Received: 11 March 2020 | Revised: 4 January 2021 | Accepted: 9 January 2021

DOI: 10.1002/fld.4957

RESEARCH ARTICLE

WILEY

A meshfree Lagrangian method for flow on manifolds

Pratik Suchde

Fraunhofer ITWM, Kaiserslautern, Germany

Correspondence
Pratik Suchde, Fraunhofer ITWM, 67663 Kaiserslautern, Germany.
Email: pratik.suchde@itwm.fraunhofer.de

Abstract

In this article, we present a novel meshfree framework for fluid flow simulations on arbitrarily curved surfaces. First, we introduce a new meshfree Lagrangian framework to model flow on surfaces. Meshfree points or particles, which are used to discretize the domain, move in a Lagrangian sense along the given surface. This is done without discretizing the bulk around the surface, without parametrizing the surface, and without a background mesh. A key novelty that is introduced is the handling of flow with evolving free boundaries on a curved surface. The use of this framework to model flow on moving and deforming surfaces is also introduced. Then, we present the application of this framework to solve fluid flow problems defined on surfaces numerically. In combination with a meshfree generalized finite difference method (GFDM), we introduce a strong form meshfree collocation scheme to solve the Navier–Stokes equations posed on manifolds. Benchmark examples are proposed to validate the Lagrangian framework and the surface Navier–Stokes equations with the presence of free boundaries.

KEYWORDS

free boundary, GFDM, fluid dynamics, Lagrangian, manifold, meshfree, Navier stokes, surface

1 | INTRODUCTION

Fluid flow on curved surfaces forms an important aspect in modeling dynamics on biomembranes and other interfacial transport processes,[1,2] in representing surfactant dynamics,[3] in thin film flow and lubrication,[4] and visualization,[5] among other applications.

One possibility for simulating flow on curved surfaces is the Navier–Stokes equations posed on manifolds. Several different and unequivalent formulations of the Navier–Stokes equations on manifolds have appeared in literature. This is mainly due to the different interpretations of the viscous stress tensor. This, in turn, arises from the different possibilities of the Laplace operator on vector fields on manifolds.[6] Using different arguments, recent work by References 6-9 clears this confusion to show that the physically accurate interpretation is one based on the so-called Boussinesq–Scriven surface stress tensor,[10,11] which has also been referred to as the deformation tensor.[12] For a comprehensive review of the different interpretations of the Navier–Stokes equations on manifolds, we refer to Reference 6 for Riemannian manifolds, and Reference 7 for the case of the surface Navier–Stokes equations, which we consider in the present work.

Most numerical methods to solve surface flow have used simplified models. Visualization driven work has mostly considered only inviscid flow.[13,14] A lot of work has been done for steady surface flow,[15,16] and surface flow on surfaces of specific shapes, such as spheres[17] or other radial manifolds.[15] Atmospheric dynamics work to model flow on the surface of the Earth solve two-dimensional simulations in the latitude-longitude frame.[18,19] Generalizing this to arbitrary geometries would involve parametrizing the surface, which could prove to be very challenging for complex or time-varying geometries, or for flow involving free boundaries.

Numerical methods to solve the full time-dependent incompressible surface Navier–Stokes equations has gained an increasing interest in the last few years.[20-22] Of particular interest in the present context is the recent work that recasts surface flow into a tangential differential calculus framework.[7,20] The advantage of this framework is that the equations are formulated entirely in Euclidean space, including the derivative computations, and thus they are easily implemented. As a result, it avoids the requirement of maintaining a parametrization of the surface, and the related issues with singularities arising in metrics. This framework is generally used to solve the underlying PDEs in a weak formulation. We adopt this framework, and use it to solve the PDEs in a strong form. Here, we focus on the use of this framework in an intrinsic setting, where the underlying PDEs are solved directly on the surface. Under the terminology of "calculus on surfaces without parametrization," similar methods have also been used in embedding methods which solve PDEs on a band around the surface, such as the closest point method.[23,24]

Solutions to the surface Navier–Stokes equations typically rely on finite element methods.[20-22,25] Here, we introduce the use of a meshfree method to solve this problem. In the last two decades, meshfree methods have started to become a popular alternative for conventional flow applications on volume domains, especially when the computational domain is complex or time-dependent. In such situations, mesh generation is very complex, and often needs several man hours. Furthermore, rapidly evolving domains result in the need of extensive remeshing which can be a very expensive portion of the simulation time. On the other hand, point cloud generation for meshfree methods has been largely automated, and is thus much faster for complex domains. Additionally, fixing distortion in point clouds can be done locally, and as a result, the meshfree equivalent of remeshing is also much cheaper. Many of these advantages carry over to the case of surface flow. The extensive need for expensive volume mesh generation for complex geometries is significantly reduced for the case of surface meshes. However, the issue of costly fixing of mesh distortion for time-dependent geometries is still present. This is relevant when the surface itself is evolving, and when there are evolving free boundaries within a surface. In our earlier work,[26] we have shown the advantage of meshfree Lagrangian frameworks for the former case to solve PDEs on evolving surfaces. Here, we explore their use in the latter in the context of flow on surfaces.

We introduce a novel meshfree Lagrangian framework to model flow with free boundaries on a moving curved surface. Meshfree points undergo Lagrangian motion with two velocities: the velocity of the fluid, and that of the surface. Another novelty in the present work is the inclusion of free boundaries in the context of Navier–Stokes on manifolds, which has not been done before to the best of our knowledge. We introduce for the first time the use of two different point clouds for surface flow. One point cloud to represent the moving fluid, and one for the surface constraining the flow. This makes it possible to have the fluid domain to be significantly smaller than the entire surface being considered, and the underlying PDEs modeling the system are only solved on the smaller fluid domain, thus, significantly reducing computational time. We emphasize that unlike a lot of particle based surface PDE literature, we only discretize the surface itself, and not a narrow volume around the surface.

This article is organized as follows. Section 2 introduces some basic notions of surface points clouds. Section 3 presents a novel Lagrangian framework to represent moving fluid points on a moving surface. This is then extended to moving free boundaries on a possibly moving surface. Section 4 provides an overview of surface differential operators and numerical methods to discretize them in a generalized finite difference method (GFDM) setting. The Navier–Stokes equations on manifolds are introduced in Section 5, along with a numerical scheme to solve them using a strong-form GFDM method. Section 6 then presents some numerical results for the surface Navier–Stokes equations, and the article is concluded with a brief discussion in Section 7.

2 | PRELIMINARIES

We consider a smooth orientable surface $M \subset \mathbb{R}^3$, which may or may not have boundaries. The notations used for surface point clouds in this article follow,[26] and those for the surface flow problem follow.[20] We start with the case of point cloud surfaces without free boundaries. The inclusion of free boundaries is done in Section 3.3.

The surface is discretized with a point cloud PC consisting of $N = N(t)$ irregularly spaced points. Approximations at each point are based on a compact support or neighborhood consisting of all points within a distance $h = h(\vec{x}, t)$ from it. An example of such a neighborhood is shown in Figure 1. All distances are computed as standard Euclidean distances, and not distances along the surface. h is referred to as the smoothing length or interaction radius. Efficient neighbor

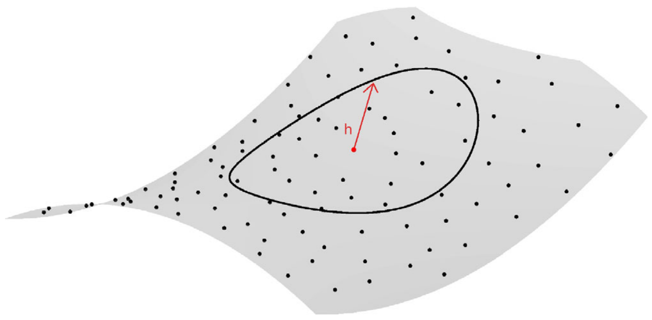


FIGURE 1 Surface neighborhood determined by distances in the embedding space, $\mathbb{R}^3$. The "circle" shown demarcating the neighborhood is the intersection of the sphere of radius h with the surface [Colour figure can be viewed at wileyonlinelibrary.com]

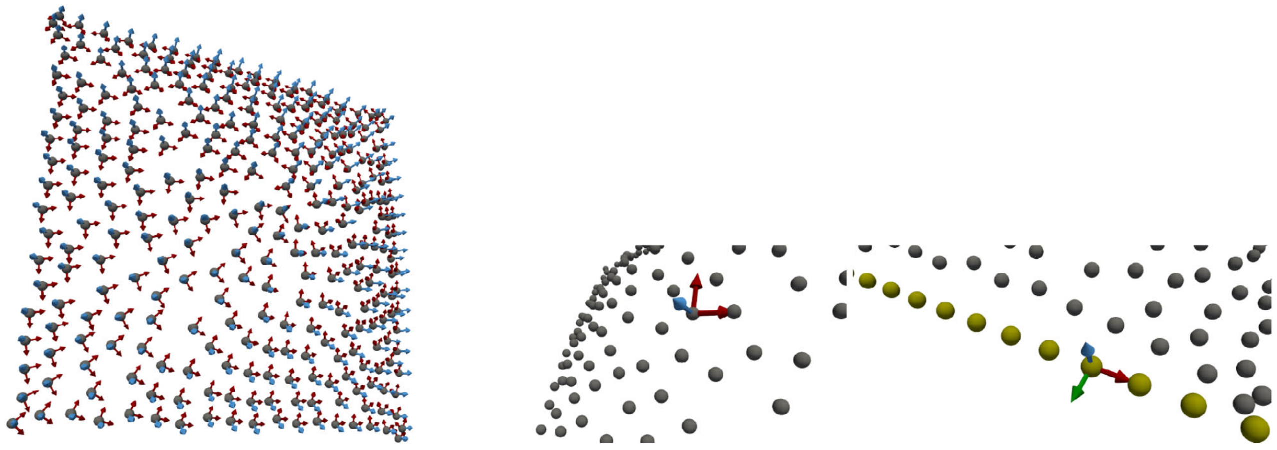


FIGURE 2 Normal and tangents to the surface. The center figure shows the case of an interior point. The right figure shows the same for a boundary point. The surface normal is shown in blue, the boundary normal is shown in green, and the tangent(s) in red [Colour figure can be viewed at wileyonlinelibrary.com]

searching algorithms for volume-domain meshfree methods (for example, References 27-29) directly carry over to the surface case here.

Inter-particle distance ranges from $r_{min}h$ to $r_{max}h$, for fixed simulation-independent parameters r_{min} and r_{max}. The parameters used here are adopted from conventions in volumetric meshfree flow simulations.[28,30,31] We set $r_{min} = 0.2$ and $r_{max} = 0.45$, as has been done for meshfree surfaces in References 26 and 32. This results in about $15 - 20$ points in each neighborhood. This set up leads to h being a measure of not just the support size, but also the discretization size. On a moving point cloud, these minimum and maximum inter-particle distances are ensured by inserting and removing points. More details of this are explained in Section 3.2.

Each point is equipped with a unit surface normal $\vec{n}$. It is ensured that the normals are defined such that the discrete surface is always oriented. Here, we assume that an initial discrete orientation is available. As the point cloud moves, the updated new normals are kept consistent with the old ones, to ensure a discrete oriented surface at all times. Boundary points further have an outward pointing unit boundary normal $\vec{v}$, which is conormal to $\vec{n}$. This is illustrated in Figure 2. Details of computation of the normals on moving point clouds follow the procedure laid out in Reference 26.

3 | THE LAGRANGIAN FRAMEWORK

In our earlier work,[26] we introduced a meshfree Lagrangian framework to model evolving-in-time surfaces numerically, including meshfree contact handling algorithms. Here, we extend that framework to model fluid flow within a (possibly moving) surface. We propose to do this using Lagrangian particles which move along the given surface. This introduces several special features, as we explain below.

We emphasize that in the Lagrangian framework that follows, we do not maintain any correlation to the domain at the initial state. At a specific time step, all approximations are performed on the point cloud at that time step directly, without mapping it to the initial configuration.

For the case of fluid motion on a moving surface, two advection velocities are relevant. The velocity $\vec{v}$ of the fluid moving on the surface, and the velocity $\vec{w}$ of the surface itself. Thus, a particle on the surface moves with a velocity of $\vec{v} + \vec{w}$, and the movement of the point cloud is given by

$$\frac{d\vec{x}}{dt} = \vec{v} + \vec{w}. \tag{1}$$

We note that $\vec{v}$ is necessarily tangential to the surface, that is, $\vec{v} \cdot \vec{n} = 0$. Whereas $\vec{w}$ is not constrained and can have components both normal and tangential to the surface. In Reference 26, we only handled the special case of $\vec{v} = 0$ and $\vec{w} \neq 0$, which was used to solve PDEs on evolving surfaces, in the absence of any flow on the surface. Here, we deal with the general case of $\vec{v} \neq 0$ and $\vec{w} \neq 0$, which allows motion relative to the moving surface. We also introduce the possibility of a fluid domain that distorts and evolves on the surface, and does not cover the entire surface.

We start by restricting ourselves to flow without free boundaries, handling of which is introduced in Section 3.3.

3.1 | The Lagrangian movement of points

On the numerical level, we split the movement into two steps. The first is based on the fluid velocity, and the second one is based on the surface velocity. Each step involves a second order in time movement. We note that most Lagrangian methods only use a first order movement, which has been shown to be extremely inaccurate in capturing rotational components of motion.[33] For time integration between time levels t^n and t^{n+1} with the velocity $\vec{v}$, we get

$$\Delta\vec{x}_1 = \vec{v}^{(n)}\Delta t + \frac{1}{2}\frac{\vec{v}^{(n)} - \vec{v}^{(n-1)}}{\Delta t_0}(\Delta t)^2, \tag{2}$$

where bracketed superscripts indicate the time level, $\Delta t = t^{n+1} - t^n$ is the current time step, and $\Delta t_0 = t^n - t^{n-1}$ is the previous time step. Here, we assume that $\vec{v}^{(n+1)}$ is unknown, as is the case if the movement step is done before computing the new velocity. If $\vec{v}^{(n+1)}$ is known during the movement step, $\vec{v}^{(n)}$ and $\vec{v}^{(n-1)}$ can be replaced by $\vec{v}^{(n+1)}$ and $\vec{v}^{(n)}$, respectively, in Equation (2).

Since the velocity $\vec{v}$ lies in the tangent space, an exact integration along the velocity streamlines would result in particles still being on the surface. However, the discrete movement based on Equation (2) can cause points to be moved off the surface to a distance of the order of $\mathcal{O}(\Delta t^3)$. To avoid accumulation of these numerical errors in this movement step, we follow it with a projection of the point cloud back to the surface given by the point cloud at t^n, before the movement. Details of the projection step follow that in Reference 26.

Equation (2) and the projection step are then followed by movement according to the surface velocity $\vec{w}$

$$\Delta\vec{x}_2 = \vec{w}^{(n)}\Delta t + \frac{1}{2}\frac{\vec{w}^{(n)} - \vec{w}^{(n-1)}}{\Delta t_0}(\Delta t)^2. \tag{3}$$

Thus, the point cloud at the new time level is given by

$$\vec{x}^{(n+1)} = \tilde{P}_{(n)}\left(\vec{x}^{(n)} + \Delta\vec{x}_1\right) + \Delta\vec{x}_2, \tag{4}$$

for the projection step $\tilde{P}_{(n)}$ to the point cloud at t^n.

We perform a second order movement step with each velocity to more accurately capture the movement. However, it must be noted that the splitting explained above is only of first order.

3.2 | Fixing distortion

The ideas of fixing distortion in surface point clouds directly carry over from Reference 26. We present a brief description here for the sake of completeness. As introduced in Section 2, to maintain regularity of the point cloud, it is ensured that no two points come closer than a distance of $r_{min}h$, and that there is no hole in the point cloud of radius $r_{max}h$ that contains

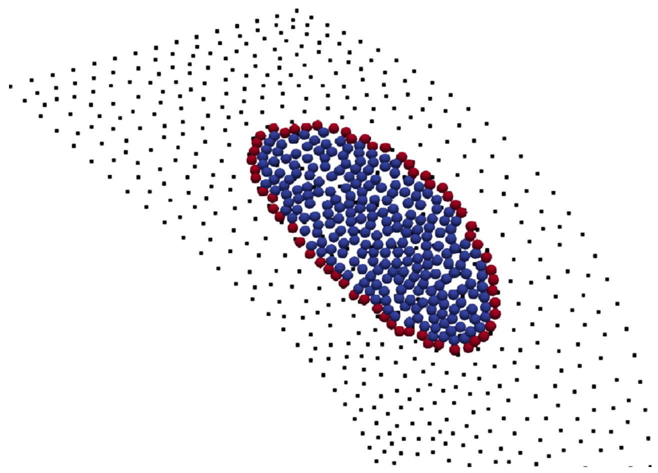


FIGURE 3 For simulating free boundaries in surface flow, two point clouds are maintained. The main point cloud representing the fluid is shown in spherical glyphs: blue for interior points, and red for free boundary points. The background point cloud representing the shape of the overall computational domain is shown with black points [Colour figure can be viewed at wileyonlinelibrary.com]

no points. As the point cloud is moved in the Lagrangian sense, this regularity condition can be violated, resulting in a distorted point cloud. Detecting and fixing these violations is always done entirely locally, unlike the case of fixing a distorted mesh which could require a global remeshing. Violations to the closeness condition can be detected trivially with distance computations within each neighborhood. If two points come closer than than $r_{min}h$ apart, they are merged into one point at the center location, with all physical properties interpolated at the new location. Detection of formed holes in the point cloud are done by checking the circumradius of triangles formed by local tessellations within a compact support. Since r_{min} and r_{max} are fixed parameters, changing h automatically triggers the addition and merging algorithms, and can thus be used to introduce adaptive refinement. Detection of holes and close points is done immediately after the Lagrangian movement step. First, the distortion fixing is done specific to boundary points, after which it is done for interior points. We refer to section 3 of Reference 26 for more details. Fixing distortion in this manner ensures that there is no need for any artificial movement step to maintain point cloud regularity, as is the case in many Lagrangian particle methods like SPH (for example, Reference 34).

3.3 | Lagrangian framework with free boundaries

We now extend the above notions of flow on point cloud surfaces to include free boundaries. In the above case, the fluid region being simulated occupies the entire computational domain, that is, PC describes not just the fluid, but also the surface that the fluid is restricted to. However, for the case of free boundaries, this is no longer true. There, the fluid occupies only a part of the surface domain. Thus, there is an additional need to represent the surface which constrains the fluid movement.

This needed surface discretization could be done in multiple ways. One possible method is to maintain a surface mesh, over which the particles move. This would represent the extension of particle-in-cell methods[35] to surface domains. This approach can be very useful if the movement of the surface is restricted to rigid body motion. However, for time-evolving surfaces undergoing arbitrary movements, the movement of the mesh could cause distortion which may require expensive remeshing to fix. Thus, the advantages of point clouds again become relevant, and we leverage them by using a point cloud even for the surface discretization. To incorporate free boundaries, we introduce the use of two different point clouds. A main point cloud PC representing the moving fluid, which moves over a background point cloud PC^{back} representing the surface. The background point cloud is only used to constrain the movement of the main point cloud. Underlying PDEs are only solved on PC and not on PC^{back}. The use of two points clouds in such a manner is illustrated in Figure 3.

Notations for the main point cloud PC follow that introduced in Section 2 with $N = N(t)$ points, and a smoothing length of $h = h(\vec{x}, t)$. The background point cloud PC^{back} to define the surface contains $N^{\text{back}} = N^{\text{back}}(t)$ points, and a smoothing length of $h^{\text{back}} = h^{\text{back}}(\vec{x}, t)$.

PC^{back} is moved only with the surface velocity $\vec{w}$, analogously to Equation (3). If the surface is stationary ($\vec{w} = 0$), the background point cloud does not move, and thus does not change in time. The main point cloud is moved with both $\vec{v}$ and $\vec{w}$, as described in Section 3.1, with the only difference being the projection. In presence of the background point cloud, the projection after the move with the fluid velocity is done to the surface given by PC^{back}. Thus the movement can

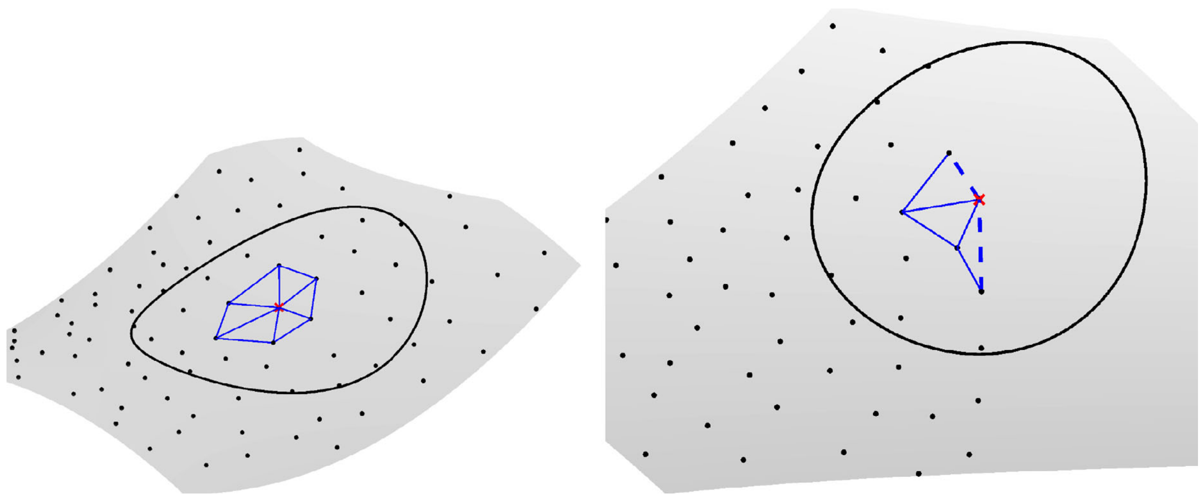


FIGURE 4 Local tessellation used for free boundary detection and fixing distortion. Interior point (left). Boundary point (right) with the open edges marked with thicker dashed lines [Colour figure can be viewed at wileyonlinelibrary.com]

be summarized as follows. First, the locations of the main fluid point cloud is updated according to Equation (2). This is then projected to the surface given by the background point cloud. Then both point clouds are moved with the surface velocity according to Equation (3).

Distortion needs to be fixed for both point clouds upon movement. For both, this follows the same procedure outlined in Section 3.2.

For ease of visualization, in some of the figures to follow, we represent the background point cloud with a tessellation of it. Henceforth, unless mentioned otherwise, the term point cloud will refer to the main point cloud representing the fluid. We emphasize here that the background point cloud is only maintained for applications involving free boundaries. Without free boundaries within the surface, there is no need to maintain PC^{back}, as the main point cloud sufficiently describes both the fluid and the underlying surface.

3.4 | Free boundary detection

An important step in simulating a fluid with free boundaries is the detection of the points that represent the free boundary. Initially, the prescription of free boundary points is straight forward and can be done directly based on the initial fluid configuration. However, extra care needs to be taken as the free boundary evolves, to handle events such as the fluid domain hitting a fixed boundary, where the distinction between free and fixed boundary points needs to be made for the correct application of the boundary conditions. An important use case is when a topological change occurs within the fluid domain, for example when a fluid droplet splits or merges with another.

Several approaches have been proposed for detecting free surfaces in volumetric flow (e.g., References 36-38), and those can be extended to the present context. We choose to reuse the local tessellations which are already being used to fix distortion. For each point, a "1-ring" of triangles is determined by tessellating its neighborhood, as shown in Figure 4. We emphasize here that these "1-rings" of triangles at every point need not stitch together to form a global mesh on the point cloud.[31] Furthermore, no restriction on the quality of these triangles is enforced, nor are any approximations done based on these triangles. Once the tessellation is computed, a point is labeled to be part of the free boundary if there are open edges incident on the point. An example of this is illustrated in Figure 4. There is an additional possibility of a fixed boundary point for points that lie on boundaries of the surface.

In computing the local tessellations, an important heuristic that is followed is to avoid forming a triangle between three boundary points of the previous time step. For boundary point detection, this is needed to prevent free boundary points from being incorrectly labeled as interior points. Since this tessellation is also used to find holes in the point cloud, avoiding triangles between boundary points of the previous time step ensures that regions outside the fluid domain are not incorrectly identified as holes, and no points are added outside the actual domain of computation. A test case which highlights the use of these algorithms is presented later, in Section 6.3.

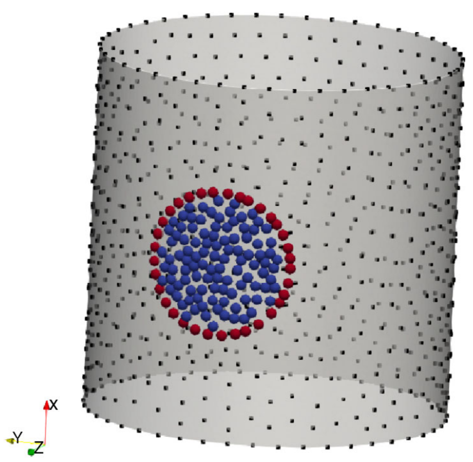


FIGURE 5 Initial condition for testing the Lagrangian motion of a free fluid droplet on a cylinder surface. The main fluid point cloud is shown with spherical glyphs: blue for interior points, and red for free boundary points. The background point cloud for the shape of the overall computational domain is shown with black points. To enhance visualization, we have added a gray shell by tessellating the background point cloud [Colour figure can be viewed at wileyonlinelibrary.com]

3.5 | Contact handling

The contact handling algorithms between meshfree surface point clouds[26] directly carry over to the present case. These are relevant when free boundaries are present, especially when the topology of the fluid domain changes. For example, when a fluid region splits into multiple disconnected regions, or when movement of a free boundary causes a “hole” to be formed in the middle of the fluid.

3.6 | Numerical examples

To verify the Lagrangian framework introduced, we consider a few examples of a fluid with free boundaries moving on an evolving surface, with prescribed velocities $\vec{v}$ and $\vec{w}$. We emphasize that only the ODE for the Lagrangian movement of the point cloud is solved here.

An important point to note is the time step requirement introduced due to the Lagrangian movement. We have

$$\Delta t \leq C_1 \frac{h}{||\vec{v}+\vec{w}||_\infty}, \quad \text{and} \quad \Delta t \leq C_2 \frac{h^{\text{back}}}{||\vec{w}||_\infty}, \tag{5}$$

with CFL numbers $C_1, C_2 \leq 1$. If h^{back} is very fine, the time step restriction due to the movement of the background point may be much stricter than that of the fluid domain movement. In such a case, two separate time steps can be maintained for the two Lagrangian steps, with a substepping procedure where multiple background point cloud steps are done for a single fluid point cloud step. This is analogous to similar substepping procedures done in multi-phase or FSI problems (see, for example, Reference 39). Furthermore, we note that there is no stability restriction on h due to the resolution of h^{back}, or vice versa.

For all simulation times mentioned in this section, and later in Section 6, we note the following: All simulations were carried out serially using a Intel XeonE5-2670 CPU rated at 2.60GHz.

3.6.1 | Moving droplet on a fixed cylinder

Consider a cylinder with axis along the x direction: $y^2 + z^2 = r^2$, limited by $-0.5 \leq x \leq 0.5$. Here, we set $r = 0.5$. An initial fluid droplet is taken as the intersection of the cylinder with the sphere $x^2 + y^2 + (z-0.5)^2 \leq 1$, as shown in Figure 5.

The surface is stationary $\vec{w} = 0$, and the droplet moves with a prescribed velocity of

$$\vec{v} = (0.2\sin(2\pi t), -\pi z, \pi y)^T. \tag{6}$$

Note that this velocity is tangential to the surface. The droplet moves in a sinusoidal motion while rotating along the cylinder. At every integer multiple of $t = 2$, the droplet returns to its original location. We measure the error of the location

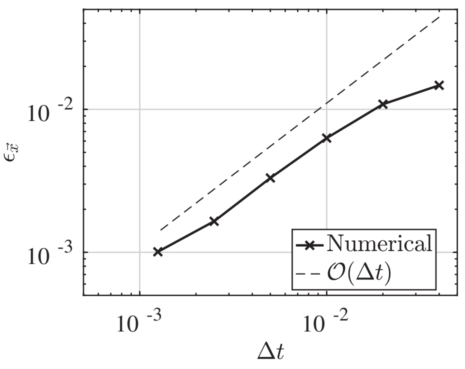


FIGURE 6 Error in location of the fluid droplet moving on a fixed cylinder, with varying time step and a fixed resolution of both fluid and background point clouds

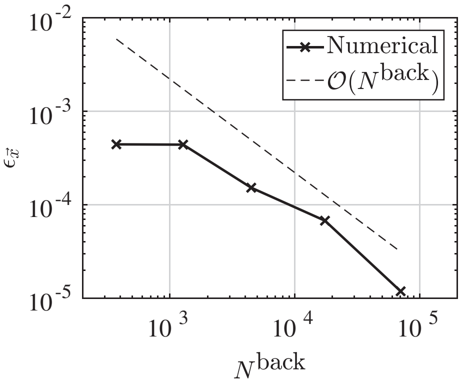


FIGURE 7 Error in location of the fluid droplet moving on a fixed cylinder, with varying background point cloud resolution and a fixed time step and fixed resolution of the fluid point cloud. The corresponding computation time required are collected in Table 1

of the fluid domain as

$$\epsilon_{\vec{x}} = \|\vec{x}_{num} - \vec{x}_{exact}\|, \tag{7}$$

where $\vec{x}_{num}$ is the location of the geometric center of the fluid point cloud, and $\vec{x}_{exact}$ is the analytical path traveled by the center of the initial fluid domain. For coarse point clouds with $N = 138$, and $N^{\text{back}} = 1574$ points initially, the convergence of errors at $t = 2$ with a decreasing time step is shown in Figure 6.

Only linear convergence with Δt is seen. This could have multiple reasons. First, the splitting of the movement is of first order. Second, as also observed in Reference 26, the largest error in the movement comes at the very first time step, where a first order move has to be done since no previous velocity is available.

The errors considered above are caused by the Lagrangian movement. To be precise, since $\vec{w} = 0$ in this example, the convergence of errors shown in Figure 6 represents the convergence of errors in the first step of the movement process, Equation (2). We now take a look at the error caused by the projection to the background point cloud, as explained in Section 3.1. For this, we consider the same example of the moving fluid drop on a stationary cylinder, this time by varying the background point cloud resolution. A fixed time step of $\Delta t = 0.01$ is used, with a finer fluid point cloud than above, $N = 343$ points initially. The resolution of the background point N^{back} representing the cylinder surface is varied from $\mathcal{O}(10^2)$ to $\mathcal{O}(10^5)$, where the number of points is approximately quadrupled in each successive resolution. The resulting errors are shown in Figure 7. The exact number of points in the different background point clouds considered are tabulated in Table 1, along the total simulation time. We note here that the total computation times mentioned includes both the pre-processing time of the seeding the point clouds, and the post-processing time of saving results and error computations.

As the background point cloud is made finer, it forms a better representation of the cylinder surface. Since the analytical solution of the location of the fluid domain assumes a perfect cylinder, we observe significantly lesser errors with the finer background point clouds. We emphasize here the surface domain is given entirely by the background point cloud. As the fluid point cloud moves along the background point cloud, one way of checking the accuracy of the surface representation is by checking the computed normals. Analytically for a point at $\vec{x} = (x, y, z)^T$, the outward pointing normal should be $\vec{n}_{\text{exact}} = \frac{1}{r}(0, y, z)^T$. Measuring the error as $\|\vec{n} - \vec{n}_{\text{exact}}\|$, the maximum error across all fluid points at $t = 2$ is 1.97×10^{-2} for the coarsest point, and 1.03×10^{-4} for the finest point cloud, which shows the effect of the background point cloud resolution. We note here that the resolution of the fluid point cloud is the same in each of the cases.

TABLE 1 Moving fluid droplet on a fixed cylinder: Total time taken for the simulations for the different background point cloud resolutions considered

N^{back}	Simulation time (s)
372	4
1275	5
4473	7
17436	15
70377	47

Note: The corresponding errors are plotted in Figure 7.

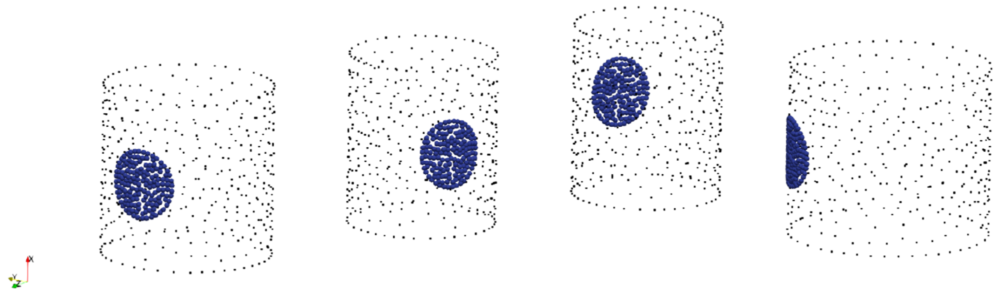
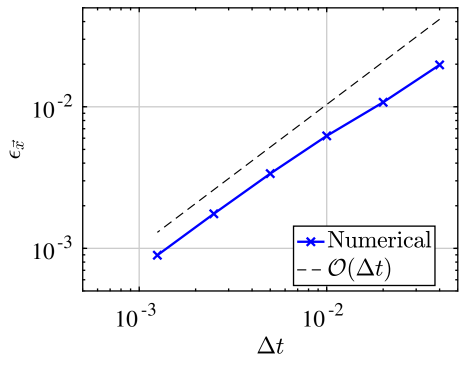


FIGURE 8 Evolution of the surface domain and the fluid for the moving droplet on a moving cylinder test case. $t = 0$, $t = 0.32$, $t = 1.32$, and $t = 1.76$. The fluid is shown with blue spherical glyphs. And the surface is shown with black points. Note that the scale and coordinate limits are the same in each of the images [Colour figure can be viewed at wileyonlinelibrary.com]

FIGURE 9 Error in location of the fluid droplet moving on a moving cylinder. Errors observed are very similar to those of the fixed cylinder example, shown in Figure 6 [Colour figure can be viewed at wileyonlinelibrary.com]



3.6.2 | Moving droplet on a moving cylinder

We now extend the above example to a moving cylinder. The same initial cylinder and fluid droplet is considered as above. In this case, the cylinder is moving and deforming with

$$\vec{w} = (0.5\sin(\pi t), 0, 0)^T - 0.5\cos(2\pi t)\vec{n}, \tag{8}$$

where $\vec{n}$ is the outward pointing unit normal of the cylinder surface. The first term causes a translation motion of the cylinder in the direction of the axis, while the second causes the cylinder to expand and contract. The normal component of the surface velocity can be integrated to obtain the analytical radius of the cylinder $r(t) = 0.5 - 0.5\frac{\sin(2\pi t)}{2\pi}$. The same velocity $\vec{v}$ is taken as in the previous example. Note that this velocity still remains tangential to the surface throughout the temporal domain. Snapshots of the evolution of the domain are shown in Figure 8.

The same initial point clouds are used as in the previous example. At $t = 2$, the cylinder domain, and the fluid droplet both return to its initial shape and position. Errors are thus measured as in the previous example, and are shown in Figure 9. The errors do not differ greatly from the fixed cylinder case of the previous example.

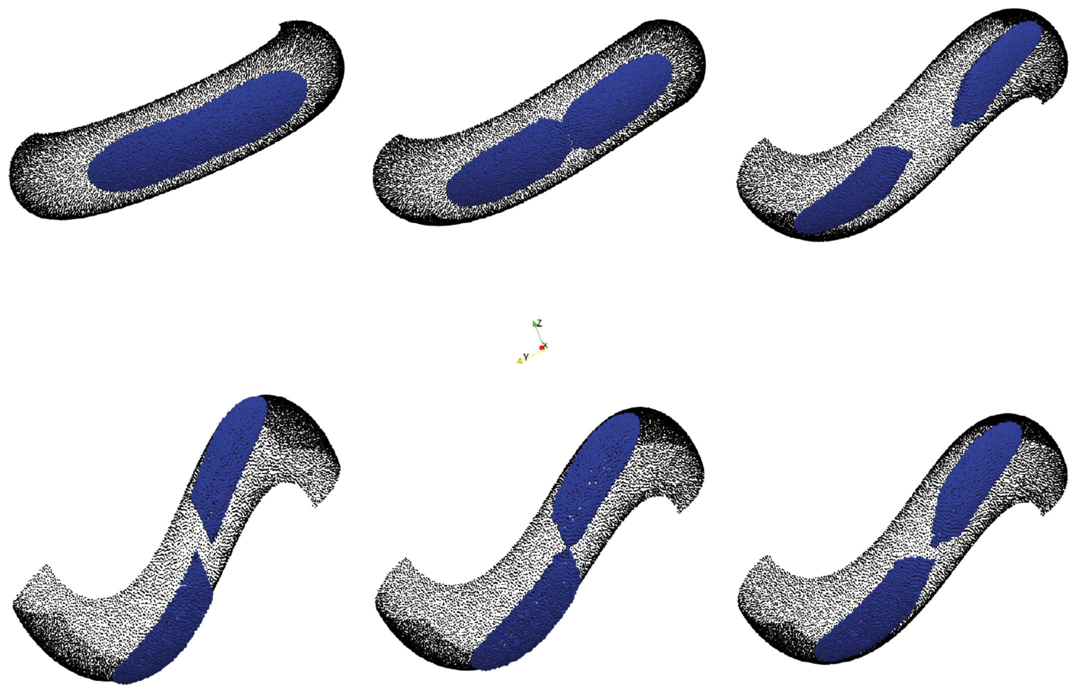


FIGURE 10 Fluid droplet undergoing a change of topology on a distorting surface. Clockwise from top left: $t = 0$ (top left), $t = 0.18$ (top center), $t = 0.525$ (top right), $t = 0.72$ (bottom right), $t = 0.855$ (bottom center), and $t = 1.14$ (bottom left). The fluid is shown with blue spherical glyphs, and the surface is shown with black points [Colour figure can be viewed at wileyonlinelibrary.com]

3.6.3 | Movement with distortion

We now consider a case where the fluid domain not just deforms, but also undergoes a topological change. The deformation of the surface itself illustrates the benefit of a background point cloud, rather than a background mesh.

For the surface movement, we consider a case similar to that proposed in.[26] The surface domain is a quarter of a torus with major radius 0.75 and minor radius 0.25, as shown in the first image of Figure 10. The surface velocity is given by

$$\vec{w} = (-yz, 0, xz)^T, \tag{9}$$

which causes an overall twisting motion in two different directions. A large fluid droplet is considered at the center of the surface, as shown in the first image of Figure 10. The fluid velocity is given by

$$\vec{v} = \boldsymbol{P}(0, \operatorname{sign}(y)\sin(2\pi t), 0)^T, \tag{10}$$

where $\boldsymbol{P}$ projects the vectors that it acts upon to the tangent space of the surface domain. A more detailed explanation of $\boldsymbol{P}$ is given in the next section. This velocity causes the droplet to split into two parts as it starts moving.

Finer point clouds are used in comparison to the previous examples to accurately capture the deformation. The simulations start with $N = 1902$ points and $N^{\text{back}} = 10,499$ points. The evolution of the fluid and surface domains are shown in Figure 10. This example illustrates the capability of the present method to handle complex geometries and arbitrary motion.

4 | SURFACE DERIVATIVES

In this section, we present a concise overview of some well-known definitions of surface-intrinsic differential operators (see, e.g., References 23,40-43), and of numerical approximations of surface differential operators using a meshfree GFDM.[32]

4.1 | Differential operators on surfaces

Consider an open subset Ω of $\mathbb{R}^3$ containing the surface M. Thus, we have $M \subset \Omega \subset \mathbb{R}^3$. For a scalar-valued function defined on the surface $f : M \to \mathbb{R}$, an extension of f to Ω is defined as $\hat{f} : \Omega \to \mathbb{R}$ such that f and $\hat{f}$ agree on M. Note that the extension is not unique. The so-called tangential differential calculus framework provides a method to define the derivatives of f along the surface M in relation to the conventional derivatives of $\hat{f}$ on Ω. This definition is independent of which extension is used, with the assumption that only "smooth enough" extensions are considered. In this framework, the surface gradient of f is defined as

$$\nabla_M f = \boldsymbol{P} \nabla \hat{f}, \tag{11}$$

with the projection operator (to the tangent plane)

$$\boldsymbol{P} = \boldsymbol{I} - \vec{n}\vec{n}^T. \tag{12}$$

For a vector valued function $\vec{v} : M \to \mathbb{R}^3$, its extension $\hat{\vec{v}} : \Omega \to \mathbb{R}^3$ is defined component wise. Thus, if $\vec{v} = (u, v, w)^T$, then $\hat{\vec{v}} = (\hat{u}, \hat{v}, \hat{w})^T$. The surface divergence of $\vec{v}$ is defined as

$$\nabla_M \cdot \vec{v} = (\boldsymbol{P}\nabla) \cdot \hat{\vec{v}} \,. \tag{13}$$

Combining Equations (11) and (13) gives the definition of the surface Laplacian or Laplace–Beltrami of f

$$\Delta_M f = \nabla_M \cdot \nabla_M f, \tag{14}$$

$$= (\boldsymbol{P}\nabla) \cdot \left(\boldsymbol{P}\nabla \hat{f}\right). \tag{15}$$

The directional surface gradient of a vector valued function is given by

$$\nabla_M^{\text{dir}} \vec{v} = \left(\nabla \hat{\vec{v}}\right) \boldsymbol{P}, \tag{16}$$

$$= (\nabla_M u \,, \nabla_M v \,, \nabla_M w)^T. \tag{17}$$

Furthermore, the covariant surface gradient of a vector valued function is defined as the projection of the directional gradient to the tangent plane, and is given by

$$\nabla_M^{\text{cov}} \vec{v} = \boldsymbol{P} \nabla_M^{\text{dir}} \vec{v}, \tag{18}$$

$$= \boldsymbol{P} \nabla \hat{\vec{v}} \boldsymbol{P}. \tag{19}$$

For the divergence of tensor-valued functions, we follow the convention of column-wise divergences.

The advantages of this framework of defining surface operators are that it avoids differential geometry complexities and avoids the need to parametrize the surface. Furthermore, it provides a straight forward interpretation for strong form PDE solvers, which shall be used in the present work. This comes at the cost of having to solve an extra unknown for vector fields. For example, if the surface can be parametrized, the velocity field only needs two components to describe it entirely. However, solving it in a global Euclidean setting requires three velocity components. For more details on differential operators on surface, we refer to References 20,32,40,41.

4.2 | Numerical surface derivatives

GFDMs are strong form collocation methods to discretize differential operators. They have been shown to be robust methods, and have been widely applied to solve PDEs on volume domains (for example, References 28,44-47). In the present context, we use GFDMs to discretize the surface differential operators. The general idea of computing numerical derivatives that we use here is to project neighboring points to a tangent space, compute regular volume 2D derivatives in the tangent space, and rotate them back to the surface to obtain surface derivative stencils. This notion was introduced in Reference 48 for the surface Laplacian on meshes, and was extended in our earlier work[32] for the use on general differential operators on point clouds. For the sake of completeness, we give an brief overview below.

We note that similar notions of using GFDMs for discretizing surface derivatives have been done in, for example, References 16 and 49, based on approximations of surface metric tensors. In contrast, the approach used here, based on Reference 32, does not rely on surface-based metrics. Instead, differential operators are discretized as a Euclidean problem on the tangent space in such a manner that techniques from volume-based numerical methods can be directly transferred over to surfaces.

The normal extension of a function is the extension which satisfies $\vec{n} \cdot \nabla \hat{f} \equiv 0$. For normally extended functions, Equations (11), (13), and (15) reduce to (see Reference 32)

$$\nabla_M f = \nabla \hat{f}, \tag{20}$$

$$\nabla_M \cdot \vec{v} = \nabla \cdot \vec{\hat{v}}, \tag{21}$$

$$\Delta_M f = \Delta \hat{f}. \tag{22}$$

Thus, the surface gradient, divergence, and Laplacian can be computed by approximating the corresponding conventional derivative of the normal extension. We use this to numerically compute the surface derivatives.

Local normal extensions are used to project neighboring points to the tangent space. For the surface gradient, at a numerical point i at location $\vec{x}_i$ on the surface, we compute the tangential components of $\nabla \hat{f}$ on the tangential plane T_i spanned by the tangents $\vec{t}_{1,i}$ and $\vec{t}_{2,i}$. For this, the neighboring points $j \in S_i$ are projected along the normal $\vec{n}_i$ to the locations j_{T_i}. First, we approximate the numerical gradient on the tangent plane ∇_T

$$\nabla_T \hat{f} \approx \tilde{\nabla}_T \hat{f} = \left(\sum_{j \in S_i} c_{ij_T}^{t_1} \hat{f}_{j_T}, \ \sum_{j \in S_i} c_{ij_T}^{t_2} \hat{f}_{j_T} \right)^T, \tag{23}$$

where the overhead $\sim$ indicates the numerical operator. Stencil coefficients c_{ij_T} are computed using a GFDM approach,[45] which employs a least squares procedure shown below. Monomials up to a certain order are differentiated exactly

$$\sum_{j \in S_i} c_{ij_T}^{t_k} m_{j_T} = \frac{\partial}{\partial t_k} m(\vec{x}_i) \quad \forall m \in \mathcal{P}_T, \tag{24}$$

$$\min J_i = \sum_{j \in S_i} \left(\frac{c_{ij_T}^{t_k}}{W_{ij_T}} \right)^2, \tag{25}$$

for $k = 1, 2$, where $\mathcal{P}_T$ is the set of monomials, taken up to order 2 in the present work, in $\vec{t}_{1,i}$ and $\vec{t}_{2,i}$ on the tangent plane. Now, the numerical surface gradient is defined as

$$\tilde{\nabla}_{M,i} f = \left(\sum_{j \in S_i} c_{ij}^{M,x} f_j, \ \sum_{j \in S_i} c_{ij}^{M,y} f_j, \ \sum_{j \in S_i} c_{ij}^{M,z} f_j \right)^T, \tag{26}$$

where $c_{ij}^{M,x}$ are the stencil coefficients for the surface gradient in the x direction, and similarly for the other directions. These stencil coefficients are given by

$$\left(c_{ij}^{M,x}, \ c_{ij}^{M,y}, \ c_{ij}^{M,z} \right)^T = R^T \left(c_{ij}^{t_1}, \ c_{ij}^{t_2}, \ c_{ij}^{n} \right)^T, \tag{27}$$

with $c_{ij}^n = 0$ and a rotation matrix R

$$R^T = \begin{pmatrix} \vec{t}_1 & \vec{t}_2 & \vec{n} \end{pmatrix}, \tag{28}$$

The numerical gradient stencils also provide the numerical divergence. A similar procedure is followed for the numerical surface Laplacian defined as

$$\tilde{\Delta}_M f = \sum_{j \in S_i} c_{ij}^{\Delta_M} f_j. \tag{29}$$

Rotational invariance of the Laplace operator means that the surface Laplacian is directly given by the tangent place Laplacian $c_{ij}^{\Delta_M} = c_{ij}^{\Delta_T}$, which is computed by

$$\sum_{j \in S_i} c_{ij_T}^{\Delta_T} m_{j_T} = \Delta_T m(\vec{x}_i) \qquad \forall m \in \mathcal{P}_T, \tag{30}$$

$$\min J_i = \sum_{j \in S_i} \left(\frac{c_{ij_T}^{\Delta_T}}{W_{ij_T}} \right)^2. \tag{31}$$

We do not discretize the covariant vector derivative directly. Only surface gradients and Laplacians are the computed numerical differential operators. For vector fields, the application of the numerical gradient operator gives us the directional vector gradient. The numerical covariant vector gradient can then be computed using the numerical projection operator, according to Equation (18). For more details of both the theory and implementation, including the necessary stabilization of the surface Laplacian defined above, we refer to Reference 32.

5 | NAVIER–STOKES EQUATIONS ON MANIFOLDS

We consider the incompressible Navier–Stokes equations posed on a curved surface.[7,9,20] The surface stress tensor is given by

$$\boldsymbol{S} = \eta \left[\left(\nabla_M^{\text{cov}} \vec{v} \right) + \left(\nabla_M^{\text{cov}} \vec{v} \right)^T \right], \tag{32}$$

for (dynamic) viscosity η. Using this, the incompressible surface Navier–Stokes equations read as

$$\frac{\partial \vec{v}}{\partial t} + \left(\vec{v} \cdot \nabla_M^{\text{cov}} \right) \vec{v} + \left(\vec{w} \cdot \nabla \right) \vec{v} = \frac{1}{\rho} \boldsymbol{P} \nabla_M \cdot \boldsymbol{S} - \frac{1}{\rho} \nabla_M p + \vec{g}, \tag{33}$$

$$\nabla_M \cdot \vec{v} = 0, \tag{34}$$

$$\vec{v} \cdot \vec{n} = 0, \tag{35}$$

with fluid velocity $\vec{v}$ on the surface, surface velocity $\vec{w}$, pressure p, and gravity and body forces term $\vec{g}$. Equation (33) arises from momentum conservation, Equation (34) is the incompressibility constraint, and Equation (35) indicates that the velocity field lies entirely in the tangent bundle, with no normal component. To simplify the model, inertial source terms arising from the movement of the surface are ignored here.

5.1 | Numerical scheme

An important point of consideration in designing the numerical scheme is the method to ensure that the velocity field lies in the tangent bundle of the surface. One approach considered in literature (only for inviscid flow) is to first solve the conservation of momentum and mass equations, Equations (33) and (34), without considering the tangential velocity constraint. The resultant velocity field is then projected to the tangent space of the surface. This has been done in References 13 and 14. While it serves their purpose of producing fast solutions to create animations very well, it produces physically inaccurate solutions, as the incompressibility constraint is no longer satisfied. Another approach is to add the tangential velocity constraint as a Lagrange multiplier to the implicit velocity system,[20] which adds the need to control another parameter in the scheme. Here, we consider a more direct approach of a velocity-pressure segregated scheme, such that the design of the splitting ensures the velocity is tangential to the surface.

We follow a Chorin projection method[50] which has been widely used to solve incompressible and weakly compressible volumetric flow, and modify it for the needs of the surface Navier–Stokes equations. First, we split the viscous term into the projection of the component wise Laplacian, which resembles the conventional viscous term in volumetric Navier–Stokes equations, and extra rotational components

$$\frac{1}{\rho}\boldsymbol{P}\nabla_M\cdot\boldsymbol{S}\left(\vec{v}\right)=\left(\frac{\eta}{\rho}\boldsymbol{P}\Delta_M\vec{v}\right)+\left(\frac{1}{\rho}\boldsymbol{P}\nabla_M\cdot\boldsymbol{S}(\vec{v})-\frac{\eta}{\rho}\boldsymbol{P}\Delta_M\vec{v}\right). \tag{36}$$

Note that we assume η to be constant in space. We start the scheme with a Lagrangian movement step, Equation (4), as explained in Sections 3.1 and 3.3. This is followed by semi-implicit intermediate velocity solve, which treats the first term in Equation (36) implicitly and second one explicitly

$$\frac{\vec{v}^{*}-\vec{v}^{(n)}}{\Delta t}-\frac{\eta}{\rho}\boldsymbol{P}\Delta_M\vec{v}^{*}=\frac{1}{\rho}\boldsymbol{P}\nabla_M\cdot\boldsymbol{S}(\vec{v}^{(n)})-\frac{\eta}{\rho}\boldsymbol{P}\Delta_M\vec{v}^{(n)}-\frac{1}{\rho}\nabla_M p^{(n)}+\vec{g}. \tag{37}$$

In projection methods for volume domain flows, this intermediate velocity would then be projected to a divergence free space with the help of a correction pressure. Here, we first project the intermediate velocity to the tangent plane

$$\vec{v}^{**}=\boldsymbol{P}\vec{v}^{*}. \tag{38}$$

This velocity is now projected to a surface divergence free space

$$\vec{v}^{(n+1)}=\vec{v}^{**}-\frac{\Delta t}{\rho}\nabla_M p_{\text{corr}}, \tag{39}$$

where the correction pressure p_{corr} is first computed by applying the surface divergence operator to Equation (39), to give the pressure Poisson equation

$$\frac{\Delta t}{\rho}\Delta_M p_{\text{corr}}=\nabla_M\cdot\vec{v}^{**}, \tag{40}$$

for spatially constant ρ. Finally, the pressure is updated

$$p^{(n+1)}=p^{(n)}+p_{\text{corr}}. \tag{41}$$

We note that the above scheme automatically enforces Equation (35). First, $\vec{v}^{**}\cdot\vec{n}=0$ by Equation (38). Furthermore, we know that the surface gradient of a scalar-valued function always lies in the tangential plane, that is, $\vec{n}\cdot\nabla_M\phi=0$, $\forall\phi : M\rightarrow\mathbb{R}$. Thus, $\vec{v}^{(n+1)}$ according to Equation (39) also lies in the tangential plane as it is given by a linear combination of $\vec{v}^{**}$ and $\nabla_M p_{\text{corr}}$.

To obtain the discrete systems, the continuous operators are replaced with the corresponding discrete ones as defined in Section 4.2, with the appropriate boundary conditions. So, for example, the discrete version of the pressure Poisson system Equation (40) on a closed surface (without boundaries) reads as

$$\frac{\Delta t}{\rho}\sum_{j\in S_i}c_{ij}^{\Delta_M}p_{\text{corr},j}=\sum_{j\in S_i}c_{ij}^{M,k}v_k^{**}, \tag{42}$$

for $k=x,y,z$, and the velocity $\vec{v}^{**}=(v_x^{**},v_y^{**},v_z^{**})^T$, and $i=1,2,\dots,N$.

We note that the explicit nature of the rotational components of the viscous term in the momentum equation leads to a CFL-like condition for stability, $\Delta t=C\frac{h^2}{\eta}$ for some constant C. However, empirically we observe significantly larger regions of stability when compared to the case of treating the entire viscous term explicitly. Completely removing this constraint by considering fully implicit schemes is part of our future work.

5.2 | Boundaries and boundary conditions

One of the advantages of meshfree GFDMs over other meshfree methods such as SPH is the ease of enforcing a variety of boundary conditions. This advantage also carries over to the surface PDE context, as illustrated in Reference 32. For example, to enforce boundary conditions on the pressure Poisson equation, the corresponding row of Equation (42) is simply replaced with the appropriate discrete boundary condition.

For the Lagrangian framework introduced earlier, a few extra modifications are needed for the imposition of certain boundary conditions. First, points on inflow and outflow boundaries are kept fixed with respect to the fluid velocity $\vec{v}$, and are only moved with the surface velocity $\vec{w}$. This ensures a certain regularity of the point cloud at the boundary. Moreover, if points at the inflow were moved in a Lagrangian sense, new points would have to be added at the inflow boundary. This would require an extrapolation of field values there, since the location would be outside the moved point cloud. On the other hand, if the inflow boundary points are fixed, points can be added in the resultant hole formed in the inflow direction (see Section 3.2), and those points would require an interpolation of field values with points present on all sides, and not an extrapolation. Furthermore, points crossing an outflow boundary are deleted. Unlike inflow and outflow boundaries, points on slip boundaries undergo the Lagrangian motion even with $\vec{v}$, but with the extra constraint that they must move only along the boundary. During the merging procedure explained in Section 3.2, priority is given to all boundary points, which ensures that an interior point and a boundary point are not merged, which would have led to artificially distorting the boundary.

Neumann boundary conditions for the pressure field $\vec{v} \cdot \nabla p = r$ are discretized as

$$\sum_{j \in S_i} c_{ij}^{M,v} p_j = r, \tag{43}$$

with

$$c_{ij}^{M,v} = \vec{v} \cdot \left(c_{ij}^{M,x}, c_{ij}^{M,y}, c_{ij}^{M,z} \right)^T. \tag{44}$$

For free boundaries, the boundary conditions are based on surface stresses

$$\vec{t}^{\,T} \boldsymbol{S}(\vec{v})\vec{v} = F_{ext}, \tag{45}$$

$$\vec{v}^{\,T} \boldsymbol{S}(\vec{v})\vec{v} = p - F_\sigma - F_c, \tag{46}$$

$$\vec{n}^{\,T} \boldsymbol{S}(\vec{v})\vec{v} = 0, \tag{47}$$

where F_{ext} is the contribution of external forces in the tangential direction, F_σ represents interface tension effects, and F_c represents contact angle effects. This is analogous to viscous stress-based free surface boundary conditions which have been shown to be very robust in volumetric flow simulations.[31,51] One of the advantages of such a formulation is the ease of modeling of different physical effects at the free boundaries, such as surface tension. However, for the sake of simplicity, we only consider the case $F_{ext} = F_\sigma = F_c = 0$ in the examples that follow. Since Equation (47) holds trivially for a tangential velocity field, we replace it with Equation (35).

6 | VALIDATION AND NUMERICAL RESULTS

In Section 3.6, we validated the Lagrangian framework for flow on (moving) surfaces. We now present a few validation examples for the surface Navier–Stokes equations solver presented above. An irregular point spacing is used in all numerical simulations. Linear systems arising from the semi-implicit intermediate velocity computation (37) and the pressure Poisson equation (40) are both solved using a BiCGSTAB solver.[52] The number of points used to discretize the domain is always indicated at initial time. Due to the addition and deletion of points to maintain regularity of the point cloud, the number of points will vary slightly in time. The surface Reynolds number is defined in a manner similar to the classical Reynolds number

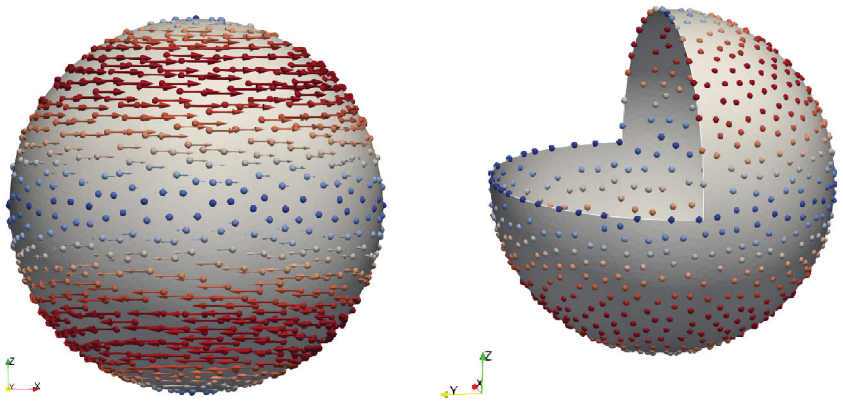


FIGURE 11 Decaying vortices on three quarters of a unit sphere. The domain is shown from two different view points. The color represents the magnitude of the velocity at $t = 0$. The arrows in the left figure show the velocity. A gray shell is added to visualize the domain [Colour figure can be viewed at wileyonlinelibrary.com]

$$\text{Re}_M = \frac{\rho U L}{\eta}, \tag{48}$$

for reference velocity U, and length L. For the first two examples presented below, which use a manufactured solution, we prescribe a solution for the pressure field p_{exact}, and one for the velocity field $\vec{v}_{\text{exact}}$ that is incompressible, and with $\vec{v}_{\text{exact}} \cdot \vec{n} = 0$. The manufactured gravity term is then computed by substituting these in the momentum equation

$$\vec{g}_{\text{manufactured}} = \frac{\partial \vec{v}_{\text{exact}}}{\partial t} + \left(\nabla_M^{cov} \vec{v}_{\text{exact}}\right) \vec{v}_{\text{exact}} - \frac{1}{\rho} \boldsymbol{P} \nabla_M \cdot \boldsymbol{S}(\vec{v}_{\text{exact}}) - \frac{1}{\rho} \nabla_M p_{\text{exact}}, \tag{49}$$

with $\vec{w} = 0$ in both cases.

Enforcing the local minimum and maximum inter-particle distances, as explained in Sections 2 and 3.2, is necessary to ensure stability of the spatial discretization scheme. A similar point also holds good for meshfree methods on volume domains. We emphasize that the parameter values used to control this minimum and maximum inter-particle distance (see Sections 2) do not result in "quasi-regular" point clouds. As can be seen in the figures that follow, for example Figures 11 and 18, all the point clouds used in the numerical examples are extremely non-uniform.

6.1 | Decaying vortices on three quarters of a sphere

For the first example, we consider a straight forward case with only Dirichlet boundaries. We start by proposing a manufactured solution on the surface of a unit sphere. The exact solution is given by

$$\vec{v}_{\text{exact}} = (-yz, xz, 0)^T \exp(-4\eta t), \tag{50}$$

$$p_{\text{exact}} = -\frac{z^4}{4} \exp(-8\eta t). \tag{51}$$

This results in two decaying vortices moving in opposite directions, one each in the positive and negative z hemispheres. The domain is taken to be three quarters of the unit sphere, as shown in Figure 11. Dirichlet boundary conditions are imposed for both the velocity and the pressure on the boundaries, based on the exact solution. For the simulations considered, we set $\rho = 1$ and $\eta = 0.01$. With $U = 1$, and $L = 1$ is the radius of the sphere, this results in a surface Reynolds number of about 100.

Errors are measured in a discrete L^2 sense, and taken relative to the analytical velocity

$$\epsilon_2(\vec{v}) = \frac{1}{||\vec{v}_{\text{exact}}||_2} \left(\frac{1}{N} \sum_{i=1}^{N} ||\vec{v}_i - \vec{v}_{\text{exact}}(\vec{x}_i)||^2 \right)^{\frac{1}{2}}. \tag{52}$$

Convergence of errors in the velocity field at $t = 1.0$ is shown in Figure 12 for varying spatial and temporal resolutions. For the convergence with smoothing length, we use $\Delta t = h^2$, with the smoothing length consecutively halving from a

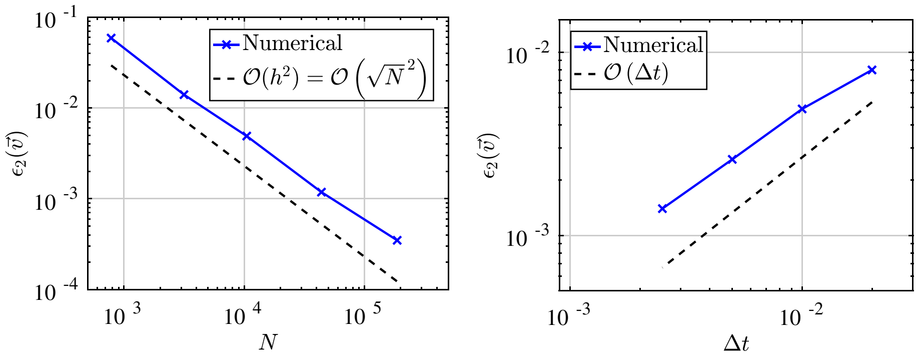


FIGURE 12 Decaying vortices on a sphere: Convergence of relative errors with a refining spatial resolution (left) and a decreasing time step (right) [Colour figure can be viewed at wileyonlinelibrary.com]

maximum of 0.4 to 0.025. This corresponds to the total number of points in the domain at $t = 0$ going from 781 to 187,430. For the temporal scale convergence, we use $h = 0.1$, which corresponds to $N = 10488$ at initial time.

Figure 12 illustrates that an experimental order of convergence of 2 is observed against varying spatial resolution, which matches the expectation due to the use of up to second order monomials in the numerical differential operators. Furthermore, an approximately first order convergence is observed with varying time step, which matches the use of the first order time integration scheme.

6.2 | Flow on a cylinder

We now take an example with more involved boundary conditions. Once again, a manufactured solution is used to test the applicability of the method proposed on a larger range of surface Reynolds flow. Consider the surface of a cylinder with axis along the x direction and unit radius, given by $y^2 + z^2 = 1$. The domain is limited by $x \in [0, 1]$. The proposed analytical solution is one where both velocity and pressure increase linearly in time

$$\vec{v}_{\text{exact}} = (y^2, -z, y)^T t, \tag{53}$$

$$p_{\text{exact}} = xt. \tag{54}$$

The boundary at $x = 1$ thus acts like an outflow, and that at $x = 0$ behaves likes an inflow, and the corresponding treatment for inflow and outflow boundaries respectively are applied, as mentioned in Section 5.2. Boundary conditions are based on the exact solution. For the pressure, a Dirichlet condition is imposed at the outflow boundary, and a Neumann condition at the inlet. For the velocity field, a Neumann condition is imposed on the outflow outflow with a Dirichlet one at the inlet.

We set $\rho = 1$, and vary η to consider different surface Reynolds numbers. For reference values $U = 1$, and $L = 1$ for the radius of the cylinder, this gives $\text{Re}_M = \frac{1}{\eta}$. Simulations are carried out on a point cloud with $h = 0.1$, which corresponds to $N = 7119$ points for the initial configuration. For varying η, the time step considered is given by

$$\Delta t = \min\left\{0.1, \frac{h^2}{\eta}\right\}. \tag{55}$$

Note the small time steps needed for stability for the high viscosity (low surface Reynolds) cases, owing to the explicit terms in the discrete momentum equation. Errors in the velocity field are measured as mentioned in the previous section. A similar definition is used for the pressure field

$$\epsilon_2(p) = \frac{1}{\|p_{\text{exact}}\|_2}\left(\frac{1}{N}\sum_{i=1}^{N}|p_i - p_{\text{exact}}(\vec{x}_i)|^2\right)^{\frac{1}{2}}. \tag{56}$$

The variation of errors in the velocity and pressure field at $t = 1.0$ with changing surface Reynolds numbers are shown in Figure 13. A similar error is observed for the velocity field in each of the cases, with a slight increase as the the viscosity increases. However, the increase in error in the pressure field in much more significant.

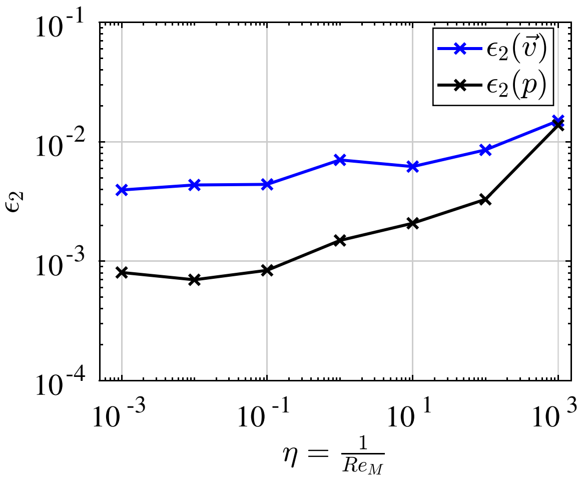


FIGURE 13 Flow on a cylinder surface: Relative errors in velocity and pressure for different surface Reynolds numbers [Colour figure can be viewed at wileyonlinelibrary.com]

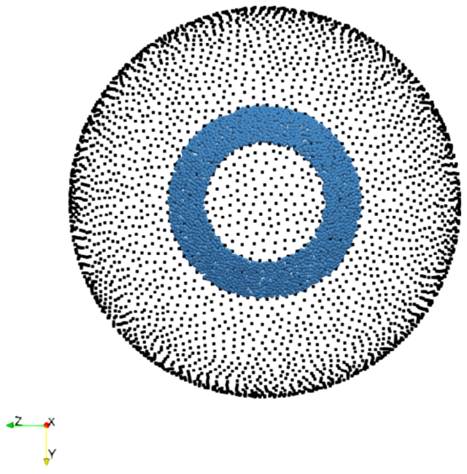


FIGURE 14 Domain and initial condition of the annular drop given by blue spherical glyphs, falling on a hemisphere marked with black points [Colour figure can be viewed at wileyonlinelibrary.com]

It is worth noting here that projection schemes used with strong form meshfree GFDM solvers for volumetric flow often use an additional pressure Poisson equation in each time step to obtain greater accuracy for the pressure field. This is typically done by taking the divergence of the momentum equation with the known velocity field $\vec{v}^{(n+1)}$, to obtain the final pressure. Exploring such methods for Navier–Stokes on manifolds, as well as fully implicit velocity schemes and coupled velocity pressure schemes in the meshfree GFDM context remains a topic of investigation beyond the scope of this article.

6.3 | Falling droplet

We now consider an example with free boundaries present. For this, we propose a benchmarking example similar to the drop falling standard case used in volumetric flow with free surfaces. The surface domain under consideration is given by a unit hemisphere with $x \leq 0$. An annular initial fluid droplet is considered as shown in Figure 14. This is given by the region of intersection of the hemisphere and an annular cylinder given by $0.3^2 \leq y^2 + z^2 \leq 0.5^2$. Gravity is set to 0, and the drop initiates movement with a prescribed initial velocity of $\vec{v}(\vec{x}, 0) = (y, -x, 0)^T$.

The physical properties used are $\rho = 1$, $\eta = 0.001$. The initial point cloud is seeded with $h = 0.06$, which corresponds $N = 1607$ points. Furthermore, a fixed time step of $\Delta t = 0.006$ is used. Slip boundary conditions are enforced at the surface boundary. The evolution of the falling drop is shown in Figure 15, with points colored according to the initial velocity.

To quantify the results, we use similar metrics to those used in the classical dam break example used to benchmark free surface volumetric flows (e.g., Reference 38). The evolution of the location of the droplet is quantified by plotting the maximum and minimum values of the x, y, and z coordinates in Figure 16. The prescribed initial velocity in the y direction, with a 0 velocity in the z direction causes the drop to elongate in the y direction. As a consequence of the mass

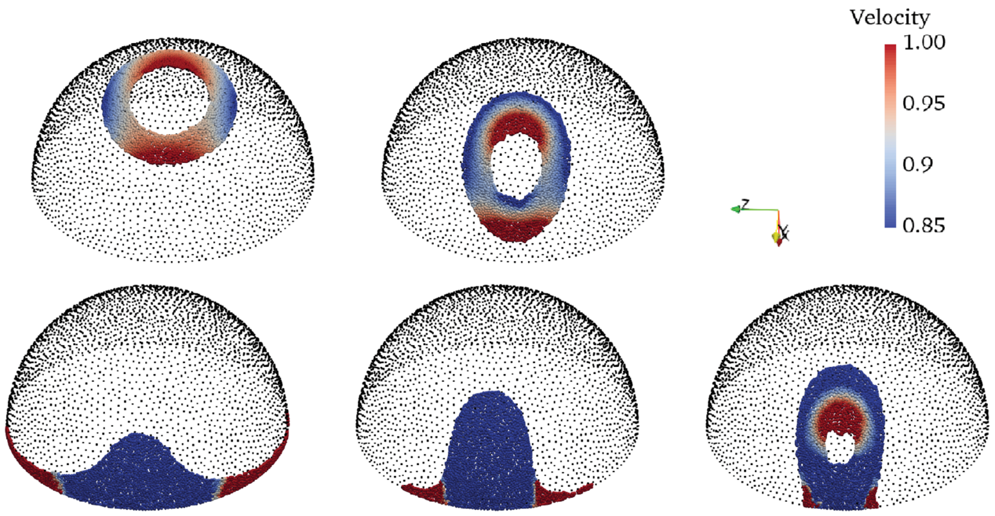


FIGURE 15 Falling annular droplet on a hemisphere. Domain at different times. Clockwise from top left: $t = 0.33$ (top left), $t = 0.86$ (top right), $t = 1.13$ (bottom right), $t = 1.4$ (bottom center), and $t = 1.88$ (bottom left). The fluid is shown with spherical colored glyphs. And the surface is shown with black points. The color of the fluid represents the velocity, colored according to the initial velocity distribution [Colour figure can be viewed at wileyonlinelibrary.com]

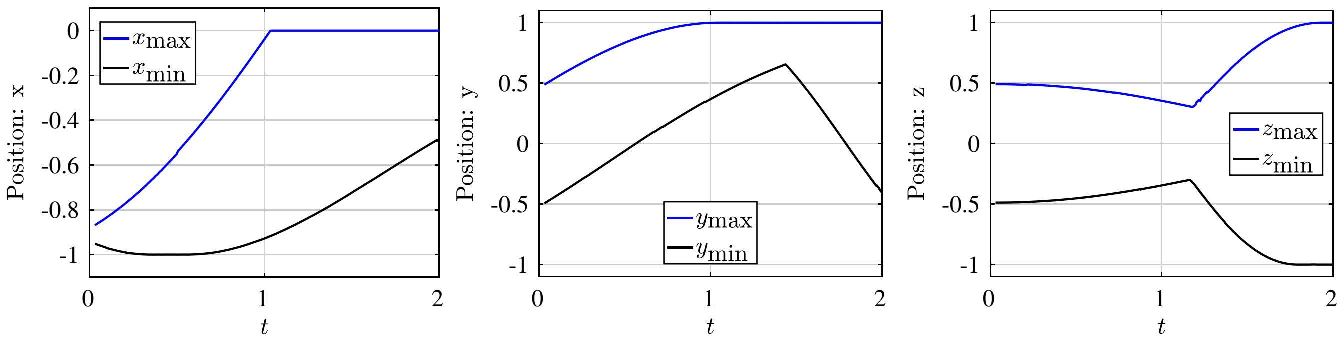


FIGURE 16 Drop falling: For the position vector $\vec{x}(t) = (x, y, z)^T$, the evolution of the maximum and minimum values of each coordinate are shown [Colour figure can be viewed at wileyonlinelibrary.com]

conservation constraint, the drop thins along the z direction as it falls. The annular drop reaches the boundary at $t = 1.03$. The increase in the z coordinate happens slightly after that. Just after hitting the boundary, the 'hole' in the droplet starts becoming smaller, till it closes at $t = 1.25$, after which the expansion in the z direction starts. Further, the maximum velocity reached is $||\vec{v}||_{\max} = 2.16$.

This example also illustrates the free surface detections algorithm, and the importance of preventing faulty addition of points outside boundaries. Since no point is added at a location surrounded by existing boundary points, the hole in the droplet is not filled too early.

It must be noted here that these results are not very representative physically as interface tension and contact angle effects have not been incorporated.

6.4 | Sloshing on an Armadillo

We now show a simulation with complex evolution of free boundaries where the fluid undergoes sloshing. The surface considered is the Stanford Armadillo.[53] At the initial time, fluid is filled on the armadillo surface up to a certain height, as shown in the first image of Figure 17. The armadillo is rotated till it becomes horizontal, which causes the fluid to fall due to gravity.

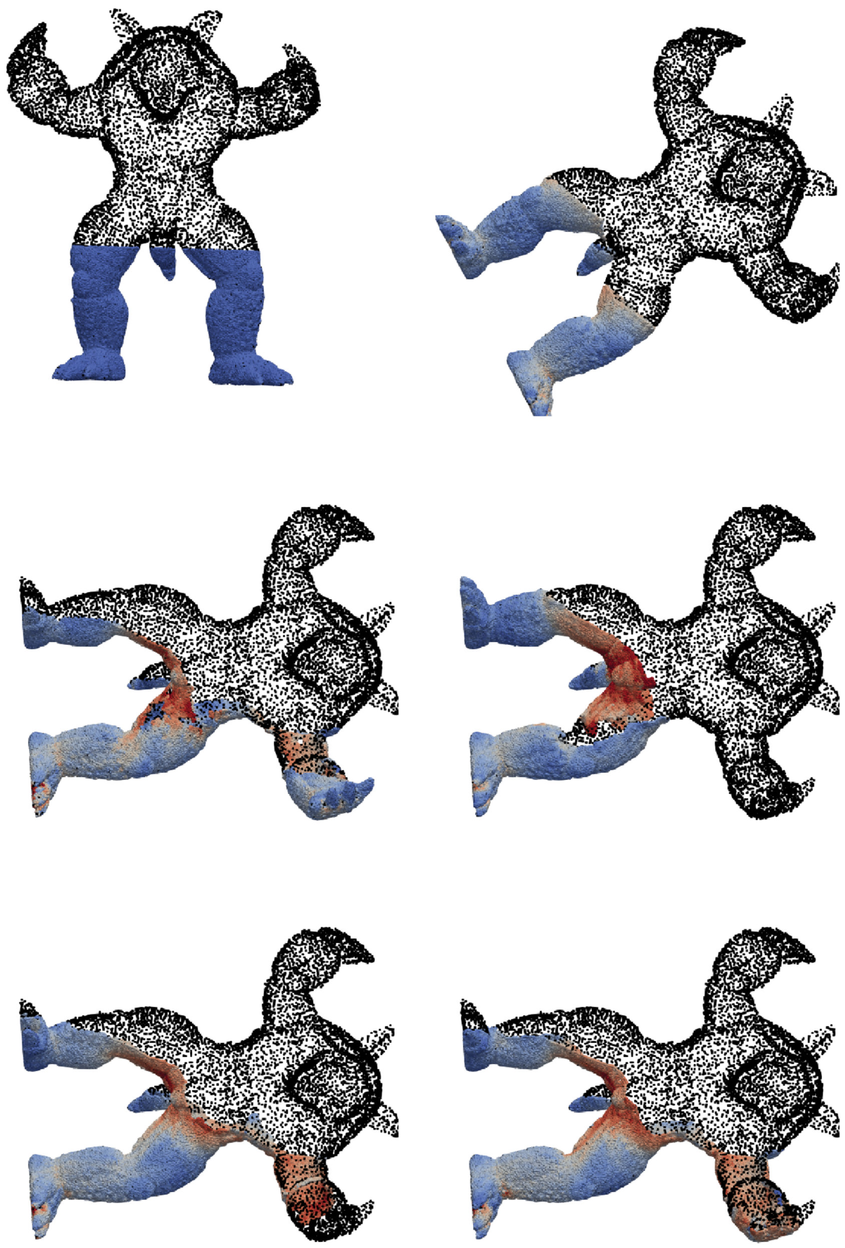


FIGURE 17 Sloshing on an Armadillo surface. Clockwise from top left: $t = 0$ (top left), $t = 1.1$ (top right), $t = 2.57$ (middle right), $t = 5.31$ (middle left), $t = 7.1$ (bottom left), and $t = 9.7$ (bottom right). The fluid is shown with spherical glyphs colored by the velocity, and the surface is shown with black points [Colour figure can be viewed at wileyonlinelibrary.com]

A scaled version of the armadillo is used with a scale factor of 0.035 to the original data set. The gravity field points downwards with respect to the initial state of the armadillo, and is prescribed by $\vec{g} = (0, 1, 0)^T$. $\rho = 1$ and $\eta = 0.01$ are used, with a varying time step of $\Delta t = 0.2h/||\vec{v}||_\infty$. The evolution of the free boundaries is shown in Figure 17. The rotation of the armadillo surface causes the fluid to fall under gravity. Initially, fluid is present in three disconnected regions, which merge as the fluid falls. This setup illustrates the capability of this method to handle complex geometries and free boundary evolution.

6.5 | Merging droplets on a deforming surface

As the last example, we consider free surface flow on a deforming surface. The initial surface is a unit sphere. The sphere deforms with a prescribed velocity

$$\vec{w} = -\cos(\pi z)(x, y, 0)^T. \tag{57}$$

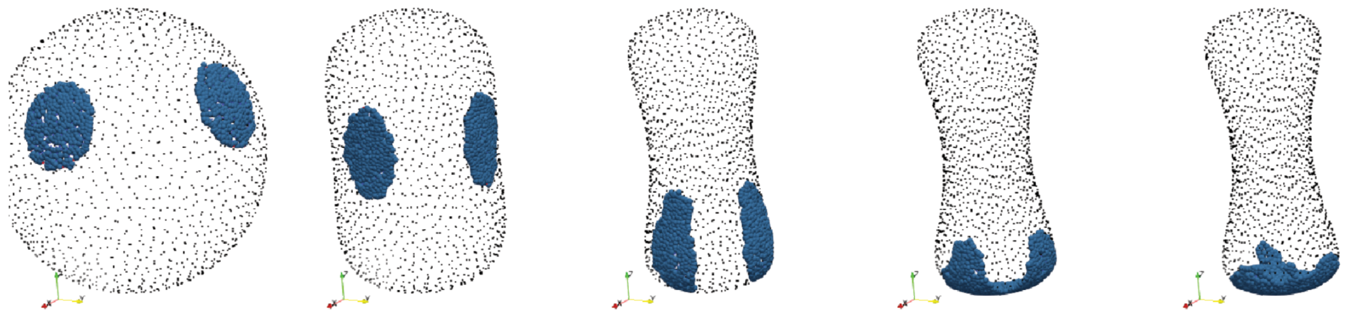
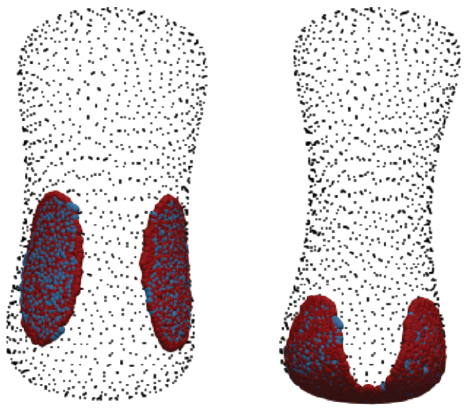


FIGURE 18 Merging droplets on a deforming surface. From left to right: $t = 0$, $t = 0.54$, $t = 1.0$, $t = 1.18$, and $t = 1.27$. The fluid is shown with blue spherical glyphs, and the surface is shown with black points. The scale and coordinate limits are the same in each of the images [Colour figure can be viewed at wileyonlinelibrary.com]

FIGURE 19 Merging droplets on a deforming surface. Overlay of simulation result of fine (red) and coarse point clouds (blue) at two different times. The fine (red) point cloud is also run on a fine background point cloud, which is not shown in the figure [Colour figure can be viewed at wileyonlinelibrary.com]



This results in the surface domain thinning around the "equator," and widening at "poles." Two initial fluid droplets are considered on the initial sphere, which move with the deforming surface, while also falling with gravity $\vec{g} = (0, 0, -2)^T$. The droplets merge as they fall down, which results in a change of the topology of the fluid domain. The evolution of the merging droplets and the deformation of the surface are shown in Figure 18. We note that, once again, interface tension has not been considered in these simulations. The physical properties of the fluid considered are $\rho = 1$ and $\eta = 0.01$.

The fluid domain consists of $N = 671$ points initially, and the initial background point cloud is composed on $N^{\text{back}} = 2866$ points. The simulation is carried out with a varying time step $\Delta t = 0.03\frac{h}{\|\vec{v}\|_\infty}$. The complete simulation, including pre and post-processing, takes a total of 17s to run serially up to a simulated time of $t = 1.5$. This example illustrates the capability of the introduced method to handle flow with free boundaries on deforming surfaces.

We also consider the same simulation with a finer resolution of both the main fluid point cloud, and the background point cloud. Here, we have $N^{\text{back}} = 73,456$ points and $N = 18,274$ points initially. The time step is defined as done above, $\Delta t = 0.03\frac{h}{\|\vec{v}\|_\infty}$. For the same simulated time of $t = 1.5$, this simulation take a total of 25 minutes to run serially. The fluid points clouds in the coarse and the fine simulations are overlayed on each other in Figure 19, with only the coarse background point cloud shown. The evolution of the droplets and the surface are qualitatively observed to be very similar in both cases, as is evident in Figure 19.

7 | CONCLUSION

We introduced a meshfree Lagrangian framework to model fluid flow on curved and moving surfaces. The method is intrinsic to the surface in the sense that only the surface is discretized without any form of discretization of the bulk around the surface. Meshfree points or particles are moved in a Lagrangian sense subject to the constraint of lying on the surface. The position of the surface itself can also be evolving in time. Methods to handle moving free boundaries within a surface were introduced, along with the boundary conditions for the same. Flow on a surface without free boundaries was investigated numerically using the basis of a point cloud that represents both the fluid domain, and the surface that constrained the fluid flow. On the other hand, for flow with free boundaries, the use of two point clouds was introduced. For the latter case, the main fluid point cloud was moved on another background point cloud that represented the (possibly moving) surface. The introduced Lagrangian framework incorporates local fixes for point cloud distortion, which removes the need for a possible global remeshing in moving mesh methods. As a result, the methods presented here provide a straightforward way to handle flow on moving and distorting surfaces, where the flow domain itself can also deform.

Coupling this with a meshfree GFDM, we presented a strong form scheme to solve the Navier–Stokes equations on manifolds. This used a modified Chorin projection approach, where the tangential velocity constraint is enforced as an additional projection step, before the projection for incompressibility. Validation cases were introduced for the Lagrangian framework, and for incompressible surface Navier–Stokes flow with free boundaries. The numerical examples illustrate the potential of the introduced method to be used for a variety of applications of surface flow.

Several directions of work could be pursued to improve the scheme presented here. Most importantly, different schemes need to be explored for improved stability with larger time steps, as has been done for meshfree collocation schemes for volumetric flow.[54] Particularly, the development of a scheme with a fully implicit velocity step, or an implicit coupled velocity-pressure scheme, need to be considered. The biggest challenge in this is handling the viscous term implicitly. Additionally, improvements in accuracy for the pressure field may also be obtained with an additional pressure Poisson equation at the end of the scheme, obtained by substituting the final velocity in the momentum equation, as has shown to be useful in meshfree GFDM schemes for volumetric flow.[30]

ACKNOWLEDGEMENT

Open access funding enabled and organized by Projekt DEAL.

DATA AVAILABILITY STATEMENT

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

ORCID

Pratik Suchde https://orcid.org/0000-0002-4807-5322

REFERENCES

1. Barrett JW, Garcke H, Nürnberg R. Numerical computations of the dynamics of fluidic membranes and vesicles. *Phys Rev E*. 2015;92:052704.
2. Edwards DA, Brenner H, Wasan DT. *Interfacial Transport Processes and Rheology*. Boston, MA: Butterworth-Heinemann; 1991.
3. Grassia P, Ubal S, Giavedoni MD, Vitasari D, Martin PJ. Surfactant flow between a plateau border and a film during foam fractionation. *Chem Eng Sci*. 2016;143:139-165.
4. O'Brien S, Schwartz L. Theory and modeling of thin film flows. *Encyclopedia of Surface and Colloid Science*. 2002;1:5283-5297. https://www.semanticscholar.org/paper/THEORY-AND-MODELING-OF-THIN-FILM-FLOWS-O'Brien-Schwartz/1cc51908e0dd5fbf43a8b8ed481844a1fc28f8e3.
5. Stam J. Flows on surfaces of arbitrary topology. *ACM Transactions on Graphics*. San Diego, California. 2003;22(3):724-731. http://dx.doi.org/10.1145/882262.882338.
6. Chan CH, Czubak M, Disconzi MM. The formulation of the Navier–Stokes equations on Riemannian manifolds. *J Geom Phys*. 2017;121:335-346.
7. Jankuhn T, Olshanskii M, Reusken A. Incompressible fluid problems on embedded surfaces: Modeling and variational formulations. *Interfaces and Free Boundaries*. 2018;20(3):353-377. http://dx.doi.org/10.4171/ifb/405.
8. Koba H, Liu C, Giga Y. Energetic variational approaches for incompressible fluid systems on an evolving surface. *Quart Appl Math*. 2017;75(2):359-389.
9. Miura T-H. On singular limit equations for incompressible fluids in moving thin domains. *Q Appl Math*. 2018;76(2):215-251.
10. Boussinesq M. Sur l'existence d'une viscosité superficielle, dans la mince couche de transition séparant un liquide d'un autre fluide contigu. *Ann Chim Phys*. 1913;29:349-357.
11. Scriven L. Dynamics of a fluid interface equation of motion for newtonian surface fluids. *Chem Eng Sci*. 1960;12(2):98-108.
12. Ebin DG, Marsden J. Groups of diffeomorphisms and the motion of an incompressible fluid. *Ann Math*. 1970;92(1):102-163.
13. Auer S, Macdonald C, Treib M, Schneider J, Westermann R. Real-time fluid effects on surfaces using the closest point method. *Comput Graph Forum*. 2012;31(6):1909-1923.
14. Auer S, Westermann R. A semi-Lagrangian closest point method for deforming surfaces. *Comput Graph Forum*. 2013;32(7):207-214.
15. Gross B, Atzberger P. Hydrodynamic flows on curved surfaces: spectral numerical methods for radial manifold shapes. *J Comput Phys*. 2018;371:663-689.

16. Gross B, Trask N, Kuberry P, Atzberger P. Meshfree methods on manifolds for hydrodynamic flows on curved surfaces: a generalized moving least-squares (GMLS) approach. *J Comput Phys*. 2020;409:109340.
17. Fengler MJ, Freeden W. A nonlinear Galerkin scheme involving vector and tensor spherical harmonics for solving the incompressible Navier–Stokes equation on the sphere. *SIAM J Sci Comput*. 2005;27(3):967-994.
18. Dritschel DG, Qi W, Marston JB. On the late-time behaviour of a bounded, inviscid two-dimensional flow. *J Fluid Mech*. 2015;783:1-22.
19. Qi W, Marston JB. Hyperviscosity and statistical equilibria of Euler turbulence on the torus and the sphere. *J Stat Mech Theory Exper*. 2014;2014(7):P07020.
20. Fries T-P. Higher-order surface FEM for incompressible Navier–Stokes flows on manifolds. *Int J Numer Methods Fluids*. 2018;88(2):55-78.
21. Olshanskii MA, Yushutin V. A penalty finite element method for a fluid system posed on embedded surface. *J Math Fluid Mech*. 2019;21(1):14.
22. Reuther S, Voigt A. Solving the incompressible surface Navier–Stokes equation by surface finite elements. *Phys Fluids*. 2018;30(1):012107.
23. März T, Macdonald CB. Calculus on surfaces with general closest point functions. *SIAM J Numer Anal*. 2012;50(6):3303-3328.
24. Ruuth SJ, Merriman B. A simple embedding method for solving partial differential equations on surfaces. *J Comput Phys*. 2008;227(3):1943-1961.
25. Nitschke I, Voigt A, Wensch J. A finite element approach to incompressible two-phase flow on manifolds. *J Fluid Mech*. 2012;708:418-438.
26. Suchde P, Kuhnert J. A fully Lagrangian meshfree framework for PDEs on evolving surfaces. *J Comput Phys*. 2019;395:38-59.
27. Domínguez JM, Crespo AJC, Gómez-Gesteira M, Marongiu JC. Neighbour lists in smoothed particle hydrodynamics. *Int J Numer Methods Fluids*. 2011;67(12):2026-2042.
28. Drumm C, Tiwari S, Kuhnert J, Bart H-J. Finite pointset method for simulation of the liquid - liquid flow field in an extractor. *Comput Chem Eng*. 2008;32(12):2946-2957.
29. Onderik J, Ďurikovič R. Efficient neighbor search for particle-based fluids. *J Appl Math Stat Inform*. 2008;4(1):29-43.
30. Jefferies A, Kuhnert J, Aschenbrenner L, Giffhorn U. Finite pointset method for the simulation of a vehicle travelling through a body of water. In: Griebel M, Schweitzer AM, eds. *Meshfree Methods for Partial Differential Equations VII*. Cham, Switzerland: Springer International Publishing; 2015:205-221.
31. Suchde P, Kuhnert J, Schröder S, Klar A. A flux conserving meshfree method for conservation laws. *Int J Numer Methods Eng*. 2017;112(3):238-256.
32. Suchde P, Kuhnert J. A meshfree generalized finite difference method for surface PDEs. *Comput Math Appl*. 2019;78(8):2789-2805.
33. Suchde P, Kuhnert J. Point cloud movement for fully Lagrangian meshfree methods. *J Comput Appl Math*. 2018;340:89-100.
34. Pahar G, Dhar A. A robust volume conservative divergence-free ISPH framework for free-surface flow problems. *Adv Water Resour*. 2016;96:423-437.
35. Evans MW, Harlow FH, Bromberg E. *The Particle-in-Cell Method for Hydrodynamic Calculations. Technical Report*. Los Alamos, NM: Los Alamos National Lab; 1957.
36. Marrone S, Colagrossi A, Touzé DL, Graziani G. Fast free-surface detection and level-set function definition in SPH solvers. *J Comput Phys*. 2010;229(10):3652-3663.
37. Saucedo-Zendejo FR, Reséndiz-Flores EO, Kuhnert J. Three-dimensional flow prediction in mould filling processes using a GFDM. *Comput Part Mech*. 2019;6(3):411-425.
38. Tiwari S, Kuhnert J. Particle method for simulation of free surface flows. In: Hou TY, Tadmor E, eds. *Hyperbolic Problems: Theory, Numerics, Applications: Proceedings of the 9th International Conference on Hyperbolic Problems held in CalTech, Pasadena, March 25–29, 2002*. Berlin, Heidelberg/Germany: Springer; 2003:889-898.
39. Kuhnert J, Michel I, Mack R. Fluid structure interaction (FSI) in the meshfree Finite Pointset Method (FPM): theory and applications. In: Griebel M, Schweitzer AM, eds. *Meshfree Methods for Partial Differential Equations IX, IWMMPDE2017*. Berlin, Heidelberg/Germany: Springer; 2019:73-92.
40. Bertalmio M, Cheng L-T, Osher S, Sapiro G. Variational problems and partial differential equations on implicit surfaces. *J Comput Phys*. 2001;174(2):759-780.
41. Delfour M, Zolesio J. A boundary differential equation for thin shells. *J Differ Equ*. 1995;119(2):426-449.
42. Dziuk G. Finite elements for the beltrami operator on arbitrary surfaces. In: Hildebrandt S, Leis R, eds. *Partial Differential Equations and Calculus of Variations*. Berlin, Heidelberg: Springer Berlin Heidelberg; 1988:142-155.
43. Williamson DL, Drake JB, Hack JJ, Jakob R, Swarztrauber PN. A standard test set for numerical approximations to the shallow water equations in spherical geometry. *J Comput Phys*. 1992;102(1):211-224.
44. Fan C-M, Chu C-N, Šarler B, Li T-H. Numerical solutions of waves-current interactions by generalized finite difference method. *Eng Anal Bound Elem*. 2019;100:150-163. http://dx.doi.org/10.1016/j.enganabound.2018.01.010.
45. Gavete L, Ureña F, Benito J, García A, Ureña M, Salete E. Solving second order non-linear elliptic partial differential equations using generalized finite difference method. *J Comput Appl Math*. 2017;318:378-387. Computational and Mathematical Methods in Science and Engineering CMMSE-2015.
46. Katz A, Jameson A. Meshless scheme based on alignment constraints. *AIAA J*. 2010;48(11):2501-2511.
47. Luo M, Koh CG, Bai W, Gao M. A particle method for two-phase flows with compressible air pocket. *Int J Numer Methods Eng*. 2016;108:695-721.

48. Demanet L. *Painless, Highly Accurate Discretizations of the Laplacian on a Smooth Manifold. Technical Report*. Stanford, CA: Stanford University; 2006.
49. Liang J, Zhao H. Solving partial differential equations on point clouds. *SIAM J Sci Comput*. 2013;35(3):A1461-A1486.
50. Chorin AJ. Numerical solution of the Navier–Stokes equations. *Math Comput*. 1968;22(104):745-762.
51. Landau L, Lifschitz E. *Lehrbuch der theoretischen Physik, Band VI: Hydrodynamik*. 5th ed. Berlin, Germany: Akademie-Verlag; 1991.
52. van der Vorst HA. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J Sci Stat Comput*. 1992;13(2):631-644.
53. Krishnamurthy V, Levoy M. Fitting smooth surfaces to dense polygon meshes. Paper presented at: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '96, New Orleans, Louisiana; 1996.
54. Suchde P, Kuhnert J, Tiwari S. On meshfree GFDM solvers for the incompressible Navier–Stokes equations. *Comput Fluids*. 2018;165:1-12.

How to cite this article: Suchde P. A meshfree Lagrangian method for flow on manifolds. *Int J Numer Meth Fluids*. 2021;93:1871–1894. https://doi.org/10.1002/fld.4957