Sharp Interface Approaches and Deep Learning Techniques for Multiphase Flows

Frederic Gibou[a,b], David Hyde[c], Ron Fedkiw[c]

[a]*Department of Mechanical Engineering, UC Santa Barbara, CA 93106-5070*
[b]*Department of Computer Science, UC Santa Barbara, CA 93106-5070*
[c]*Department of Computer Science, Stanford University, CA 94305*

Abstract

We present a review on numerical methods for simulating multiphase and free surface flows. We focus in particular on numerical methods that seek to preserve the discontinuous nature of the solutions across the interface between phases. We provide a discussion on the Ghost-Fluid and Voronoi Interface methods, on the treatment of surface tension forces that avoid stringent time step restrictions, on adaptive grid refinement techniques for improved efficiency and on parallel computing approaches. We present the results of some simulations obtained with these treatments in two and three spatial dimensions. We also provide a discussion of Machine Learning and Deep Learning techniques in the context of multiphase flows and propose several future potential research thrusts for using deep learning to enhance the study and simulation of multiphase flows.

Keywords: Multiphase-Flows, Ghost-Fluid Method, Voronoi Interface Method, Quadtree and Octree, Deep Learning, Machine Learning

1. Introduction

Multiphase flows are ubiquitous in the physical and life sciences, and their simulations are used in a wide array of applications. Examples where simulations are key include the study of cavitation in the naval industry, the understanding of solidification processes of most metals, the analysis of boiling flows in nuclear plants, the discovery of novel surfactant and liquid delivery into the lung, the prediction of porous media flows for oil and gas operations, and even in the arts, where simulations are used to produce believable flows in computer graphics.

The simulation of multiphase flows presents several challenges. First, one must keep track of the interfaces between phases; these interfaces evolve in time and can experience changes in topology. Second, the large variations in some key physical quantities, say the fluid's density, velocity or the pressure, can only be represented as jump conditions at the mesoscopic and macroscopic scales so that numerical methods must enforce such a representation. Third, the multiscale nature of multiphase flows, where fine features develop locally in some fluid regions, and the limitation of computational resources demand that numerical methods on adaptive grids and parallel architecture be considered.

Early work on the treatment of multiphase flows can be traced back to the immersed boundary method of Peskin [120, 119]. Although multiphase flows per se were not considered, the use of a discrete δ-function to solve blood flows in an immersed elastic interface paved the way to extending to arbitrary domains the discretization of partial differential equations in rectangular domains. Unverdi *et al.* [158] used this formulation in tandem with the front-tracking method to introduce a multiphase incompressible flow solver. In this case, the interface between phases was allowed to undergo large deformations including changes in topology. In [147] and [19], the authors used the

*Corresponding author: fgibou@engineering.ucsb.edu

level-set method to capture the interface motion instead of the front tracking approach, easing the treatment of topological changes. However, even if the front tracking and the level-set methods are sharp as opposed to phase-field models for example, which represent the interface between phase as a numerical mushy zone, the use of a δ-function formulation smears the physical quantities across the interface. In this work, we focus on 'sharp' interface methods, i.e. schemes that numerically preserve the discontinuity in discontinuous quantities.

In what follows, we consider a computational domain $\Omega = \Omega^- \cup \Omega^+$, where Ω^- and Ω^+ are separated by an interface Γ. The fluid's motion is modeled by the incompressible Navier-Stokes equations:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\Delta\mathbf{u} + \mathbf{f} \quad \text{in } \Omega^- \cup \Omega^+, \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \quad \text{in } \Omega^- \cup \Omega^+, \tag{2}$$

where t is the time, $\mathbf{u} = (u, v, w)$ is the velocity field, p is the pressure and $\mathbf{f}$ includes the external forces such as gravity. We consider here the case of a fluid with uniform viscosity μ and uniform density ρ in each subdomain Ω^- and Ω^+. Appropriate boundary conditions are imposed on Γ and on the boundary of the computational domain, $\partial\Omega$. These will be discussed in details in the remaining of this manuscript.

The framework we use to solve those equations is the standard projection approach introduced by Chorin in 1967 [22], which consists of the following three steps. The first step is to compute an intermediate velocity field $\mathbf{u}^*$ using the momentum equation (1) and ignoring the pressure gradient term, e.g. in the case where the temporal derivative is discretized with a forward Euler step:

$$\rho\left(\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + \mathbf{u}^n\cdot\nabla\mathbf{u}^n\right) = \mu\Delta\mathbf{u}^* + \mathbf{f}. \tag{3}$$

The second step is based on the Helmoltz-Hodge decomposition that decomposes a twice continuously differentiable bounded vector field $\mathbf{u}^*$ into a divergence-free component $\mathbf{u}^{n+1}$ and a curl-free component $\nabla\Phi$:

$$\mathbf{u}^* = \mathbf{u}^{n+1} + \nabla\Phi, \quad \text{or} \quad \mathbf{u}^{n+1} = \mathbf{u}^* - \nabla\Phi, \tag{4}$$

where Φ is referred to as the Hodge variable. Imposing the incompressibility constraint on $\mathbf{u}^{n+1}$ gives an equation for Φ:

$$\nabla\cdot\nabla\Phi = \nabla\cdot\mathbf{u}^*. \tag{5}$$

The third step is to solve this Poisson system with appropriate boundary conditions. The boundary conditions and how to impose them will be described throughout this manuscript.

In section 2, we review the level-set and the particle-level-set methods. In section 3, we discuss the level-set method on Quad-/Oc-tree Cartesian grids in parallel as well as how to solve free surface flows. In section 4, we discuss numerical methods related to solving multiphase flows. Section 5 is focused on free surface flows. Section 6 discusses machine and deep learning opportunities for multiphase flows. A short conclusion is given in section 7.

2. Surface Tracking vs. Capturing

Explicit representation of the interface between phases can be achieved with the use of massless particles (see e.g. front tracking [47, 48, 65, 66, 64, 156]). Since the particles' coordinates can be updated with high accuracy, explicit methods provide very accurate interface evolutions. On the other hand, changes in the topology need to be handled explicitly as well, which adds to the complexity of those methods. Implicit representation treats changes in topology in a straightforward manner. In the context of multiphase flows, the most used implicit methods are the Volume of Fluid (VOF) [107, 10, 9, 26, 58] Moment of Fluid [30] and Level-Set methods [110, 109, 138, 139, 42]. Implicit methods represent the interface between phases as an isocontour of a continuous function (the fraction of a cell that is occupied by one phase in the case of VOF or a Lipschitz continuous level-set function in the case of the level-set method). The VOF approach guarantees

mass conservation by construction, but its discontinuous description of the interface makes the computation of geometrical quantities (normal to the interface, interface curvature, etc.) difficult, although we point to the work of Sussman [146] for an attempt to couple the level-set and VOF methods. The level-set representation enables the accurate computation of geometrical quantities but does not ensure mass conservation unless fine grids are used. Two complimentary approaches have been introduced to address this issue: the use of massless particles to complement the level-set representation and the use of adaptive grids. In this review, we focus on the particle level-set method [34] and on the level-set method on adaptive Octree Cartesian grids [83, 84, 94].

2.1. The Level-Set Method

Given a computational domain Ω and an interface Γ separating an interior Ω^- and an exterior Ω^+ sub-domains such that $\Omega = \Omega^- \cup \Omega^+$, the level-set representation describes the different regions with the help of a Lipschitz continuous function ϕ:

$$\Gamma = \{\mathbf{x} \in \Omega : \phi(\mathbf{x}) = 0\}, \quad \Omega^- = \{\mathbf{x} \in \Omega : \phi(\mathbf{x}) < 0\}, \quad \Omega^+ = \{\mathbf{x} \in \Omega : \phi(\mathbf{x}) > 0\}.$$

From this representation, the outward normal $\mathbf{n}$ to the interface and the mean curvature κ are expressed as:

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|}, \quad \kappa = \nabla \cdot \mathbf{n}.$$

The two main equations of the level-set method are that which evolves the level-set function under a given velocity field $\mathbf{u}$:

$$\phi_t + \mathbf{u} \cdot \nabla\phi = 0, \tag{6}$$

and that which transforms an arbitrary function into a signed distance function, i.e. a function satisfying $|\nabla\phi| = 1$. Although this last equation can be solved directly by fast methods (including their parallel counterparts) such as the fast marching method [137, 157, 18] or the fast sweeping method [165, 28, 27], one can also transform an arbitrary level-set function ϕ^0 into a signed distance function by solving:

$$\phi_\tau + \mathtt{Sign}(\phi^0)(|\nabla\phi| - 1) = 0,$$

in pseudo time τ. In the last equation, `Sign` refers to the signum function.

Numerically, since the level-set function is smooth near Γ, one uses central differencing to approximate the normal and the mean curvature. In the general case, equation (6) is approximated with a Godunov scheme using a HJ-WENO approximation of the first-order derivatives [63] and a third-order TVD Runge-Kutta scheme in time [140]. In the case where $\mathbf{u}$ is independent of ϕ, then shocks and rarefaction phenomena do not occur and semi-Lagrangian methods are sometimes preferred because only accuracy constrain the time steps. The reinitialization equation is always solved with a Godunov scheme. It is important to note that one should be careful to explicitly enforce the proper origin of the rarefaction wave when solving the reinitialization equation. Fortunately, this is straightforward to enforce since by definition $\phi = 0$ on Γ. This remark is due to Russo and Smereka [132] and has been further developed in [29] in order to compute second-order accurate curvatures. We refer the interested reader to the textbooks [108, 138] and to the recent review on the level-set method [42] for more details on the method and on recent advances.

2.2. The Particle Level-Set Method

The central idea of the particle-level-set method [34] is to introduce *positive* and *negative* particles, located in the $\phi > 0$ and in the $\phi < 0$ regions, respectively and to use them to correct for the inherent loss of mass in level-set methods. Each particle is given a radius, r_p defined as:

$$r_p = \begin{cases} r_{\mathtt{max}} & \mathtt{if}\ \ s_p\phi(\boldsymbol{x}_p) > r_{\mathtt{max}} \\ s_p\phi(\boldsymbol{x}_p) & \mathtt{if}\ \ r_{\mathtt{min}} \le s_p\phi(\boldsymbol{x}_p) \le r_{\mathtt{max}} \\ r_{\mathtt{min}} & \mathtt{if}\ \ s_p\phi(\boldsymbol{x}_p) < r_{\mathtt{max}} \end{cases}, \tag{7}$$

where s_p is +1 for positive particles and -1 for negative particles, $r_{\mathtt{min}} = .1\min(\Delta x, \Delta y, \Delta z)$ and $r_{\mathtt{max}} = .5\min(\Delta x, \Delta y, \Delta z)$. Their dynamics is described by $\dfrac{d\mathbf{x}}{dt} = \mathbf{u}$, which is integrated forward

in time (typically using a 3rd-order TVD Runge-Kutta scheme) using multilinear interpolation to define $\mathbf{u}$ at the correct locations.

Once the level-set function and the particles are evolved for one time step, the escaped particles (e.g. positive particles in the $\phi < 0$ region) are used to correct any errors in the level-set function. Specifically, for each particle p, one associates a spherical level-set function with radius r_p:

$$\phi_p(\boldsymbol{x}) = s_p(r_p - |\boldsymbol{x} - \boldsymbol{x}_p|). \tag{8}$$

Any variation of ϕ from ϕ_p at any of the eight corners of the current computational cell indicates possible errors in the level set solution. The correction proceeds by changing ϕ to the smallest absolute value of the ϕ_p's and itself.

Several implementation details are associated with the particle level-set method. Typically, one uses 16 particles per grid cell in two spatial dimensions and 32 in three spatial dimensions. Also, in order to retain a smooth interface, escaped particles which are more than 1.5 times their radius removed from the appropriate side of the interface are deleted. An example is given in figure 1 and we refer the interested reader to [34, 35] for more details on the particle level-set method.

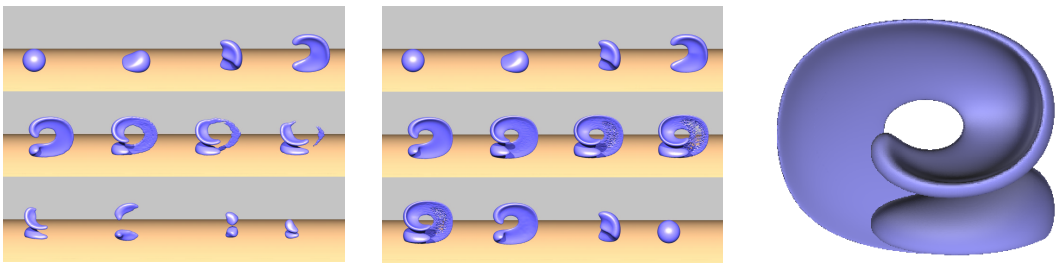


Figure 1: *Example of improved mass conservation between the level-set method (left) and the particle-level-set method (middle) on uniform grids (from [34]) and the level-set method on Octree grids (from [94]) in the case of the standard Enright test [34] (right).*

3. Adaptive Cartesian Oc-/Quad-tree Grids

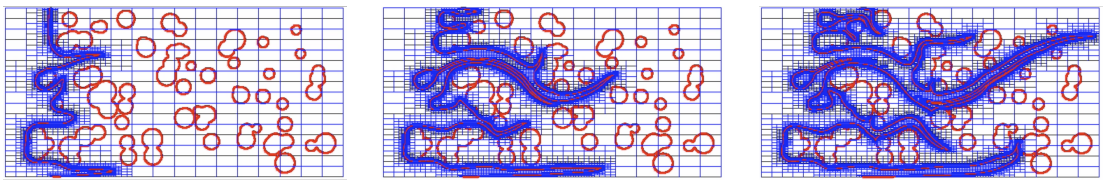


Figure 2: *Quadtree Cartesian grid adapted to capture the interface between two fluids in the case of two-phase flows in porous media.*

Adaptive grids are desired in problems where different length scales occur. This is the case with fluids in general (turbulence requires a grid size proportional to $Re^{9/4}$, with Re the Reynolds number) and multiphase flows in particular since the interface between phases can stretch into thin features. An example is given in figure 2, where a lower-viscosity fluid pushes another fluid with higher viscosity, inducing viscous fingering.

Adaptive grids based on Quad-/Oc-tree data structures are particularly convenient because they enable a continuous change in refinement. In addition, in the context of a capturing interface approach such as the the level-set method, the construction of the grid is straightforward, fast and thus can be adapted at every time step with little computational effort. Adaptivity is also effective

for combating mass loss in the level-set method since increased resolution can be used near the propagating front without imposing that level of refinement in the entire computational domain. Figure 1 (right) gives an example of the Enright test in three spatial dimensions, illustrating that the loss of mass is negligible and that thin sheets can be considered [94].

Discretizing the Navier-Stokes equations on adaptive Cartesian grids instead of uniform grids adds some complexity due to the presence of T-junction grid points, i.e. grid points for which there is a missing a neighbor in one of the Cartesian directions. In addition, in order to guarantee numerical stability it is necessary to define the discretizations of the gradient and divergence operator in such a way that they preserve the following relationship at the discrete level: $\nabla = -(\nabla\cdot)^T$. The standard MAC grid arrangement provides a straightforward way to guarantee such constraint, although some other work have considered solving the Navier-Stokes equations in a node-based fashion [92].

For example, Popinet considered Octree grids to discretize the Navier-Stokes equations [121] by introducing a discretization of the pressure equation (5) that leads to a nonsymmetric linear system. Losasso *et al.* introduced a symmetric discretization of the Poisson equation and applied it to the inviscid Navier-Stokes equations [84]. An example of that simulation is given in figure 3, where the Navier-Stokes equations are used to advect a scalar quantity (smoke density) that can be rendered and are also applied to the case of free surface flows. In this work, only the numerical viscosity is used to mimic viscous effects. Later, Losasso *et al.* [83] introduced a Poisson solver on Octrees that produces second-order accurate solutions while preserving the symmetry positive definiteness of the linear system. Finally, Guittet *et al.* [52] used that solver for solving the Navier-Stokes equations and introduced an approach to consider the viscous stress tensor based on Voronoi tessellations.

3.1. First-order accurate Poisson Solver

Consider a general cell configuration where a computational cell is adjacent to smaller computational cells. A finite volume discretization of the pressure equation (5) leads to:

$$\sum_f (\nabla\Phi\cdot\mathbf{n})\,A_f = \sum_f (\mathbf{u}^*\cdot\mathbf{n})\,A_f,$$

where the index f refers to the faces of the current computational cell and A_f refers to their area. It is natural to define the components of $\nabla\Phi$ at the faces, for example one needs to define Φ_x at the same location as u^*. In [84], the authors used the notion that first-order perturbation in the location of the variable leads to a convergent solution of the Poisson equation [43] to propose a simple formula for $\nabla\Phi$. For example, consider a large computational cell (indexed by L) on the right of a few smaller cells (indexed by s), then x-derivative of Φ for a small cell is given by:

$$(\Phi_x)_s = \frac{\Phi_L - \Phi_s}{\Delta_s},$$

where Δ_s is the distance in the x-direction between the center of the large cell L and the center of the small cell s. All the components of $\nabla\Phi$ are built similarly for all computational cells. This discretization produces a symmetric positive-definite linear system that can be solved efficiently using a standard PCG solver [136]. Combining this solver with the particle-level-set method, [84] obtained realistic simulation of smoke and free surface flows (see figure 3).

3.2. Second-order accurate Poisson Solver

The discretization above produces first-order accurate solution only for the Hodge variable Φ. In addition, this discretization defines a different pressure gradient on each cell face. In [83], the authors take a different approach and construct the discretization of Φ_x on all the adjacent cells of a large cell to be equal to the discretization of the large cell. One thus has:

$$(\Phi_x)_L = \sum_s \frac{\Phi_L - \Phi_s}{\Delta_s}\frac{A_s}{A_L},$$

In addition, in order to make the linear system symmetric, Δ_s is replaced by $\sum_s \frac{A_s}{A_L}\Delta_s$.

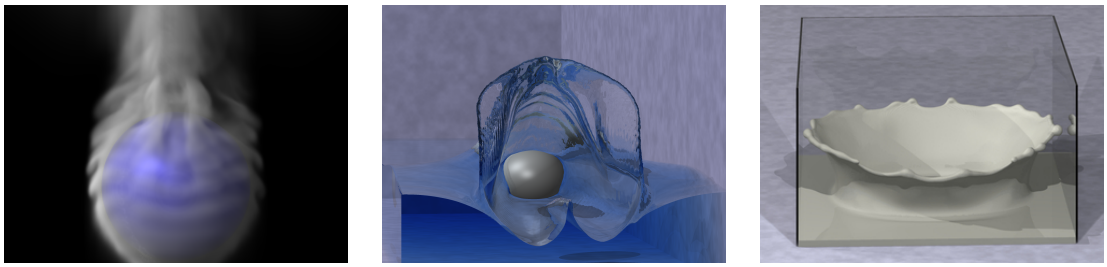


Figure 3: *Simulation of smoke (left) and free surface flows (center and right) using Octree Cartesian grids [84].*

Table 1: Accuracay of the Poisson equation solver of section 3.2 on an Octree example.

L^1 error	order	L^∞ error	order
3.241×10^{-3}	–	6.990×10^{-3}	–
7.219×10^{-4}	2.17	1.596×10^{-3}	2.13
1.700×10^{-4}	2.08	3.841×10^{-4}	2.06
4.121×10^{-5}	2.04	9.460×10^{-5}	2.02
1.014×10^{-5}	2.02	2.354×10^{-5}	2.01

Numerical examples given in [83] show that the order of accuracy is second-order (see e.g. table 1). Later, [54] gave a proof that the order of accuracy is indeed second-order, showing that the increase in accuracy comes from the cancellation of the errors in the transversal direction, reminiscent to the construction of higher order accurate definition of T-junction values in [96, 94].

3.3. Implicit Viscosity on Octrees

In the case of adaptive grids, it is convenient to use a finite volume approach on Voronoi tesselations to discretize implicitly the viscous stress tensor because in a MAC grid arrangement, the components of the velocity field are staggered. In [57] the authors introduced an implicit second-order accurate Voronoi solver for the Vlasov-Poisson and Vlasov-Maxwell equations. This solver was used in [32, 33] to compute a divergence-free fluid's velocity in the context of Chimera grids. In [54], Guittet *et al.* introduced an implicit discretization of the stress tensor in the context of Quad-/Oc-tree Cartesian grids using the `Voro++` library [134] in three dimensions. An example of a Voronoi partition on a Quadtree grid is given in figure 4; in this case, the discretization for one of the x-component, u_0, of the velocity field is:

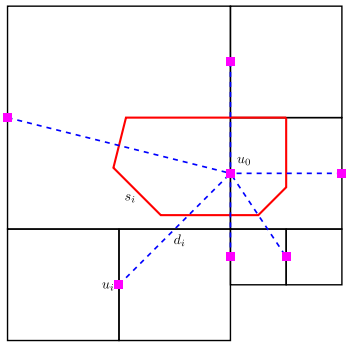


Figure 4: *Example of a Voronoi cell for a horizontal velocity data point u_0.*

$$\int_{\mathcal{C}} \mu \Delta \mathbf{u} = \mu \int_{\partial \mathcal{C}} \nabla \mathbf{u} \cdot \mathbf{n} = \mu \sum_{i \in \mathrm{Voro}(u_0)} s_i \frac{u_i - u_0}{d_i}, \tag{9}$$

where $\mathrm{Voro}(u_0)$ is the set of neighbors of u_0 involved in its Voronoi cell, d_i is the distance between the location of u_0 and u_i, and s_i is the length of the edge of the Voronoi cell orthogonal to the segment $[u_0, u_i]$. Applying this discretization to all degrees of freedom produces a symmetric positive-definite linear system and second-order accurate solutions (see table 2 considering an exact solution $u(x, y) = \cos(x)\sin(y)$). A geometric approach is used to compute integrals with second-order accuracy [93, 95].

Table 2: Convergence of the Voronoi diagram-based Poisson solver on an arbitrary grid. $l_{\min}$ and $l_{\max}$ are the minimum and maximum number of refinements performed on grid cells.

$(l_{\min}, l_{\max})$	error	order
(1, 8)	$3.21 \cdot 10^{-2}$	-
(2, 9)	$1.76 \cdot 10^{-2}$	0.87
(3, 10)	$5.60 \cdot 10^{-3}$	1.65
(4, 11)	$1.53 \cdot 10^{-3}$	1.87
(5, 12)	$3.97 \cdot 10^{-4}$	1.95

3.4. Parallel Octrees

In [99], Mirzadeh *et al.* introduced a computational framework for solving free boundary problems on a forest of Octrees. Instead of considering a single Octree grid, they consider an arbitrary number of Octree grids that are distributed among the available processors. In order to ensure load balancing, the library `p4est` is used to uniformly distribute the degrees of freedom on the set of processors. Specifically, a Z-ordering is used to order and group the grid points that are contiguous; the resulting array is then distributed straightforwardly. A second algorithm is employed to construct local Octree grids on each processor, ensuring that the leaves of each local Octree matches the local elements of the array. Subsequent discretizations on Octrees is then done locally on each processor, and ghost padding is used to exchange data between processors.

This framework has been used in Guittet *et al.* [51] to simulate single phase fluid flows on a forest of Octrees. A snapshot of the simulation of a flow past a sphere computed on 1024 cores is given in figure 5, depicting the $Q = 0.006$ isocontour of the Q-criterion [61]. In this simulation, the Reynolds number is $Re = 300$, and the Octrees have levels (6, 9), which translates into 6 million leaves. The authors also demonstrated parallel scaling on up to 4000 cores.

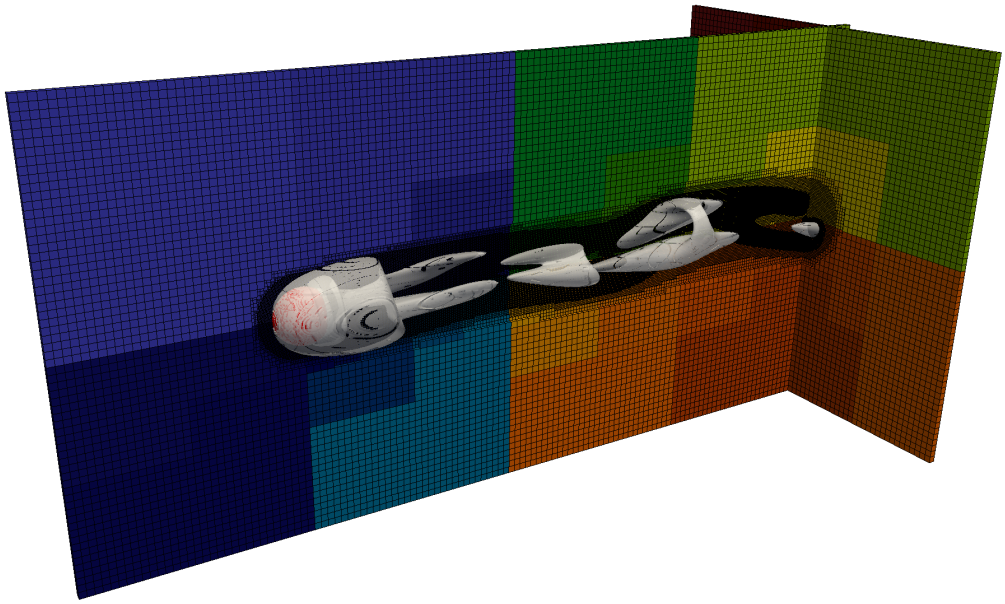


Figure 5: *Simulation of a flow past a sphere at time $t = 221$ seconds. The different colors correspond to different processor ranks.*

4. Multiphase Flows

4.1. Poisson Solver with Jumps

Solutions to multiphase flow problems admit jumps across the interface between phases. In particular, the pressure jump is related to surface tension forces. In the context of the projection method outlined in section 1, this boundary condition occurs when solving for the Hodge variable Φ, i.e. one needs to solve a problem of the form:

$$\begin{cases} \nabla \cdot (\beta \nabla \Phi) &= f & \text{in } \Omega^- \cup \Omega^+ \\ [\Phi] &= g_\Gamma & \text{on } \Gamma \\ [\beta \nabla \Phi \cdot \boldsymbol{n}] &= h_\Gamma & \text{on } \Gamma \end{cases}, \tag{10}$$

where $\beta = \dfrac{1}{\rho} > 0$ is taken to be constant in each subdomain but experiences a discontinuity across the interface. The functions $f \in L^2(\Omega)$, g_Γ, h_Γ and k are given. The jump in a quantity q across the interface Γ is defined as $[q] = q_\Gamma^+ - q_\Gamma^-$. In what follows, we describe the Ghost-Fluid Method and its extension, the Voronoi Interface Method, for solving the above system of equations.

4.1.1. The Ghost-Fluid Method

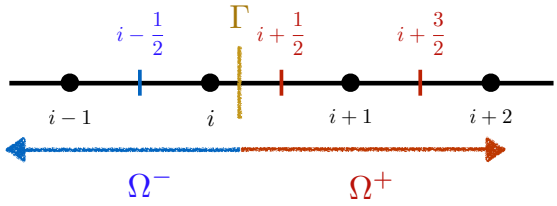


Figure 6: *Local grid near an interface* Γ.

Consider the one-dimensional case first, where one seeks to discretize the following equation at a grid point x_i:

$$\begin{cases} (\beta \Phi_x)_x &= f & \text{in } \Omega^- \cup \Omega^+ \\ [\Phi] &= g_\Gamma & \text{on } \Gamma \\ [\beta \Phi_x] &= h_\Gamma & \text{on } \Gamma \end{cases}. \tag{11}$$

The Ghost-Fluid Method for the Poisson equation with jumps [80] introduces a virtual solution across the interface in such as way as to reflect the jump conditions, while avoiding $O(1/\Delta x)$ errors when approximating derivatives. Specifically, the standard central differencing formula:

$$\frac{(\Phi_x)^{+R}_{i+\frac{1}{2}} - (\Phi_x)^{-R}_{i-\frac{1}{2}}}{\Delta x} = f_i,$$

where $(q)^{\pm R}$ refers to a quantity q in the $\Omega^\pm$ region of the real fluid, is modified as:

$$\frac{(\Phi_x)^{-G}_{i+\frac{1}{2}} - (\Phi_x)^{-R}_{i-\frac{1}{2}}}{\Delta x} = f_i,$$

where $(q)^{\pm G}$ refers to a quantity q in the $\Omega^\pm$ region of the ghost fluid. The jump condition in $[\Phi_x]$ is then used to write:

$$\frac{(\Phi_x)^{+R}_{i+\frac{1}{2}} - h_\Gamma - (\Phi_x)^{-R}_{i-\frac{1}{2}}}{\Delta x} = f_i \iff \frac{(\Phi_x)^{+R}_{i+\frac{1}{2}} - (\Phi_x)^{-R}_{i-\frac{1}{2}}}{\Delta x} = f_i + \Delta x\, h_\Gamma.$$

This simple transformation imposes the jump in Φ_x in a sharp manner and only affect the right-hand side of the linear system. The same procedure is used to account for the jump in Φ, giving the final discretization at x_i:

$$\frac{1}{\Delta x}\left(\frac{\Phi_{i+1}^{+R}-\Phi_i^{+R}}{\Delta x}-\frac{\Phi_i^{+R}-\Phi_{i-1}^{+R}}{\Delta x}\right)=f_i+\Delta x\, h_\Gamma+\Delta x^2\, g_\Gamma. \tag{12}$$

This methodology produces a symmetric positive-definite linear system of equations, where the jump conditions only influence the right-hand side of the system. We refer the interested reader to [80] for more details.

In two spatial dimensions, the original system of equations (10) can be altered as follows:

$$\begin{cases} (\beta\Phi_x)_x+(\beta\Phi_y)_y &= f & \text{in } \Omega^-\cup\Omega^+\\ [\Phi] &= g_\Gamma & \text{on } \Gamma\\ [\beta\Phi_x] &= h_\Gamma\, n_1 & \text{on } \Gamma\\ [\beta\Phi_y] &= h_\Gamma\, n_2 & \text{on } \Gamma \end{cases},$$

where n_1 and n_2 are the x- and y- components of the normal vector $\boldsymbol{n}$. Although this formulation is different from the original problem, it provides a simple extension of the one-dimensional case and still preserves the jumps in the normal direction to the interface. In fact, Liu and Sideris proved that the solution is first-order accurate [82].

4.1.2. The Voronoi Interface Method

The Ghost-Fluid method for the Poisson equation ignores the jumps in the tangential direction, which lead to solution gradients that do not converge in general. Guittet *et al.* [52] solved that problem by modifying the grid local to the interface in order to align the degrees of freedom with the line orthogonal to the fluxes. Then, the discretization can use the Ghost-Fluid idea in one dimension. Specifically, the degrees of freedom adjacent to the interface are deleted and their projection onto Γ determined. Two additional degrees of freedom are then constructed on each side of the interface in its normal direction and a Voronoi partition is then constructed with the entire set of degrees of freedom (since the grid is Cartesian, only a small band near the interface is changed). Figure 7 illustrates this process.

The system of equations (10) is then discretized at each degree of freedom i with a finite volume approach:

$$\int_{\mathcal{C}}\nabla\cdot(\beta\nabla\Phi)\,dV=\int_{\partial\mathcal{C}}(\beta\nabla\Phi)\cdot\boldsymbol{n}_{\mathcal{C}}\,dl\approx\sum_j s_{ij}\beta_i\frac{u_{ij}-u_i}{d_{ij}/2},$$

where $\mathcal{C}$ is the control volume around i and Φ_{ij} is the value of Φ midway between i and j. Since by construction the edge of the Voronoi cell conforms to the interface, one can consider that $\Phi_{ij}=\Phi_\Gamma$. Using the idea of the Ghost-Fluid Method in one dimension, each neighboring point j of a grid point i contributes to the left-hand side of the linear system an amount:

$$\tilde{\beta}_{ij}s_{ij}\frac{u_j-u_i}{d_{ij}},$$

and to its right-hand side:

$$\tilde{\beta}_{ij}\frac{s_{ij}}{d_{ij}}\left(-[\Phi]+\frac{d_{ij}}{2\beta_j}\,[\beta\nabla\Phi\cdot\boldsymbol{n}]\right),$$

where $\tilde{\beta}_{ij}=\dfrac{2\beta_i\beta_j}{\beta_i+\beta_j}$ is the harmonic mean between β_i and β_j.

As is the case for the Ghost-Fluid Method, the solution in each phase is decoupled from the other, which enables the computation of solutions when the jumps in β, Φ and $\nabla\Phi\cdot\boldsymbol{n}$ are arbitrarily

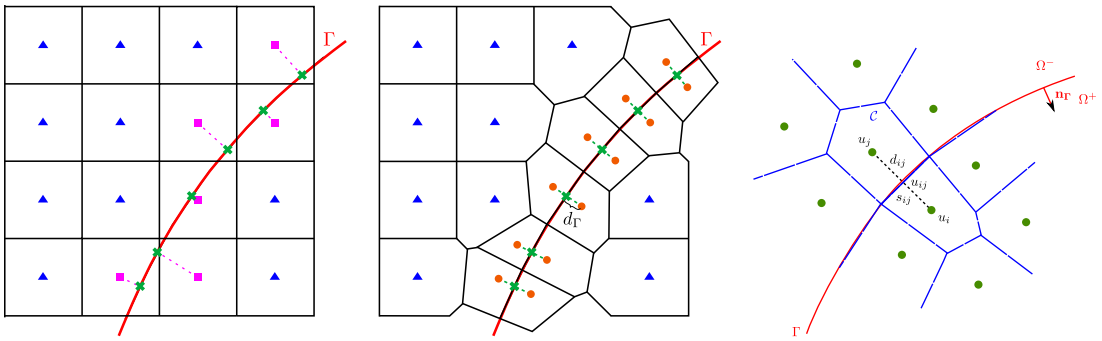


Figure 7: *Procedure to build a Voronoi mesh near the interface: (1) project of the degrees of freedom adjacent to the interface onto* Γ *(left); (2) Introduce two additional degrees of freedom on each side of* Γ *at an arbitrary distance in the normal direction to the interface and construct the Voronoi tessellation with the current degrees of freedom (center). Right: a local control volume* $\mathcal{C}$*, with* s_{ij} *the length of the edge of* $\mathcal{C}$ *conforming to* Γ *and* d_{ij} *the distance between* Φ_i *and* Φ_j*.*

large. Figure 8 illustrates a typical solution to (10) and table 3 gives the order of accuracy of the method in the L^∞-norm(from [52]). This example considers an exact solution of:

$$\Phi(x,y) = \begin{cases} e^x + 1.3 & \text{if } (x,y) \in \Omega^0 = \left\{(x,y),\ \sqrt{x^2+y^2} - 0.2 \le 0\right\}, \\ \cos(y) + 1.8 & \text{if } (x,y) \in \Omega^1 = \left\{(x,y),\ \sqrt{x^2+y^2} - 0.5 + 0.1\cos(5\theta) \le 0\right\}, \\ \sin(x) + 0.5 & \text{if } (x,y) \in \Omega^2 = \left\{(x,y),\ \sqrt{x^2+y^2} - 0.8 \le 0\right\}, \\ -x + \ln(y+2) & \text{otherwise,} \end{cases}$$

where θ is the angle between (x,y) and the x-axis. The coefficient β is taken to be:

$$\beta(x,y) = \begin{cases} y^2 + 1 & \text{if } (x,y) \in \Omega^0, \\ e^x & \text{if } (x,y) \in \Omega^1, \\ y + 1 & \text{if } (x,y) \in \Omega^2, \\ x^2 + 1 & \text{otherwise.} \end{cases}$$

The Voronoi Interface Method has also considered harder problems where the jump condition satisfies a partial differential equation on a co-dimension one surface and we refer the interested reader to [53] for details.

Figure 8: *Solution of section 4.1.2 using the VIM (from [52]).*

resolution	solution	order	gradient	order
2^4	$1.00 \cdot 10^{-3}$	–	$3.33 \cdot 10^{-3}$	-
2^5	$2.33 \cdot 10^{-4}$	2.11	$1.01 \cdot 10^{-3}$	1.72
2^6	$6.23 \cdot 10^{-5}$	1.90	$3.46 \cdot 10^{-4}$	1.54
2^7	$1.56 \cdot 10^{-5}$	2.00	$1.59 \cdot 10^{-4}$	1.12
2^8	$4.00 \cdot 10^{-6}$	1.96	$6.82 \cdot 10^{-5}$	1.22
2^9	$1.01 \cdot 10^{-6}$	1.99	$4.00 \cdot 10^{-5}$	0.77
2^{10}	$2.55 \cdot 10^{-7}$	1.99	$2.24 \cdot 10^{-5}$	0.84

Table 3: *Accuracy of the solution of section 4.1.2 and its gradient in the* L^∞*-norm using the VIM (from [52]).*

4.2. Multiple Interfaces

One of the main difficulties when considering multiple fluids (more than two) is that one needs to keep track of multiple interfaces. If one considers a different level-set function for each fluid and evolves that level-set function according to that fluid's velocity field, numerical errors may introduce invalid regions. This is illustrated in figure 9 where overlaps and/or vacuums are created. Additional methods, called projections, must be introduced to resolve those scenarios [89, 133, 142]. An alternative, often used in the computer vision community, is to use the different sign combinations of n level-set functions to represent up to 2^n regions. This representation, however is prone to biasing artifacts.

In [85], Losasso *et al.* solved that problem by introducing a projection method that removes overlaps and vacuums, preserves the property of signed distance functions, and preserves the exact interface locations. The strategy is to consider n level-set functions to keep track of n distinct fluids and to evolve the level-set functions according to their velocity fields. The level-set functions are then updated by a simple rule: consider the case of two level-set functions ϕ_1 and ϕ_2 (see figure 10) that have been updated according to their respective velocity fields, the projection step is simply to subtract the average of ϕ_1 and ϕ_2, point-wise. In the case of multiple level-set functions ϕ_i, the projection amounts to subtracting the average of the smallest two level-set functions from all the ϕ_i's. An example of the simulation of the Rayleigh-Taylor instability using this technique as its backbone is given in figure 11.

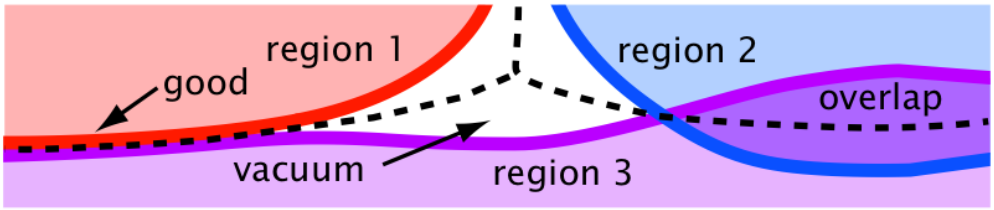


Figure 9: *Vacuums are regions where the level-set functions are all positive. Overlaps are regions where at least two level-set functions are negative.*

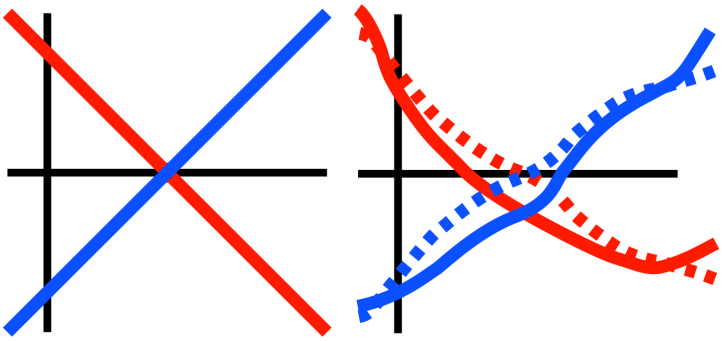


Figure 10: *Projection of [85] for multiple interfaces. Left: original level-set functions. Right: level-set functions after update (solid lines) and after the projection step (dashed lines).*

4.3. Surface Tension

Surface tension forces are responsible for important physical phenomena like drop breakups or Marangoni effects. Numerically, explicit treatments lead to the standard $O(\Delta x^{\frac{3}{2}})$ time step restriction [14]. Implicit methods have been proposed to reduce the time step restriction. For

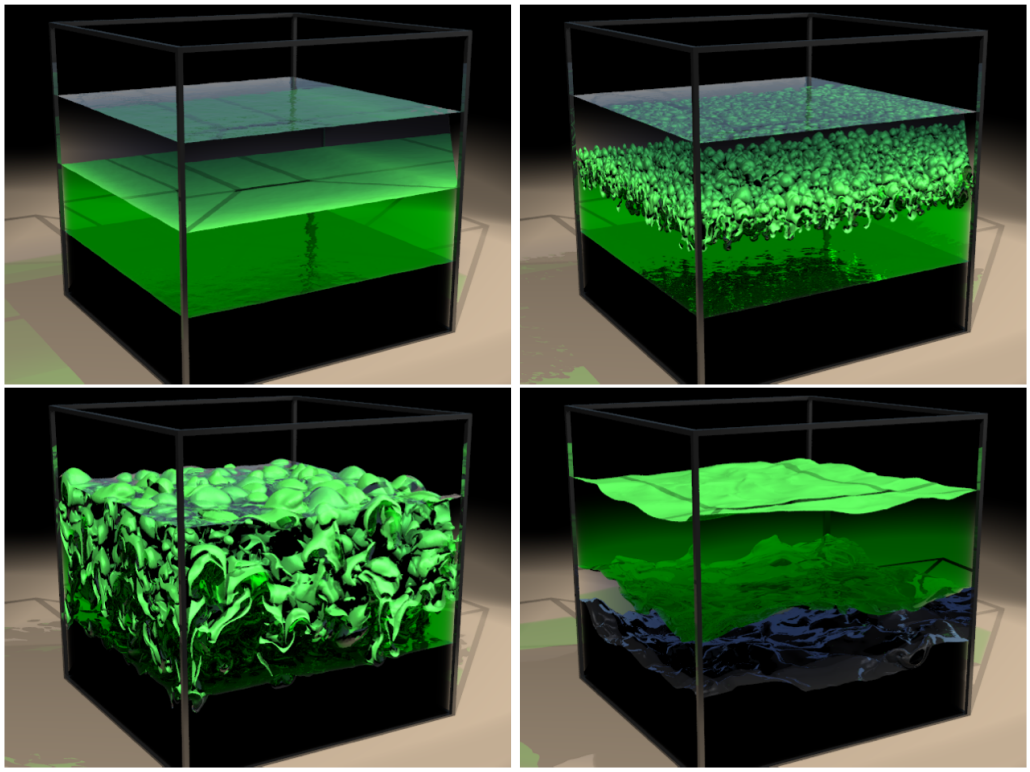


Figure 11: *Rayleigh-Taylor instability (from [85])*

example [59] focused on the changes in curvature over time and [62] simplified this approach by the use of the Laplace-Beltrami operator (see also [123] for a finite volume implementation). Observing that the Laplace-Beltrami can be decomposed as the sum of the Laplace operator and a nonlinear term, [141, 164, 166] proposed a semi-implicit discretization by discretizing the Laplace operator implicitly and the nonlinear term explicitly. Other works in the Volume of Fluid community have been proposed to improve the stability of surface tension methods [145, 78, 144].

In [167], Zheng *et al.* introduced a hybrid particle/grid method that allows for a fully implicit discretization, lifting the time step restriction. The surface is approximated by a triangular mesh and the surface tension force is computed as the sum over the boundary segments on each element. The force derivatives can then be computed over each element and composed into a symmetric matrix, $\mathbf{D}$, that can be made negative semi-definite and written as $\mathbf{D} = -\mathbf{C}^T\mathbf{C}$ with $\mathbf{C}$ related to the surface tension coefficient σ (see [167] for details).

The temporal discretization of the Navier-Stokes equations is written as:

$$\begin{aligned} \hat{\mathbf{M}}^{n+1}(\hat{\mathbf{u}}^{n+1} - \hat{\mathbf{u}}^n) + \hat{\mathbf{G}}^{n+1} - \Delta t\hat{\mathbf{f}}^{n+1} - \Delta t\hat{\mathbf{M}}^{n+1}\hat{\mathbf{g}} &= \mathbf{0}, \\ \left(\hat{\mathbf{G}}^{n+1}\right)^T \hat{\mathbf{u}}^{n+1} &= \mathbf{0}, \end{aligned}$$

where $\hat{\mathbf{G}}$ is the volume-weighted gradient matrix, $\hat{\mathbf{M}}$ is the mass matrix with the mass of each particle defined as the density-weighted control volume, $\hat{\mathbf{p}}$ is the vector of Δt-weighted pressures, $\hat{\mathbf{f}}$ is the vector of Δt-weighted surface tension forces and $\hat{\mathbf{g}}$ is the vector of Δt-weighted gravity. This nonlinear system of equations can be written as:

$$\mathbf{F}(\mathbf{w}) = \mathbf{0}, \quad \text{with} \quad \mathbf{w} = (\hat{\mathbf{u}}^{n+1}, \hat{\mathbf{p}}^{n+1}).$$

The Newton-Raphson method is used to solve this nonlinear system:

$$\begin{aligned} \mathbf{J}_k \boldsymbol{\Delta}\mathbf{w} &= -\mathbf{F}(\mathbf{w}_k), \\ \mathbf{w}_{k+1} &= \mathbf{w}_k + \boldsymbol{\Delta}\mathbf{w}, \end{aligned}$$

where $\mathbf{J}$ is the Jacobian of $\mathbf{F}$. In order to compute an approximate Jacobian, one treats $\hat{\mathbf{M}}$ and $\hat{\mathbf{G}}$ as constant at each iteration. The derivative of $\hat{\mathbf{f}}^{n+1}$ with respect to $\hat{\mathbf{u}}^{n+1}$ is $-\Delta t^2 \mathbf{D}$ so that the approximate Jacobian is:

$$\mathbf{J}_k = \begin{pmatrix} \hat{\mathbf{M}}_k - \Delta t^2 \mathbf{D}_k & \hat{\mathbf{G}}_k \\ \hat{\mathbf{G}}_k^T & \mathbf{0} \end{pmatrix}.$$

Finally, one is left with solving a symmetric positive-definite linear system of equations:

$$(\mathbf{P} + \mathbf{K}_k \hat{\mathbf{M}}_k^{-1} \mathbf{K}_k^T)\hat{\mathbf{z}} = \mathbf{K}_k \hat{\mathbf{u}}^* + \begin{pmatrix} \mathbf{G}_k^T \hat{\mathbf{u}}_k \\ \mathbf{0} \end{pmatrix}, \tag{13}$$

with

$$\mathbf{P} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \qquad \mathbf{K}_k = \begin{pmatrix} \mathbf{G}_k^T \\ \Delta t \mathbf{C}_k \end{pmatrix} \qquad \hat{\mathbf{z}} = \begin{pmatrix} \hat{\mathbf{p}}_{k+1} \\ \Delta t \mathbf{C}_k (\hat{\mathbf{u}}_{k+1} - \hat{\mathbf{u}}_k) \end{pmatrix}$$

from which the particles' locations are updated:

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \Delta t \hat{\mathbf{u}}_{k+1}.$$

In addition, [167] adds a constraint on the volume, hence proposing a volume-conserving method that can handle surface tension implicitly. This is achieved by replacing equation (13) by:

$$(\mathbf{P} + \mathbf{K}_k \hat{\mathbf{M}}_k^{-1} \mathbf{K}_k^T)\hat{\mathbf{z}} = \mathbf{K}_k \hat{\mathbf{u}}^* + \begin{pmatrix} \frac{1}{\Delta t}(\hat{v}_k - \hat{v}^n) \\ 0 \end{pmatrix},$$

enforcing the conservation of volume discretely.

Examples of implicitly treating the surface tension force is given in figure 12, simulating drop oscillation. In particular, the time step is taken large enough that the steady state is obtained in a single time step. Figure 13 illustrates the simulation of drop breakup into several satellite droplets.

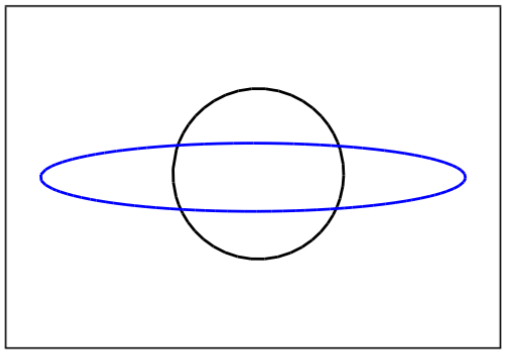
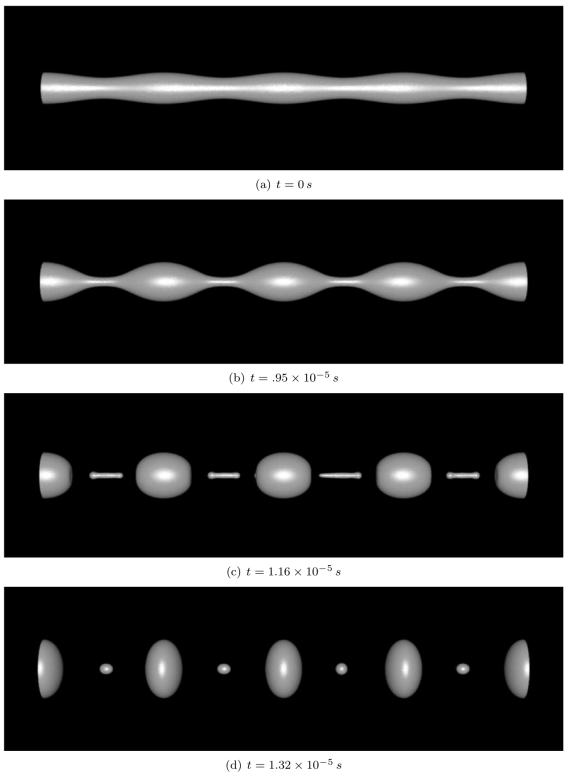


Figure 12: *Original drop (blue) deforming under surface tension after a single time step (black) (from [167]).*

4.4. Monolithic Mass Tracking for Bubbles Dynamics

One of the difficulties when considering bubbles in a fluid is that naive numerical treatments can lead to bubble collapsing. Methods to combat this artificial collapse have considered computing the pressure inside the bubbles by keeping track of their enclosed volume and using a given equation of state [143]. However, tracking volume is delicate as the volume of cloud of bubbles can change



(a) $t = 0\,s$

(b) $t = .95 \times 10^{-5}\,s$

(c) $t = 1.16 \times 10^{-5}\,s$

(d) $t = 1.32 \times 10^{-5}\,s$

Figure 13: *Column breaking into droplets (from [167])*

drastically throughout the simulation due to changes in topology. In [1], Aanjaneya *et al.* proposed instead to keep track of the mass, which removes the aforementioned difficulties. Specifically, they define the mass inside each bubble and advect it with the air velocity using the conservative scheme of [73], making sure that the mass is enclosed within the zero-level set following [72]. In addition, starting from the coupled incompressible/compressible fluid solver of Caiden *et al.* [17], they develop a monolithic solver for the pressure that is similar to that developed for fluid-solid coupling [129, 50, 44] and which avoids the standard stability issues of partitioned methods. This solver fully couples the degree of freedom for the pressure inside the bubble (assumed constant) and the pressure in the incompressible fluid. The resulting pressure is then used in a standard projection method to update the fluid's velocity field. The velocity field in the bubble is updated using a second Poisson solver with Neumann boundary conditions in order to ensure that the bubble velocity is consistent with the fluid's velocity. This treatment allows for the bubbles to react to the compression and expansion forces induced by the fluid.

The authors in [1] also include the effect of surface tension forces and show that this approach produces stable simulations. An example from [1] is given in figure 14, where a single bubble rises in an ambient fluid and interacts with a network of spherical solid objects, forcing it to split into several smaller bubbles.

5. Free Surface Flows

Free surface flows are considered in applications where the effect of the momentum of the gas is negligible. In this case, the step that has the largest impact on the accuracy is the pressure boundary treatment at the free surface. This boundary condition is applied when solving for Φ in

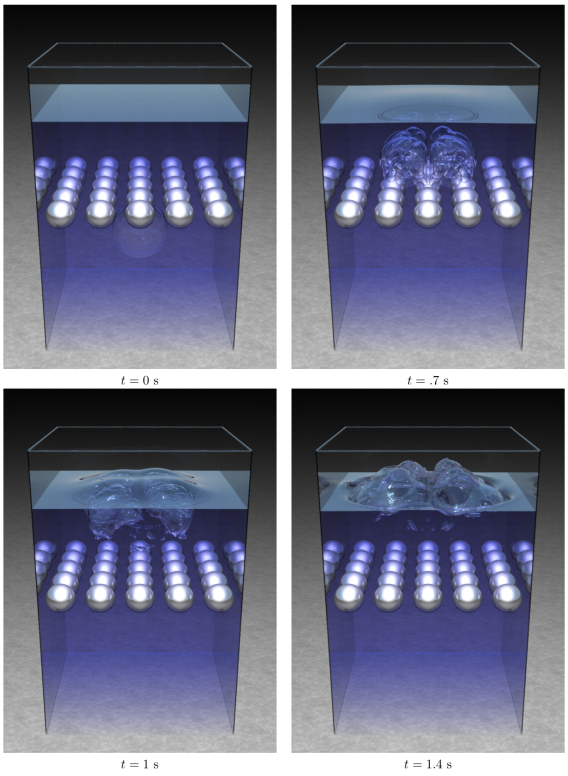


Figure 14: *Air bubble rising over a network of spheres (from [1]).*

equation 5, i.e. one solves the system:

$$\begin{array}{rcll} \nabla \cdot \nabla \Phi & = & \nabla \cdot \mathbf{u}^* & \text{in } \Omega^-, \\ \Phi & = & \sigma\kappa & \text{on } \Gamma. \end{array} \tag{14}$$

and controls the effect of surface tension through the boundary condition $\Phi_\Gamma = \sigma\kappa$. In [43], a second-order accurate discretization of the Poisson equation with Dirichlet boundary condition was introduced and later applied to the case of free surface flows [36].

The method introduced in [43] modifies the standard implicit central differencing scheme for grid points near the free surface. Since it uses a dimension by dimension approach, we describe only the treatment in the case of the one-dimensional constant coefficient Poisson equation, $\Phi_{xx} = u^*_x$, noting that extension to the variable coefficient case is straightforward. Suppose that an interior grid point x_i is adjacent to the interface, located at x_Γ with $x_i < x_\Gamma < x_{i+1}$. In this case the boundary condition $\Phi = \Phi_\Gamma$ is imposed by defining a ghost value for Φ^G_{i+1} and considering the following modified discretization

$$\frac{p^G_{i+1} - 2p_i + p_{i-1}}{(\Delta x)^2} = (u^*_x)_i$$

to fill the corresponding row in the linear system of equations for Φ. The definition of Φ^G_{i+1} is derived by extrapolation using Φ_Γ as one of the data points. The nature of the resulting linear system and the accuracy of the solution depends on the degree of extrapolation [40, 102]: for a constant or linear extrapolation the corresponding linear system is symmetric positive-definite and non-symmetric (but diagonally dominant) otherwise. One obtains first-order accuracy in the solution for a constant extrapolation and an additional order for each additional degree of extrapolation. In

[36], the authors used a linear extrapolation to define Φ_{i+1}^G:

$$\Phi_{i+1}^G = \frac{\Phi_\Gamma + (\theta - 1)\,\Phi_i}{\theta}, \text{ with } \theta = \frac{x_\Gamma - x_i}{\Delta x}.$$

The discretization at x_i becomes:

$$\frac{\left(\dfrac{\Phi_\Gamma - \Phi_i}{\theta \Delta x}\right) - \left(\dfrac{\Phi_i - \Phi_{i-1}}{\Delta x}\right)}{\Delta x} = (u_x^*)_i.$$

Since the linear system is symmetric positive-definite, [36] used a PCG method with a modified incomplete Cholesky preconditioner [136] (see also [45] for a simple parallel implementation of the Poisson solver in irregular domains). These authors demonstrated the impact of the pressure treatment on the amplitude of an oscillating bubble and showed that the pressure treatment is paramount to the accuracy of a free surface solver.

Since the discretization of Φ is implicit, the boundary condition $\Phi = \sigma\kappa$ must be imposed at time t^{n+1}, which means that the level-set function ϕ^{n+1} is used when computing the mean curvature κ. The motion of the level-set function is given by the fluid velocity field at time t^n and in order to evolve the level-set function it is necessary to define a valid value for that velocity in a band near the interface. Since the velocity field is not known outside the fluid region, it is extrapolated constantly in the normal direction to the interface using a few iterations of the following equation:

$$\frac{\partial \boldsymbol{u}_{\text{ext}}}{\partial \tau} + \boldsymbol{n} \cdot \nabla \boldsymbol{u}_{\text{ext}} = 0,$$

where τ is a fictitious time. Fast solvers can be used to extend the velocity field in the air region by solving $\nabla\phi \cdot \nabla \mathbf{u}_{\text{ext}} = 0$ [4, 7]

The treatment of the free boundary pressure condition has been combined with the particle level-set method and other techniques to simulate spray and foam (see e.g. [86] and the references therein) and has had a large impact in the graphics community for special effects in the Hollywood industry, and in fact led to an Academy Award from the Motion Picture Association for fluid simulation (cyber fluid) in 2008. An example of such a realistic simulation is provided in figure 15.

We also note that the Dirichlet boundary condition on a free surface is also one of the central pieces for solving the Stefan problem [41, 42, 40, 99, 114, 115, 128, 20, 21] as well as for solving for the conservation of energy in boiling-type flows [39, 74, 149, 150, 151, 159]. We refer the interested reader to those references for a discussion on numerical methods for the Stefan problem and boiling-type flows.

We also refer the interested reader to [98, 101, 100, 111, 112, 135, 153, 154, 155, 56, 46, 103] for solvers considering Robin/Neumann boundary conditions in irregular domains and some applications in multiphase polymer physics, electrokinetics and multiphase solidification.

6. Machine Learning, Deep Learning, and Multiphase Flows

6.1. The Data Explosion

Data is becoming plentiful. Over the past several years, the digital data collectively generated by humans has doubled every two years or less [38]. Some of this data comes from physical sensors, such as those on General Electric's locomotives that generate 150,000 data points each minute[1], or the various cameras and sensors in forthcoming self-driving cars, estimated by Intel to be 4TB of data per car per day[2]. Data also comes from virtual sensors, such as the 600TB+ of data Facebook processes daily[3] from both automated analytics collection and user-supplied data such as messages and photos.

[1] See https://www.cnet.com/news/at-ge-making-the-most-advanced-locomotives-in-history
[2] See https://newsroom.intel.com/editorials/self-driving-cars-big-meaning-behind-one-number-4-terabytes/
[3] See https://code.facebook.com/posts/229861827208629

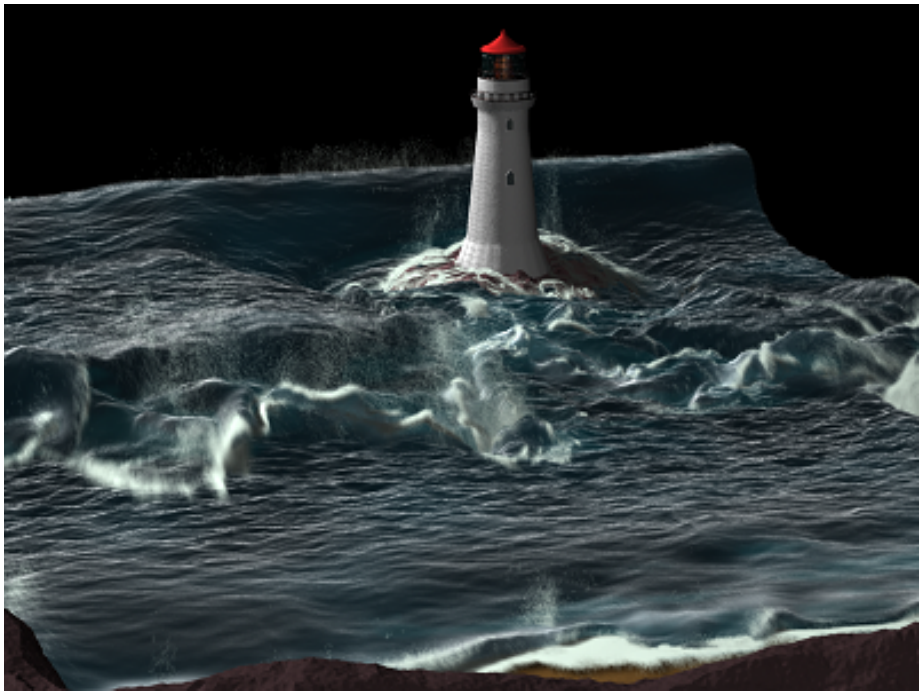


Figure 15: *Realistic simulation of free surface flows [86].*

A major contributor to this trend is the continually improving ways to collect and process data. For example, the ubiquity of smartphones has led to fundamental societal change, but contemporaneously, it has delivered systematic changes in the way data is collected and transmitted. In the smartphone generation, multiple high-resolution digital cameras are carried in every pocket; capturing and sharing both photos and videos is now trivial and costless. It is predicted that of the record 1.2 trillion photos to be taken in 2017, a record 85% will come from mobile phones, with a corresponding 4.7 trillion photos expected to be stored this year[4]. Jointly and symbiotically, there are ever-improving means of storing and transmitting data, e.g., exabyte-scale cloud storage providers such as YouTube [8] and gigabit fiber optic Internet connections from all major U.S. telecoms. There has also been a drastic rise in ways to efficiently process data, a combination of both hardware and software. For example, the NVIDIA Tesla K20X, a modern workstation GPU, features 3,584 floating-point processing cores, enabling massive performance gains over CPUs for parallel numerical algorithms[5]. Thanks in large part to the general-purpose CUDA platform by NVIDIA [105], GPUs have become ubiquitous in computers, from mobile devices with integrated GPUs to the Titan supercomputer at Oak Ridge National Laboratory, which has 18,688 K20X GPUs in its cluster[5]. In software, the TensorFlow library [2] by Google enables massive, heterogeneous computations on data, such as language modeling using document models with terabytes of parameters [3], search result ranking for Google [2], and drug discovery based on 40 million biological data points [127]. All these developments have led to researchers striving to get even more data, which has created a virtuous circle of increasing data and improved data technology—it is truly a data explosion.

Statisticians look at lots of data. Historically, they are used to using small-scale data sets to predict larger behaviors—but such investigations are very much focused on small data, not dealing with big data. Thus, statisticians have largely not been ready for the data explosion. In fact, even in the case of something as important as the U.S. presidential election, reputable news agencies make

[4]See http://mylio.com/true-stories/tech-today/how-many-digital-photos-will-be-taken-2017-repost
[5]See https://www.olcf.ornl.gov/support/system-user-guides/accelerated-computing-guide/

predictions using only thousands of samples, when modern technology could easily enable collecting orders of magnitude more samples, as argued and demonstrated in [161]. Statisticians' predilection for small data has led to others coming to the forefront of data science; in part, those in artificial intelligence laboratories, especially machine learning specialists. Many of these practitioners still do collaborate with statisticians, use the language of probability/statistics, and work to develop lots of computational tools leveraging statistics—possibly too exclusively. In fact, there is a well-known equivalence between the maximum likelihood estimator (MLE) for parameters and least squares: given a collection of data points (X_i, Y_i), one may consider $Y_1, \ldots, Y_n$ to be random variables that follow a linear model $Y_i = \alpha X_i + \beta + \epsilon_i$, where α and β are unknown parameters and the $\epsilon_i \sim N(0, \sigma^2)$ are independent, identically distributed Gaussian noise. In this case, to determine the MLE for α and β, one maximizes the likelihood function

$$L(Y_1, \ldots, Y_n; \alpha, \beta, \sigma^2) = \frac{1}{(2\pi)^{n/2}\sigma^n} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i - \alpha X_i - \beta)^2\right).$$

Taking the logarithm and discarding terms constant in α, β, it is obvious that maximizing L is equivalent to minimizing

$$E(Y_1, \ldots, Y_n; \alpha, \beta) = \sum_{i=1}^{n}(Y_i - \alpha X_i - \beta)^2,$$

which is a standard least squares problem presented in introductory linear algebra courses. Given this equivalence, it seems rather naïve to not use the entire field of numerical liner algebra in machine learning; yet machine learning practitioners seem overly fixated on statistical descriptions.

6.2. Machine Learning

Traditional machine learning techniques are divided, broadly speaking, into two categories: unsupervised and supervised learning [49]. In unsupervised learning, no training labels are specified for the data. Thus, unsupervised learning is typically used for tasks including clustering and anomaly detection, where one wishes to identify similarities and differences in data without a priori knowledge of labels and/or classes for the data. A classic unsupervised learning technique is principal component analysis [117], which learns a basis of linearly uncorrelated features for the given training data and projects the data onto this space. In this sense, unsupervised learning can aid with both reducing the dimensionality of complex data sets and can lend insights into latent variables underlying the data.

In contrast, supervised learning makes use of labeled training sets. Common supervised learning tasks include classification and regression. Support vector machines[6], random forests [16], and logistic regression [24] are all examples of supervised learning algorithms. Supervised learning essentially means to create a function $y = f(x)$ for a collection of sample points (x, y). Given this function, one can then make predictions about points that lie in between or nearby samples in the training set. As noted in [49], there are not always clear distinctions between supervised and unsupervised learning tasks; for instance, rather than solving the unsupervised learning problem of modeling the probability distribution $p(x)$ for a vector $x \in \mathbb{R}^n$, one can apply the chain rule and instead apply supervised learning techniques to learn each term of $\prod_{i=1}^{n} p(x_i|x_1, \ldots, x_{i-1})$.

Success with traditional machine learning was actually quite limited. Although machine learning specialists used quite sophisticated techniques in probability and statistics, they used more sophisticated applied and computational mathematics only sparingly. For example, they used only the most basic linear algebra and forced all of their formulations to be linear and/or convex so that they could use convex optimization. Non-convex optimization or nonlinearities of any sort were avoided. For instance, support vector machines, workhorses of supervised learning, solve a convex optimization problem in order to find a linear separation between the input data [49]. In

[6] See http://www.svms.org/history.html

unsupervised learning, the popular k-means clustering algorithm is a least squares problem, again a convex optimization problem that may be written as a linear system.

While these linear approaches in traditional machine learning were effective for linear, convex data and models, they could not learn even the most basic nonlinear functions. XOR is an excellent example in this sense, defined on the binary input pairs $X = \{(0,0)^T, (0,1)^T, (1,0)^T, (1,1)^T\}$ by the respective outputs $Y = \{0,1,1,0\}$. Aiming to determine the parameters θ of a model $y = f(x;\theta)$ that optimize the model for the data, the traditional machine learning approach is to minimize the mean-squared error loss function $J(\theta) = \frac{1}{4}\sum_{x\in X}\left(Y(x) - f(x;\theta)\right)^2$, where $Y(x)$ is the true XOR function. A linear model $f(x;\theta) \equiv f(x;w,b) = x^T w + b$ is employed and, using the normal equations or another method, this least squares problem yields the solution $w = \mathbf{0}$, $b = 0.5$. In other words, the optimal linear fit to the XOR function outputs 0.5 for all inputs, i.e., always yielding incorrect results. This is as expected because the data is not linearly separable.

Historically, the limitations of traditional machine learning algorithms were observed by early works such as [97], and these observations contributed to a lack of funding of machine learning research in the U.S. These periods of limited funding are often referred to as "AI Winters," and included significant portions of both the 1970s and 1980s [106, 49]. Due to the funding climate in the U.S., many top machine learning researchers gathered in other research hubs. In one of the most important examples, noted practitioners Geoffrey Hinton (who invented backpropagation for multi-layer neural networks), Yoshua Bengio (one of the most highly cited authors on deep learning and neural networks), and Yann LeCun (a pioneer of convolutional neural networks, one of the leading deep learning techniques)—the "Canadian Mafia"[7]—all collaborated through the Canadian Institute for Advanced Research and worked as researchers or professors at the University of Toronto, McGill University, and the Université de Montréal, fueling a strong trend of machine learning and deep learning innovation in eastern Canada. Indeed, in order to move beyond the artificial barriers of linearity and convexity imposed by traditional machine learning—as a result, ending the AI Winters—specialists developed deep learning.

6.3. Deep Learning

Deep learning discovers a representation, or feature set, for data using a hierarchical model. The fundamental insight of deep learning is to construct complex representations using layers of simpler representations. These layers are typically chained together, such that the visible layers—those that represent directly observable features in the input data—send their outputs into a sequence of hidden layers, where the representations are more abstract concepts not directly provided by the data. An output layer at the top of the hierarchical model provides a learned representation to the end user. Deep learning differentiates itself from traditional machine learning by using "deep" models of potentially many layers in order to learn complex, abstract, and nonlinear representations that cannot be suitably captured by traditional methods.

Revisiting the earlier example of learning the XOR function, instead of a traditional machine learning approach, one may instead take a deep learning approach and define a simple feedforward network with nonlinear activation and a single hidden layer that can successfully learn XOR. An activation function specifies how to map inputs to outputs for a layer of the network. The key insight, as described in [49], is to use a nonlinear activation function such as

$$g(x) = \begin{pmatrix} 1 & -2 \end{pmatrix} \max\left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} x + \begin{pmatrix} 0 \\ -1 \end{pmatrix} \right\}$$

for the hidden layer, where $x \in X$ and max acts componentwise. It is trivial to verify that $g(X) = Y$, i.e., this network exactly recovers the XOR function. The nonlinear component of the above function, $h(z) = \max(0, z)$ is called a rectified linear unit (ReLU) and is the typical recommendation for activation functions in current deep learning models[8] [49]. Examples like

[7] See https://www.recode.net/2015/7/15/11614684

[8] Depending on the implementation, one may write this network with a pure ReLU layer and two linear layers, or as one combined layer.

the XOR function led to the idea of using deep learning with multiple layers to learn nonlinear representations, a significant step change beyond traditional machine learning. This mathematical power of deep learning ended the AI Winters and gave rise to a number of highly successful learning-based systems.

An explosion of development has resulted from this introduction of deep learning into machine learning. For example, deep learning approaches applied to many vision tasks have beaten state-of-the-art results, e.g., on the ILSVRC (ImageNet) image classification challenge [131] and on the COCO object recognition dataset [76]. Deep learning has also been successfully applied in other fields, such as robotics, medical imaging, self-driving cars, 3D shape analysis, etc.

In traditional machine learning, and even in deep learning, algorithms are often described in terms of finding the minimum of an objective function subject to certain constraints (stochastic gradient descent is an ubiquitous example); yet, because of the inherent nonlinearities and complex energy landscapes in most practical problems, one has no hope of finding actual minima. However, machine learning specialists often are not solving for minima anyways[9]; not only are minima unimportant, they are often undesired. The basic idea is that given a collection of samples (x, y), finding a function $\hat{y} = f(x)$ that interpolates them well is not necessarily the goal. We already know from basic polynomial interpolation and schemes like ENO [55], WENO [81], etc., that fitting a function to the data is far less important than the way the function behaves away from and in between the data. In deep learning, specialists refer to how well the data fits as the training error, and how well the function works away from the data as the generalization error. So minimizing the training error could give terrible generalization error, with overshoots, undershoots, etc. Thus, the real insight of using optimization algorithms in deep learning is that one is merely moving around in a parameter space, loosely guided by some optimization heuristic, in order to hopefully find a set of parameters that gives good generalization error.

6.4. Continuous and Applied Mathematics

Most recently in deep learning, especially driven by ubiquitous nonlinearities and the fact that the entire goal of supervised learning is function interpolation, there has been a resurgence in continuous and applied mathematics. It was only a few years ago that some top computer scientists were claiming that calculus should be replaced by discrete mathematics at the high school level[10]. However, this has changed radically as some of the most recent development has been around back-propagation for computing gradients of a function. Not surprisingly, when probability and machine learning books and courses talk about the difference between continuous/discrete probabilities and the necessity of integrating over an interval for the continuous case, there is usually at most only a footnote to measure theory[11]. However, we believe that a full embrace of continuous mathematics is coming, needed, and will be of high impact. Thus we feel that applied mathematics should take the lead in this area.

Interestingly, computational fluid dynamics and computational solid dynamics researchers have really mastered function interpolation as an art form. They use every kind of grid—structured and unstructured, refined and unrefined, hierarchical and overlapping grids, meshfree methods such as SPH, material point methods, hybridizations such as PIC and FLIP, every version of a finite basis one could imagine, isogeometric models based on CAD, etc. Then, using these finite ordered pairs, they carry out tasks including interpolation, differentiation, and quadrature. Every single one of these formulations provides ordered pairings (x, y) at a finite number of discrete points, or more generally, a finite number of coefficients for compact or even global functions, such as in spectral methods. The only difference between this community and the machine learning community is that the machine learning community is focused typically on (x, y) of higher dimension. But the top researchers in machine learning have come across one very important notion, identical to what top researchers found in solid and fluid mechanics: that this is an art, not a science. It does require

[9] See pp. 84–85 of [49].

[10] Anonymized personal communications, Stanford Computer Science department.

[11] See e.g. p. 191 of [130], p. 275 of [25], or pp. 1–2 of http://cs229.stanford.edu/section/cs229-prob.pdf

good mathematical intuition, polished technical skills, etc., but also requires what mathematicians refer to as originality or creativity. We have already trained an army of people good at creating low generalization error and only loosely guided by training error, working on computational fluids and solids, and we could unleash this army onto the machine learning community. Moreover, applied mathematicians and researchers in computational fluids and solids are in an ideal position to harness models and algorithms from machine learning, especially deep learning, in order to build physical simulation systems that better allow for the incorporation of data.

6.5. Recent Papers Integrating Learning and Computational Physics

A number of recent papers in the Journal of Computational Physics have addressed machine learning and deep learning, presenting applications across a diverse range of topics in computational physics. For instance, in [11], the authors apply PCA to a database of high-resolution incompressible flow simulations; new simulations are run using traditional high-resolution simulation in a small patch of the domain, while inferred data from the reduced-order model is used to simulate the rest of the domain. [113] uses Gaussian process regression and Bayesian optimization to learn order parameters in variable-order fractional advection-diffusion equations. Learning techniques were also applied in [5], where Bayesian linear regression is employed to predict an exchange-correlation functional for density functional theory simulations of quantum mechanical systems. In the domain of condensed matter physics, [122] uses a variety of supervised learning algorithms, including decision trees, random forests, k-nearest neighbors, and neural networks, to predict energies and magnetizations of Ising model configurations from a large database of configurations mapped using PCA. In yet another application domain, [37] designs neural networks to model the superconducting magnets in tokamak fusion reactors, and in turn, uses these models to efficiently optimize system design choices. These works suggest the broad applicability of learning techniques throughout computational physics.

One particularly active application domain for integrating learning and physics is the modeling and simulation of turbulent flow. Turbulence modeling is a topic ripe for learning approaches since turbulence models are often empirical and target narrow flow regimes; thus, it is difficult to manually select models and model parameters when attempting to model and simulate complex turbulent flows. Rather than manual parameter selection, [75] learns optimal closure coefficients for the Reynolds-averaged Navier-Stokes (RANS) k-ω turbulence model by solving a weighted least squares regression problem on each iteration, minimizing weighted distance between simulated quantities and experimental data. [162] uses gene expression programming, a type of evolutionary learning algorithm, to learn algebraic models for the Reynolds stress tensor in RANS simulations, based on a database of high-fidelity simulation data. [77] demonstrates the use of random forests and neural networks for learning the second invariant of the Reynolds stress anisotropy from RANS simulation training data. The authors in [15] reduce simulation data using nonlinear Laplacian spectral analysis and use regression analysis to study the reduced basis in the context of simulating two-dimensional turbulent Rayleigh-Bénard convection. [163] presents a Bayesian framework based on an iterative Kalman method for incorporating experimental data and physical constraints into both the prediction and uncertainty quantification of RANS simulations. [116] presents a general-purpose technique where Gaussian processes are used with ensembles of inverse problem solution estimates in order to learn model parameters for linear and nonlinear problems, such as RANS closure modeling.

Another topic where learning techniques have generated considerable interest is the modeling and simulation of materials, especially in multiscale frameworks. For example, [12] uses Isomap, a nonlinear manifold learning technique, to build a reduced-order model for multiscale analysis of nonlinear hyperelastic materials in a finite strain setting. Complex subsurface geological models are studied in [160], which presents a variant of kernel PCA in order to build reduced-order models that are more consistent with data. [69] applies multivariate linear regression on a trained principal component feature basis in order to learn optimal yield strength and strain partitioning model parameters in the context of multiphase materials. In [6], a Bayesian linear regression model and the cluster expansion spectral decomposition approach are used to predict, with uncertainties, thermodynamic properties of metal alloys. [88] presents a general review of predictive multiscale

modeling of heterogeneous media; of note are the authors' surveys of data-driven model generation and learning techniques for dimensionality reduction and simulation.

Several recent works have presented learning techniques that are broader or more foundational in their applications. For example, [71, 70] describe a framework that uses learning techniques including information fusion, multi-level Gaussian process regression, and diffusion maps in order to perform fluid dynamics simulations that can scale better in parallel environments, that can be resilient against faulty hardware in a large cluster, and that can use provided data to improve simulation results. [13] integrates sparse grid techniques with the principal manifold learning algorithm in order to provide a more efficient generative dimensionality reduction technique, demonstrating their method on a finite element simulation of an automotive crash test. In [124, 125], methods are described that use Gaussian processes to learn from sparse, possibly noisy training data the parameters of general linear differential operators, such as those found in heat equations and reaction-diffusion equations.

We emphasize the work of [87], which directly connects deep learning to multiphase flow simulation. The authors run a number of direct numerical simulation (DNS) experiments of two-fluid bubbly flow to use as training data. This simulation data is used to train a three-layer neural network that learns models for their system's closure terms, namely the gas flux and the streaming stresses. Subsequently, initial conditions like void fractions are varied, and new simulations are run using the results of the neural network; the authors find surprising agreement of the learning-driven results with DNS results of the same scenarios. [87] also references several older works related to deep learning and multiphase flows, such as [90], which uses neural networks to identify different flow regimes in gas-liquid flows, as well as [91], which uses neural networks to reconstruct near-wall flow in a turbulent channel flow. Multiphase flow is a promising target for learning-enabled modeling and simulation paradigms, and we remark in the next section on potential research directions at the intersection of these fields.

JCP is not the only place for simulation. Besides JCP, fundamental numerical methods are published in a number of other venues, such as CMAME or IJNME for computational solids, or even SIAM J. Num. Anal., JFM, Phys. Fluids, etc. In fact, even the most recent SIGGRAPH and SCA conferences have a number of papers combining machine learning and deep learning with simulation-oriented activities. For instance, in [23], a least squares-style regression is used to optimize parameters for a cloth simulation model based on experimentally captured data. [31] describes a method that can highly compress incompressible fluid data, which could serve as a type of lossy reduced-order model for data-driven simulation algorithms. Deep learning-based approaches for 3D animated character control are considered in [118, 126, 60, 79], with deep reinforcement learning approaches studied in [118] and sampling based controllers trained with neural networks examined in [126]. A dynamic neural network architecture for character control is demonstrated in [60], and deep Q-learning is used to train a control system based on short control fragments in [79]. Deep learning is applied to the problems of facial animation capture and generation in [68, 152, 67, 148]. [68] uses deep CNNs to learn a model of an actor's facial performance, which enables accurate 3D facial animation reconstruction by querying the CNN with monocular performance capture. Speech animation in particular is considered using a deep neural network trained on video and audio in [152], a deep neural network trained on audio and a description of emotional state in [67], and a recurrent neural network trained on audio samples in [148].

6.6. Opportunities for Deep Learning in Multiphase Flow

Following the above review of multiphase flow, machine learning, and deep learning, we suggest several future avenues of inquiry for using deep learning to enhance the study and simulation of multiphase flow.

- Deep learning models include feedfoward neural networks, reinforcement learning, convolutional neural nets, etc. Within these categories, there are further design choices, such as selecting activation functions for hidden layers in a neural network. How does the type of problem being studied, e.g., gas-liquid bubbly flow versus porous media infiltration, affect the type of learning model that should be used, along with the particular design choices involved in that model?

- Statistical methods typically ignore underlying physical principles, such as the incompressibility of a fluid. How can learning algorithms be designed to best incorporate physical constraints while avoiding overfitting to imposed physics?
- Can learning techniques aid in creating robust, massively parallel multiphase flow simulation, as suggested in [71, 70]?
- What coarsening and upscaling strategies are most amenable to learning techniques? More broadly, how can one best use data—possibly obtained across multiple scales—to improve the fidelity and efficiency of multiscale multiphase flow simulations?
- What general-purpose reduced-order modeling and dimension reduction techniques are applicable to the interplay between learning and multiphase flow? Can dimension reduction techniques help identify hidden low-dimensional features and manifolds in data, or reduce the complexity of data that needs to be collected and models that need to be predicted?
- How will continued advances in computing affect multiphase flow, especially in the case of learning-enabled techniques? What depth or complexity of deep learning models, and what amount of training data, is sufficient to produce state-of-the-art predictive models?
- How does the advent of deep learning and data-driven simulation impact real-world collaboration between experimentalists, computational scientists, and theorists? How can these communities collaborate, for example, to produce useful, large-scale training data sets for multiphase flow simulation?

7. Conclusion

We have presented a review of sharp interface methods as well as a discussion of machine learning and deep learning techniques for the simulation of multiphase flows. We have also discussed the recent advances in adaptive grid refinement techniques and their parallel extensions.

Acknowledgment

The research of Frederic Gibou was supported by ONR under MURI award N00014-17-1-2676 and by ARO W911NF-16-1-0136. The research of David Hyde and Ron Fedkiw was supported by ONR N00014-13-1-0346, ONR N00014-17-1-2174, ARL AHPCRC W911NF-07-0027 and NSF CNS1409847.

References

[1] M. Aanjaneya, S. Patkar, and R. Fedkiw. A monolithic mass tracking formulation for bubbles in incompressible flow. *J. Comput. Phys.*, 247:17 – 61, 2013.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[4] D. Adalsteinsson and J. Sethian. The Fast Construction of Extension Velocities in Level Set Methods. *J. Comput. Phys.*, 148:2–22, 1999.

[5] M. Aldegunde, J. R. Kermode, and N. Zabaras. Development of an exchange-correlation functional with uncertainty quantification capabilities for density functional theory. *J. Comput. Phys.*, 311:173 – 195, 2016.

[6] M. Aldegunde, N. Zabaras, and J. Kristensen. Quantifying uncertainties in first-principles alloy thermodynamics using cluster expansions. *J. Comput. Phys.*, 323:17 – 44, 2016.

[7] T. Aslam, S. Luo, and H. Zhao. A static pde approach for multi-dimensional extrapolation using fast sweeping methods. *SIAM Journal on Scientific Computing*, 36, 2014.

[8] B. Baesens. *Analytics in a big data world: The essential guide to data science and its applications*. John Wiley & Sons, 2014.

[9] D. Benson. Computational methods in Lagrangian and Eulerian hydrocodes. *Comput. Meth. in Appl. Mech. and Eng.*, 99:235–394, 1992.

[10] D. Benson. Volume of Fluid Interface Reconstruction Methods for Multimaterial Problems. *Applied Mechanics Reviews*, 52:151–165, 2002.

[11] M. Bergmann, A. Ferrero, A. Iollo, E. Lombardi, A. Scardigli, and H. Telib. A zonal Galerkin-free POD model for incompressible flows. *J. Comput. Phys.*, 352:301 – 325, 2018.

[12] S. Bhattacharjee and K. Matouš. A nonlinear manifold-based reduced order model for multiscale analysis of heterogeneous hyperelastic materials. *J. Comput. Phys.*, 313:635 – 653, 2016.

[13] B. Bohn, J. Garcke, and M. Griebel. A sparse grid based method for generative dimensionality reduction of high-dimensional data. *J. Comput. Phys.*, 309:1 – 17, 2016.

[14] J. U. Brackbill, D. B. Kothe, and C. Zemach. A Continuum method for modelling surface tension. *J. Comput. Phys.*, 100:335–353, 1992.

[15] N. Brenowitz, D. Giannakis, and A. Majda. Nonlinear Laplacian spectral analysis of Rayleigh-Bénard convection. *J. Comput. Phys.*, 315:536 – 553, 2016.

[16] L. Brieman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[17] R. Caiden, R. P. Fedkiw, and C. Anderson. A numerical method for two-phase flow consisting of separate compressible and incompressible regions. *J. Comput. Phys.*, 166(1):1 – 27, 2001.

[18] A. Chacon and A. Vladimirsky. A parallel two-scale method for Eikonal equations. *SIAM J. on Scientific Computing*, 37(A156-A180), 2015.

[19] Y.-C. Chang, T. Hou, B. Merriman, and S. Osher. Eulerian Capturing Methods Based on a Level Set Formulation for Incompressible Fluid Interfaces. *J. Comput. Phys.*, 124:449–464, 1996.

[20] H. Chen, C. Min, and F. Gibou. A Supra-Convergent Finite Difference Scheme for the Poisson and Heat Equations on Irregular Domains and Non-Graded Adaptive Cartesian Grids. *Journal of Scientific Computing*, 31(1-2):19–60, Mar. 2007.

[21] H. Chen, C. Min, and F. Gibou. A numerical scheme for the Stefan problem on adaptive Cartesian grids with supralinear convergence rate. *J. Comput. Phys.*, 228(16):5803–5818, 2009.

[22] A. J. Chorin. A Numerical Method for Solving Incompressible Viscous Flow Problems. *J. Comput. Phys.*, 2:12–26, 1967.

[23] D. Clyde, J. Teran, and R. Tamstorf. Modeling and data-driven parameter estimation for woven fabrics. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, pages 17:1–17:11, New York, NY, USA, 2017. ACM.

[24] D. Cox. The regression analysis of binary sequences (with discussion). *J. Roy. Stat. Soc. B*, 20:215–242, 1958.

[25] A. DasGupta. *Probability for statistics and machine learning: fundamentals and advanced topics*. Springer Science & Business Media, 2011.

[26] R. DeBar. Fundamentals of the KRAKEN code. Technical report, Lawrence Livermore National Laboratory (UCID- 17366), 1974.

[27] M. Detrixhe and F. Gibou. Hybrid Massively Parallel Fast Sweeping Method for static Hamilton-Jacobi Equations. *J. Comput. Phys.*, 322:199 – 223, 2016.

[28] M. Detrixhe, F. Gibou, and C. Min. A parallel fast sweeping method for the eikonal equation. *J. Comput. Phys.*, 237:46 – 55, 2013.

[29] A. du Chene, C. Min, and F. Gibou. Second-order accurate computation of interface curvature in a level set framework. *J. Sci. Computing.*, 35:114–131, 2008.

[30] V. Dyadechko and M. Shashkov. Moment-of-Fluid Interface Reconstruction. Technical report, Los Alamos National Laboratory (LA-UR-05-7571), 2006.

[31] S. Eberhardt, S. Weissmann, U. Pinkall, and N. Thuerey. Hierarchical vorticity skeletons. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, pages 6:1–6:11, New York, NY, USA, 2017. ACM.

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

[33] R. E. English, L. Qiu, Y. Yu, and R. Fedkiw. Chimera grids for water simulation. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2013.

[34] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, 183(1):83 – 116, 2002.

[35] D. Enright, F. Losasso, and R. Fedkiw. A fast and accurate semi-Lagrangian particle level set method. *Computers and Structures*, 83:479–490, 2005.

[36] D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proc. 4th ASME-JSME Joint Fluids Eng. Conf.*, number FEDSM2003–45144. ASME, 2003.

[37] A. Froio, R. Bonifetto, S. Carli, A. Quartararo, L. Savoldi, and R. Zanino. Design and optimization of artificial neural networks for the modelling of superconducting magnets operation in tokamak fusion reactors. *J. Comput. Phys.*, 321:476 – 491, 2016.

[38] J. Gantz and D. Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Anal. Future 2012*, pages 1–16, 2012.

[39] F. Gibou, L. Chen, D. Nguyen, and S. Banerjee. A level set based sharp interface method for the multiphase incompressible navier–stokes equations with phase change. *J. Comput. Phys.*, 222(2):536–555, Mar. 2007.

[40] F. Gibou and R. Fedkiw. A fourth order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to the stefan problem. *J. Comput. Phys.*, 202(2):577 – 601, 2005.

[41] F. Gibou, R. Fedkiw, R. Caflisch, and S. Osher. A level set approach for the numerical simulation of dendritic growth. *J. Sci. Comput.*, 19:183–199, 2003.

[42] F. Gibou, R. Fedkiw, and S. Osher. A review of level-set methods and some recent applications. *J. Comput. Phys.*, 353:82 – 109, 2018.

[43] F. Gibou, R. P. Fedkiw, L.-T. Cheng, and M. Kang. A second-order-accurate symmetric discretization of the poisson equation on irregular domains. *J. Comput. Phys.*, 176(1):205 – 227, 2002.

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

[45] F. Gibou and C. Min. On the performance of a simple parallel implementation of the ilu-pcg for the poisson equation on irregular domains. *J. Comput. Phys.*, 231(14):4531 – 4536, 2012.

[46] F. Gibou, C. Min, and R. Fedkiw. High resolution sharp computational methods for elliptic and parabolic problems in complex geometries. *J. Sci. Comput.*, 54:369–413, 2013.

[47] J. Glimm, J. Grove, X. L. Li, K. Shyue, Y.Zeng, and Q. Zhang. Three Dimensional Front Tracking. *Society of Industrial and Applied Mathematics Journal of Scientific Computing*, 1998:703–729, 1998.

[48] J. Glimm, J. W. Grove, X. L. Li, and N.Zhao. Simple Front Tracking. *Contemporary Math.*, 238:133–149, 1999.

[49] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[50] J. T. Grétarsson, N. Kwatra, and R. Fedkiw. Numerically Stable Fluid-Structure Interactions Between Compressible Flow and Solid Structures. *J. Comput. Phys.*, 230:3062–3084, Jan. 2011.

[51] A. Guittet, T. Isaac, C. Burstedde, and F. Gibou. Direct numerical simulation of incompressible flows on parallel Octree grids. In preparation.

[52] A. Guittet, M. Lepilliez, S. Tanguy, and F. Gibou. Solving elliptic problems with discontinuities on irregular domains – the voronoi interface method. *J. Comput. Phys.*, 298:747 – 765, 2015.

[53] A. Guittet, C. Poignard, and F. Gibou. A voronoi interface approach to cell aggregate electropermeabilization. *J. Comput. Phys.*, 332:143 – 159, 2017.

[54] A. Guittet, M. Theillard, and F. Gibou. A stable projection method for the incompressible navier–stokes equations on arbitrary geometries and adaptive quad/octrees. *J. Comput. Phys.*, 292:215 – 238, 2015.

[55] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, III. *J. Comput. Phys.*, 71(2):231–303, 1987.

[56] A. Helgadóttir and F. Gibou. A Poisson-Boltzmann solver on irregular domains with Neumann or Robin boundary conditions on non-graded adaptive grid. *J. Comput. Phys.*, 230(10):3830–3848, May 2011.

[57] F. Hermeline. Two coupled particle-finite volume methods using delaunay-voronoi meshes for the approximation of vlasov-poisson and vlasov-maxwell equations. *J. Comput. Phys.*, 106:1–18, 1993.

[58] C. Hirt and B. Nichols. Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries. *J. Comput. Phys.*, 39:201–225, 1981.

[59] J. Hochstein and T. William. An implicit surface tension model. In 34^{th} *AIAA, Aerospace Sciences Meeting and Exhibit*, 1996.

[60] D. Holden, T. Komura, and J. Saito. Phase-functioned neural networks for character control. *ACM Trans. Graph.*, 36(4):42:1–42:13, July 2017.

[61] J. Hunt, A. Wray, and P. Moin. Eddies, streams, and convergence zones in turbulent flows. *NASA Technical Report*, pages 193 – 208, 1988.

[62] S. Hysing. A new implicit surface tension implementation for interfacial flows. *International Journal for Numerical Methods in Fluids*, 51:659–672, 2006.

[63] G.-S. Jiang and D. Peng. Weighted ENO Schemes for Hamilton-Jacobi Equations. *SIAM J. Sci. Comput.*, 21:2126–2143, 2000.

[64] D. Juric. A Front-Tracking Method for Dendritic Solidification. *J. Comput. Phys.*, 123(1):127–148, Jan. 1996.

[65] D. Juric and G. Tryggvason. A Front Tracking Method for Dendritic Solidification. *J. Comput. Phys*, 123:127–148, 1996.

[66] D. Juric and G. Tryggvason. Computations of Boiling Flows. *Int. J. Multiphase. Flow.*, 24:387–410, 1998.

[67] T. Karras, T. Aila, S. Laine, A. Herva, and J. Lehtinen. Audio-driven facial animation by joint end-to-end learning of pose and emotion. *ACM Trans. Graph.*, 36(4):94:1–94:12, July 2017.

[68] S. Laine, T. Karras, T. Aila, A. Herva, S. Saito, R. Yu, H. Li, and J. Lehtinen. Production-level facial performance capture using deep convolutional neural networks. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, pages 10:1–10:10, New York, NY, USA, 2017. ACM.

[69] M. I. Latypov and S. R. Kalidindi. Data-driven reduced order models for effective yield strength and partitioning of strain in multiphase materials. *J. Comput. Phys.*, 346:242 – 261, 2017.

[70] S. Lee, I. G. Kevrekidis, and G. E. Karniadakis. A general CFD framework for fault-resistant simulations based on multi-resolution information fusion. *J. Comp. Phys.*, 347:290–304, 2017.

[71] S. Lee, I. G. Kevrekidis, and G. E. Karniadakis. A resilient and efficient CFD framework: statistical learning tools for multi-fidelity and heterogeneous information fusion. *J. Comp. Phys.*, 344:516–533, 2017.

[72] M. Lentine, M. Cong, S. Patkar, and R. Fedkiw. Simulating free surface flow with very large time steps. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 107 – 116, 2012.

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

[74] M. Lepilliez, E. R. Popescu, F. Gibou, and S. Tanguy. On two-phase flow solvers in irregular domains with contact line. *J. Comput. Phys.*, 321:1217 – 1251, 2016.

[75] Z. Li, H. Zhang, S. C. Bailey, J. B. Hoagg, and A. Martin. A data-driven adaptive Reynolds-averaged Navier-Stokes k-ω model for turbulent flow. *J. Comput. Phys.*, 345:111 – 131, 2017.

[76] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[77] J. Ling, R. Jones, and J. Templeton. Machine learning strategies for systems with invariance properties. *J. Comput. Phys.*, 318:22 – 35, 2016.

[78] P. Liovic, M. Francois, M. Rudman, and M. R. Efficient simulation of surface tension-dominated flows through enhanced interface geometry interrogation. *J. Comp. Phys.*, 229:7520 – 7544, 2010.

[79] L. Liu and J. Hodgins. Learning to schedule control fragments for physics-based characters using deep q-learning. *ACM Trans. Graph.*, 36(3):29:1–29:14, June 2017.

[80] X.-D. Liu, R. P. Fedkiw, and M. Kang. A boundary condition capturing method for poisson's equation on irregular domains. *J. Comput. Phys.*, 160(1):151 – 178, 2000.

[81] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *J. Comput. Phys.*, 115(1):200–212, 1994.

[82] X.-D. Liu and T. Sideris. Convergence of the ghost-fluid method for elliptic equations with interfaces. *Math. Comp.*, 72:1731–1746, 2003.

[83] F. Losasso, R. Fedkiw, and S. Osher. Spatially Adaptive Techniques for Level Set Methods and Incompressible Flow. *Computers and Fluids*, 35:995–1010, 2006.

[84] F. Losasso, F. Gibou, and R. Fedkiw. Simulating Water and Smoke with an Octree Data Structure. *ACM Trans. Graph. (SIGGRAPH Proc.)*, pages 457–462, 2004.

[85] F. Losasso, T. Shinar, A. Selle, and R. Fedkiw. Multiple interacting liquids. *SIGGRAPH ACM TOG*, 25:812–819, 2006.

[86] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE TVCG*, 14:797–804, 2008.

[87] M. Ma, J. Lu, and G. Tryggvason. Using statistical learning to close two-fluid multiphase flow equations for a simple bubbly system. *Physics of Fluids*, 27(092101), 2015.

[88] K. Matouš, M. Geers, V. Kouznetsova, and A. Gillman. A review of predictive nonlinear theories for multiscale modeling of heterogeneous materials. *J. Comp. Phys.*, 330:192–220, 2017.

[89] B. Merriman, J. Bence, and S. Osher. Motion of multiple junctions; a level set approach. *J. Comp. Phys.*, 112:334 – 363, 1994.

[90] Y. Mi, M. Ishii, and L. H. Tsoukalas. Flow regime identification methodology with neural networks and two-phase flow models. *Nucl. Eng. Des.*, 204:87–100, 2001.

[91] M. Milano and P. Koumoutsakos. Neural network modeling for near wall turbulent flow. *J. Comp. Phys.*, 182:1–26, 2002.

[92] C. Min and F. Gibou. A second order accurate projection method for the incompressible Navier-Stokes equations on non-graded adaptive grids. *J. Comput. Phys.*, 219(2):912–929, Dec. 2006.

[93] C. Min and F. Gibou. Geometric integration over irregular domains with application to level-set methods. *J. Comput. Phys.*, 226(2):1432–1443, Oct. 2007.

[94] C. Min and F. Gibou. A second order accurate level set method on non-graded adaptive Cartesian grids. *J. Comput. Phys.*, 225(1):300–321, 2007.

[95] C. Min and F. Gibou. Robust second-order accurate discretizations of the multi-dimensional Heaviside and Dirac delta functions. *J. Comput. Phys.*, 227(22):9686–9695, Nov. 2008.

[96] C. Min, F. Gibou, and H. Ceniceros. A supra-convergent finite difference scheme for the variable coefficient Poisson equation on non-graded grids. *J. Comput. Phys.*, 218(1):123–140, Oct. 2006.

[97] M. Minsky and S. Papert. *Perceptrons*. MIT Press, 1969.

[98] M. Mirzadeh and F. Gibou. A conservative discretization of the Poisson–Nernst–Planck equations on adaptive cartesian grids. *J. Comput. Phys.*, 274:633–653, 2014.

[99] M. Mirzadeh, A. Guittet, C. Burstedde, and F. Gibou. Parallel level-set methods on adaptive tree-based grids. *J. Comput. Phys.*, 322:345 – 364, 2016.

[100] M. Mirzadeh, M. Theillard, and F. Gibou. A Second-Order Discretization of the Nonlinear Poisson-Boltzmann Equation over Irregular Geometries using Non-Graded Adaptive Cartesian Grids. *J. Comput. Phys.*, 230(5):2125–2140, Dec. 2010.

[101] M. Mirzadeh, M. Theillard, and F. Gibou. A second-order discretization of the nonlinear Poisson–Boltzmann equation over irregular geometries using non-graded adaptive Cartesian grids. *J. Comput. Phys.*, 230(5):2125 – 2140, 2011.

[102] Y. T. Ng, H. Chen, C. Min, and F. Gibou. Guidelines for Poisson Solvers on Irregular Domains with Dirichlet Boundary Conditions Using the Ghost Fluid Method. *Journal of Scientific Computing*, 41(2):300–320, May 2009.

[103] Y. T. Ng, C. Min, and F. Gibou. An efficient fluid–solid coupling algorithm for single-phase flows. *J. Comput. Phys.*, 228(23):8807 – 8829, 2009.

[104] D. Nguyen, F. Gibou, and R. Fedkiw. A Fully Conservative Ghost Fluid Method and Stiff Detonation Waves. In *12th Int. Detonation Symposium, San Diego, CA*, 2002.

[105] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with CUDA. *Queue*, 6(2):40–53, 2008.

[106] N. Nilsson. *The Quest for Artificial Intelligence*. Cambridge University Press, 2009.

[107] W. Noh and P. Woodward. SLIC (simple line interface calculation). In *5th International Conference on Numerical Methods in Fluid Dynamics*, pages 330–340, 1976.

[108] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002. New York, NY.

[109] S. Osher and R. P. Fedkiw. Level set methods: An overview and some recent results. *J. Comput. Phys.*, 169(2):463 – 502, 2001.

[110] S. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[111] G. Ouaknin, N. Laachi, D. Bochkov, K. Delaney, G. Fredrickson, and F. Gibou. Functional level-set derivative for self consistent field theory. *J. Comput. Phys.*, 345:207 – 223, 2017.

[112] G. Ouaknin, N. Laachi, K. Delaney, G. H. Fredrickson, and F. Gibou. Self-consistent field theory simulations of polymers on arbitrary domains. *J. Comput. Phys.*, 327:168 – 185, 2016.

[113] G. Pang, P. Perdikaris, W. Cai, and G. E. Karniadakis. Discovering variable fractional orders of advection–dispersion equations from field data using multi-fidelity Bayesian optimization. *J. Comput. Phys.*, 348:694 – 714, 2017.

[114] J. Papac, F. Gibou, and C. Ratsch. Efficient symmetric discretization for the Poisson, heat and Stefan-type problems with Robin boundary conditions. *J. Comput. Phys.*, 229(3):875–889, 2010.

[115] J. Papac, A. Helgadottir, C. Ratsch, and F. Gibou. A level set approach for diffusion and stefan-type problems with Robin boundary conditions on quadtree/octree adaptive cartesian grids. *J. Comput. Phys.*, 233:241, 2013.

[116] E. J. Parish and K. Duraisamy. A paradigm for data-driven predictive modeling using field inversion and machine learning. *J. Comput. Phys.*, 305:758 – 774, 2016.

[117] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[118] X. B. Peng and M. van de Panne. Learning locomotion skills using DeepRL: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, pages 12:1–12:13, New York, NY, USA, 2017. ACM.

[119] C. Peskin. Numerical Analysis of Blood Flow in the Heart. *J. Comput. Phys.*, 25:220–252, 1977.

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

[121] S. Popinet. Gerris: A Tree-Based Adaptive Solver for the Incompressible Euler Equations in Complex Geometries. *J. Comput. Phys.*, 190:572–600, 2003.

[122] N. Portman and I. Tamblyn. Sampling algorithms for validation of supervised learning models for Ising-like systems. *J. Comput. Phys.*, 350:871 – 890, 2017.

[123] M. Raessi, M. Bussmann, and J. Mostaghimi. A semi-implicit finite volume implementation of the csf method for treating surface tension in interfacial flows. *International Journal for Numerical Methods in Fluids*, 59:1093 – 1110, 2009.

[124] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Inferring solutions of differential equations using noisy multi-fidelity data. *J. Comput. Phys.*, 335:736 – 746, 2017.

[125] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Machine learning of linear differential equations using Gaussian processes. *J. Comput. Phys.*, 348:683 – 693, 2017.

[126] J. Rajamäki and P. Hämäläinen. Augmenting sampling based controllers with machine learning. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, pages 11:1–11:9, New York, NY, USA, 2017. ACM.

[127] B. Ramsundar, S. Kearnes, P. Riley, D. Webster, D. Konerding, and V. Pande. Massively multitask networks for drug discovery. 2015.

[128] C. Ratsch, M. Gyure, F. Gibou, M. Petersen, M. Kang, J. Garcia, and D. Vvedensky. Level-Set Method for Island Dynamics in Epitaxial Growth. *Phys. Rev. B.*, 65:195403, 2002.

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

[130] S. Ross. *A first course in probability.* Pearson, 9th edition, 2014.

[131] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[132] G. Russo and P. Smereka. A remark on computing distance functions. *J. Comput. Phys.*, 163:51–67, 2000.

[133] S. Ruuth. A diffusion-generated appraoch to multiphase motion. *J. Comp. Phys.*, 145:166–192, 1998.

[134] C. H. Rycroft. Voro++: A three-dimensional voronoi cell library in C++. *Chaos*, 19, 2009.

[135] C. H. Rycroft and F. Gibou. Simulations of a stretching bar using a plasticity model from the shear transformation zone theory. *J. Comput. Phys.*, 231(5):2155 – 2179, 2012.

[136] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, 2003.

[137] J. Sethian. Fast marching methods. *SIAM Review*, 41:199–235, 1999.

[138] J. A. Sethian. *Level set methods and fast marching methods*, volume 3 of *Cambridge Monographs on Applied and Computational Mathematics.* Cambridge University Press, Cambridge, second edition, 1999. Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science.

[139] J. A. Sethian and P. Smereka. Level Set Methods for Fluid Interfaces. *Annual Review of Fluid Mechanics*, 35:341–372, 2003.

[140] C.-W. Shu and S. Osher. Efficient Implementation of Essentially Non-Oscillatory Shock Capturing Schemes II. *J. Comput. Phys.*, 83:32–78, 1989.

[141] P. Smereka. Semi-Implicit Level Set Methods for Curvature and Surface Diffusion Motion. *J. Sci. Comput.*, 19:439–456, 2003.

[142] K. Smith, F. Solis, and D. Chopp. A projection method for motion of triple junction by level sets. *Interface and Free Boundaries*, 4:263 – 276, 2002.

[143] M. Sussman. A second-order coupled level-set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *J. Comp. Phys.*, 187:110 – 136, 2003.

[144] M. Sussman. A method for overcoming the surface tension time step constraint in multiphase flows ii. *International Journal for Numerical Methods in Fluids*, 68:1343 – 1361, 2012.

[145] M. Sussman and M. Ohta. A stable and efficient method for treating surface tension in incompressible two-phase flow. *J. Sci. Comput.*, 31:2447 – 2471, 2009.

[146] M. Sussman and E. G. Puckett. A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. *J. Comput. Phys.*, 162(2):301 – 337, 2000.

[147] M. Sussman, P. Smereka, and S. Osher. A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *J. Comput. Phys.*, 114:146–159, 1994.

[148] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman. Synthesizing obama: Learning lip sync from audio. *ACM Trans. Graph.*, 36(4):95:1–95:13, July 2017.

[149] S. Tanguy and A. Berlemont. Development and Applications of a Level Set Method for Interface Tracking. In *Proceedings of the Nineth International Conference on Liquid Atomization and Spray Systems*, Sorrento, Italy, 2003.

[150] S. Tanguy, T. Ménard, and A. Berlemont. A level set method for vaporizing two-phase flows. *J. Comput. Phys.*, 221(2):837 – 853, 2007.

[151] S. Tanguy, M. Sagan, B. Lalanne, F. Couderc, and C. Colin. Benchmarks and numerical methods for the simulation of boiling flows. *J. Comput. Phys.*, 264(Supplement C):1 – 22, 2014.

[152] S. Taylor, T. Kim, Y. Yue, M. Mahler, J. Krahe, A. G. Rodriguez, J. Hodgins, and I. Matthews. A deep learning approach for generalized speech animation. *ACM Trans. Graph.*, 36(4):93:1–93:11, July 2017.

[153] M. Theillard, L. F. Djodom, J.-L. Vié, and F. Gibou. A second-order sharp numerical method for solving the linear elasticity equations on irregular domains and adaptive grids – application to shape optimization. *J. Comput. Phys.*, 233:430 – 448, 2013.

[154] M. Theillard, F. Gibou, and T. Pollock. A sharp computational method for the simulation of the solidification of binary alloys. *J. Sci. Comput.*, 63:330–354, 2014.

[155] M. Theillard, C. H. Rycroft, and F. Gibou. A multigrid method on non-graded adaptive octree and quadtree cartesian grids. *J. Scientific Comput.*, 55:1–15, 2012.

[156] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan. A Front-Tracking Method for the Computations of Multiphase Flow. *J. Comput. Phys.*, 169:708–759, 2001.

[157] J. Tsitsiklis. Efficient Algorithms for Globally Optimal Trajectories. *IEEE Trans. on Automatic Control*, 40:1528–1538, 1995.

[158] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multifluid flows. *J. Comput. Phys.*, 100:25–37, 1992.

[159] L. R. Villegas, R. Alis, M. Lepilliez, and S. Tanguy. A ghost fluid/level set method for boiling flows and liquid evaporation: Application to the leidenfrost effect. *J. Comput. Phys.*, 316(Supplement C):789 – 813, 2016.

[160] H. X. Vo and L. J. Durlofsky. Regularized kernel PCA for the efficient parameterization of complex geological models. *J. Comput. Phys.*, 322:859 – 881, 2016.

[161] W. Wang, D. Rothschild, S. Goel, and A. Gelman. Forecasting elections with non-representative polls. *International Journal of Forecasting*, 31(3):980 – 991, 2015.

[162] J. Weatheritt and R. Sandberg. A novel evolutionary algorithm applied to algebraic modifications of the RANS stress-strain relationship. *J. Comput. Phys.*, 325:22 – 37, 2016.

[163] H. Xiao, J.-L. Wu, J.-X. Wang, R. Sun, and C. Roy. Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier-Stokes simulations: A data-driven, physics-informed Bayesian approach. *J. Comput. Phys.*, 324:115 – 136, 2016.

[164] J. Xu and H. Zhao. A Eulerian formulation for solving partial differential equations along a moving interface. *J. Sci. Comput.*, 19:573 – 594, 2003.

[165] H. Zhao. A fast sweeping method for eikonal equations. *Mathematics of Computation*, 74:603–627, 2004.

[166] W. Zheng, J.-H. Yong, and J.-C. Paul. Simulation of bubbles. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 325 – 333, 2006.

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