Received: 1 June 2016 | Revised: 29 March 2017 | Accepted: 15 April 2017

DOI: 10.1002/fld.4392

RESEARCH ARTICLE

WILEY

Computing the force distribution on the surface of complex, deforming geometries using vortex methods and Brinkman penalization

Siddhartha Verma[1] | Gabriele Abbati[1] | Guido Novati[1] | Petros Koumoutsakos[1,2,3]

[1]Computational Science and Engineering Laboratory, ETH Zürich, Clausiusstrasse 33 Zürich, CH-8092, Switzerland

[2]Radcliffe Institute of Advanced Study, Harvard University, MA, United States of America

[3]Wallace Visiting Professor, Massachusetts Institute of Technology, MA, United States of America

Correspondence
Petros Koumoutsakos, Computational Science and Engineering Laboratory, ETH Zürich, Clausiusstrasse 33, Zürich CH-8092, Switzerland.
Email: petros@ethz.ch

Funding information
European Research Council Advanced Investigator Award, Grant/Award Number: 341117; Swiss National Science Foundation Sinergia Award, Grant/Award Number: CRSII3_147675; Swiss National Supercomputing Centre (CSCS), Grant/Award Number: s436, s658

Summary

The distribution of forces on the surface of complex, deforming geometries is an invaluable output of flow simulations. One particular example of such geometries involves self-propelled swimmers. Surface forces can provide significant information about the flow field sensed by the swimmers and are difficult to obtain experimentally. At the same time, simulations of flow around complex, deforming shapes can be computationally prohibitive when body-fitted grids are used. Alternatively, such simulations may use penalization techniques. Penalization methods rely on simple Cartesian grids to discretize the governing equations, which are enhanced by a penalty term to account for the boundary conditions. They have been shown to provide a robust estimation of mean quantities, such as drag and propulsion velocity, but the computation of surface force distribution remains a challenge. We present a method for determining flow-induced forces on the surface of both rigid and deforming bodies, in simulations using remeshed vortex methods and Brinkman penalization. The pressure field is recovered from the velocity by solving a Poisson's equation using the Green's function approach, augmented with a fast multipole expansion and a tree-code algorithm. The viscous forces are determined by evaluating the strain-rate tensor on the surface of deforming bodies, and on a “lifted” surface in simulations involving rigid objects. We present results for benchmark flows demonstrating that we can obtain an accurate distribution of flow-induced surface forces. The capabilities of our method are demonstrated using simulations of self-propelled swimmers, where we obtain the pressure and shear distribution on their deforming surfaces.

KEYWORDS

Brinkman penalization, complex shapes, deforming geometry, fluid-structure interaction, surface forces, vortex methods

1 | INTRODUCTION

The distribution of surface forces, and its dependence on the shape and motion of biological organisms and engineering devices, is among the most valuable information that can be acquired through numerical simulations. Such information may not be easy to obtain experimentally, particularly for biological organisms, such as swimmers that involve complex deforming bodies. The measurement of surface forces was the subject of some of the pioneering observations concerned with swimming organisms by Gray,[1] and experiments by DuBois and Ogilvy.[2] Theoretical studies of fish swimming by Lighthill[3] and Wu[4] used potential flow theory to provide estimates for the pressure field on the body surface. More recent work[5,6] has provided new impetus for the measurement of such quantities, which are deemed to be essential for answering some of the most critical questions concerning our understanding of fish swimming.[7]

Simulations can complement experiments by providing complete information regarding the distribution of surface forces on swimming organisms. Direct numerical simulations of swimming organisms were first performed using Arbitrary Lagrangian-Eulerian methods.[8–10] Arbitrary Lagrangian-Eulerian methods[11–14] use body-fitted grids, where the surface of the solid is delineated explicitly by the mesh. This makes enforcing boundary conditions relatively straightforward. However, body-fitted grids may encounter difficulties with bodies experiencing large deformations, and the grid must be reconstructed frequently to account for moving and deforming objects. Frequent mesh regeneration is undesirable, as it entails significant computational overhead. This overhead can be especially prohibitive when performing shape and motion optimization of fish swimming using evolutionary algorithms.[14] Alternative approaches, referred to as immersed boundary methods,[15–20] reduce computational cost by using simple, non–body-fitted meshes. The no-slip condition is imposed on curved boundaries using appropriate interpolation of the velocity field across the fluid-solid interface. In a different approach, Cartesian grids have been combined with penalization techniques[21] for simulations of fish swimming.[22,23] Penalization techniques allow for easy computation of the rotational and translational motion of self-propelled swimmers, via integrals of the flow field over the body volume. However, penalization techniques present challenges when detailed distributions of the pressure-induced and viscous forces are required on the body surface. Such force fields may be noisy owing to the irregular intersection of the curved body geometry with the Cartesian grid.

Early developments of immersed boundary methods were coupled with vortex methods[24] for enforcing the no-slip boundary condition. Vortex methods primarily discretize the vorticity present in the flow field and can prove to be advantageous in flows involving a limited support of the vorticity field, such as fish wakes. These early simulations were abandoned because of inaccuracies of vortex methods at that time. Such limitations were overcome in later work that used remeshing to enforce the regularization of particle locations,[25] or used a vorticity flux algorithm for the enforcement of the no-slip condition,[26] and made use of variable kernel sizes and domain-decomposition techniques to account for multiple bodies.[27] However, their applicability to 3D and multiple deforming bodies was limited because of the requirement of body fitted grids, or because of limitations imposed on the Cartesian meshes by remeshing.

Recent work using the combination of vortex methods with Brinkman penalization has demonstrated that the approach is extremely effective in simulating fluid-structure interaction of complex, temporally deforming geometries.[22,28,29] A number of studies have relied on this technique for coupling simulations of self-propelled swimmers with machine learning algorithms,[30] and for design-optimization of swimmer morphology and kinematics.[31–34] These studies demonstrate the capability of the penalization algorithm to handle multiple, complex, and deforming geometries with relative ease. Nonetheless, these studies have not overcome the difficulty of determining pointwise forces exerted on a solid surface. Surface forces are the primary means through which fluids exert an influence on solid objects, and knowledge of these forces is essential for analyzing fluid-solid interactions. The absence of an explicit calculation for the pressure term in vortex methods, and the smooth interface used for the penalization algorithm, pose considerable challenges when determining surface forces. These issues are overcome in the current work. A complete methodology is presented to accurately compute surface forces in simulations of 2-dimensional flows past complex deforming geometries, using Cartesian grids.

The paper is organized as follows. The equations for determining surface forces, and the numerical techniques used, are described in Section 2. Validations with simulations of impulsively started 2-dimensional cylinders, simulations of flow over a rigid streamlined profile, and simulations of a self-propelled swimmer are discussed in Section 3. Section 4 summarizes the results and the numerical techniques outlined in the paper.

2 | NUMERICAL METHODS

This work is concerned with 2-dimensional, viscous, incompressible flows past complex, rigid, and deforming geometries. In this section, we provide a brief description of the equations governing the interaction between fluid flow and solid objects, and the numerical procedure used to compute the pressure-induced and viscous forces on a solid.

2.1 | Vortex methods and Brinkman penalization

The spatial and temporal evolution of the velocity field in our simulations is based on the incompressible Navier-Stokes equations:

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

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = \frac{-\nabla P}{\rho} + \nu \nabla^2 \boldsymbol{u}. \tag{2}$$

The interaction between fluid-flow and solid objects is achieved by introducing a penalty term in the momentum equation (Brinkman penalization[21]), which enforces the no-slip boundary condition at the fluid-solid interface:

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = \frac{-\nabla P}{\rho} + \nu \nabla^2 \boldsymbol{u} + \lambda \chi (\boldsymbol{u}_s - \boldsymbol{u}). \tag{3}$$

Here, λ is the penalization parameter, and χ is the characteristic function which represents the discretized solid on a Cartesian grid. Grid points with $\chi = 0$ are occupied entirely by the fluid, and those with $\chi = 1$ are occupied entirely by the solid. To minimize numerical oscillations, χ transitions smoothly from 0 to 1 within 2 grid points at the fluid-solid interface, using a discrete representation of the Heaviside function.[35] The symbol $\boldsymbol{u}_s$ in Equation 3 denotes the pointwise velocity of the discretized solid and accounts for translation, rotation, and deformation of the body.

To solve the motion resulting from fluid-solid interaction, we use an open-source solver[29] based on remeshed vortex methods.[36] These methods use the vorticity form of the momentum equation, which may be obtained by taking the curl of Equation 3:

$$\frac{\partial \omega}{\partial t} + (\boldsymbol{u} \cdot \nabla)\omega = \nu \nabla^2 \omega + \lambda \nabla \times (\chi (\boldsymbol{u}_s - \boldsymbol{u})). \tag{4}$$

In deriving Equation 4, we have used $\nabla \cdot \boldsymbol{u} = 0$ (incompressibility), and $\nabla \rho = 0$, since we restrict our investigation to neutrally buoyant solids (ie, $\rho_{\text{solid}} = \rho_{\text{liquid}}$). Furthermore, the vortex-stretching term $(\boldsymbol{\omega} \cdot \nabla \boldsymbol{u})$ is absent from the momentum equation for 2-dimensional cases. Time splitting of Equation 4 is done via Godunov splitting, and the partial differential equation is solved in sequential steps as shown below:

1. Recovery of velocity from vorticity

$$\nabla^2 \psi = -\omega, \tag{5a}$$

$$\boldsymbol{u} = \nabla \times \psi. \tag{5b}$$

2. Penalization of the vorticity equation

$$\frac{\partial \omega}{\partial t} = \lambda \nabla \times (\chi (\boldsymbol{u}_s - \boldsymbol{u})). \tag{6}$$

3. Diffusion of the vorticity field

$$\frac{\partial \omega}{\partial t} = \nu \nabla^2 \omega. \tag{7}$$

4. Semi-Lagrangian advection of the vorticity field

$$\frac{D\omega}{Dt} = 0 \tag{8}$$

Here, time integration of the penalty step is performed via the fully implicit backward Euler method, whereas the diffusion step is integrated via the fully explicit forward Euler method. Discretization of spatial derivatives is done via fourth-order centered differences. The operator D/Dt in Equation 8 represents the material derivative. The Poisson's equation in Equation 5b is solved using the fast multipole method on an unbounded domain, with both the velocity and the vorticity vanishing at infinity. We note that the solution for the streamfunction ψ is unique only up to an additive constant. However, the velocity field $\boldsymbol{u}$, which depends on derivatives of ψ, is uniquely determined. Additional details regarding the solution of each of the steps listed above may be found in Gazzola et al [22] and Rossinelli et al. [29]

2.2 | Recovering pressure from the velocity field

Advancing the vorticity field in time using Equation 4 requires the computation of velocity at every time step (Equation 5, which may also be presented in another form: $\nabla^2 \boldsymbol{u} = -\nabla \times \boldsymbol{\omega}$). The pressure field can be recovered from the velocity by taking the divergence of the penalized momentum equation (Equation 3):

$$\begin{aligned}\frac{\partial(\nabla \cdot \boldsymbol{u})}{\partial t} + \left(\nabla \boldsymbol{u}^T : \nabla \boldsymbol{u}\right) + (\boldsymbol{u} \cdot \nabla)(\nabla \cdot \boldsymbol{u}) = \frac{-\nabla^2 P}{\rho} + \frac{1}{\rho^2}\nabla\rho \cdot \nabla P + \nu\nabla^2(\nabla \cdot \boldsymbol{u}) \\ + \lambda\nabla \cdot (\chi(\boldsymbol{u}_s - \boldsymbol{u})).\end{aligned} \tag{9}$$

The colon operator in $\nabla \boldsymbol{u}^T : \nabla \boldsymbol{u}$ denotes the tensor dot product, which can be written in index notation as $u_{j,i}u_{i,j}$. Using the incompressibility condition ($\nabla \cdot \boldsymbol{u} = 0$) and $\nabla\rho = 0$ (neutrally buoyant solid), Equation 9 simplifies to a Poisson's equation for the pressure term:

$$\nabla^2 P = -\rho\left(\nabla \boldsymbol{u}^T : \nabla \boldsymbol{u}\right) + \rho\lambda\nabla \cdot (\chi(\boldsymbol{u}_s - \boldsymbol{u})) \tag{10}$$

This Poisson's equation is solved using a fast tree-code algorithm based on multipole expansions,[37] as described in Appendix A. The penalty term in this equation ($\rho\lambda\nabla \cdot \chi(\boldsymbol{u}_s - \boldsymbol{u})$) accounts for the pressure-source contribution induced by fluid-solid interactions.

2.3 | Determining the surface forces

When using Brinkman penalization, the absence of a sharply defined fluid-solid interface poses the biggest obstacle in determining the forces exerted on a solid surface. To overcome this difficulty, we consider the pressure-induced (isotropic) and viscosity-induced (deviatoric) parts of the forces separately.

2.3.1 | Pressure-induced forces

Pressure-induced forces act on a surface immersed in a fluid, even when both the fluid and the solid are at rest. Computing these forces ($\boldsymbol{F}_P$) requires knowledge of the pressure, the surface normal ($\boldsymbol{n}$), and the infinitesimal surface area (dS):

$$d\boldsymbol{F}_P = -P\boldsymbol{n}\, dS. \tag{11}$$

The surface coordinates ($\boldsymbol{x}_{surf}$) and normals defining the solid boundary are known precisely at each time step in simulations. These exact surface coordinates ($\boldsymbol{x}_{surf}$), which may not necessarily coincide with the grid nodes, are used as target locations z for computing $P\left(\boldsymbol{x}_{surf}\right)$ using Equation A14. Evaluating the pressure at the body-surface coordinates instead of the entire computational domain reduces the cost of the Poisson solver significantly.

Once the pointwise pressure-induced surface forces are known from Equation 11, their resultant on the entire body can be determined using a closed integral over the surface of the solid object:

$$\boldsymbol{F}_P = \oint\!\!\oint d\boldsymbol{F}_P = \oint\!\!\oint -P\boldsymbol{n}\, dS. \tag{12}$$

$\boldsymbol{F}_P$ can be projected along the velocity vector to obtain the contribution of pressure-induced forces to thrust and drag. For stationary solid objects, the pressure-induced drag and the corresponding drag coefficient are defined as follows:

$$(Drag)_P = \frac{\boldsymbol{F}_P \cdot \boldsymbol{U}_\infty}{\|\boldsymbol{U}_\infty\|} \quad \text{and} \quad Cd_P = \frac{2(Drag)_P}{\rho\|\boldsymbol{U}_\infty\|^2 A}, \tag{13}$$

where $\boldsymbol{U}_\infty$ is the free-stream velocity and A represents the reference area, with $A = (2 \cdot \text{radius})$ for 2-dimensional cylinders.

2.3.2 | Viscous forces

To compute viscous forces ($\boldsymbol{F}_\nu$) resulting from relative motion between the fluid and the solid, we use the pointwise strain-rate tensor $\boldsymbol{D}(\boldsymbol{u}(\boldsymbol{x}))$:

$$d\boldsymbol{F}_\nu = 2\mu\boldsymbol{D} \cdot \boldsymbol{n}\, dS \quad \text{where} \quad \boldsymbol{D} = \frac{1}{2}\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right). \tag{14}$$

Here, μ is the dynamic viscosity of the fluid. For simulations involving deforming objects (ie, self-propelled swimmers), $\nabla\boldsymbol{u}$ in Equation 14 is evaluated at the surface of the solid. However, in the case of rigid objects this leads to inaccurate estimation of velocity gradients. Examining the flow-field adjacent to rigid objects reveals that the velocity magnitude and velocity-derivatives are essentially zero next to the solid boundary. We conjecture that the smoothing of the characteristic function on the computational grid gives rise to this behavior under steady-flow conditions. To overcome this difficulty, $\nabla\boldsymbol{u}$ is computed on a "lifted" body surface via nearest-neighbor interpolation. The lifted surface is formed by moving the original solid surface outward along the surface normal. Empirical tests indicate that a distance of $2h$ (where h is the grid-cell size) from the exact edge of rigid objects yields the best accuracy for determining $\nabla\boldsymbol{u}$. We emphasize that such a correction is not necessary in the case of temporally deforming shapes (such as self-propelled swimmers), which are of primary interest for the algorithm presented in this paper. It is possible that a similar correction may be necessary in the case of rigid 3D objects and that the issue may not manifest for deforming 3D geometries; however, this would need to be verified via 3-dimensional simulations. Certain techniques overcome the issue of inaccurate derivatives by computing fluxes on a surface away from the body, and combining these with time derivatives and volume and surface integrals.[38,39] However, this approach is not feasible in the current situation, since our goal is to obtain the spatial distribution of the flow-induced forces on the body, whereas the integral methods yield only the net force, or net acceleration.

With the pointwise viscous forces known from Equation 14, the resultant quantities $\boldsymbol{F}_v$, $(Drag)_v$, and Cd_v may be determined as follows:

$$\boldsymbol{F}_v = \oiint 2\mu\boldsymbol{D}\cdot\boldsymbol{n}\, dS, \tag{15}$$

$$(Drag)_v = \frac{\boldsymbol{F}_v\cdot\boldsymbol{U}_\infty}{||\boldsymbol{U}_\infty||} \quad \text{and} \quad Cd_v = \frac{2(Drag)_v}{\rho||\boldsymbol{U}_\infty||^2 A}. \tag{16}$$

The closed integral in Equation 15 is performed over the surface of the solid object.

3 | RESULTS

In this section, we present results obtained using the numerical procedure described in Section 2. All the simulations discussed were run using an open-source 2D incompressible Navier-Stokes solver.[29] The penalization parameter in Equation 4 was set to $\lambda = 10^6$, and the multipole expansion in Equation A14a was truncated to $p = 12$.

Surface force computations for rigid objects are validated first using data available in the literature for impulsively started cylinders. This is followed by simulations of rigid fish-like profiles, where the drag force computed using surface forces is compared to the drag obtained using the penalization algorithm. Validation for deforming geometries (self-propelled swimmers) is done by comparing the unsteady acceleration computed using surface forces, to the acceleration determined using the penalization algorithm. We emphasize that the penalization algorithm yields just the resultant force acting on a body, whereas the surface force computations provide a detailed picture of pointwise forces acting on the object's surface.

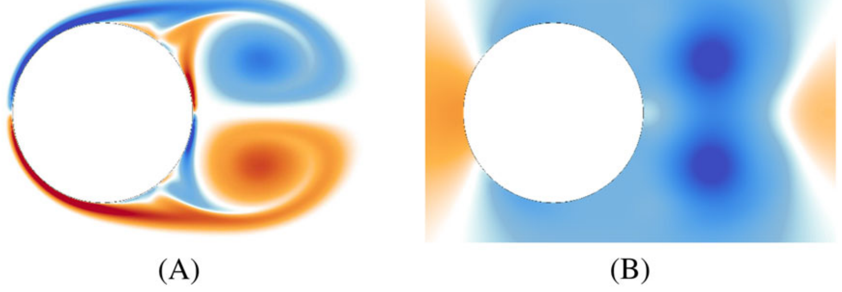


FIGURE 1 A, The vorticity field and B, the pressure field generated by flow over a stationary cylinder at $Re = 1000$. Red regions correspond to positive vorticity and pressure, and blue regions correspond to negative values

3.1 | Impulsively started cylinders

We first examine impulsively started flow over rigid cylinders, for diameter-based Reynolds numbers of $Re = 550$ and 1000 ($Re = 2\|\boldsymbol{U}_\infty\|R/\nu$). Both these simulations were run in a unit square domain using a uniform grid of 4096^2 points, with $\boldsymbol{U}_\infty = (0.1, 0)$ and $R = 0.05$. A snapshot of the resulting vorticity field, and the corresponding pressure field, is shown in Figure 1. To verify the accuracy of the spatial distribution of pressure, the surface pressure coefficient ($C_P = 2P/\rho\|\boldsymbol{U}_\infty\|^2$) is plotted as a function of the azimuthal angle in Figure 2 ($\theta = 0$ at the rear stagnation point). The pressure distribution compares well with reference data from Li et al,[40] at the 3 different time instances shown. To ensure that the velocity derivatives used for estimating the viscous forces (Section 2.3.2) are computed accurately, we examine the spatial distribution of vorticity on the lifted surface of the cylinder (Figure 3). The distribution compares well with reference data from Koumoutsakos and Leonard[36] at 2 different time instances, for both $Re = 550$ (Figure 3A) and $Re = 1000$ (Figure 3B). The temporal evolution of

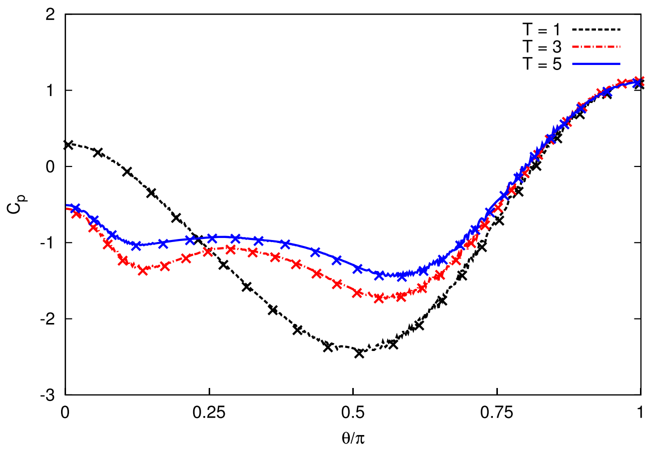


FIGURE 2 Pressure coefficient (C_P) distribution on the cylinder's surface for $Re = 550$, at 3 different nondimensionalized times ($T = t\|\boldsymbol{U}_\infty\|/2R$). Lines: current work, symbols: reference data from Li et al[40] [Colour figure can be viewed at wileyonlinelibrary.com]

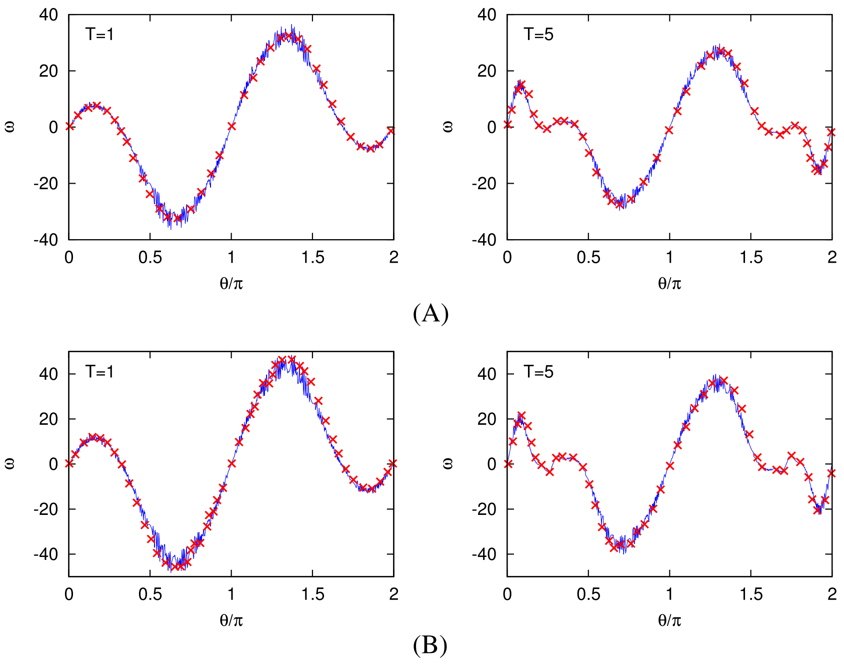
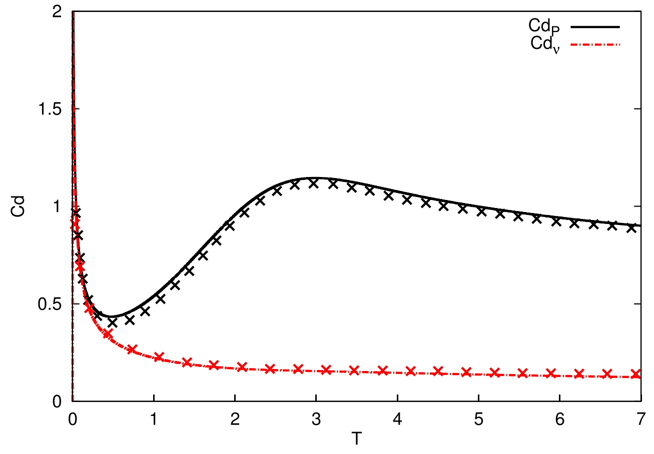
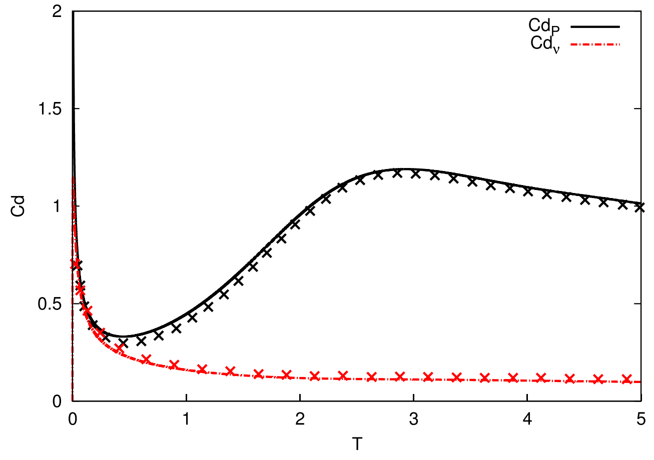


FIGURE 3 Spatial distribution of nondimensionalized vorticity computed on the lifted surface of the cylinders at 2 time instances ($T = 1$ and $T = 5$), for A, $Re = 550$ and B, $Re = 1000$. Lines represent current data, symbols correspond to reference data from Koumoutsakos and Leonard[36] [Colour figure can be viewed at wileyonlinelibrary.com]



(A)



(B)

FIGURE 4 Temporal evolution of pressure-drag coefficient (Cd_P) and viscous-drag coefficient (Cd_v) for impulsively started cylinders at A, $Re = 550$ and B, $Re = 1000$. Lines represent data from current work, symbols correspond to reference data from Koumoutsakos and Leonard[36] [Colour figure can be viewed at wileyonlinelibrary.com]

the pressure-induced and viscous contributions to the net drag force (Equations 13 and (16)) is shown in Figure 4 and agrees reasonably well with reference data from Koumoutsakos and Leonard.[36] The results discussed in this section suggest that the numerical algorithms described for determining pressure-induced and viscous forces work well for the case of rigid cylindrical objects.

3.2 | Impulsively started flow over a rigid streamlined object

To ensure that the surface force computations perform well for noncylindrical shapes, we examine impulsively started flow over a rigid streamlined object. The profile shape, shown in Figure 5, is based on a simplified 2D model of zebrafish.[10,14,22] The half width of the profile is described as follows:

$$w(s) = \begin{cases} \sqrt{2w_h s - s^2} & 0 \leqslant s < s_b \\ w_h - (w_h - w_t)\left(\frac{s-s_b}{s_t-s_b}\right) & s_b \leqslant s < s_t \\ w_t \frac{L-s}{L-s_t} & s_t \leqslant s \leqslant L \end{cases}, \tag{17}$$

where s is the arc length along the midline of the geometry, L is the body length, $w_h = s_b = 0.04$ L, $s_t = 0.95$ L, and $w_t = 0.01$ L. The relevant drag coefficient, the normalized time, and the Reynolds number are defined below:

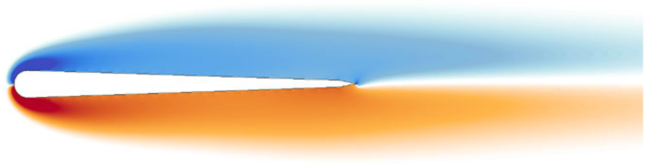


FIGURE 5 Vorticity field around a rigid fish-shaped object in a uniform flow ($T = 2$, $Re = 400$). Regions of positive vorticity are colored in red, and regions with negative values are colored in blue

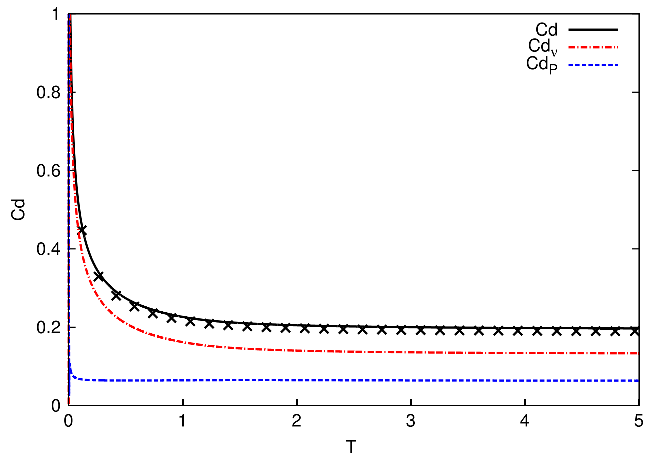


FIGURE 6 Temporal evolution of pressure-drag coefficient (Cd_p) and viscous-drag coefficient (Cd_v) for flow over the fish-shaped object shown in Figure 5. Lines: computed using surface forces, symbols: computed using the penalization algorithm (Equation 19) [Colour figure can be viewed at wileyonlinelibrary.com]

$$Cd = \frac{(Drag)_P + (Drag)_v}{\rho \|\boldsymbol{U}_\infty\|^2 L/2}, \tag{18a}$$

$$T = tL/\|\boldsymbol{U}_\infty\|, \tag{18b}$$

$$Re = \frac{\|\boldsymbol{U}_\infty\| L}{\nu}. \tag{18c}$$

Figure 6 shows the temporal evolution of drag coefficients for the body at a Reynolds number of 400, with $L = 0.1$ and $\boldsymbol{U}_\infty = (0.1, 0)$. The simulation was run on an adaptive grid with an effective resolution of 4096^2 grid points. The viscous-drag coefficient (Cd_v) was obtained by evaluating $\boldsymbol{D}(\boldsymbol{u})$ on a lifted surface, as described in Section 2.3.2. The net drag coefficient (the solid line in Figure 6) was computed as the sum of Cd_v and Cd_P (broken lines). The symbols in Figure 6 indicate drag determined by integrating the penalization term in Equation 3, over the entire volume in the computational domain (with $\boldsymbol{u}_s = 0$) [28]:

$$\boldsymbol{F}_{\text{penal}} = \iiint \lambda \chi \boldsymbol{u} \, dV \quad \text{and} \quad Cd_{\text{penal}} = \frac{2\boldsymbol{F}_{\text{penal}} \cdot \boldsymbol{U}_\infty}{\rho \|\boldsymbol{U}_\infty\|^2 L} \tag{19}$$

While Equation 19 provides a straightforward way of computing the net force acting on the object, it provides no information regarding the localized force distribution on the body and does not distinguish between pressure- and shear-based effects. The pointwise forces computed via Equations 11 and 14 help us overcome these shortcomings, providing us the capability to examine how time-varying changes in the flow-field impact flow-induced forces at various locations along the geometry.

In Figure 6, we observe that the static body experiences a large amount of drag at the start of the simulation ($T \ll 1$) owing to the impulsively started flow. As the flow approaches steady state ($T > 2$), the drag coefficients asymptote to constant values. We observe that the pressure-induced drag is small, even in the early stages of the simulation, which is a consequence of the streamlined shape of the body. This is not the case for blunt-shaped profiles (eg, cylinders: Figure 4), where the pressure-induced drag (or form drag) dominates throughout the simulation. Overall, the time evolution of Cd determined using surface forces compares well with that computed using Equation 19, which suggests that the numerical procedures outlined in Section 2 are suitable for use with rigid streamlined geometries.

3.3 | Self-propelled swimmers

To validate the surface force computations for dynamically deforming objects, simulations of self-propelled swimmers were performed by imposing time-varying deformations on 2D models of zebrafish.[10,14,22] The half width of the swimmer's profile is defined by Equation 17. The traveling wave that describes the lateral displacement of the swimmer's midline is given as[10,22]:

$$y_s(s,t) = \frac{4}{33}\left(s + \frac{L}{32}\right) sin\left(2\pi\left(\frac{s}{L} - \frac{t}{T_p}\right)\right), \tag{20}$$

where T_p represents the tail-beat period imposed on the swimmer. Over the first tail-beat period, the amplitude of y_s is increased from 0 to the expression shown in Equation 20, as a quarter sine wave in time. Further details regarding the deformation and discretization of the profile may be found in Gazzola et al.[22]

The simulation of the swimmer was conducted at a Reynolds number of 4000, on an adaptive grid with an effective resolution of 8192^2 grid points. This Re value corresponds to the swimming of adult zebrafish and is based on the length of the fish

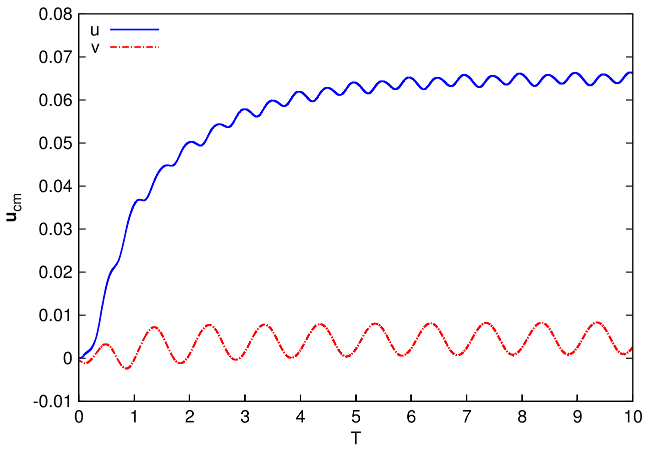


FIGURE 7 The horizontal and vertical components of the center-of-mass velocity for the swimmer [Colour figure can be viewed at wileyonlinelibrary.com]

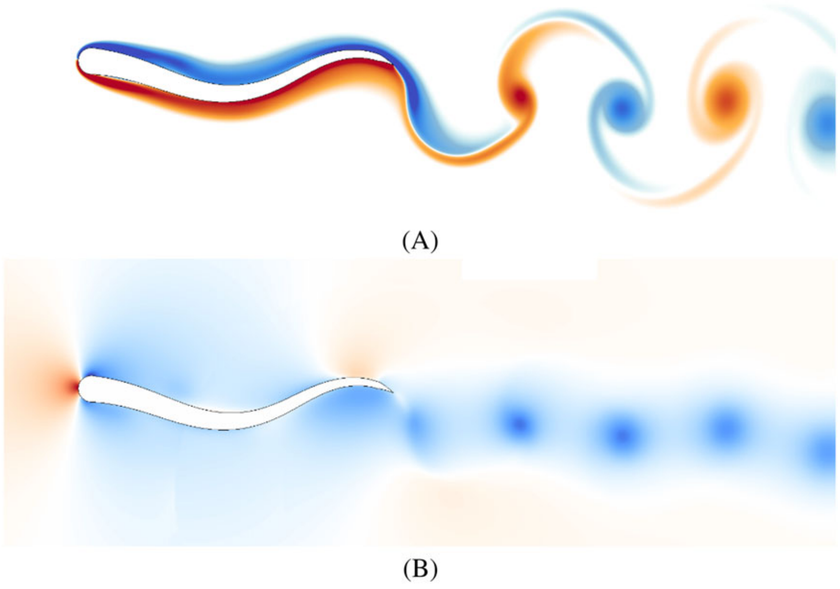


FIGURE 8 A, The vorticity field and B, the pressure field around the self-propelled swimmer. Regions of positive vorticity (and pressure) are colored in red, and regions with negative values are colored in blue

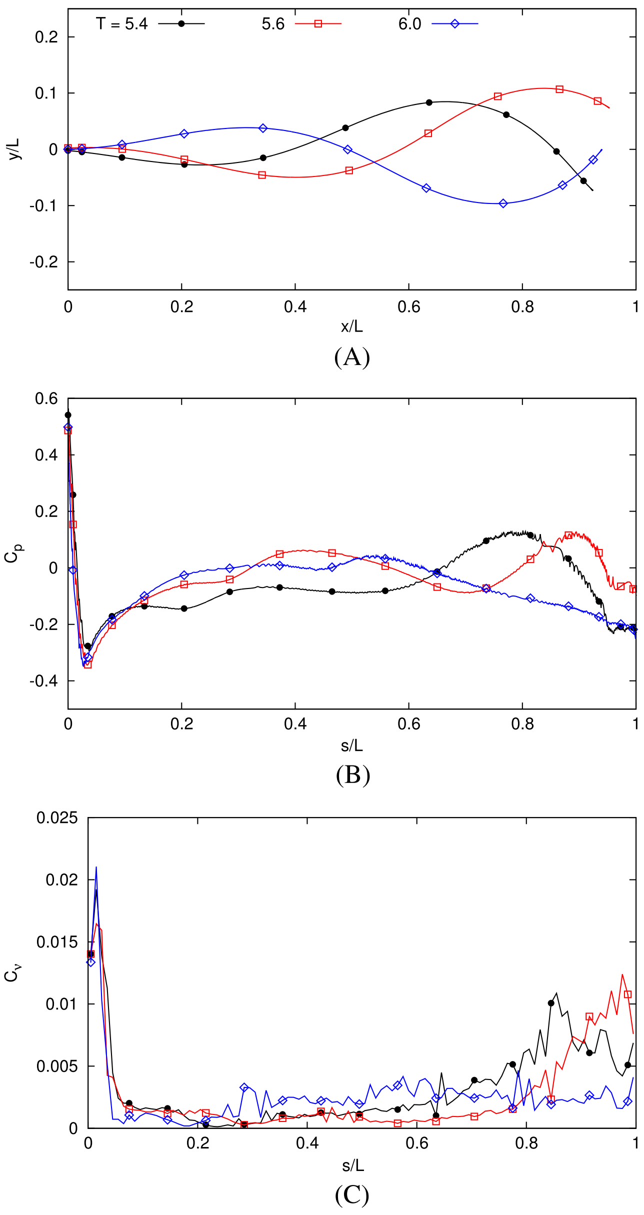


FIGURE 9 A, Fish midline shape at various instances during a tail-beat period. Distribution of B, pressure and C, shear-traction magnitude on the right lateral surface, as a function of the midline arclength s. The curves in panel C have been averaged over intervals of length 0.01 L [Colour figure can be viewed at wileyonlinelibrary.com]

$L = 0.1$, and the tail-beat period $T_p = 1$ ($Re = L^2/T_p\nu$). The resulting time evolution of the velocity of the center of mass of the swimmer ($\boldsymbol{u}_{cm}$) is shown in Figure 7. The swimmer accelerates from rest in the first few tail-beat periods ($T = t/T_p \leqslant 4$), before settling down to steady, periodically varying motion. The vorticity and pressure fields resulting from the simulation are shown in Figure 8. The pressure field was computed by evaluating Equation A14 with all grid nodes in the domain designated as target locations. A high pressure region develops in front of the head owing to the presence of a stagnation point. Acceleration of fluid around the head creates regions of low pressure on either side of the head, and the vortex-cores visible in Figure 8A give rise to regions of low pressure in the fish's wake (Figure 8B).

In Figure 9, we examine the spatial distribution of pressure and shear-based quantities on the surface of the swimmer's body. Figure 9A depicts the shape of the fish midline at 3 different instances during a tail-beat period. Figure 9B shows the corresponding distribution of pressure coefficient on the right lateral surface of the fish's body. Figure 9C depicts the magnitude of the shear (viscous) traction vector ($\|\boldsymbol{t}_v\| = \|2\mu\boldsymbol{D}\cdot\boldsymbol{n}\|$) on the right lateral surface. Both the pressure and the traction vector magnitude have been nondimensionalized with M/LT_P^2 (M represents the mass of the swimmer) to obtain the respective coefficients C_p and C_v shown in the plots. From Figure 9B and 9C, we can surmise that the pointwise contribution from pressure is much larger, $\max(C_p) \approx 0.5$, than the contribution from shear stress, $\max(C_v) \approx 0.021$, which is expected for moderately large

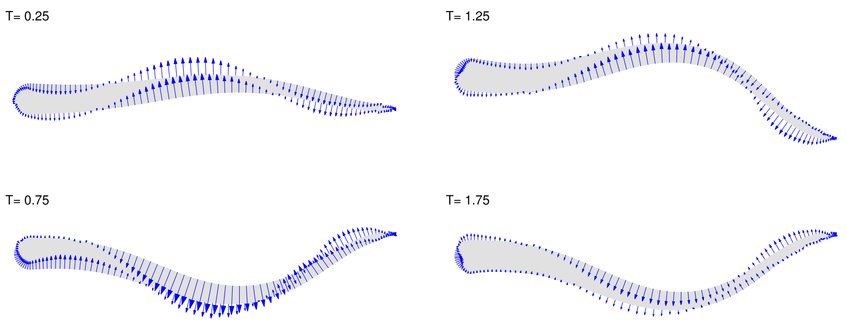


FIGURE 10 Pointwise force distribution on the swimmer's body, at various time instances during the simulation [Colour figure can be viewed at wileyonlinelibrary.com]

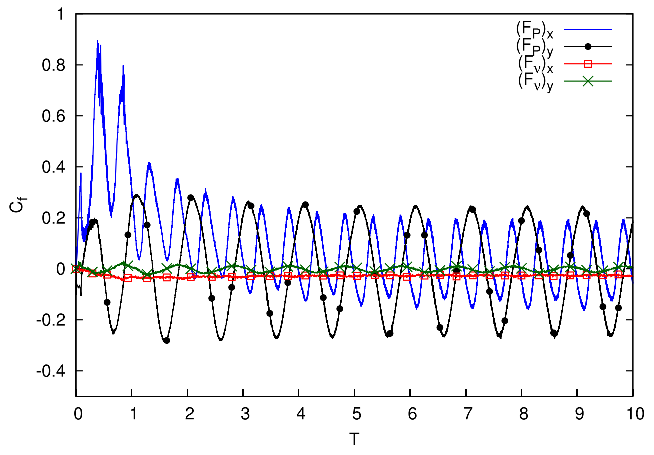


FIGURE 11 Time evolution of the resultant surface forces experienced by a self-propelled swimmer at $Re = 4000$. The forces have been normalized by ML/T_P^2 to obtain the force coefficient C_f [Colour figure can be viewed at wileyonlinelibrary.com]

Reynolds numbers ($Re = 4000$). The shear stress contribution is largest close to the tip of the head, followed by a pronounced drop along the midsection of the body, and a recovery towards the tail-end. The most dominant pressure contribution occurs at the head tip, followed by a persistent negative C_p region at the side of the head and periodically varying C_p along the rest of the body. The pointwise forces that emerge as a consequence of these stress distributions are depicted in Figure 10, as well as in an animation provided as part of the supplementary materials (Movie S1). When accelerating from rest ($T \leqslant 0.25$), the front region of the head experiences relatively little force. However, as the swimmer picks up speed ($T \geqslant 0.75$), flow-stagnation in front of the head generates drag-inducing forces. We emphasize that such a detailed picture of the dynamics cannot be obtained via techniques that involve surface or volume integrals,[38,39] which instead provide the *net* acceleration at any given instant. To ascertain that the surface force vectors shown in Figure 10 are accurate, the net acceleration computed from the force distribution is validated later on in Figure 12.

The main drawback of the technique presented here is the computational overhead associated with solving the pressure Poisson equation. This accounts for approximately 22% of the simulation time step. In comparison, the velocity solve in the entire domain (Equation 5) requires close to 58% of the time step. For the velocity solve, approximately 20% of the simulation time step is spent in generating the quadtree structure (forward pass), whereas the remaining 38% is spent in evaluating the multipole expansion (reverse pass). In the case of pressure computations, the forward pass takes 21% of the simulation time, whereas the reverse pass, involving just the body points as targets, takes about 1% of the time. We note that the time taken for generating the tree structure is similar in both cases, which is expected, since the number and distribution of source terms for both quantities depend on velocity derivatives.

The time evolution of the resultant of the surface forces (Equations 12 and 15) is shown in Figure 11. We observe that the contribution of viscous forces is small compared to the contribution of pressure-induced forces, which matches our observation in Figure 9. The spikes in $(F_P)_x$ for $T < 1$ suggest that the pressure distribution created by undulations of the swimmer plays a major role in accelerating the body from rest. For quantitative validation of the pressure-induced and viscous force computations, the horizontal (x) and vertical (y) components of net acceleration determined using surface forces are compared to

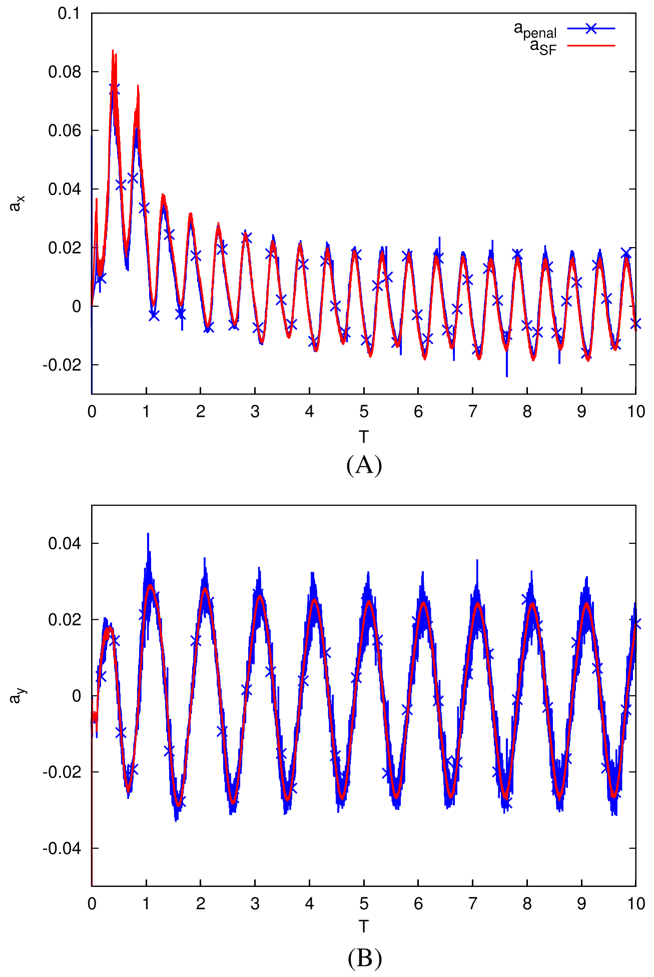


FIGURE 12 Comparison of A, horizontal acceleration and B, vertical acceleration of the self-propelled swimmer, computed using Equations 21 and 22 [Colour figure can be viewed at wileyonlinelibrary.com]

the acceleration predicted by the penalization algorithm. The acceleration at time step "n" from the penalization algorithm is recovered from the object's center-of-mass velocity using second-order centered difference:

$$\boldsymbol{a}^n_{penal} = \frac{\boldsymbol{u}^{n+1}_{CM} - \boldsymbol{u}^{n-1}_{CM}}{2\Delta t} \tag{21}$$

The acceleration corresponding to surface forces is computed as follows:

$$\boldsymbol{a}^n_{SF} = \frac{\boldsymbol{F}^n_P + \boldsymbol{F}^n_\nu}{M}, \tag{22}$$

where M is the total mass of the solid object. The temporal evolution of $\boldsymbol{a}_{penal}$ and $\boldsymbol{a}_{SF}$ is shown in Figure 12. The plots indicate that the net acceleration computed using surface forces compares well with that determined using the penalization algorithm. The noise observed in the case of $\boldsymbol{a}_{penal}$ can be attributed to the use of finite differences for computing the temporal derivative (Equation (21)). The spikes in a_x for $T < 1$ correlate with the large $(F_P)_x$ values observed in Figure 11. At later stages in the simulation, both a_x and a_y oscillate about a mean value of 0, which corresponds to the horizontal and vertical velocities approaching a steady mean value. Overall, the trends in a_x and a_y follow those observed for $(F_P)_x$ and $(F_P)_y$ in Figure 11. To ensure that the algorithm works well on coarse grids, we compare the swimmer's net acceleration computed using surface forces on 4 different grid resolutions (Figure 13). The acceleration exhibits increased levels of noise at lower resolutions; however, the values compare well to the data computed at higher resolutions. This result, as well as the good agreement between $\boldsymbol{a}_{penal}$ and $\boldsymbol{a}_{SF}$ in Figure 12, confirms that the surface force computations described in Section 2 work well for the case of dynamically deforming solid objects.

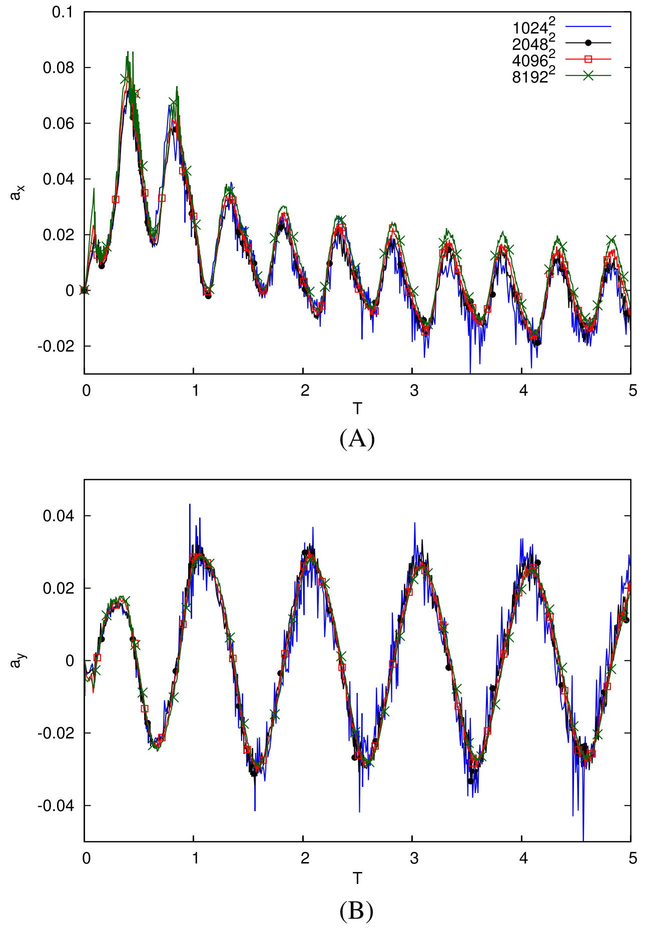


FIGURE 13 A, Horizontal acceleration and B, vertical acceleration of the self-propelled swimmer, computed using Equation 22 for simulations conducted at 4 different grid resolutions [Colour figure can be viewed at wileyonlinelibrary.com]

4 | SUMMARY

In this paper, we describe numerical procedures for determining flow-induced surface forces on rigid and deforming bodies, when using vortex methods combined with Brinkman penalization. The method presented may be considered to be an extension of the technique described in Koumoutsakos and Leonard,[36] which used vorticity and its flux at the boundary to determine friction- and pressure-based forces in simulations involving 2-dimensional cylinders. The novelty of the current approach is that it enables the computation of force distribution along a body surface in the context of a penalization method, which relies on a non–body-fitted framework and is especially suitable for complex deforming geometries. Moreover, the method proposed allows us to quantify explicitly the 2 main components of drag force, namely, the pressure- and the friction-drag. The pressure-Poisson equation is solved using a fast tree-code algorithm based on multipole expansions. Viscous forces are computed by evaluating the strain-rate tensor either on the object's surface or on a lifted-surface, depending on whether a deforming or a rigid object is being simulated. Numerical tests involving impulsively started cylinders, a streamlined rigid fish-shaped body and a self-propelled swimmer, are used to assess the efficacy of the method described. For the case of impulsively started cylinders, the spatial distribution of surface pressure compares well with the results of benchmark simulations. Moreover, the temporal evolution of the pressure-induced and viscous drag coefficients shows good agreement with reference data. For streamlined fish-shaped profiles, drag measurements computed using surface forces match those determined using the penalization algorithm. In simulations of self-propelled swimmers, the net unsteady acceleration calculated using the surface force computations agrees well with the acceleration determined from the penalization algorithm. The tests presented indicate that the numerical procedures described are quite effective for determining pointwise surface forces on complex, temporally evolving geometries. Future work involves the extension of this method to 3-dimensional flows.

ACKNOWLEDGMENTS

We gratefully acknowldge support by the European Research Council Advanced Investigator Award (341117) and the Swiss National Science Foundation Sinergia Award (CRSII3_147675). This work used computational resources granted by the Swiss National Supercomputing Centre (CSCS) under project ID "s436" and "s658".

REFERENCES

1. Gray J. How fishes swim. *Sci Am*. 1957;197:29-35.
2. Dubois AB, Ogilvy CS. Forces on the tail surface of swimming fish: thrust, drag and acceleration in bluefish (Pomatomus saltatrix). *J Exp Biol*. 1978;77(1):225-241.
3. Lighthill MJ. Aquatic animal propulsion of high hydromechanical efficiency. *J Fluid Mech*. 1970;44:265-301.
4. Wu TYT. Hydromechanics of swimming propulsion. Part 3. Swimming and optimum movements of slender fish with side fins. *J Fluid Mech*. 1971;46:545-568.
5. Dabiri JO. On the estimation of swimming and flying forces from wake measurements. *J Expt Biol*. 2005;208(18):3519-3532.
6. Dabiri JO, Bose S, Gemmell BJ, Colin SP, Costello JH. An algorithm to estimate unsteady and quasi-steady pressure fields from velocity field measurements. *J Expt Biol*. 2014;217(3):331-336.
7. Lauder GV. Swimming hydrodynamics: ten questions and the technical approaches needed to resolve them. *Exp Fluids*. 2011;51(1):23-35.
8. Liu H, Wassersug R, Kawachi K. A computational fluid dynamics study of tadpole swimming. *J Exp Biol*. 1996;199(6):1245-1260.
9. Liu H, Wassersug R, Kawachi K. The three-dimensional hydrodynamics of tadpole locomotion. *J Exp Biol*. 1997;200(22):2807-2819.
10. Carling J, Williams TL, Bowtell G. Self-propelled anguilliform swimming: simultaneous solution of the two-dimensional Navier-Stokes equations and Newton's laws of motion. *J Exp Biol*. 1998;201(23):3143-3166.
11. Hirt CW, Amsden AA, Cook JL. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *J Comput Phys*. 1974;14(3):227-253.
12. Hughes TJR, Liu WK, Zimmermann TK. Lagrangian-Eulerian finite element formulation for incompressible viscous flows. *Comput Meth Appl Mech Eng*. 1981;29(3):329-349.
13. Liu H, Kawachi K. A numerical study of undulatory swimming. *J Comput Phys*. 1999;155(2):223-247.
14. Kern S, Koumoutsakos P. Simulations of optimized anguilliform swimming. *J Exp Biol*. 2006;209:4841-4857.
15. Peskin CS. Flow patterns around heart valves: a numerical method. *J Comput Phys*. 1972;10(2):252-271.
16. Peskin CS. The immersed boundary method. *Acta Numerica*. 2002;11:479-517.
17. Mittal R, Iaccarino G. Immersed boundary methods. *Ann Rev Fluid Mech*. 2005;37:239-261.
18. Gilmanov A, Sotiropoulos F. A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies. *J Comput Phys*. 2005;207(2):457-492.
19. Hieber SE, Koumoutsakos P. An immersed boundary method for smoothed particle hydrodynamics of self-propelled swimmers. *J Comput Phys*. 2008;227(19):8636-8654.
20. Borazjani I, Sotiropoulos F. Numerical investigation of the hydrodynamics of anguilliform swimming in the transitional and inertial flow regimes. *J Exp Biol*. 2009;212(4):576-592.
21. Angot P, Bruneau CH, Fabrie P. A penalization method to take into account obstacles in incompressible viscous flows. *Numer Math*. 1999;81:497-520.
22. Gazzola M, Chatelain P, van Rees WM, Koumoutsakos P. Simulations of single and multiple swimmers with non-divergence free deforming geometries. *J Comput Phys*. 2011;230:7093-7114.
23. Bergmann M, Iollo A. Modeling and simulation of fish-like swimming. *J Comput Phys*. 2011;230:329-348.
24. McCracken MF, Peskin CS. A vortex method for blood flow through heart valves. *J Comput Phys*. 1980;35(2):183-205.
25. Koumoutsakos P. Inviscid axisymmetrization of an elliptical vortex. *J Comput Phys*. 1997;138(2):821-857.
26. Koumoutsakos P, Leonard A, Pépin F. Boundary conditions for viscous vortex methods. *J Comput Phys*. 1994;113(1):52-61.
27. Cottet GH, Koumoutsakos P, Salihi MLO. Vortex methods with spatially varying cores. *J Comput Phys*. 2000;162:164-185.
28. Coquerelle M, Cottet GH. A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies. *J Comput Phys*. 2008;227:9121-9137.
29. Rossinelli D, Hejazialhosseini B, van Rees WM, Gazzola M, Bergdorf M, Koumoutsakos P. MRAG-I2D: multi-resolution adapted grids for remeshed vortex methods on multicore architectures. *J Comput Phys*. 2015;288:1-18.
30. Gazzola M, Hejazialhosseini B, Koumoutsakos P. Reinforcement learning and wavelet adapted vortex methods for simulations of self-propelled swimmers. *SIAM J Sci Comput*. 2014;36(3):B622-B639.
31. van Rees WM, Gazzola M, Koumoutsakos P. Optimal shapes for anguilliform swimmers at intermediate Reynolds numbers. *J Fluid Mech*. 2013;722:R3 1-12.
32. Gazzola M, van Rees WM, Koumoutsakos P. C-start: optimal start of larval fish. *J Fluid Mech*. 2012;698:5-18.
33. Gazzola M, Mimeau C, Tchieu AA, Koumoutsakos Petros. Flow mediated interactions between two cylinders at finite Re numbers. *Phys Fluids*. 2012;24(4):1-17.
34. van Rees WM, Novati G, Koumoutsakos P. Self-propulsion of a counter-rotating cylinder pair in a viscous fluid. *Phys Fluids*. 2015;27(6):1-11.

35. Towers JD. Finite difference methods for approximating Heaviside functions. *J Comput Phys*. 2009;228:3478-3489.
36. Koumoutsakos P, Leonard A. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *J Fluid Mech*. 1995;296:1-38.
37. Greengard L, Rokhlin V. A fast algorithm for particle simulations. *J Comput Phys*. 1987;73:325-348.
38. Mimeau C, Gallizio F, Cottet GH, Mortazavi I. Vortex penalization method for bluff body flows. *Int J Numer Meth Fluids*. 2015;79(2):55-83.
39. Noca F, Shiels D, Jeon D. A comparison of methods for evaluating time-dependent fluid dynamic forces on bodies, using only velocity fields and their derivatives. *J Fluids Struct*. 1999;13(5):551-578.
40. Li Y, Shock R, Zhang R, Chen H. Numerical study of flow past an impulsively started cylinder by the lattice-Boltzmann method. *J Fluid Mech*. 2004;519:273-300.
41. Hockney RW. A fast direct solution of poisson's equation using fourier analysis. *J ACM*. 1965;12(1):95-113.
42. Stüben K, Trottenberg U. Multigrid methods: fundamental algorithms, model problem analysis and applications. In: Hackbusch W, Trottenberg U, eds. *Multigrid Methods*, Lecture Notes in Mathematics, vol. 960. Berlin, Heidelberg: Springer, Berlin, Heidelberg; 1982:1-176.
43. Gupta MM, Kouatchou J, Zhang J. Comparison of second- and fourth-order discretizations for multigrid poisson solvers. *J Comput Phys*. 1997;132(2):226-232.
44. Appel AW. An efficient program for many-body simulation. *SIAM J Sci Stat Comput*. 1985;6:85-103.
45. Rokhlin V. Rapid solution of integral equations of classical potential theory. *J Comput Phys*. 1985;60(2):187-207.
46. Barnes J, Hut P. A hierarchical *O*(*NlogN*) force-calculation algorithm. *Nature*. 1986;324:446-449.
47. Pinchover Y, Rubinstein J. *An Introduction to Partial Differential Equations*. Cambridge: Cambridge University Press; 2005.
48. Greengard L. The numerical solution of the N-Body problem. *Comput Phys*. 1990;4(2):142-152.
49. Pépin F. Simulation of the Flow Past an Impulsively Started Cylinder Using a Discrete Vortex Method. *Ph.D. Thesis*: California Institute of Technology; 1990.

SUPPORTING INFORMATION

Additional Supporting Information may be found online in the supporting information tab for this article.

How to cite this article: Verma S, Abbati G, Novati G, Koumoutsakos P. Computing the force distribution on the surface of complex, deforming geometries using vortex methods and Brinkman penalization. *Int J Numer Meth Fluids*. 2017;0:1-18. https://doi.org/10.1002/fld.4392

APPENDIX A: NUMERICAL SOLUTION OF THE PRESSURE POISSON EQUATION

A.1. The Green's function

There are several ways of numerically solving the Poisson's equation shown in Equation 10, for instance, using Fourier-based fast Poisson solvers,[41] multigrid methods,[42,43] or tree-code algorithms.[37,44–46] For solving the Poisson's equation on the multiresolution adaptive grids[29] used in our simulations, we use the tree-code algorithm, which requires knowledge of the Green's function (or the fundamental solution) for the Laplacian.

The free-space Green's function $G\left(\boldsymbol{x},\boldsymbol{x}'\right)$ for the Laplacian is the solution of the following equation, with the Neumann boundary condition specified on a circular domain dS, with radius $r \to \infty$:

$$\nabla^2 G\left(\boldsymbol{x},\boldsymbol{x}'\right) = \delta\left(\boldsymbol{x}-\boldsymbol{x}'\right), \tag{A1}$$

$$\nabla G\left(\boldsymbol{x},\boldsymbol{x}'\right)\cdot\boldsymbol{n} = 0 \quad \text{on} \quad dS. \tag{A2}$$

Here, $\delta\left(\boldsymbol{x}-\boldsymbol{x}'\right)$ represents the Dirac delta function, and $\boldsymbol{n}$ is the outward unit normal. Physically, the Neumann boundary condition in Equation A2 corresponds to the pressure-induced forces being 0 on the domain boundary, as $r \to \infty$. The Green's function that satisfies Equations A1 and A2 in 2 dimensions is

$$G\left(\boldsymbol{x},\boldsymbol{x}'\right) = \frac{1}{2\pi}\log\left(\|\boldsymbol{x}-\boldsymbol{x}'\|\right). \tag{A3}$$

This fundamental solution can be used to solve the inhomogeneous pressure-Poisson equation, by computing its convolution with the forcing term on the right hand side of Equation 10:

$$P(\boldsymbol{x}) = \int G\left(\boldsymbol{x},\boldsymbol{x}'\right)\left(-\rho\left(\nabla\boldsymbol{u}^T : \nabla\boldsymbol{u}\right) + \rho\lambda'\nabla\cdot(\chi(\boldsymbol{u}_s - \boldsymbol{u}))\right)d\boldsymbol{x}', \tag{A4}$$

$$= \int G\left(\boldsymbol{x},\boldsymbol{x}'\right)f(\boldsymbol{x}')d\boldsymbol{x}'. \tag{A5}$$

Here, $f(\boldsymbol{x}')$ represents the right-hand side of the Poisson's equation, $\boldsymbol{u}$, $\boldsymbol{u}_s$, and χ are functions of $\boldsymbol{x}'$, and the domain of integration encompasses only the nonzero source points $\boldsymbol{x}'$ in the entire computational volume. Equation A5 may be derived as follows:

$$P(\boldsymbol{x}) = \int \delta\left(\boldsymbol{x}-\boldsymbol{x}'\right)P\left(\boldsymbol{x}'\right)d\boldsymbol{x}' = \int \nabla^2 G\left(\boldsymbol{x},\boldsymbol{x}'\right)P\left(\boldsymbol{x}'\right)d\boldsymbol{x}', \tag{A6}$$

$$= \int G\left(\boldsymbol{x},\boldsymbol{x}'\right)\nabla^2 P\left(\boldsymbol{x}'\right)d\boldsymbol{x}' = \int G\left(\boldsymbol{x},\boldsymbol{x}'\right)f(\boldsymbol{x}')d\boldsymbol{x}'. \tag{A7}$$

To obtain Equation A7 from Equation A6, we have made use of Green's second identity[47] and zero Neumann boundary conditions for both $G\left(\boldsymbol{x},\boldsymbol{x}'\right)$ and $P\left(\boldsymbol{x}'\right)$. The problem is well posed; however, the solution that we obtain for $P(\boldsymbol{x})$ is not unique, in that any additive constant to the solution also satisfies the original Poisson's equation (Equation (10)). Since the flow-induced forces depend primarily on the gauge pressure, this additive constant is set to 0 to obtain the relevant solution.

Discretizing the integral in Equation A5 yields the following summation:

$$P(\boldsymbol{x}) = \sum_{\boldsymbol{x}'} G\left(\boldsymbol{x},\boldsymbol{x}'\right)f(\boldsymbol{x}')h^2(\boldsymbol{x}'), \tag{A8}$$

where h^2 represents the area of the grid cell at location $\boldsymbol{x}'$. Note that h is a function of $\boldsymbol{x}'$, as the cell size may vary spatially when using adaptive grids. Equation A8 sums up contributions from source points located throughout the domain (hence the summation over $\boldsymbol{x}'$), to compute the pressure at location $\boldsymbol{x}$. Evaluating this sum directly is computationally intractable even for problems that are relatively modest in size, and a tree-code[46] combined with multipole expansions[37] is used to speed-up the computation.

A.2. Multipole expansion

Tree-codes and multipole methods have been used extensively for solving "N-body" problems involving gravitational and electric fields,[37,46,48] and for fluid simulations based on vortex dynamics.[22,36,49] These methods can be used to reduce the computational complexity of the summation in Equation A8 from $O(N^2)$ to $O(N\log N)$[46] or to $O(N)$,[37,48] depending on the specific algorithm used. The computational savings result from the combined use of a hierarchical quadtree data structure, and a truncated series expansion for $G\left(\boldsymbol{x},\boldsymbol{x}'\right)$. To obtain the multipole expansion for the logarithmic function in Equation A3, the coordinates $\boldsymbol{x}$ and $\boldsymbol{x}'$ are expressed as complex numbers $z = x + iy$ and $z' = x' + iy'$. To facilitate series expansion of the resulting expression ($\log\left(\|z-z'\|\right)$), we express the difference $z - z'$ of polar coordinates $re^{i\theta}$ (where $r = \|z-z'\|$ and $\theta = \arg\left(z-z'\right)$). The complex logarithm $\log\left(z-z'\right)$ and the real-valued function in Equation A3 are related as follows:

$$\log\left(z-z'\right) = \log(r) + i\theta, \tag{A9}$$

$$\Rightarrow \log\left(\|\boldsymbol{x}-\boldsymbol{x}'\|\right) = R\{\log\left(z-z'\right)\}. \tag{A10}$$

Equation A10 suggests that $G\left(\boldsymbol{x},\boldsymbol{x}'\right)$ can be represented as the real part of the series expansion for $\log\left(z-z'\right)$:

$$\log\left(z-z'\right) = \log\left(z\left(1-\frac{z'}{z}\right)\right), \tag{A11}$$

$$= \log(z) - \sum_{k=1}^{\infty}\frac{1}{k}\left(\frac{z'}{z}\right)^k \quad \text{iff} \quad \|z'\| < \|z\|. \tag{A12}$$

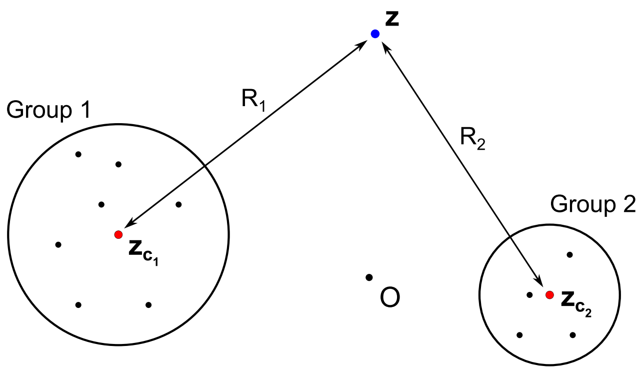


FIGURE A1 Schematic showing pressure source-points (black dots) grouped together into 2 distinct clusters. R_j denotes the distance from the center of the clusters $\left(z_{c_j}\right)$ to the point of pressure evaluation (z). O represents the origin in the complex plane [Colour figure can be viewed at wileyonlinelibrary.com]

Assuming that the computation for $P(z)$ involves N source particles, we combine Equations A3, A8, A10, and A12 to get

$$P(z) = \frac{1}{2\pi}\sum_{i=1}^{N} f(z_i')h^2(z_i')R\{\log(z)\} - \frac{1}{2\pi}\sum_{i=1}^{N} f(z_i')h^2(z_i')R\left\{\sum_{k=1}^{\infty}\frac{1}{k}\left(\frac{z_i'}{z}\right)^k\right\}. \tag{A13}$$

The summation in the first term in Equation A13 simplifies to a constant value, which represents the total contribution S from all source points. The summation signs in the second term are interchanged, and the summation over k is truncated to p terms:

$$P(z) = \frac{S}{2\pi}R\{\log(z)\} - \sum_{k=1}^{p} R\left\{\frac{\alpha_k}{z^k}\right\}, \tag{A14a}$$

$$S = \sum_{i=1}^{N} f(z_i')h^2(z_i'), \tag{A14b}$$

$$\alpha_k = \sum_{i=1}^{N} f(z_i')h^2(z_i')\frac{z_i'^k}{k}. \tag{A14c}$$

Note that S and the coefficients α_k are evaluated and stored in a single pass through the quadtree, as they do not depend on z. The choice of truncation parameter "p" depends on the desired level of accuracy.[37]

The multipole expansion in Equation A14 and the hierarchical quadtree allow us to group source points together, as depicted in Figure A1. Each group is perceived as a single source point (the red dots located at z_{c_j} in Figure A1) at sufficiently large distances.[37,44,46,48] Because of the linear nature of Equation A8, the contributions from individual groups can be combined to obtain $P(z)$:

$$P(z) = P(z)_{\text{Group}_1} + P(z)_{\text{Group}_2} + P(z)_{\text{Group}_3} + \dots \tag{A15}$$

The group contributions $P(z)_{\text{Group}_j}$ can be evaluated by replacing z with $\left(z - z_{c_j}\right)$, and z' with $\left(z' - z_{c_j}\right)$ in Equation A14 and by limiting the summation over "i" to the sources contained within each group.

The multipole expansions for parent clusters (ie, large clusters composed of smaller clusters) can be generated by combining their childrens' expansions. This reduces the computational cost, since we no longer need to repeatedly evaluate the contributions from all source points enclosed within larger clusters. The multipole expansion of a child-cluster centered at z_{c_j} can be shifted to the appropriate parent-cluster's center at z_p, using an exact formula[37]:

$$P(z)_{\text{Group}_j} = \frac{S_j}{2\pi}R\left\{\log\left(z - z_{c_j}\right)\right\} - \sum_{k=1}^{p} R\left\{\frac{\alpha_k}{\left(z - z_{c_j}\right)^k}\right\}, \tag{A16}$$

$$= \frac{S_j}{2\pi}R\left\{\log\left(z - z_p\right)\right\} - \sum_{k=1}^{p} R\left\{\frac{\beta_k}{\left(z - z_p\right)^k}\right\}, \tag{A17}$$

where

$$\beta_k = -\frac{S_j\left(z_{c_j} - z_p\right)^k}{k} + \sum_{l=1}^{k} \alpha_l \left(z_{c_j} - z_p\right)^{k-l} \binom{k-1}{l-1}. \tag{A18}$$

Starting from the finest level in the quadtree, Equations A17 and A18 can be used to generate the expansions for all parent nodes recursively. Further details regarding the implementation of tree-codes may be found in Greengard and Rokhlin,[37] Barnes and Hut,[46] and Greengard.[48]