy in the solid and ignoring gravity on the fluid is more accurate on the discrete level. Finally, the pull due to gravity is

$$\boldsymbol{F}_{\text{gravity}} = m_{\mathcal{B}}\boldsymbol{g}.$$

To complete the system (1) we need to impose boundary conditions. We shall consider $\Omega$ as a closed aquarium, and therefore take the no-slip boundary conditions

$$\boldsymbol{u} = \boldsymbol{0} \text{ on } \partial\Omega \quad \text{and} \quad \boldsymbol{u} = U_i^{\text{total}} \text{ on } \partial\mathcal{B}_i, \tag{3}$$

the second of which couples the fluid and solid equations. The complete system is then defined as the solution to (1), (2), and (3). The pressure is uniquely defined by taking $p \in \mathcal{L}_0^2(\mathcal{F})$.

*Remark* 1. We note that in two spatial dimensions, the quadratic term $\boldsymbol{\omega} \times \boldsymbol{I}_{\mathcal{B}}\boldsymbol{\omega}$ vanishes in (2b) and the moment of inertia reduces to the scalar $I_{\mathcal{B}} = \rho_{\mathcal{S}} \int_{\mathcal{B}} \|\boldsymbol{r}\|^2 d\boldsymbol{x}$.

## 2.1 | Choice of parameters

We choose our fluid and solid material parameters in order to have a setting that can be given some physical meaning while remaining at small to moderate Reynolds numbers.

The fluid and solid parameters are chosen to approximate coarse sand in a glycerol/water mixture. We take a mixture of one part water to four parts glycerol at a temperature of 21°C. The resulting relevant material parameters are summarized in Table 3. The fluid parameters are obtained through an online calculator tool[1], and the density of the solid is taken as the density of quartz[2]. The ISO standard 14688-1:2017[3] defines coarse sand to have a particle size of 0.63–2.0 mm. For the remainder of this work, we restrict ourselves to the spatial dimension $d = 2$. As an idealized and easily parameterized prototype of a sand particle, we consider an equilateral triangle. Therefore, we take the side length of our triangles as $d_{\mathcal{B}} = 2.0 \cdot 10^{-3}$ m.

With these parameters, we have established the terminal settling velocity of a particle without rotational or horizontal motion to be $v = 0.047\,\mathrm{m\,s^{-1}}$, cf. Subsubsection 3.2.1. Taking the side-length of the triangular particle as the reference length, this leads to a Reynolds number of $Re = \frac{vL}{\nu_{\mathcal{F}}} \approx 1.33$. This is sufficiently large to justify considering the full Navier–Stokes equations, rather than the creeping flow Stokes equations, and small enough to ensure that the local flow around the triangle is not turbulent.

*Remark* 2. We note that the restriction to two spatial dimensions for the fluid and solid, and the consideration of an equilateral triangle as the particle shape, are significant simplifications of the described scenario. This limits the scope in

[1]http://www.met.reading.ac.uk/~sws04cdw/viscosity_calc.html, accessed on September 25, 2020.
[2]http://www.matweb.com/search/datasheet_print.aspx?matguid=8715a9d3d1a149babe853b465c79f73e, accessed on September 25, 2020.
[3]https://www.sis.se/api/document/preview/80000191, accessed on September 25, 2020.

which the approach can be applied to more general cases in the presented form. As we will see below, simple input data and a small neural network are sufficient to predict the transfer forces accurately. As a result, we consider it likely, that more complex input data are necessary for a neural network to capture the forces accurately for three-dimensional flows and more complex particle shapes.

## 3 | NEURAL NETWORK

As we have discussed above, our aim is to train a (deep) neural network that predicts the forces acting on a small triangular rigid body, based on information retrieved from the fluid solution within the bulk of the fluid near the obstacle. In principle, this should be possible, for example, Raissi et al.[35] were able to reproduce the Navier–Stokes solution from images using a physics informed neural network[36] or Minakowska et al.[18] used a very simple deep neural network to predict the forces acting on blood platelets of different shapes in a Stokes flow.

### 3.1 | Network design

As the work in Reference 18 is relatively similar with respect to the aim we have here, we shall take this as our reference point for the design of our network. We note that while in Reference 18 it was the aim to use a neural network to predict the forces based on the speed and shape of very small particles in a linear Stokes flow and without back-coupling of the particles onto the fluid, we want to predict the forces based on the fluid solution near our particle which couples back to the fluid which is governed by the nonlinear Navier–Stokes equations.

#### 3.1.1 | Network input

The first question we have to consider is the choice of flow features to feed into the network. To capture the force acting on the triangular rigid body, it makes sense that the features have to be in some sense local to the rigid body. Point evaluations of the velocity and pressure near the rigid body would be one choice. Unfortunately, while we found that this does indeed capture the necessary information needed by a neural network, the unresolved nature of the CutFEM discretization means that we do not have a chance of obtaining sufficiently accurate values to feed to the network at run-time.

As we have seen in Subsection 1.1, an integral of the velocity components, can be obtained accurately in a coarse CutFEM computation. For a rigid body $\mathcal{B}$, we define $\mathcal{O}(\mathcal{B}) := \{\boldsymbol{x} \in \mathcal{F} | \|\boldsymbol{x} - c_{\mathcal{B}}\|_2^2 < (d_{\mathcal{O}}/2)^2\}$ to be a circular fluid domain around the solid, centered at the solid's center of mass with radius $d_{\mathcal{S}}$. As the network input, we then take the mean fluid velocity, relative to the solids velocity in $\mathcal{O}(\mathcal{B})$, that is,

$$\textbf{rel vel} := \int_{\mathcal{O}(\mathcal{B})} \boldsymbol{u} - U^{\text{total}} d\boldsymbol{x}.$$

We have found $d_{\mathcal{O}} = 4d_{\mathcal{B}}$ to be an appropriate choice. For the input we then dedimensionalize the data and take $\textbf{input} = \textbf{rel vel}/v$, where $v$ is the characteristic velocity, which we take as the terminal settling velocity $v = 0.047\,\text{m}\,\text{s}^{-1}$.

Assuming that the mesh size is not coarser than the size of the triangular rigid body, we can then safely apply the isoparametric mapping technique[29,30] to accurately integrate over this domain in the CutFEM setting without distorting the three level sets describing the sides of the triangular body.

As the forces only depend on the angle of attack between the mean (integrated) flow relative to the triangles velocity and the orientation of the triangle, we shall consider the input in a reference configuration. For this, we choose the bottom side of the triangle to be parallel to the $x$-axis. As a result, the network learns the forces resulting from any angle of attack in this reference orientation. To obtain the physical forces, we rotate the input into the reference configuration and rotate the drag/lift predictions back into the physical orientation. We also refer to Figure 3 for a sketch of this configuration. Since the torque in two spatial dimensions is a scalar quantity, it is invariant with respect to rotation.

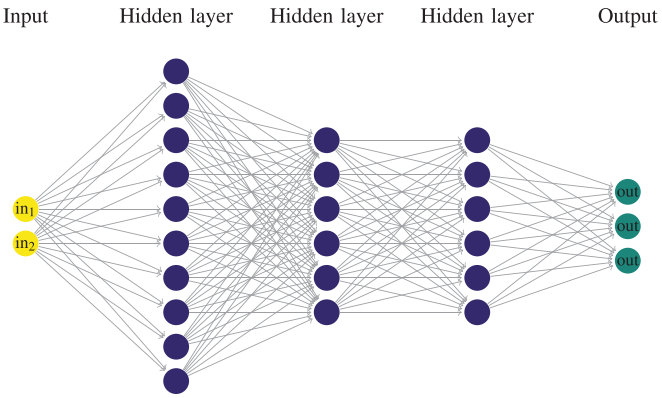


**FIGURE 2** Illustration of a fully connected feed-forward network with two inputs, three hidden layers consisting of 10, 6, and 6 neurones, respectively, and a three outputs [Colour figure can be viewed at wileyonlinelibrary.com]

### 3.1.2 | Network output

To keep the network general, we train the network to learn the dimensionless drag, lift, and torque coefficients

$$C_{\text{drag}} = \frac{2}{v^2 \rho_{\mathcal{F}} L} \boldsymbol{F}_1, \quad C_{\text{lift}} = \frac{2}{v^2 \rho_{\mathcal{F}} L} \boldsymbol{F}_2, \quad \text{and} \quad C_{\text{torque}} = \frac{4}{v^2 \rho_{\mathcal{F}} L^2} \boldsymbol{T}.$$

The reference speed is taken as the terminal velocity established below as $v = 0.047\,\text{m}\,\text{s}^{-1}$ and the reference length $L = d_{\mathcal{S}} = 0.002\,\text{m}$.

### 3.1.3 | Network architecture

We shall consider a fully connected feed-forward network with at least three hidden layers and the ReLU activation function, that is, $f(x) = \max\{0, x\}$. An example of such a network can be seen in Figure 2. We shall refer to networks by the number of neurones. For example, with $l_1/l_2/l_3$, we denote a network with three hidden layers consisting of $l_1$, $l_2$, and $l_3$ neurones, respectively. The number of neurones per layer and the number of layers we need for our network will be determined experimentally by inspecting the results achieved during training. See Subsection 3.3 below.

## 3.2 | Training data

In order to train a network, we need to generate the appropriate input and target data.

### 3.2.1 | A priori computations

Neural networks are generally only accurate if they are applied in scenarios that lie in the data range with which the network was trained. Therefore, we need some information on how fast a triangular rigid body will fall under the acceleration due to gravity as described in Section 2 with the material parameters described in Subsection 2.1.

To get a better sense of the speeds we need to generate the training data, we consider an ALE discretization of a single triangular rigid body falling without rotation in a viscous fluid. To this end we consider the domain $\Omega = (0, 0.1) \times (0, 0.2)$ and a triangular particle of side-length 0.002 with the centroid $(0.05, 0.15)$ at $t = 0$. For the fluid and solid parameters, we take the parameters described in Table 3. We then solve a simplified (restricted to vertical motion) form of the system (1), (2), and (3) until $t = 10$. To solve the coupled fluid–solid problem, we consider a partitioned approach, that is, we solve the fluid and solid problem separately and iterate between the two until the system is solved implicitly, see Section 4 below and Reference 11. To make the ALE mapping simple, we only allow movement in the vertical direction, that is, we ignore

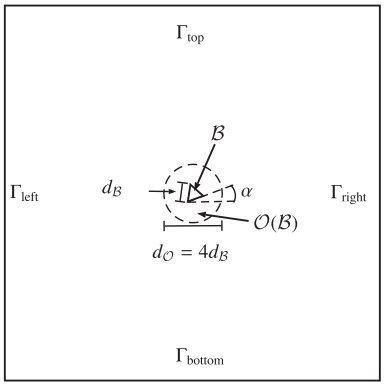


**FIGURE 3** Sketch of the computational domain to generate the learning data set

the effects of horizontal drag and torque. For the construction of such a mapping, we refer to Reference 11. We consider this setup for 10 different angles of attack of the triangle and evaluate the terminal velocity.

For this computation, we use a mesh with $h_{\max} = 1.0 \times 10^{-3}$ in the bulk and $h_{\max} = 2.5 \times 10^{-4}$ on the triangle boundary with $\mathbb{P}_5/\mathbb{P}_3^{dc}$ finite elements, resulting in a FE space with[4] $\texttt{dof} = 2.1 \cdot 10^5$ and $\texttt{dof}_{\texttt{cond}} = 8.4 \cdot 10^4$. The time-derivative and mesh velocity are approximated using the BDF1 scheme, and the time-step used is $\Delta t = 1/500$.

In the resulting computations, we observed that the maximal terminal velocity of the triangle was $0.047\,\text{m}\,\text{s}^{-1}$. This gives us a good indication of the maximal velocity we need to consider to generate the training data.

### 3.2.2 | Generating training data

In order to create a network from which we can expect sufficient accuracy, we generate the training data in a setting that is as close to the application as possible. However, in contrast to the following application of the network, where extremely coarse discretizations are used for efficient simulations, the training data are obtained on highly resolved meshes using prototypical scenarios with a prescribed motion of the particle.

*Setup: Translational motion*

To generate learning data for translational motion, we take the domain $(0, 0.5)^2$ with the equilateral triangular obstacle located at $(0.25, 0.25)$ in the reference configuration. This rigid body is then moved from $(0.1, 0.25)$ to $(0.4, 0.25)$ and back again over a time interval $[0, t_{\text{end}}]$. To get a wide range of relative velocities between the triangle and the mean flow around the triangle, we accelerate the particle at different rates, that is, consider different values for $t_{\text{end}}$. The physical location of the body is then given by

$$\left(0.25 - 0.15\cos\left(\frac{\pi t}{2t_{\text{end}}}\right), 0.25\right)$$

for $t \in [0, t_{\text{end}}]$. To implement this motion, we use a prescribed ALE mapping, as above. In order to generate the data with different angles of attack, we rotate the rigid body by an angle $\alpha$ around its center of mass and rotate the resulting relative velocities and forces back into the reference configuration. As a result, we can reuse the same ALE mapping to simulate different directions of relative motion of the solid body. A sketch of this configuration in the ALE reference configuration can be seen in Figure 3.

*Setup: Rotational motion*

Since the above ALE computations only include translational motion, this is equivalent to only considering a parallel flow around a fixed obstacle. For the network to be universally usable, we therefore also need to include rotational flow

[4] $\texttt{dof}$ denotes the degrees of freedom of the full finite element space. $\texttt{dof}_{\texttt{cond}}$ denotes the number of degrees of freedom remaining after static condensation of unknowns internal to each element, that is, the size of the system to be solved in every time-step.

**TABLE 4** Validation results of the discretization to generate the training data set

| Case (method) | $h_{\max}$ | $h_{\text{loc}}$ | $\Delta t$ | $C_{\text{drag,max}}$ | $C_{\text{lift,min}}$ | $C_{\text{torque,max}}$ |
|---|---|---|---|---|---|---|
| Translational (ALE) | 0.02 | $2 \cdot 10^{-4}$ | 1/250 | $2.6094 \cdot 10^{2}$ | $-5.9548 \cdot 10^{1}$ | $2.9674 \cdot 10^{1}$ |
| | 0.01 | $1 \cdot 10^{-4}$ | 1/500 | $2.6105 \cdot 10^{2}$ | $-5.9673 \cdot 10^{1}$ | $2.9699 \cdot 10^{1}$ |
| | 0.005 | $5 \cdot 10^{-5}$ | 1/1000 | $2.6111 \cdot 10^{2}$ | $-5.9751 \cdot 10^{1}$ | $2.9694 \cdot 10^{1}$ |
| Rotational (CutFEM) | 0.0084 | – | 1/250 | $1.6025 \cdot 10^{-2}$ | $-1.3880 \cdot 10^{-2}$ | $3.3697 \cdot 10^{-1}$ |
| | 0.0042 | – | 1/500 | $1.3443 \cdot 10^{-2}$ | $-1.5876 \cdot 10^{-2}$ | $3.5969 \cdot 10^{-1}$ |
| | 0.0021 | – | 1/1000 | $1.0735 \cdot 10^{-2}$ | $-8.9824 \cdot 10^{-3}$ | $3.5636 \cdot 10^{-1}$ |

data. This corresponds to rotating the triangle. To this end, we consider a triangular rigid body located at the center of the domain $\Omega = (0, 0.1)^2$. The triangular body is then rotated at different speeds, clockwise and anticlockwise, such that the total rotation at time $t$ with respect to the initial configuration is given by $\sin(2\pi \cdot t/t_{\text{end}})$.

In this situation, using ALE is not suitable, as relatively small rotations will lead to mesh-entanglement. Therefore, we use a highly resolved moving domain CutFEM simulation to generate data with rotational input. Further details of this method are given below in Section 4.

*Validation*

In order to ensure that the generated learning data is computed sufficiently accurate, we consider the above cases over a series of meshes and time-steps. For this test case, we take $t_{\text{end}} = 2.0$ for both setups and $\alpha = 0$ for the translational setup.

For the ALE discretization, we take $\text{TH}_4$ elements on a mesh with diameter $h_{\max}$ in the bulk and a local mesh parameter $h_{\text{loc}}$ on the boundary of the rigid body. In time we discretize using the BDF1 scheme with the time-step $\Delta t$.

For the CutFEM discretization, we take $\text{TH}_2$ elements on a mesh with global meshing parameter $h_{\max}$ and performing three levels of mesh bisections in the domain where we compute the average relative velocity with an additional five levels of mesh bisections in the bounding circle of the rotating triangle.

The results for the convergence study can be seen in Table 4. For the ALE computations, we see that the discretization is accurate and that the second mesh already provides 2–3 significant figures of accuracy in the target data. Since the neural network prediction will introduce an additional approximation error, we consider this to be sufficiently accurate for the training data. For the rotational CutFEM data, we see that the forces are (in absolute value) significantly smaller than the translational data. However, we also see that the second finest discretization should be accurate enough for the training of the network.

*Remark* 3. In general, the (fully) implicit BDF1 method is not a good choice to discretize the time-derivative in the Navier–Stokes equations, since the scheme is too diffusive, thus preventing, for example, vortex shedding.[37] However, the choice of BDF1 here is not problematic, since we are considering flows with small Reynolds numbers. Furthermore, we have tested computations with the less diffusive BDF2 scheme, and we did not observe any significant differences in the solution.

*Generation*

To generate the data set, we consider $t_{\text{end}} \in \{2, 2.5, 3, 4, 6, 8, 1\}$ and rotate the triangle with angle $\alpha \in \{\frac{2i\pi}{3 \cdot 40} | i = 0, \dots, 39\}$. Since the triangle is equilateral, the remaining angles of attack $\alpha \in [2\pi/3, 2\pi)$ can be obtained by postprocessing the data appropriately. Based on the above validation computations, we consider the mesh with $h_{\max} = 0.01$ and take the time-step $\Delta t = 1/500$. The resulting data set then contains $2.1 \cdot 10^6$ input/output pairs.

For the rotational data we choose $t_{\text{end}} \in \{0.5, 1, 2, 3, 5, 6\}$. Again, from the above computations, we take the mesh with $h_{\max} = 0.0042$ and the time-step is chosen as $\Delta t = 1/500$. As a result, we then obtain an additional $8.8 \cdot 10^3$ data sets.

## 3.3 | Training

We implement the neural network described in Subsection 3.1 using `PyTorch`.[38] We use the mean squared error as the loss function and take the Adam algorithm as the optimizer with a step size of $10^{-4}$. We trained the network for a total of

**TABLE 5** Prediction errors in a weighted $\ell^2$ norm and the maximum norm on the training and validation data-sets after 20,000 epochs of training for a number of different network sizes

| | | | Training data | | Validation data | |
|---|---|---|---|---|---|---|
| **Architecture** | **Parameters** | **Target** | $\|\cdot\|_{\text{mean}}$ | $\|\cdot\|_\infty$ | $\|\cdot\|_{\text{mean}}$ | $\|\cdot\|_\infty$ |
| 30/20/10 | 953 | $C_{\text{drag}}$ | $1.7 \cdot 10^{0}$ | $1.7 \cdot 10^{1}$ | $1.9 \cdot 10^{0}$ | $1.2 \cdot 10^{1}$ |
| | | $C_{\text{lift}}$ | $1.6 \cdot 10^{0}$ | $1.9 \cdot 10^{1}$ | $1.8 \cdot 10^{0}$ | $1.5 \cdot 10^{1}$ |
| | | $C_{\text{torque}}$ | $5.9 \cdot 10^{-1}$ | $5.0 \cdot 10^{0}$ | $6.6 \cdot 10^{-1}$ | $4.3 \cdot 10^{0}$ |
| 50/20/20 | 1653 | $C_{\text{drag}}$ | $1.0 \cdot 10^{0}$ | $8.6 \cdot 10^{0}$ | $1.2 \cdot 10^{0}$ | $7.9 \cdot 10^{0}$ |
| | | $C_{\text{lift}}$ | $1.0 \cdot 10^{0}$ | $8.9 \cdot 10^{0}$ | $1.2 \cdot 10^{0}$ | $8.1 \cdot 10^{0}$ |
| | | $C_{\text{torque}}$ | $5.6 \cdot 10^{-1}$ | $4.2 \cdot 10^{0}$ | $6.3 \cdot 10^{-1}$ | $3.7 \cdot 10^{0}$ |
| 50/20/20/10 | 1833 | $C_{\text{drag}}$ | $1.2 \cdot 10^{0}$ | $1.1 \cdot 10^{1}$ | $1.3 \cdot 10^{0}$ | $7.9 \cdot 10^{0}$ |
| | | $C_{\text{lift}}$ | $1.2 \cdot 10^{0}$ | $8.6 \cdot 10^{0}$ | $1.3 \cdot 10^{0}$ | $7.9 \cdot 10^{0}$ |
| | | $C_{\text{torque}}$ | $5.6 \cdot 10^{-1}$ | $4.5 \cdot 10^{0}$ | $6.3 \cdot 10^{-1}$ | $3.8 \cdot 10^{0}$ |
| 100/50/20/20 | 6853 | $C_{\text{drag}}$ | $9.6 \cdot 10^{-1}$ | $8.8 \cdot 10^{0}$ | $1.1 \cdot 10^{0}$ | $8.0 \cdot 10^{0}$ |
| | | $C_{\text{lift}}$ | $9.7 \cdot 10^{-1}$ | $8.6 \cdot 10^{0}$ | $1.1 \cdot 10^{0}$ | $7.9 \cdot 10^{0}$ |
| | | $C_{\text{torque}}$ | $5.5 \cdot 10^{-1}$ | $4.1 \cdot 10^{0}$ | $6.2 \cdot 10^{-1}$ | $3.8 \cdot 10^{0}$ |
| 100/50/50/20 | 8983 | $C_{\text{drag}}$ | $9.7 \cdot 10^{-1}$ | $8.6 \cdot 10^{0}$ | $1.1 \cdot 10^{0}$ | $7.8 \cdot 10^{0}$ |
| | | $C_{\text{lift}}$ | $9.6 \cdot 10^{-1}$ | $8.8 \cdot 10^{0}$ | $1.1 \cdot 10^{0}$ | $8.0 \cdot 10^{0}$ |
| | | $C_{\text{torque}}$ | $5.5 \cdot 10^{-1}$ | $4.0 \cdot 10^{0}$ | $6.2 \cdot 10^{-1}$ | $3.6 \cdot 10^{0}$ |

*Note*: Each network predicts all three functional values simultaneously.

20,000 epochs. For the network to predict all three values simultaneously, we scale both the input and output data to be in the interval $[-1, 1]$. In practice, the predictions are then scaled back appropriately, so that the appropriate coefficients are obtained.

To ensure that we do not overfit the network to the training data set, we also generated a validation data set in the same fashion as the translational part of the training data set but with different angles of attack and values for $t_{\text{end}}$. This then consists of $4.3 \cdot 10^5$ data points. We evaluate the network on the validation data set during training and ensure that the training error does not decrease while the validation error increases. To make the results on data sets of different sizes comparable, we use the norm $\|\text{err}\|^2_{\text{mean}} := \frac{1}{N}\sum_{i=1}^{N}(\text{prediction}_i - \text{value}_i)^2$ and $\|\text{err}\|_\infty = \max_{i=1,\dots,N}(|\text{prediction}_i - \text{value}_i|)$.

To find an appropriate network size, which is large enough to capture all the information contained in the data set while also being small enough for fast evaluations in the final solver, we consider several different networks. The chosen number of layers and neurones per layer can be seen in the first column of Table 5.

Looking at the results in Table 5, we see that these are broadly similar for all six considered network architectures. For the three-layer networks, we observe that the prediction error resulting from the 30/20/10 network are almost twice as large as that of the 50/20/20 network. Looking at the four-layer networks, we see that the errors are generally the same as for the 50/20/20 network, with some small deviations in both directions.

In order to check whether we can gain more accuracy by considering separate networks for the three functionals, we train three separate 50/20/20 networks. The results thereof can be seen in Table 6, where we observe that the prediction errors are about the same as those realized by the single network with the same architecture in Table 5.

Both in Tables 5 and 6, we observe that the errors are generally similar and very large. To give these errors more meaning, we plot the predictions and the target values for a random selection of points from the validation data set in Figure 4. Here we see that, in general, the predicted values match the target values well. This indicates that the size of the errors in Tables 5 and 6 are due to the targets' value size. The errors in Table 5 indicate that overall the force dynamics are captured with about 1–2 significant figures of accuracy.

As a result of the above considerations, we choose the single 50/20/20 network to predict the drag lift and torque from the average velocity around the rigid particle in our applications. Considering this network as a function $\mathbb{R}^2 \to \mathbb{R}^3$, we can then plot the individual components as a function of the input in a three-dimensional plot. This can be seen in Figure 5.

**TABLE 6** Prediction errors in a weighted $\ell^2$ norm and the maximum norm on the training and validation data-sets after 20,000 epochs

| | Training data | | Validation data | |
|---|---|---|---|---|
| **Target** | $\|\cdot\|_{\text{mean}}$ | $\|\cdot\|_\infty$ | $\|\cdot\|_{\text{mean}}$ | $\|\cdot\|_\infty$ |
| $C_{\text{drag}}$ | $9.6 \cdot 10^{-1}$ | $8.8 \cdot 10^{0}$ | $1.1 \cdot 10^{0}$ | $8.1 \cdot 10^{0}$ |
| $C_{\text{lift}}$ | $1.0 \cdot 10^{0}$ | $8.7 \cdot 10^{0}$ | $1.1 \cdot 10^{0}$ | $7.9 \cdot 10^{0}$ |
| $C_{\text{torque}}$ | $5.5 \cdot 10^{-1}$ | $4.1 \cdot 10^{0}$ | $6.2 \cdot 10^{-1}$ | $3.6 \cdot 10^{0}$ |

*Note*: Separate networks of the architecture 50/20/20 with 1611 parameters are used to predict the three functional values.

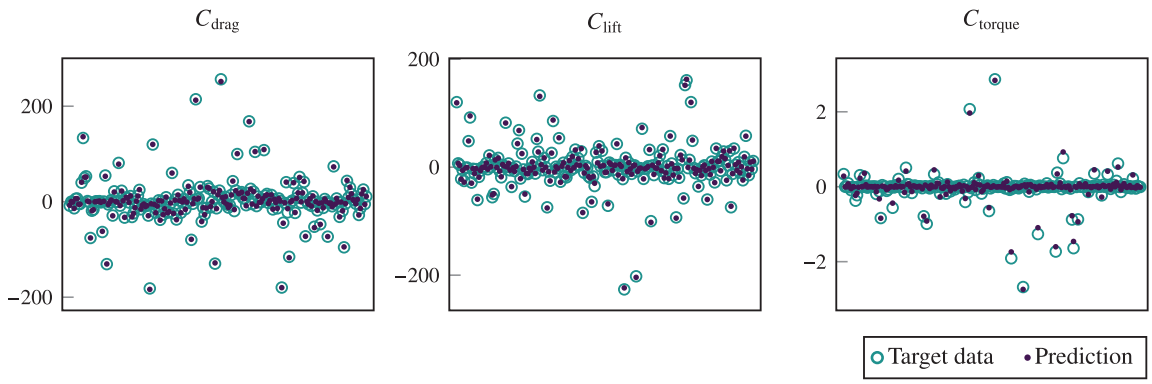


**FIGURE 4** Target data and network prediction for 300 random points in the training data set. Network architecture: 50/20/20, 1653 parameters [Colour figure can be viewed at wileyonlinelibrary.com]

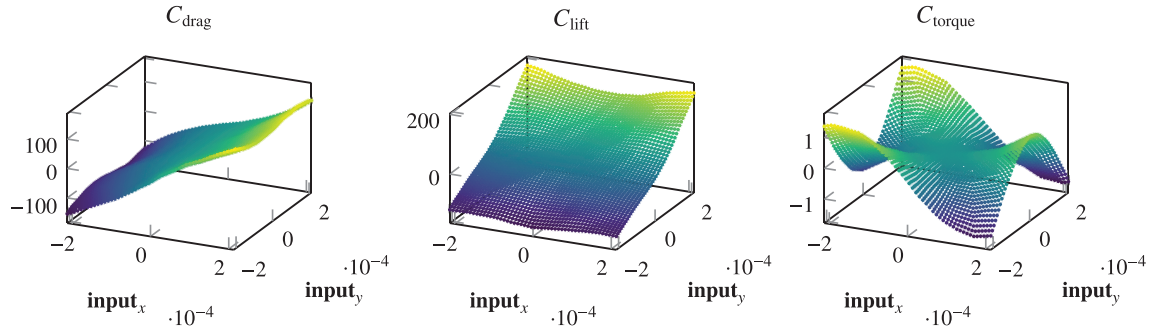


**FIGURE 5** Network prediction components from the 50/20/20 network as functions of input variables [Colour figure can be viewed at wileyonlinelibrary.com]

This illustrates that while a relatively simple function represents the drag and lift coefficients, the torque coefficient is significantly more involved.

Due to the simple input for the network, requiring two integrals over a relatively small area and a rotation of these two values into the reference configuration, as well as the small size of the network itself, the additional computational effort introduced to the solver by predicting the forces, rather than evaluating them directly, is negligible compared with the effort required to solve the nonlinear system resulting from the FEM discretization of the Navier–Stokes equations, see Remark 6.

*Remark* 4. *Training times*. The training of the above networks was performed on a Tesla V100 PCIe 16GB graphics card with `PyTorch` using `CUDA` version 10.1. Due to the small size and simple structure of the networks, the training times for 20,000 epochs ranged between 277 s for the 30/20/10 network and 936 s for the 100/50/20/20 network.

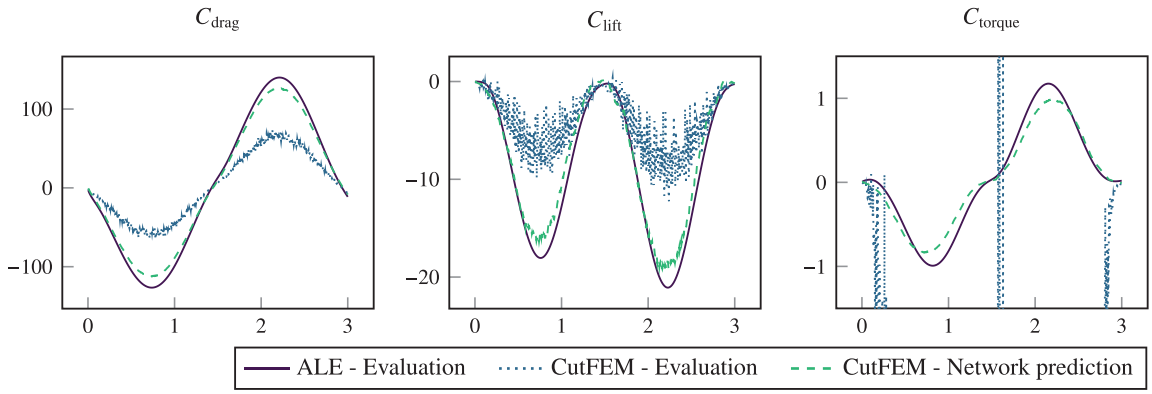


**FIGURE 6** Prediction and evaluation in a CutFEM simulation with $\text{TH}_2$ elements compared against the values evaluated in a fitted ALE computation [Colour figure can be viewed at wileyonlinelibrary.com]

**TABLE 7** Absolute errors in the forces resulting from the direct evaluation of the boundary integrals and the prediction made by the neural network in a CutFEM computation

| | | $C_{\text{drag}}$ | | $C_{\text{lift}}$ | | $C_{\text{torque}}$ | |
|---|---|---|---|---|---|---|---|
| **Force comp.** | **Discr.** | $\|\cdot\|_{\text{mean}}$ | $\|\cdot\|_\infty$ | $\|\cdot\|_{\text{mean}}$ | $\|\cdot\|_\infty$ | $\|\cdot\|_{\text{mean}}$ | $\|\cdot\|_\infty$ |
| Evaluation | $\text{TH}_2$ | $7.9 \cdot 10^{0}$ | $8.4 \cdot 10^{1}$ | $1.3 \cdot 10^{0}$ | $3.3 \cdot 10^{1}$ | $1.4 \cdot 10^{2}$ | $2.3 \cdot 10^{3}$ |
| Prediction | $\text{TH}_2$ | $1.5 \cdot 10^{0}$ | $2.3 \cdot 10^{1}$ | $2.3 \cdot 10^{-1}$ | $4.8 \cdot 10^{0}$ | $1.5 \cdot 10^{-1}$ | $2.9 \cdot 10^{0}$ |
| Evaluation | $\text{TH}_3$ | $6.6 \cdot 10^{0}$ | $7.2 \cdot 10^{1}$ | $1.5 \cdot 10^{0}$ | $4.1 \cdot 10^{1}$ | $2.4 \cdot 10^{2}$ | $3.9 \cdot 10^{3}$ |
| Prediction | $\text{TH}_3$ | $1.5 \cdot 10^{0}$ | $2.3 \cdot 10^{1}$ | $2.4 \cdot 10^{-1}$ | $5.1 \cdot 10^{0}$ | $1.5 \cdot 10^{-1}$ | $2.9 \cdot 10^{0}$ |

## 3.4 | Validation

In the previous section, we have compared the network predictions against the "true" values in the sense of the ALE training data. However, the network aims to predict the force values in a CutFEM simulation and be more accurate than the boundary-integral evaluation.

To validate that the neural network predictions are more accurate than the direct evaluation of the forces from the boundary integral, we run a moving domain CutFEM simulation of the setup, with which we generated the training data for $t_{\text{end}} = 3$. Here we took a background mesh with $h_{\max} = 10^{-3}$ in the area where we compute the average velocity around the triangle and $h_{\max} = 0.04$ in the remaining part of the domain. Therefore, the mesh in the averaging area is a factor of two smaller than the size of the rigid body. On this mesh we work with unfitted $\text{TH}_2$ and $\text{TH}_3$ elements. We then compute the force prediction and evaluation errors by comparing the values against the direct evaluation of a highly resolved ALE computation of the identical setup. The spatial discretization is as for the training data generation above. In both cases, we chose the time-step $\Delta t = 1/300$.

The resulting prediction errors of the forces in the CutFEM simulation are given in Table 7. Here we find that mean and maximal prediction errors are at least one order of magnitude smaller than the direct evaluation. In Figure 6, we plot the resulting forces for a single run of the above computations ($\alpha = 0$). This shows that while there is a visible difference between the prediction and the "ground truth," the predictions mirror the real forces much better than the direct force computation via the boundary-integral evaluation. We also note, that there was no significant improvement in the direct evaluation when using higher-order elements ($k = 3$). We conclude that while the predictions are not perfect, we have significantly improved on the direct computation of the forces on an underresolved computational mesh. However, we also note that we cannot expect any asymptotic mesh convergence here, as the prediction error will begin to dominate once the interface is sufficiently resolved in every time-step.

# 4 | NUMERICAL EXAMPLES

We consider several numerical examples that use the neural network trained in the previous section in an underresolved, moving domain, CutFEM discretization. The details and analysis of this method applied to the time-dependent Stokes equations on moving domains are given in Reference 3. This method uses ghost-penalty stabilization to implement a discrete, implicit extension of the solution into the exterior of the fluid domain, to apply a BDF formula to the time derivative in the case of a moving interface. The only modification here is the addition of the convective term. See also Reference 11, where this method has also been applied to a fluid–structure interaction problem with contact and in which the fluid is described by the incompressible Navier–Stokes equations.

To solve the coupled fluid–solid system, we use a partitioned approach. In the substep to update the solid velocity, we use an update scheme with relaxation, together with Aitken's $\Delta^2$ method[39] to determine the appropriate relaxation parameter and thereby accelerate the relaxation scheme. A relaxation in the scheme is necessary for the stability of the partitioned solver of the coupled system, and different relaxation strategies, including Aitken's $\Delta^2$ method, are discussed in the literature.[40]

*Remark* 5. *Geometry description in CutFEM*. In CutFEM, numerical integration on elements that are cut by the level-set function describing the geometry is based on a piecewise linear interpolation of the (smooth) level set function. Such straight cuts are necessary for a robust construction of quadrature rules on cut elements.[4] To represent the triangular rigid bodies' three sides, we use three separate level set functions $\phi_1$, $\phi_2$, and $\phi_3$. `ngsxfem`[33] then provides the functionality to integrate with respect to each level set, for example, the domain where all level sets are negative. Since the level sets describing the sides are linear. The numerical integration on the geometry posed here is exact, and we do not have to construct a single level set function $\phi = \max(\phi_1, \phi_2, \phi_3)$ which would then be interpolated into the $\mathbb{P}^1$ space on the mesh, which in turn would remove the sharp corners of the particles.

## 4.1 | Example 1: Free-fall restricted to vertical motion

In Subsection 3.4, we validated the neural network approach in the setting of prescribed solid motion. To validate the method in our target setting of free-fall, we consider the settling of a single particle, restricted to vertical motion as in Subsubsection 3.2.1. We consider this simplification in order to compare the results against a resolved ALE simulation.

To this end, we take the domain $\Omega = (0, 0.1)^2$, assert no-slip boundary conditions at the left and right wall and the do-nothing condition $\mu_F \nabla \boldsymbol{un} = p\boldsymbol{n}$ at the top and bottom, see also Reference 41 for details thereon. The rigid body is an equilateral triangle with side-length $2 \cdot 10^{-3}$. The fluid and solid material parameters are as in Table 3. At $t = 0$, the center of mass of the rigid body is at $(0.05, 0.08)$, and we rotate the body by an angle $\alpha$ with respect to reference configuration, in which the bottom of the triangle is parallel to the $x$-axis. We shall consider $\alpha = 0, \pi/12, \pi/6$. See also the left sketch in Figure 7 for an illustration of this initial configuration.

For the ALE computation, we consider $\mathrm{TH}_5$ elements on a mesh with $h_{\max} = 5.0 \cdot 10^{-3}$, $h = 1.3 \cdot 10^{-3}$ in a horizontal strip of height $8.0 \cdot 10^{-3}$ around the rigid body and $h = 4.0 \cdot 10^{-4}$ on the interface of the rigid body. Based on our validation experiments in Subsubsection 3.2.2, this discretization is sufficiently accurate to serve as a reference here.

To ensure that the observed motion is due to our neural network approach, rather than of the discretization itself, we again consider two different CutFEM simulations. First, we base the solid motion on the forces predicted by the neural network, while in the second simulation, the forces are computed by the direct evaluation of the boundary integral with the FEM solution. For both CutFEM computations, we consider a mesh with $h_{\max} = 1.0 \cdot 10^{-3}$ using $\mathrm{TH}_2$ elements. For the unfitted discretization used, we take the Nitsche parameter to enforce the Dirichlet boundary conditions on the level set interfaces as $\gamma_{\mathrm{N}} = 100$ while the stability and extension ghost-penalty parameters are $\gamma_{\mathrm{gp,stab}} = 0.1$ and $\gamma_{\mathrm{gp,ext}} = 0.001$, respectively. For details on these parameters, we refer to Reference 3.

We take the time-step $\Delta t = 1/250$ and compute until $t = 1.0$. For the partitioned fluid–solid solver, we take the initial relaxation parameter as $\omega = 0.5$, allow a maximum of 10 subiterations per time step and a tolerance of 1% in the relative update.

The forces acting from the fluid onto the rigid body are evaluated using the single 50/20/20 network for drag, lift, and torque. As discussed above, we use an isoparametric approach to accurately integrate the fluid velocity relative to the solid velocity in $\mathcal{O}(\mathcal{B})$.

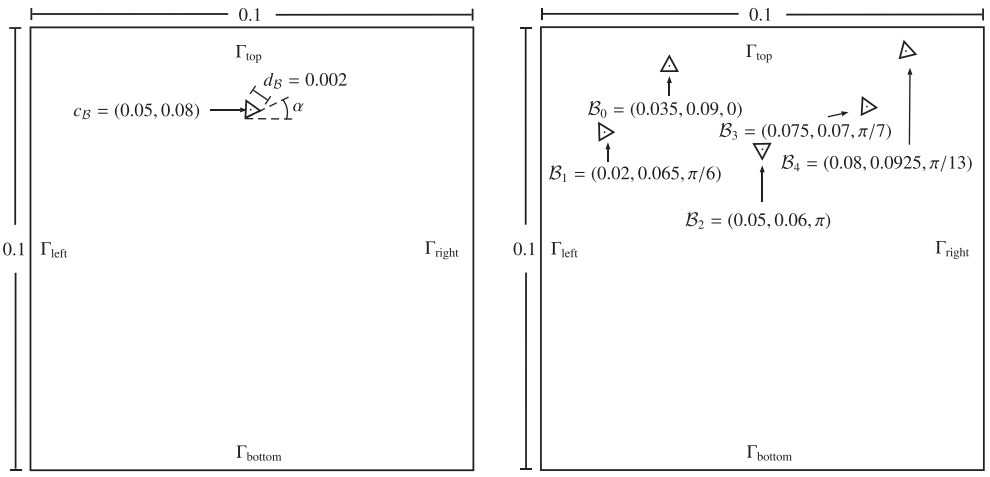


**FIGURE 7** Domain sketches for the initial configurations of the computational examples. Left: a single particle (Subsections 4.1 and 4.2). Right: five particles (Subsection 4.3 3)

In Figure 8, we can see the vertical force, the vertical speed, and the vertical position of the center of mass of the rigid body over time for each of the three triangle orientations. Here we see that for all three orientations, the drag obtained by the network prediction is significantly more accurate than the direct evaluation of the boundary integral. Looking at the body's vertical speed and position, we see a visible difference between the ALE reference solution and the movement resulting from the predicted forces. The uniformly faster speed in the CutFEM prediction computations can be attributed to the fact that the network underestimated the drag at low speeds of the particle, as shown in the left column of Figure 8. However, we also see that the predictions are significantly better than the result from the direct evaluation of the forces. This clearly shows that the neural network approach can realize accurate results on very coarse and underresolved meshes, where the unfitted approach with the standard evaluation of the resulting forces does not lead to realistic motion.

*Remark* 6. *Neural network overhead*. Using an Intel Core i7-7700 CPU and four threads in parallel, the computational time of the hybrid CutFEM/neural network computation was 510.07 s, of which 13.94 s (2.7%) was computational overhead to obtain the necessary data to feed into the neural network and postprocess the neural network output and 0.164 s (0.032%) to evaluate the neural network. As a result, we note that the additional computational effort introduced by the neural network is not significant. In contrast, the computational time of the standard CutFEM method was 679.04 s, of which 2.45 s (0.36%) was needed to evaluate the forces. Compared with the hybrid approach, the longer computational time is due to more subiterations needed in every time-step because of the observed instabilities in the inaccurate force evaluation. We also note it is not only that computational time that is increased but that the approximation quality is significantly lower, as shown in Figure 8. Finally, the computational time of the ALE reference was 297.37 s, of which 5.03 s (1.7%) was used to evaluate the forces. We emphasize that the strength of the hybrid approach is not computational speed compared with the ALE approach but that it gives accurate results on very coarse meshes, such that the fluid–solid interface is underresolved, and the applicability of the approach in situations where an ALE discretization is not straightforward, such as the examples below. In fact, in situations where an ALE discretization can be applied, we consider it the method of choice.

## 4.2 | Example 2: Single triangular rigid body

We now consider the full fluid-rigid body system, including translational movement in all directions and rotational motion. As a result, we do not have an ALE reference with which to compare the results. The initial configuration is chosen as in Subsection 4.1, with the initial rotation $\alpha = \pi/3$.

We again take the background mesh with $h_{\max} = 1.0 \cdot 10^{-3}$ and a second mesh with $h_{\max} = 7.5 \cdot 10^{-4}$ to investigate the mesh dependence. On each mesh, we take $\mathrm{TH}_2$ elements. The remaining discretization parameters are chosen as in the unfitted simulation in Subsection 4.1.

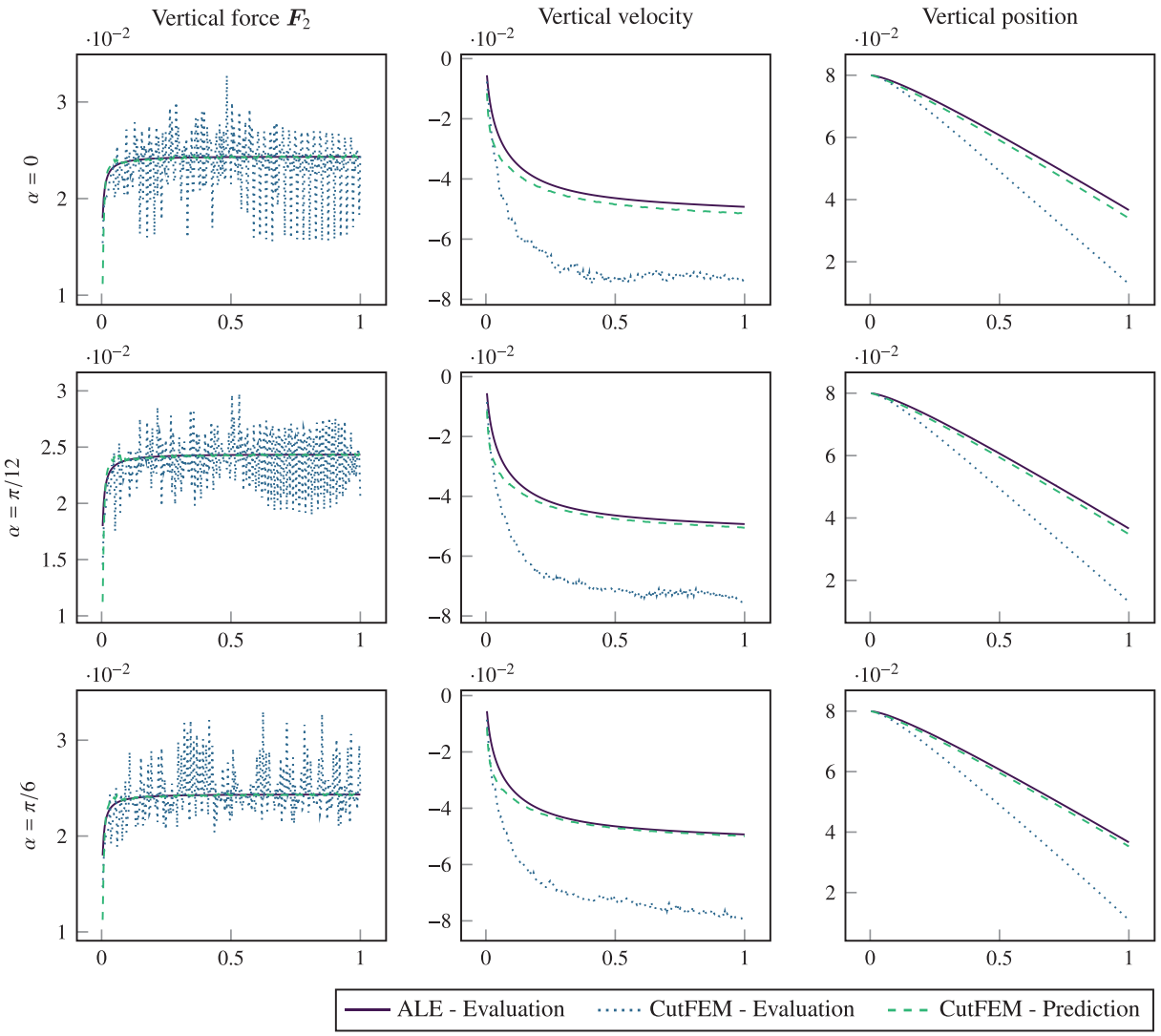


**FIGURE 8** Vertical force, velocity, and position of a single falling triangular rigid body restricted to vertical motion. Comparison between an ALE reference computation, the force computation in an under resolved CutFEM computation (Evaluation) and the network prediction (Prediction) in an underresolved CutFEM computation [Colour figure can be viewed at wileyonlinelibrary.com]

In Figure 9, we see the resulting velocity components that make up the rigid body movement for the two considered meshes. The fluid solution at time $t = 1.0$ on the coarser of the two meshes is visualized in Figure 10.

Looking at velocity components in Figure 9, we immediately see that the influence of the mesh (in the considered under resolved range) is minimal. While this does not indicate that the method is accurate, it does show that the method is stable. As is to be expected, the vertical velocity component dominates the translational motion. Furthermore, we see that while a terminal velocity is not fully reached, that the acceleration between $t = 0.5$ and $t = 1.5$ is small, and the velocity during this time is very similar and only about 10% faster compared with the a priori computations in Subsubsection 3.2.1 and in our validation example with restricted motion in Subsection 4.1.

The angular velocity in Figure 9 appears very large at first sight. Factoring in that the side-length of the body is $2 \cdot 10^{-3}$, we find that the resulting maximal velocity at the corners of the triangular rigid bodies is of order $10^{-2}$. So, in general, the velocity of the triangle resulting from rotation is smaller than the vertical velocity component.

Overall, this example shows that our method leads to plausible results on highly underresolved meshes, on which the standard CutFEM approach cannot realize accurate forces and thereby cannot realize accurate solid motion. Finally, we emphasize that a comparison to a fitted ALE simulation is out of scope for this example since this would have to include remeshing procedures and projections of the solution onto the resulting new meshes to avoid mesh-entanglement resulting from the rotation of the particles.

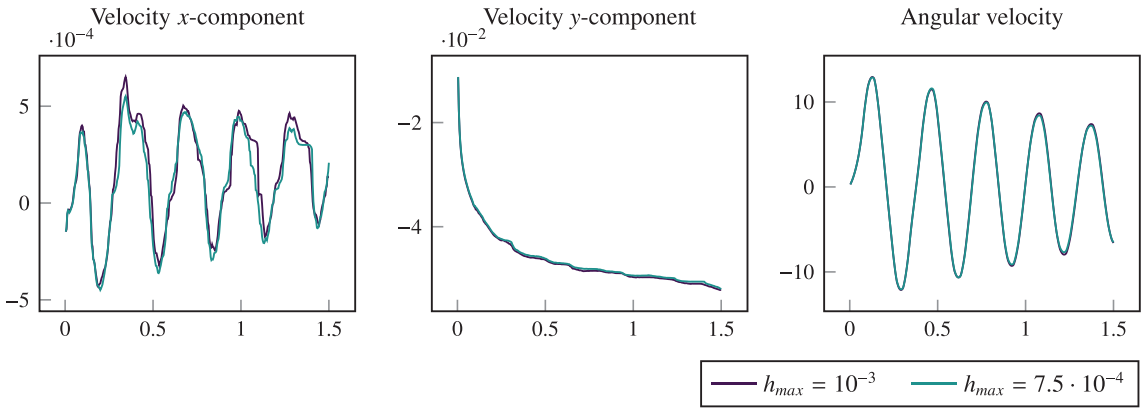


**FIGURE 9** Translational and rotational velocity components of a single triangular rigid body in free fall on an underresolved mesh with the forces from the predictions by a deep neural network [Colour figure can be viewed at wileyonlinelibrary.com]

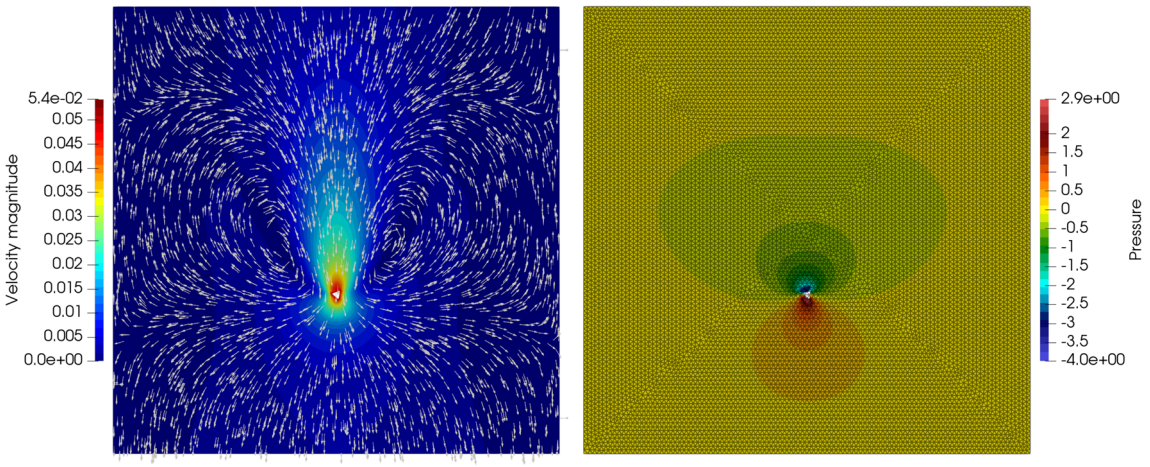


**FIGURE 10** Velocity field (left) and pressure with mesh (right) at $t = 1.0$ resulting from a single triangular rigid body in free fall on a mesh with $h_{\max} = 10^{-3}$ with the forces governing the solid motion obtained from a neural network [Colour figure can be viewed at wileyonlinelibrary.com]

## 4.3 | Example 3: Multiple rigid bodies

As a final more advanced example, consider the same basic setup as in Subsection 4.2 and take five rigid triangular rigid bodies denoted by $\mathcal{B}_i, i = 0, \dots, 4$. Their initial position and orientation are then given by their center of mass as the angle of rotation with respect to the reference configuration. The reference orientation is such that the bottom side of the triangle is parallel to the $x$-axis. We denote the initial center of mass and rotation by $(c_x, c_y, \alpha)$. We consider the initial states $(0.035, 0.09, 0)$, $(0.02, 0.065, \pi/6)$, $(0.05, 0.06, \pi)$, $(0.075, 0.07, \pi/7)$, and $(0.08, 0.0925, \pi/13)$ for $\mathcal{B}_0, \dots, \mathcal{B}_4$, respectively. A sketch of this configuration can be seen in the right of Figure 7. The discretization parameters of the moving domain CutFEM method are all as in Subsection 4.2.

The results from these computations are shown in Figures 11 and 12. The translational and rotational velocity components are shown in Figure 11, while the fluid solution at $t = 1.0$ is shown in Figure 12.

In Figure 11, we see that there is again no significant dependence on the mesh. In general, the translational velocity of all particles is larger than in the case of a single particle above. Nevertheless, the order of magnitude is the same as before. The slightly faster translational speeds of the particles make sense, as each particle sets the fluid in motion, which then helps to transport the other particles. Furthermore, we see both in Figures 11 and 12 that $\mathcal{B}_4$ has the largest (vertical) velocity. This observation is also meaningful, as $\mathcal{B}_4$ is directly in the wake of $\mathcal{B}_3$. Furthermore, we note that the angular velocity of $\mathcal{B}_0$ and $\mathcal{B}_4$, which are above the other particles, is smaller than that of the other particles. We attribute this

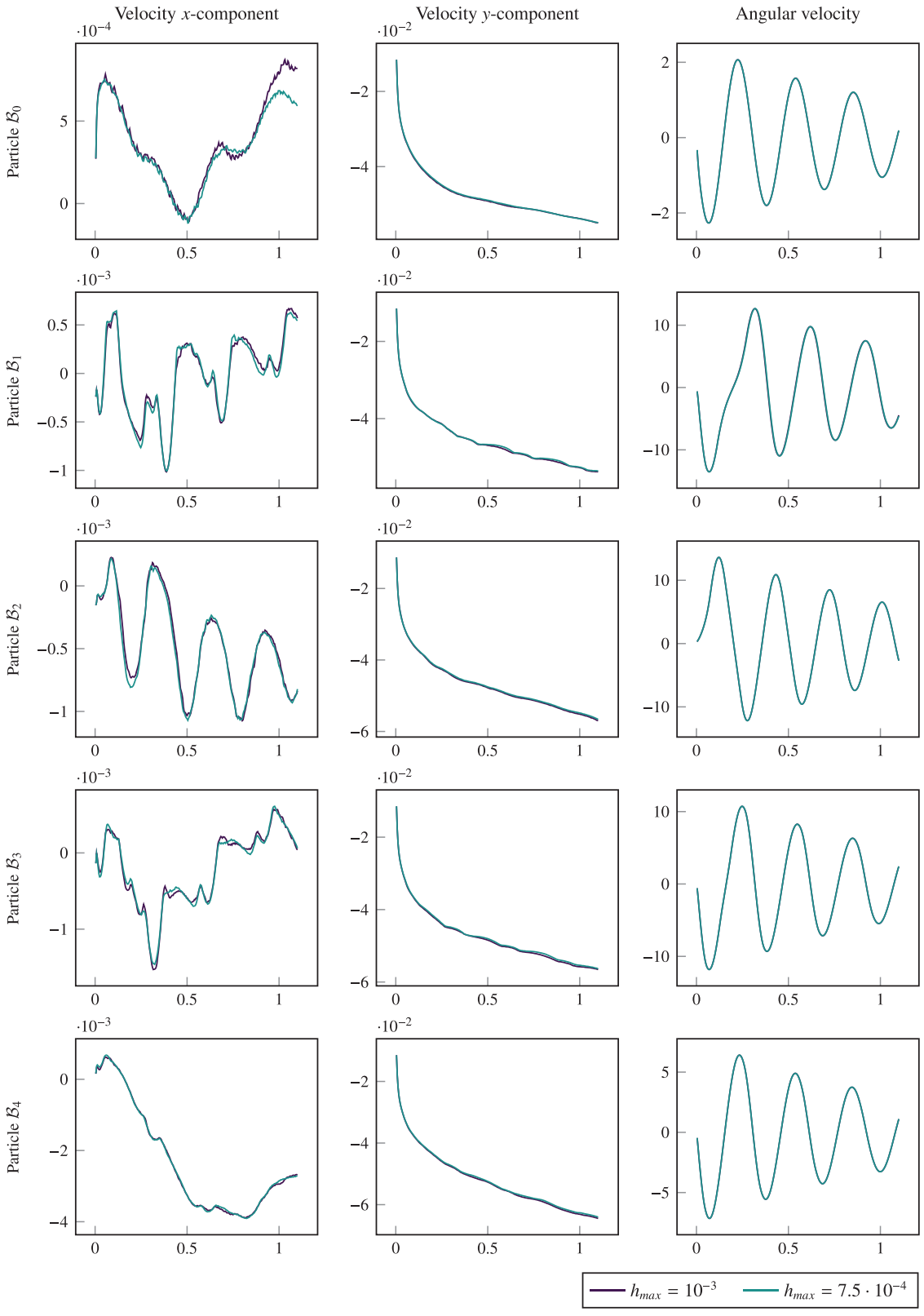


**FIGURE 11** Translational and rotational velocity of five triangular rigid bodies in free fall in an underresolved CutFEM simulation where the forces governing the solid motion are from the evaluation of a deep neural network [Colour figure can be viewed at wileyonlinelibrary.com]

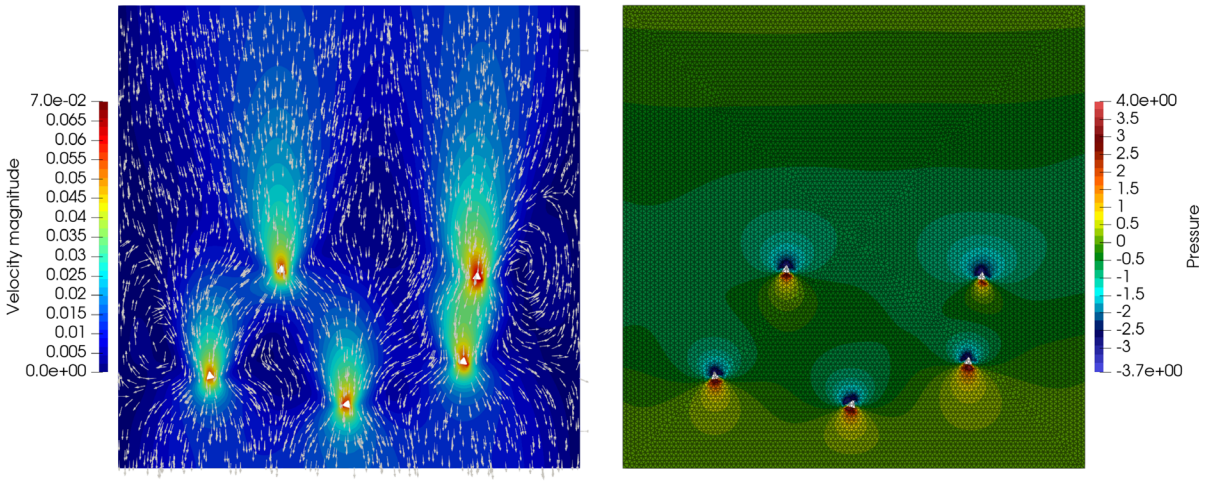


**FIGURE 12** Velocity field (left) and pressure with mesh (right) at $t = 1.0$ resulting from five triangular rigid bodies in free fall on a mesh with $h_{\max} = 10^{-3}$ with the forces governing the solid motion obtained from a neural network [Colour figure can be viewed at wileyonlinelibrary.com]

to the fact that these two particles are in the parallel flow wake of the other three particles, resulting in a smaller torque acting on these two particles.

## 5 | CONCLUSIONS AND OUTLOOK

### 5.1 | Conclusions

We have presented a framework using a hybrid finite element/neural network approach to obtain improved drag lift and torque values acting on a triangular rigid body. For this, we trained a small deep neural network using training data obtained from prototypical motion configurations in resolved finite element simulations. We considered a range of networks by varying the number of layers as well as the number of neurones per layer. Here we saw that the mapping from the average velocity around the particle to the forces acting on this particle was accurately captured by relatively small networks.

Applying one of these networks in an underresolved moving domain CutFEM simulation, we showed that our neural network approach results in significantly more accurate force values than the direct evaluation of the forces from the finite element solution. We saw that the predicted values were, on average, an order of magnitude more accurate than the direct evaluation of the forces from the Navier–Stokes solution. At the same time, the hybrid neural network approach did not introduce a significant overhead in the computations at run time.

Finally, we note that the presented scope of the method remains limited due to the restriction to two-dimensional fluid and particle domains. Also, our choice for the neural network input restricted the application to a relatively small number of particles, which were at a significant distance to each other and the wall boundaries of the fluid domain.

### 5.2 | Outlook

A number of interesting questions remain for future research. For example, it would be very interesting to compare the particle motion realized in Subsections 4.2 and 4.3 to fitted ALE simulations using remeshing techniques. Furthermore, it remains to generalize this approach to other particle shapes and sizes. In this context, it would also be interesting to see if a different choice of input data could generalize the approach to use a single neural network to predict accurate forces with different material parameters.

Finally, the method should be extended to the interaction of several closely neighboring particles and the particle-wall interaction. In particular, extended training data sets have to be created for this purpose. Also, it has to be explored whether

it is sufficient to consider the mean ambient velocity as network input in this case or whether further information, for example, the velocity gradient or acceleration rates of the particles or the fluid, should be taken into account. In particular, when extending the approach to three-dimensional scenarios, it needs to be investigated whether such simple input data can capture the necessary information for a neural network to predict the forces accurately. It is reasonable to expect that more input data is necessary since the three-dimensional force and torque are both three-dimensional, such that the network needs to predict six values rather than three.

## ACKNOWLEDGMENT

The authors acknowledge support by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)-314838170, GRK 2297 MathCoRe.

## DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

## ORCID

*Henry von Wahl* https://orcid.org/0000-0002-0793-1647
*Thomas Richter* https://orcid.org/0000-0003-0206-3606

## REFERENCES

1. Lehrenfeld C, Olshanskii MA. An Eulerian finite element method for PDEs in time-dependent domains. *ESAIM Math Model Numer Anal*. 2019;53(2):585-614. https://doi.org/10.1051/m2an/2018068
2. Burman E, Frei S, Massing A. Eulerian time-stepping schemes for the non-stationary Stokes equations on time-dependent domains; 2019. arXiv:1910.03054.
3. von Wahl H, Richter T, Lehrenfeld C. An unfitted Eulerian finite element method for the time-dependent Stokes problem on moving domains. *IMA J Numer Anal*. 2021. https://doi.org/10.1093/imanum/drab044
4. Burman E, Claus S, Hansbo P, Larson MG, Massing A. CutFEM: discretizing geometry and partial differential equations. *Int J Numer Methods Eng*. 2014;104(7):472-501. https://doi.org/10.1002/nme.4823
5. Felice RD. The voidage function for fluid-particle interaction systems. *Int J Multiphase Flow*. 1994;20:153-159. https://doi.org/10.1016/0301-9322(94)90011-6
6. Hölzer A, Sommerfeld M. New simple correlation formula for the drag coefficient of non-spherical particles. *Powder Technol*. 2008;184(3):361-365. https://doi.org/10.1016/j.powtec.2007.08.021
7. Mandø M, Rosendahl L. On the motion of non-spherical particles at high Reynolds number. *Powder Technol*. 2010;202(1-3):1-13. https://doi.org/10.1016/j.powtec.2010.05.001
8. Zastawny M, Mallouppas G, Zhao F, van Wachem B. Derivation of drag and lift force and torque coefficients for non-spherical particles in flows. *Int J Multiphase Flow*. 2012;39:227-239. https://doi.org/10.1016/j.ijmultiphaseflow.2011.09.004
9. Zhong W, Yu A, Liu X, Tong Z, Zhang H. DEM/CFD-DEM modelling of non-spherical particulate systems: theoretical developments and applications. *Powder Technol*. 2016;302:108-152. https://doi.org/10.1016/j.powtec.2016.07.010
10. Wan D, Turek S. Fictitious boundary and moving mesh methods for the numerical simulation of rigid particulate flows. *J Comput Phys*. 2007;222(1):28-56. https://doi.org/10.1016/j.jcp.2006.06.002
11. von Wahl H, Richter T, Frei S, Hagemeier T. Falling balls in a viscous fluid with contact: comparing numerical simulations with experimental data. *Phys Fluids*. 2021;33(2). https://doi.org/10.1063/5.0037971
12. Hughes TJR, Liu WK, Zimmermann TK. Lagrangian-Eulerian finite element formulations for incompressible viscous flows. *Comput Methods Appl Mech Eng*. 1981;29:329-349. https://doi.org/10.1016/0045-7825(81)90049-9
13. Donea J. An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions. *Comput Methods Appl Mech Eng*. 1982;33:689-723. https://doi.org/10.1016/0045-7825(82)90128-1
14. Richter T, Wick T. Finite elements for fluid-structure interaction in ALE and fully Eulerian coordinates. *Comput Methods Appl Mech Eng*. 2010;199(41-44):2633-2642. https://doi.org/10.1016/j.cma.2010.04.016
15. Peskin CS. The immersed boundary method. *Acta Numer*. 2002;11:479-517. https://doi.org/10.1017/s0962492902000077
16. Azis MHA, Evrard F, van Wachem B. An immersed boundary method for flows with dense particle suspensions. *Acta Mech*. 2018;230(2):485-515. https://doi.org/10.1007/s00707-018-2296-y
17. Happel J, Brenner H. *Low Reynolds Number Hydrodynamics*. Springer; 1981.
18. Minakowska M, Richter T, Sager S. A finite element / neural network framework for modeling suspensions of non-spherical particles. *Vietnam J Math*. 2021;49(1):207-235. https://doi.org/10.1007/s10013-021-00477-9
19. Hansbo A, Hansbo P. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. *Comput Methods Appl Mech Eng*. 2002;191(47-48):5537-5552. https://doi.org/10.1016/s0045-7825(02)00524-8
20. Fidkowski K, Darmofal D. A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations. *J Comput Phys*. 2007;225(2):1653-1672. https://doi.org/10.1016/j.jcp.2007.02.007

21. Bastian P, Engwer C. An unfitted finite element method using discontinuous Galerkin. *Int J Numer Methods Eng*. 2009;79(12):1557-1576. https://doi.org/10.1002/nme.2631
22. Langer U, Yang H. Numerical simulation of parabolic moving and growing interface problems using small mesh deformation; 2015. arXiv:1507.08784.
23. Fang X. An isoparametric finite element method for elliptic interface problems with nonhomogeneous jump conditions. *WSEAS Trans Math*. 2013;12(1). http://www.wseas.org/multimedia/journals/mathematics/2013/56-222.pdf.
24. Frei S, Richter T. A locally modified parametric finite element method for interface problems. *SIAM J Num Anal*. 2014;52:2315-2334. https://doi.org/10.1137/130919489
25. Schäfer M, Turek S. Benchmark computations of laminar flow around a cylinder. In: Hirschel EH, ed. *Flow Simulation with High-Performance Computers II. Notes on Numerical Fluid Mechanics (NNFM)*. Vieweg+Teubner Verlag; 1996:547-566.
26. Babuška I, Miller A. The post-processing approach in the finite element method-part 1: calculation of displacements, stresses and other higher derivatives of the displacements. *Int J Numer Methods Eng*. 1984;20(6):1085-1109. https://doi.org/10.1002/nme.1620200610
27. Burman E, Hansbo P. Fictitious domain methods using cut elements: III. a stabilized Nitsche method for Stokes' problem. *ESAIM Math Model Numer Anal*. 2014;48(3):859-874. https://doi.org/10.1051/m2an/2013123
28. Massing A, Larson MG, Logg A, Rognes ME. A stabilized Nitsche fictitious domain method for the Stokes problem. *J Sci Comput*. 2014;61(3):604-628. https://doi.org/10.1007/s10915-014-9838-9
29. Lehrenfeld C. High order unfitted finite element methods on level set domains using isoparametric mappings. *Comput Methods Appl Mech Eng*. 2016;300:716-733. https://doi.org/10.1016/j.cma.2015.12.005
30. Lehrenfeld C. A higher order isoparametric fictitious domain method for level set domains. In: Bordas S, Burman E, Larson M, Olshanskii M, eds. *Lecture Notes in Computational Science and Engineering*. Vol 121. Springer; 2017:65-92.
31. Schöberl J. C++11 implementation of finite elements in NGSolve. ASC Report No. 30/2014; 2014.
32. Schöberl J. NETGEN an advancing front 2D/3D-mesh generator based on abstract rules. *Comput Vis Sci*. 1997;1(1):41-52. https://doi.org/10.1007/s007910050004
33. Lehrenfeld C, Heimann F, Preuß J, von Wahl H. ngsxfem: an add-on to NGSolve for unfitted finite element discretizations. *J Open Source Softw*. 2021;6(64):3237. https://doi.org/10.21105/joss.03237
34. John V, Linke A, Merdon C, Neilan M, Rebholz LG. On the divergence constraint in mixed finite element methods for incompressible flows. *SIAM Rev*. 2017;59(3):492-544. https://doi.org/10.1137/15m1047696
35. Raissi M, Yazdani A, Karniadakis GE. Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations. *Science*. 2020;367(6481):1026-1030. https://doi.org/10.1126/science.aaw4741
36. Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys*. 2019;378:686-707. https://doi.org/10.1016/j.jcp.2018.10.045
37. John V, Matthies G, Rang J. A comparison of time-discretization/linearization approaches for the incompressible Navier-Stokes equations. *Comput Methods Appl Mech Eng*. 2006;195(44-47):5995-6010. https://doi.org/10.1016/j.cma.2005.10.007
38. Paszke A, Gross S, Massa F, et al. PyTorch: an imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R, eds. *Advances in Neural Information Processing Systems NIPS*. Vol 32. Curran Associates, Inc; 2019:8026-8037.
39. Irons BM, Tuck RC. A version of the Aitken accelerator for computer iteration. *Int J Numer Methods Eng*. 1969;1(3):275-277. https://doi.org/10.1002/nme.1620010306
40. Küttler U, Wall WA. Fixed-point fluid-structure interaction solvers with dynamic relaxation. *Comput Mech*. 2008;43(1):61-72. https://doi.org/10.1007/s00466-008-0255-5
41. Heywood JG, Rannacher R, Turek S. Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations. *Int J Numer Methods Fluids*. 1996;22:325-352. https://doi.org/10.1002/(SICI)1097-0363(19960315)22:5<325::AID-FLD307>3.0.CO;2-Y


**How to cite this article:** von Wahl H, Richter T. Using a deep neural network to predict the motion of underresolved triangular rigid bodies in an incompressible flow. *Int J Numer Meth Fluids*. 2021;93(12):3364-3383. https://doi.org/10.1002/fld.5037