An octree-based immersogeometric approach for modeling inertial migration of particles in channels

Songzhe Xu[a], Boshun Gao[a], Alec Lofquist[a], Milinda Fernando[b], Ming-Chen Hsu[a], Hari Sundar[b,*], Baskar Ganapathysubramanian[a,*]

[a]*Department of Mechanical Engineering, Iowa State University, 2080 Black Engineering, Ames, IA 50011, USA*
[b]*School of Computing, University of Utah, 50 Central Campus Dr, Salt Lake City, UT 84112, USA*

Abstract

In this paper, we develop a scalable, adaptively refined, octree-based finite element approach with immersogeometric analysis to track inertial migration of particles in microchannels. Fluid physics is modeled using a residual-based variational multiscale method, and the particle movement is modeled as rigid body motion. A parallel, hierarchically refined octree mesh is employed as the background mesh, on which a variationally consistent immersogeometric formulation is adopted for tracking the particle motion in the fluid. Adaptations of immersogeometric analysis on an octree background mesh are developed to enable efficient searching of background element for a given surface quadrature point, as well as a distribution of surface quadrature points over processors to reduce memory overhead and better parallelize the surface assembly. An octree-based adaptive mesh refinement algorithm adapted to in-out test in the immersogeometric approach is also developed. The validation of our octree-based immersogeometric approach is carried out using a benchmark case of a sphere settling in quiescent fluid, with good agreement presented. In addition, good strong (and weak) scalability on supercomputing resources for this benchmark case up to 16,384 processes is demonstrated. The proposed method is further deployed for exploring particle migration in straight and converging-diverging channels. This example illustrates the potential of the octree-based immersogeometric approach for efficiently tracking particle motion in complex channel flows – a problem with a diverse array of applications.

Keywords: Octree-based mesh; Adaptive mesh refinement; Immersogeometric analysis; Particle inertial migration

*Corresponding authors
Email addresses: hari@cs.utah.edu (Hari Sundar), baskarg@iastate.edu (Baskar Ganapathysubramanian)

The final publication is available at Elsevier via https://doi.org/10.1016/j.compfluid.2020.104764

Contents

1. Introduction

Control and localization of particles (e.g. cells and precipitates) in flow is useful in biological processing, chemical reaction control, and for creating structured materials [1, 2]. One promising approach to control localization (or 'focusing') of particles of different sizes is via sequential placement of obstacles in microchannels. This is based on the idea that obstacles produce different forces on particles with distinct sizes when inertial flow conditions are considered (i.e., when the Reynolds number based on the channel hydraulic diameter is greater than 5) [3]. However, highly

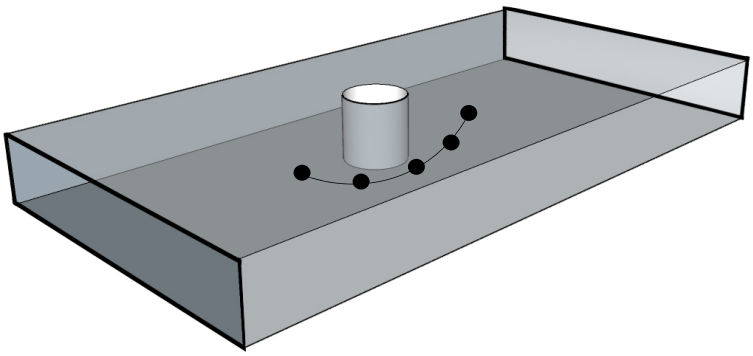


Figure 1: Canonical problem: A rigid particle traversing a microchannel decorated with a pillar obstacle under inertial flow conditions

accurate force calculations are required to design devices that can exploit these small variations in forces, which essentially becomes a comput55ational exercise. The availability of validated approaches to compute these dynamics can enable diverse applications in separation, concentration, and sorting of cells and biomolecules with high specificity.

The general problem of force (and trajectory) calculation on a moving particle in channel flow can be framed as a canonical problem. The canonical problem is to track the lateral migration of a single, rigid particle as it traverses a microchannel that is decorated with a pillar obstacle (see Figure 1). We consider a rigid particle (of size a) moving in an incompressible fluid inside a channel. We are interested in tracking the motion and lateral forces acting on this particle. This is a challenging computational problem due to the full fluid–solid coupling, associated small time step sizes, and adaptive (re)meshing requirements. The full fluid–solid coupling is needed due to the finite Reynolds number ($5 \leq Re \leq 100$) which necessitates solving the full Navier–Stokes equations. Furthermore, the construction of migration maps for particles (i.e. how does a particle traverse a decorated channel, which is a function of initial particle release locations, particle sizes and flow rates) requires several hundreds of simulations tracking individual particles under various configurations (of release locations and particle sizes).

To track particles in decorated channels, immersed boundary methods [4, 5] are a popular class of approaches, since these methods exhibit greater geometric flexibility than their boundary-fitted counterparts, and simplify (re)meshing process especially when the object is moving. The immersed boundary method embeds the solid geometry into a background mesh without the need for conforming the solid and fluid meshes. As a result, it becomes computationally convenient to track the motion of arbitrary particles while avoiding a cumbersome boundary-fitted (re)meshing process. In the context of finite elements, several adaptations of immersed methods have been explored for the simulation of fluid interacting with moving objects [6–13]. Among these immersed methods, immersogeometric analysis [14–18] (IMGA) is a promising approach for accurately predicting flow results by faithfully capturing the immersed geometry using adaptively refined quadra-

ture rules in the intersected elements, and weakly enforcing the Dirichlet boundary conditions on the surface of the immersed object using an extension of Nitsche's method [19]. Our prior work has shown that the IMGA is a viable approach to track particles in microchannels [20], and we extend the IMGA approach to 3D simulations in this work.

The IMGA approach incorporates a residual-based variational multiscale (VMS) formulation [21] to model fluid physics, which has been a successful approach to model complex flow phenomena. The VMS approach is analogous to a large eddy simulation (LES) model which uses variational projections in place of the traditional filtered equations in LES and focuses on modeling the fine-scale equations. The VMS approach does not employ any eddy viscosity, and has been successfully used to perform accurate flow condition agnostic (laminar or turbulent) simulations. It has been extended to a wide range of engineering applications, such as buoyancy driven flows, particle laden flows, fluid–structure interaction, multiphase and free-surface flows, space-time thermal flows, magnetohydrodynamic flows, and compressible flows [22–30]. While the IMGA is a promising approach to model particle migration in channels, it is challenging to optimize the computational efficiency on an unstructured background mesh. Thus, in this work, we proposed a new computational framework that performs large-scale finite element computations using IMGA on an octree-based background mesh due to its convenience and efficiency of fast adaptive mesh refinement and partitioning compared with traditional unstructured meshing approaches[1].

Octree-based meshing has been successfully applied to finite element computations for many engineering problems [33–38]. Specifically in this work, we employ the optimized parallel octree-based meshing library, Dendro, which has been deployed for simulating binary black hole inspiral, solving PDEs in 4D space-time octrees, reconstructing the motion of the ventricular walls in cardiac images, and computing information theoretic similarity measures for medical images [39–43]. Some of the features of Dendro include a bottom-up construction of octrees and a 2:1 balancing on distributed architectures [44, 45], and a space filling curve partitioning for load balancing [46–48]. While the concept of octree-based adaptive space partitions is well studied, developing such methods for the IMGA on large distributed systems is novel. In addition, adaptations of existing IMGA on the octree background mesh are developed to enable an efficient searching of background element for a given surface quadrature point, as well as a distribution of surface quadrature points over processors to reduce memory overhead and better parallelize the surface assembly in the IMGA. The original octree-based adaptive mesh refinement algorithm is also adjusted for the in-out test in the IMGA. Finally, the proposed octree-based IMGA framework is applied to the simulations of tracking particles in channels to demonstrate its accuracy and scalability.

The paper is organized as follows. In Section 2, we summarize the formulations of the in-

[1]This work is based on the thesis work of the authors, Xu [31] and Lofquist [32].

compressible Navier–Stokes equations and particle motion. Section 3 describes the semi-discrete formulations and time stepping. Section 4 briefs the octree-based mesh implementations and adaptations of extending IMGA on the octree-based mesh. In Section 5, we validate the framework and show scaling results using a case of sphere dropping in a quiescent fluid. We also show a few applications of this framework for tracking particles in different microchannels. Finally, we draw conclusions and motivate future research in Section 6.

2. Governing equations for particles moving in fluids

We consider the non-dimensional two-way coupled governing equations describing the interactions between the particle and the fluid in the channel.

2.1. Incompressible Navier–Stokes equations

The flow is described by the dimensionless Navier–Stokes equations posed on a fluid domain Ω_t:

$$\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\boldsymbol{\nabla}\mathbf{u}\right) - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma} = \mathbf{0}\,, \tag{1}$$

$$\boldsymbol{\nabla}\cdot\mathbf{u} = 0\,, \tag{2}$$

where t is the time, and $\mathbf{u}$ is the flow velocity. The stress and strain-rate tensors are defined respectively as

$$\boldsymbol{\sigma}(\mathbf{u}, p) = -p\,\mathbf{I} + 2\frac{1}{Re}\boldsymbol{\varepsilon}(\mathbf{u})\,, \tag{3}$$

$$\boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}\left(\boldsymbol{\nabla}\mathbf{u} + \boldsymbol{\nabla}\mathbf{u}^{\mathrm{T}}\right)\,, \tag{4}$$

where p is the pressure, $\mathbf{I}$ is an identity tensor, and Re is the channel hydraulic diameter based Reynolds number. The problem (1)–(4) is accompanied by suitable boundary conditions, defined on the boundary of the fluid domain, $\Gamma_t = \Gamma_t^{\mathrm{D}} \cup \Gamma_t^{\mathrm{N}}$:

$$\mathbf{u} = \mathbf{u}_{\mathbf{g}} \qquad \text{on } \Gamma_t^{\mathrm{D}}\,, \tag{5}$$

$$-p\,\mathbf{n} + 2\frac{1}{Re}\boldsymbol{\varepsilon}(\mathbf{u})\,\mathbf{n} = \mathbf{h} \qquad \text{on } \Gamma_t^{\mathrm{N}}\,, \tag{6}$$

where $\mathbf{u}_{\mathbf{g}}$ denotes the prescribed velocity at the Dirichlet boundary Γ_t^{D}, $\mathbf{h}$ is the traction vector at the Neumann boundary Γ_t^{N}, and $\mathbf{n}$ is the unit normal vector pointing in the wall-outward direction.

2.2. Particle motion

The particle is modeled as a rigid body. We denote the dimensionless kinematic state of the object as $\mathbf{Y}$, and the equations of motion may be written in a Lagrange frame of reference as follows [49]:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{x}^{\mathrm{c}} \\ \mathbf{R} \\ \mathbf{P} \\ \mathbf{L} \end{bmatrix}, \qquad \dot{\mathbf{Y}} = \frac{d\mathbf{Y}}{dt} = \begin{bmatrix} \mathbf{v}^{\mathrm{c}} \\ \boldsymbol{\omega}^{*}\mathbf{R} \\ \mathbf{F} \\ \mathbf{T} \end{bmatrix}, \tag{7}$$

where

$$\mathbf{v}^{\mathrm{c}} = \frac{\mathbf{P}}{M}, \qquad \boldsymbol{\omega} = \mathbf{I}^{-1}\mathbf{L}, \tag{8}$$

and

$$\mathbf{I} = \mathbf{R}\mathbf{I}_{\mathbf{ini}}\mathbf{R}^{T}, \quad \boldsymbol{\omega}^{*} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & \omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \tag{9}$$

In Equations (7)–(9), $\mathbf{x}^{\mathrm{c}}$ is the position of the center of mass of the object, $\mathbf{v}^{\mathrm{c}}$ is the velocity of the center of mass of the object, $\mathbf{R}$ is the rotation matrix mapping from initial configuration to current configuration, $\boldsymbol{\omega}$ is the object's angular velocity, $\mathbf{P}$ and $\mathbf{L}$ are linear and angular momentum, respectively. M and $\mathbf{I}$ are the mass and inertia tensor of the object, both of which are non-dimensinalized using the fluid density and characteristic length scales. and the inertia tensor $\mathbf{I}$ can be further defined using $\mathbf{I}_{\mathbf{ini}}$ in initial configuration, which is constant during the motion. In Equation (7), $\mathbf{F}$ and $\mathbf{T}$ are the integral of force and torque acting on the particle surface which are computed from the solution of the fluid field, and defined as follows:

$$\mathbf{F} = \oint_{\Gamma_t^{\mathrm{I}}} \boldsymbol{\sigma}\left(\mathbf{u}, p\right)\mathbf{n} d\Gamma_t, \qquad \mathbf{T} = \oint_{\Gamma_t^{\mathrm{I}}} \mathbf{r} \times \left(\boldsymbol{\sigma}\left(\mathbf{u}, p\right)\mathbf{n}\right) d\Gamma_t, \tag{10}$$

where $\mathbf{r}$ is the distance vector from the particle's center of mass to its surface points in the current configuration, Γ_t^{I} is the particle boundary, and the coordinates $\mathbf{x}$ and velocities $\mathbf{v}$ of its surface points are computed as

$$\mathbf{x} = \mathbf{x}^{\mathrm{c}} + \mathbf{r}, \qquad \mathbf{v} = \mathbf{v}^{\mathrm{c}} + \boldsymbol{\omega} \times \mathbf{r}. \tag{11}$$

Finally, $\mathbf{n}$ is the unit normal vector that points outward from the particle surface. Note, most microfluidic applications involve neutrally buoyant particles (i.e. the density of particle matches the density of fluid). Density matching allows us to omit the buoyancy term in Equation 10. It is trivial to include buoyancy for case of unmatched densities.

3. Semi-discrete formulation and time discretization

3.1. Variational multiscale formulation

Consider a collection of disjoint elements $\{\Omega_t^e\}$, $\cup_e \Omega_t^e \subset \mathbb{R}^d$. The fluid domain is covered by the closure of the collection: $\Omega_t \subset \cup_e \overline{\Omega_t^e}$. Note that Ω_t^e is not necessarily a subset of Ω_t with the immersed boundary method. Let $\mathcal{V}_u^h$ and $\mathcal{V}_p^h$ be the finite-dimensional spaces of discrete test functions and trial solutions for velocity and pressure, which are denoted as superscript h, and represent resolved scales (coarse scale) produced by the finite element discretization. The strong problem (1)–(6) may be recast in a weak form and posed over these discrete spaces to produce the following semi-discrete problem using the VMS modeling approach: Find $\mathbf{u}^h \in \mathcal{V}_u^h$ and $p^h \in \mathcal{V}_p^h$ such that for all $\mathbf{w}^h \in \mathcal{V}_u^h$ and $q^h \in \mathcal{V}_p^h$:

$$B^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) - F^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}\right) = 0\,. \tag{12}$$

The bilinear form B^{VMS} and the load vector F^{VMS} are given as

$$\begin{aligned}
&B^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) \\
&= \int_{\Omega_t} \mathbf{w}^h \cdot \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{u}^h\right) d\Omega_t \\
&+ \int_{\Omega_t} \boldsymbol{\nabla}\mathbf{w}^h : \boldsymbol{\sigma}\left(\mathbf{u}^h, p^h\right)\, d\Omega_t \\
&+ \int_{\Omega_t} q^h\, \boldsymbol{\nabla}\cdot\mathbf{u}^h\, d\Omega_t \\
&- \sum_e \int_{\Omega_t^e \cap \Omega_t} \left(\mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{w}^h + \boldsymbol{\nabla} q^h\right) \cdot \mathbf{u}'\, d\Omega_t \\
&- \sum_e \int_{\Omega_t^e \cap \Omega_t} p'\, \boldsymbol{\nabla}\cdot\mathbf{w}^h\, d\Omega_t \\
&+ \sum_e \int_{\Omega_t^e \cap \Omega_t} \mathbf{w}^h \cdot (\mathbf{u}' \cdot \boldsymbol{\nabla}\mathbf{u}^h)\, d\Omega_t \\
&- \sum_e \int_{\Omega_t^e \cap \Omega_t} \boldsymbol{\nabla}\mathbf{w}^h : (\mathbf{u}' \otimes \mathbf{u}')\, d\Omega_t,
\end{aligned} \tag{13}$$

and

$$F^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}\right) = \int_{\Omega_t} \mathbf{w}^h \cdot \mathbf{f}\, d\Omega_t + \int_{\Gamma_t^N} \mathbf{w}^h \cdot \mathbf{h}\, d\Gamma_t\,, \tag{14}$$

where the variables with superscript primes denote the unsolved scales (fine scale) that need to be modeled, and their effect is added onto the coarse scale. $\mathbf{u}'$ is defined as

$$\mathbf{u}' = -\tau_{\text{M}}\left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{u}^h - \mathbf{f} - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}\left(\mathbf{u}^h, p^h\right)\right)\,, \tag{15}$$

and p' is given by

$$p' = -\tau_{\text{C}}\,\boldsymbol{\nabla}\cdot\mathbf{u}^h\,. \tag{16}$$

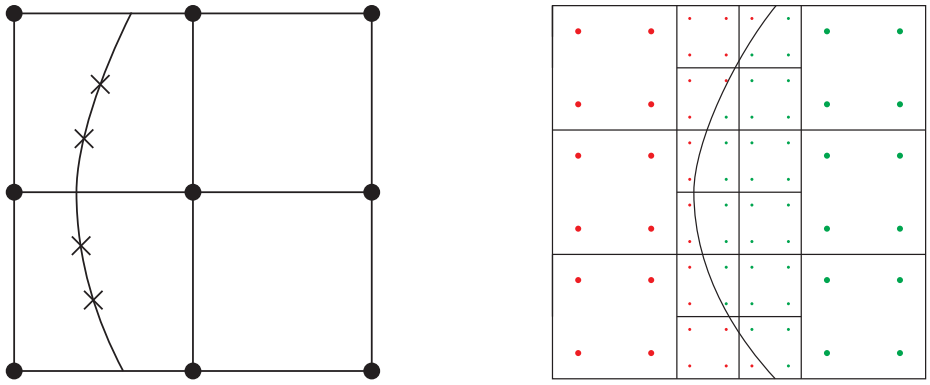
Here, $\mathbf{u}'$ and p' are approximated by the residuals of momentum equation and continuity equation, respectively, and τ_M and τ_C are corresponding coefficients with the definitions in Bazilevs et al. [21]. Equations (12)–(16) feature the VMS formulation of Navier–Stokes equations of incompressible flows. The additional terms added onto the standard weak Galerkin form can be interpreted as a combination of streamline/upwind Petrov–Galerkin (SUPG) stabilization and VMS large-eddy simulation of turbulence modeling.

3.2. Immersogeometric analysis

The no-slip boundary condition (which is a Dirichlet boundary condition) on an immersed particle surface is converted into an equivalent Neumann condition in the sense of the Nitsche's method [50]. We perform a surface integral over the immersed boundary to weakly impose the Dirichlet boundary condition [19, 51, 52]. Assuming the immersed boundary Γ_t^{I} is decomposed into N_{eb} surface elements each denoted by $(\Gamma_t^{\text{I}})_b$, the semi-discrete problem becomes

$$\begin{aligned}
&B^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) - F^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}\right) \\
&- \sum_{b=1}^{N_{eb}} \int_{(\Gamma_t^{\text{I}})_b} \mathbf{w}^h \cdot \left(-p^h\,\mathbf{n} + 2\frac{1}{Re}\,\boldsymbol{\varepsilon}(\mathbf{u}^h)\,\mathbf{n}\right)\, d\Gamma \\
&- \sum_{b=1}^{N_{eb}} \int_{(\Gamma_t^{\text{I}})_b} \left(2\frac{1}{Re}\,\boldsymbol{\varepsilon}(\mathbf{w}^h)\,\mathbf{n} + q^h\,\mathbf{n}\right) \cdot \left(\mathbf{u}^h - \mathbf{v}\right)\, d\Gamma \\
&+ \sum_{b=1}^{N_{eb}} \int_{(\Gamma_t^{\text{I}})_b} \tau^B \mathbf{w}^h \cdot \left(\mathbf{u}^h - \mathbf{v}\right)\, d\Gamma \;=\; 0\,,
\end{aligned} \tag{17}$$

where $\mathbf{n}$ is the normal vector of the immersed boundary. The boundary terms added to the governing equations are the second, third and last lines in Equation (17), and a detailed interpretation of



(a) (b)

Figure 2: Implementation of the immersogeometric method. (a) A schematic (2D example) showing how the surface assembly of IMGA is performed. The surface mesh is used to identify surface Gauss points (the 'X' locations). The immersed boundary condition terms (i.e. the last three terms in Equation (17)) are computed at these surface Gauss points, and then distributed to their background nodes. (b) A schematic (2D example) of the volume assembly in the IMGA method. An in-out test is performed to identify whether each volume Gauss point lies inside the particle (green points) or inside the fluid (red points). Only the Gauss points in the fluid domain are used to assemble the elemental matrices.

different terms can be found in Bazilevs and Hughes [19]. Only the penalty-like stabilization parameter, τ^B, is a heuristic that has to be appropriately chosen. We use the definition proposed in Wu et al. [53], which scales the stabilization parameter as $\tau^B = \max\{C^B_{\text{inv}} h/\Delta t, C^B_{\text{vis}}/(Re\, h)\}$, where the C^B's are positive dimensionless constants, h is the size of the cut element, and Δt is the time step size.

The boundary terms are imposed onto the surface Gauss points, which are then interpolated by their background Cartesian grids as shown in Figure 2a. In this way we can apply the Dirichlet boundary condition on the immersed boundary of the object. The implementation of the IMGA requires some refinement of the background mesh across the immersed surface to better capture the shape of the interface. The volume assembly of IMGA is accomplished by using selective quadrature (i.e. only using the Gauss points that lie in the fluid and not inside the immersed particle). This necessitates performing an in-out test to determine the Gauss points inside the fluid domain (red points) on which we assemble, while discarding the Gauss points inside the object (green points), as shown in Figure 2b. Note that the in-out test is also required for the adaptive refinement, and selective quadrature is performed only for the cut elements during volume assembly. The particle evolution is then computed by evaluating the force and torque exerted from the fluids on the particle. Finally, when the particle is moving, freshly-cleared nodes, i.e., some background mesh nodes that are inside the object at one time step, but are in the fluid domain at the next time step will occur. These mesh nodes have no fluid history and require interpolation from their neighbors. We refer readers to our previous work [20] for a detailed treatment.

3.3. Time stepping for fluids and particle motion

The time-dependent Navier–Stokes equations are solved using a backward Euler implicit scheme as follows

$$\frac{\partial \mathbf{u}}{\partial t} = \frac{\mathbf{u}^n - \mathbf{u}^{n-1}}{\Delta t} = \mathcal{L}(\mathbf{u}^n, p^n), \tag{18}$$

where the operator $\mathcal{L}(\mathbf{u}^n, p^n)$ represents all the other terms except the time-dependent term evaluated at the current time step in Equation (1). Δt is chosen to respect the CFL condition [2]. The (non)linear solution procedure is taken care by PETSc [54]. We utilize the SNES construct (line search quasi-Newton), which uses the KSP construct, specifically the stabilized bi-conjugate gradient (BCGS) solver. An additive Schwarz preconditioner (ASM) is used to enable parallel preconditioning and solving on decomposed sub-domains.

An explicit forward Euler time-stepper is used to update the particle location and velocity. In the discrete form we have

$$\mathbf{Y}^{n+1} = \mathbf{Y}^n + \Delta t \dot{\mathbf{Y}}^n. \tag{19}$$

$\mathbf{F}^n$ and $\mathbf{T}^n$ at each time step are discretized in space and computed with weakly imposed boundary conditions as follows

$$\mathbf{F}^n = \sum_{b=1}^{N_{eb}} \int_{(\Gamma_t^{\mathrm{I}})_b} \boldsymbol{\sigma}(\mathbf{u}^n, p^n)\mathbf{n} d\Gamma - \sum_{b=1}^{N_{eb}} \int_{(\Gamma_t^{\mathrm{I}})_b} \tau^B(\mathbf{u}^n - \mathbf{v}^n) d\Gamma, \tag{20}$$

$$\mathbf{T}^n = \sum_{b=1}^{N_{eb}} \int_{(\Gamma_t^{\mathrm{I}})_b} \mathbf{r} \times (\boldsymbol{\sigma}(\mathbf{u}^n, p^n)\mathbf{n})\, d\Gamma - \sum_{b=1}^{N_{eb}} \int_{(\Gamma_t^{\mathrm{I}})_b} \mathbf{r} \times \tau^B(\mathbf{u}^n - \mathbf{v}^n) d\Gamma \tag{21}$$

The last terms in Eqs. (20) and (21) are the penalty-like term that are added onto the surface force calculation. The total force acting on the object is the summation of the surface force and any external body forces (such as gravity and buoyancy).

4. Scalable immersogeometric analysis on octree meshes

While the IMGA is a promising approach to model particle migration in channels, it is challenging to optimize the computational efficiency on an unstructured background mesh. As a result, in this paper, we propose an octree-based IMGA that extends the IMGA on an octree-based background mesh. We employ the optimized parallel octree-based meshing library, Dendro. In this

[2] Due to the explicit time stepping used to track object motion, the Δt is usually set to a small value.

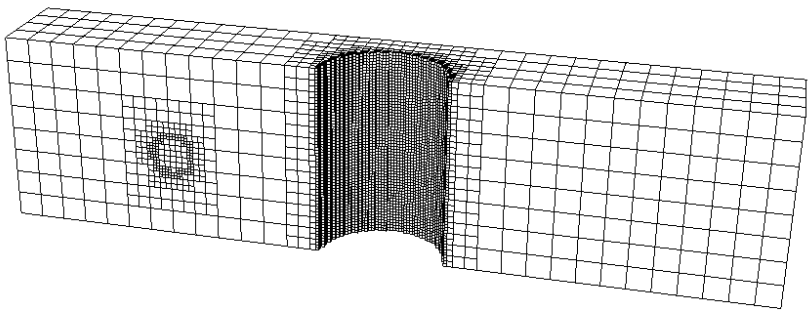


Figure 3: An illustrative example of a rigid particle traversing a microchannel decorated with obstacles. Figure shows a slice cut through the geometry.

section, we discuss adaptations of IMGA as well as some computational aspects required and developed on DENDRO that enable integration of IMGA on octree-based adaptive meshes.

4.1. Octree mesh implementation

While the elemental matrix computations are done using a separate external module (described in Section 4.2), DENDRO provides the adaptive mesh refinement and all parallel data-structures. For this work, DENDRO is extended to support meshes of (long) rectangular channels in order to account for non-cubic geometry domains. Octants outside the channels will be removed from the octree structure. Note that for channels with pillar obstacles, we can either immerse the pillars or create boundary-fitted structures since pillars will not move. The latter approach requires removal of octants inside pillars. An example of such an adaptively refined mesh (with a boundary-fitted pillar) is shown in Figure 3. The process in DENDRO used to build and maintain an adaptively refined octree mesh in parallel includes refinement, 2:1 balancing, partition, and meshing, and the algorithms of DENDRO are detailed in [55]. We also refer readers to [44, 56–59] for details on implementation of DENDRO.

4.2. Elemental computation

Node coordinates and elemental connectivity are implicit in the octree's structure, so DENDRO recalculates these values on the fly as the octree is traversed. To avoid memory overhead, we consider a single hexahedral element. As we iterate through the octree mesh for assembly we re-position the nodes in this hexahedral element to match the octree element from DENDRO. Since the octree mesh has only one possible element shape, we pre-calculate and cache the isoparametric to physical mapping at each integration point. During initialization, we create an 'index' element at each refinement level in the octree and evaluate the basis functions at the integration points. When the assembly code needs to access these values, we pull them from the corresponding refinement level in the cache instead of recalculating them at each element.

4.3. Refinement according to in-out test and subdomains

To adapt the mesh refinement to the in-out test in IMGA, a coarse mesh is first constructed based on the geometry. Proceeding in a top-down fashion, each cell in the mesh is refined if a surface (pillar/particle) passes through it, which is determined using an in-out test. If all eight corners of an octant are outside of the immersed geometry, then we retain this element, but do not refine further. If all eight points are inside the immersed geometry, then this element is performed with the same manner as outside element for a immersed strategy, while it is removed from the octree for a boundary-fitted strategy. If some of the corners of the octant are inside and others outside, then this octant is refined. This process is repeated until the desired level of refinement is achieved. Similarily, the octants outside the rectangular channels are also removed by a channel boundary in-out test (as boundary-fitted strategy) during the refinement process. Channel boundaries may also be refined using the same way for a better approximation of the channel dimensions (and boundary layers if needed). Since in our case, the pillars, particles and channels are all regular geometries (i.e., cylinder, sphere and rectangular), the in-out test can be performed analytically. However for complex geometries, a ray-tracing algorithm may be employed in the in-out test.

In addition, subdomains, which leverage the original mesh data-structure, are created to handle meshes with rectangular geometry and holes for boundary-fitted pillars as octants outside the channel and inside the boundary-fitted pillars will be removed, and also no communications are needed for them. A different scattering mapping within the subdomains for current mesh is also uniquely defined afterwards. The finite element computations will only take place in subdomains (and we can discard the main octree structure for the original domain). Therefore, subdomains have an overall (much) smaller computation domain and store (significantly) less data than the original mesh (for example, in our case of a very long channel). Re-partitioning is required as creating subdomains will result in load imbalance. For our target application, it is important to identify both the external (channel) boundary as well as the internal boundary (boundary-fitted pillar surface). The subdomain stores two bits to keep track of whether a node is non-boundary, external, or internal boundary.

4.4. Sampling the immersed boundary and adding corrections

In order to reduce memory overhead and better parallelize the surface assembly in IMGA, we distribute surface quadrature points over processors. The object boundary mesh is generated as a triangulated mesh. Surface quadrature point coordinates, along with other necessary parameters, such as the unit normal vector and boundary values of velocity at each quadrature point, are then calculated in each triangle element using standard Gaussian quadrature. The surface quadrature points are then sorted and distributed over processes. This is done by associating each surface quadrature point with an octree element (real or virtual) with the maximum refinement that contains

the quadrature point. Note that this octree element is not necessarily an existing octant in the octree mesh. This associated octree element represented by its bottom-left-back (minimum) node can be then aligned on the space-filling-curve, and the processor it belongs to can be easily found by the partitioning of the space-filling-curve. To find the actual background octree element that contains the quadrature point, we loop over all the octree elements in the process to check if the octree element is an ancestor of the associated octree element, or if they are exactly the same octree element. Since the octree elements and the surface quadrature points are both sorted based on the space-filling-curve, we can loop over them – in parallel – with an efficiency of $O(m + n)$ instead of a nested loop with an efficiency of $O(m \times n)$, (unstructured meshes usually have to perform a nested loop), where m is the local number of elements in the background mesh and n is the local number of surface quadrature points. Boundary conditions imposed on the surface quadrature points can be then evaluated and distributed to the nodes of the background octree element. The distribution of surface quadrature points over processes and finding their background elements are challenging in unstructured meshes, as the process boundaries are usually complex in most graph-based partitioning approaches. When the object is moving, this is even more cumbersome since it has to be performed at each time step.

Another computational efficiency issue caused by the IMGA is that the immersed geometry is likely localized on a small subset of processes. These processes are the only ones that perform the surface assembly for weakly imposing no-slip boundary condition on the immersed boundary. A potential solution is to perform a weighted partition – increasing the weight of the intercepted elements by the additional relative cost of surface assembly with volume assembly. This weighted partition will ensure better load balancing. We defer this development to a subsequent paper.

4.5. Adaptive remeshing and intergrid transfers

An essential requirement for computational efficiency is to adapt the spatial mesh as the particle moves across the channel. In the distributed memory setting, this also indicates a need to re-partition and re-balance the load. We adaptively remesh the domain at each time step based on the current position of the particle using results of the in-out test in IMGA followed by the subsequent 2:1 balance enforcement, partitioning and meshing process. Once the new mesh is generated, we transfer the data from the old mesh to the new mesh using interpolation. To keep things simple at this stage, we remesh each time from scratch followed by the repartition of new octree (reuse the same code of initial octree generation and partition because they are sufficiently optimized in Dendro [59]), and then interpolate the local new mesh. The nodes of the local new mesh are distributed over processes based on the old mesh partitioning similarly as described in Section 4.4 to perform interpolation. Again, the intergrid transfer is challenging in unstructured meshes as repartitioning usually offers no guarantee of good overlap between the old and new partitions

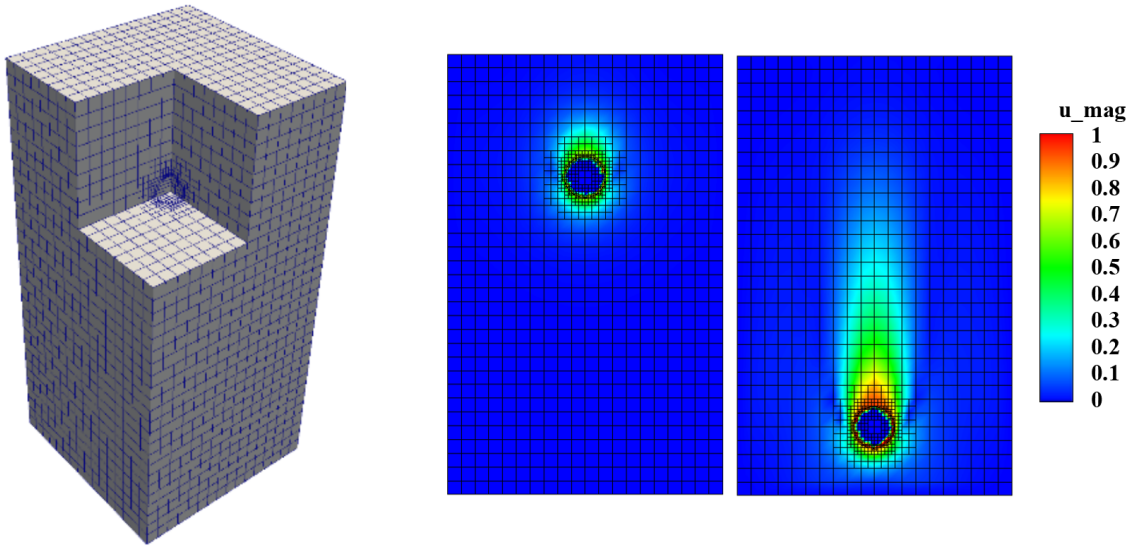


Figure 4: A representative mesh illustrating the refinement around the particle, and contours of velocity magnitude at two representative time instances.

in most graph-based approaches, and the distribution of local new mesh will be difficult across complex process boundaries. A construction of global old mesh may be required for intergrid transfer in unstructured meshes.

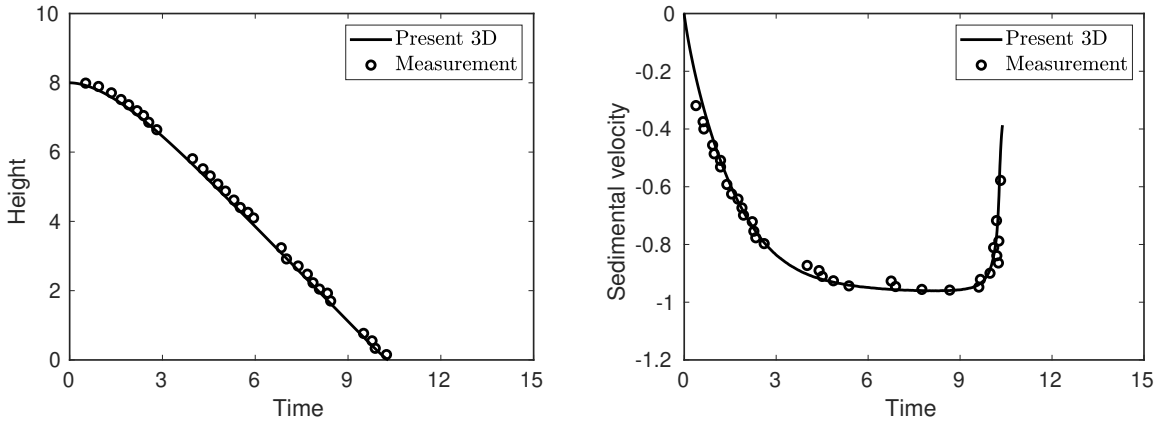
5. Experiments and results

5.1. Implementation specification

The Dendro framework is implemented in `C++` using `MPI` for distributed memory parallelism and `OpenMP` for shared memory parallelism. This is integrated with a `C++` module (Section 4.2) for evaluating basis functions and weak form of governing equations to support elemental computation. Our code is tightly integrated with `PETSc v3.7`'s distributed matrix and vector data-structures and utilizes its SNES and KSP solvers. These tests were compiled and run on Oak Ridge's Titan supercomputer (before its decommissioning in 2019). PETSc, Dendro, and the main program were compiled with the GNU 4.9.3 compiler with -O2 optimization flags. Timing information was reported using PETSc's logging framework.

5.2. Validation

We first validate the framework by comparing the particle trajectory and velocity against a benchmark experimental data of a sphere dropping in a quiescent fluid [60]. We consider a container with dimension of $0.1\,m \times 0.16\,m \times 0.1\,m$. We simulate a sphere with a diameter of $D = 0.015\,m$, released at a height of $0.12\,m$ in the middle. The fluid has a density of $\rho_f = 960\,kg/m^3$, and a dynamic viscosity of $\mu = 0.058\,kg/(m \cdot s)$. The density of the sphere



(a) Height vs. time.

(b) Sedimental velocity vs. time.

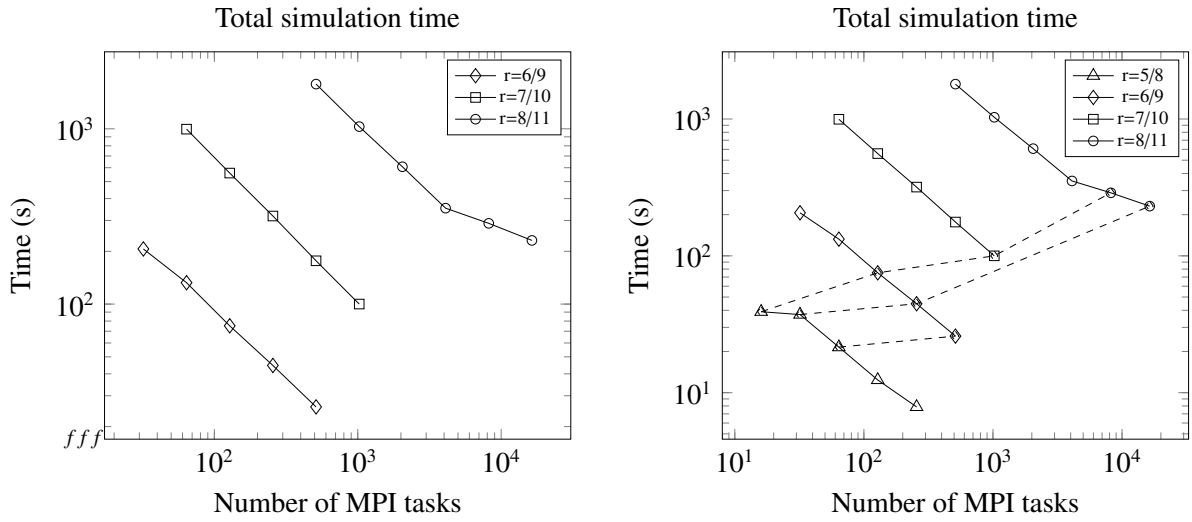
Figure 5: Comparisons of the non-dimensional height and sedimental velocity of the particle as it settles downwards with an experimental benchmark of a particle setting in a viscous fluid [60]. Note as the particle nears the bottom surface, its velocity rapidly zeros out.

is $\rho_s = 1120\,kg/m^3$. Reynolds number, defined as $\frac{\rho_f u_0 D}{\mu}$, is $Re = 31.9$ with a reference velocity $u_0 = 0.128\,m/s$. Time step size Δt is set to 1.2×10^{-3} s. Initial conditions are set as zero velocity in the whole fluid domain. No-slip boundary condition is imposed on lateral and bottom walls, and traction-free boundary condition is imposed on the top wall. We adaptively refine the mesh around the interface of the sphere and fluid. We refine three levels deeper than the rest of the background mesh. Specifically, we refine to a minimum/maximum level, $r = 5/8$ (successively bisect and divide the octree root five and eight times, respectively). We remesh after each time step as the sphere drops. Note that such frequent adaptive remeshing is one of the challenges of our target application [3]. We set the surface triangular mesh size of the sphere in sync with the background interface element size, keeping a ratio of 1:2 (surface to background) to ensure adequate surface integration. The mesh example and visualizations of velocity magnitude contour, and the validation of non-dimensional height and sedimental velocity as the sphere settles downwards are presented in Figure 4 and 5, respectively. As can be seen, both the sedimental velocity and the trajectory of the sphere match with the experiment results well.

5.3. Parallel scalability

We next show scaling performance of the framework. We collect timing for the case of a dropping sphere. We run each case for 5 time steps. The same setup and (re)meshing strategy as in last section is adopted. We run this experiment on four minimum/maximum refinement levels: $r = 5/8, 6/9, 7/10$, and $8/11$. Each refinement level has roughly seven to eight times more degrees

[3] While remeshing after every time step is not necessary, we perform this to illustrate scaling behavior of each part of the framework



(a) Strong scaling for refinement level $r = 6/9$, $7/10$, and $8/11$

(b) Weak (dashed lines) scaling approximated from strong (solid lines) scaling results

Figure 6: Strong and approximated weak scaling for a non-dimensional sphere of unit size dropping in a channel of size $8 \times 8 \times 8$ running for 5 time steps with number of processes up to 16,384 on Titan.

of freedom to solve for than the previous level, with $r = 5/8$ having 203,000 and $r = 8/11$ reaching 70.2 million degrees of freedom.

We note that given specific minimum/maximum refinement level and the same initial and boundary conditions, the overall problem size (total degrees of freedom) in spite of remeshing is independent of the number of processes being used for the simulation. To this effect, we believe presenting performance for different minimum/maximum refinement levels with different numbers of processes, in the style of a strong scaling is appropriate. Indeed, performing weak scaling for such real-world applications is more difficult than strong scaling, since it is much harder to ensure that N/p, i.e., the grain size stays relatively constant with such frequent adaptive remeshing and consequently changes in problem size, where N is the total degrees of freedom and p is the number of processes. Therefore, given the somewhat fixed increase in problem size with increasing minimum/maximum refinement level and corresponding increase of number of processes, we can combine multiple strong scaling results to derive approximate weak scaling results for the overall simulation time. The approximated weak scaling results are presented in Figure 6b. Note that minor fluctuations in the approximation of the weak scalability are expected due to the inconsistent grain size.

5.3.1. Strong scalability

For our target application, the key goal is to be able to perform the simulations quickly. Given this, and the relatively moderate size of our problems, the focus is on strong scalability. We first

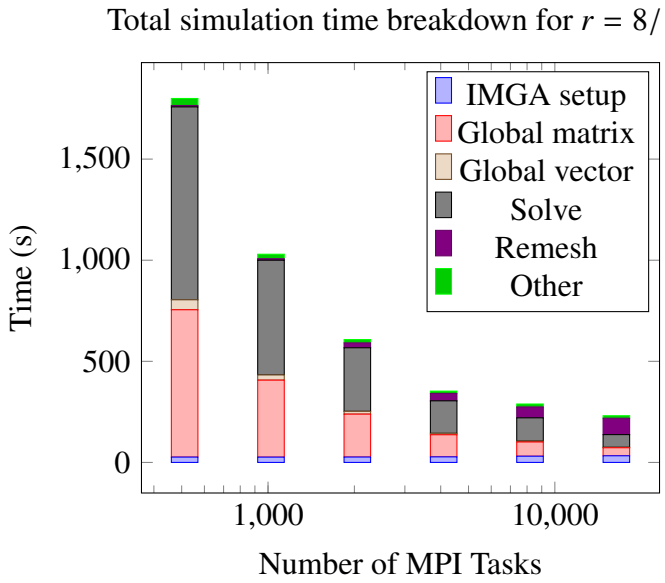


Figure 7: Total simulation time broken down by category for refinement level $r = 8/11$. IMGA setup refers to the setup required to perform immersed boundary method. Global matrix and Global vector refer to the time taken to build the global Jacobian matrix and residual vector. Solve refers to the time taken to actually solve the system (i.e. PETSc BCGS solver + ASM preconditioner). Remesh refers to the time taken to create the mesh of next time step and interpolate data onto it.

present strong scalability results for the overall simulation time including the cost of everything in Figure 6a for three problem sizes. Overall our code scales well, with continued reductions in simulation time. A breakdown of the total simulation time into various significant components for the refinement level of $r = 8/11$ is also presented in Figure 7. We can see that the amount of solve time and matrix assembly time, which are comparable, dominate the total simulation time for most of the cases. The immersed boundary method corrections (IMGA setup) involving the distribution of immersed surface points on the octree mesh also scales reasonably well.

One significant trend with increasing number of processes is "total remeshing", as shown in Figure 8 (also listed as "Remesh" in Figure 7). This refers to the overall remeshing stage combining generating a new mesh, interpolating between two meshes and reinitializing the matrix, vector and solver. Effectively, this is the overhead paid for having good adaptivity. The scaling of remeshing is poor compared with other parts of the code, but the magnitude of time it takes is much smaller than solving the Navier–Stokes equations for most of the cases except using relatively large numbers of processes (last two data points) in the refinement level of $r = 8/11$. The remeshing time is comparable with the solve time (shown in Figure 7) in these two cases when the communication becomes considerable. This is due to the interpolation between two meshes because the generation of a new adaptively refined mesh is sufficiently optimized. Note at the current stage, we perform a remeshing and repartitioning from scratch (due to the sufficiently optimized meshing code) first,

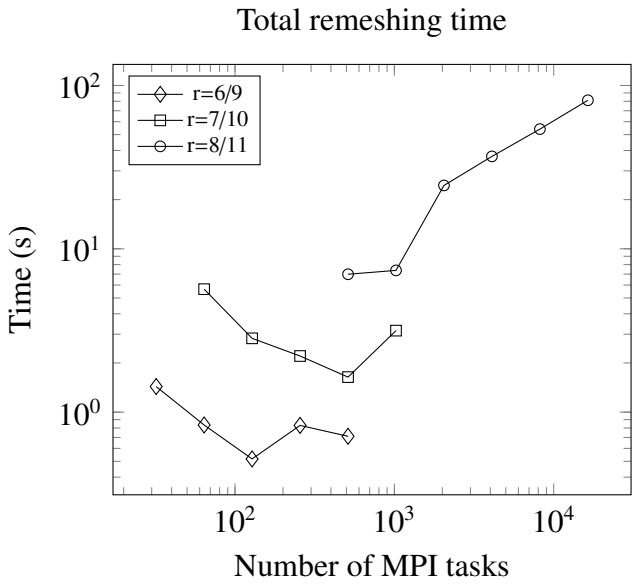


Figure 8: Total time of adaptive remeshing for refinement level $r = 6/9$, $7/10$, and $8/11$.

followed by a subsequent interpolation. However, the interpolation may not be optimal since a large amount of communication may be required by distributing new local mesh. We are exploring alternatives (to be reported in a subsequent paper). Specifically, we could remesh (refine or coarsen octants) in each process while keeping the local geometry domain unchanged in each process. This means the old and new local mesh are overlapping and consequently there is no need to distribute new local mesh over processes during interpolation. We could then perform interpolation (no communication needed) first in each process, and then repartition the new octree and corresponding newly interpolated solution vector for load balancing.

5.4. Results for particle tracking in microchannels

We finally present two results for the application of this framework. The first is particle tracking in a channel with a square cross-section, and the other is our canonical problem of particle tracking in a channel with pillar obstacles. The schematics of both cases are shown in Figure 9, and we present both cases in non-dimensional units.

5.4.1. Square channel

Case setup: We consider a long channel with dimensions $96 \times 4 \times 4$. We simulate a spherical particle released at $(10, 2, 1)$ with diameter of $D = 1$. The origin is located at the bottom-left-front corner (oriented as shown in Figure 9) of the channel. We set the particle Reynolds number, $Re_D = 5$. We assume the particle is of the same density as the fluid, so that there is no buoyancy effect. Time step size Δt is set to 0.05. The inlet has unit velocity normal to the inlet. No-slip boundary condition is imposed on surrounding walls, and zero pressure is imposed on the outlet.

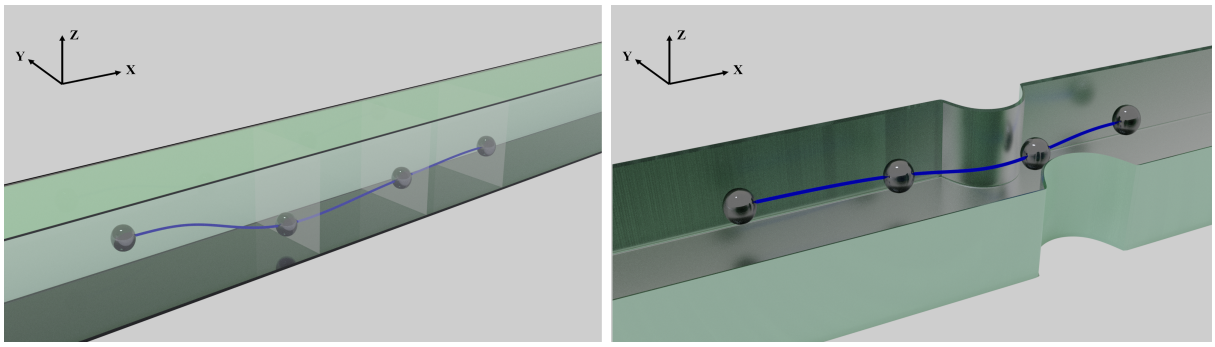


Figure 9: Schematics of particle tracking in different configurations.

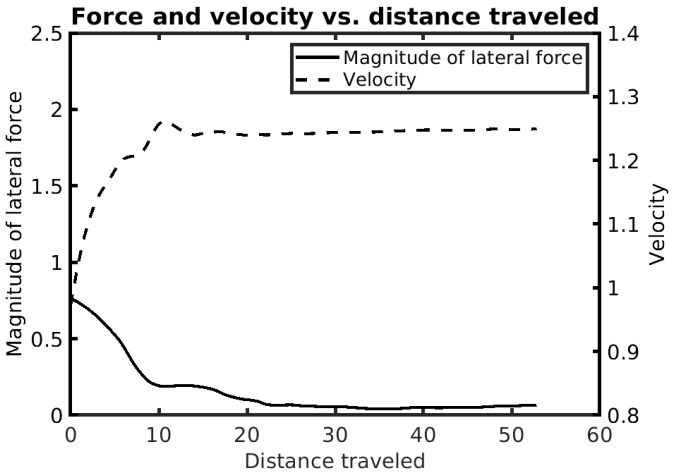


Figure 10: Lateral force and velocity magnitude of the particle vs. distance it traveled downstream.

The initial condition for the fluid velocity as well as the particle velocity are both set to be the same as the inlet velocity. We note that such long simulations – tracking the temporal evolution of the particle as it traverses nearly 50D downstream – is fairly atypical in the microfluidics community.

Results: We are interested in the magnitude of lateral force acting on the particle and the magnitude of the particle velocity as the particle reaches its equilibrium cross-sectional position as shown in Figure 10. After the particle has traveled $30D$ downstream, it is clear that the velocity have converged to a steady value (representing pure streamwise motion, and very small lateral motion), which suggests that the particle has reached its equilibrium position. Additionally, the net lateral force becomes negligibly small. The final cross sectional location in $y-z$ plane is (2.0, 1.04) which matches the experimentally determined equilibrium position [61].

5.4.2. Channel with pillars

Case setup: In our canonical problem, we consider a sphere of diameter of $D = 1$ in a channel of dimensions of 32×5×2.5. Two half pillars of radius 1.25 and height 2.5 are placed in the channel, forming a converging-diverging type of cross section. The particle is released from $(3, 1.4, 1.25)$

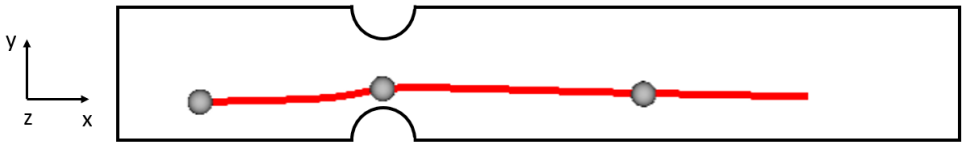
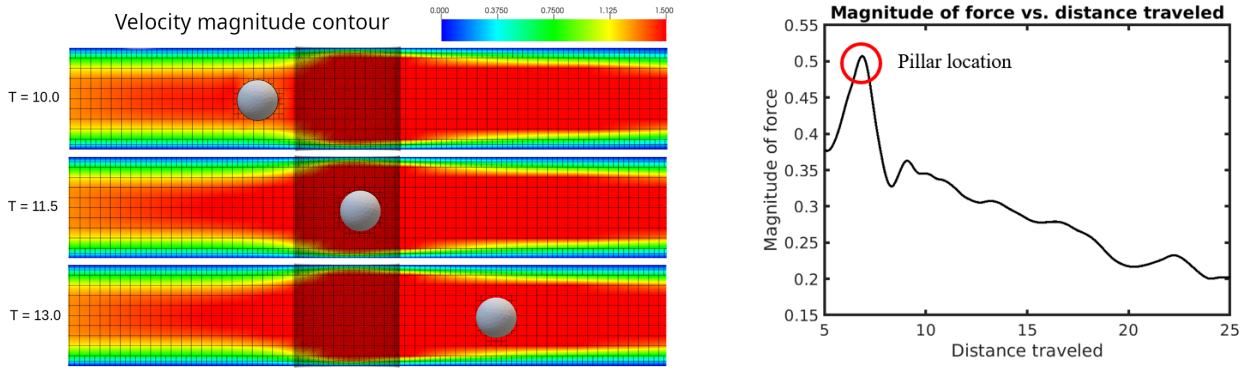


Figure 11: Top-down view ($x-y$ plane) of pathline of particle movement in a channel with pillars. Boundary-fitted pillars are plotted for visualization purpose while they are immersed in the actual simulation.



(a) Side view ($x-z$ plane) of flow velocity magnitude in channel with pillars at different time steps. The mesh shown is sub-sampled from the actual mesh for ease of visualization purpose.

(b) Magnitude of force on the particle vs. distance it traveled downstream.

Figure 12: Velocity magnitude of particle in channel with pillars at different time steps, and force magnitude on the particle during its motion.

with the same placement of the origin as the previous example. The particle Reynolds number is $Re_D = 50$ using the channel flow rate, which ensures that inertial effects are prominent [3]. Time step size Δt is set to 0.015. The boundary conditions are the same as the previous case except for the two half pillars. Note, we also immerse the two half pillars, and therefore the no-slip boundary condition on the pillars are weakly enforced. The initial condition of fluid velocity is the same as inlet velocity. The sphere is held stationary until $t = 5$ to wait for the channel flow to fully develop, so that a physically meaningful force is imposed on the released particle.

Results: Figure 11 plots the particle path as it navigates the channel with the pillar obstacles. This path curves in as the particle passes the pillars, and due to the inertial regime that the flow is in, the cross-sectional position of the particle close to outlet is offset from the initial cross-sectional location. This is in line with expected behavior from experiments in Stoecklein and Di Carlo [3], which suggest that inertial microfluidics with pillars can produce irreversible cross-sectional displacements. Figure 12(a) illustrates the 'squeezing' effect due to the presence of pillars and plots the flow velocity magnitude contour along the $x-z$ plane (i.e., side view) at 3 different time

steps (before, during and after the particle interacts with the pillars). Note that there is no direct interaction between the particle and the pillars, but instead all interactions are mediated by the fluid. Figure 12(b) quantifies this observation by plotting the force acting on the particle. Note the large jump in force as the particle traverses the channel (close to the pillar) is due to the squeezing effect. Furthermore, the simulation was performed on a mesh with around 105,000 hexahedra elements, the average number of degrees of freedom for this problem is around 350K, and the number of time steps needed to track the particle across the channel dimension is 790 steps. The total simulation time for this canonical problem is around 10 hours using 12 KNL nodes on TACC Stampede2. This is very promising as it allows us to proceed with computing cross-sectional displacement maps under different pillar configurations, which essentially translates to executing this type of simulation for a large set of different release locations across the inlet cross-section.

6. Conclusions and future directions

We developed a scalable, adaptively refined octree-based immersogeometric analysis framework, and validated this framework using a benchmark case of sphere dropping in fluids. Our framework demonstrates excellent strong (and weak) scalability for the overall simulation time, even with frequent remeshing in the benchmark case. Our framework can keep the overhead of adaptive remeshing and IMGA corrections relatively low. We anticipate additional code optimization will make the approach even more scalable. We further deployed the framework to track particle in microchannels with different (complex) geometries. This framework allows us to efficiently construct the deformation maps for particles under a broad range of experimentally accessible parameters, which will result in a passive approach for particle localization. We identify several avenues of future work. Immediate computational goals include (1) transitioning to a matrix-free solver that can significantly reduce the solve-time, while ensuring sustained adaptivity for larger processor counts, (2) designing a local refinement/coarsening algorithm and subsequent repartitioning to optimize intergrid transfer, (3) incorporating a more rigorous dynamic load balancing that accounts for the additional work involved in the surface computations, and (4) accounting for multiple moving objects. From the flow physics perspective, we plan to deploy this framework to characterize the inertial displacement maps for a range of particle sizes and pillar placements.

Acknowledgment

The authors acknowledge XSEDE grant number TG-CTS110007 for computing time on TACC Stampede2, Oak Ridge's Titan supercomputing resource and Iowa State University computing resources. Ganapathysubramanian and Gao were funded in part by NSF grant 1935255.

References

[1] D. Stoecklein and D. Di Carlo. Nonlinear microfluidics. *Anal. Chem.*, 91(1):296–314, January 2019.

[2] D. R. Gossett, W. M. Weaver, A. J. Mach, S. C. Hur, H. T. K. Tse, W. Lee, H. Amini, and D. Di Carlo. Label-free cell separation and sorting in microfluidic systems. *Anal. Bioanal. Chem.*, 397(8):3249–3267, August 2010.

[3] D. Stoecklein and D. Di Carlo. Nonlinear microfluidics. *Analytical chemistry*, 91(1):296–314, 2018.

[4] C. S. Peskin. Flow patterns around heart valves: a numerical method. *Journal of computational physics*, 10(2):252–271, 1972.

[5] C. S. Peskin. Flow patterns around heart valves: a digital computer method for solving the equations of motion. *IEEE Transactions on Biomedical Engineering*, (4):316–317, 1973.

[6] R. Glowinski, T.-W. Pan, T. I. Hesla, and D. D. Joseph. A distributed Lagrange multiplier/fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 25(5):755–794, 1999.

[7] R. Glowinski, T. W. Pan, T. I. Hesla, D. D. Joseph, and J. Périaux. A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: Application to particulate flow. *Journal of Computational Physics*, 169(2):363–426, 2001.

[8] R. Glowinski and Y. Kuznetsov. Distributed Lagrange multipliers based on fictitious domain method for second order elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 196:1498–1506, 2007.

[9] L. Zhang, A. Gerstenberger, X. Wang, and W. K. Liu. Immersed finite element method. *Computer Methods in Applied Mechanics and Engineering*, 193:2051–2067, 2004.

[10] W. K. Liu, D. W. Kim, and S. Tang. Mathematical foundations of the immersed finite element method. *Computational Mechanics*, 39(3):211–222, 2007.

[11] X. S. Wang, L. T. Zhang, and W. K. Liu. On computational issues of immersed finite element methods. *Journal of Computational Physics*, 228(7):2535–2551, 2009.

[12] X. Wang and L. T. Zhang. Modified immersed finite element method for fully-coupled fluid–structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 267:150–169, 2013.

[13] H. Casquero, C. Bona-Casas, and H. Gomez. A NURBS-based immersed methodology for fluid–structure interaction. *Computer Methods in Applied Mechanics and Engineering*, 284: 943–970, 2015.

[14] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, and T. J. R. Hughes. An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves. *Computer Methods in Applied Mechanics and Engineering*, 284:1005–1053, 2015.

[15] F. Xu, D. Schillinger, D. Kamensky, V. Varduhn, C. Wang, and M.-C. Hsu. The tetrahedral finite cell method for fluids: Immersogeometric analysis of turbulent flow around complex geometries. *Computers & Fluids*, 141:135–154, 2016.

[16] M.-C. Hsu, C. Wang, F. Xu, A. J. Herrema, and A. Krishnamurthy. Direct immersogeometric fluid flow analysis using B-rep CAD models. *Computer Aided Geometric Design*, 43:143–158, 2016.

[17] C. Wang, F. Xu, M.-C. Hsu, and A. Krishnamurthy. Rapid B-rep model preprocessing for immersogeometric analysis using analytic surfaces. *Computer Aided Geometric Design*, 52-53:190–204, 2017.

[18] F. Xu, Y. Bazilevs, and M.-C. Hsu. Immersogeometric analysis of compressible flows with application to aerodynamic simulation of rotorcraft. *Mathematical Models and Methods in Applied Sciences*, 29:905–938, 2019.

[19] Y. Bazilevs and T. J. R. Hughes. Weak imposition of Dirichlet boundary conditions in fluid mechanics. *Computers & Fluids*, 36:12–26, 2007.

[20] S. Xu, F. Xu, A. Kommajosula, M.-C. Hsu, and B. Ganapathysubramanian. Immersogeometric analysis of moving objects in incompressible flows. *Computers & Fluids*, 189:24–33, 2019.

[21] Y. Bazilevs, V. M. Calo, J. A. Cottrel, T. J. R. Hughes, A. Reali, and G. Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 197:173–201, 2007.

[22] S. Xu, B. Gao, M.-C. Hsu, and B. Ganapathysubramanian. A residual-based variational multiscale method with weak imposition of boundary conditions for buoyancy-driven flows. *Computer Methods in Applied Mechanics and Engineering*, 352:345–368, 2019.

[23] S. Xu, N. Liu, and J. Yan. Residual-based variational multi-scale modeling for particle-laden gravity currents over flat and triangular wavy terrains. *Computers & Fluids*, 188:114–124, 2019.

[24] Y. Bazilevs, K. Takizawa, T. E. Tezduyar, M.-C. Hsu, N. Kostov, and S. McIntyre. Aerodynamic and FSI analysis of wind turbines with the ALE–VMS and ST–VMS methods. *Archives of Computational Methods in Engineering*, 21:359–398, 2014.

[25] J. Yan, S. Lin, Y. Bazilevs, and G. J. Wagner. Isogeometric analysis of multi-phase flows

with surface tension and with application to dynamics of rising bubbles. *Computers & Fluids*, 2018.

[26] Q. Zhu, F. Xu, S. Xu, M.-C. Hsu, and J. Yan. An immersogeometric formulation for free-surface flows with application to marine engineering problems. *Computer Methods in Applied Mechanics and Engineering*, 361:112748, 2020.

[27] J. Yan, X. Deng, F. Xu, S. Xu, and Q. Zhu. Numerical simulations of two back-to-back horizontal axis tidal stream turbines in free-surface flows. *Journal of Applied Mechanics*, 87 (6), 2020.

[28] K. Takizawa, T. E. Tezduyar, and T. Kuraishi. Multiscale space–time methods for thermo-fluid analysis of a ground vehicle and its tires. *Mathematical Models and Methods in Applied Sciences*, 25(12):2227–2255, 2015.

[29] D. Sondak, J. N. Shadid, A. A. Oberai, R. P. Pawlowski, E. C. Cyr, and T. M. Smith. A new class of finite element variational multiscale turbulence models for incompressible magneto-hydrodynamics. *Journal of Computational Physics*, 295:596–616, 2015.

[30] J. Liu and A. A. Oberai. The residual-based variational multiscale formulation for the large eddy simulation of compressible flows. *Computer Methods in Applied Mechanics and Engineering*, 245:176–193, 2012.

[31] S. Xu. Buoyancy-driven flow and fluid-structure interaction with moving boundaries. *Graduate Theses and Dissertations*, Iowa State University, 2018. https://lib.dr.iastate.edu/etd/17364.

[32] A. Lofquist. A scalable software framework for solving PDEs on distributed octree meshes using finite element methods. *Graduate Theses and Dissertations*, Iowa State University, 2018. https://lib.dr.iastate.edu/etd/16842.

[33] J. Bielak, O. Ghattas, and E. J. Kim. Parallel octree-based finite element method for large-scale earthquake ground motion simulation. *Computer Modeling in Engineering and Sciences*, 10(2):99, 2005.

[34] R. Becker and M. Braack. Multigrid techniques for finite elements on locally refined meshes. *Numerical linear algebra with applications*, 7(6):363–379, 2000.

[35] G. Legrain, R. Allais, and P. Cartraud. On the use of the extended finite element method with quadtree/octree meshes. *International Journal for Numerical Methods in Engineering*, 86(6): 717–743, 2011.

[36] C. Thieulot, P. Fullsack, and J. Braun. Adaptive octree-based finite element analysis of two- and three-dimensional indentation problems. *Journal of Geophysical Research: Solid Earth*, 113(B12), 2008.

[37] A. K. Patra, A. Laszloffy, and J. Long. Data structures and load balancing for parallel adaptive hp finite-element methods. *Computers & Mathematics with Applications*, 46(1):105–123, 2003.

[38] J. E. Flaherty, R. M. Loy, C. Özturan, M. S. Shephard, B. K. Szymanski, J. D. Teresco, and L. H. Ziantz. Parallel structures and dynamic load balancing for adaptive finite element computation. *Applied Numerical Mathematics*, 26(1-2):241–263, 1998.

[39] M. Fernando, D. Neilsen, H. Lim, E. Hirschmann, and H. Sundar. Massively parallel simulations of binary black hole intermediate-mass-ratio inspirals. *SIAM Journal on Scientific Computing*, 41(2):C97–C138, 2019.

[40] M. Fernando, D. Neilsen, E. W. Hirschmann, and H. Sundar. A scalable framework for adaptive computational general relativity on heterogeneous clusters. In *Proceedings of the ACM International Conference on Supercomputing*, pages 1–12. ACM, 2019.

[41] M. Ishii, M. Fernando, K. Saurabh, B. Khara, B. Ganapathysubramanian, and H. Sundar. Solving pdes in space-time: 4d tree-based adaptivity, mesh-free and matrix-free approaches. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '19, pages 61:1–61:61. ACM, 2019.

[42] H. Sundar, C. Davatzikos, and G. Biros. Biomechanically-constrained 4d estimation of myocardial motion. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2009*, pages 257–265. Springer Berlin Heidelberg, 2009.

[43] H. Sundar, D. Shen, G. Biros, C. Xu, and C. Davatzikos. Robust computation of mutual information using spatially adaptive meshes. In *Proceedings of the 10th International Conference on Medical Image Computing and Computer-assisted Intervention - Volume Part I*, MICCAI '07, pages 950–958. Springer-Verlag, 2007.

[44] H. Sundar, R. S. Sampath, S. S. Adavani, C. Davatzikos, and G. Biros. Low-constant parallel algorithms for finite element simulations using linear octrees. In *SC'07: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. ACM/IEEE, 2007.

[45] H. Sundar, R. S. Sampath, and G. Biros. Bottom-up construction and 2: 1 balance refinement of linear octrees in parallel. *SIAM Journal on Scientific Computing*, 30(5):2675–2708, 2008.

[46] T. Tu, D. R. O'Hallaron, and O. Ghattas. Scalable parallel octree meshing for terascale applications. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 4. IEEE Computer Society, 2005.

[47] M. S. Warren and J. K. Salmon. A parallel hashed oct-tree N-body algorithm. *Supercomputing '93. Proceedings*, pages 12–21, 1993.

[48] L. Ying, G. Biros, D. Zorin, and H. Langston. A new parallel kernel-independent fast multipole method. In *SC'03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, pages 14–14. IEEE, 2003.

[49] D. Baraff. An introduction to physically based modeling: rigid body simulation I–unconstrained rigid body dynamics. *SIGGRAPH Course Notes*, pages D31–D68, 1997.

[50] J. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 36:9–15, 1971.

[51] Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Weak Dirichlet boundary conditions for wall-bounded turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 196:4853–4862, 2007.

[52] Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes. *Computer Methods in Applied Mechanics and Engineering*, 199:780–790, 2010.

[53] M. C. H. Wu, D. Kamensky, C. Wang, A. J. Herrema, F. Xu, M. S. Pigazzini, A. Verma, A. L. Marsden, Y. Bazilevs, and M.-C. Hsu. Optimizing fluid–structure interaction systems with immersogeometric analysis and surrogate modeling: Application to a hydraulic arresting gear. *Computer Methods in Applied Mechanics and Engineering*, 316:668–693, 2017.

[54] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.10, Argonne National Laboratory, 2018. http://www.mcs.anl.gov/petsc.

[55] H. Sundar, R. Sampath, and G. Biros. Bottom-up construction and 2:1 balance refinement of linear octrees in parallel. *SIAM Journal on Scientific Computing*, 30(5):2675–2708, 2008.

[56] M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quadtrees and quality triangulations. *International Journal of Computational Geometry & Applications*, 9(06):517–532, 1999.

[57] C. Burstedde, L. C. Wilcox, and O. Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.

[58] Hari Sundar, George Biros, Carsten Burstedde, Johann Rudi, Omar Ghattas, and Georg Stadler. Parallel geometric-algebraic multigrid on unstructured forests of octrees. In *Proceed-*

ings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, pages 43:1–43:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press. ISBN 978-1-4673-0804-5. URL http://dl.acm.org/citation.cfm?id=2388996.2389055.

[59] M. Fernando, D. Duplyakin, and H. Sundar. Machine and application aware partitioning for adaptive mesh refinement applications. In *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*, pages 231–242. ACM, 2017.

[60] A. Ten Cate, C. H. Nieuwstad, J. J. Derksen, and H. E. A Van den Akker. Particle imaging velocimetry experiments and lattice-boltzmann simulations on a single sphere settling under gravity. *Physics of Fluids*, 14(11):4012–4025, 2002.

[61] D. Di Carlo, J. F. Edd, K. J. Humphry, H. A. Stone, and M. Toner. Particle segregation and dynamics in confined flows. *Physical review letters*, 102(9):094503, 2009.