On thin gaps between rigid bodies two-way coupled to incompressible flow

Linhai Qiu*, Yue Yu*, Ronald Fedkiw*

Stanford University, 353 Serra Mall Room 207, Stanford, CA 94305

Abstract

Two-way solid fluid coupling techniques typically calculate fluid pressure forces that in turn drive the solid motion. However, when solids are in close proximity (e.g. touching or in contact), the fluid in the thin gap region between the solids is difficult to resolve with a background fluid grid. Although one might attempt to address this difficulty using an adaptive, body-fitted, or ALE fluid grid, the size of the fluid cells can shrink to zero as the bodies collide. The inability to apply pressure forces in a thin lubricated gap tends to make the solids stick together, since collision forces stop interpenetration but vanish when the solids are separating leaving the fluid pressure forces on the surface of the solid unbalanced in regards to the gap region. We address this problem by adding pressure degrees of freedom onto surfaces of rigid bodies, and subsequently using the resulting pressure forces to provide solid fluid coupling in the thin gap region. These pressure degrees of freedom readily resolve the tangential flow along the solid surface inside the gap and are two-way coupled to the pressure degrees of freedom on the grid allowing the fluid to freely flow into and out of the gap region. The two-way coupled system is formulated as a symmetric positive-definite matrix which is solved using the preconditioned conjugate gradient method. Additionally, we provide a mechanism for advecting tangential velocities on solid surfaces in the gap region by extending semi-Lagrangian advection onto a curved surface mesh where a codimension-one velocity field tangential to the surface is defined. We demonstrate the convergence of our method on a number of examples, such as underwater rigid body separation and collision in both two and three spatial dimensions where typical methods do not converge. Finally, we demonstrate that our method not only works for the aforementioned "wet" contact, but also works in conjunction with "dry" contact where there is no fluid in the gap between the solids.

1. Introduction

Solid fluid coupling is important in many areas of science and engineering such as ship and aircraft design, the study of aneurysms and heart valves, etc. Numerical methods for modeling two-way solid fluid coupling can be loosely classified into two categories: partitioned and monolithic. Partitioned methods typically separately evolve the fluid and solids using the results of one as boundary conditions for the other in an alternating one-way coupled fashion, see e.g. [30, 42, 11]. Multiple iterations are typically required for stability, although there is no guarantee that iterations converge at all often forcing one to take excessively small time steps. Partitioned approaches can also suffer from other problems such as the added-mass instability, see e.g. [4]. The main advantage of partitioned methods revolves around the ability to reuse existing code targeted to either solids-only or fluids-only problems. Monolithic methods aim to more fully two-way couple the fluid and solids alleviating a number of the aforementioned issues, see e.g. [25, 20, 2, 27, 33, 12].

Our method is based on the method in [33] which evolves the explicitly treated components of both the fluid and solids independently and solves for the fluid pressures and the solid velocities together in a single monolithic two-way coupled system during the projection step. Whenever a line segment connecting two adjacent grid cell centers intersects a solid surface, a solid fluid coupling equal velocity constraint is enforced

*{lqiu,yuey,rfedkiw}@stanford.edu, Stanford University

on the common grid face as long as at least one of the neighboring cells is a fluid cell. An interpolation operator is defined to interpolate solid velocities from solid surfaces to these constrained grid faces, and the transpose of this interpolation operator conservatively distributes fluid pressure impulses from these grid faces back to the solid surfaces. The gradient and divergence operators are discretized to be the negated transpose of each other resulting in a symmetric linear system, which can be made positive-definite in the deformable body case by factoring the damping matrix allowing for the use of fast linear solvers such as preconditioned conjugate gradients.

Although the method of [33] has been demonstrated to be a suitable approach for two-way coupling both rigid and deformable bodies (both volumetric and thin shell) with an incompressible fluid, we have observed erroneous results when solid bodies are in close proximity and multiple solid boundaries are rasterized to a single grid face. For example, when two rigid bodies are in contact, there will be grid faces in the gap between the two solids whose two adjacent grid cell centers are rasterized to different solids. The method of [33] does not assign solid fluid coupling constraints to such grid faces, since cells on the two sides of such a grid face are both inside solids. Thus, fluid pressure forces are absent inside this gap region producing nonphysical behavior. Although collision and contact stops the solids from interpenetrating, these inequality constraints vanish when the solids are separating leaving only an unbalanced fluid pressure force to spuriously push the solids back together. Moreover, in order for the solids to separate, fluid must flow into the thin gap region. Thus it is also important to model tangential flow in the gap region, incorporate the resulting velocity flux into the velocity divergence, and allow such flows to couple with the exterior flow on the grid.

It is instructive to consider how other approaches behave in the scenario where two solids come into contact with each other. One could use adaptive and/or moving body-fitted grids in order to increase the grid resolution in the gap region, for example solving with an Arbitrary Lagrangian-Eulerian (ALE) method, see e.g. [19, 10]. However, as the solids approach and eventually touch, the grid resolution required for resolving the gap region will become infinite in finite time. As an interesting alternative to body-fitted grids, methods such as those proposed in [9, 26] use a fixed-size Cartesian grid modifying the stencil near the solid boundary. In such methods, grid cell sizes will not approach zero as the solids come together because boundary conditions are enforced using ghost cells inside the solids. However, the ghost cells only work when there are real fluid cells for them to interact with, and all the real fluid cells disappear when solids come into close contact. Similarly, methods such as [3, 17, 18, 36] which rely on cut cells that are merged with full cells to avoid accuracy and time step restrictions also cannot be applied when all the full cells vanish as the solids come into close contact. Fictitious domain (see e.g. [13]) and immersed boundary methods (see e.g. [28], [29] and the references therein) intrinsically avoid the problem of vanishing or disappearing fluid grid cells by discretizing the fluid on every grid cell whether it is inside the solid or not. The fictitious domain method has an incompressibility constraint in the fluid region and a solidification constraint for the grid cells which are determined to be in the solid region. If two solids are in close proximity and both the incompressibility constraint and the solidification constraint are enforced precisely, the method lacks the degrees of freedom to represent the incompressible flow that allows fluid to flow into the gaps as the solids separate. On the other hand, relaxing the constraints even via numerical means could allow degrees of freedom between the two solids to become unsolidified and instead represent tangential fluid flow in the gap. However, this gives a much less precise description of the solid and may produce artifacts of its own. The immersed boundary method uses a smoothed out approximation of the Dirac delta function in order to define a forcing back and forth between the fluid and the solid, inherently more loosely enforcing the solidification constraint than using the fictitious domain method. Therefore, the immersed boundary method more readily allows the degrees of freedom between the two solids to model the necessary tangential flow allowing bodies to separate while immersed in the incompressible fluid. Accuracy on the other hand is another matter. This tangential flow will compete with the smeared out region where the immersed boundary method is attempting to constrain the motion of the underlying fluid degrees of freedom to follow the solid motion. This lack of accuracy can cause unwanted stress in the solid structure, and attempts to minimize this stress can stop the unwanted deformation but also stop the desired tangential flow.

In the framework of [33], we propose adding fluid pressure degrees of freedom to solid surfaces in order to provide fluid pressure forces in thin lubricated gap regions. These additional fluid pressure degrees of freedom are added to the particles of solid surface meshes which can be readily refined or coarsened according to the resolution needed for resolving the fluid flow in the gap. We only deal with volumetric rigid bodies in this paper and rasterize each rigid body as a closed hull consisting of both grid faces between the fluid and solids as well as grid faces between two different solids. We treat each grid face between two solids, where solid fluid coupling constraints are missing in [33], as two virtual faces sandwiching a virtual fluid cell in between. Additional solid fluid coupling constraints are added to these virtual faces, and the fluid pressure forces on these faces are computed using an interpolation operator that interpolates fluid pressures from solid surface pressure degree of freedom particles to virtual fluid cells. The transpose of this interpolation operator is then used to conservatively distribute the velocity flux through these virtual faces to pressure degree of freedom particles for computing the volume weighted velocity divergence. As mentioned above, the velocity divergence in the gap region should incorporate the tangential velocity fluxes. Thus, we add tangential velocity degrees of freedom to solid surface particles in order to model the fluid flow tangential to the solid surfaces. Furthermore, we couple both the fluid pressure and velocity degrees of freedom in the gap to those on the grid through another interpolation operator that interpolates pressures from fluid grid cells to solid surface particles near the boundary of the gap. Note that the transpose of this interpolation operator conservatively distributes tangential velocity fluxes from particles near the boundary of the gap to nearby grid cells. Both of the aforementioned interpolation operators are incorporated into a modified gradient operator, and their transposes are incorporated into a modified divergence operator. By folding our gap flow model into the discretized equations of [33] and maintaining the negated transpose relationship between the gradient and divergence operators, the symmetric positive-definiteness of the entire system is maintained.

2. Equations and Methods

2.1. Fluid equations

We consider the inviscid Navier-Stokes equations for incompressible flow,

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \frac{1}{\rho}\nabla p = \frac{1}{\rho}\mathbf{f} \tag{1}$$

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

in the presence of two-way coupled solid structures. Here, ρ is the constant density, $\mathbf{u}$ is the velocity, p is the pressure, and $\mathbf{f}$ is the sum of body forces such as gravity. We use a standard Marker and Cell (MAC) discretization [16] where p is defined at the cell centers while $\mathbf{u}$ and $\mathbf{f}$ are defined on the cell faces. We solve these equations using the operator splitting method [5]

$$\mathbf{u}^\star = \mathbf{u}^n - \Delta t(\mathbf{u}^n\cdot\nabla)\mathbf{u}^n + \frac{\Delta t}{\rho}\mathbf{f} \tag{3}$$

$$\nabla\cdot\left(\frac{\Delta t}{\rho}\nabla p\right) = \nabla\cdot\mathbf{u}^\star \tag{4}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^\star - \frac{\Delta t}{\rho}\nabla p. \tag{5}$$

The advection update in Equation 3 is performed using unconditionally stable semi-Lagrangian methods [6, 22]. In the presence of solids, this requires clipping [15] or ghost cells (with extrapolation into the solid) since the back-cast rays may intersect the solid.

2.2. Solid equations

A deformable body can be described by vectors of nodal positions $\mathbf{x}(t)$ and velocities $\mathbf{v}(t)$. The evolution of positions and velocities satisfies Newton's second law

$$\mathbf{x}_t = \mathbf{v} \tag{6}$$

$$\mathbf{M}\mathbf{v}_t = \mathbf{F}(\mathbf{x},\mathbf{v}), \tag{7}$$

where $\mathbf{M}$ is the solid mass matrix and $\mathbf{F}$ is the vector of all forces acting on the solid. We discretize and compute elastic and body forces explicitly, and discretize damping terms explicitly in position but implicitly in velocity. This results in

$$\begin{aligned}
\mathbf{v}^{n+1/2} &= \mathbf{v}^n + \tfrac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{F}_e(\mathbf{x}^n) + \tfrac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{D}(\mathbf{x}^n)\mathbf{v}^{n+1/2} && (8)\\
\tilde{\mathbf{x}}^{n+1} &= \mathbf{x}^n + \Delta t\mathbf{v}^{n+1/2} && (9)\\
(\mathbf{x}^{n+1}, \tilde{\mathbf{v}}^n) &= \text{CollisionAndContact}(\tilde{\mathbf{x}}^{n+1}, \mathbf{v}^n) && (10)\\
\mathbf{v}^{n+1} &= \tilde{\mathbf{v}}^n + \Delta t\mathbf{M}^{-1}\mathbf{F}_e(\mathbf{x}^{n+1}) + \Delta t\mathbf{M}^{-1}\mathbf{D}(\mathbf{x}^{n+1})\mathbf{v}^{n+1} && (11)
\end{aligned}$$

where $\mathbf{F}_e(\mathbf{x})$ is the sum of elastic and body forces, and $\mathbf{D}(\mathbf{x})$ is the damping matrix. Collision and contact are handled for deformable and rigid bodies as outlined in [37] and [14] respectively.

For a rigid body, we define the generalized position vector as $\mathbf{x} = (\mathbf{x}_{cm}^T, \boldsymbol{\theta}^T)^T$ and the generalized velocity vector as $\mathbf{v} = (\mathbf{v}_{cm}^T, \boldsymbol{\omega}^T)^T$ where $\mathbf{x}_{cm}$ and $\mathbf{v}_{cm}$ are the position and velocity of the center of mass, and $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ are the angular position and velocity. The rigid body equivalent of Equation 7 is

$$\left(\begin{bmatrix}\mathbf{M}_r & 0\\ 0 & \mathbf{I}_r\end{bmatrix}\mathbf{v}\right)_t = \begin{bmatrix}\mathbf{f}_{cm}\\ \boldsymbol{\tau}\end{bmatrix}, \qquad (12)$$

where $\mathbf{M}_r$ is a 3×3 diagonal matrix with the rigid body mass on the diagonal, and $\mathbf{f}_{cm}$ and $\boldsymbol{\tau}$ are the net force and torque. Note that the inertia tensor $\mathbf{I}_r$ is a function of $\boldsymbol{\theta}$ and thus a function of time t. The rigid body equivalents of Equations 8 − 11 are

$$\begin{aligned}
(\mathbf{M}\mathbf{v})^{n+1/2} &= (\mathbf{M}\mathbf{v})^n + \tfrac{\Delta t}{2}\mathbf{F}_e(\mathbf{x}^n) + \tfrac{\Delta t}{2}\mathbf{D}(\mathbf{x}^n)\mathbf{v}^{n+1/2} && (13)\\
\tilde{\mathbf{x}}^{n+1} &= \mathbf{x}^n + \Delta t\mathbf{v}^{n+1/2} && (14)\\
(\mathbf{x}^{n+1}, (\widetilde{\mathbf{M}\mathbf{v}})^n) &= \text{CollisionAndContact}(\tilde{\mathbf{x}}^{n+1}, (\mathbf{M}\mathbf{v})^n) && (15)\\
(\mathbf{M}\mathbf{v})^{n+1} &= (\widetilde{\mathbf{M}\mathbf{v}})^n + \Delta t\mathbf{F}_e(\mathbf{x}^{n+1}) + \Delta t\mathbf{D}(\mathbf{x}^{n+1})\mathbf{v}^{n+1} && (16)
\end{aligned}$$

Since we keep the solid positions frozen at time t^n when solving Equation 13 and frozen at time t^{n+1} when solving Equation 16, Equations 13 and 16 degenerate into Equations 8 and 11.

2.3. Solid-fluid coupled evolution

Our approach to two-way solid fluid coupling follows that of [34, 33], i.e.

$$\begin{aligned}
\hat{\mathbf{v}}^\star &= \mathbf{v}^n + \tfrac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{F}_e(\mathbf{x}^n) && (17)\\
\hat{\mathbf{u}}^\star &= \mathbf{u}^n + \tfrac{\Delta t}{2\rho}\mathbf{f} && (18)\\
(\mathbf{u}^{n+1/2}, \mathbf{v}^{n+1/2}) &= \text{SolidFluidCoupledSolve}(\mathbf{x}^n, \hat{\mathbf{u}}^\star, \hat{\mathbf{v}}^\star) && (19)\\
\tilde{\mathbf{x}}^{n+1} &= \mathbf{x}^n + \Delta t\mathbf{v}^{n+1/2} && (20)\\
(\mathbf{x}^{n+1}, (\widetilde{\mathbf{M}\mathbf{v}})^n) &= \text{CollisionAndContact}(\tilde{\mathbf{x}}^{n+1}, (\mathbf{M}\mathbf{v})^n) && (21)\\
\mathbf{v}^\star &= \tilde{\mathbf{v}}^n + \Delta t\mathbf{M}^{-1}\mathbf{F}_e(\mathbf{x}^{n+1}) && (22)\\
\mathbf{u}^\star &= \mathbf{u}^n - \Delta t(\mathbf{u}^n\cdot\nabla)\mathbf{u}^n + \tfrac{\Delta t}{\rho}\mathbf{f} && (23)\\
(\mathbf{u}^{n+1}, \mathbf{v}^{n+1}) &= \text{SolidFluidCoupledSolve}(\mathbf{x}^{n+1}, \mathbf{u}^\star, \mathbf{v}^\star) && (24)
\end{aligned}$$

The solution procedure for Equation 19 is the same as that for Equation 24, except with Δt replaced by $\frac{\Delta t}{2}$ and $\mathbf{x}^{n+1}$ replaced by $\mathbf{x}^n$. Thus, we will only focus on Equation 24 going forward.

Following the method of [33, 34], we rasterize the solid onto the MAC grid by classifying grid cell centers as either inside the fluid or inside one of the solids. The fluid pressure degrees of freedom are defined only in the cells rasterized into the fluid. The rasterization results in two types of grid cells: fluid cells and

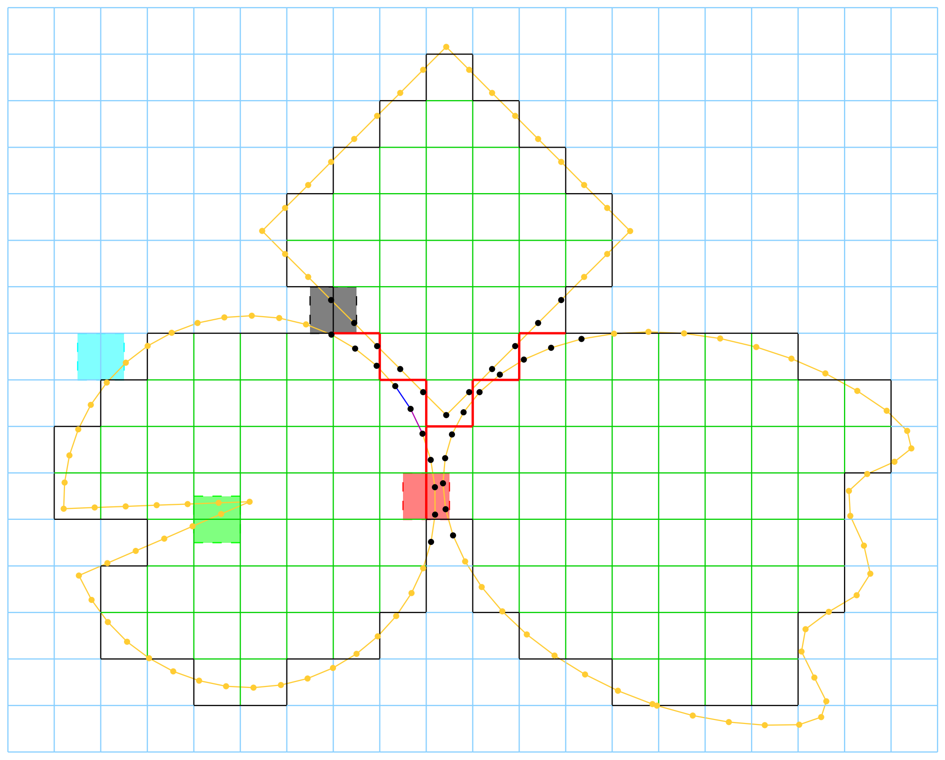


Figure 1: The blue lines indicate faces between two fluid cells. The black lines indicate faces between a solid cell and a fluid cell. The green lines indicate faces between two cells inside the same solid. The red lines indicate faces between two cells inside two different solids. The dual cell of one face of each kind is shaded as an example. The dots are the vertices of the solids. Black dots indicate additional pressure degrees of freedom in our proposed solver. Similar depiction is used throughout the paper.

solid cells, and four types of grid faces: fluid-fluid faces, solid-fluid faces, faces inside a single solid (both adjacent grid cell centers are inside the same solid), and faces between two solids (the adjacent grid cell centers are inside different solids). See Figure 1. The fluid velocity degrees of freedom are defined only on the fluid-fluid faces and the solid-fluid faces. We enforce the solid fluid coupling constraint at grid faces by enforcing equality between the fluid velocity defined on the grid face and the grid face normal component of the solid velocity interpolated to the center of that face. We denote the grid faces where this equal velocity constraint is enforced as “solid fluid coupling faces” throughout the paper. This equal velocity constraint can be described as

$$\mathbf{W}\mathbf{u} - \mathbf{J}\mathbf{v} = 0 \tag{25}$$

where $\mathbf{W}$ is a matrix of zeros and ones that picks out the solid fluid coupling faces, and $\mathbf{J}$ can be decomposed into

$$\mathbf{J} = \mathbf{W}\mathbf{N}\hat{\mathbf{J}}\mathbf{R} \tag{26}$$

where $\mathbf{R}$ samples the solids’ velocity, $\hat{\mathbf{J}}$ maps these samples to the cell faces, and $\mathbf{N}$ uses grid face normals to extract the normal component of the velocity.

For deformable bodies, $\mathbf{R} = \mathbf{I}$ since we use the surface particles of deformable bodies as velocity samples. For each grid face f, we construct a row of $\hat{\mathbf{J}}$ as follows (see [33]). First, all triangles intersecting the dual cell of f are identified and each is clipped to a subpolygon contained entirely within the dual cell. Then the weight for each subpolygon is computed as a ratio between the orthogonally projected area of this subpolygon

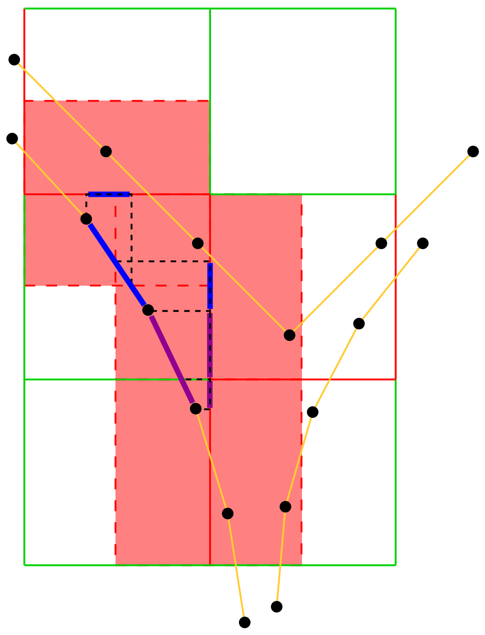


Figure 2: A zoomed-in view of six grid cells near the center of Figure 1. Three dual cell corresponding to faces between cells inside two different solids are shaded (two horizontal and one vertical). The purple segment spans two dual cells, and part of the segment is clipped and mapped to each face. The blue segment also spans two dual cells, and the parts of the blue segment within each dual cell are clipped and mapped orthogonally to the corresponding face. Note that overlapping subsets of the blue segment are mapped to the horizontal and vertical faces.

and the orthogonally projected area of all clipped subpolygons within this dual cell. Weights computed in this fashion sum to one. See Figure 2 for an example of clipping and projecting surface triangles. Whereas these weights map from triangles to grid cell faces, $\hat{\mathbf{J}}$ needs to map particles to grid cell faces. The simplest way to compute a weight for each particle is to assign one third of the weight of each subpolygon to each of the three particles that make up that subpolygon's parent triangle. Note that this does not add the correct bias if a subpolygon is closer to one particle than another. However, this is readily corrected by using the barycentric weights of the centroid of the subpolygon in order to redistribute its weight to its three parent particles. Since $\hat{\mathbf{J}}$ has rows that sum to one, it is an interpolation operator interpolating from particles of deformable bodies to grid cell faces.

For a rigid body, the velocity of a rigid body b at a sample point p is given by $\mathbf{v}_p = \mathbf{v}_b + [\mathbf{r}_{pb}]_\times^T \boldsymbol{\omega}_b$, where $\mathbf{r}_{pb}$ is the offset from the rigid body center of mass to the point p where the velocity is measured and $[\mathbf{r}_{pb}]_\times$ is the skew-symmetric matrix such that $[\mathbf{r}_{pb}]_\times \boldsymbol{\omega} = \mathbf{r}_{pb} \times \boldsymbol{\omega}$ for any $\boldsymbol{\omega}$. Thus a row of $\mathbf{R}_{pb} = \begin{pmatrix} \mathbf{I} & [\mathbf{r}_{pb}]_\times^T \end{pmatrix}$ maps from the six degree of freedom velocity of a rigid body ($\mathbf{v}_b$ and $\boldsymbol{\omega}_b$) to the sample point p at the center of a grid cell face. For each grid face f, a row of $\hat{\mathbf{J}}$ is constructed as follows. We first compute the weights that map from triangles to grid cell faces using the same method as was used for deformable bodies. However, instead of assigning weights to parent particles, weights are assigned to parent rigid bodies. If a dual cell contains only a single rigid body, then the row of $\hat{\mathbf{J}}$ corresponding to that dual cell has only a single nonzero value of 1 corresponding to the velocity sample of that rigid body at the center of the grid cell face. If multiple rigid bodies intersect a dual cell, then there will be multiple nonzero entries in that row of $\hat{\mathbf{J}}$ with each corresponding to a rigid body velocity sample at the grid cell face center. That is, $\hat{\mathbf{J}}$ interpolates a velocity from the multiple rigid body velocities sampled at the center of the grid cell face. If a dual cell contains both rigid and deformable bodies, we compute the weights from triangles to grid cell faces in the usual manner and assign weights to parent particles of deformable bodies as well as parent rigid bodies. Then $\hat{\mathbf{J}}$ is an interpolation operator that considers all relevant deformable body particles as well as all relevant rigid bodies with their velocity point-sampled to the center of the grid cell face.

The interaction between the solid and the fluid is modeled by an impulse transfer such that the equal velocity constraint (Equation 25) is satisfied at time t^{n+1} at the solid fluid coupling faces. If we denote $\boldsymbol{\lambda}$ as the impulse transfer between the solid and the fluid, then the update in Equation 24 can be expressed as

$$\boldsymbol{\beta}\mathbf{u}^{n+1} = \boldsymbol{\beta}\mathbf{u}^{\star} - \mathbf{G}\hat{\mathbf{p}} + \mathbf{W}^T\boldsymbol{\lambda} \tag{27}$$

$$\mathbf{M}\mathbf{v}^{n+1} = \mathbf{M}\mathbf{v}^{\star} + \Delta t\mathbf{D}\mathbf{v}^{n+1} - \mathbf{J}^T\boldsymbol{\lambda} \tag{28}$$

where $\boldsymbol{\beta}$ is the diagonal fluid mass matrix, $\mathbf{G}$ is the discrete volume-weighted gradient operator with solid cell columns dropped, and $\hat{\mathbf{p}}$ is the pressure scaled by Δt. The fluid mass in the dual cell of a grid face is computed by multiplying the fluid density by the fluid volume in this dual cell. The fluid volume in the dual cell of a fluid-fluid face is approximated as a full dual cell size, and the fluid volume in the dual cell of a solid-fluid face is approximated as half a dual cell size. Note that $\mathbf{J}^T$ conservatively distributes the impulse $\boldsymbol{\lambda}$ defined on the solid fluid coupling faces to the solid velocity degrees of freedom. The system of equations is solved via

$$\begin{bmatrix} \mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{G} & -\mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{W}^T & 0 \\ -\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G} & \mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{W}^T{+}\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T & \mathbf{J}\mathbf{M}^{-1}\mathbf{C}^T \\ 0 & \mathbf{C}\mathbf{M}^{-1}\mathbf{J}^T & \mathbf{I}+\mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T \end{bmatrix} \begin{bmatrix} \hat{\mathbf{p}} \\ \boldsymbol{\lambda} \\ \hat{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{G}^T\mathbf{u}^{\star} \\ \mathbf{J}\mathbf{v}^{\star} - \mathbf{W}\mathbf{u}^{\star} \\ \mathbf{C}\mathbf{v}^{\star} \end{bmatrix} \tag{29}$$

where $\hat{\mathbf{v}} = \mathbf{C}\mathbf{v}^{n+1}$, and $\mathbf{C}$ is computed by a factorization $-\Delta t\mathbf{D} = \mathbf{C}^T\mathbf{C}$ as in [33]. This is a symmetric positive-definite (SPD) system that is invertible via the preconditioned conjugate gradient method.

3. Solid Fluid Coupling Faces

[33] identifies solid fluid coupling faces by creating a line segment between the two adjacent cell centers. If the ray intersects a solid's triangle and at least one of the two cells is a fluid cell, then the common face is considered to be a solid fluid coupling face. This method correctly identifies all of the solid-fluid faces (depicted in black in Figure 1) as solid fluid coupling faces. It also identifies various fluid-fluid faces (depicted in blue in Figure 1) as solid fluid coupling faces when the line segment connecting two cell centers intersects a solid's surface triangle. However, there are other dual cells containing triangles that will be missed in this approach, such as Figure 3 (right). In addition, there are two other situations where dual cells may contain triangles but the method of [33] will not consider forces on them. Both situations pertain to solid-solid faces in Figure 1 (faces between two cells in the same solid are depicted in green and faces between two cells in different solids are depicted in red). In the case of rigid bodies, faces between two cells in the same solid may be excluded as one expects equal and opposite forces canceling in both linear and angular momentum. However, faces between two cells in different solids are quite important even in the rigid body case.

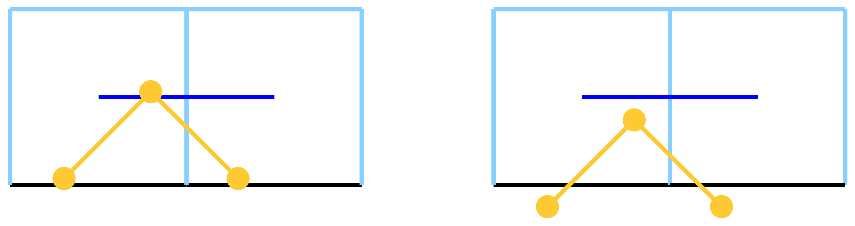


Figure 3: Two fluid-fluid face scenarios with slightly different solid positions are treated differently by [33]. (Left) Since triangles in the dual cell intersect the line segment between the cell centers, this grid cell face is regarded as a solid fluid coupling face. (Right) Since triangles in the dual cell do not intersect the line segment between the cell centers, this grid cell face is not labeled as a solid fluid coupling face.

In the case of deformable bodies, ignoring certain dual cells and the subpolygons within them is equivalent to dropping columns in $\mathbf{J}^T$ that distribute impulses from these dual cells to the velocity degrees of freedom at the parent particles of the subpolygons within these dual cells. This can lead to non-physical deformation of the deformable body. In the special case of a fluid-fluid face as in Figure 3 (right), one could treat this as a solid fluid coupling face and just assume the cast ray intersected the triangle. However, if only a small fraction of the solid is in contact with the dual cell, this could lead to an erroneously large force distributed from the fluid face to the much smaller area of the solid. A better approach would be to modify Equation 25 so that **Wu** is set equal to a fraction of **Jv** plus a fraction of a velocity determined using the gradient of the fluid pressure across the face, i.e. mixing contributions from the solid and the fluid on that face. In fact, this would be a better treatment for the case depicted in Figure 3 (left) as well, although at least this case has a solid fraction corresponding to half or more of the cell, i.e. it will not be infinitesimal as Figure 3 (right) could be.

In the rigid body case, the treatment of solid fluid coupling faces is simplified due to the fact that all the forces and torques are mapped back to the rigid body's six degrees of freedom defined at its center of mass. Thus, rasterizing a rigid body into a closed "hull" represented by grid cell faces (instead of rigid body triangles) provides for a sufficient treatment. In fact, both diagrams in Figure 3 and all other fluid-fluid faces can simply be ignored, as well as faces between cells in the same solid. Only the solid-fluid faces (depicted in black in Figure 1) and the faces between two different solids (depicted in red in Figure 1) need to be considered. Note that we leave thin shells for future work. Whereas [33] considered all the solid-fluid faces (depicted in black in Figure 1), we also address faces between two cells in different solids (depicted in red).

It is important to note that using our rasterized hull approach eliminates another difficulty with the treatment of [33]. Consider Figure 4 (left) where the fluid pressure force of the right-hand cell on the cell face (shown by the arrow) will be mapped to all triangles intersecting the dual cell. This correctly applies a force to the left on the solid body on the left, but incorrectly applies a force to the left on the solid body on the right because of the rasterized purple triangles. A similar case is shown in Figure 4 (right) where the fluid pressure force of the left-hand cell (shown by the rightward-pointing arrow) is correctly applied on the purple segments of both solid bodies, but is incorrectly applied on the blue segments. The pressure force from the right-hand cell (shown by the leftward-pointing arrow) is correctly applied on the blue segments, but is incorrectly applied on the purple segments of both solid bodies. These are quite difficult cases to resolve when the bodies are deformable; however, our rasterized hull representation adequately resolves them when the bodies are rigid. For Figure 4 (left) we only rasterize the triangles from the body on the left when computing **J** for the dual cell, and thus erroneous forces on the other body are avoided, whereas for Figure 4 (right) no triangles from either body are rasterized to the dual cell.

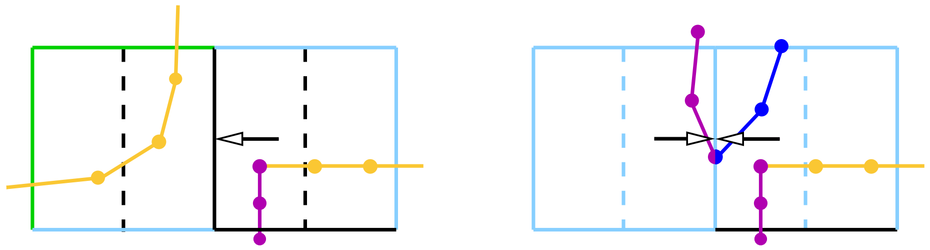


Figure 4: Two typical situations where solids receive incorrect fluid pressure forces. The arrows indicate the direction of the fluid pressure forces exerted onto the solids. The purple and blue dots and segments indicate subsets of solid surfaces that receive erroneous fluid pressure forces from the right and left side, respectively.

4. Two-Way Coupled Gap Flow

Between two different solids (red faces in Figure 1), the fluid is not resolved by the grid leading to the incorrect absence of the solid fluid coupling constraint at these grid faces. Either solid-solid collision forces or pressure forces from the fluid in these thin gaps are required to stop the solids from interpenetrating.

Whereas a contact constraint between the solids alleviates interpenetration, solid fluid coupling forces from a thin layer of fluid in the gap are necessary in order to allow the solids to separate from each other – since their separation is opposed by the exterior fluid pressure.

We treat each face between two different solids as two virtual solid-fluid faces corresponding to the two adjacent rigid bodies. These two virtual faces are translated along their face normal directions such that their face centers lie on the corresponding solid surfaces creating a virtual fluid cell in between the two solids. We place a virtual fluid pressure sample at the center of the virtual fluid cell, and then treat each virtual face as a solid fluid coupling face. See Figure 5. Although this virtual fluid cell can be used to model pressure forces orthogonal to the solid surfaces, neighboring virtual cells are unaligned in a staircased fashion making tangential flow difficult to resolve. Thus we propose a novel approach where fluid pressure degrees of freedom are added to the surface particles of the solids. Then not only can one more readily resolve the tangential flow, but also increasing the resolution is computationally exponentially cheaper since the solid surface mesh is a codimension-one surface. Although the fluid pressure values are stored on the surfaces of the two solids, we treat them as if they were inside the virtual fluid cells as shown in Figure 6. We define a matrix $\mathbf{K}$ that interpolates pressure from the two layers of surface particles to the virtual pressure samples at the centers of the virtual fluid cells. The construction of $\mathbf{K}$ is similar to that of $\hat{\mathbf{J}}$ for a deformable body with dimensions adjusted to interpolate scalars instead of vectors. After clipping surface triangles into the dual cell of a solid-solid grid face, we separately normalize the weights from each of the two relevant rigid bodies making $\frac{1}{2}\mathbf{K}$ an interpolation operator. By separately normalizing, we have assumed that the pressure degree of freedom particles on each rigid body represent half of the total fluid control volume in the gap. In practice we drop the columns of $\mathbf{K}$ that are entirely zero, since these correspond to particles that do not contribute to the gap flow and thus do not need pressure degrees of freedom.

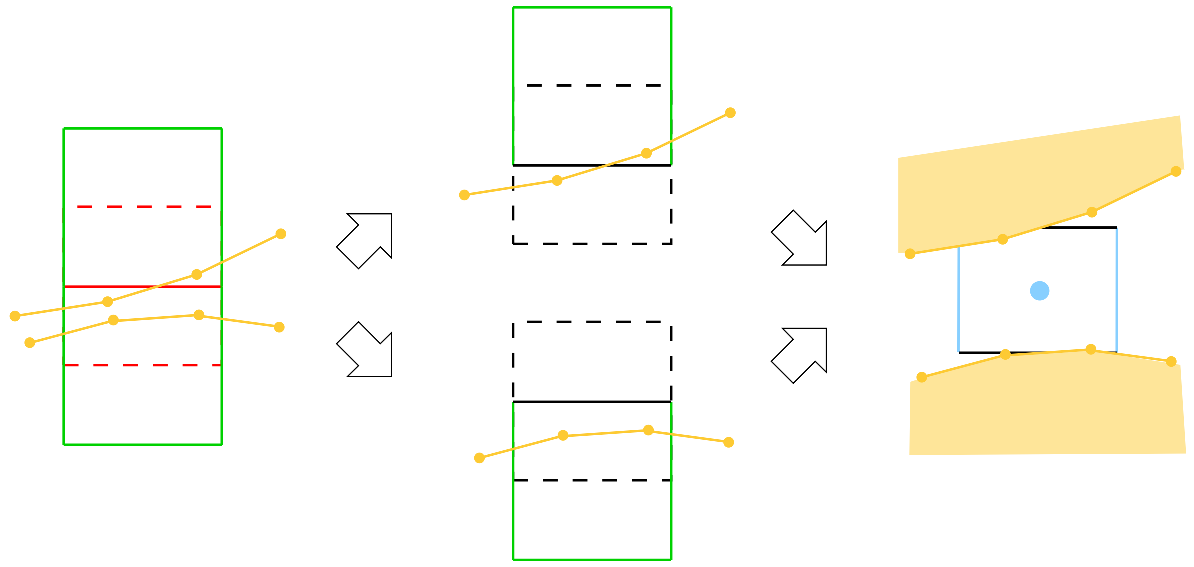


Figure 5: A grid face in between two different solids (left) is split into two virtual faces (middle), and then the virtual faces are translated to create a virtual fluid cell between the two solid surfaces (right).

Furthermore, we note that when two solids are in close proximity, some fluid grid cells between the two solids may be isolated from the exterior fluid grid cells. These may be identified using a flood fill algorithm, and treated as though they were inside the nearest solid resulting in a new set of grid faces between the two solids. Then, our method for creating fluid pressure degrees of freedom on solid surfaces and hence the construction of $\mathbf{K}$ is slightly modified as follows. When clipping surface triangles into the dual cell of a solid-solid grid face, one needs to elongate the dual cell in the direction normal to the grid face in order to include any adjacent solid cells that originated from spuriously isolated fluid grid cells.

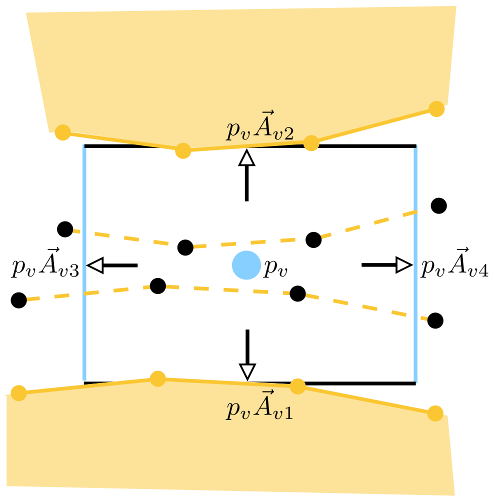


Figure 6: The fluid pressure degrees of freedom in the gap are stored on surface particles of the solids (yellow dots), but are treated as if they were in the fluid region between the two solids (black dots). Using the pressure value interpolated to the virtual pressure sample p_v (blue dot), a virtual fluid cell exerts fluid pressure forces on its faces.

When two solids are separating, incompressibility dictates that fluid flows into the gap. Thus, we define tangential velocity degrees of freedom on the surface particles of the solid as a codimension-one vector field. The tangential velocity at each surface particle lies within the plane perpendicular to the vertex surface normal, defined as the normalized area weighted average of triangle normals in the one-ring. Tangential velocity degrees of freedom are defined not only on the pressure degree of freedom particles, but also on any surface particle which is a one-ring neighbor of a pressure degree of freedom particle.

In summary, the fluid velocity and pressure degrees of freedom are augmented to become

$$\mathbf{u} = \begin{bmatrix}\mathbf{u}_n\\ \mathbf{u}_s\\ \mathbf{u}_f\end{bmatrix},\ \mathbf{p} = \begin{bmatrix}\mathbf{p}_s\\ \mathbf{p}_c\end{bmatrix} \tag{30}$$

where $\mathbf{u}_n$, $\mathbf{u}_s$, and $\mathbf{u}_f$ are the velocity degrees of freedom defined on virtual solid-fluid faces, solid surface particles, and MAC grid faces, respectively, while $\mathbf{p}_s$ and $\mathbf{p}_c$ are the pressure degrees of freedom defined on solid surface particles and MAC grid cells. The additional solid fluid coupling faces are reflected in $\mathbf{W}$, $\mathbf{J}$ and $\boldsymbol{\lambda}$ as

$$\mathbf{W} = \begin{bmatrix}\mathbf{W}_n & 0 & 0\\ 0 & 0 & \mathbf{W}_f\end{bmatrix},\ \mathbf{J} = \begin{bmatrix}\mathbf{J}_n\\ \mathbf{J}_f\end{bmatrix},\ \boldsymbol{\lambda} = \begin{bmatrix}\boldsymbol{\lambda}_n\\ \boldsymbol{\lambda}_f\end{bmatrix} \tag{31}$$

where $\mathbf{W}_n$ is an identity matrix that selects all virtual solid-fluid faces (two for each red face in Figure 1), and $\mathbf{W}_f$ selects the standard solid-fluid faces (black faces in Figure 1). The zero block in the middle column of $\mathbf{W}$ is due to the absence of coupling constraints on $\mathbf{u}_s$. Since each solid-solid face between different solids is decomposed into two virtual faces, both $\mathbf{u}_n$ and $\boldsymbol{\lambda}_n$ have length equal to twice the number of solid-solid faces. Similarly, each solid-solid face creates two rows in $\mathbf{J}_n$, which are each constructed in the usual fashion except only considering one solid or the other. These modifications to $\mathbf{W}$ and $\mathbf{J}$ are sufficient for the solid fluid coupling terms in Equations 25, 27 and 28.

A virtual fluid cell is treated similar to a regular MAC grid fluid cell. Although the tangential components of the flow are modeled on the solid surface meshes, the normal components of the flow variables including fluid masses, pressure forces, and velocity fluxes are defined on the virtual solid-fluid faces. In particular, the volume of a virtual dual cell is equal to the area of the MAC grid face times one half the distance between the two solids, measured at the center of the cell and orthogonal to the face. We define an operator $\hat{\mathbf{G}}_n$ that operates on virtual pressure samples $\mathbf{p}_v$ and returns volume weighted pressure gradients on virtual solid-fluid faces. Similar to $\mathbf{G}\hat{\mathbf{p}}$ in Equation 27 which for a regular solid-fluid face is the product of the face area and the pressure, $\hat{\mathbf{G}}_n\mathbf{p}_v$ is the product of the virtual solid-fluid face area and the pressure value $\mathbf{p}_v$. Since $\mathbf{p}_v = \frac{1}{2}\mathbf{K}\mathbf{p}_s$, the volume weighted pressure gradient on virtual solid-fluid faces is $\hat{\mathbf{G}}_n(\frac{1}{2}\mathbf{K})\mathbf{p}_s$, and the new gradient operator is

$$\mathbf{G}_n = \hat{\mathbf{G}}_n(\frac{1}{2}\mathbf{K}). \tag{32}$$

The negated transpose of $\mathbf{G}_n$ is

$$-\mathbf{G}_n^T = (\frac{1}{2}\mathbf{K}^T)(-\hat{\mathbf{G}}_n^T), \tag{33}$$

where $-\hat{\mathbf{G}}_n^T$ computes the part of the volume weighted divergence accounting for fluxes through the two virtual solid-fluid faces, and $\frac{1}{2}\mathbf{K}^T$ conservatively distribute this divergence back to the pressure degree of freedom particles on the solids. Note that exactly half of this volume weighted divergence is distributed to the relevant pressure degree of freedom particles on each side of the gap, because we separately normalized the weights from each rigid body.

4.1. Tangential component discretization

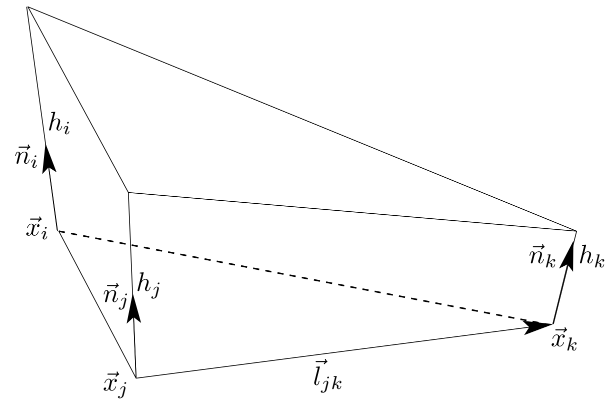


Figure 7: The wedge above a solid surface triangle. Note that $\vec{n}_i$, $\vec{n}_j$ and $\vec{n}_k$ are typically not parallel. We denote $\vec{l}_{jk} = \vec{x}_k - \vec{x}_j$ as the edge vector counterclockwise relative to $\vec{x}_i$ and $\vec{n}_i$.

The gap thickness at each velocity degree of freedom particle i is defined as the length of the ray shot from $\vec{x}_i$ along its vertex normal $\vec{n}_i$ ending when it hits the other rigid body. In the case of no intersections which can happen at a gap boundary, we extrapolate the gap thickness from nearby velocity degree of freedom particles where the gap thickness is well-defined. We denote h_i as half of the gap thickness at particle i, noting that the other half of the gap will be accounted for by the surface particles of the other solid. The wedge above a solid surface triangle $(\vec{x}_i, \vec{x}_j, \vec{x}_k)$ is defined as the volume enclosed by the bottom triangle, the top triangle $(\vec{x}_i + h_i\vec{n}_i, \vec{x}_j + h_j\vec{n}_j, \vec{x}_k + h_k\vec{n}_k)$, and the three typically non-planar (skew) quadrilateral side walls. See Figure 7. Although one could compute the minimum bounding surface of a skew quadrilateral (see for example [35, 39]) and then strive to define a meaningful velocity normal to this surface, we instead use a simplified approximation since modeling the exact wedge geometry for a thin wedge in a thin gap is less important than simply providing a mechanism that allows the fluid to flow tangentially along the solid surfaces due to incompressibility. Therefore, we approximate the solid surfaces as locally flat and compute the volume of the wedge and the areas of the quadrilaterals as if $\vec{n}_i$, $\vec{n}_j$, and $\vec{n}_k$ were parallel to each other and orthogonal to the triangle base. One third of the volume of the wedge is assigned to each of its three particles.

Viewed from particle i, the velocity outflux on the quadrilateral above edge $\vec{l}_{jk}$ is computed as $(\vec{u}_s)_j \cdot \vec{A}_{jk,j} + (\vec{u}_s)_k \cdot \vec{A}_{jk,k}$, where $(\vec{u}_s)_j$ and $(\vec{u}_s)_k$ are the tangential velocities at particles j and k, and $\vec{A}_{jk,j}$ and $\vec{A}_{jk,k}$ are the area weighted normals for particle j and k respectively on the quadrilateral above $\vec{l}_{jk}$. The area of this quadrilateral is approximated by $A_{jk} = \frac{1}{2}(h_j + h_k)||\vec{l}_{jk}||$ and distributed to particles j and k weighted by their thicknesses, i.e. weights $\frac{h_j}{h_j+h_k}$ and $\frac{h_k}{h_j+h_k}$, so that particle j gets area $\frac{1}{2}h_j||\vec{l}_{jk}||$ and particle k gets $\frac{1}{2}h_k||\vec{l}_{jk}||$. The tangential velocity at paritcle j is by definition orthogonal to $\vec{n}_j$, and its component orthogonal to the quadrilateral face should be orthogonal to $\vec{l}_{jk}$ at least near $\vec{x}_j$. Thus we approximate the surface normal of the quadrilateral at least for the purpose of $\vec{x}_j$ as $\frac{\vec{l}_{jk}\times\vec{n}_j}{||\vec{l}_{jk}\times\vec{n}_j||}$ which results in $\vec{A}_{jk,j} = \frac{1}{2}h_j||\vec{l}_{jk}||\frac{\vec{l}_{jk}\times\vec{n}_j}{||\vec{l}_{jk}\times\vec{n}_j||}$. Similarly, $\vec{A}_{jk,k} = \frac{1}{2}h_k||\vec{l}_{jk}||\frac{\vec{l}_{jk}\times\vec{n}_k}{||\vec{l}_{jk}\times\vec{n}_k||}$. Extending the approach in [43] from triangles to triangle wedges, we compute the tangential component of the volume weighted velocity divergence at a pressure degree of freedom particle i as one third the sum of the velocity outfluxes through the quadrilateral faces above the boundary edges of the particle's one-ring incident triangles. Defining $pdof(i)$ as the pressure degree of freedom index of particle i, the entries of this divergence operator $\mathbf{D}_s$ are defined via

$$(\mathbf{D}_s\mathbf{u}_s)_{pdof(i)} = \frac{1}{3}\sum_{e\in BE_i}\sum_{j\in P_e}(\vec{u}_s)_j \cdot \frac{h_j||\vec{l}_e||}{2}\frac{\vec{l}_e\times\vec{n}_j}{||\vec{l}_e\times\vec{n}_j||}, \tag{34}$$

where BE_i contains the boundary edges of particle i's one-ring incident triangles, P_e contains the two particles on edge e, and $\vec{l}_e$ is the counterclockwise edge vector relative to particle i.

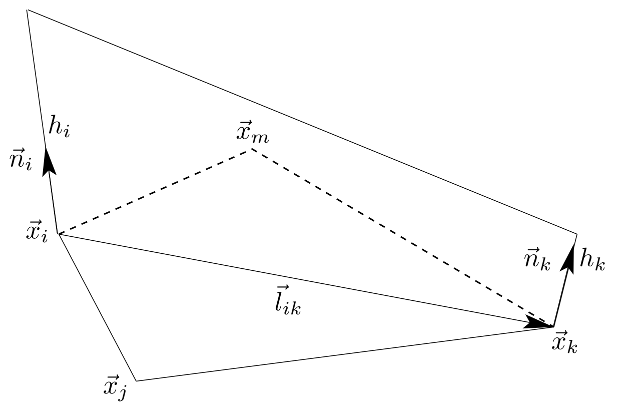


Figure 8: A quadrilateral face used for computing the pressure gradient force on particle i at $\vec{x}_i$.

We compute the tangential component of the volume weighted pressure gradient at a velocity degree of freedom particle i as the sum of the pressure gradient forces on the control areas of particle i associated with each of its incident edges (again extending [43]). For example considering Figure 8, the pressure gradient force felt by particle i from the quadrilateral above $\vec{l}_{ik}$ is $\frac{1}{3}((p_s)_j - (p_s)_m)\vec{A}_{ik,i}$, where $(p_s)_j$ and $(p_s)_m$ are the pressure values at particles j and m. Defining $vdof(i)$ as the velocity degree of freedom index of particle i, the entries of this gradient matrix $\mathbf{G}_s$ can be defined via

$$(\mathbf{G}_s\mathbf{p}_s)_{vdof(i)} = \sum_{e\in IE_i}\frac{(p_s)_{e^+} - (p_s)_{e^-}}{3}\frac{h_i||\vec{l}_e||}{2}\frac{\vec{l}_e\times\vec{n}_i}{||\vec{l}_e\times\vec{n}_i||}, \tag{35}$$

where IE_i is the set of edges incident to particle i, e^+ and e^- are the two particles which are element-wise opposite edge e, and $\vec{l}_e$ is the edge vector pointing away from particle i. Note that $\mathbf{G}_s$ defined in this fashion is the negated transpose of $\mathbf{D}_s$.

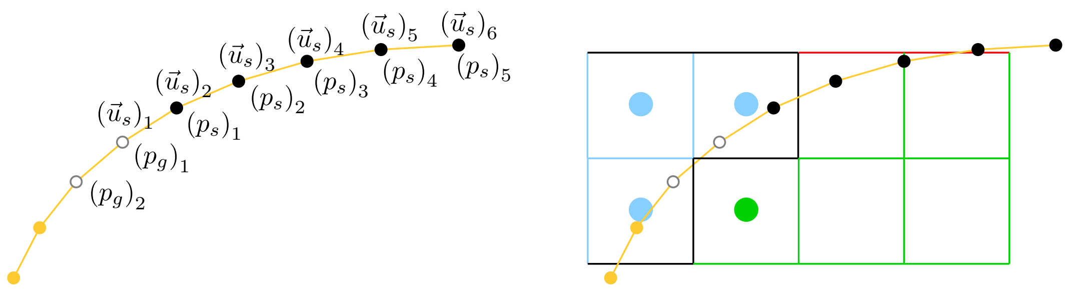


Figure 9: (Left) The black dots denote the pressure degree of freedom particles, and the hollow dots denote ghost pressure particles. In the gap we have pressures $(p_s)_1$, $(p_s)_2$, $(p_s)_3$, $(p_s)_4$, and $(p_s)_5$. We add an additional one-ring of velocity degrees of freedom outside the pressure degrees of freedom, so $(\vec{u}_s)_1$ starts one particle to the left of $(p_s)_1$. We need the pressure value at $(p_g)_1$ in order to obtain a pressure gradient for $(\vec{u}_s)_2$, and the pressure value at $(p_g)_2$ in order to obtain a pressure gradient for $(\vec{u}_s)_1$. (Right) We interpolate the pressure values on ghost pressure particles (hollow dots) from the pressure degrees of freedom on the grid (blue dots) discarding the values from solid cells (green dots) in the interpolation stencils and renormalizing the weights.

If a velocity degree of freedom particle i lies on or outside the outermost ring of pressure degree of freedom particles, some of its incident particles are not pressure degree of freedom particles and we denote such incident particles as ghost pressure particles. The coefficients from Equation 35 associated with these ghost pressure particles are moved to a separate matrix $\mathbf{G}_g$. We interpolate pressure values at ghost pressure particles from the three-dimensional MAC grid pressure degrees of freedom using multilinear interpolation, discarding solid grid cells from the interpolation stencils and renormalizing the weights. This two-way couples pressure degrees of freedom in the gap with those from the three-dimensional MAC grid near the boundary of the gap. See Figure 9. If we denote this interpolation matrix as $\mathbf{J}_{sc}$, then the full volume weighted gradient operator $\mathbf{G}$ can be expressed as

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_n & 0 \\ \mathbf{G}_s & \mathbf{G}_g \mathbf{J}_{sc} \\ 0 & \mathbf{G}_f \end{bmatrix}. \tag{36}$$

where $\mathbf{G}_f$ is the typical volume weighted gradient operator on fluid-fluid faces and solid-fluid faces in Equation 27.

The volume weighted divergence operator is obtained by taking the negated transpose of $\mathbf{G}$. Applying $-\mathbf{G}^T$ to the vector of velocity degrees of freedom in Equation 30, we obtain the volume weighted velocity divergence at pressure degree of freedom particles as $-\mathbf{G}_n^T\mathbf{u}_n - \mathbf{G}_s^T\mathbf{u}_s$, where $-\mathbf{G}_n^T\mathbf{u}_n$ accounts for the normal velocity fluxes through virtual solid-fluid faces and $-\mathbf{G}_s^T\mathbf{u}_s$ accounts for the tangential velocity fluxes through surface particles. We also obtain the volume weighted velocity divergence for MAC grid cells as $-\mathbf{J}_{sc}^T\mathbf{G}_g^T\mathbf{u}_s - \mathbf{G}_f^T\mathbf{u}_f$, where $-\mathbf{J}_{sc}^T\mathbf{G}_g^T\mathbf{u}_s$ accounts for the velocity fluxes from velocity degree of freedom particles on solid surfaces. Note that $\mathbf{J}_{sc}^T$ conservatively distributes the velocity fluxes from the particles near the gap boundary to the surrounding MAC grid cells. This formulation fully enforces the incompressibility of the fluid on the grid and the fluid in the gap in a two-way coupled manner.

Note that there can be some ghost pressure particles whose pressure values cannot be interpolated from the MAC grid, because all of the surrounding grid cells in their interpolation stencils are solid cells. See for example Figure 10. To resolve this issue, we use a flood fill algorithm. In each iteration, we check the interpolation stencils of all ghost pressure particles, set those without MAC grid fluid cells in their interpolation stencils to be pressure degree of freedom particles, and then reset the two-ring ghost pressure particles based on the new set of pressure degree of freedom particles. This terminates when all ghost pressure particles have fluid cells in their interpolation stencils. We extend $\mathbf{p}_s$, $\mathbf{u}_s$ and $\mathbf{G}_s$ accordingly. These particles allow for tangential flow that connects different regions of the solid, but do not participate in the pressure forces normal to the solid surfaces, leaving $\mathbf{G}_n\mathbf{p}_s$ unchanged.

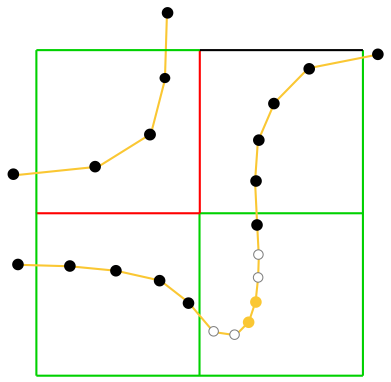


Figure 10: Ghost pressure particles on the interior of the gap (hollow dots) do not have surrounding MAC grid fluid pressures that can be used for interpolation. Therefore we designate these as new pressure degree of freedom particles. This is done iteratively until no ghost pressure particles remain or those that do remain can have their values interpolated from the surrounding fluid grid. In this particular example, both the hollow and yellow dots will become pressure degree of freedom particles.

5. Advection

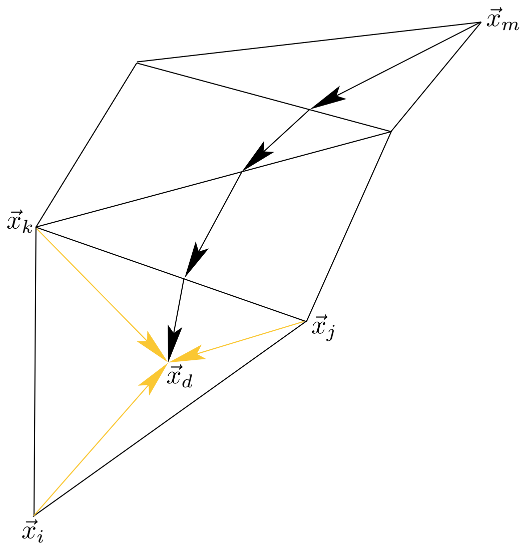


Figure 11: The semi-Lagrangian ray originating from particle m is folded onto the solid surface mesh forming a segmented curve (the black arrows) of total length $||\vec{v}_m||\Delta t$ ending at point $\vec{x}_d$. The tangential velocity at $\vec{x}_d$ is interpolated from the values at $\vec{x}_i$, $\vec{x}_j$, and $\vec{x}_k$.

As the rigid body spins and moves, we assume that the gap fluid is not dragged in the tangential direction by any frictional forces (slip boundary condition). Thus, we trace semi-Lagrangian rays in the direction $-\vec{v} = -(\vec{u}_s - \vec{v}_t)$, where $\vec{v}_t$ is the tangential velocity of the rigid body at the surface particle computed by projecting the pointwise velocity of the rigid body at the particle location into the tangent plane dictated by the particle's normal. The semi-Lagrangian update of the tangential velocity at particle m is illustrated in Figure 11. The first step is to project all the exterior boundary edges of particle m's one-ring triangles into the tangent plane of particle m, and to intersect the semi-Lagrangian ray against these projected one-ring edges. Although the intersection point is calculated on the projected geometry, we use barycentric coordinates to find it on the unprojected solid surface. If the distance from $\vec{x}_m$ to the intersection point exceeds $||\vec{v}||\Delta t$, then the semi-Lagrangian ray ends in this first triangle. Otherwise, we compute the normal at the new intersection point by interpolating normals from the surrounding vertices. Then we rotate the semi-Lagrangian ray into the local tangent plane. This is accomplished by computing consistent local coordinate systems for both the current intersection point and $\vec{x}_m$, and subsequently ensuring that the x, y, and z coordinates of the semi-Lagrangian ray are identical in these two coordinate systems. The local unit normal is chosen as the z-axis, the projection into the local tangent plane of the closest segment of the semi-Lagrangian ray is chosen as the x-axis, and finally the cross product is used to define the y-axis. The two edges of the triangle on the other side of the intersected edge are then projected into the new tangent plane and tested for intersection. For robustness if the projection of one of these other two edges is nearly parallel to the semi-Lagrangian ray, we replace the nearly parallel edge with the two edges of the triangle on the other side of that parallel edge. In addition, if an intersection point is close to a particle, we project the entire one-ring exterior boundary edges of this particle and check these for intersections. Once an intersection is found and mapped back to the surface of the rigid body, the sum of the lengths of all the unprojected line segments is compared to $||\vec{v}||\Delta t$ with the last segment shortened once $||\vec{v}||\Delta t$ is reached.

When interpolating the tangential velocity from the three particles of a triangle, we use the MAC grid to define the velocities at particles that may not be velocity degree of freedom particles. When doing this we discard solid-solid grid faces from the interpolation stencil and renormalize the weights similar to interpolating pressures from MAC grid cells in Section 4.1. The surface normal of the particle is used to project out the normal component of the interpolated velocity in order to obtain the tangential velocity. The tangent plane at $\vec{x}_d$ is defined by interpolating surface normals from $\vec{x}_i$, $\vec{x}_j$, and $\vec{x}_k$. Before interpolating the tangential velocity at $\vec{x}_d$, the velocities at $\vec{x}_i$, $\vec{x}_j$, and $\vec{x}_k$ are rotated into the tangent plane at $\vec{x}_d$ similarly to the way the semi-Lagrangian ray was rotated into the tangent plane of the intersection point in the discussion above. The local surface normals are still used as the z-axes, but the line segment connecting $\vec{x}_i$, $\vec{x}_j$, or $\vec{x}_k$ with $\vec{x}_d$ is used in order to define the x-axes.

The interpolated velocity that lives in the tangent plane at $\vec{x}_d$ at time t^n needs to be rotated into the tangent plane at $\vec{x}_m$ at time t^{n+1}. We first rotate the interpolated velocity into the tangent plane at $\vec{x}_m$ at time t^n. Again the local surface normals are used as the z-axes, but the first and last segments of the semi-Lagrangian ray are used define the x-axes at $\vec{x}_m$ and $\vec{x}_d$, respectively. We then further rotate the updated tangential velocity at $\vec{x}_m$ from the time t^n tangent plane into the time t^{n+1} tangent plane by rotating the velocity about the cross product of the old and the new surface normals by the angle between these normals. This aligns the tangent planes using the rotation that minimizes the amount of rotation.

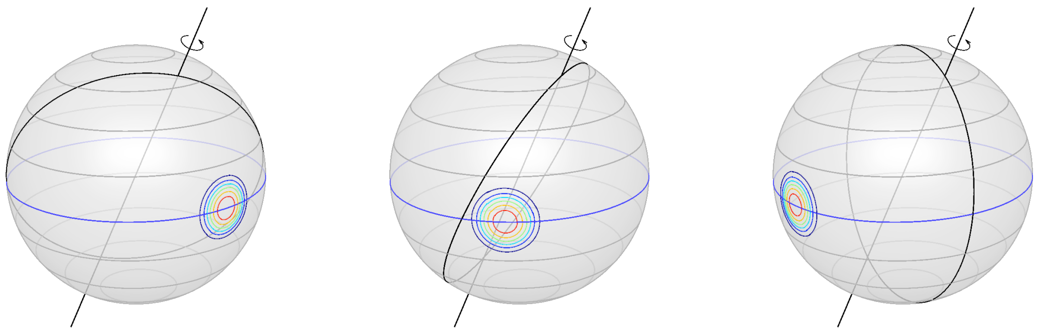
To illustrate the efficacy of our approach, we consider advection on a unit sphere. To tessellate the sphere, we start from a regular octahedron, refine the mesh iteratively using $\sqrt{3}$-subdivision [21], and finally project the particles onto the surface of the sphere. In the convergence tests, we refine the mesh from one resolution to the next by performing two more iterations of $\sqrt{3}$-subdivision resulting in edges which are on average one-third the length of those on the prior mesh. The time step size is chosen as $\Delta t = \frac{\alpha \Delta x}{\max ||\vec{v}||}$, where $\alpha = 3$ is the CFL number and Δx is the average edge length of the surface mesh of the sphere.

In the first test, we initialize a scalar field on the sphere defined by a three-dimensional bump function

$$\phi(\vec{x}) = \begin{cases} e^{\left(2 - \frac{2}{1-(\|\vec{x}-\vec{x}_c\|/r)^2}\right)} & \text{if } \|\vec{x}-\vec{x}_c\| < r \\ 0 & \text{if } \|\vec{x}-\vec{x}_c\| \geq r \end{cases} \tag{37}$$

where $\vec{x}_c = (1, 0, 0)$ and $r = .3$ are the center and radius of the bump. Note that the advection method described above also applies to advecting a scalar field on the solid surface, if one omits the rotation of the tangential vector quantity to be advected. This scalar field is advected on the sphere by a constant vortex tangential velocity field $\vec{u}_s = \frac{\pi}{12}(-z, 0, x)$, while the sphere is rotating about the axis $(\frac{1}{2}, \frac{\sqrt{3}}{2}, 0)$ with an angular velocity of $\frac{\pi}{10}$. Typical numerical results are shown in Figure 12. Comparison of our numerical result with the analytic solution shows first order accuracy in both the L^1 norm and the L^∞ norm (Table 1).

In the second and the third examples, we test the full advection algorithm by self-advecting a tangential velocity field on rotating spheres. The tangential velocity field is initialized so that its streamlines align with the longitudes, originating from the north pole $(0, 1, 0)$ and ending at the south pole $(0, -1, 0)$. The magnitude of the tangential velocity is set as constant (we use $||\vec{u}_s|| = .1$) so that analytically this tangential



(a) t=0 (b) t=3 (c) t=6

Figure 12: A bump function is advected by a vortex velocity field on a sphere that is rotating about an axis different from the vortex. The isocontours of the bump function are shown as concentric circles of different colors. The streamlines of the vortex velocity field are shown as the latitudes of the sphere.

Average mesh edge length	L^1 Error	Order	L^∞ Error	Order
7.02×10^{-2}	5.27×10^{-3}	–	4.83×10^{-1}	–
2.34×10^{-2}	1.94×10^{-3}	.91	2.51×10^{-1}	.60
7.80×10^{-3}	7.07×10^{-4}	.92	1.02×10^{-1}	.82
2.60×10^{-3}	2.44×10^{-4}	.97	3.74×10^{-2}	.91

Table 1: The errors and orders of accuracy of advecting a bump function by a vortex velocity field on a rotating sphere ($t = 6$).

velocity field is steady during self-advection. At surface particles near the poles with $|y| > .8$, the tangential velocities are assigned analytic values as a boundary condition. The sphere is rotating at an angular velocity of $\frac{\pi}{10}$ and about axes $(0, 1, 0)$ and $(0, \frac{1}{2}, \frac{\sqrt{3}}{2})$ for the second and the third tests, respectively. In both tests, the numerical results converge to the analytic solution with first order accuracy (Tables 2 and 3).

6. Filling Uncovered Grid Faces

As the solid moves, some solid-solid grid faces will be uncovered and require valid fluid velocity values. Equations 20 and 21 dictate this motion of the solid, and thus new values for the fluid velocity are required at time t^n for use in Equation 23, etc. One can accomplish this with a number of methods such as extrapolating from nearby neighbors or using upwind information, see e.g. [23, 38, 24]. We use a level set description of the solid at time t^n for extrapolating time t^n velocities from solid-fluid and fluid-fluid grid faces to solid-solid grid faces along the lines of [1]. Then these ghost values of the fluid velocity on solid-solid faces can be readily used as new time t^n velocities for any solid-solid face which is uncovered by the motion of the solid.

The aforementioned extrapolation can be quite inaccurate at long distances from solid-fluid and fluid-fluid grid faces, especially for solid-solid faces uncovered deep within the gap region. We remedy this by using the

Average mesh edge length	L^1 Error	Order	L^∞ Error	Order
7.02×10^{-2}	2.76×10^{-3}	–	1.59×10^{-2}	–
2.34×10^{-2}	1.12×10^{-3}	.82	6.21×10^{-3}	.86
7.80×10^{-3}	3.84×10^{-4}	.97	2.13×10^{-3}	.97
2.60×10^{-3}	1.31×10^{-4}	.98	7.24×10^{-4}	.98

Table 2: The orders of accuracy of self-advecting a tangential velocity field with streamlines along the longitudes of a sphere which is rotating about the axis $(0, 1, 0)$. The errors are computed at time $t = 2$.

Average mesh edge length	L^1 Error	Order	L^∞ Error	Order
7.02×10^{-2}	1.87×10^{-3}	–	2.27×10^{-2}	–
2.34×10^{-2}	5.64×10^{-4}	1.09	7.42×10^{-3}	1.02
7.80×10^{-3}	1.83×10^{-4}	1.02	2.43×10^{-3}	1.02
2.60×10^{-3}	6.21×10^{-5}	.98	8.10×10^{-4}	1.00

Table 3: The orders of accuracy of self-advecting a tangential velocity field with streamlines along the longitudes of a sphere which is rotating about the axis $(0, \frac{1}{2}, \frac{\sqrt{3}}{2})$. The errors are computed at time $t = 2$.

velocity degree of freedom particles to directly set velocities at solid-solid grid faces whose dual cells intersect and only intersect solid surface triangles whose three vertices are all velocity degree of freedom particles. These solid-solid grid faces are then also used as Dirichlet boundary conditions for the extrapolation along with the solid-fluid and fluid-fluid grid faces. This is accomplished by first constructing a three-dimensional velocity vector on each velocity degree of freedom particle, using its tangential velocity and the normal component of the pointwise solid velocity at the particle's location. Then we interpolate velocities from particles to a solid-solid grid face using interpolation weights computed similarly to $\hat{\mathbf{J}}$ for a deformable body, except that the unprojected areas of subpolygons are used. Using unprojected areas is important because the newly exposed solid-solid face needs to capture both the normal and tangential motion of the surrounding fluid. After interpolating a full velocity vector to a solid-solid grid face, we compute the face velocity by taking the dot product between the face normal and the velocity vector as usual.

7. Examples and Validation

The modifications to Equation 29 that we have discussed in Section 4 retain the symmetric positive-definite nature of the system, and thus it can be solved via the preconditioned conjugate gradient (PCG) method. We utilize a block diagonal preconditioner in which we partition the variables into two sets corresponding to the pressure degrees of freedom on the grid $\hat{\mathbf{p}}_c$ and the rest of the variables $\hat{\mathbf{p}}_s, \boldsymbol{\lambda}$, and $\hat{\mathbf{v}}$. The diagonal block corresponding to the pressure degrees of freedom on the grid is preconditioned using incomplete Cholesky factorization, and the diagonal block corresponding to the rest of the variables is diagonally preconditioned. Using this approach, the number of iterations to converge was within the same order of magnitude as that for a similar sized simulation from [33] without gap flow in most of our test examples. However, we did observe a significant slowdown in convergence when the gap was *much* thinner than Δx. To alleviate this problem, we can set a tolerance for minimum gap thickness such as $.1\Delta x$.

Note that the top row of the matrix blocks of Equation 29 is weighted by the control volumes of fluid pressure degrees of freedom due to the use of the volume weighted divergence operator $-\mathbf{G}^T$. Thus, we follow the suggestion from [8] to compute an unweighted residual $\hat{\mathbf{r}}$ which is computed by dividing the elements of the residual corresponding to the top row of the matrix blocks of Equation 29 by the control volumes of fluid pressure degrees of freedom. Then, the PCG convergence criterion is set to be $|\hat{\mathbf{r}}|_\infty < 10^{-5}$. We find this tolerance sufficient since further reducing the tolerance did not change the significant digits we present in the numerical results.

In the examples below, all quantities are in SI units, and the gravitational force always points in the negative y direction. Various tables in this section use n as the number of grid cells per spatial dimension. We denote $H/\Delta x$ as the thickness of the gap divided by the grid cell size, N_{surface} as the number of fluid pressure degrees of freedom added to solid surfaces, and N_{grid} as the number of fluid pressure degrees of freedom on the grid.

7.1. Rigid blocks in a spatially constant pressure environment with zero gravity

In this test, two rigid blocks are placed inside a domain with zero gravity and Dirichlet constant pressure boundary conditions on the domain exterior. The analytic solution has spatially and temporally constant pressure everywhere and the two rigid bodies at rest. Figure 13 illustrates the problem in two and three

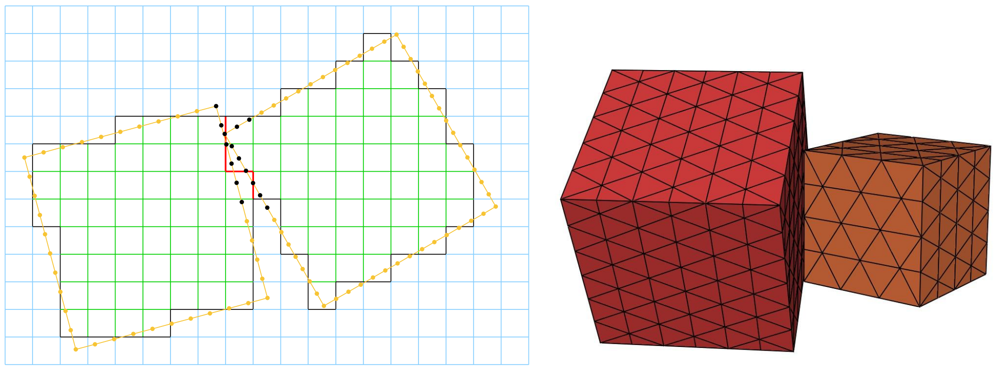


Figure 13: Two rigid blocks in a spatially constant pressure environment with zero gravity in two and three spatial dimensions. The surfaces of the two-dimensional blocks are tessellated using line segments, and the surfaces of the three-dimensional blocks are tessellated using isosceles triangles.

spatial dimensions. Our two-way coupled gap flow solver passed all such tests in two and three spatial dimensions for arbitrarily large pressures, various fluid grid and solid mesh resolutions, and various spatial configurations of the two solid bodies. This indicates that our solver computes the correct pressure and pressure forces in the gap region in order to balance the exterior fluid pressure.

7.2. Separation of underwater rigid blocks in contact

Consider the separation of two underwater rigid blocks initially in contact as illustrated in Figure 14. The two-dimensional examples are simulated in a domain of $[-.36, .36] \times [-.36, .36]$. Dirichlet pressure boundary conditions are set on all sides of the grid domain as $p = 9.8 \times 10^3(1 - y)$. One block of size $.6 \times .3$ is centered at $(0, .15)$ and kept kinematically stationary, while the other block of size $.5 \times .3$ is centered at $(0, -.15)$ and two-way coupled to the fluid.

If the surface of the two rigid blocks is perfectly flat with no presence of fluid in the gap between them, then there is no fluid pressure in the contact region and the bottom block will separate only if the gravitational force on it is greater than the hydrostatic pressure force on its bottom face. We denote this situation as *dry contact* throughout the rest of the paper. In order to model this numerically, we do not create any solid fluid coupling faces or fluid pressure degrees of freedom on solid surfaces in the dry contact region. Using this approach, we can calculate the minimum solid density for separation and compare it to the analytic solution in Table 4.

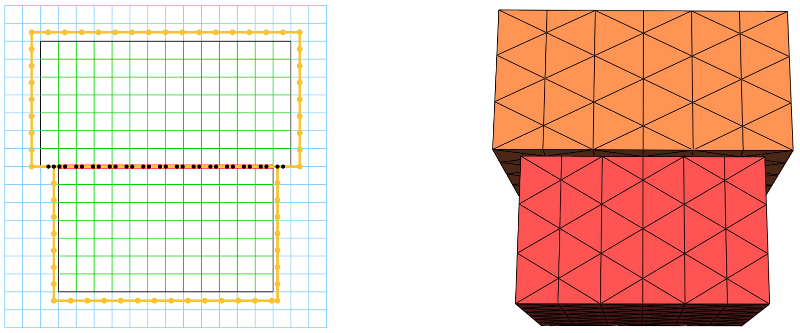


Figure 14: Illustration of the initial setup of the block separation example. The grid and solid mesh resolutions shown in the figure are lower than what were used in the actual simulations.

n	Minimum solid density	Error	Order
72	4316.7	16.6	–
144	4325.0	8.3	1.00
288	4329.2	4.1	1.02
576	4331.3	2.0	1.04

Table 4: Convergence results of the minimum solid density for the bottom block to separate when the two rigid blocks are in dry contact. The errors are computed by comparing to the analytic minimum solid density of 4333.3.

Immediately after the blocks separate enough for fluid to flow between them, the nature of the problem changes into *wet contact* which is the primary focus of our paper. While one typically has to wait for the fluid grid pressure degrees of freedom to appear in between the blocks in order to switch from dry to wet contact, our method alleviates this restriction. Moreover, for many scenarios the surfaces are not smooth enough or close enough to have the dry contact in the first place, and the wet contact is the appropriate solution. For example, a free rigid block with its top face initially touching a ceiling will fall as long as the block density is greater than the air density, instead of having to overcome the full atmospheric pressure. For the example under consideration, if even a very small amount of fluid exists between the two blocks, the analytic minimum solid density for the bottom block to separate will be identical to the water density. Numerically we assume the gap thickness between the two blocks in contact to be 10^{-4}, noting that our method does not preclude the use of a smaller gap thickness, and we follow Section 4 as usual to create solid fluid coupling grid faces and fluid pressure degrees of freedom on solid surfaces in the gap region. As shown in Table 5, the numerical solution to the minimum solid density for separation converges to the analytic solution.

n	Minimum solid density	Error	Order
72	972.1	27.9	–
144	986.0	14.0	.99
288	992.9	7.1	.98
576	996.4	3.6	.98

Table 5: Convergence results of the minimum solid density for the bottom block to separate when the two rigid blocks are in wet contact. The errors are computed by comparing to the analytic minimum solid density, i.e. the water density of 1000. The mesh edge length is set to 2.5×10^{-3} when $n = 72$ and is refined at the same rate as the grid.

We next consider a situation where the contact is partially dry and partially wet. For the example under consideration, we assume that the left half of the contact is initially dry and that the right half of the contact is wet. In this situation the unbalanced fluid pressure force will rotate the bottom block clockwise. In order to model this contact numerically, we only create solid fluid coupling faces and solid surface pressure degrees of freedom at the grid faces between two solids that lie in the wet contact region, while the dry contact region degenerates to the method in [33]. We analyze the angular acceleration of the bottom block after the first two-way coupled solve (Equation 19) in the first time step of a fixed size .01. Table 6 shows that the bottom block is rotating clockwise and the angular acceleration converges under fluid grid and solid mesh refinement. Once the bottom block has a nonzero rotation angle, the fluid flows into the initially dry contact region turning it into the wet contact. Figure 15 shows the snapshots of this simulation illustrating the bottom block tilting while falling.

7.3. Separation of underwater rigid blocks at two water depths

Under the wet contact assumption, when simulating rigid bodies two-way coupled to an incompressible fluid, the motion of the fluid and solids is invariant to any spatially constant change in the Dirichlet pressure boundary condition. We verify this by simulating the block separation examples in Section 7.2 at two different water depths in both two and three spatial dimensions.

n	Angular acceleration	Difference	Order
72	-0.5561	–	–
144	-0.6545	9.84×10^{-2}	–
288	-0.7026	4.81×10^{-2}	1.03
576	-0.7125	9.88×10^{-3}	2.28
1152	-0.7073	5.17×10^{-3}	.93

Table 6: Convergence results of the two-dimensional block separation example when the contact is half dry and half wet. The mesh edge length is set to 2.5×10^{-3} when $n = 72$ and is refined at the same rate as the grid. The density of the bottom block is set to be 5000 in this test. The thickness of the gap in the wet contact region is assumed to be 10^{-3}.

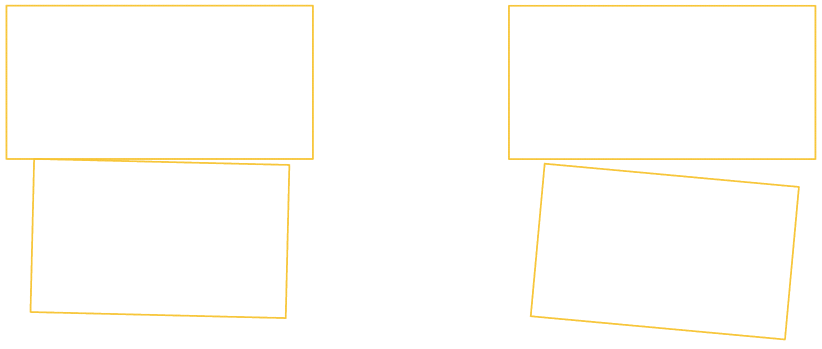

(a) t=.274 (b) t=.419

Figure 15: Starting from a contact that is dry on the left and wet on the right, the bottom block is rotating clockwise while falling. Once the bottom block has a nonzero rotation angle, the fluid flows in and wets the left half of the contact.

The two-dimensional examples use the same computational domain as in Section 7.2 with grid resolution 72×72. Dirichlet pressure boundary conditions are set on all sides of the grid domain as $p = 9.8 \times 10^3(10-y)$ and $p = 9.8 \times 10^3(2000 - y)$ for the shallower and deeper water cases, respectively. The positions and sizes of the two rigid blocks are the same as those in Section 7.2, while the density of the bottom block is set to 2×10^3. The edge length of the solid surface is set to 2.5×10^{-3}, and the gap thickness between the two rigid blocks is again assumed to be 10^{-4}. We analyze the acceleration of the bottom block resulting from the first two-way coupled solve in the first time step of a fixed size .01. Using the approach in [33] without the gap flow treatment, the two blocks are incorrectly pushed against each other with the force that increases with the surrounding pressure. On the other hand, our method produces the same downward acceleration invariant of a spatially constant change in the surrounding pressure, see Table 7. The three-dimensional example is simulated in a domain of $[-.36, .36] \times [-.36, .36] \times [-.36, .36]$ with grid dimensions of $72 \times 72 \times 72$. The Dirichlet pressure boundary conditions are the same as for the two-dimensional example. The top block is size $.6 \times .3 \times .6$ and centered at $(0, .15, 0)$, while the bottom block of density 2×10^3 is size $.5 \times .3 \times .5$ and centered at $(0, -.15, 0)$. The average edge length of the solid surface meshes is 8.96×10^{-3}. The gap thickness between the two rigid blocks is assumed to be 10^{-3}. The results are consistent with those in two spatial dimensions, see Table 8.

	$p = 9.8 \times 10^3(10 - y)$	$p = 9.8 \times 10^3(2000 - y)$
Method in [33]	1.429×10^2	2.948×10^4
Our method	-1.381×10^{-2}	-1.381×10^{-2}

Table 7: The y-component of the bottom block acceleration after the first two-way coupled solve in the two-dimensional block separation example.

	$p = 9.8 \times 10^3(10 - y)$	$p = 9.8 \times 10^3(2000 - y)$
Method in [33]	1.439×10^2	2.968×10^4
Our method	-2.661×10^{-1}	-2.661×10^{-1}

Table 8: The y-component of the bottom block acceleration after the first two-way coupled solve in the three-dimensional block separation example.

We further test the convergence of these examples under fluid grid and solid mesh refinement. The orders of accuracy are computed using three successive resolutions and shown in Tables 9 and 10. For a given fluid grid resolution, modeling thinner gaps between solids leads to larger control volume ratios between the pressure degrees of freedom on the fluid grid and those in the gap, typically requiring more PCG iterations. Here we report the number of PCG iterations required to converge as a function of the thickness of the wet contact gap in Table 11.

n	Acceleration	Difference	Order
72	-1.381×10^{-2}	–	–
144	-1.391×10^{-2}	1.06×10^{-4}	–
288	-1.397×10^{-2}	5.06×10^{-5}	1.06
576	-1.399×10^{-2}	2.21×10^{-5}	1.20
1152	-1.400×10^{-2}	7.40×10^{-6}	1.58

Table 9: Convergence results of the two-dimensional block separation example. The mesh edge length is set to 2.5×10^{-3} when $n = 72$ and is refined at the same rate as the grid.

n	Acceleration	Difference	Order
72	-2.661×10^{-1}	–	–
144	-2.832×10^{-1}	1.71×10^{-2}	–
288	-2.916×10^{-1}	8.41×10^{-3}	1.02
576	-2.953×10^{-1}	3.65×10^{-3}	1.20

Table 10: Convergence results of the three-dimensional block separation example. The average mesh edge length is 8.96×10^{-3} when $n = 72$ and is refined at the same rate as the grid.

7.4. Separation of underwater rigid blocks with a nonzero initial distance

We further modify these examples to have a nonzero gap between the surface meshes of the two blocks. This allows one to compare the results obtained at grid resolutions too coarse to resolve the gap to the results obtained at grid resolutions fine enough to resolve the gap, illustrating the convergence of our gap flow solver to the standard solver in the case where the standard solver produces an adequate solution. We initially place the two blocks such that there is exactly one layer of grid cells in the gap, and then refine the grid from this resolution to obtain the ground truth. In the two-dimensional example, we set the initial position of the center of the bottom block at $(0, -.1525)$ so that there is exactly one layer of grid cells in the gap of thickness 2.5×10^{-3} at a grid resolution of 288×288. Table 12 shows the result under refinement. Next, we compute the results for another sequence of grid resolutions in which the initial gap thickness is integer plus one half grid cell sizes. Table 13 shows that the errors are much larger than those in Table 12. However, using the last row of Table 12 as ground truth indicates that the results in Table 13 are still converging with first order accuracy. We also tested the convergence of other sequences of grid resolutions such as $H/\Delta x = .75$, 1.75, etc. and obtained a first order convergence rate in all of them. In summary, the grid based solutions converge, but with a degree of noise based on a particular fraction of grid cells that occur in the gap region.

$\hat{H}$	$\hat{H}/\Delta x$	PCG iterations (2D)	PCG iterations (3D)
2×10^{-3}	.2	1,498	2,106
10^{-3}	.1	4,533	6,709
5×10^{-4}	.05	15,808	23,573
2.5×10^{-4}	.025	53,992	95,552

Table 11: Number of PCG iterations as a function of the wet contact gap thickness $\hat{H}$ for both the two-dimensional and the three-dimensional block separation tests. The grid resolution is $n = 72$, and the number of degrees of freedom is fixed for this analysis. For the two-dimensional example, $N_{\text{surface}} = 402$ and $N_{\text{grid}} = 1,884$. For the three-dimensional example, $N_{\text{surface}} = 7,502$ and $N_{\text{grid}} = 190,248$.

n	$H/\Delta x$	Acceleration	Difference	Order
288	1	-3.03019×10^{-1}	–	–
576	2	-3.02958×10^{-1}	6.1×10^{-5}	–
1152	4	-3.02789×10^{-1}	1.69×10^{-4}	-1.47
2304	8	-3.02656×10^{-1}	1.33×10^{-4}	.35
4608	16	-3.02572×10^{-1}	8.4×10^{-5}	.66
9216	32	-3.02524×10^{-1}	4.8×10^{-5}	.81
18432	64	-3.02497×10^{-1}	2.7×10^{-5}	.83

Table 12: Convergence results of the two-dimensional block separation example with the gap resolved by integer multiples of grid cells.

Next, we coarsen the grid from the lowest grid resolution in Table 13 so that the gap cannot be resolved by the grid requiring the use of our gap flow solver. The results with the use of the gap flow solver are shown in Table 14. Note that the errors at these lower grid resolutions with the use of the gap flow solver are typically even smaller than those at higher grid resolutions such as those depicted in Table 13. We report the number of PCG iterations required to converge as a function of the resolution in Table 18. The gap flow solver increases the number of PCG iterations required to converge; however, the gap flow solver greatly increases the overall efficiency by allowing one to obtain even smaller errors at lower grid resolutions.

In the three-dimensional example, we set the initial position of the center of the bottom block to $(0, -.1525, 0)$ so that there is one layer of grid cells in the gap of thickness 2.5×10^{-3} at grid resolution $288 \times 288 \times 288$, see Table 15. The results using a gap thickness that is integer plus one half grid cells are shown in Table 16. The results on coarser grids using the gap flow solver are shown in Table 17. We report the number of PCG iterations required to converge as a function of the resolution in Table 19.

n	$H/\Delta x$	Acceleration	Error compared to $n = 18432$	Order
144	.5	-5.424×10^{-1}	2.40×10^{-1}	–
432	1.5	-3.882×10^{-1}	8.57×10^{-2}	.94
720	2.5	-3.547×10^{-1}	5.22×10^{-2}	.97
1008	3.5	-3.400×10^{-1}	3.75×10^{-2}	.98
1296	4.5	-3.318×10^{-1}	2.93×10^{-2}	.98
1584	5.5	-3.265×10^{-1}	2.40×10^{-2}	.99
3024	10.5	-3.151×10^{-1}	1.26×10^{-2}	1.00
4464	15.5	-3.111×10^{-1}	8.6×10^{-3}	.98
5904	20.5	-3.090×10^{-1}	6.5×10^{-3}	1.00

Table 13: Convergence results of the two-dimensional block separation example with the gap resolved by integer and a half multiples of grid cells.

n	$H/\Delta x$	Acceleration	Error compared to $n = 18432$
18	.0625	-3.391×10^{-1}	3.66×10^{-2}
36	.125	-3.590×10^{-1}	5.65×10^{-2}
72	.25	-3.120×10^{-1}	9.5×10^{-3}

Table 14: Results at grid resolutions requiring the use of the gap flow solver in the two-dimensional block separation example.

n	$H/\Delta x$	Acceleration	Difference	Order
288	1	-6.4416×10^{-1}	–	–
576	2	-6.4462×10^{-1}	4.6×10^{-4}	–
1152	4	-6.4449×10^{-1}	1.3×10^{-4}	1.82

Table 15: Convergence results of the three-dimensional block separation example with the gap resolved by integer multiples of grid cells.

7.5. Separation of underwater rigid blocks with rounded corners

We repeat the experiments from Section 7.4 using a bottom block with smooth curvature as opposed to sharp corners, see Figure 16. In the two-dimensional case, the integer and integer and a half refinement cases are shown in Tables 20 and 21, respectively. Note that we again tested other sequences of grid resolutions, such as $H/\Delta x = .75$, 1.75, etc., and once again the results are similar. The results for lower grid resolutions with the use of the gap flow solver are shown in Table 22. Similarly, the three-dimensional results are shown in Tables 23, 24, and 25. It is interesting to note that the errors in Tables 22 and 25 seem better behaved than those in Tables 14 and 17. Generally speaking, we do not expect the errors to be well behaved or even decrease with increasing grid resolutions in Tables 14, 17, 22, and 25, as we may not be in the asymptotic regime for convergence. Of course, further refining the grids should eventually lead to convergence of the solutions. However, further refining leads to the presence of grid cells in the gap thus turning off the gap flow solver. Therefore, our goal is to create a gap flow solver, i.e. *model*, which does a good job approximating the behavior in the gap region when it is not resolved by the grid – small errors in these tables seem to indicate this is true. We report the number of PCG iterations required to converge in Tables 26 and 27.

7.6. Collisions between underwater rigid blocks

We test the convergence of our method in an example where two underwater rigid blocks collide and then separate. To handle the collision and contact between rigid bodies, we follow the method in [14]. However, when a surface particle of one solid penetrates another solid, [14] uses the closest surface triangle

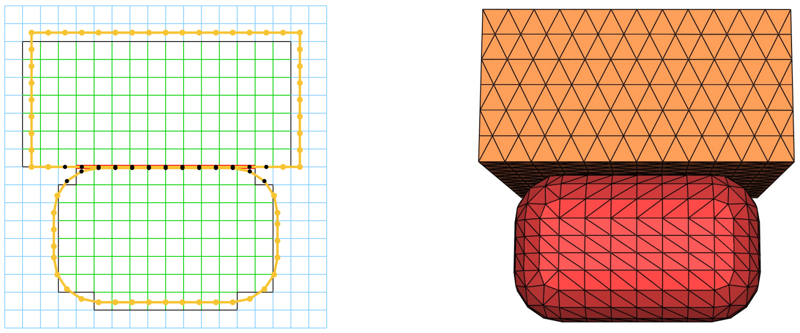


Figure 16: Illustration of the initial setup of the rounded corner block separation example. The curvature radius of the rounded corners is .1 in both the two and three-dimensional examples. The grid and solid mesh resolutions shown in the figure are lower than what were used in the actual simulations.

n	$H/\Delta x$	Acceleration	Error compared to $n = 1152$	Order
144	.5	-1.0730	4.29×10^{-1}	–
432	1.5	-8.0437×10^{-1}	1.60×10^{-1}	.90
720	2.5	-7.4272×10^{-1}	9.82×10^{-2}	.96

Table 16: Convergence results of the three-dimensional block separation example with the gap resolved by integer and a half multiples of grid cells.

n	$H/\Delta x$	Acceleration	Error compared to $n = 1152$
18	.0625	-5.3515×10^{-1}	1.09×10^{-1}
36	.125	-6.6424×10^{-1}	1.97×10^{-2}
72	.25	-6.2356×10^{-1}	2.09×10^{-2}

Table 17: Results at grid resolutions requiring the use of the gap flow solver in the three-dimensional block separation example.

to determine a direction for alleviating interpenetration. Motivated by [7], we instead use continuous collision detection to determine which face of the other solid the penetrating particle went through and use the point of penetration to determine the direction for collision, contact, etc. In particular, we use the face normal of the triangle the particle penetrated through in all the collision and contact algorithms.

We set the coefficients of restitution of both rigid blocks to be zero. We choose the time step by setting the CFL number to 3 noting that either advection on the grid or advection on the surface mesh will determine the maximum allowable time step. For the two-way coupled solve, Dirichlet pressure boundary conditions are set on all sides of the domain as $p = 9.8 \times 10^3(2000 - y)$. In the two-dimensional example, the grid domain is $[-.72, .72] \times [-.72, .72]$, and one block of density 2×10^3 and size $.4 \times .2$ is centered at $(0, .45)$, while the other block of density 5×10^2 and size $.2 \times .2$ is centered at $(.1, .15)$, as illustrated in Figure 17. The two blocks start from zero velocity and move toward each other due to their differing densities. Figure 18 shows positions and orientations of the two-dimensional rigid blocks under refinement. Figure 19 computes the convergence rates using either the average change in the positions of the eight corners of the two blocks or the worst corner. For the sake of comparison, we ran the same example using the method of [33] and obtained spurious non-convergent results, see e.g. Figure 20.

A similar three-dimensional block collision example is constructed by placing one block of density 2×10^3 and size $.4 \times .2 \times .2$ at $(0, .45, 0)$ and another block of density 5×10^2 and size $.2 \times .2 \times .2$ at $(.1, .15, 0)$, with grid dimensions $[-.72, .72] \times [-.72, .72] \times [-.72, .72]$, as illustrated in Figure 21. Figures 22 and 23 show snapshots and convergence results both during collision and after separation.

7.7. Omitting tangential velocity self-advection in the gap

We note that one can obtain a reasonable gap flow model even if advection of the velocity field is ignored in the gap. When the solid is moving, disabling velocity advection on the solid surface has negligible performance benefits as the advection routine is still required for remapping tangential velocities into their correct locations and directions as discussed in Section 5. Thus, we tested a modified method in which we omit *both* velocity advection on solid surfaces and the remapping step. Figure 24 shows that the results obtained for the underwater collision test in two spatial dimensions. Note that the results obtained with the simplified approach are very close to those obtained with the full approach – moreover, bigger differences are obtained with a single grid refinement. See also Figure 25.

n	$H/\Delta x$	N_{surface}	N_{grid}	PCG iterations	PCG solve time
18	.0625	98	130	795	.0134 s
36	.125	194	486	857	.0331 s
72	.25	402	1,884	1,037	.125 s
288	1	0	30,144	699	1.05 s

Table 18: Statistics at grid resolutions requiring the use of the gap flow solver compared with those at grid resolution $n = 288$ where the gap is resolved by one layer of grid cells for the two-dimensional block separation example. Note that the gap flow solver is not required for $n = 288$.

n	$H/\Delta x$	N_{surface}	N_{grid}	PCG iterations	PCG solve time
18	.0625	577	3,308	2,080	.683 s
36	.125	1,891	24,516	1,707	3.21 s
72	.25	7,502	190,248	1,526	25.2 s
288	1	0	12,175,872	783	1134 s

Table 19: Statistics at grid resolutions requiring the use of the gap flow solver compared with those at grid resolution $n = 288$ where the gap is resolved by one layer of grid cells for the three-dimensional block separation example. Note that the gap flow solver is not required for $n = 288$.

n	$H/\Delta x$	Acceleration	Difference	Order
288	1	-6.4772×10^{-1}	–	–
576	2	-6.5685×10^{-1}	9.13×10^{-3}	–
1152	4	-6.6103×10^{-1}	4.18×10^{-3}	1.13
2304	8	-6.6393×10^{-1}	2.90×10^{-3}	.53
4608	16	-6.6485×10^{-1}	9.2×10^{-4}	1.66

Table 20: Convergence results of the two-dimensional rounded corner block separation example with the flat part of the gap region resolved by integer multiples of grid cells. The average mesh edge length is 6.25×10^{-4} at grid resolution $n = 288$ and is refined at the same rate as the grid.

n	$H/\Delta x$	Acceleration	Error compared to $n = 4608$	Order
144	.5	-8.9712×10^{-1}	2.32×10^{-1}	–
432	1.5	-7.5198×10^{-1}	8.71×10^{-2}	.89
720	2.5	-7.1999×10^{-1}	5.51×10^{-2}	.90
1008	3.5	-7.0360×10^{-1}	3.88×10^{-2}	1.05

Table 21: Convergence results of the two-dimensional rounded corner block separation example with the flat part of the gap region resolved by integer and a half multiples of grid cells.

n	$H/\Delta x$	Acceleration	Error compared to $n = 4608$
18	.0625	-7.1644×10^{-1}	5.16×10^{-2}
36	.125	-6.9057×10^{-1}	2.57×10^{-2}
72	.25	-6.7862×10^{-1}	1.38×10^{-2}

Table 22: Results at grid resolutions requiring the use of the gap flow solver in the two-dimensional rounded corner block separation example.

n	$H/\Delta x$	Acceleration	Difference	Order
288	1	-1.4247	–	–
576	2	-1.4431	1.84×10^{-2}	–
1152	4	-1.4513	8.2×10^{-3}	1.17

Table 23: Convergence results of the three-dimensional rounded corner block separation example with the flat part of the gap region resolved by integer multiples of grid cells. The average mesh edge length is 1.15×10^{-3} at grid resolution $n = 288$ and is refined at the same rate as the grid.

n	$H/\Delta x$	Acceleration	Error compared to $n = 1152$	Order
144	.5	-1.7487	2.92×10^{-1}	–
432	1.5	-1.5711	1.20×10^{-1}	.83
720	2.5	-1.5316	8.03×10^{-2}	.78

Table 24: Convergence results of the three-dimensional rounded corner block separation example with the flat part of the gap region resolved by integer and a half multiples of grid cells.

n	$H/\Delta x$	Acceleration	Error compared to $n = 1152$
18	.0625	-1.8412	3.90×10^{-1}
36	.125	-1.5634	1.12×10^{-1}
72	.25	-1.4760	2.47×10^{-2}

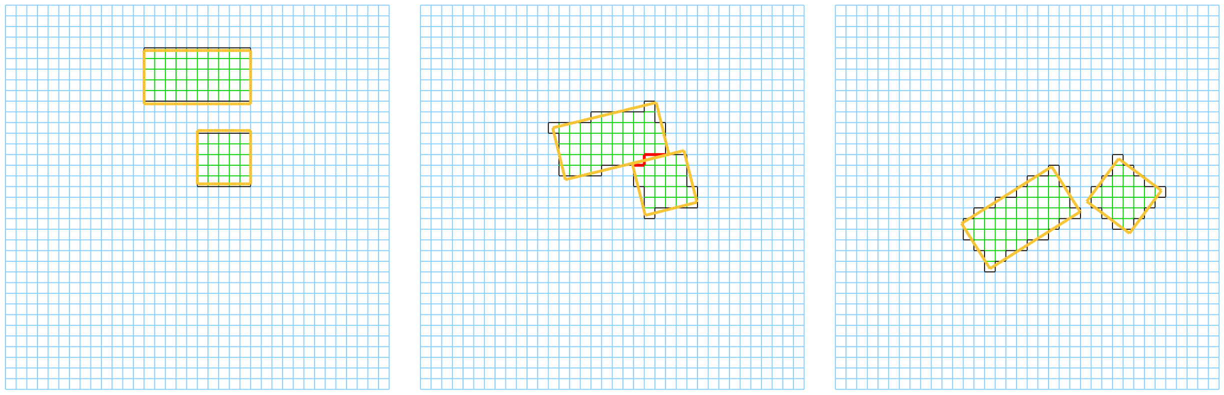
Table 25: Results at grid resolutions requiring the use of the gap flow solver in the three-dimensional rounded corner block separation example.

n	$H/\Delta x$	N_{surface}	N_{grid}	PCG iterations	PCG solve time
18	.0625	84	129	738	.0115 s
36	.125	148	490	662	.0239 s
72	.25	276	1,968	684	.0798 s
288	1	0	31,520	571	.888 s

Table 26: Statistics at grid resolutions requiring the use of the gap flow solver compared with those at grid resolution $n = 288$ where the gap is resolved by one layer of grid cells for the two-dimensional rounded corner block separation example. Note that the gap flow solver is not required for $n = 288$.

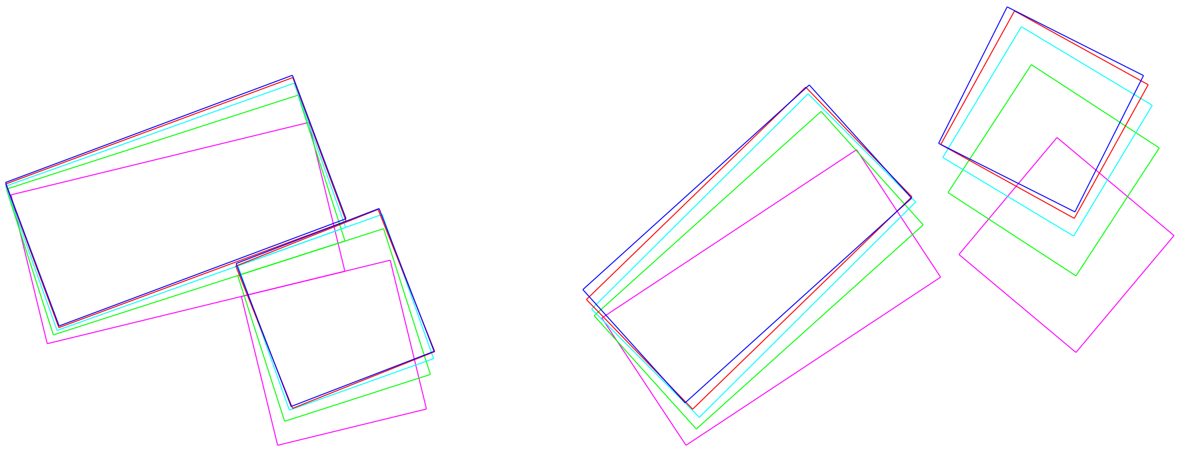
n	$H/\Delta x$	N_{surface}	N_{grid}	PCG iterations	PCG solve time
18	.0625	1,354	3,444	1,491	.709 s
36	.125	4,096	25,276	1,383	3.21 s
72	.25	13,924	199,968	829	16.2 s
288	1	0	12,805,056	561	936 s

Table 27: Statistics at grid resolutions requiring the use of the gap flow solver compared with those at grid resolution $n = 288$ where the gap is resolved by one layer of grid cells for the three-dimensional rounded corner block separation example. Note that the gap flow solver is not required for $n = 288$.



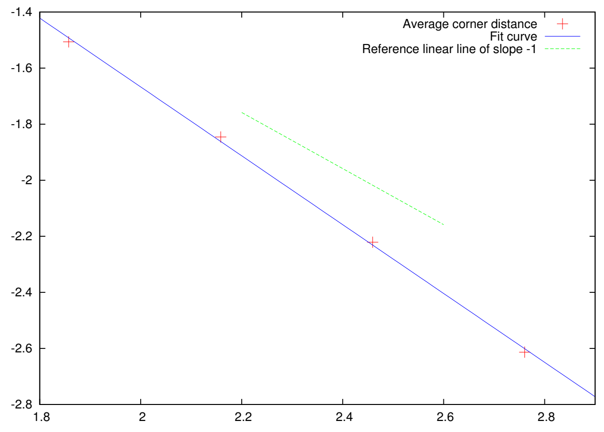
(a) $t = 0$ (b) $t = .45$ (c) $t = .70$

Figure 17: Two underwater rigid blocks collide and separate in two spatial dimensions.

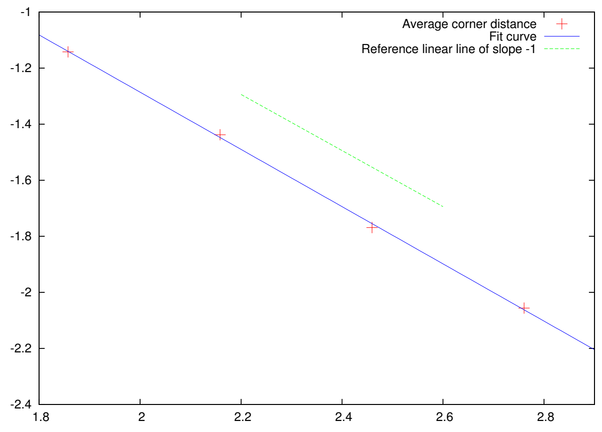


(a) $t = .45$ (b) $t = .70$

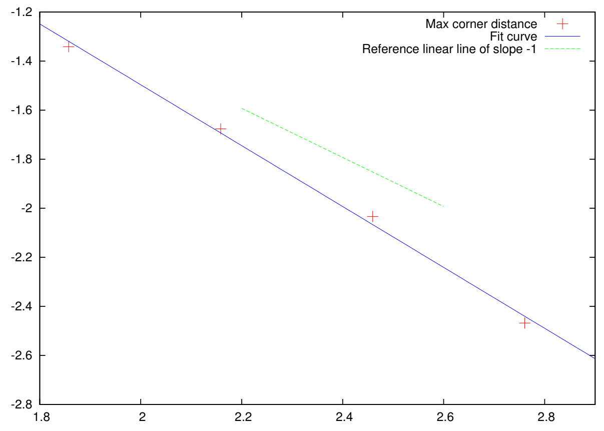
Figure 18: The positions and orientations of the two-dimensional rigid blocks from the results of different grid resolutions, shown in different colors: magenta 36×36, green 72×72, cyan 144×144, red 288×288, and blue 576×576. The mesh edge length is .01 when the grid resolution is 36×36 and is refined at the same rate as the grid cell size.



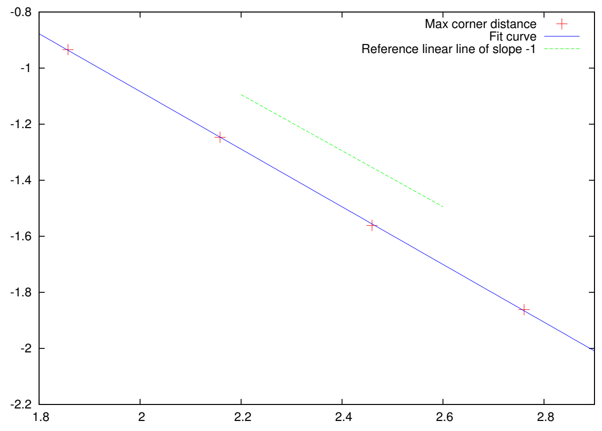
(a) Avg. corner distance converges at a rate of 1.23 at $t = .45$



(b) Avg. corner distance converges at a rate of 1.02 at $t = .70$



(c) Max corner distance converges at a rate of 1.24 at $t = .45$



(d) Max corner distance converges at a rate of 1.03 at $t = .70$

Figure 19: The convergence rate of corner positions of the two rigid blocks in two spatial dimensions. In figures (a) and (b) we compute the average distance between corresponding rigid block corners for two successive resolutions. In figures (c) and (d) we compute the maximum distance between corresponding rigid block corners. The convergence rate is computed as the slope of the linear least squares fit on the log-log scale.

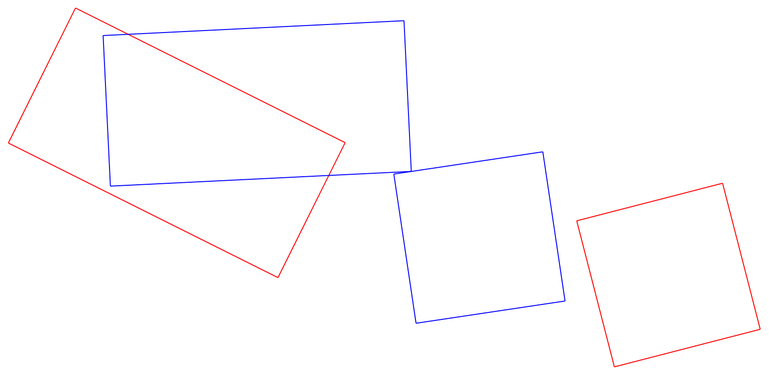
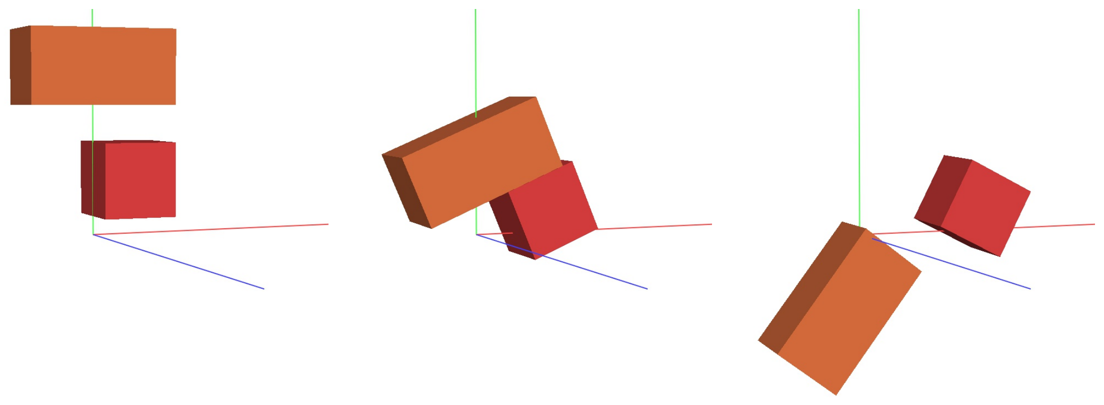
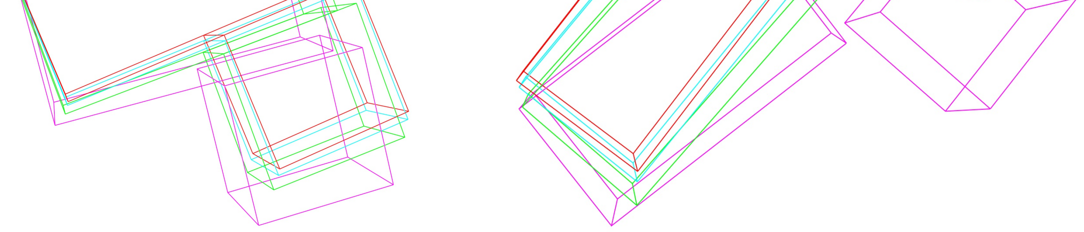


Figure 20: The results of the two-dimensional block collision example at $t = .45$ using the method of [33], with the same set of grid resolutions as in Figure 18. Note that for lower resolutions 36×36, 72×72, 144×144 the blocks are not shown in the figure because they have received spuriously high forces and have moved out of the computational domain.



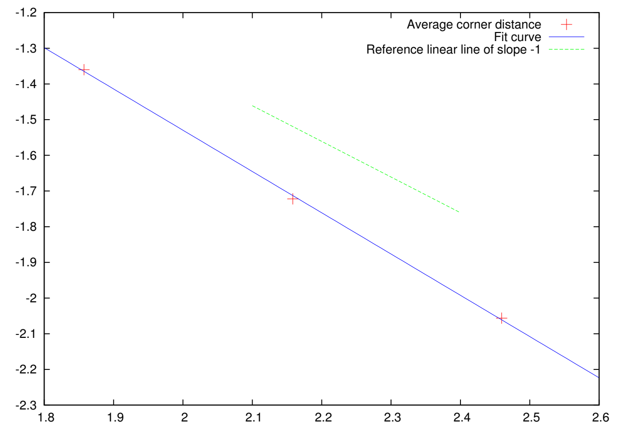
(a) $t = 0$ (b) $t = .45$ (c) $t = .75$

Figure 21: Two underwater rigid blocks collide and separate in three spatial dimensions.

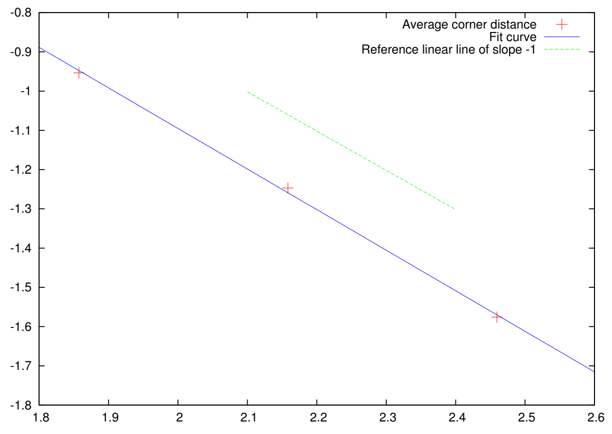


(a) $t = .45$ (b) $t = .75$

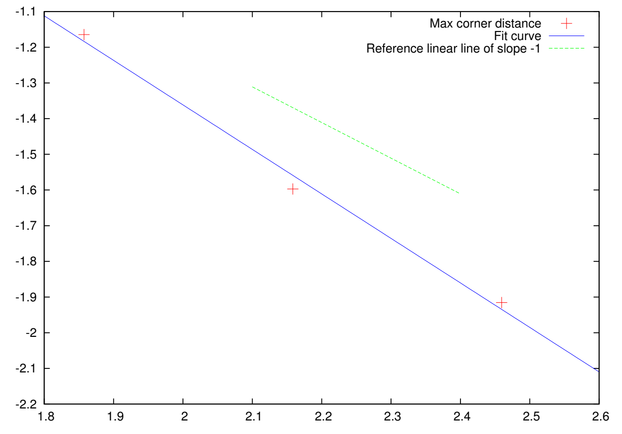
Figure 22: The positions and orientations of the three-dimensional rigid blocks from the results of different grid resolutions, shown in different colors: magenta $36 \times 36 \times 36$, green $72 \times 72 \times 72$, cyan $144 \times 144 \times 144$, and red $288 \times 288 \times 288$. The average mesh edge length is 2.65×10^{-2} when the grid resolution is $36 \times 36 \times 36$, and is refined at the same rate as the grid.



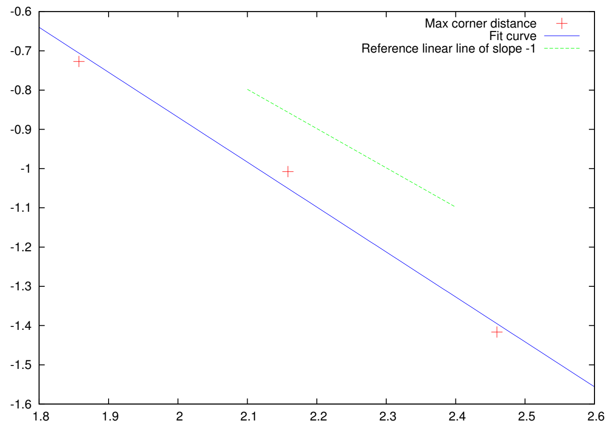
(a) Avg. corner distance converges at a rate of 1.16 at $t = .45$



(b) Avg. corner distance converges at a rate of 1.03 at $t = .75$

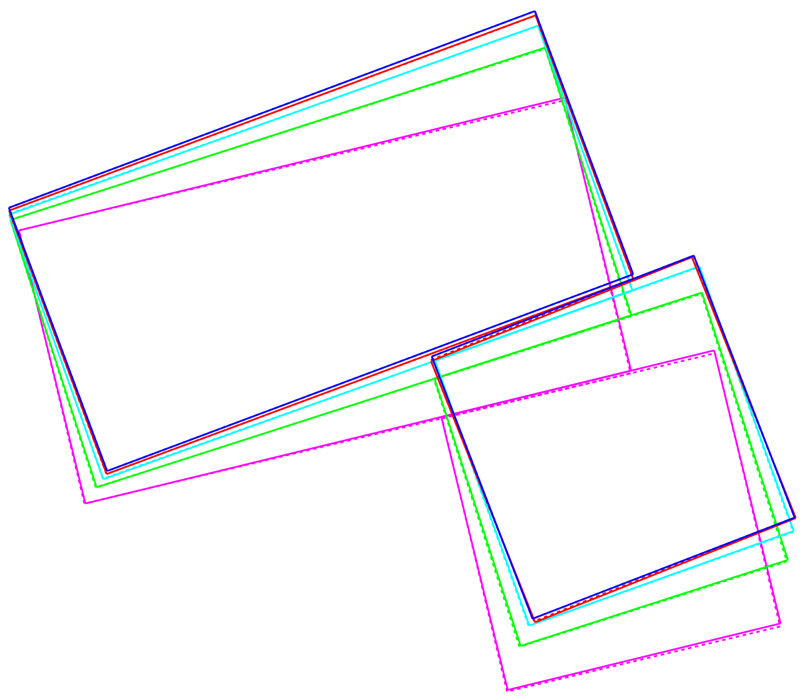


(c) Max corner distance converges at a rate of 1.25 at $t = .45$

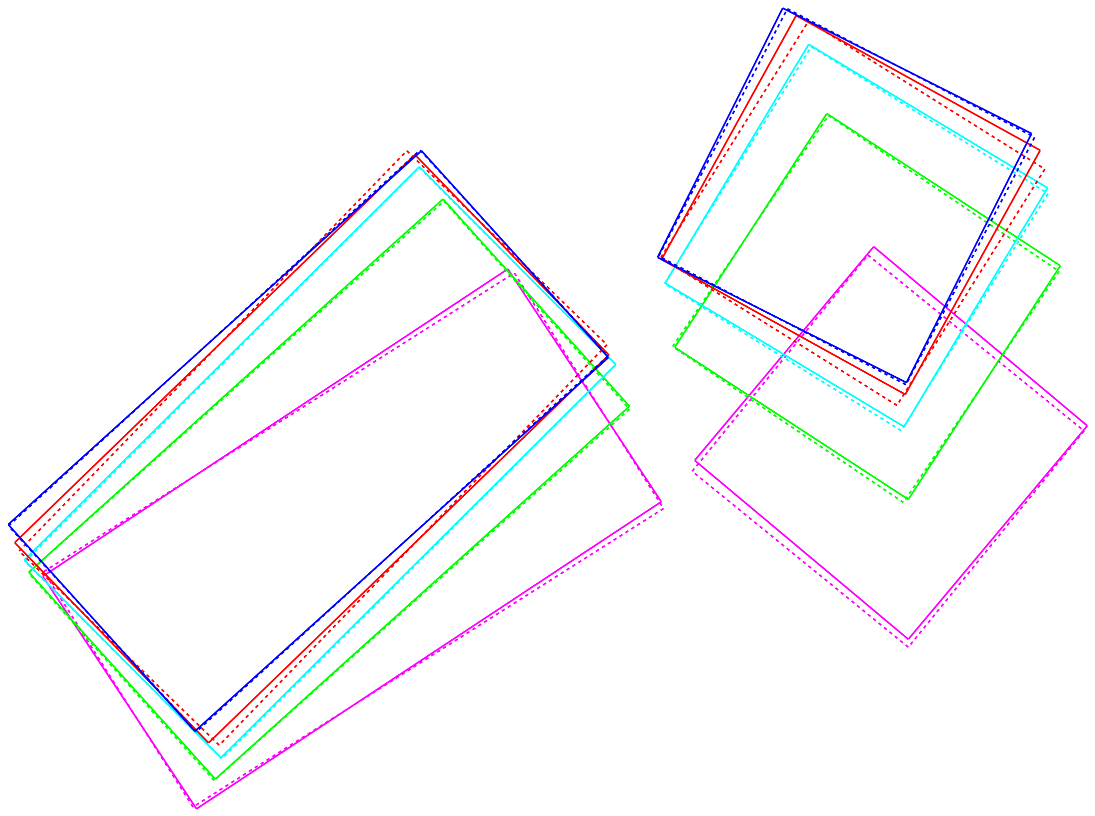


(d) Max corner distance converges at a rate of 1.14 at $t = .75$

Figure 23: The convergence rate of corner positions of the two rigid blocks in three spatial dimensions. In figures (a) and (b) we compute the average distance between corresponding rigid block corners for two successive resolutions. In figures (c) and (d) we compute the maximum distance between corresponding rigid block corners. The convergence rate is computed as the slope of the linear least squares fit on the log-log scale.

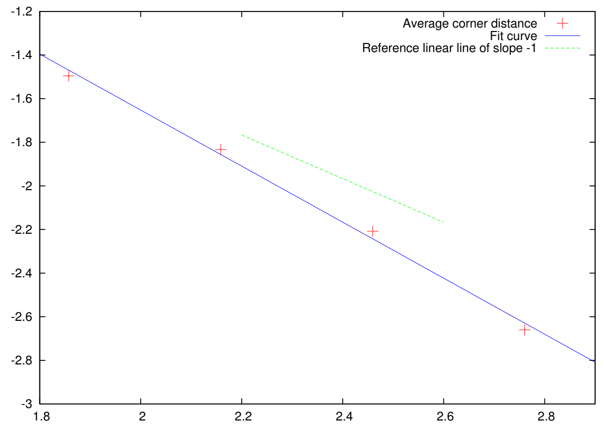


(a) $t = .45$

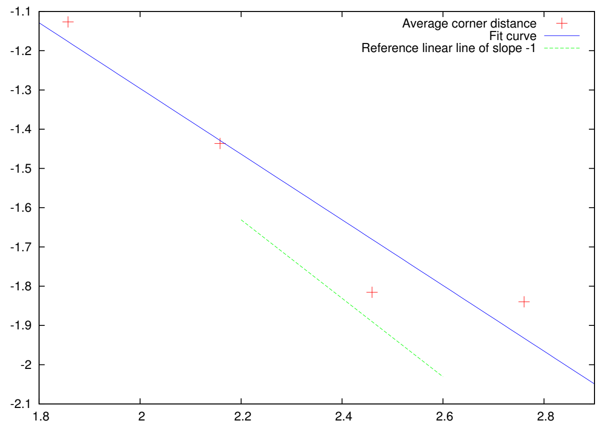


(b) $t = .70$

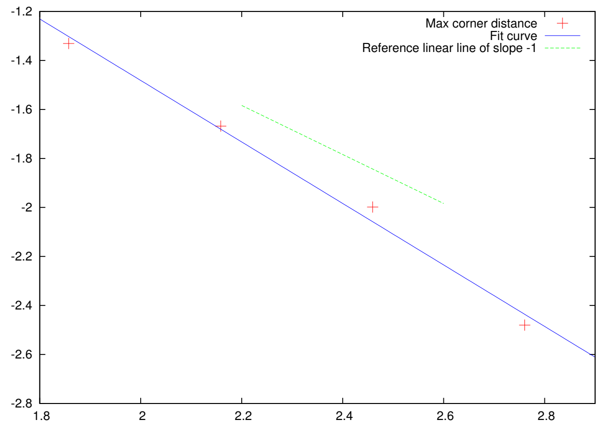
Figure 24: Comparison between the results from using the full algorithm (solid lines, same as Figure 18) and those from omitting both gap advection and the remapping step (dotted lines) in the two-dimensional block collision example. Different colors indicate different resolutions: magenta 36×36, green 72×72, cyan 144×144, red 288×288, blue 576×576.



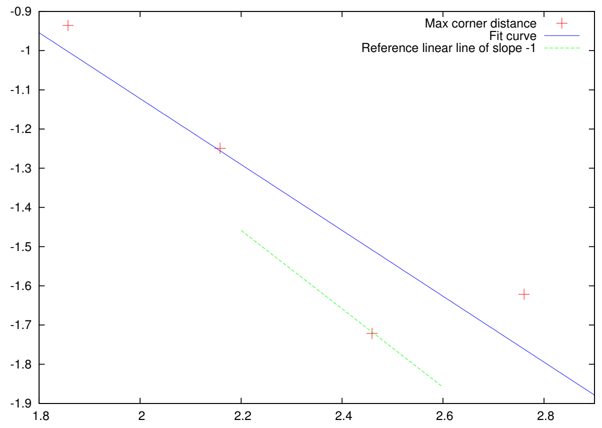
(a) Avg. corner distance converges at a rate of 1.28 at $t = .45$



(b) Avg. corner distance converges at a rate of .84 at $t = .70$



(c) Max corner distance converges at a rate of 1.26 at $t = .45$



(d) Max corner distance converges at a rate of .84 at $t = .70$

Figure 25: The convergence rate of the positions of all the corners of the two rigid blocks for the results omitting both gap advection and the remapping step.

7.8. Two interlocking gears immersed in single phase inviscid incompressible flow

Similar to [40, 41], we design a simplified two-dimensional example with two interlocking involute gears immersed in single phase inviscid incompressible flow. A 26-tooth gear centered at $(-.33, 0)$ with a pitch diameter of .8 is meshed with a 17-tooth pinion on the right side. The pressure angle is 25°. The addenda are both .02 and the deddenda of the gear and the pinion are .025 and .0245, respectively. The gear and the pinion are both kinematically driven at a pitch line speed of 40. Backlash as thick as 5% of the circular pitch is introduced and evenly distributed to the front and rear flanks, so that on a sufficiently fine grid the gap between the gear and the pinion can be resolved. The surface meshes of the two gears are made such that there are 8 segments on each involute profile and 4 segments on each top land as well as each bottom land, as shown in Figure 26. The computational domain is $[-21, 21] \times [-14, 14]$ with zero gravity and Dirichlet pressure boundary conditions $p = 0$ on all sides. We apply the Chimera grid method in [8] for domain extension, so that the domain boundary is far away from the two gears. We list the domains and cell sizes of the overlapping grids in Table 28, noting that the two gears are fully inside the third grid.

We analyze the torques exerted on the gear and the pinion due to the fluid pressure and compare the results of the gap flow solver to the results on sufficiently fine grids that resolve the gap. In order to more efficiently obtain the ground truth solutions on the grids without the gap flow solver, we add a local refinement grid of domain $[-.01, .13] \times [-.12, .12]$ and cell size $\Delta x = .06/n$ to resolve the gap region between the two gears. The results under grid refinement are obtained in Figure 27, where the gap region is resolved by the local refinement grid for all those grid resolutions. Note that the time-axes in Figure 27 start from .006, since we are interested in the behaviors after the magnitude and the period of the torques have become steady. The positive directions of the torque-axes are all chosen to be the directions that resist the motion of the gear/pinion. Thus, it is observed that on average the torques due to the fluid pressure are resisting the motion of both the gear and the pinion as expected, and the torque on the pinion differs exactly π radians in phase from the torque on the gear since the gear and the pinion are interlocked and in the gap region the fluid pressure forces push the gear and the pinion in opposite directions. The high frequency noise in Figure 27 (a) and (b) is due to the non-physical temporal oscillations of the fluid pressure field which arise when grid cells are covered or uncovered by moving solids in a non-conservative fashion, as pointed out by [36].

Next, we remove the local refinement grid so that the gap region is not adequately resolved by the fluid grid. Figure 28 shows that the results obtained from the gap flow solver agree with those obtained by adding the local refinement grid very well, while the results obtained without the local refinement grid or the gap flow solver seem erroneous.

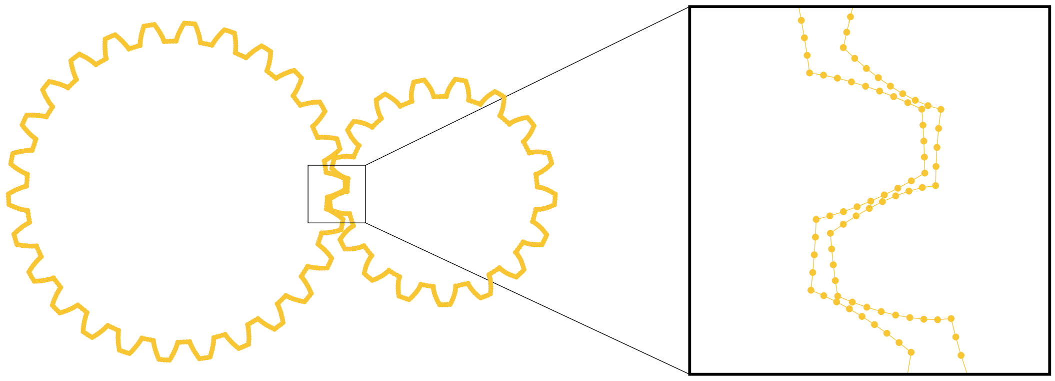


Figure 26: Illustration of the two interlocking gears and their surface meshes.

We compare the computational cost of the fine grid calculation to that of the gap flow solver in Table 29. Although the gap flow solver is only 4.4 times faster than using the local grid refinement for each PCG solve in terms of wall clock time, the overall simulation is about 35 times faster because of the ability to use larger

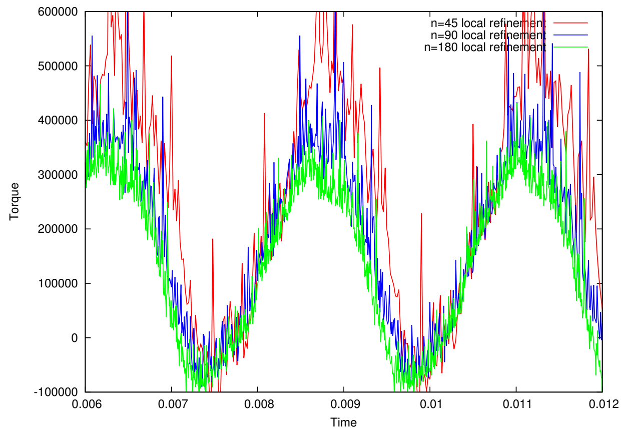
	Domain	Δx
1	$[-21, 21] \times [-14, 14]$	$42/n$
2	$[-2, 2] \times [-1, 1]$	$3/n$
3	$[-1, 1] \times [-.5, .5]$	$.5/n$

Table 28: The domains and cell sizes of the grids used in the example of two interlocking gears. n indicates the number of grid cells in the x direction on the coarsest grid, i.e. the first grid.

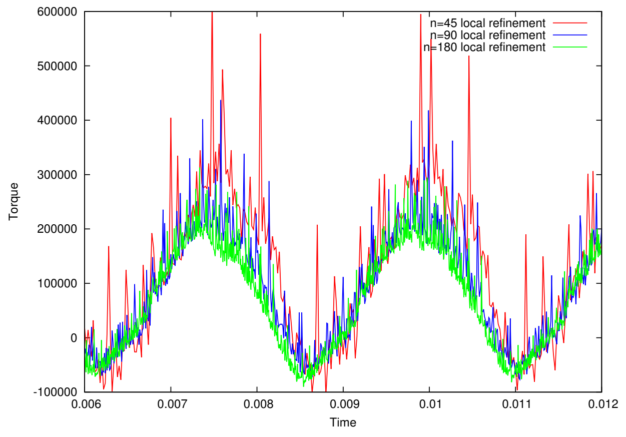
time steps on coarser grids. Moreover, decreasing the backlash makes the local grid refinement approach even slower without adversely affecting the gap flow solver at all.

	Typical N_{surface}	Typical N_{grid}	Typical iteration	Typical PCG solve time
Gap flow solver	55	41,862	337	.727 s
Local refinement grid	0	53,610	1,206	3.19 s

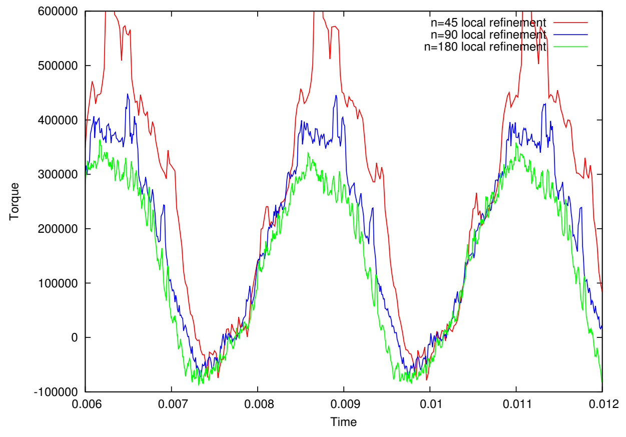
Table 29: Comparison between using the gap flow solver and using the local refinement grid at grid resolution $n = 90$ for the example of two interlocking gears. Typical numbers are shown, and variation of the numbers throughout the simulations is small.



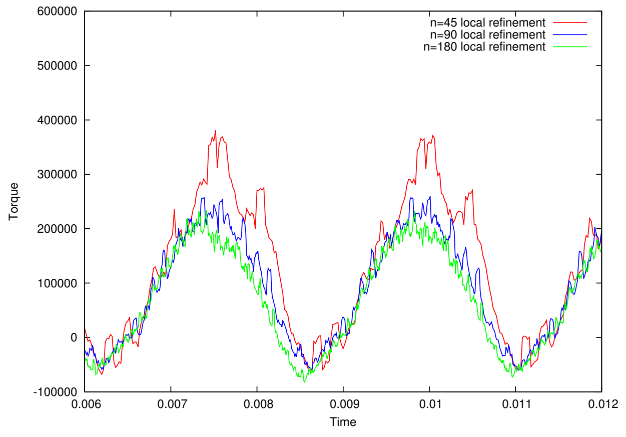
(a) Torque on the gear



(b) Torque on the pinion

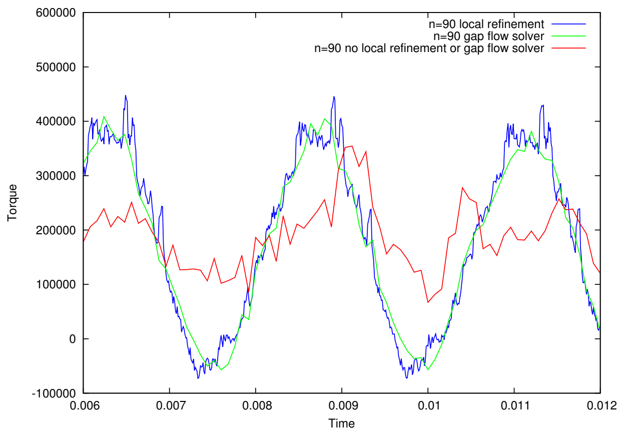


(c) Torque on the gear (denoised)

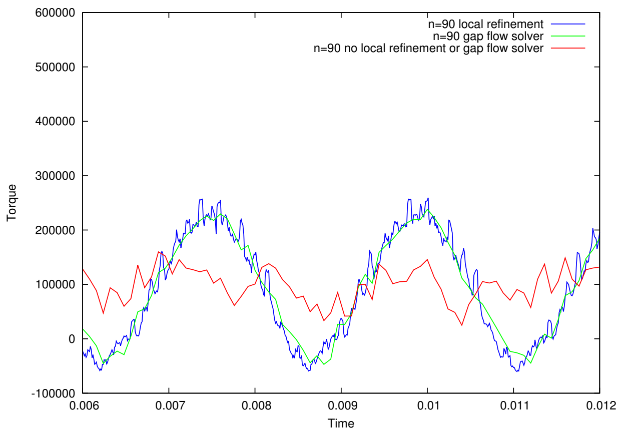


(d) Torque on the pinion (denoised)

Figure 27: The torques exerted on the two gears due to the fluid pressure. The denoised results in (c) and (d) are obtained by applying a simple averaging with a five-point temporal stencil to the results in (a) and (b). A uniform time step size 2×10^{-5} is used for grid resolution $n = 45$ and is refined at the same rate as the grid, noting that a global time step size is used on all the overlapping grids of different resolutions in the same simulation.



(a) Torque on the gear (denoised)



(b) Torque on the pinion (denoised)

Figure 28: The torques exerted on the two gears due to the fluid pressure. The results have been denoised by applying a simple averaging with a five-point temporal stencil. A uniform time step of 1×10^{-5} is used for the simulation with the local refinement grid, while a uniform time step of 8×10^{-5} is used for the other two simulations without local refinement.

8. Conclusion and Future Work

We proposed a method that uses solid surface particles to correctly resolve fluid pressure forces inside thin lubricated gaps between rigid bodies which the grid alone fails to resolve, coupling together the fluid in the gap with both the fluid on the grid and the rigid bodies in order to provide a monolithic and symmetric positive-definite system. Leaving thin rigid bodies and deformable solids as future work, the current method rasterizes the boundary of each rigid body into a closed hull. When the gap between two rigid bodies is under-resolved by the grid, pressure forces are missing on grid faces that lie between two grid cells that are rasterized into two different rigid bodies. To compute pressure forces on such grid faces, we treat each of these grid faces as two virtual solid-fluid faces and create a virtual fluid cell between the two virtual faces. To compute fluid pressure values for these virtual fluid cells, we added fluid pressure degrees of freedom to solid surface particles in the gap region and designed an interpolation operator that interpolates from the solid surface particles to the virtual fluid cells. Finally, the fluid pressure forces are computed on the virtual faces using the pressures in virtual fluid cells in the same manner as for regular solid fluid coupling grid faces.

In order to allow the fluid to flow tangentially through the gap, tangential velocity degrees of freedom were also added. The tangential flow in the gap is two-way coupled to the flow on the grid by interpolating pressures from the grid to ghost pressure particles at the boundary of the gap region, and substituting the interpolation stencils into the symmetric positive-definite system. We advect the tangential velocity on the surface mesh particles using an extension of semi-Lagrangian advection to codimension-one advected quantities on a curved mesh that can spin and move.

Apart from extending the current method to thin shells, deformable bodies, compressible flows, etc., there are also some limitations that could be addressed in future work. For example, we often need to set a positive lower bound on the gap thickness in order for the two-way coupled system solver to converge in an acceptable number of iterations. Another issue is the difficulty in assigning a meaningful gap thickness to a velocity degree of freedom particle that is near the gap boundary (where the ray does not hit another rigid body – see Section 4.1). Using the grid cell size Δx as the gap thickness for these particles does not result in convergence, since the gap thickness varies with the grid resolution. To remedy this, we set gap thicknesses on these particles by extrapolating the gap thickness from nearby velocity degree of freedom particles where the thickness is well-defined. So far we have only considered slip boundary condition on the solid surfaces, and leave no-slip boundary condition and viscosity for future work. For extension to compressible flows, see the preliminary work in the appendix of [32] as well as [31].

9. Acknowledgements

Research supported in part by ONR N00014-13-1-0346, ONR N00014-09-1-0101, ONR N-00014-11-1-0027, ONR N00014-11-1-0707, ARL AHPCRC W911NF-07-0027, and the Intel Science and Technology Center for Visual Computing. Computing resources were provided in part by ONR N00014-05-1-0479. Y.Y. was supported in part by a Stanford Graduate Fellowship.

Bibliography

[1] D. Adalsteinsson and J.A. Sethian. The fast construction of extension velocities in level set methods. *J. Comput. Phys.*, 148(1):2–22, 1999.

[2] A.T. Barker and X.-C. Cai. Scalable parallel methods for monolithic coupling in fluid–structure interaction with application to blood flow modeling. *J. Comput. Phys.*, 229(3):642–659, 2010.

[3] P.T. Barton, B. Obadia, and D. Drikakis. A conservative level-set based method for compressible solid/fluid problems on fixed grids. *J. Comput. Phys.*, 230(21):7867–7890, 2011.

[4] P. Causin, J.-F. Gerbeau, and F. Nobile. Added-mass effect in the design of partitioned algorithms for fluid-structure problems. *Comp. Meth. Appl. Mech. Eng.*, 194(42-44):4506–4527, 2005.

[5] A.J. Chorin. A numerical method for solving incompressible viscous flow problems. *J. Comput. Phys.*, 2(1):12–26, 1967.

[6] R. Courant, E. Issacson, and M. Rees. On the solution of nonlinear hyperbolic differential equations by finite differences. *Comm. Pure and Applied Math*, 5(3):243–255, 1952.

[7] R.E. English, M. Lentine, and R. Fedkiw. Interpenetration free simulation of thin shell rigid bodies. *IEEE TVCG*, 19(6):991–1004, 2013.

[8] R.E. English, L. Qiu, Y. Yu, and R. Fedkiw. An adaptive discretization of incompressible flow using a multitude of moving Cartesian grids. *J. Comput. Phys.*, 254(0):107 – 154, 2013.

[9] E.A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *J. Comput. Phys.*, 161(1):35–60, 2000.

[10] C. Farhat, M. Lesoinne, and P. Le Tallec. Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity. *Comp. Meth. Appl. Mech. Eng.*, 157(1-2):95–114, 1998.

[11] C. Farhat, K. G. van der Zee, and P. Geuzaine. Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity. *Comp. Meth. Appl. Mech. Eng.*, 195(17-18):1973–2001, 2006.

[12] F. Gibou and C. Min. Efficient symmetric positive definite second-order accurate monolithic solver for fluid/solid interactions. *J. Comput. Phys.*, 231(8):3246–3263, 2012.

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

[14] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, 22(3):871–878, 2003.

[15] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 24(3):973–981, 2005.

[16] F. Harlow and J. Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Phys. Fluids*, 8(12):2182–2189, 1965.

[17] D. Hartmann, M. Meinke, and W. Schröder. A cartesian cut-cell solver for compressible flows. In *Computational Science and High Performance Computing IV*, volume 115 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pages 363–376. Springer Berlin Heidelberg, 2011.

[18] D. Hartmann, M. Meinke, and W. Schröder. A strictly conservative cartesian cut-cell method for compressible viscous flows on adaptive grids. *Comp. Meth. Appl. Mech. Eng.*, 200(9-12):1038–1052, 2011.

[19] C. Hirt, A. Amsden, and J. Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *J. Comput. Phys.*, 14(3):227–253, 1974.

[20] B. Hübner, E. Walhorn, and D. Dinkler. A monolithic approach to fluid–structure interaction using space–time finite elements. *Comp. Meth. Appl. Mech. Eng.*, 193(23-26):2087–2104, 2004.

[21] L. Kobbelt. $\sqrt{3}$-subdivision. In *Proc. of ACM SIGGRAPH*, pages 103–112, 2000.

[22] M. Lentine, J.T. Grétarsson, and R. Fedkiw. An unconditionally stable fully conservative semi-lagrangian method. *J. Comput. Phys.*, 230(8):2857–2879, April 2011.

[23] R.J. LeVeque and K.-M. Shyue. Two-dimensional front tracking based on high resolution wave propagation methods. *J. Comput. Phys.*, 123(2):354–368, 1996.

[24] W. Liu, L. Yuan, and C.-W. Shu. A conservative modification to the ghost fluid method for compressible multiphase flows. *Commun. Comput. Phys.*, 10(4):785–806, 2011.

[25] C. Michler, S.J. Hulshoff, E.H. van Brummelen, and R. de Borst. A monolithic approach to fluid–structure interaction. *Computers & Fluids*, 33(5-6):839–848, 2004. Applied Mathematics for Industrial Flow Problems.

[26] R. Mittal, H. Dong, M. Bozkurttas, F.M. Najjar, A. Vargas, and A. von Loebbecke. A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *J. Comput. Phys.*, 227(10):4825–4852, 2008.

[27] D. Nordsletten, D. Kay, and N. Smith. A non-conforming monolithic finite element method for problems of coupled mechanics. *J. Comput. Phys.*, 229(20):7571–7593, 2010.

[28] C. Peskin. Flow patterns around heart valves: A numerical method. *J. Comput. Phys.*, 10(2):252–271, 1972.

[29] C. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[30] S. Piperno, C. Farhat, and B. Larrouturou. Partitioned procedures for the transient solution of coupled aeroelastic problems part I: Model problem, theory and two-dimensional application. *Comp. Meth. Appl. Mech. Eng.*, 124(1-2):79–112, 1995.

[31] L. Qiu. *An adaptive discretization for incompressible and compressible flow using a multitude of moving Cartesian grids with gap flow treatment.* PhD thesis, Stanford University, June 2015.

[32] L. Qiu, W. Lu, and R. Fedkiw. An adaptive discretization of compressible flow using a multitude of moving Cartesian grids. *(In Preparation)*, 2014.

[33] A. Robinson-Mosher, C. Schroeder, and R. Fedkiw. A symmetric positive definite formulation for monolithic fluid structure interaction. *J. Comput. Phys.*, 230(4):1547–1566, 2011.

[34] A. Robinson-Mosher, T. Shinar, J. T. Grétarsson, J. Su, and R. Fedkiw. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 27(3):46:1–46:9, August 2008.

[35] H.A. Schwarz. *Gesammelte mathematische abhandlungen.* J. Springer, 1890.

[36] J. H. Seo and R. Mittal. A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations. *J. Comput. Phys.*, 230(19):7347–7363, 2011.

[37] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw. Hybrid simulation of deformable solids. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 81–90, 2007.

[38] H.S. Udaykumar, H.-C. Kan, W. Shyy, and R. Tran-Son-Tay. Multiphase dynamics in arbitrary geometries on fixed cartesian grids. *J. Comput. Phys.*, 137(2):366–405, 1997.

[39] D.G. Wells. *The Penguin dictionary of curious and interesting geometry.* Penguin Mathematics Series. Penguin Books, 1991.

[40] Miad Yazdani and Marios C. Soteriou. A novel approach for modeling the multiscale thermo-fluids of geared systems. *International Journal of Heat and Mass Transfer*, 72(0):517 – 530, 2014.

[41] Miad Yazdani, Marios C. Soteriou, Fanping Sun, and Zaffir Chaudhry. Prediction of the thermo-fluids of gearbox systems. *International Journal of Heat and Mass Transfer*, 81(0):337 – 346, 2015.

[42] H. Zhang, X. Zhang, S. Ji, Y. Guo, G. Ledezma, N. Elabbasi, and H. deCougny. Recent development of fluid-structure interaction capabilities in the ADINA system. *Comput. and Struct.*, 81(8-11):1071–1085, 2003.

[43] W. Zheng, B. Zhu, B. Kim, and R. Fedkiw. A new incompressibility discretization for a hybrid particle MAC grid representation with surface tension. *J. Comput. Phys.*, 280:96–142, 2015.