RBF Liquids: An Adaptive PIC Solver Using RBF-FD

RAFAEL NAKANISHI*, ICMC-USP, Brazil
FILIPE NASCIMENTO*, ICMC-USP, Brazil
RAFAEL CAMPOS, ICMC-USP, Brazil
PAULO PAGLIOSA, FACOM-UFMS, Brazil
AFONSO PAIVA, ICMC-USP, Brazil

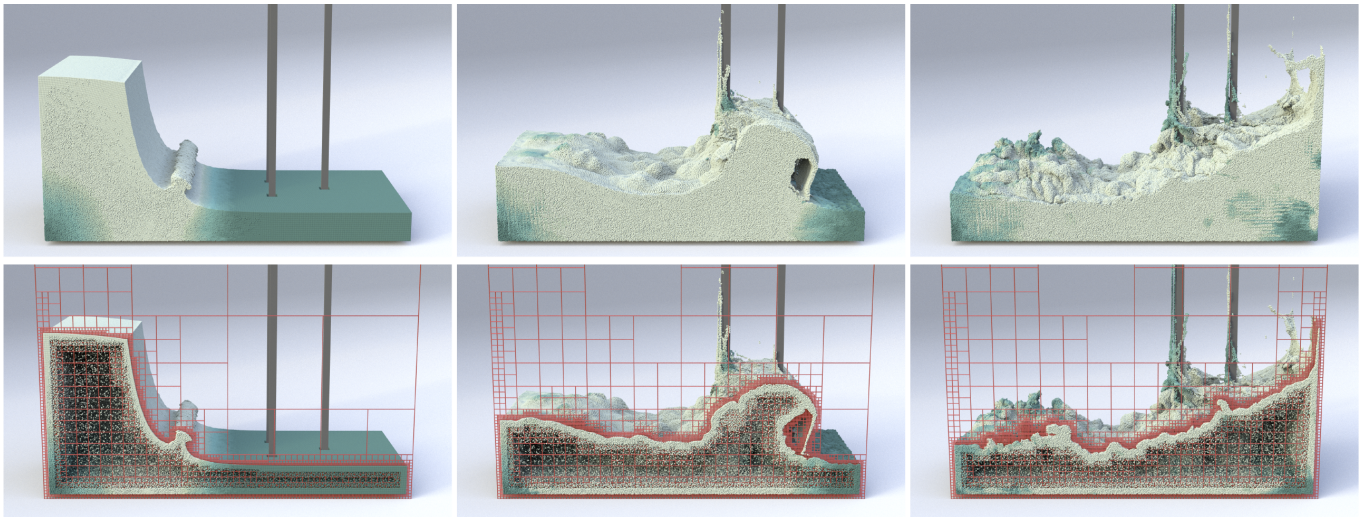


Fig. 1. Simulation of a dam break with obstacles using an octree with a characteristic grid size of 2^{-8} and 4M PIC particles (*top*). Our method adapts the grid and particles around the free-surface, generating 68% fewer particles and 55% fewer fluid cells than regular PIC-based solvers. Besides, our adaptive pressure solver is at least twice faster than regular ones. A cutaway view shows our tree-based grid (red) and its underlying particle sampling (*bottom*). Lighter particle colors indicate fast velocities.

We introduce a novel liquid simulation approach that combines a spatially adaptive pressure projection solver with the Particle-in-Cell (PIC) method. The solver relies on a generalized version of the Finite Difference (FD) method to approximate the pressure field and its gradients in tree-based grid discretizations, possibly non-graded. In our approach, FD stencils are computed by using meshfree interpolations provided by a variant of Radial Basis Function (RBF), known as RBF-Finite-Difference (RBF-FD). This meshfree version of the FD produces differentiation weights on scattered nodes with high-order accuracy. Our method adapts a quadtree/octree dynamically in a narrow-band around the liquid interface, providing an adaptive particle sampling for the PIC advection step. Furthermore, RBF affords an accurate scheme for velocity transfer between the grid and particles, keeping the system's stability and avoiding numerical dissipation. We also present a data structure that connects the spatial subdivision of a quadtree/octree with the topology of its corresponding dual-graph. Our data structure makes the setup of stencils straightforward, allowing its updating without the need to rebuild it from scratch at each time-step. We show the effectiveness and accuracy of our solver by simulating incompressible inviscid fluids and comparing results with regular PIC-based solvers available in the literature.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Radial Basis Function (RBF), Particle-In-Cell (PIC), Adaptive grids, Liquid animation

ACM Reference Format:
Rafael Nakanishi, Filipe Nascimento, Rafael Campos, Paulo Pagliosa, and Afonso Paiva. 2020. RBF Liquids: An Adaptive PIC Solver Using RBF-FD. *ACM Trans. Graph.* 39, 6, Article 1 (December 2020), 13 pages. https://doi.org/10.1145/3414685.3417794

1 INTRODUCTION

Fluid simulation is one of the most prominent areas in computer animation, demanding intense research activity. From small-scale raindrops to large-scale tsunami scenes, FX artists and directors strive for realistic visual effects capable of reproducing the intricate motion of liquids in different scales. This motivates the search for efficient fluid simulation methods that can be made spatially adaptive over the computational domain to reduce the computational

*Joint first authors.

Authors' addresses: Rafael Nakanishi, rnakanishi@usp.br, ICMC-USP, São Carlos, SP, 13566-590, Brazil; Filipe Nascimento, filipecn@usp.br, ICMC-USP, São Carlos, SP, Brazil; Rafael Campos, rafael_campos@usp.br, ICMC-USP, São Carlos, SP, Brazil; Paulo Pagliosa, pagliosa@facom.ufms.br, FACOM-UFMS, Campo Grande, MS, Brazil; Afonso Paiva, apneto@icmc.usp.br, ICMC-USP, São Carlos, SP, Brazil.

burden and the memory load on empty regions (i.e., without fluid) and regions far from the liquid surface.

Adaptive meshes have become attractive in physically-based modeling [Manteaux et al. 2017], primarily for the *pressure projection* [Stam 1999] step in incompressible fluid flow simulations. Usually, in these simulations, the spatial adaptivity is achieved by replacing uniform grids by quadtree (in 2D) or octree (in 3D) grids. Tree-based grids combine the best of both worlds: the ability to use fast and compact Cartesian discretizations, such as finite differences (FD) and finite volumes (FV), and the versatility and accuracy of local mesh refinement. However, the refinement of quadtrees/octrees can produce non-conforming meshes that contain the undesirable *T-junctions* at regions where the mesh resolution changes. This configuration causes a severe limitation in traditional numerical methods because we cannot directly use the standard FD or FV stencils, causing the adaptive creation of the stencil closer to coarse-to-fine grid interfaces a delicate task, as it can ruin the entire simulation due to numerical instabilities that can occur in these regions. A way to overcome this problem is to build stencils using local high-order geometrical interpolations [Batty 2017; Guittet et al. 2015; Min and Gibou 2007; Min et al. 2006]. These interpolations depend on the arrangement of the cells in the neighborhood of the evaluation node and can become quite complicated in 3D simulations, mainly in staggered discretizations which store pressure samples at cell centers and velocity components at cell faces. Another alternative to computing generalized FD stencils is the use of meshfree approximations [Olshanskii et al. 2013; Sousa et al. 2019]. This class of methods requires solving a small least-squares problem at the T-junctions to determine the suitable stencil.

In this paper, we introduce a novel staggered tree-based grid discretization, which provides a high-order approximation of the pressure field and its gradients even in *non-graded* quadtree/octree configurations, i.e., the difference of the resolution levels (depth) between adjacent cells is not constrained. Our method relies on meshfree interpolations computed via *Radial Basis Function* (RBF) [Fasshauer 2007] to enhance and generalize the FD method, providing differentiation weights on scattered node stencils in spatial discretizations. This meshfree version of the FD method is known as *RBF-Finite-Difference* (RBF-FD) [Wright and Fornberg 2006] and in our best knowledge, this is the first time RBF-FD has been used to simulate free-surface flows in the Computer Graphics and Computational Physics literature.

Additionally, we combine our robust and accurate tree-based pressure projection solver with the Particle-in-Cell (PIC) method [Bridson 2015] straightforwardly and efficiently. Our fluid solver gracefully adapts a quadtree/octree dynamically during the simulation in a narrow-band around the liquid surface; this strategy also provides an adaptive particle sampling for the PIC advection step, drastically reducing the number of PIC particles inside the fluid without compromising the visual details of the liquid surface. Moreover, the RBF interpolation gives a higher-order scheme for velocity transfer between the grid and particles. This procedure keeps the stability of the system, avoiding numerical dissipation even in coarse grid cells or in regions with a low density of particles, and is visually competitive with Affine Particle-in-Cell (APIC) [Jiang et al. 2015]. Fig. 1 shows our method in action.

In summary, the contributions of our method are:

- a novel fully adaptive fluid solver that uses both tree-based grids and adaptive particle sampling;
- an approach which generalizes FD stencils in graded and non-graded grids;
- a pressure projection solver on adaptive grids that allows second-order accuracy for the pressure field and its gradients in L_∞ norm;
- a stable and accurate scheme for PIC advection tailored to adaptive grids and non-uniform particle distribution;
- a tree-based data structure able to adapt itself along with the simulation, providing the stencil layouts automatically, i.e., without queries.

2 RELATED WORK

To better contextualize our approach, we organize the existing methods for spatially-adaptive fluid simulation into three groups, according to their discretization.

Adaptive grid structures. In Computer Graphics, octree discretizations of the fluid flow equations are typically built around a staggered FV discretization combined with level set advection. Losasso et al. [2004] introduced an innovative method combining particle level set method and a first-order accurate pressure projection in L_∞ to simulate liquid and smoke on non-graded octrees. Although their method results in a symmetric positive definite (SPD) linear system, which can be efficiently solved by the Conjugate Gradient, the low order accuracy for the pressure can produce spurious velocity fields. Later, Losasso et al. [2006] improved the previous method to achieve second-order accuracy for pressure projection while preserving the non-graded tree structure and the resulting SPD system.

However, the overall accuracy depends on the accuracy of the pressure gradients that are limited to first-order accuracy. Hong et al. [2009] combined the Losasso's octree-based method of first-order of accuracy with Fluid Implicit Particle (FLIP) [Bridson 2015]. Another spatially-adaptive alternative is the use of tall cell grids [Chentanez and Müller 2011; Irving et al. 2006], where the cells in one direction are combined to coarsen the mesh. This approach leads to increased complexity in handling faces shared between regular and tall cells. Ferstl et al. [2014] presented a multigrid solver devoted to fluid simulations on octree grids using finite element method (FEM). However, their method only deals with *graded* octrees, where the difference of depth between adjacent octree cells is constrained to one. As stated by Batty [2017], graded discretizations lead to an unwanted increase in the number of cells. Nielsen and Bridson [2016] provided a short discussion (without complete details) in coupling FLIP with a FEM-based adaptive tile tree structure fully implemented in Autodesk Maya's Bifrost. Gao et al. [2017] introduced an adaptive variant of the Material Point Method (MPM) to simulate a wide range of elastic and plastic materials on non-graded grids. However, this adaptive MPM does not always outperform uniform MPM due to explicit time integration. Recently, Goldade et al. [2019] proposed an adaptive variational FD framework on graded octrees to provide an SPD discretization for viscous liquids.

Adaptive particle sampling. There are adaptive methods that address only the particle sampling aspect in PIC/FLIP solvers. Ando

et al. [2012] proposed adaptive FLIP, which preserves thin sheets of fluid by inserting particles in poorly sampled regions. Recently, the narrow-band FLIP method [Ferstl et al. 2016; Sato et al. 2018] was developed to alleviate the number of particles in FLIP simulations using particles only near the liquid surface. However, they rely on a dense and uniform grid for the pressure projection. On the other hand, particle-based methods, which are typically based on Smoothed Particle Hydrodynamics (SPH) [Ihmsen et al. 2014] and do not employ meshes, can benefit from adaptivity by modifying the number of particles in the simulation and their distribution [Adams et al. 2007; Solenthaler and Gross 2011; Winchenbach et al. 2017].

Tetrahedral meshes and Voronoi diagrams. Klingner et al. [2006] and Chentanez et al. [2007] highlighted how unstructured tetrahedral meshes are inherently adaptive by conforming to complex boundaries and allowing for differently-sized tetrahedra, enabling finer resolution where computational effort is needed. Since colliding objects and boundaries can be animated, the mesh has to be continuously regenerated. Batty et al. [2010] combined embedded boundary techniques with tetrahedral meshes to avoid the remeshing problem. Ando et al. [2013] proposed a second-order accurate liquid solver on adaptive tetrahedral meshes, which combines a variant of FEM with FLIP advection. However, their method drops to the first-order accuracy due to poorly shaped tetrahedra, which occur in coarse-to-fine resolution interfaces. Voronoi-based methods can adaptively remove otherwise uniform sample points [Sin et al. 2009] or adaptively place them [Brochu et al. 2010]. Aanjaneya et al. [2017] proposed an octree-based fluid solver combining power diagrams and *sparse paged grids* [Setaluri et al. 2014]. The key idea of this method is the computation of a power diagram of the octree cell centers providing second-order accuracy for pressure projection while still resulting in an SPD system. Like Ferstl's method, this approach is restricted to graded octrees.

3 AN OVERVIEW OF THE RBF-FD METHOD

RBF is a ubiquitous tool in scattered data interpolation, providing efficient and flexible meshfree methods to solve partial differential equations on arbitrary domains [Fasshauer 2007; Fornberg and Flyer 2015]. In Computer Graphics, RBFs are popular in geometric modeling applications, mainly in surface reconstruction from scattered *Hermite data* (surface points and its normals) [Carr et al. 2001; Macêdo et al. 2011; Ohtake et al. 2005; Turk and O'Brien 2002].

Since the RBF-FD formulas are derived from RBF interpolants, we review the basic concepts of RBF interpolation and then provide a brief introduction to RBF-FD.

3.1 Radial basis function (RBF)

An RBF is a radially symmetric function $\Phi_k : \Omega \subseteq \mathbb{R}^d \to \mathbb{R}$ which relies on the Euclidean distance between its center $\mathbf{x}_k$ and the evaluated point $\mathbf{x} = (x_1, \ldots, x_d)$, both in the domain Ω. Mathematically, an RBF can be represented by $\Phi_k(\mathbf{x}) = \phi(r)$ with $r = \|\mathbf{x} - \mathbf{x}_k\|$, where ϕ is a scalar function on $[0, \infty)$ and $\|\cdot\|$ denotes the Euclidean norm. There are many possible choices for $\phi(r)$, as can be seen in [Fasshauer 2007]. In particular, we use a *Polyharmonic Spline* (PHS) given by odd radial powers:

$$\phi(r) = r^s\,, \quad s = 1, 3, 5, \ldots$$

The main advantage of PHS, in contrast to infinitely smooth RBFs (e.g., Gaussian and Multiquadric), is that they do not require a *shape parameter*, avoiding a laborious fine-tuning for finding a suitable parameter w.r.t. the accuracy and stability of the RBF interpolation.

3.2 RBF interpolation with polynomials

Given a set of distinct nodes $\{\mathbf{x}_k\}_{k=1}^N$ and function values $y_k = y(\mathbf{x}_k) \in \mathbb{R}$, we want to find an interpolant $\mathcal{S}_y : \Omega \to \mathbb{R}$ such that

$$\mathcal{S}_y(\mathbf{x}_i) = y_i\,, \quad \forall i = 1, \ldots, N\,. \tag{1}$$

The general form of an RBF interpolant is given by:

$$\mathcal{S}_y(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k\, \phi(\|\mathbf{x} - \mathbf{x}_k\|) \;+\; \sum_{j=1}^{M} \beta_j\, P_j(\mathbf{x})\,, \tag{2}$$

where $\{P_1(\mathbf{x}), \ldots, P_M(\mathbf{x})\}$ is a basis for the $M = \binom{m+d}{d}$-dimensional space Π_m^d of all d-variate polynomials with degree less than or equal to m. To guarantee uniqueness of the coefficients α_k and β_j, we need to enforce additional constraints:

$$\sum_{k=1}^{N} \alpha_k\, P_j(\mathbf{x}_k) = 0\,, \quad \forall j = 1, \ldots, M\,. \tag{3}$$

The constraints (1) and (3) result in the following linear system of order $N + M$:

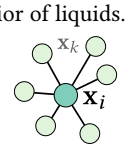
$$\begin{bmatrix} \mathbf{A} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}, \tag{4}$$

where the entries of the square matrix $\mathbf{A}$ (of order N) are given by $\mathbf{A}_{ij} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$, $\mathbf{P}$ is a $N \times M$ matrix with entries $\mathbf{P}_{ij} = P_j(\mathbf{x}_i)$, $\mathbf{O}$ is a square zero matrix of order M, $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^\top$, $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_M]^\top$, $\mathbf{y} = [y_1, \ldots, y_N]^\top$ and $\mathbf{0}$ is a zero vector of length M. Furthermore, the block matrix of the system (4) is SPD (or negative definite), thus ensuring that the system has a unique solution.

An essential remark on RBFs with polynomial augmentation is the guarantee of polynomial precision, i.e., if $y \in \Pi_m^d$ then the set of function values $\{y_k\}_{k=1}^N$ with $N \geq M$ is fitted by $\mathcal{S}_y = y$, except for errors on the order of machine accuracy.

3.3 RBF-FD formulas

In this section, we derive RBF-FD formulas for approximating the set of differential equations that describe the behavior of liquids.



Let $\mathcal{L}$ be a linear differential operator (e.g., partial derivatives $\partial/\partial x_k$, Laplacian operator Δ) and the set $\mathfrak{X}_i$ be a stencil composed by the scattered nodes $\{\mathbf{x}_k\}_{k=1}^N \subseteq \Omega$. For a given location $\mathbf{x}_i$ in the domain Ω (itself included in the stencil $\mathfrak{X}_i$ or not), we desire to approximate $\mathcal{L}y(\mathbf{x})$ evaluated at the center node $\mathbf{x}_i$ as a linear combination of the function values $\{y_k\}_{k=1}^N$ as follows:

$$\mathcal{L}y(\mathbf{x}_i) = \sum_{k=1}^{N} \omega_k\, y_k\,. \tag{5}$$

The coefficients ω_k are known as *differentiation weights* and N is the *stencil size*.

Similar to the RBF interpolation, the weights ω_k are obtained by solving the following linear system:

$$\begin{bmatrix} \mathbf{A} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{\gamma} \end{bmatrix} = \begin{bmatrix} \mathcal{L}\boldsymbol{\phi} \\ \mathcal{L}\mathbf{P} \end{bmatrix}, \tag{6}$$

with $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_N]^\top, \boldsymbol{\gamma} \in \mathbb{R}^M, \mathcal{L}\mathbf{P} = [\mathcal{L}P_1(\mathbf{x}_i), \ldots, \mathcal{L}P_M(\mathbf{x}_i)]^\top$ and $\mathcal{L}\boldsymbol{\phi} = [\mathcal{L}\phi(\|\mathbf{x}_i - \mathbf{x}_1\|), \ldots, \mathcal{L}\phi(\|\mathbf{x}_i - \mathbf{x}_N\|)]^\top$. Since $\mathcal{L}$ operates on the right-hand side of Eqn. (6), the RBF-FD weights for the partial derivatives require the PHS derivatives. By the chain rule, we have:

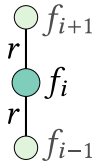
$$\frac{\partial \phi}{\partial x_k}(r) = s x_k r^{s-2} \quad \text{and} \quad \frac{\partial^2 \phi}{\partial x_k^2}(r) = s r^{s-2} + s(s-2) x_k^2 r^{s-4},$$

After solving the linear system (6), the weights stored in $\boldsymbol{\gamma}$ are discarded. For more details about the derivation of the RBF-FD approximation, see the App. A.

The choice of the PHS-polynomial basis is due to its many attractive properties, as reported by Flyer et al. [2016b]. Among these properties, we highlight:

- The approximations are shape parameter-free, and their interpolation matrix has an acceptable condition number;
- The accuracy of $\mathcal{S}y$ and $\mathcal{L}y$ can be improved by increasing the polynomial basis and the PHS order;
- The convergence is dominated by the highest degree in Π_m^d.

Connection to standard FD. Regardless of the choice of the RBF, the standard FD method can be derived from RBF-FD when the stencil is parallel to coordinate axes, and its nodes are collinear. In this case, polynomials in Π_1^1 are used to approximate partial derivatives in the direction x_k. For instance, given a scalar-valued function f and denoting $f(\mathbf{x}_i)$ by f_i, we want to compute $\partial f_i/\partial x_k \approx \omega_{i-1} f_{i-1} + \omega_{i+1} f_{i+1}$. The weights are the solution of the following linear system:



$$\begin{bmatrix} \phi(0) & \phi(2r) & 1 & (\mathbf{x}_{i-1})_k \\ \phi(2r) & \phi(0) & 1 & (\mathbf{x}_{i+1})_k \\ 1 & 1 & 0 & 0 \\ (\mathbf{x}_{i-1})_k & (\mathbf{x}_{i+1})_k & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_{i-1} \\ \omega_{i+1} \\ \gamma_{i-1} \\ \gamma_{i+1} \end{bmatrix} = \begin{bmatrix} \frac{\partial \phi}{\partial z}(r) \\ \frac{\partial \phi}{\partial z}(r) \\ 0 \\ 1 \end{bmatrix}$$

where $r = \|\mathbf{x}_{i+1} - \mathbf{x}_i\| = \|\mathbf{x}_{i-1} - \mathbf{x}_i\|$ and $(\cdot)_k$ is the k-th coordinate of a point. Thus, we have $\omega_{i+1} = 1/2r$ and $\omega_{i-1} = -1/2r$. Therefore,

$$\frac{\partial f_i}{\partial x_k} \approx \frac{f_{i+1} - f_{i-1}}{2r}. \tag{7}$$

Implementation. The weights can easily be computed in parallel architectures. To simplify the evaluation of $\mathcal{L}\mathbf{P}$ on the right-hand side of Eqn. (6), the stencil nodes of $\mathfrak{X}_i$ are translated such that the center $\mathbf{x}_i$ coincides with the origin in $\mathbb{R}^d$. Thus, the entries of $\mathcal{L}\mathbf{P}$ become all zero, except for the few entries equal to 1 or 2, depending on whether the derivative order of $\mathcal{L}$ is of the first or second order. However, switching an RBF center by the center node causes a change of sign in the first-order derivatives, because $\partial\phi/\partial x_k(\|-\mathbf{x}\|) = -\partial\phi/\partial x_k(\|\mathbf{x}\|)$. A Python code snippet to compute RBF-FD weights for Laplacian operator and first-order derivatives in 3D for a single stencil can be found in the App. B.

4 OUR ADAPTIVE PIC SOLVER

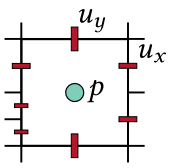
In this section, we describe each component of our fully adaptive PIC solver. Our primary objective is to solve the *Euler equations* for incompressible inviscid fluid flows, given by

$$\frac{D\mathbf{u}}{Dt} = -\frac{\nabla p}{\rho} + \mathbf{f} \quad \text{and} \quad \nabla \cdot \mathbf{u} = 0,$$

where $\mathbf{u}$ represents the fluid velocity; ρ the fluid density, which we assume to be constant and equal to 1; p is the inner fluid pressure; $\mathbf{f}$ is the accumulation of external forces to the fluid; and D/Dt denotes the material derivative.

PIC methods try to combine the benefits of both grid and meshfree discretizations: Lagrangian particles are used to advect all transported quantities, while the pressure projection scheme [Stam 1999] is computed on an Eulerian grid. This scheme projects the velocities into a divergence-free space by solving the *pressure Poisson equation* (PPE): $\Delta p = \delta t\, \rho^{-1} \nabla \cdot \mathbf{u}^*$, where δt denotes the time-step and $\mathbf{u}^*$ an intermediate velocity after the advection. Then, the velocities are updated by $\mathbf{u} = \mathbf{u}^* - \delta t\, \rho^{-1} \nabla p$. In Sec. 4.2, we provide discretizations of these differential operators on adaptive grids.

4.1 Spatial discretization and adaptivity



Our spatial discretization is based on staggered quadtrees/octrees that store velocity components at cell faces (in this section, we use the term *faces* to refer to both faces, in 3D, and edges, in 2D) and pressure samples at cell centers, similar to the staggered grid introduced with the Marker-And-Cell (MAC) method [McKee et al. 2008]. This discretization has more degrees of freedom for velocity quantities than a usual collocated regular grid since they are stored into cell faces of an adaptive grid (where T-junctions may occur). Furthermore, in the MAC method, a grid cell is labeled *empty* if it contains no particles in its interior. Otherwise, it is labeled as a *fluid cell*.

Tree-based structures provide an efficient way to decompose the domain adaptively with a reduced number of resulting cells when compared to other adaptive structures, like tetrahedral meshes. The smaller the number of cells, the smaller the PPE system's size to be solved at each time-step of the simulation.

The adaptivity is lead by the liquid regions, which demand higher numerical accuracy and better representation of the visual details, such as the liquid surface and liquid-solid interfaces. For each time-step, our adaptive grid splits and/or merges cells when needed, keeping a locally-refined grid in a narrow-band around the liquid surface and liquid-solid interfaces, since higher resolutions capture sharper and smaller features of the free-surface and accurate solid-liquid interactions that may disappear when using coarser cells. Also, a region of finer cells is kept at the air (empty) cells near the surface, to avoid that particles change abruptly from finer to coarser cells.

Fluid cells far from the free-surface are coarser than *surface cells* (i.e., fluid cells whose 1-ring cell neighborhood contains at least one cell that is not a fluid cell), since no visual details are obtained from deep inside the liquid and the RBF interpolation does not dissipate velocity quantities at coarse cells.

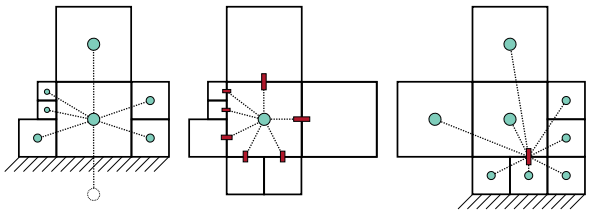


Fig. 2. RBF-FD stencil layouts from 1-ring neighborhood: Laplacian (*left*), divergence (*middle*), and gradient (*right*).

4.2 Discretization of the differential operators

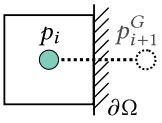
Before computing the RBF-FD weights of each differential operator by solving the system (6), we need to build the stencil layouts based on a MAC grid discretization.

Given a fluid cell C^F, the stencil layout of the *Laplacian* approximation for pressure at the center of C^F is formed by the cell centers of its adjacent face cells in the k-ring cell neighborhood. We initially create the stencils taking $k = 1$, as shown in Fig. 2. This strategy allows us to increase the stencil size, just by increasing k, improving the accuracy of the discretization [Bayona et al. 2017]. Special treatment is needed when C^F is adjacent to a *solid cell* (i.e., a cell containing at least part of a wall or another obstacle); in this case, we create *ghost nodes* outside of the domain Ω containing liquid by reflecting the center of C^F across the boundary $\partial\Omega$ to enforce boundary conditions (Sec. 4.3). These weights are used to assemble a sparse Laplacian matrix associated with the resulting PPE system. The resulting Laplacian matrix from RBF-FD is not symmetric due to the weights derived from distinct arrangements of the stencils. This phenomenon may occur even when the topology of stencils are the same since the distances between neighboring cells may vary, causing such an asymmetry.

Regarding the stencil setup for the first-order partial derivatives, the stencil layout associated with *divergence* of the velocity is also computed at the center of C^F using the velocity components stored in adjacent faces of its k-ring neighborhood. The stencil of the pressure *gradient* components is centered on the faces of C^F; if the cells that share the same face have different resolutions, the remaining nodes are taken from the centers of the adjacent face cells in their k-ring neighborhood.

Notice that if the cells and faces involved in the approximation of the differential operators have the same resolution (i.e., without T-junctions), we can directly apply the standard FD instead of RBF-FD.

4.3 Boundary conditions



Similarly to the standard FD, to determine pressure values p^G at ghost nodes placed outside of the domain Ω, we need to impose the homogeneous Neumann boundary condition $\nabla p \cdot \mathbf{n} = 0\,,\ \forall p \in \partial\Omega$, where $\mathbf{n}$ is the unit normal vector pointing out of $\partial\Omega$. Discretizing Neumann boundary condition with RBF-FD, by Eqn. (7), we have:

$$0 = \nabla p \cdot \mathbf{n} = \frac{\partial p}{\partial x_i} \approx \frac{p^G_{i+1} - p_i}{\|\mathbf{x}^G_{i+1} - \mathbf{x}_i\|}\,.$$

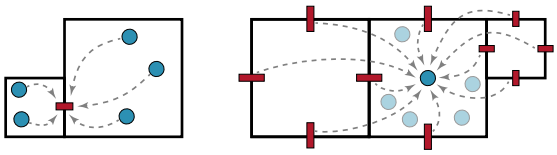


Fig. 3. Velocity transfers between cell faces (*red*) and particles (*blue*): particle → grid (*left*) and grid → particle (*right*).

Therefore, $p^G_{i+1} = p_i$. Hence, during the Laplacian matrix assembly, when a ghost node is present in the stencil, we add its weight to the center node weight, like in the standard FD scheme.

For boundary conditions at solid walls, we impose the free-slip condition, i.e., velocity components normal to the walls are zero, while tangential velocities are equal to the fluid velocity. For the free-surface, we enforce Dirichlet boundary conditions, i.e., we set the pressures in air cells to zero. For implementation details, please see [Kim 2016].

4.4 Particle reseeding

To keep the number of particles in our method low, we resample the particles when the number of particles in a cell is insufficient or when there are many more particles than necessary, regardless of the cell size. For each procedure, we use a user-defined threshold range to manage the number of particles inside a fluid cell C^F. If the number of particles of C^F is out of the threshold range, we delete its particles and create 2^d new sampled particles inside C^F using *Halton points* [Fasshauer 2007].

The reseeding is necessary to preserve fluid cells with particles, avoiding velocity transferences from particles that are far from interpolation centers. Our particle reseeding maintains low density of particles during the simulation without losing details on the surface or affecting the accuracy of our fluid solver. In our experiments, the number of particles varies in the threshold ranges $[3, 12]$ and $[5, 15]$ for 2D and 3D simulations, respectively.

4.5 Particle-grid transfers

Exchange of information between grid and particles is an essential step in PIC/FLIP advection. In our method, we transfer velocities between each structure using the RBF interpolation provided by Eqn. (2) and illustrated by Fig. 3.

For transferring velocities from particles to a given cell face, we look for particles inside its adjacent cells. Thus, the velocity components of these particles are interpolated via RBF to the face center. Reciprocally, to obtain the velocity transfer from the grid to a given particle, we take the velocity components of all faces of the cells incident to the cell which contains this particle. Then, we interpolate the velocities stored in the faces to the particles using RBF again. In particular, for the RBF interpolation, we use PHS of order $s = 5$ augmented with polynomials up to the 2nd degree, where the degree is governed by the number of particles or cell faces.

To demonstrate the accuracy of our transfer, we perform a comparison against Moving Least Square (MLS) with a polynomial basis of Π^d_3 as used by *Higher-Order Particle-in-Cell* (HOPIC) [Edwards and Bridson 2012]. Fig. 4 shows RBF and MLS approximations of a

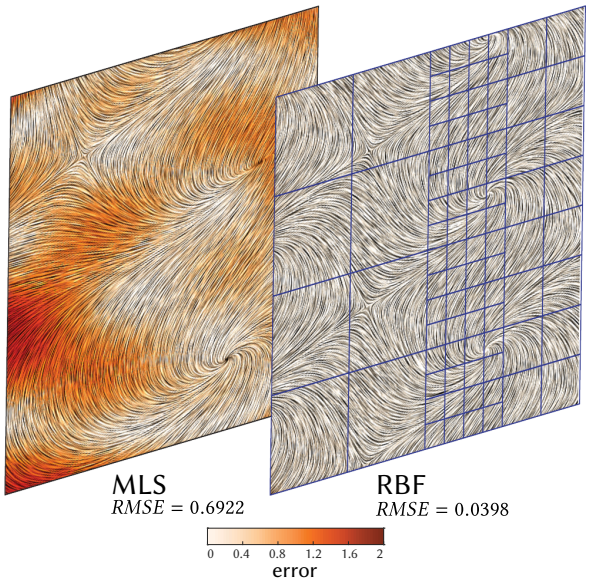


Fig. 4. RBF and MLS approximations of a vector field sampled in a non-graded quadtree (blue).

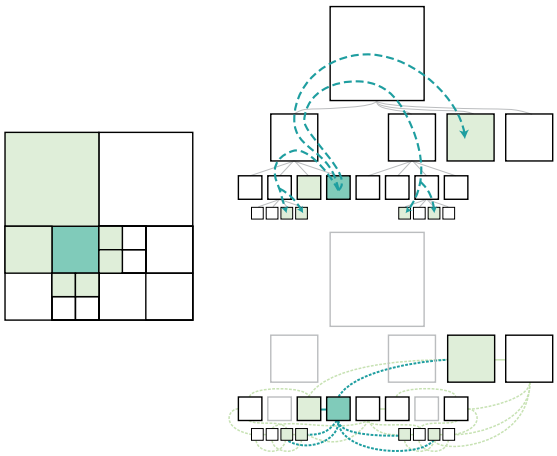


Fig. 5. A 2D stencil defined by a center cell and its direct neighbors. The stencil depicted in a quadtree with the center cell in dark green and its adjacent cells in light green (*left*). With a quadtree representation, the neighborhood calculation requires several tree traversals, drawn as dashed arrows (*top-right*). With a dual quadtree, the adjacency of neighboring cells is explicitly represented (*bottom-right*).

vector field given by:

$$X(x, y) = (\cos(x + 2y), \sin(x - 2y)) \ , \quad (x, y) \in \Omega = [-2, 2]^2 \ .$$

The approximated field is obtained sampling X at the midpoints of the edges of a quadtree discretization of Ω, splatting the sampled field onto a regular background grid of resolution 64^2. We can notice that RBF achieves higher accuracy regarding *root-mean-square error* (*RMSE*), preserving vector field singularities such as vortices and saddles even when using a polynomial space of a lower degree.

5 OUR TREE-BASED GRID DATA STRUCTURE

The implementation of a tree-based staggered grid as a classical octree, in 3D, or a quadtree, in 2D, is not trivial for many reasons:

- The data structure should deal with the tree faces which belong to the staggered grid since a (large) lower depth cell can have as neighbors multiple (small) higher-level cells in the adaptive grid;
- The computation of RBF-FD stencils resulting from different neighborhood configurations require the traversal of several levels in the tree (see Fig. 5);
- As the simulation advances over time, the structure should be updated by splitting or merging leaf cells at each time-step. Otherwise, a new one should be rebuilt.

To address these issues, we consider the vertices and edges of the associated dual quadtree/octree, where each dual vertex corresponds to a single leaf cell, and each dual-edge corresponds to a unique primal tree face which is shared by two adjacent leaf cells. That set of dual edges provides the tree faces where the velocity components should be stored as well as the explicit topological representation required to construct the RBF-FD stencils.

In our framework, we introduce a data structure for adaptive staggered grids, called *Cellgraph*, which combines the geometry of the quadtree/octree and the topology of the corresponding dual-tree into a single graph structure. Each node of the graph is associated with a leaf cell, i.e., a dual vertex of the grid, and holds the axis-aligned bounding box (AABB) of the associated cell, while each edge of the graph is associated with a dual-edge. Such a structure makes the computation of RBF-FD stencils straightforward, as adjacent leaf cells are identified explicitly by the graph edges, and larger stencils can be resolved with simple graph traversal procedures.

Different refinement levels can be achieved by splitting and/or merging nodes, allowing the graph to change its configuration from one simulation step to the next, without the need to construct an entirely new structure from scratch.

5.1 Cell refinement approach

The splitting of a leaf cell C of a quadtree/octree is reflected in the Cellgraph as follows: the node V of the graph associated with C is removed and then new nodes corresponding to the child cells of C, $\{C_i^\star\}$, $i = 1, \ldots, 2^d$, are added to the structure. The region given by the AABB of C is split following the quadtree/octree subdivision rules, and the resulting AABBs are properly assigned to the new graph nodes. All the edges adjacent to V are deleted, and new ones are created to represent, in the graph, the adjacency between each $C_i^\star$ and the 1-ring cell neighborhood of the parent cell C.

Merging a set of leaf cells $\{C_i^\star\}$ into a cell C is equivalent to removing the nodes associated with each $C_i^\star$ from the Cellgraph and then adding a new node V corresponding to the parent cell C. The AABB resulting from the union of the removed nodes is assigned to V. Similarly to splitting, all the edges adjacent to the removed nodes are deleted, and new ones are created to reflect the adjacency between C and its 1-ring cell neighborhood.

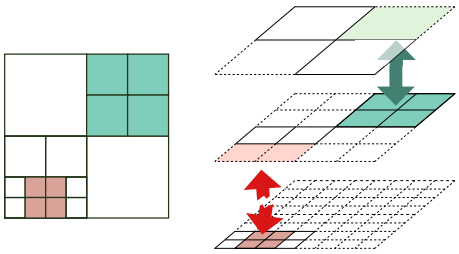


Fig. 6. The address code of a node can be used to check if a group of nodes (cells) can be merged together. On the left, a quadtree depicting two groups of cells, the green group can be merged, but not the red group. On the right, the two groups depicted in the resolution pyramid and their respective address codes on one level up.

It is worth noting that the cells $\{C_i^\star\}$ can be merged if the cell C is their parent. Cellgraph checks if the cells associated with a group of nodes can be merged since each node stores its *level* and its *address code*. Two nodes are siblings and may have the same parent if they have the same level value and compatible address codes. The level of a node is the depth level of its associated cell in the quadtree/octree. The address code of a node represents the index of its associated cell at the same level as a *resolution pyramid* (i.e., a fully refined quadtree/octree). Two address codes are compatible if their indices are mapped to the same address code on a higher level of the pyramid (see Fig. 6).

5.2 Surface tracking

The tracking of the free-surface controls the refinement level of our adaptive data structure. Before starting the simulation, we create a Cellgraph with only one node corresponding to the tree's root cell. The AABB of the initial node is set to cover the whole region of the computational domain, and the root cell is classified as a fluid cell. Next, we split each fluid cell until we reach the maximum depth of the tree, i.e., all fluid cells are fully refined. Then, we label the cells associated with the nodes of the resulting Cellgraph as empty, fluid, surface, or solid cells (see Sec. 4). Moreover, our setup scheme enforces that cells do not contain both particles and solids. Finally, all surface nodes are added to a list.

In addition to the surface cell list, we define the cells belonging to a narrow-band around the liquid surface. While there is a cell C in the k-ring neighborhood of a surface cell C^S, if the level of C is greater than the level of C^S, then C is refined. Otherwise, C is considered to be in the narrow-band. All the cells with the same classification outside the narrow-band are merged up to the lowest level possible. In our experiments, we adopted $k = 2$.

At each time-step of the simulation, the surface cell list is updated to redefine the narrow-band. When updating the surface cell list at the current time-step, we detect new fluid cells and new air cells without the need to iterate over cells not belonging to the neighborhood of the surface cells defined at the previous time-step. One of the advantages of keeping the explicit list of surface cells is that it optimizes the task of cell material type classification. We can avoid checking the presence of particles in all cells based on the assumption that fluid cells become air cells and vice-versa only when changes happen in the list of surface nodes. This strategy also prevents the appearance of air cells inside the liquid or close to walls or obstacles.

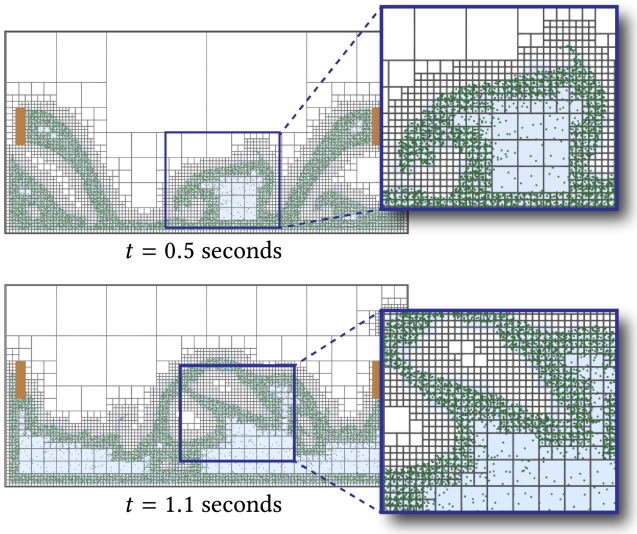


Fig. 7. Two sources (brown) pour a liquid in a container. Our RBF-based method can generate FD stencils in non-graded grids. The fluid cells are highlighted in light blue.

6 RESULTS

To demonstrate the effectiveness of our approach, we simulate incompressible inviscid liquids over different domains. In our experiments, we compute the RBF-FD stencils using PHS of order $s = 3$ and order $s = 5$ for the velocity transfers, both augmented with polynomials of Π_2^d. We use the `Eigen` library [Guennebaud et al. 2010] to solve all linear systems required by our method since the library supports parallel implementations. Specifically, we use Householder QR factorization for the small RBF systems. For the large, sparse, and non-symmetric PPE system, we use Stabilized Biconjugate Gradient (BiCGSTAB) with Jacobi preconditioner [Barrett et al. 1994] in double precision and relative tolerance of 10^{-6}. Regarding the time integration, we use explicit Euler method with the time-step dictated by the CFL condition, when necessary. The rendered images and surface reconstruction were produced using SideFX Houdini.

Adaptivity. In our experiments, we employ up to five levels of refinement for fluid cells, although this is not a strict limitation of our method. Moreover, we denote the smallest *characteristic grid size* by h.

Fig. 1 shows a liquid splash from a dam break with obstacles. Our method can simulate this flow very efficiently in a fully adaptive manner, refining the grid and increasing the number of PIC particles around the free-surface while reducing the overall number of particles and grid cells.

Fig. 7 shows the dynamic adaptability of our method in a simulation where two sources are pouring a liquid into a container. As can be seen, our meshfree approximation can efficiently handle non-graded grid discretizations, allowing for complex stencil configurations near T-junctions.

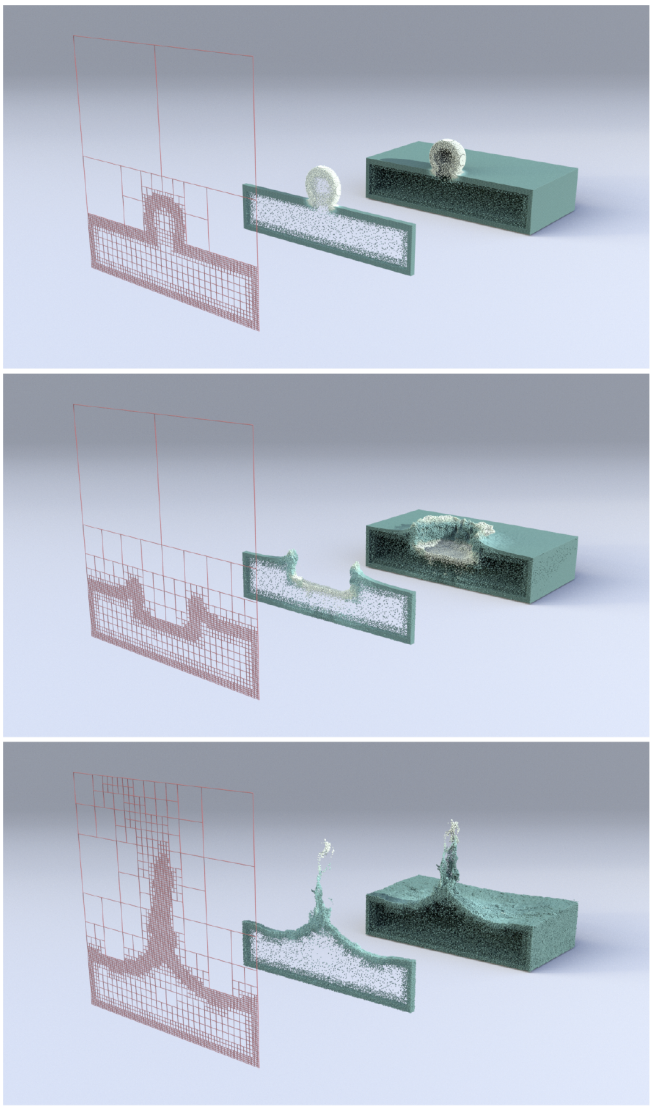


Fig. 8. Our Cellgraph dynamically controls the grid refinement/coarsening as well as the particle sampling in a narrow-band of the interface of a liquid drop without the need to build an entirely new grid from scratch.

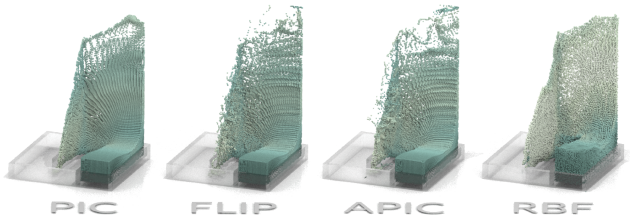


Fig. 9. Liquid flowing around an S-shaped corridor in an octree with characteristic grid size $h = 2^{-6}$. Our RBF-based method preserves fluid sheets in comparison with regular PIC solvers. Also, our method produces stable and energetic splashes, even when using fewer particles.

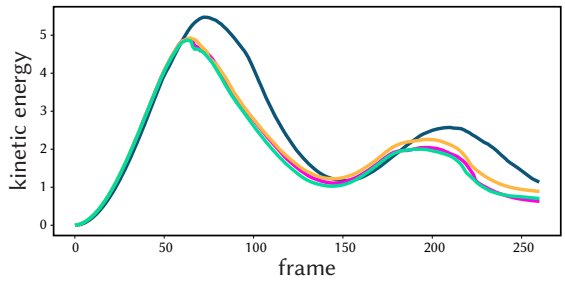


Fig. 10. The plot of the particles' kinetic energy after velocity transfers from the grid for the simulation depicted by Fig. 9: PIC (■), FLIP (■), APIC (■), and RBF (■). Our RBF-based PIC preserves more energy than regular solvers.

Fig. 8 shows the impact of a drop on a liquid layer, the Cellgraph adapts the grid around the interface dynamically as the simulation progresses, tracking the topological changes of the free-surface gracefully. Our structure refines the fluid cells near the liquid surface and is coarse far away from it, without discarding any information of the previous time-step.

Comparisons against regular PIC-based solvers. Fig. 9 shows the behavior of our adaptive RBF-based PIC compared to PIC, FLIP, and APIC (implementations provided by Kim [2016]). Our approach produces a smooth and well-behaved spreading of particles during a liquid splash around an S-shaped corridor's walls due to its accuracy and stability of velocity transfers between the grid and particles.

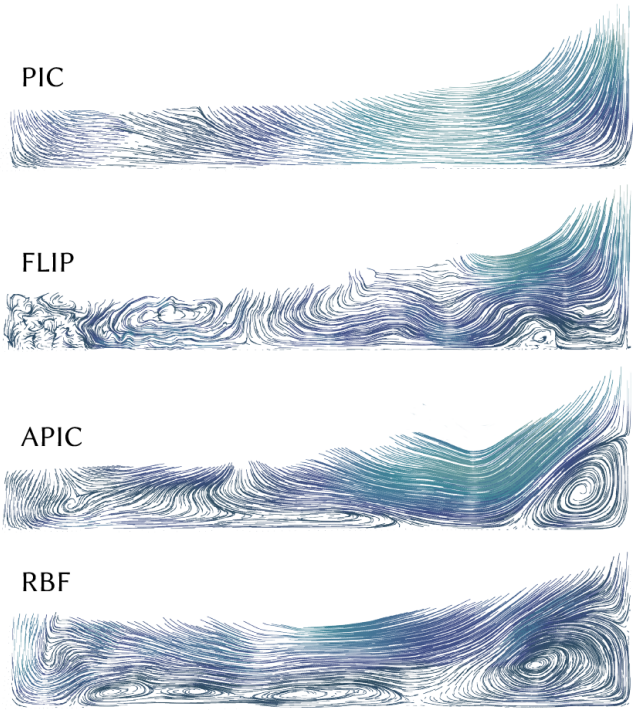


Fig. 11. Streamline visualization of the velocity field of a dam break simulation at $t = 2.2$ seconds, with characteristic grid size of $h = 2^{-6}$ (lighter colors indicate slower velocities). Note that our method preserves more vorticity than regular PIC/FLIP and similar to APIC.

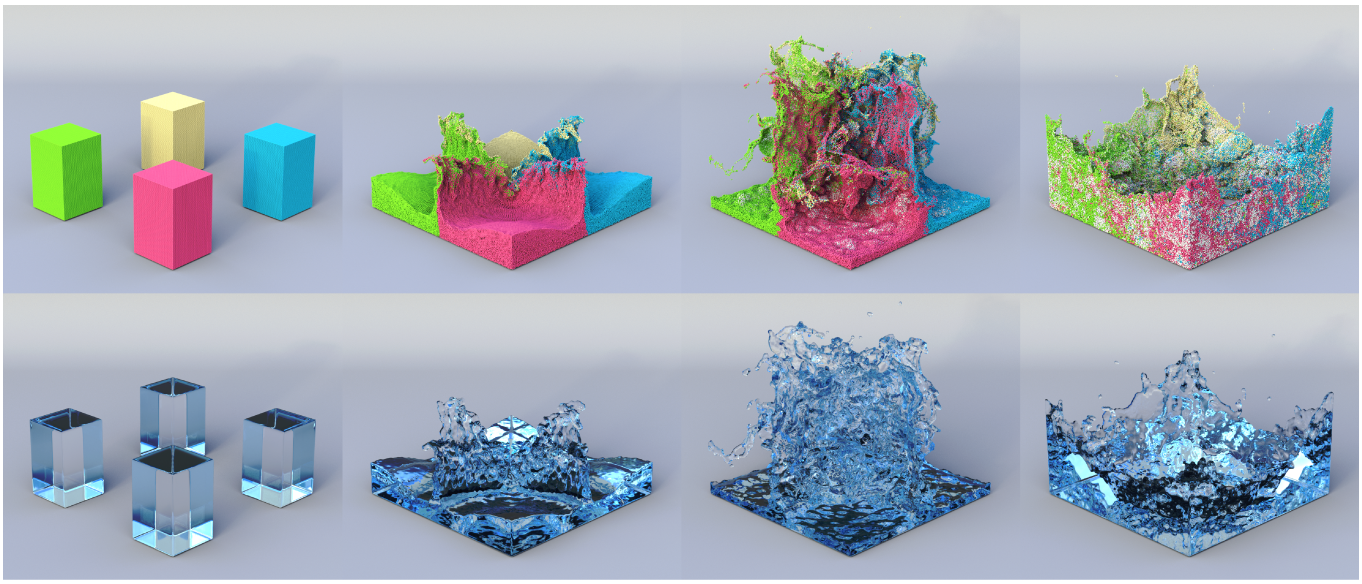


Fig. 12. A quadruple dam break in an octree with characteristic grid size $h = 2^{-7}$ and 1M particles (*top*) and its surface reconstruction (*bottom*). The RBF-FD discretization reaches a speed-up of 2.6× in the PPE linear system solver. The white particles are created during particle reseeding.

As can be seen, our method preserves thin fluid sheets without particle clumping, even using 40% fewer particles than regular PIC-based solvers. Moreover, Fig. 10 shows a plot of the kinetic energy over time. Our RBF-based PIC is more resilient to artificial dissipation without sacrificing stability, keeping the splashes "alive" throughout the simulation.

Besides, the high-order accuracy presented by RBF interpolation and RBF-FD is attested by our experiments. Our method preserves vorticities over long periods, as can be seen in Fig. 11. Due to the severe numerical diffusion of the bilinear/trilinear interpolations commonly present in regular PIC/FLIP simulations, the fine details of the flow vanish quickly. Although APIC produces comparable results for vorticity preservation, their technique is more complicated to implement and stores additional information per particle, such as the affine state matrix and velocity derivatives.

PPE linear system solver. We analyze the efficiency of our adaptive PPE solver using BiCGSTAB with a parallel Jacobi preconditioner in a dam break simulation with four water columns creating a huge splash, as illustrated by Fig. 12. It is worth mentioning that there is a trade-off between the computational cost of building and applying the preconditioner and the convergence speed gain. Note that the PPE matrix size changes throughout the simulation, which requires the preconditioner to be re-computed at each iteration. Thus, we compare the PPE solver's performance when applied to the non-symmetric Laplacian matrix from our adaptive discretization and the SPD system arising from the regular PIC solvers. To eliminate external factors and guarantee a fair comparison, we apply a preconditioned Conjugate Gradient (PCG) for regular solvers, also implemented by `Eigen`, using the same tolerance of the BiCGSTAB and the same parallel preconditioner as well. In this case, we observe that our solver is 2.6× faster than PCG with Jacobi. For more sophisticated preconditioners[1], we perform a comparison between our adaptive solver using BiCGSTAB preconditioned with ILU and the regular solver using PCG with incomplete Cholesky. For this experiment, we obtain a speed-up of 2.5× with 3× fewer iterations to converge.

Table 1. Average timings (in seconds) and statistics of our method per time-step.

Scene	h	#Cells	#Particles	Time				
				T_{grid}	T_{PPE}	T_{Lap}	T_{vel}	total
Fig. 1	2^{-7}	169K (71%)	0.8M (58%)	3.54	0.43	0.10	0.70	6.18
Fig. 1	2^{-8}	745K (45%)	4.0M (32%)	27.97	6.57	0.60	5.88	49.09
Fig. 8	2^{-7}	240K (44%)	1.4M (37%)	7.36	1.25	0.25	1.52	12.97
Fig. 12	2^{-7}	201K (73%)	1.0M (56%)	5.82	0.46	0.12	0.92	9.10

Perfomance analysis. Table 1 presents the average timings and statistics per time-step run on a 16-core AMD Ryzen 9 3950X processor at 3.5GHz and 32GB RAM. The first column *Scene* indicates the experiment performed using our fluid solver. The columns *#Cells* and *#Particles* show the average number of fluid cells of the adaptive grid and the average number of particles, respectively. To show the efficiency of our method, we include in parenthesis the percentage ratio of the number of fluid cells and particles in our method to the number of fluid cells and particles in a regular PIC solver with the same h. In the last columns, we present the average computational time consumed by adaptation of the grid (T_{grid}), solving the PPE linear system (T_{PPE}), assembling the Laplacian matrix (T_{Lap}), velocity transfers between particles and grid (T_{vel}), and the total time. All stages of our code, except the grid update, have been parallelized using OpenMP. This explains why this update is the current bottleneck of our method.

[1] `Eigen` does not provide a parallel implementation for these preconditioners.

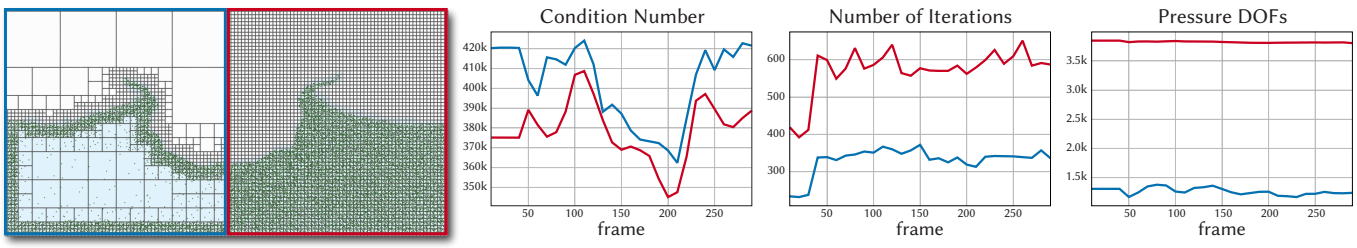


Fig. 13. Liquid drop simulation (*leftmost*). Comparison between our non-graded quadtree (■) and a regular grid (■) with same characteristic size, $h = 2^{-7}$. From left to right, the condition number of the PPE matrix, the number of iterations of the linear system solver, and the number of pressure DOFs.

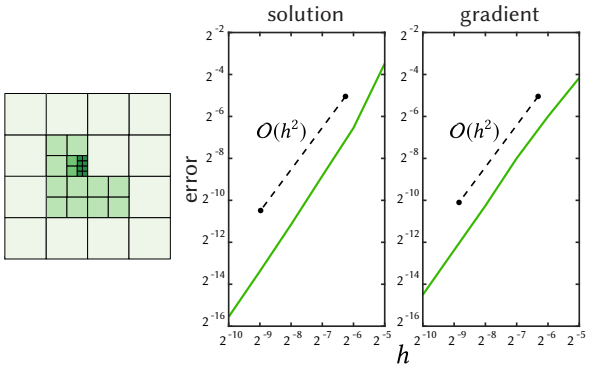


Fig. 14. Initial non-graded quadtree for a Poisson problem (*left*). The log-log plots of the error in L_∞ norm varying with h (*right*). Dashed reference line indicates the second-order slope.

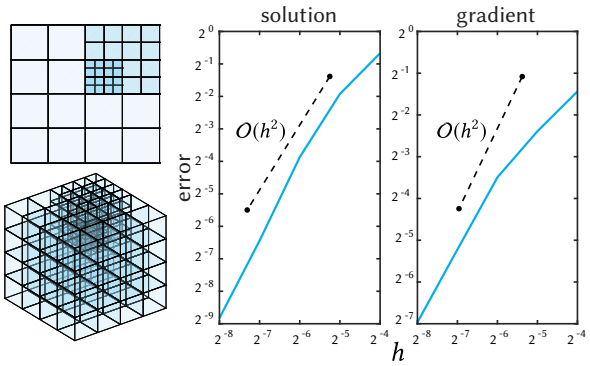


Fig. 15. Initial non-graded octree for a Poisson problem. A 3D view of the adaptive grid (*bottom-left*) and a cutting plane corresponding to $z = \pi/2$ (*top-left*). The log-log plots of the error in L_∞ norm varies with h (*right*). Dashed reference line indicates the second-order slope.

7 NUMERICAL STUDIES

Conditioning. Fig. 13 shows a liquid drop simulation in a regular and adaptive grid discretization side by side. Our adaptive method decreases the computational efforts needed to solve it, employing a non-graded grid with a difference of two levels of resolution between adjacent cells, achieving similar results using 68% fewer fluid cells when compared to the same simulation performed in a regular grid with the same characteristic size. To evaluate how the resulting PPE system is affected by the RBF interpolation, we compare the condition numbers of the PPE system matrices and the convergence of the BiCGSTAB (with relative tolerance set as the machine epsilon in double precision) for each simulation frame. The figure shows that although our non-graded discretization increases the condition number, it does not affect the convergence of the iterative linear solver. Meanwhile, the number of iterations of the linear system solver is less than the uniform case due to its reduced number of degrees of freedom (DOF).

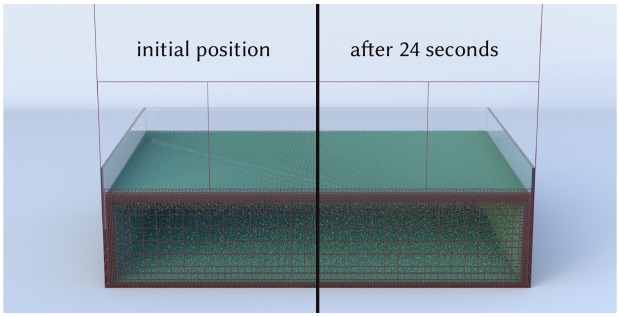


Fig. 16. Standing pool in an octree (red) with characteristic grid size $h = 2^{-8}$.

Convergence test. To numerically attest that our solver achieves a spatial accuracy of second-order in L_∞ norm, we consider 2D and 3D Poisson problems with homogeneous Neumann boundary condition in the grids depicted by Fig. 14 and Fig. 15, where the exact solutions are $p(x, y) = \cos(x)\cos(y) - 1$ and $p(x, y, z) = \cos(x)\cos(y)\cos(z) - 1$ on $\Omega = [0, \pi]^d$, respectively. The grid cells are recursively subdivided, and the L_∞ error for each refinement is computed. The log-log plots (base 2) show second-order accuracy for the solution and its gradient using our approach. An important remark is that the resulting linear systems are singular. To avoid this problem, we need to impose a Dirichlet boundary condition at one center node of the domain to obtain a nonsingular system.

Volume preservation. Fig. 16 shows a hydrostatic standing pool simulation, which yields a linear pressure distribution. Our method increases the resolution on the boundaries to provide an accurate pressure field, preventing error propagation, and consequently avoiding volume loss even after a long simulation period.

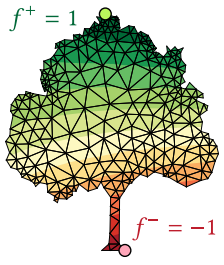
8 DISCUSSION AND LIMITATIONS

In this section, we highlight limitations present in our method, as well as potential improvements to the current state of the simulator and future projects we are working on.

Stencil size. Recent research has shown promising applications of the RBF-FD technique in the numerical solution of PDEs [Bayona et al. 2017; Flyer et al. 2016a]. In those applications, the stencil size is vital for the accuracy of the solution. For the best accuracy, it is recommended the stencil size should be approximately twice the number of polynomial terms. In our framework, we can increase the stencil size, expand the k-ring neighborhood, or complement it by using the k-nearest neighbors of the center node. In our simulations, the stencil size can reach up to 48 nodes. High-order approximations with minimal meshfree stencils [Seibold 2008] are yet a topic we have to explore further since the polynomial degree, the size, and the layout of the stencil can impact the final result of the simulation.

Levels of resolution. Due to the RBF interpolation and RBF-FD scheme we have used, the proposed method can deal with non-graded grids whose difference of the resolution levels between adjacent cells may be higher than two, although in our experiments we allow a level difference of up to two. While this is already an improvement regarding other adaptive approaches found in the literature, allowing more than two resolution levels between cells has to be further studied since it requires more stencil nodes to compute the differential weights and maintaining a reasonable number of particles over coarser cells.

Lookup table. Concerning the performance, the computation of the weights of generalized stencils is a compute-intensive task in our simulations. We have to calculate their differential operators at each time-step for each cell and its neighbors in the staggered grid. An alternative approach would be to build a geometric dictionary storing all different stencil configurations and their respective weights as they appear during the simulation. Such a scheme would avoid computing the weights of already mapped stencils, thus increasing the overall performance of the method. Moreover, recent studies have shown promising solutions for solving RBF-FD stencils in parallel by using point indexing [Elliott et al. 2019], which would be another alternative to be investigated for our framework.



Complex domains. The differential operators derived from RBF-FD can be trivially adapted to complex domains modeled by unstructured meshes. For instance, given a simplicial mesh, we can solve the Laplace equation $\Delta f = 0$ with Dirichlet boundary conditions using RBF-FD (see inset). In this experiment, we take the 2-ring vertex neighborhood to build the RBF-FD stencils, while the boundary conditions $f^- = -1$ and $f^+ = 1$ are imposed on the bottommost and topmost vertices, respectively. However, for fluid flow simulations, the Neumann boundary condition on arbitrary domains is not a trivial task for RBF-FD and is a limitation of our method. As future investigation, we intend to study the possibility of enforcing the Neumann boundary condition as an additional interpolation constraint using Hermite RBFs [Macêdo et al. 2011].

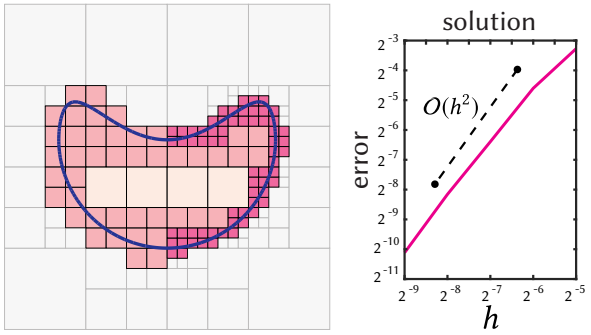
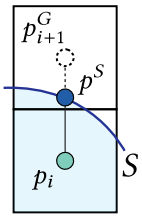


Fig. 17. On the left, initial non-graded quadtree for a Poisson problem with Dirichlet boundary condition at the level-set (blue), the gray cells are discarded. On the right, the log-log plot of the error in L_∞ norm varying with h. Dashed reference line indicate ideal second-order slope.

RBFs near domain boundaries. Our velocity transfer step relies on an RBF-based interpolation scheme known to present suitable results over smooth continuous functions, as assumed for the Euler equations. Near the solid walls and obstacles, the velocity field can display some oscillatory behavior (Runge's phenomenon) because the RBF nodes become highly one-sided. To avoid these unwanted oscillations, during the velocity transfer from the grid cells to the particles, we clamp the interpolated particle velocities in the range of the velocities stored in the cell edges/faces. A more detailed discussion about RBF interpolations near boundaries can be found in [Bayona et al. 2019].



Boundary condition for free-surface. For the free-surface flow, the exact Dirichlet pressure boundary condition, $p = 0$, needs to be ensured at the free-surface. Similar to [Enright et al. 2003], we can improve the order accuracy of our discretization by using level-sets. Let S be the zero level-set, the cell centers inside S contain pressure DOFs, the cell centers outside S and belonging to a stencil cut by S are assigned ghost values p^G, and the remaining centers are discarded. For each stencil edge $(\mathbf{x}_i, \mathbf{x}_{i+1})$ cutted by S, we compute the point $\mathbf{x}^S \in S$ where the interface intersects the edge. Then, the Dirichlet boundary condition $p^S = 0$ is applied at $\mathbf{x}^S$. To assess the accuracy of this strategy, we solve a Poisson equation with an analytic pressure field given by $p(x, y) = (y-x^2+1)^4+(x^2+y^2)-1$ and the level-set $S = p^{-1}(0)$ in the domain $\Omega = [-1.5, 1.5]\times[-1.75, 1.25]$. Fig. 17 depicts a discretization of Ω provided by a quadtree where adjacent cells can differ in depth by up to 2 levels. The log-log plot shows a second-order accurate discretization of the Dirichlet pressure boundary condition at S using our approach. Regarding level-sets, to avoid the usage of an additional high-resolution grid to compute Eulerian level-sets [Aanjaneya et al. 2017], we intend to take advantage of the particles to track the free-surface by using the boundary particles to define the level-set [Sandim et al. 2019]. We leave the inclusion of this boundary treatment to our framework as future work.

9 CONCLUSION AND FUTURE WORK

We have introduced a novel approach for liquid simulations which combines an RBF-FD pressure projection solver with the PIC method. Our RBF-FD solver can compute the pressure field and its gradients from generalized stencils on staggered cells of graded/non-graded tree-based grids. As far as we know, this is the first application of RBF-FD in Computer Graphics. Besides, we implement an adaptive grid representation, called Cellgraph, capable of updating itself over time without discarding its previous state. The results obtained with our framework, both in 2D and 3D, are visually convincing, preserving details and energy even in the presence of coarser cells and T-junctions, due to the accurate interpolation scheme and discrete differentiation we adopted.

Future work. Our current implementation only simulates single-phase inviscid free-surface flow. It opens some opportunities for more features to be added to our framework, like viscosity [Goldade et al. 2019], multiple fluids [Yang et al. 2015], two-way solid-fluid interactions [Ng et al. 2009], and surface tension [Da et al. 2016]. For instance, we believe that including explicit surface tension in our pipeline would be straightforward once we are able to track particles on the free-surface [Sandim et al. 2016, 2020]. Regarding the PPE solver, a topic for further investigation is to build a multigrid method based on RBF due to its excellent scalability and opportunities for parallelization. The Cellgraph refinement can be improved by choosing other refinement functions like vorticity or curvature at the surface, as used by Ando et al. [2013]. The dynamic updating of the topology throughout the simulation without the need for reinitialization brings other advantages, such as the detection of solids only when needed and tracking the surface cells, which in the future can help represent interfaces in multi-material simulations or in the computation of level sets. There is plenty of room for further optimizations in Cellgraph, such as arranging nodes and edges in memory layouts to improve cache access and parallel processing to update the graph's disjoint regions. Currently, our particle reseeding does not act on surface cells. Consequently, it can considerably increase the number of PIC particles in the system. Thus, another direction of investigation would be to study an efficient particle reseeding scheme capable of handling the surface cells. Also, we intend to investigate the linear and angular momentum conservation. Finally, we also plan to implement parallel versions of our solver in heterogeneous environments made up of multiple CPUs and GPUs.

ACKNOWLEDGMENTS

We want to thank Fabrício Simeoni de Sousa and Antonio Castelo for insightful discussions throughout this work, and the SIGGRAPH Asia reviewers for their comments. We also thank Cristin Barghiel from SideFX for their kind donation of the Houdini licenses and Leonardo Martinussi for the computing infrastructure at ICMC-USP. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brazil (CAPES), National Council for Scientific and Technological Development – Brazil (CNPq) fellowship #301642/2017-6, and São Paulo Research Foundation (FAPESP) under grants #2018/06145-4 and #2019/23215-9. The computational resources were provided by the Center for Mathematical Sciences Applied to Industry (CeMEAI), also funded by FAPESP (grant #2013/07375).

REFERENCES

M. Aanjaneya, M. Gao, H. Liu, C. Batty, and E. Sifakis. 2017. Power diagrams and sparse paged grids for high resolution adaptive liquids. *ACM Trans. Graph.* 36, 4 (2017), 1–12.

B. Adams, M. Pauly, R. Keiser, and L. J. Guibas. 2007. Adaptively sampled particle fluids. *ACM Trans. Graph.* 26, 3 (2007).

R. Ando, N. Thurey, and R. Tsuruno. 2012. Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE Trans. Vis. Comput. Graph.* 18, 8 (2012), 1202–1214.

R. Ando, N. Thürey, and C. Wojtan. 2013. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph.* 32, 4 (2013), 103:1–103:10.

R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. 1994. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM.

C. Batty. 2017. A cell-centred finite volume method for the Poisson problem on non-graded quadtrees with second order accurate gradients. *J. Comput. Phys.* 331 (2017), 49–72.

C. Batty, S. Xenos, and B. Houston. 2010. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. *Comput. Graph. Forum* 29, 2 (2010), 695–704.

V. Bayona, N. Flyer, and B. Fornberg. 2019. On the role of polynomials in RBF-FD approximations: III. Behavior near domain boundaries. *J. Comput. Phys.* 380 (2019), 378 – 399.

V. Bayona, N. Flyer, B. Fornberg, and G. A. Barnett. 2017. On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs. *J. Comput. Phys.* 332 (2017), 257–273.

R. Bridson. 2015. *Fluid Simulation* (2nd ed.). A. K. Peters.

T. Brochu, C. Batty, and R. Bridson. 2010. Matching fluid simulation elements to surface geometry and topology. In *ACM Trans. Graph.*, Vol. 29.

J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. 2001. Reconstruction and representation of 3D objects with radial basis functions. In *Proc. of SIGGRAPH '01*. ACM, 67–76.

N. Chentanez, B. E. Feldman, F. Labelle, J. F. O'Brien, and J. R. Shewchuk. 2007. Liquid simulation on lattice-based tetrahedral meshes. *Comput. Animat.* (2007), 219–228.

N. Chentanez and M. Müller. 2011. Real-time Eulerian water simulation using a restricted tall cell grid. In *ACM Trans. Graph.*, Vol. 30. 82:1–82:10.

F. Da, D. Hahn, C. Batty, C. Wojtan, and E. Grinspun. 2016. Surface-only liquids. *ACM Trans. Graph.* 35, 4 (2016), 1–12.

E. Edwards and R. Bridson. 2012. A high-order accurate particle-in-cell method. *Int. J. Numer. Meth. Eng.* 90, 9 (2012), 1073–1088.

S. Elliott, R. R. P. Kumar, N. Flyer, T. Ta, and R. Loft. 2019. Implementation of a scalable, performance portable shallow water equation solver using radial basis function-generated finite difference methods. *Int. J. High Perform. Comput. Appl.* 33, 4 (2019), 619–631.

D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw. 2003. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proc. of the 4th ASME-JSME Joint Fluids Engineering Conference*. 337–342.

G. F. Fasshauer. 2007. *Meshfree Approximation Methods with MATLAB.* World Scientific.

F. Ferstl, R. Ando, C. Wojtan, R. Westermann, and N. Thuerey. 2016. Narrow band FLIP for liquid simulations. *Comput. Graph. Forum* 35, 2 (2016), 225–232.

F. Ferstl, R. Westermann, and C. Dick. 2014. Large-scale liquid simulation on adaptive hexahedral grids. *IEEE Trans. Vis. Comput. Graph.* 20, 10 (2014), 1405–1417.

N. Flyer, G. A. Barnett, and L. J. Wicker. 2016a. Enhancing finite differences with radial basis functions: experiments on the Navier-Stokes equations. *J. Comput. Phys.* 316 (2016), 39–62.

N. Flyer, B. Fornberg, V. Bayona, and G. A. Barnett. 2016b. On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy. *J. Comput. Phys.* 321 (2016), 21–38.

B. Fornberg and N. Flyer. 2015. Solving PDEs with radial basis functions. *Acta Numerica* 24 (2015), 215–258.

M. Gao, A. P. Tampubolon, C. Jiang, and E. Sifakis. 2017. An adaptive generalized interpolation material point method for simulating elastoplastic materials. *ACM Trans. Graph.* 36, 6 (2017), 1–12.

R. Goldade, Y. Wang, M. Aanjaneya, and C. Batty. 2019. An adaptive variational finite difference framework for efficient symmetric octree viscosity. *ACM Trans. Graph.* 38, 4 (2019).

G. Guennebaud, B. Jacob, and Others. 2010. Eigen v3. http://eigen.tuxfamily.org.

A. Guittet, M. Theillard, and F. Gibou. 2015. A stable projection method for the incompressible Navier-Stokes equations on arbitrary geometries and adaptive quad/octrees. *J. Comput. Phys.* 292 (2015), 215–238.

W. Hong, D. H. House, and J. Keyser. 2009. *An Adaptive Sampling Approach to Incompressible Particle-Based Fluid.* Ph.D. Dissertation. Texas A & M University.

M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner. 2014. SPH Fluids in Computer Graphics. In *Eurographics 2014 - State of the Art Reports.*
G. Irving, E. Guendelman, F. Losasso, and R. Fedkiw. 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph.* 25, 3 (2006), 805–811.
C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. 2015. The affine particle-in-cell method. *ACM Trans. Graph.* 34, 4 (2015), 51:1–51:10.
D. Kim. 2016. *Fluid Engine Development.* CRC Press.
B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'Brien. 2006. Fluid animation with dynamic meshes. In *ACM Trans. Graph.*, Vol. 25. 820–825.
F. Losasso, R. Fedkiw, and S. Osher. 2006. Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids* 35, 10 (2006), 995 – 1010.
F. Losasso, F. Gibou, and R. Fedkiw. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. Graph.* 23, 3 (2004), 457–462.
I. Macêdo, J. P. Gois, and L. Velho. 2011. Hermite radial basis functions implicits. *Comput. Graph. Forum* 30, 1 (2011), 27–42.
P. L. Manteaux, C. Wojtan, R. Narain, S. Redon, F. Faure, and M. P. Cani. 2017. Adaptive physically based models in computer graphics. *Comput. Graph. Forum* 36, 6 (2017), 312–337.
S. McKee, M. F. Tomé, V. G. Ferreira, J. A. Cuminato, A. Castelo, F. S. Sousa, and N. Mangiavacchi. 2008. The MAC method. *Computers & Fluids* 37, 8 (2008), 907–930.
C. Min and F. Gibou. 2007. A second order accurate level set method on non-graded adaptive cartesian grids. *J. Comput. Phys.* 225, 1 (2007), 300–321.
C. Min, F. Gibou, and H. D. Ceniceros. 2006. A supra-convergent finite difference scheme for the variable coefficient Poisson equation on non-graded grids. *J. Comput. Phys.* 218, 1 (2006), 123–140.
Y. T. Ng, C. Min, and F. Gibou. 2009. An efficient fluid–solid coupling algorithm for single-phase flows. *J. Comput. Phys.* 228, 23 (2009), 8807–8829.
M. B. Nielsen and R. Bridson. 2016. Spatially adaptive FLIP fluid simulations in Bifrost. In *ACM SIGGRAPH 2016 Talks.* 41:1–41:2.
Y. Ohtake, A. Belyaev, and H. Seidel. 2005. 3D scattered data interpolation and approximation with multilevel compactly supported RBFs. *Graph. Models* 67, 3 (2005), 150–165.
M. A. Olshanskii, K. M. Terekhov, and Y. V. Vassilevski. 2013. An octree-based solver for the incompressible Navier–Stokes equations with enhanced stability and low dissipation. *Computers & Fluids* 84 (2013), 231–246.
M. Sandim, D. Cedrim, L. G. Nonato, P. Pagliosa, and A. Paiva. 2016. Boundary detection in particle-based fluids. *Comput. Graph. Forum* 35, 2 (2016), 215–224.
M. Sandim, N. Oe, D. Cedrim, P. Pagliosa, and A. Paiva. 2019. Boundary particle resampling for surface reconstruction in liquid animation. *Computers & Graphics* 84 (2019), 55 – 65.
M. Sandim, A. Paiva, and L. H. de Figueiredo. 2020. Simple and reliable boundary detection for meshfree particle methods using interval analysis. *J. Comput. Phys.* 420 (2020), 109702.
T. Sato, C. Wojtan, N. Thuerey, T. Igarashi, and R. Ando. 2018. Extended narrow band FLIP for liquid simulations. *Comput. Graph. Forum* 37, 2 (2018), 169–177.
B. Seibold. 2008. Minimal positive stencils in meshfree finite difference methods for the Poisson equation. *Comput. Methods Appl. Mech. Eng.* 198, 3–4 (2008), 592–601.
R. Setaluri, M. Aanjaneya, S. Bauer, and E. Sifakis. 2014. SPGrid: a sparse paged grid structure applied to adaptive smoke simulation. *ACM Trans. Graph.* 33, 6 (2014).
F. Sin, A. W. Bargteil, and J. K. Hodgins. 2009. A point-based method for animating incompressible flow. In *Comput. Animat.* ACM, 247–255.
B. Solenthaler and M. Gross. 2011. Two-scale particle simulation. *ACM Trans. Graph.* 30, 4 (2011), 81:1–81:8.
F. S. Sousa, C. F. Lages, J. L. Ansoni, A. Castelo, and A. Simao. 2019. A finite difference method with meshless interpolation for incompressible flows in non-graded tree-based grids. *J. Comput. Phys.* 396 (2019), 848–866.
J. Stam. 1999. Stable fluids. In *Proc. of SIGGRAPH '99.* ACM, 121–128.
G. Turk and J. F. O'Brien. 2002. Modelling with implicit surfaces that interpolate. *ACM Trans. Graph.* 21, 4 (2002), 855–873.
R. Winchenbach, H. Hochstetter, and A. Kolb. 2017. Infinite continuous adaptivity for incompressible SPH. *ACM Trans. Graph.* 36, 4 (2017), 102:1–102:10.
G. B. Wright and B. Fornberg. 2006. Scattered node compact finite difference-type formulas generated from radial basis functions. *J. Comput. Phys.* 212, 1 (2006), 99–123.
T. Yang, J. Chang, B. Ren, M. C. Lin, J. J. Zhang, and S. Hu. 2015. Fast Multiple-Fluid Simulation Using Helmholtz Free Energy. *ACM Trans. Graph.* 34, 6 (2015).

A DERIVING THE RBF-FD APPROXIMATION

The approximation (5) is obtained by imposing that $\mathcal{L}y(\mathbf{x}_i)$ be exact for the interpolant $\mathcal{S}_y(\mathbf{x})$ with the constraints (1) and (3). Applying the operator $\mathcal{L}$ on Eqn. (2) and then evaluating at the point $\mathbf{x}_i$, we have

$$\begin{aligned}
\mathcal{L}\mathcal{S}_y(\mathbf{x}_i) &= \sum_{k=1}^{N} \alpha_k \, \mathcal{L}\phi(\|\mathbf{x}_i - \mathbf{x}_k\|) + \sum_{j=1}^{M} \beta_j \, \mathcal{L}P_j(\mathbf{x}_i) \\
&= \begin{bmatrix} \mathcal{L}\boldsymbol{\phi}^\top & \mathcal{L}\mathbf{P}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{L}\boldsymbol{\phi}^\top & \mathcal{L}\mathbf{P}^\top \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{O} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{\omega}^\top & \boldsymbol{\gamma}^\top \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} \\
&= \sum_{k=1}^{N} \omega_k \, y_k \, .
\end{aligned}$$

The transpose of $[\boldsymbol{\omega}^\top, \boldsymbol{\gamma}^\top]$ is the solution of the linear system (6). Notice that the weights stored in $\boldsymbol{\gamma}$ are ignored because their entries are multiplied by zero.

B PYTHON CODE FOR RBF-FD WEIGHTS

```python
import numpy as np
import scipy.spatial.distance as sd

def rbf_fd_weights(X,ctr,s,d):
  #   X : each row contains one node in R^3
  # ctr : center (evaluation) node
  # s,d : PHS order and polynomial degree
  rbf  = lambda r,s: r**s
  Drbf = lambda r,s,xi: s*xi*r**(s-2)
  Lrbf = lambda r,s: s*(s+1)*r**(s-2)
  n = X.shape[0]
  for i in range(3): X[:,i] -= ctr[i]
  DM = sd.squareform(sd.pdist(X))
  D = np.sqrt(X[:,0]**2 + X[:,1]**2 + X[:,2]**2)
  A,lap,dx = rbf(DM,s),Lrbf(D,s),Drbf(D,s,-X[:,0])
  dy,dz = Drbf(D,s,-X[:,1]),Drbf(D,s,-X[:,2])
  b = np.vstack((lap,dx,dy,dz)).T
  if d>-1: # adding polynomials
    m = int((d+3)*(d+2)*(d+1)/6)
    O,c = np.zeros((m,m)),0
    P,LP = np.zeros((n,m)),np.zeros((m,4))
    for k in range(d+1):
      for j in range(d+1):
        for i in range(d+1):
          if i+j+k > d: continue
          if i+j+k == 1 and i == 1: cx1 = c
          if i+j+k == 1 and j == 1: cy1 = c
          if i+j+k == 1 and k == 1: cz1 = c
          if i+j+k == 2 and i == 2: cx2 = c
          if i+j+k == 2 and j == 2: cy2 = c
          if i+j+k == 2 and k == 2: cz2 = c
          P[:,c],c = X[:,0]**i*X[:,1]**j*X[:,2]**k,c+1
    if d>0: LP[cx1,1],LP[cy1,2],LP[cz1,3] = 1,1,1
    if d>1: LP[cx2,0],LP[cy2,0],LP[cz2,0] = 2,2,2
    A = np.block([[A,P],[P.T,O]])
    b = np.block([[b],[LP]])
  # each column contains the weights for the
  # Laplacian, d/dx1, d/dx2, d/dx3, respectivly.
  weights = np.linalg.solve(A,b)
  return weights[:n,:]
```

Received January 2020