Power Particles: An incompressible fluid solver based on power diagrams

Fernando de Goes, Caltech
Corentin Wallez, École Polytechnique
Jin Huang, Zhejiang U.
Dmitry Pavlov, Imperial College London
Mathieu Desbrun, Caltech

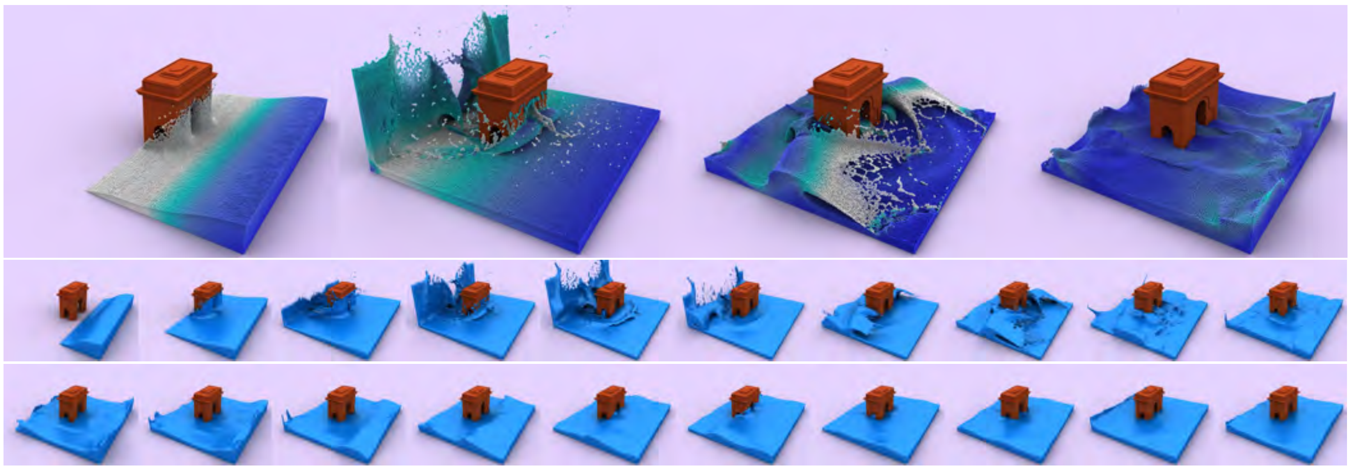


Figure 1: ***Splash de Triomphe.*** *A dam break near the Arc de Triomphe is simulated using 600k particles for the fluid and 80k for the solid obstacle. Bottom two rows show every sixth frame of the whole sequence, with surfacing achieved via OpenVDB [Museth 2013].*

Abstract

This paper introduces a new particle-based approach to incompressible fluid simulation. We depart from previous Lagrangian methods by considering fluid particles no longer purely as material points, but also as volumetric parcels that partition the fluid domain. The fluid motion is described as a time series of well-shaped power diagrams (hence the name *power particles*), offering evenly spaced particles and accurate pressure computations. As a result, we circumvent the typical excess damping arising from kernel-based evaluations of internal forces or density without having recourse to auxiliary Eulerian grids. The versatility of our solver is demonstrated by the simulation of multiphase flows and free surfaces.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.7 [Computer Graphics]: 3D Graphics and Realism—Animation.

Keywords: Lagrangian fluid simulation, power diagrams, incompressibility, multiphase flows, free surface.

1 Introduction

Fluid motion is a visually rich and complex phenomenon that remains, to this day, a challenge to reproduce numerically. Intricate patterns such as vortices in liquids and volutes in smoke are typical visual cues of incompressible flows that a numerical simulation needs to capture accurately. While Lagrangian methods have been shown simple and practical to efficiently generate fluid motion, they often suffer from numerical artifacts that severely impact liveliness of the flow. In particular, the manner in which (and the degree to which) incompressibility is enforced has strong implications for its dynamics. For instance, the Monte Carlo nature of kernel evaluations in Smoothed Particle Hydrodynamics (SPH) [Lucy 1977] can only control local particle density at the cost of significant motion damping [Ihmsen et al. 2014]. The addition of an auxiliary grid to provide more accurate and efficient pressure projections also causes kinetic energy dissipation and particle drifting as velocities are transferred from/to the grid [Harlow 1963; Brackbill et al. 1988]. Even moving mesh approaches struggle to enforce incompressibility due to mesh distortion and tangling [Oñate et al. 2004; Tang 2004; Clausen et al. 2013]. Consequently, none of the current Lagrangian approaches offers both strong incompressibility and low numerical dissipation.

We propose a radically different Lagrangian method to simulate incompressible fluid. Particles are considered as non-overlapping fluid parcels that partition the space occupied by the fluid through a moving power diagram. By leveraging computational tools on power diagrams, we formulate a time integrator for these "power" particles that precisely controls particle density and pressure forces, without kernel estimates or significant artificial viscosity.

1.1 Related Work

A broad variety of methods for fluid simulation have been devised in computational physics and computer graphics (see surveys in [Bridson 2008; Ihmsen et al. 2014]). Here we restrict our discussion to Lagrangian methods based on particles and moving meshes.

Early SPH work. Smoothed Particle Hydrodynamics (SPH) discretizes fluids as material points with interaction forces derived from smooth kernel functions [Monaghan 2005]. In computer graphics, this methodology was introduced for smoke and fire de-

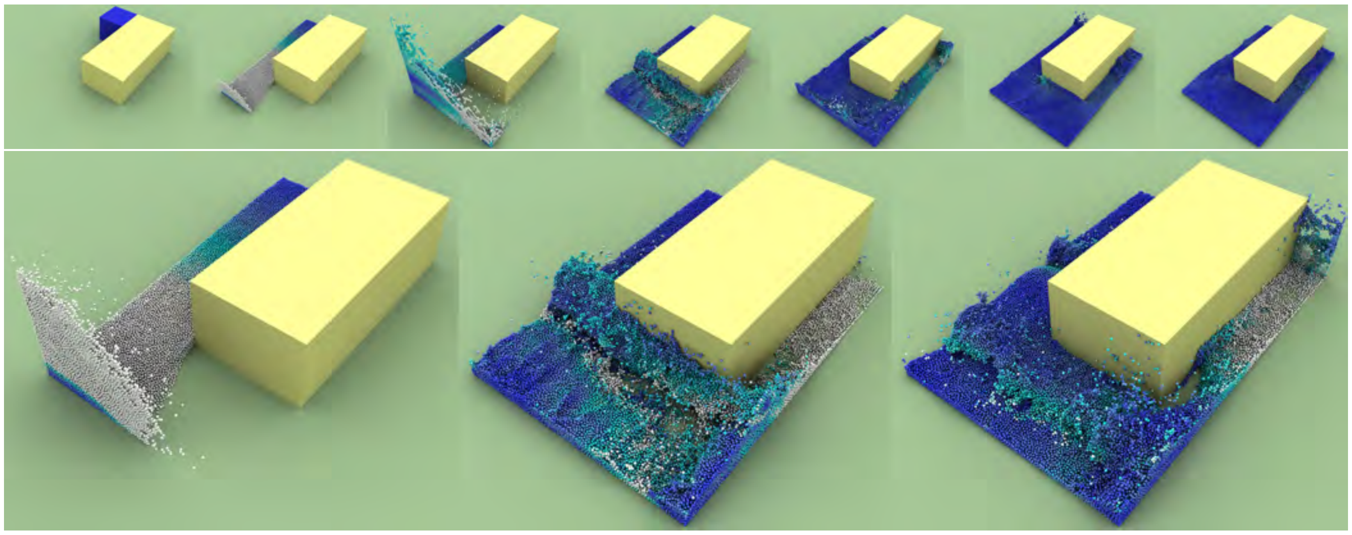


Figure 2: ***Wall-confined dam break.*** *A dam break confined to a U-shaped corridor is simulated with 65k particles. Top row shows every* 70 *frames of the sequence. The water first splashes against the opposite wall (bottom left), before ricocheting onto the inner wall (bottom center), then splashing at the end of the domain (bottom right).*

piction in [Stam and Fiume 1995] before being used for weakly compressible flows [Desbrun and Gascuel 1996; Müller et al. 2003; Becker and Teschner 2007]. These approaches were quickly extended to handle adaptive sampling [Adams et al. 2007], multiple fluids [Solenthaler and Pajarola 2008; Ren et al. 2014], viscous flows [Takahashi et al. 2015], shallow water simulation [Solenthaler et al. 2011], and parallel computations [Ihmsen et al. 2011], but remained largely limited to the compressible case.

Density-constrained SPH. A family of SPH methods incorporated incompressibility by controlling particle density throughout the simulation. The methods of [Ellero et al. 2007; Bodin et al. 2012], for instance, constrained densities via Lagrangian multipliers, while predictive-corrective incompressible SPH (PCISPH) [Solenthaler and Pajarola 2009] used an iterative Jacobi-style algorithm that accumulates pressure changes to correct particle location. The convergence of PCISPH was accelerated in [He et al. 2012], and Macklin and Müller [2013] further relaxed time step requirements by incorporating pressure estimates into the Position-based Dynamics framework. Although these methods retain evenly spaced particle distributions over time, the iterative computations of pressure forces induced by intermediate density fluctuations introduce severe numerical damping [Ihmsen et al. 2014], which can be mitigated via vorticity confinement [Steinhoff and Underhill 1994].

Divergence-free SPH. Other techniques address fluid incompressibility by moving particles along a divergence-free velocity field. Cummins and Rudman [1999] proposed a splitting scheme that projects intermediate velocities to a divergence-free field. In [Shao and Lo 2003], the projection step was approximated by estimating density compressions from intermediate velocities. The work of [Hu and Adams 2007; Kang and Sagong 2014], instead, computed pressure forces based on both divergence and density corrections. However, as noted in [Premože et al. 2003], displacing particles along divergence-free velocities for a *finite* time step irremediably leads to spurious particle clumping and interface tension effects, which impacts simulation stability and accuracy. These issues are commonly alleviated through a modified pressure solve [Ihmsen et al. 2013], the addition of non-physical anti-clustering forces [Alduan and Otaduy 2011; Akinci et al. 2013], artificial viscosity [Monaghan 2000; Clavet et al. 2005], or particle smoothing [Shadloo et al. 2011]—again, at the cost of extra numerical damping.

Hybrid schemes. Coupling non-diffusive Lagrangian advection with accurate Eulerian pressure projection on an auxiliary grid can partially remedy the inherent issues of SPH methods. In [Raveendran et al. 2011], SPH particles were displaced along a divergence-free velocity field computed on a coarse Eulerian grid. Losasso et al. [2008] proposed a two-way coupled SPH and particle level set method to simultaneously capture small and large scale phenomena. In [Zheng et al. 2015], a hybrid pressure projection was proposed with particles near the liquid interface and an Eulerian grid for the interior of the liquid domain. The fluid-implicit particle method (FLIP) [Brackbill et al. 1988] was adopted in [Zhu and Bridson 2005] to simulate sand, and later extended to viscous materials [Batty and Bridson 2008] and two-phase flows [Boyd and Bridson 2012]. The work of [Ando et al. 2013] further combined FLIP with adaptive sampling of particles, while Cornelis et al. [2014] enriched the SPH methodology with FLIP velocities.

Moving mesh approaches. While a vast majority of animation techniques involve particles and kernel evaluations, fluid solvers based on moving meshes have also been proposed. Feldman et al. [2005] and Klingner et al. [2006] used mesh optimization in order to better resolve turbulent regions with semi-Lagrangian advection. Arbitrary Lagrangian Eulerian (ALE) methods [Hietel et al. 2000] mix moderate mesh optimization with Eulerian-based finite volume integration, but are mostly restricted to compressible fluids. A finite element discretization for fluids was presented in [Oñate et al. 2004], requiring global restructuring of the underlying triangulation every simulation time step. The work of [Erleben et al. 2011; Misztal et al. 2013] proposed to tessellate the entire ambient space, forcing the mesh to conform to the fluid interface in time via local remeshing operations. In [Clausen et al. 2013], a fluid solver on tetrahedral meshes was introduced by ensuring incompressibility per vertex stencil [Irving et al. 2007], combined with local connectivity updates. While these methods are partially Lagrangian and leverage accurate pressure projection on meshes, they require a large amount of mesh updates, making them less practical and robust than particle methods.

Voronoi-based methods. Our work is closely related to Voronoi diagrams, a common tool in Eulerian integrators [Mullen et al. 2009; Brochu et al. 2010]. Their use has been proposed in Lagrangian methods as well: moving Voronoi cells combined with Eulerian remapping were introduced in [Whitehurst 1995] for gas dynamics, while a Voronoi-based finite volume pressure solver was proposed in [Sin et al. 2009]. Serrano et al. [2005] derived first-order accurate discrete divergence and gradient operators acting on moving Voronoi cells to conserve both linear and angular momenta. The work of [Springel 2010] leveraged these Voronoi-based discrete operators to design an ALE simulation for cosmological systems. Regularization steps based on Lloyd iterations [Lloyd 1982] and volume corrections are nevertheless required to ensure stability, limiting this approach to weakly compressible flows. Instead, we show that the use of power diagrams results in a purely Lagrangian method with precise local density control suitable for incompressible, free surface flows.

1.2 Contributions

We propose a new Lagrangian method for fluid simulation based on power diagrams—a generalization of Voronoi diagrams that incorporates extra degrees of freedom to partition the domain occupied by the fluid. The use of power diagrams results in several numerical benefits. We can now discretize fluids as volumetric parcels formed by cells of a power diagram that determine, at each time step, a well-shaped mesh from which accurate discrete differential operators are derived. The additional variables afforded by a power diagram also offer complete control over the volume of each fluid cell, resulting in evenly spaced particles over time. The enforcement of incompressibility is thus achieved both geometrically (through precise density control) and dynamically (via accurate projection onto divergence-free velocities). Since our method remains purely Lagrangian, it does not suffer from significant numerical diffusion or mass fluctuations as commonly seen in Eulerian methods. Finally, it is entirely devoid of error-prone, kernel-based estimation of pressure, force, or density. We demonstrate the effectiveness of power particles in scenarios with free surfaces and moving obstacles, and provide extensions to compressible and multiphase flows.

2 Computational Foundations

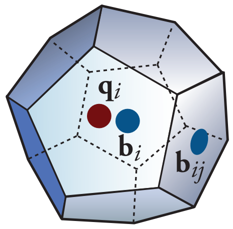
Before presenting our approach to fluid simulation, we first describe core concepts upon which our fluid discretization is based. We denote the fluid domain by Ω, with volume $|\Omega|$ and boundary $\partial\Omega$.

2.1 Power Diagrams

A power diagram [Aurenhammer 1987] is a partitioning of Ω into a cell complex defined by a list of n points $\{\mathbf{q}_i\}_i$, called *sites*, and their associated scalar values $\{w_i\}$, called *weights*. For each weighted point $(\mathbf{q}_i, w_i)$, its power cell $\mathcal{V}_i$ is defined as

$$\mathcal{V}_i = \{\mathbf{x} \in \Omega \,|\, \|\mathbf{x} - \mathbf{q}_i\|^2 - w_i \leq \|\mathbf{x} - \mathbf{q}_j\|^2 - w_j \,\forall j\}, \quad (1)$$

where $\|\cdot\|$ indicates the Euclidean norm. Two sites $\mathbf{q}_i$ and $\mathbf{q}_j$ are said to be neighbors if the intersection $\mathcal{V}_i \bigcap \mathcal{V}_j$ is a non-empty power facet $\mathcal{A}_{ij}$ of codimension 1, corresponding to a planar polygon in 3D and a line segment in 2D. We indicate by $\mathcal{N}_i$ the set of neighboring sites sharing a power facet with site i, and use l_{ij} to indicate the distance $\|\mathbf{q}_i - \mathbf{q}_j\|$. We also denote d_{ij} as the distance from $\mathbf{q}_i$ to $\mathcal{A}_{ij}$ so that $d_{ij} + d_{ji} = l_{ij}$. The centroid (i.e., center of mass) of cell $\mathcal{V}_i$ is denoted as $\mathbf{b}_i$, while the centroid of power facet $\mathcal{A}_{ij}$ is denoted as $\mathbf{b}_{ij}$. Finally, we refer to V_i as the volume in 3D (resp., the area in 2D) of $\mathcal{V}_i$, and A_{ij} as the area in 3D (resp., the length in 2D) of $\mathcal{A}_{ij}$. Note that when



weights are all equal, power diagrams reduce to Voronoi diagrams. As for the Voronoi-Delaunay pair, a power diagram also defines by duality a triangulation of the sites (known as the weighted Delaunay triangulation), in which each neighboring pair of sites forms an edge that is orthogonal to its associated power facet. Furthermore, the weighted circumcenters of the tetrahedra of this triangulation in 3D (resp., triangles in 2D) correspond to dual vertices of the power diagram, i.e., they are at the intersection of four power cells in 3D (resp., three in 2D). Although power diagrams have found many applications in geometry processing [Mullen et al. 2011; de Goes et al. 2013; Liu et al. 2013], their use in simulation has been restricted to the formation of bubbles and foams [Busaryev et al. 2012].

Robust construction of power diagrams is best left to one of the several existing libraries [Rycroft 2009; CGAL 2015] which return, from a series of sites and weights, a geometric description of the power cells and their power facets, as well as iterators encoding cell connectivity. Power cells that intersect the boundary $\partial\Omega$ can be further clipped either using the Sutherland-Hodgman algorithm [Yan et al. 2013], or based on mirrored copies of the subset of sites with dual facets straddling $\partial\Omega$. If the domain Ω is periodic, a similar mirroring procedure can be used too [Yan et al. 2011]. The resulting clipped cells form finite (and possibly non-convex) polytopes, from which one can compute all relevant geometric quantities (centroids $\mathbf{b}_i$ and $\mathbf{b}_{ij}$, volumes V_i, areas A_{ij}, and lengths l_{ij}). While the worst-case computational cost to construct power diagrams in 3D is quadratic in the number of sites, the average-case scenario is linear in the number of sites (see, e.g., [Golin and Na 2003]).

2.2 Volume Constraints

Besides defining connectivity, the weights of a power diagram offer full control over cell volumes. Indeed, Aurenhammer et al. [1998] proved that for a set of sites $\{\mathbf{q}_i\}_i$ and a target volume $\overline{V}_i$ for each of them, a power diagram satisfying $V_i = \overline{V}_i$ can be uniquely constructed by solving for weights that maximize a concave energy

$$\mathcal{E}(\{w_i\}_i) = \sum_i \int_{\mathcal{V}_i} \|\mathbf{x} - \mathbf{q}_i\|^2 d\mathbf{x} - \sum_i w_i \left(V_i - \overline{V}_i\right). \quad (2)$$

It is worth pointing out that the energy $\mathcal{E}$ is smooth, even through local connectivity changes; this implies that even if the weighted Delaunay triangulation changes discretely over time through flips, its dual power diagram is evolving continuously. It was recently shown that this weight optimization is equivalent to an optimal transport problem with volume constraints, and that the Hessian of $\mathcal{E}$ has a simple, closed-form expression [de Goes et al. 2012]. Optimal weights for a given set of cell volumes are thus efficiently found by iteratively updating them through Newton's steps of the form

$$\tfrac{1}{2}\mathbf{\Delta}\ \delta w = V - \overline{V}, \quad (3)$$

where $\mathbf{\Delta}$ is the usual finite-volume Laplacian matrix built from the facets of the power diagram, with entries $\mathbf{\Delta}_{ij} = A_{ij}/l_{ij}$. Notice that, as power facets always have positive area (resp., length in 2D), the Laplacian is negative semi-definite, with linear functions in its kernel. We will make use of this weight optimization in §4 as a means to preserve local volumes (and thereby densities, since each particle will have a fixed mass) in our fluid solver.

2.3 Discrete Operators

Another consequence of using power diagrams is that it naturally leads to discrete operators on scalar and vector fields stored on power cells that mimic their continuous counterparts, which will be crucial in enforcing divergence-free velocity fields. Since we have a well defined notion of volume per particle (namely, V_i), the notion of divergence can be measured by how the local volume is affected by the motion of the sites with fixed weights. We thus define the discrete divergence operator $\mathbf{D}$ as the volume rate of change induced by displacing sites, i.e., $\mathbf{D} = \nabla_{\mathbf{q}} V$. For n particles in a d-dimensional domain Ω, this operator corresponds to an nxn matrix with row-valued entries:

$$\begin{cases} \mathbf{D}_{ij} := \left(\nabla_{\mathbf{q}_j} V_i\right)^t = \dfrac{A_{ij}}{l_{ij}} \left(\mathbf{q}_j - \mathbf{b}_{ij}\right)^t, \\ \mathbf{D}_{ii} := \left(\nabla_{\mathbf{q}_i} V_i\right)^t = - \displaystyle\sum_{j \in \mathcal{N}_i} \left(\nabla_{\mathbf{q}_i} V_j\right)^t. \end{cases} \tag{4}$$

The corresponding gradient is obtained through Stokes' theorem:

$$\int_\Omega \nabla p \cdot \mathbf{v}\, d\mathbf{x} + \int_\Omega p\, \nabla \cdot \mathbf{v}\, d\mathbf{x} = \int_{\partial\Omega} p\, (\mathbf{v} \cdot \mathbf{n})\, d\mathbf{x}, \tag{5}$$

where p is a scalar function and $\mathbf{v}$ is a vector field. Assuming Neumann boundary conditions (we will discuss boundary conditions in more detail in §4), the gradient operator $\mathbf{G}$ is defined as the negated transpose of the divergence operator, i.e., $\mathbf{G} = -\mathbf{D}^t$. For a discrete function with values $\{p_i\}_i$, our gradient then reduces to:

$$[\mathbf{G}p]_i = \sum_{j \in \mathcal{N}_i} (\mathbf{D}_{ji})^t (p_i - p_j) = \sum_{j \in \mathcal{N}_i} \frac{A_{ij}}{l_{ij}} (\mathbf{q}_i - \mathbf{b}_{ij})\, (p_i - p_j).$$

It bears pointing out that this discrete gradient is symmetric (i.e., gradient forces from particle i to j cancel forces from j to i) and, as such, exactly preserves linear momentum. This is in sharp contrast to the SPH methodology, where the kernel-based gradient needs to be systematically symmetrized. Angular momentum is also preserved due to the invariance of the volume of power cells to global rotations. Furthermore, since this definition is directly derived from a consistent partition of the domain into local volumes exactly summing up to the total volume $|\Omega|$, we do not suffer the usual inaccuracies of kernel-based operators: our gradient operator is exact for linear functions, while the symmetrized gradient operator used in SPH methods fails to be accurate even on constant fields. We finally note that the approximation error of our operators on arbitrary functions are further minimized when sites $\{\mathbf{q}_i\}_i$ of the power diagram coincide with their respective centroids $\{\mathbf{b}_i\}_i$, that is, for *centroidal* power diagrams [de Goes et al. 2012]. We point the reader to the Appendix A for more details.

3 Power Particles

We now introduce our new computational method for incompressible fluid simulation based on power diagrams.

3.1 Rationale

Most Lagrangian approaches for incompressible fluid simulation use time-evolving material points that interact via pressure forces. Whether these forces are computed through error-prone kernel evaluations as in SPH methods or through an auxiliary grid as in FLIP schemes, numerical dissipation and particle drifting tend to plague the visual liveliness of such particle methods [Brackbill et al. 1988; Cornelis et al. 2014]. Moving meshes and ALE methods provide an alternative by carrying an adaptive mesh along with the material points to combine non-diffusive Lagrangian advection and accurate Eulerian pressure estimates; however, connectivity updates required to resolve mesh entanglement introduce resampling error, often in the form of numerical viscosity [Clausen et al. 2013].

Instead, we discretize fluid particles as volumetric parcels described by cells of a time-evolving power diagram—hence, the name *power particles*. Our approach adds to the usual set of sites $\{\mathbf{q}_i\}_i$ used in material point discretizations a new set of dynamical variables represented by time varying weights $\{w_i\}_i$. The power diagram determined by both sites and weights then defines the geometry of the fluid parcels. Thus, these power particles combine the simplicity of Lagrangian methods with the accuracy of mesh-based approaches. In particular, we can leverage the computational tools presented in §2 to deform power diagrams based on the dynamics of incompressible fluids, while retaining well-shaped, volume-constrained cells.

Starting from n cells of volume $\{\overline{V}_i\}_i$ and fixed mass $\{m_i\}_i$, the motion of power particles is updated at each time step in two stages: a *velocity update* followed by a *cell advection*. The former is based on a linear-accurate projection method to enforce zero divergence of the velocity field $\{\mathbf{v}_i\}_i$, while the latter advects cells in time by updating the sites $\{\mathbf{q}_i\}_i$ based on the current cell centroids and velocities before optimizing the weights $\{w_i\}_i$ to determine spatial occupancy. Since the volume of each power particle is precisely controlled, so is the local density since the mass of each particle is fixed in time. As we will demonstrate, this computational approach can handle arbitrary large deformations while ensuring accurate pressure forces and evenly spaced particles.

3.2 Velocity update

The velocity update begins by doing an explicit time integration of the external forces followed (if needed) by an implicit integration of viscosity forces, leading to an intermediate velocity $\mathbf{v}_i^*$ for each particle i. To enforce incompressibility instantaneously, we then compute pressure forces that project velocities to a zero divergence field. We leverage the discrete operators defined in §2.3 and solve for pressure values $\{p_i\}_i$ by minimizing the change in kinetic energy necessary to reach a divergence-free vector field , i.e.:

$$\min_p \sum\nolimits_i m_i \|\mathbf{v}_i^* - \tfrac{dt}{m_i} [\mathbf{G}p]_i\|^2. \tag{6}$$

The discrete Poisson equation for pressure is thus:

$$dt\, \mathbf{L}p = \mathbf{D}\mathbf{v}^*, \tag{7}$$

where $\mathbf{L} = \mathbf{D}\, \mathrm{diag}(m)^{-1} \mathbf{G}$ is the discrete Laplacian with $\mathrm{diag}(m)$ as the diagonal matrix containing every particle mass. The velocity field $\mathbf{v}$ is then updated by subtracting $dt\, \mathrm{diag}(m)^{-1} \mathbf{G}p$ from $\mathbf{v}^*$, rendering the final velocity divergence-free. Note that our Laplace operator $\mathbf{L}$ is, by construction, negative semi-definite with linear functions in its kernel. We also point out that $\mathbf{L}$ has a two-away stencil, in contrast to the finite-volume Laplacian $\boldsymbol{\Delta}$ introduced in §2.2 with a one-ring stencil that acts on weights. Both operators approximate the same continuous operator, but play very different roles in our discrete setting.

3.3 Cell advection

After updating velocities, we now need to advance in time the volumetric fluid parcels represented by power particles. Advection is achieved by computing new sites $\{\mathbf{q}_i\}_i$ and weights $\{w_i\}_i$ to evolve the power diagram while controlling the spacing and volumes of the new cells. To this end, we first explicitly update the

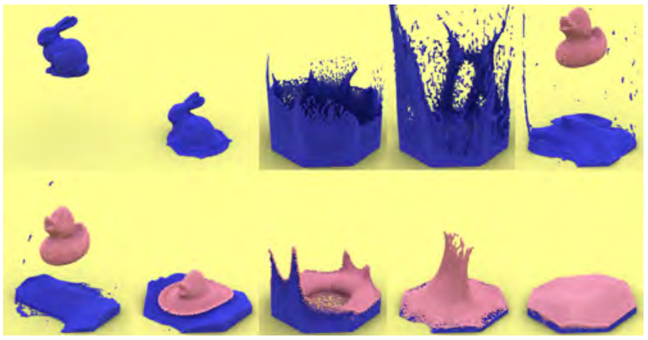


Figure 3: ***Two-phase flow.*** *A heavy (blue) fluid is dropped in an octogonal container (130k particles); a (pink) fluid which is four times lighter is then dropped on top (70k particles).*

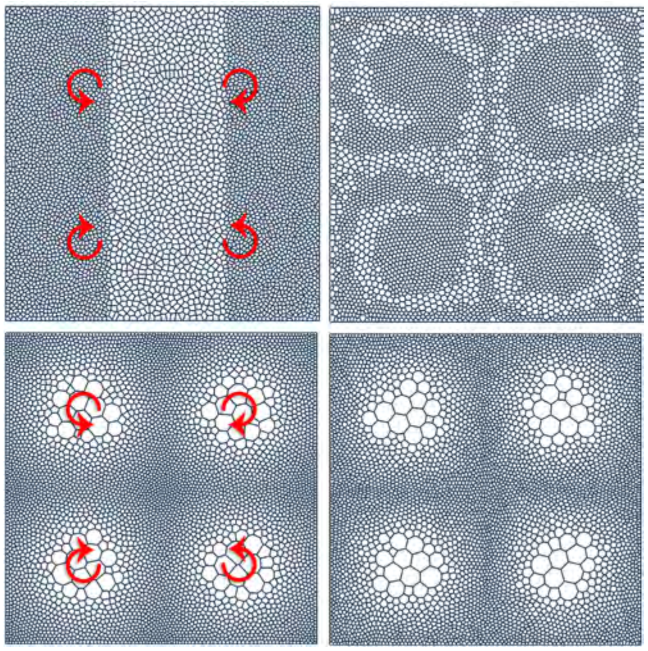


Figure 4: ***Adaptive sampling.*** *Whether initialized with variable volume sizes (top left) or with a velocity-adapted distribution (bottom left), a four-vortex example is properly captured (right), maintaining the flow symmetry over long periods of time.*

sites $\{\mathbf{q}_i\}_i$ to the positions of their respective centroids $\{\mathbf{b}_i\}_i$ once advected along velocities $\{\mathbf{v}_i\}_i$, i.e., $\mathbf{q}_i = \mathbf{b}_i + dt\,\mathbf{v}_i$. With these new sites, we then make use of the concave optimization in §2.2 and solve for new weights $\{w_i\}_i$ so that the volume V_i of each cell equates its target volume $\overline{V}_i$. Note that if the target volumes remain constant in time, then this enforcement of local volumes implies exact preservation of fluid densities (hence, incompressibility) since each particle mass is constant. Target volumes can also be determined via a state equation in the case of compressible fluids (see §4). Also, observe that the use of centroids favors evenly spaced distributions, and thus improves the accuracy of our discrete operators as mentioned in §2.3. Moreover, the weight update controls the relative sizing among particles, reducing artificial particle smoothing and velocity damping that are ubiquitous in existing Lagrangian methods. Finally, we point out that updating sites and weights also adapts the connectivity of the power diagram directly and continuously, eliminating the risk of mesh entanglement that moving mesh methods typically face.

3.4 Discussion

While our discretization of fluid flows contains only one additional variable per particle compared to typical Lagrangian methods, the consequences of this added degree of freedom are manifold. First, it induces a tiling of the fluid as each particle is now associated with a spatial cell of controllable volume. Second, the geometric definition of these cells allows us to derive consistent discrete differential operators (divergence, gradient, and Laplacian) based on the analytical expression of their volumes. Note that these operators are, in fact, extensions of the ones derived for Voronoi diagrams in [Serrano et al. 2005], and thus share their properties of being linear accurate and matching to first order their Eulerian counterparts. The accuracy of these operators combined with our pressure projection thus leads to a time integrator with low numerical dissipation. Furthermore, our Lagrangian cell advection via centroids maintains well-centered power cells in time; as power particles come to rest, they actually form a blue noise distribution [de Goes et al. 2012]. As we will demonstrate in §5, these benefits keep at bay the usual issues of density fluctuations, spurious pressure modes, and excessive artificial viscosity. It also entirely sidesteps the problem of mesh tangling in moving mesh methods (e.g., [Misztal et al. 2013; Clausen et al. 2013]). Finally, Voronoi-based approaches such as [Springel 2010] simply cannot control the volume of their cells and thus require mesh regularization to bound volume drift, while our weights determine volume (and thus density) exactly.

4 Algorithm

Next we delve into the algorithmic details of our fluid solver based on power particles. We first describe incompressible power particles, and then show that simple modifications turn our approach into a compressible and multiphase fluid solver. We also discuss extensions such as free surfaces, surface tension, moving obstacles, and viscosity. An outline of the simulation loop is given in Algorithm 1.

Algorithm 1 Simulation Loop

```
1: Update velocity $\mathbf{v}^*$ via Eq. (8)
2: Compute pressure $p$ via Poisson Eq. (7), or state Eq. (9)
3: Add pressure forces: $\mathbf{v} \leftarrow \mathbf{v}^* - dt\ \mathrm{diag}(m)^{-1}\mathbf{G}p$
4: For compressible fluids, update volumes $\{\overline{V}_i\}_i$ via Eq. (10)
5: Advect sites: $\mathbf{q} \leftarrow \mathbf{b} + dt\,\mathbf{v}$
6: Enforce volumes $\{\overline{V}_i\}_i$ through weight optimization (§2.2)
```

Initialization. We begin by instantiating a set of n power particles in the fluid domain Ω with prescribed cell volumes $\{\overline{V}_i\}_i$. This partitioning is initialized by first seeding sites $\{\mathbf{q}_i\}_i$ on a grid inside Ω, and then alternating Lloyd iterations and weight optimizations to form a centroidal power diagram [de Goes et al. 2012]. The resulting power particles are each given a density ρ_i and a mass m_i such that $m_i = \rho_i \overline{V}_i$, in addition to an initial velocity. The target volumes can be set to uniform values $\{\overline{V}_i \equiv |\Omega|/n\}_i$, or initialized with an arbitrary density function. In the latter case, particles masses $\{m_i\}_i$ are no longer uniform in order to define a constant local density throughout the fluid. Fig. 4 shows an example of an incompressible flow initialized with different sampling patterns. One can also adapt the number of particles dynamically during the animation similar to [Ando et al. 2013], but we leave the design of a fully adaptive strategy for power particles as future work.

External and viscosity forces. Our simulation loop starts by updating the velocity of each power particle through an explicit time integration based on external forces, such as gravity or buoyancy. For viscous flows, we ensure robustness to large viscosity coefficients ν through an unconditionally stable implicit time integrator [Stam 1999]. We thus find an updated intermediate velocity field $\mathbf{v}^*$ by solving the following sparse linear system:

$$[\mathrm{diag}(V) - \nu dt\mathbf{L}]\,\mathbf{v}^* = \mathrm{diag}(V)\big(\mathbf{v} + dt\,\mathrm{diag}(m)^{-1}\mathbf{F}^{\text{ext}}\big), \quad (8)$$

where $\mathrm{diag}(V)$ and $\mathrm{diag}(m)$ are the diagonal matrices containing all local volumes and masses, respectively.

Incompressible flows. In the case of incompressible fluids, internal forces come directly from the gradient of pressure that enforces a divergence-free velocity field. We compute this pressure in Line 2 of Algorithm 1 by solving for the scalars $\{p_i\}_i$ that satisfy the discrete Poisson equation given in Eq. (7). In addition to controlling the velocity divergence, we also enforce incompressibility geometrically by keeping the volume of each cell constant. This means that Line 4 is rendered unnecessary for incompressible flows as each $\overline{V}_i$ remains fixed throughout the animation. The weights are finally computed as described in §2.2.

Compressible flows. The case of compressible fluids requires minor alterations. First, the pressure p in Line 2 is computed via a state equation. Our implementation uses the simple constitutive law [Desbrun and Gascuel 1996; Becker and Teschner 2007]:

$$p_i = \kappa\, m_i \left(1/V_i - 1/\overline{V}_i\right), \tag{9}$$

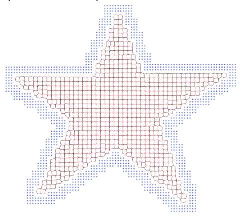
where κ is tantamount to a stiffness parameter. The velocity is then updated with the pressure forces in Line 3 which, unlike in the incompressible case, will no longer result in a divergence-free field. We then apply, in Line 4, the local volume change induced by the velocity divergence as an update of the target volumes $\{\overline{V}_i\}_i$, i.e.:

$$\overline{V}_i \leftarrow \overline{V}_i + dt\,[\mathbf{D}\mathbf{v}]_i\,. \tag{10}$$

Observe that the new volumes always sum up to the domain volume, $\sum_i \overline{V}_i = |\Omega|$, as a result of our definition of the divergence in §2.3 verifying the identity $\sum_i [\mathbf{D}\mathbf{v}]_i = 0$ (see Appendix A).

Multiphase flows. Power particles also accommodate the simulation of multiphase flows, i.e., when multiple fluid types interact within the same domain. For interacting fluids of the same kind but different densities, our method is applicable with no modification other than proper initialization of the densities (and, consequently, of the masses) per particle. For the interaction of incompressible and compressible fluids, Lines 2 and 4 of the algorithm are implemented in such a way that compressible particles are updated through a pressure from a state equation, while incompressible particles are adjusted via a pressure projection. We also use the pressure of the compressible particles along the fluids interface to set the Dirichlet boundary conditon for the pressure projection of the incompressible fluid in Eq (7).

Free surfaces. When fluid simulation involving a moving interface between fluid and air is desired, we represent free surfaces via air "ghost" particles [Schechter and Bridson 2012]. Before every simulation time step, we populate air ghost particles around the power particles in a band of thickness proportional to the smallest target volume $\overline{V}$ (see inset). We then use these ghost particles to clip the power cells of the fluid particles encroaching on this interface. We also exploit air particles to impose proper boundary conditions at the fluid-air power facets $\mathcal{A}_{ij}$. In the divergence computation (Eq. (4)), we equate the velocity of air particles to the fluid particle in order to enforce zero Neumann boundary condition. In the projection solve (Eq. (7)), we ensure that the Dirichlet boundary condition $p = 0$ is satisfied on every interface facet $\mathcal{A}_{ij}$ between a fluid particle i and an air ghost particle j. Similar to [Enright et al. 2005], we compute the pressure value p_j so that a linear interpolation along the edge ij reaches zero at $\mathcal{A}_{ij}$, which is achieved with $p_j = -(d_{ji}/d_{ij})p_i$. Finally, we set air particles to zero weights and remove them from the weight optimization (Eq. (3)), since their volumes are just temporary.



Obstacles. Static or moving obstacles can be discretized in our power particle framework either as clipping objects [Rycroft 2009] or via solid ghost particles [Schechter and Bridson 2012]. Clipping is more effective to represent simple obstacles, such as container walls in $\partial\Omega$ or convex objects (Fig. 2), while solid particles are more practical to describe arbitrary shapes (Fig. 1). In contrast to air particles, solid particles are instantiated only once in the initialization step. We maintain the volumes of these solid particles invariant in time by optimizing their weights in Eq. (3) along with all other fluid particles. In order to ensure Neumann boundary condition at a fluid-solid interface $\mathcal{A}_{ij}$, we add the boundary flux $(A_{ij}/l_{ij})(\mathbf{q}_j - \mathbf{q}_i)^t\mathbf{v}_j$ to the divergence of the fluid particle i in Eq. (4). For the pressure projection, we set zero Neumann boundary condition by removing the solid particles from Eq. (7). Lastly, particles in moving obstacles are also included in the cell advection (§3.3) to make them follow a prescribed path. This simple treatment allows proper transfer of momentum to the fluid.

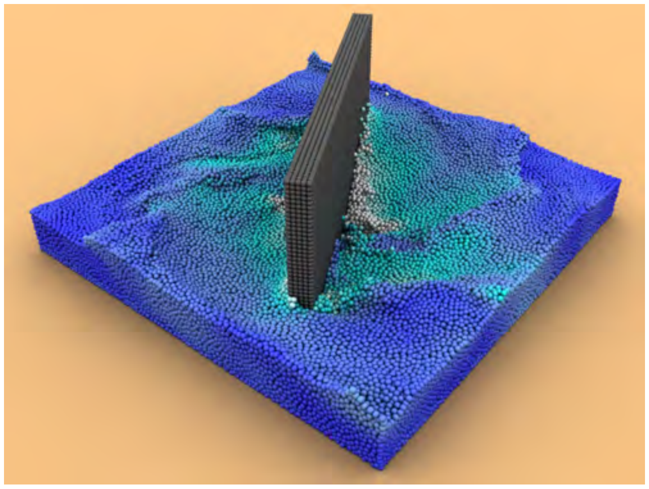


Figure 5: ***Moving obstacle.*** *A clockwise rotating blade continuously stirs a fluid made out of 100k particles. After transient splashes, a surface-wrinkling maelstrom develops in the container.*

Surface Tension. A fluid-air interface often exhibits surface tension due to a difference in cohesion forces on each side. It is usually modeled as an elastic membrane, resulting in forces minimizing the interface area. The use of power particles is particularly convenient as it can incorporate many of the existing strategies to approximate surface tension. Our implementation follows the approach of [Hong and Kim 2005], which adds the interface curvature as discontinuities of pressure in the projection step (Line 3). More specifically, we first compute an area-weighted normal $\mathbf{n}_i$ per particle i by summing area vectors of dual facets $\mathcal{A}_{ij}$ in the fluid-air interface, and deduce the mean curvature at interface particles as $\kappa = -\mathbf{D}(\mathbf{n}/\|\mathbf{n}\|)$. We then add the term $\tau(A_{ij}/l_{ij})\,(d_{ji}\kappa_i + d_{ij}\kappa_j)$ to the right-hand side of Eq. (7) to account for the integrated curvature at the interface $\mathcal{A}_{ij}$ scaled by the tension coefficient τ.

5 Results

We performed a series of tests in order to validate our method both in terms of performance and visual quality. A selection of these examples is included in the accompanying video. We describe our experiments next, and discuss results compared to previous work.

Implementation. Our implementation is based on our own thread-safe patch to the VORO++ library [Rycroft 2009] to construct clipped power diagrams in 2D and 3D. In contrast to [CGAL 2015] that tiles the entire Euclidean space before clipping cells to the domain Ω, VORO++ employs a local approach that calculates clipped power cell individually. We exploit this strategy to generate, in parallel, power cells for fluid and solid particles only—thus reducing the overhead of air particles and speeding up performance by two orders of magnitude compared to [Yan et al. 2013]. We use a preconditioned conjugate gradient solver for the sparse linear systems in Eqs. (3), (7) and (8), as suggested in [Bridson 2008]. We also adopt a substepping scheme so that every substep dt satisfies the CFL condition $dt < \max_i \sqrt[d]{V_i}/\max_i \|\mathbf{v}_i\|$ $(d = 2, 3)$, and the sum of substeps between two frames is $1/24$ s. In our experiments, there never were more than three substeps between two frames. We set the tolerance for volume enforcement in §2.2 to 0.1%, which typically requires a single Newton step every simulation substep. We

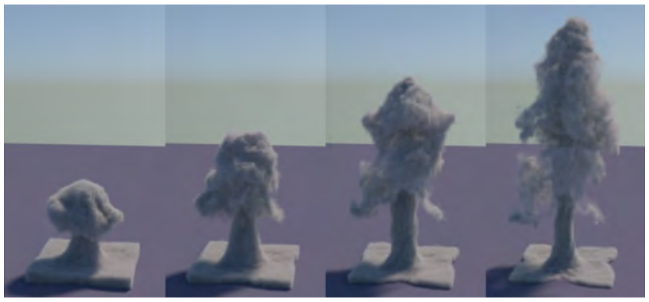


Figure 6: ***Smoke.*** *Fluid simulation with 20k particles in a box, with a heat source and buoyancy forces. For visualization, two millions markers were advected passively along the power particle velocity field, forming a rising smoke cloud.*

used the Mantra renderer in Houdini [Side Effects 2014] for the final rendering, and OpenVDB [Museth 2013] for surfacing in Fig. 1.

Examples in 2D. The supplemental video contains a series of 2D experiments showcasing various features of our power particle framework, including splashes with air particles and surface tension, liquid falling onto multiple obstacles, and a fluid of uniform density but varying local volumes. We also provide a simulation with and without weight optimization, demonstrating that the addition of weights keeps particle spacing, thus avoiding spurious drifting. Fig. 10 shows side by side comparisons for the animation of an incompressible flow with four vortices in a unit square, generated by our algorithm versus two variants of the FLIP method (code released by the authors of [Ando et al. 2013]) and our own implementation of the Position Based Fluids (PBF) [Macklin and Müller 2013]. We use these methods as representative candidates of divergence-free and density-constrained Lagrangian solvers, and point the reader to [Cornelis et al. 2014; Ihmsen et al. 2014] for further comparisons. We initialized all cases with the same distribution of 5k particles, and set a time step of $dt = 0.01$. Observe that PBF maintains an even distribution of particles at the cost of large numerical damping, even when viscosity is set to zero (Fig. 10, top-left). The FLIP method, on the other hand, exhibits less numerical dissipation, but clumps particles in between vortices (Fig. 10, top-right). As suggested in [Ando et al. 2013], we also interleaved FLIP steps with particle smoothing, which improves the particle distribution but still presents numerical viscosity (Fig. 10, middle-left). By contrast, our approach results in an artifact-free motion, keeping particles evenly spaced and preserving the vortical structures over time (Fig. 10, middle-right). Finally, the bottom row of Fig. 10 shows the kinetic energy in time for these three methods. Notice that our method has the smallest decay rate, and provides consistent results even with adaptive sampling (see video). We also tested the well-known case of a flow in a channel past a disk for a low viscosity fluid. We used 20k particles with non-slip boundary conditions around the disk, and periodic boundary conditions through the vertical walls as a means to implement inlet and outlet particles in the flow direction. Fig. 9 shows the expected vortex shedding (von Kármán vortex street) formed by the adherence of the fluid to the boundary, in agreement with physical experiments.

Examples in 3D. We also experimented with our approach to incompressible fluids in 3D. Fig. 6 shows a smoke simulation in a closed container filled with 20k power particles dragged by buoyancy forces and visualized by markers passively advected along the flow. For free surface examples, we visualized the velocity magnitude via pseudocolors ranging from dark blue (low) to white (high). Fig. 8 shows a splash simulation with 450k particles using two different coefficients of viscosity. We display in Fig. 1 snapshots of a 600k-particle animation after surfacing, where a fluid comes splashing onto the Arc de Triomphe before subsiding. Fig. 5 depicts a turbulent flow of 100k particles stirred by a spinning blade, while Fig. 2 shows a liquid motion along a U-shaped corridor with 65k particles. Notice that, even at a coarse particle resolution, our method can resolve detailed splashes and breaking waves with no significant visual dissipation. In addition to the examples listed above, the accompanying video also includes a double splash simulation with 1M particles.

Two-phase flows. Our method can simulate fluids of variable density as well. Empirically, we observed that our approach is robust up to a density ratio of 30. For easy comparison to existing multiphase Lagrangian approaches (see, e.g., [Hu and Adams 2007]), we reproduce the 2D Rayleigh-Taylor instability test described in [Cummins and Rudman 1999] in the supplemental video. Fig. 3 shows a 3D simulation with two incompressible fluids of different densities and local volumes falling into an octagonal container.

Compressible flows. We also demonstrate the flexibility of our approach to simulate compressible fluids. We tested our 2D compressible fluid model to simulate shallow water, in which densities represent particles heights as in [Lee and Han 2010]. Fig. 7 shows the smooth transition of power cell areas for a shallow water wave in a domain containing an obstacle.

Performance. Our experiments were clocked on an Intel Xeon 3.1GHz workstation with 32GB RAM and 12 cores. Examples in 2D using between 5k and 20k particles took less than 10 milliseconds per substep. The average time spent per substep in 3D was 1 second for 20k particles (Fig. 6), 2.7 seconds for 65k particles (Fig. 2), 7 seconds for 100k particles (Fig. 5), 17 seconds for 200k particles (Fig. 3), 28 seconds for 450k particles (Fig. 8), and 34 seconds for 600k particles (Fig. 1), indicating a linear asymptotic complexity w.r.t. the number of particles. For profiling purposes, we also broke down the timing of one substep for a 3D simulation with 1M particles (see video): it took a total of 62.5 seconds, with 54% for the power diagram construction, 16% for the projection step, 11.2% for the diffusion solve, 9.6% for seeding air particles, and 9.2% for the weight update. As noted in [Schechter and Bridson 2012], ghost particles create a memory overhead proportional to the surface area and particle size, varying from 30% up to 160% in Fig. 3. Overall, our method incurs an increase in computational complexity when compared to state-of-the-art particle methods [Ihmsen et al. 2014] due to the construction and update of the power diagram. However, the benefits of this extra step are numerous, varying from significantly reduced numerical damping (Fig. 10) to the ability to handle a large variety of fluids and their interactions with the environment. Consequently, our approach re-

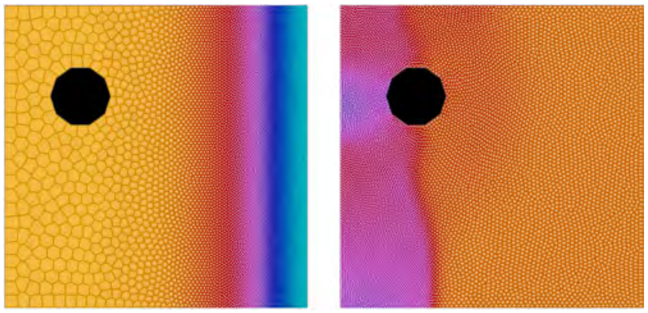


Figure 7: ***Shallow water.*** *Power particles can handle compressible flows as well. In this example, the shallow water equations are simulated in a square domain with a polygonal obstacle. Starting from a step-like height field (represented via color ramp), a wave comes crashing onto the obstacle and the left wall, creating a wide variety of particle sizes indicating various degrees of compression.*

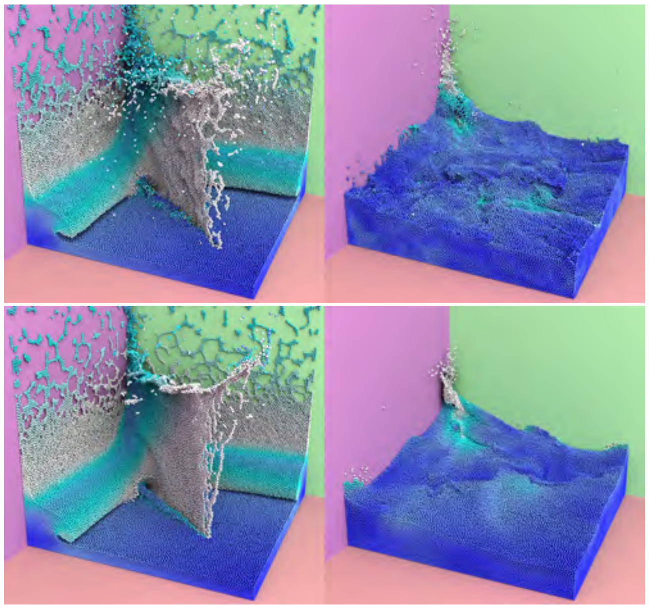


Figure 8: ***Viscosity control.*** *As our method does not suffer from significant numerical dissipation, the effect of kinematic viscosity is finely captured: the same 450k particles simulation with viscosity* $\nu = 0.001$ *(top) versus* $\nu = 0.01$ *(bottom) exhibits a striking difference in turbulence. Displayed are times* $1s$ *(left) and* $4s$ *(right).*

duces the wall-clock time to design a fluid animation sequence, as the results are consistent across time step and particle sizes, and no additional energy-injecting post-processing stages are required to obtain fine detail in the motion.

6 Conclusion and Future Work

By combining a simple Lagrangian discretization with robust computational geometry tools, power particles offer a stable and flexible approach for fluid simulation that reduces numerical damping and particle drift by affording accurate mesh-based pressure projection, precise control over particle distributions, and seamless mesh connectivity updates. As future work, we are interested in applying our power particle formulation to a broader family of physical models, including incompressible elasticity, elasto-plastic flows and granular materials. We are also investigating extensions to the VORO++ library to construct power diagrams on GPUs.

Acknowledgements.

We thank Chris Rydalch for his help with Houdini, Chris Rycroft for sharing the VORO++ library, Jiong Chen for rendering Fig. 6, and Santiago V. Lombeyda for help with figures and video. MD gratefully acknowledges the TITANE team and the INRIA International Chair program for their support. Partial funding was provided by NSF grant CCF-1011944 and a Google PhD Fellowship.

References

ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. 2007. Adaptively sampled particle fluids. *ACM Trans. Graph. 26*, 3.

AKINCI, N., AKINCI, G., AND TESCHNER, M. 2013. Versatile surface tension and adhesion for SPH fluids. *ACM Trans. Graph. 32*, 6.

ALDUAN, I., AND OTADUY, M. A. 2011. SPH granular flow with friction and cohesion. In *Symp. on Comp. Anim.*, 219–228.

ANDO, R., THÜREY, N., AND WOJTAN, C. 2013. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph. 32*, 4.

AURENHAMMER, F., HOFFMANN, F., AND ARONOV, B. 1998. Minkowski-type theorems and least-squares clustering. *Algorithmica 20*, 1, 61–76.

AURENHAMMER, F. 1987. Power diagrams: properties, algorithms and applications. *SIAM J. on Computing 16(1)*, 78–96.

BATTY, C., AND BRIDSON, R. 2008. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Symp. on Comp. Anim.*, 219–228.

BECKER, M., AND TESCHNER, M. 2007. Weakly compressible SPH for free surface flows. In *Symp. on Comp. Anim.*, 209–217.

BODIN, K., LACOURSIERE, C., AND SERVIN, M. 2012. Constraint Fluids. *IEEE Trans. Vis. Comput. Graphics 18*, 516–526.

BOYD, L., AND BRIDSON, R. 2012. MultiFLIP for energetic two-phase fluid simulation. *ACM Trans. Graph. 31*, 2.

BRACKBILL, J., KOTHE, D., AND RUPPEL, H. 1988. FLIP: A low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications 48*, 1, 25 – 38.

BRIDSON, R. 2008. *Fluid Simulation for Computer Graphics*. AK Peters/CRC Press.

BROCHU, T., BATTY, C., AND BRIDSON, R. 2010. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. 29*, 4.

BUSARYEV, O., DEY, T., WANG, H., AND REN, Z. 2012. Animating bubble interactions in a liquid foam. *ACM Trans. Graph. 31*, 4.

CGAL, 2015. Computational Geometry Algorithms Library (release 4.5). http://www.cgal.org.

CLAUSEN, P., WICKE, M., SHEWCHUK, J., AND O'BRIEN, J. 2013. Simulating liquids and solid-liquid interactions with lagrangian meshes. *ACM Trans. Graph. 32*, 2.

CLAVET, S., BEAUDOIN, P., AND POULIN, P. 2005. Particle-based viscoelastic fluid simulation. In *Symp. on Comp. Anim.*, 219–228.

CORNELIS, J., IHMSEN, M., PEER, A., AND TESCHNER, M. 2014. IISPH-FLIP for incompressible fluids. *Comp. Graph. Forum 33*, 2.

CUMMINS, S. J., AND RUDMAN, M. 1999. An SPH projection method. *J. Comp. Physics 152*, 2, 584–607.

DE GOES, F., BREEDEN, K., OSTROMOUKHOV, V., AND DESBRUN, M. 2012. Blue noise through optimal transport. *ACM Trans. Graph. 31*, 6.

DE GOES, F., ALLIEZ, P., OWHADI, H., AND DESBRUN, M. 2013. On the equilibrium of simplicial masonry structures. *ACM Trans. Graph. 32*, 4.

DESBRUN, M., AND GASCUEL, M.-P. 1996. Smoothed Particles: A new paradigm for animating highly deformable bodies. In *EG Computer Animation Symp.*, 61–76.

ELLERO, M., SERRANO, M., AND ESPAÑOL, P. 2007. Incompressible Smoothed Particle Hydrodynamics. *J. Comp. Physics 226*, 2, 1731–1752.

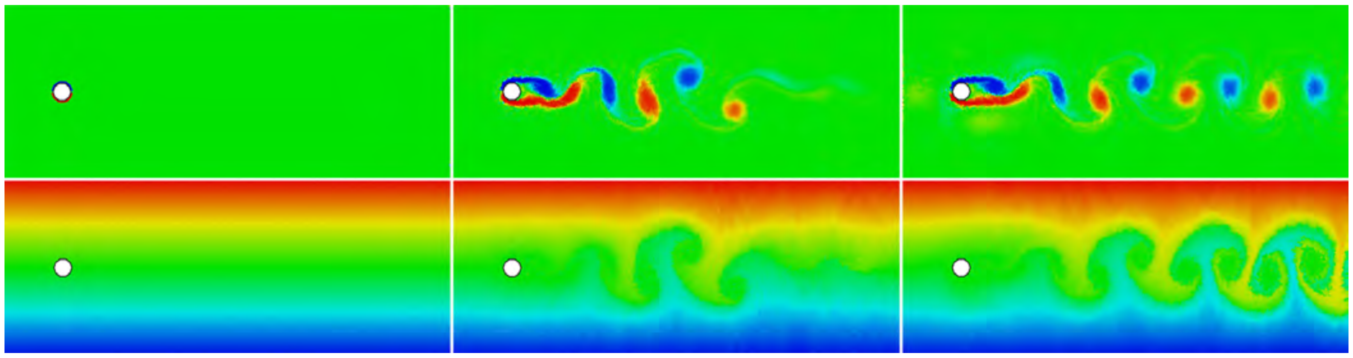


Figure 9: ***Flow past obstacle.*** *The wall-confined flow of an incompressible viscous fluid past a circular obstacle exhibits the expected von Kármán vortex street: the obstacle is "shedding" swirling vortices in its wake (top: vorticity plot; bottom: advected dye).*

ENRIGHT, D., LOSASSO, F., AND FEDKIW, R. 2005. A fast and accurate semi-Lagrangian particle level set method. *Comput. Struct. 83*, 6-7, 479–490.

ERLEBEN, K., MISZTAL, M. K., AND BÆRENTZEN, J. A. 2011. Mathematical foundation of the optimization-based fluid animation method. In *Symp. on Comp. Anim.*, 101–110.

FELDMAN, B. E., O'BRIEN, J. F., KLINGNER, B. M., AND GOKTEKIN, T. G. 2005. Fluids in deforming meshes. In *Symp. on Comp. Anim.*, 255–259.

GOLIN, M. J., AND NA, H. 2003. On the average complexity of 3D Voronoi diagrams of random points on convex polytopes. *Computational Geometry 25*, 3, 197–231.

HARLOW, F. H. 1963. The particle-in-cell method for numerical solution of problems in fluid dynamics. *Experimental arithmetic, high-speed computations and mathematics*.

HE, X., LIU, N., LI, S., WANG, H., AND WANG, G. 2012. Local Poisson SPH for viscous incompressible fluids. *Comp. Graph. Forum 31*, 6, 1948–1958.

HIETEL, D., STEINER, K., AND STRUCKMEIER, J. 2000. A finite-volume particle method for compressible flows. *Math. Models Methods Appl. Sci. 10*, 9, 1363–1382.

HONG, J.-M., AND KIM, C.-H. 2005. Discontinuous fluids. *ACM Trans. Graph. 24*, 3.

HU, X. Y., AND ADAMS, N. A. 2007. An incompressible multiphase SPH method. *J. Comp. Physics 227*, 1, 264–278.

IHMSEN, M., AKINCI, N., BECKER, M., AND TESCHNER, M. 2011. A parallel SPH implementation on multi-core CPUs. *Comp. Graph. Forum 30*, 1, 99–112.

IHMSEN, M., CORNELIS, J., SOLENTHALER, B., HORVATH, C., AND TESCHNER, M. 2013. Implicit Incompressible SPH. *IEEE Trans. Vis. Comput. Graphics 99*.

IHMSEN, M., ORTHMANN, J., SOLENTHALER, B., KOLB, A., AND TESCHNER, M. 2014. SPH fluids in computer graphics. *Eurographics STAR Report*.

IRVING, G., SCHROEDER, C., AND FEDKIW, R. 2007. Volume conserving finite element simulations of deformable models. *ACM Trans. Graph. 26*, 3.

KANG, N., AND SAGONG, D. 2014. Incompressible SPH using the divergence-free condition. *Comp. Graph. Forum 33*, 7, 219–228.

KLINGNER, B. M., FELDMAN, B. E., CHENTANEZ, N., AND O'BRIEN, J. F. 2006. Fluid animation with dynamic meshes. *ACM Trans. Graph. 25*, 3.

LEE, H., AND HAN, S. 2010. Solving the shallow water equations using 2D SPH particles for interactive applications. *The Visual Computer 26*, 6-8, 865–872.

LIU, Y., HAO, P., SNYDER, J., WANG, W., AND GUO, B. 2013. Computing self-supporting surfaces by regular triangulation. *ACM Trans. Graph. 32*, 4.

LLOYD, S. 1982. Least squares quantization in PCM. *IEEE Trans. Inf. Theory 28*, 2, 129–137.

LOSASSO, F., TALTON, J., KWATRA, N., AND FEDKIW, R. 2008. Two-way coupled SPH and particle level set fluid simulation. *IEEE Trans. Vis. Comput. Graphics 14*, 4, 797–804.

LUCY, L. B. 1977. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal 82*, 1013–1024.

MACKLIN, M., AND MÜLLER, M. 2013. Position based fluids. *ACM Trans. Graph. 32*, 4.

MISZTAL, M., ERLEBEN, K., BARGTEIL, A., FURSUND, J., CHRISTENSEN, B., BAERENTZEN, J., AND BRIDSON, R. 2013. Multiphase flow of immiscible fluids on unstructured moving meshes. *IEEE Trans. Vis. Comput. Graphics 20*, 1.

MONAGHAN, J. 2000. SPH without a tensile instability. *J. Comp. Physics 159*, 2, 290 – 311.

MONAGHAN, J. J. 2005. Smoothed Particle Hydrodynamics. *Reports on Progress in Physics 68*, 1703–1759.

MULLEN, P., CRANE, K., PAVLOV, D., TONG, Y., AND DESBRUN, M. 2009. Energy-preserving integrators for fluid animation. *ACM Trans. Graph. 28*, 3.

MULLEN, P., MEMARI, P., DE GOES, F., AND DESBRUN, M. 2011. HOT: Hodge-Optimized Triangulations. *ACM Trans. Graph. 30*, 4.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *Symp. on Comp. Anim.*, 154–159.

MUSETH, K. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph. 32*, 3.

OÑATE, E., IDELSOHN, S., DEL PIN, F., AND AUBRY, R. 2004. The Particle Finite Element Method: An Overview. *Int. J. Comp. Meth. 1*, 2, 267–307.

PREMOŽE, S., TASDIZEN, T., BIGLER, J., LEFOHN, A., AND WHITAKER, R. T. 2003. Particle-based simulation of fluids. *Comp. Graph. Forum 22*, 3, 401–410.

RAVEENDRAN, K., WOJTAN, C., AND TURK, G. 2011. Hybrid Smoothed Particle Hydrodynamics. In *Symp. on Comp. Anim.*, 33–42.

REN, B., LI, C., YAN, X., LIN, M. C., BONET, J., AND HU, S.-M. 2014. Multiple-fluid sph simulation using a mixture model. *ACM Trans. Graph. 33*, 5.

RYCROFT, C. H. 2009. VORO++: A three-dimensional Voronoi cell library in C++. *Chaos: An Interdisciplinary Journal of Nonlinear Science 19*, 4.

SCHECHTER, H., AND BRIDSON, R. 2012. Ghost SPH for animating water. *ACM Trans. Graph. 31*, 4.

SERRANO, M., ESPAÑOL, P., AND ZÚNIGA, I. 2005. Voronoi fluid particle model for Euler equations. *J. Statistical Physics 121*, 133–147.

SHADLOO, M. S., ZAINALI, A., SADEK, S. H., AND YILDIZ, M. 2011. Improved incompressible Smoothed Particle Hydrodynamics method for simulating flow around bluff bodies. *Comput. Methods in Appl. Mech. Eng. 200*, 9-12, 1008 – 1020.

SHAO, S., AND LO, E. Y. 2003. Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Advances in Water Resources 26*, 7, 787–800.

SIDE EFFECTS, 2014. Houdini engine. http://www.sidefx.com.

SIN, F., BARGTEIL, A. W., AND HODGINS, J. K. 2009. A point-based method for animating incompressible flow. In *Symp. on Comp. Anim.*, 247–255.

SOLENTHALER, B., AND PAJAROLA, R. 2008. Density contrast SPH interfaces. In *Symp. on Comp. Anim.*, 211–218.

SOLENTHALER, B., AND PAJAROLA, R. 2009. Predictive-corrective incompressible SPH. *ACM Trans. Graph. 28*, 3.

SOLENTHALER, B., BUCHER, P., CHENTANEZ, N., MLLER, M., AND GROSS, M. 2011. SPH-based shallow water simulation. In *VRIPHYS*, 39–46.

SPRINGEL, V. 2010. *E pur si muove:* Galilean-invariant cosmological hydrodynamical simulations on a moving mesh. *Mon. Not. Roy. Astron. Soc. 401*, 2, 791–851.

STAM, J., AND FIUME, E. 1995. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH*, 129–136.

STAM, J. 1999. Stable fluids. In *SIGGRAPH*, 121–128.

STEINHOFF, J., AND UNDERHILL, D. 1994. Modification of the Euler equations for vorticity confinement: Application to the computation of interacting vortex rings. *Physics of Fluids 6*, 8, 2738–2744.

TAKAHASHI, T., DOBASHI, Y., FUJISHIRO, I., NISHITA, T., AND LIN, M. C. 2015. Implicit formulation for SPH-based viscous fluids. *Comp. Graph. Forum*.

TANG, T. 2004. Moving mesh methods for computational fluid dynamics. In *Contemporary Mathematics*, no. 383, 141–173.

WHITEHURST, R. 1995. A free Lagrange method for gas dynamics. *Mon. Not. Roy. Astron. Soc. 277*, 2, 655–680.

YAN, D.-M., WANG, K., LÉVY, B., AND ALONSO, L. 2011. Computing 2D periodic centroidal Voronoi tessellation. In *Int. Symp. on Voronoi Diagrams*, 177–184.

YAN, D.-M., WANG, W., LÉVY, B., AND LIU, Y. 2013. Efficient computation of clipped Voronoi diagram. *Computer-Aided Design 45*, 4, 843 – 852.

ZHENG, W., ZHU, B., KIM, B., AND FEDKIW, R. 2015. A new incompressibility discretization for a hybrid particle MAC grid representation with surface tension. *J. of Comp. Physics 280*, 96–142.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Trans. Graph. 24*, 3, 965–972.

A Volume-based Derivatives

In this appendix, we detail the construction of our discrete operators based on power diagrams, derived from gradients of local volumes with respect to the site locations $\{\mathbf{q}_i\}_i$ and weights $\{w_i\}_i$. Reynolds' theorem will be a useful tool for our derivations, as it states that the rate of change of the integral of a scalar function f within a region $\mathcal{V}$ is equal to the region integral of the change of f, plus the boundary integral of the rate at which f flows through the region boundary $\partial\mathcal{V}$ with outward unit normal $\mathbf{n}$; i.e.:

$$\nabla\left(\int_{\mathcal{V}} f(\mathbf{x})\ d\mathbf{x}\right) = \int_{\mathcal{V}} \nabla f(\mathbf{x})\, d\mathbf{x} + \int_{\partial\mathcal{V}} f(\mathbf{x})\ (\nabla\mathbf{x})^t\, \mathbf{n}\, d\mathbf{x}.$$

When applied to power cell volumes, the above expression simplifies to boundary integrals (in $\mathbf{q}_i$ or w_i) of the form:

$$\nabla V_i = \sum_{j\in\mathcal{N}_i} \frac{1}{l_{ij}} \int_{\mathcal{A}_{ij}} (\nabla\mathbf{x})^t\, (\mathbf{q}_j - \mathbf{q}_i)\ d\mathbf{x}. \tag{11}$$

We also introduce two identities relevant to our computations. As the power diagram partitions the domain Ω, power cell volumes $\{V_i\}_i$ always sum to the volume of the domain $|\Omega|$, and thus:

$$\nabla_i V_i = -\sum_j \nabla_i V_j. \tag{12}$$

We further express any point $\mathbf{x}$ on a power facet $\mathcal{A}_{ij}$ as:

$$\mathbf{x} = \frac{1}{2}(\mathbf{q}_i + \mathbf{q}_j) + \frac{w_i - w_j}{2\, l_{ij}^2}(\mathbf{q}_j - \mathbf{q}_i) + s_{\mathbf{x}} \mathbf{R}_{\mathbf{x}}(\mathbf{q}_j - \mathbf{q}_i), \tag{13}$$

where the three terms indicate, respectively, the midpoint of the edge connecting $\mathbf{q}_i$ and $\mathbf{q}_j$, a displacement along the edge vector $\mathbf{q}_j - \mathbf{q}_i$, and finally a displacement within the facet $\mathcal{A}_{ij}$ (thus orthogonal to $\mathbf{q}_j - \mathbf{q}_i$) of magnitude $s_{\mathbf{x}}$ along the rotated edge $\mathbf{R}_{\mathbf{x}}(\mathbf{q}_j - \mathbf{q}_i)$, with $\mathbf{R}_{\mathbf{x}}$ as a rotation matrix.

Divergence operator. In §2.3 we defined the divergence operator $\mathbf{D}$ as an nxn row-valued matrix obtained from the derivatives of cell volumes w.r.t. sites $\{\mathbf{q}_i\}_i$. The diagonal term in Eq. (4) is a direct result of Eq. (12), while the off-diagonal terms are computed using Eq. (11) as follows. Noticing that the directional derivative of $s_{\mathbf{x}}\mathbf{R}_{\mathbf{x}}(\mathbf{q}_j - \mathbf{q}_i)$ along $\mathbf{q}_j - \mathbf{q}_i$ is idempotent (due to the orthogonality between these two vectors), we get:

$$\forall \mathbf{x} \in \mathcal{A}_{ij}, \quad (\nabla_{\mathbf{q}_j}\mathbf{x})^t\, (\mathbf{q}_j - \mathbf{q}_i) = \mathbf{q}_j - \mathbf{x}.$$

Thus the off-diagonal term in Eq. (4) reduces to:

$$\nabla_{\mathbf{q}_j} V_i = \frac{1}{l_{ij}} \int_{\mathcal{A}_{ij}} (\mathbf{q}_j - \mathbf{x})\ d\mathbf{x} = \frac{A_{ij}}{l_{ij}}(\mathbf{q}_j - \mathbf{b}_{ij}). \tag{14}$$

Our discrete divergence also satisfies the identity $\sum_i [\mathbf{D}\mathbf{v}]_i = 0$ for any vector field $\{\mathbf{v}_i\}_i$, which is easily verified by writing

$$\begin{aligned}\textstyle\sum_i [\mathbf{D}\mathbf{v}]_i &= \left(\textstyle\sum_i \sum_{j\in\mathcal{N}_i} \mathbf{D}_{ij}\mathbf{v}_j\right) + \left(\textstyle\sum_i \mathbf{D}_{ii}\mathbf{v}_i\right) \\ &= \textstyle\sum_i \sum_{j\in\mathcal{N}_i} \left[\left(\nabla_{\mathbf{q}_j} V_i\right)\mathbf{v}_j - \left(\nabla_{\mathbf{q}_i} V_j\right)\mathbf{v}_i\right] = 0.\end{aligned}$$

Gradient operator. We defined the gradient operator $\mathbf{G}$ in §2.3 as the negated transpose of the divergence. This construction is desirable due to two properties originally discussed in [Serrano et al. 2005], but still valid in our power diagram context. First, this operator is linear accurate, i.e., for any linear function of the form $\{f_i \equiv \mathbf{a}^t\mathbf{q}_i + b\}_i$, with constant vector $\mathbf{a}$ and scalar b, we have: $[\mathbf{G}f]_i = V_i\,\mathbf{a}$. Second, our discrete gradient preserves both linear and angular momentum due to the invariance of cells volumes to global translations and rotations.

Laplacian operator. The Laplacian operator $\mathbf{\Delta}$ used in Eq. (3) was originally computed in [de Goes et al. 2012] based on the derivatives of cell volumes w.r.t. weights. This proof remains valid in our context: we can in fact recover their result by noticing that the directional derivative of a point $\mathbf{x} \in \mathcal{A}_{ij}$ w.r.t. weights is simply $-1/2$, which applied to Eq. (11) yields $\nabla_{w_j} V_i = -A_{ij}/(2l_{ij})$.

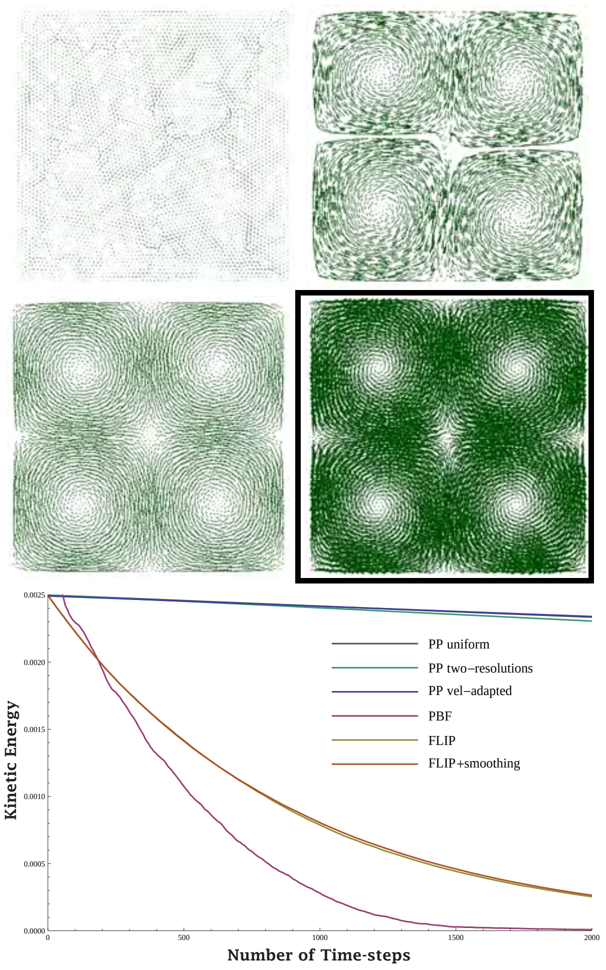


Figure 10: ***Comparisons.*** *For an initial divergence-free velocity field forming four vortices, we compare three Lagrangian methods after 2000 time steps with $dt = 0.01$. Density-constrained methods such as Position Based Fluids [Macklin and Müller 2013] (top left) keep a good particle distribution at the cost of significant motion damping. Projection-based methods such as FLIP [Ando et al. 2013] (top right) exhibit less artificial viscosity, but fail to maintain spacing. Combining FLIP with particle smoothing (middle left) restores evenly-spaced particle, but dissipates kinetic energy. Power particles (middle right) preserve the flow symmetry over time and offer low dissipation as evidenced by the energy plot (bottom).*