On the use of Particle-in-Cell methods in Visual Effects

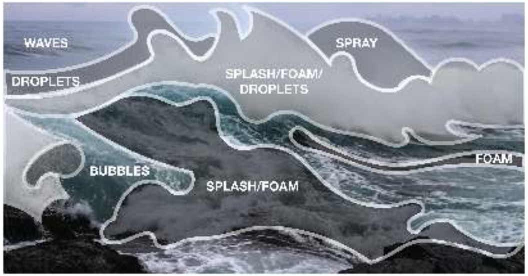
Richard Southern[*]

National Centre of Computer Animation
Bournemouth University

December 20, 2018

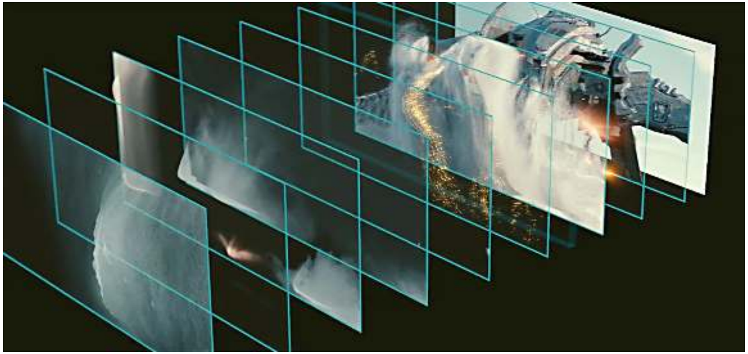
[*] rsouthern@bournemouth.ac.uk

Components of a breaking wave [9]



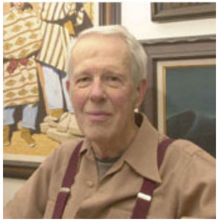
- Ocean waves populating the background of the shot,
- Main or "hero" waves in the foreground with potentially more complex behaviour,
- Droplets and splashes on the peak of the wave and against the rocks,
- Mist and vapour,
- Foam and churn in front of the wave near the short, and
- Bubbles entrained in the liquid creating lighter areas beneath the surface.

Artists think in *layers*



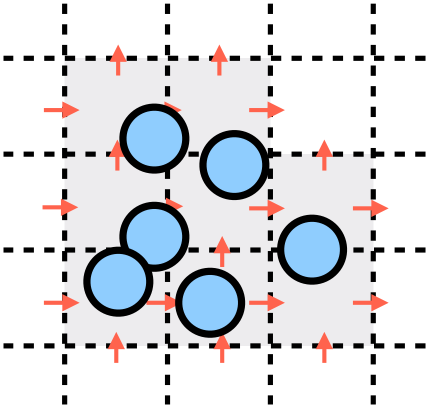
- Combining many physically plausible elements **reinforces believability**.
- A shot from *American Assassin* (2017) contained 78 layers.
- Many different methods used, but **Particle-In-Cell** drives main simulation.

Frank Harlow and the Los Alamos National Laboratory


Francis Harlow, Jan. 22, 1928 — July 1, 2016

One of the things about supersonic flows that I soon learned was that there were at that time two main ways to think about zoning space for resolving the behavior of a fluid. One of them we call the Lagrangian approach, which means having a mesh of computational cells that follow the motion of a fluid through its contortions and whatever processes that take place. The mesh could follow interfaces beautifully. The other, the Eulerian viewpoint, has a fixed mesh of cells that stay in one place in the laboratory frame and the fluid flows through it. This second approach is great for looking at large distortions; there is no movable computational mesh to get tangled. But on the other hand, it has problems if you want to follow a sharp interface between two fluids. Eulerian techniques tend to diffuse the interface and to smear out sharp shocks. *Harlow [5]*

A hybrid method



- Harlow and Welch [6] proposed Particle-In-Cell (PIC) method and the Marker-And-Cell (MAC) grid representation.
- Many incremental developments in following years, e.g. FLIP[2, 4], FLIP/PIC[16], APIC[8].
- Combines "best" properties from both Lagrangian and Eulerian methods:
  - Uses *particles* for advection and surfacing
  - Uses a *grid* to enforce incompressibility, resolve viscosity etc.
- Complex properties can be resolved using finite differencing.

Contributors to modern fluid in VFX

Walt Disney

DreamWorks

INDUSTRIAL LIGHT & MAGIC

A LUCASFILM COMPANY

AUTODESK.

DNEG

R&H

RHYTHM & HUES STUDIOS

Table of contents

Section 1

Separating the Navier–Stokes Equations

The Navier–Stokes Equations

The Navier-Stokes equation is typically defined as:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{\nabla p}{\rho} + \mathbf{g} + \nu \nabla^2 \mathbf{u}$$
$$\nabla \cdot \mathbf{u} = 0$$

where

- $\mathbf{u}$ is the fluid velocity vector,
- p is the fluid pressure,
- ρ is the fluid density,
- ν is the kinematic viscosity coefficient,
- and ∇ and ∇^2 represent the gradient differential and Laplacian operators respectively.

Dropping Viscosity

Standard practice in VFX is to simply drop viscosity and use the Euler equations[†]:

$$
\begin{aligned}
\frac{D\mathbf{u}}{Dt} &= -\frac{\nabla p}{\rho} + \mathbf{g} \\
\nabla \cdot \mathbf{u} &= 0
\end{aligned}
$$

[. . .] most numerical methods for simulating fluids unavoidably introduce errors that can be physically reinterpreted as viscosity [. . .] so even if we drop viscosity in the equations, we will still get something that looks like it. In fact, one of the big challenges in computational fluid dynamics is avoiding this spurious viscous error as much as possible. *Bridson [3]*

[†]Note the material derivative $\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \frac{D\mathbf{u}}{Dt}$ allows further simplification

Splitting the Equations

The first step is to define a simple forward Euler integration by splitting the Equation into separate components. This method is inspired by the approach proposed by Stam [14] and is crucial to enforcing stability. For a separable differential equation

$$\frac{dq}{dt} = f(q) + g(q)$$

we can split this with Euler integration to read:

$$\begin{aligned} \tilde{q} &= q^n + \Delta t f(q^n) \quad (1) \\ q^{n+1} &= \tilde{q} + \Delta t g(\tilde{q}) \quad (2) \end{aligned}$$

which can be shown to be first–order accurate (see Appendix).

Splitting the Navier–Stokes equations

The Euler equations can now be divided into three effective steps:

1. Application of **body forces** $\frac{\partial \mathbf{u}}{\partial t} = \mathbf{g}$, solved according to the forward Euler method $\tilde{\mathbf{u}} = \mathbf{u} + \Delta t \mathbf{g}$.
2. **Pressure projection** while preserving **incompressibility**:

$$\frac{\partial \mathbf{u}}{\partial t} = -\frac{1}{\rho} \Delta p \tag{3}$$

   subject to $\nabla \cdot \mathbf{u} = 0$. Covered later.

3. **Advecting**[‡] the properties due to the velocity field $\frac{D\mathbf{u}}{Dt} = 0$. Performed on the particles, dependent on integration scheme. A forward Euler advection step:

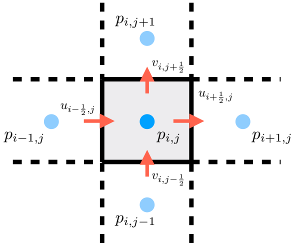
$$x_p^{n+1} = x_p + \Delta t \mathbf{u}_p \tag{4}$$

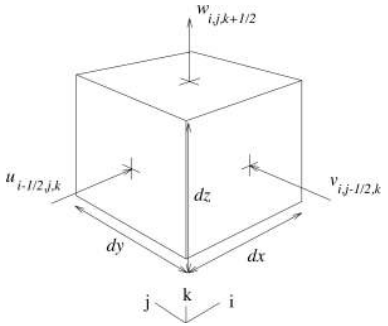
[‡]While any properties may be advected by the velocity field, this is typically particle positions x.

Section 2

Solving on a Grid

Discretization





- Properties from the particles are transfered to points on a grid.
- Velocities projected to the centre of cell faces (e.g. $u_{i+1/2,j}$)[§]
- Pressure defined at centres of cells.
- Properties transfered using *bi-* or *trilinear interpolation*.
- This staggered grid is called the **Marker-And-Cell (MAC) grid**[6].

[§]Note: neighbouring cells share points (e.g. $v_{i,j-1/2,k} \equiv v_{i,(j-1)+1/2,k}$).

Discrete approximations

The main advantage of the grid representation is that is allows the use of finite differencing to estimate quantities via second order accurate differences, e.g. at grid cell (i, j) with width Δx in $\mathbb{R}^2$:

$$\begin{aligned}\nabla \cdot \mathbf{u} &= \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \\ &\approx \frac{u_{i+1/2,j} - u_{i-1/2,j}}{\Delta x} + \frac{v_{i,j+1/2} - v_{i,j-1/2}}{\Delta x}\end{aligned} \tag{5}$$

and similarly at the centre of face $(i - 1/2, j)$,

$$\frac{\partial p}{\partial x} \approx \frac{p_{i,j} - p_{i-1,j}}{\Delta x} \tag{6}$$

Pressure Projection

- Must subtract the pressure gradient from the intermediate velocity field **u** to ensuring incompressibility and enforce boundary conditions, e.g.

$$\hat{\mathbf{u}} = \tilde{\mathbf{u}} - \nabla p. \tag{7}$$

- This method is the most common but differs from original approaches[2, 4, 6].
- The pressure projection step in Eq. 3 can be expressed as:

$$\mathbf{u}^{n+1} = \mathbf{u} - \frac{\Delta t}{\rho} \nabla p \tag{8}$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0 \tag{9}$$

Discretizing the Pressure Projection

In the discrete sense, Eq. 8 can be written as:

$$\begin{aligned} u^{n+1}_{i+1/2,j} =& u_{i+1/2,j} - \frac{\Delta t}{\rho}\frac{p_{i+1,j} - p_{i,j}}{\Delta x} \\ v^{n+1}_{i+1/2,j} =& v_{i,j+1/2} - \frac{\Delta t}{\rho}\frac{p_{i,j+1} - p_{i,j}}{\Delta x} \end{aligned} \tag{10}$$

while Eq. 9 can be written as

$$\nabla \cdot \mathbf{u}^{n+1} = \frac{u^{n+1}_{i+1/2,j} - u^{n+1}_{i-1/2,j}}{\Delta x} + \frac{v^{n+1}_{i,j+1/2} - v^{n+1}_{i,j-1/2}}{\Delta x} = 0 \tag{11}$$

Poisson Formulation

Substituting Eq. 10 into Eq. 11 and simplifying yields:

$$\frac{\Delta t}{\rho}\left(\frac{4p_{i,j} - p_{i+1,j} - p_{i,j+1} - p_{i-1,j} - p_{i,j-1}}{\Delta x^2}\right) = \\ -\left(\frac{u_{i+1/2,j} - u_{i-1/2,j}}{\Delta x} + \frac{v_{i,j+1/2} - v_{i,j-1/2}}{\Delta x}\right) \qquad (12)$$

This is the Poisson problem, reformulated as the system $\mathbf{Ap} = \mathbf{b}$ where

- $\mathbf{A}$ is a 5 (in $\mathbb{R}^2$) or 7 (in $\mathbb{R}^3$) point Laplacian matrix,
- $\mathbf{p}$ is the solution vector of pressures, and
- $\mathbf{b}$ is the negative vector of divergence values at each cell.

$\mathbf{A}$ is sparse, symmetric and positive definite, easily solved using Preconditioned Conjugate Gradient or Multi-Grid methods.

Section 3

Property Transfer

Property transfer with PIC

- In each time step mass and momentum from each particle is transfered to the centre of faces on the MAC grid.
- Harlow and Welch [6] transfers from particles to grid with

$$m_i^n = \sum_p w_{ip}^n m_p \tag{13}$$

$$m_i^n \mathbf{u}_i^n = \sum_p w_{ip}^n m_p \mathbf{u}_p^n \tag{14}$$

  where p indicates particle property and i indicates cell property, m^n is mass at time step n, $\mathbf{u}^n$ is velocity at time step n, and w_{ip}^n is the interpolation weights.
- Transfering velocity back to particles:

$$\mathbf{u}_{p,\mathrm{PIC}}^{n+1} = \sum_i w_{ip}^n \mathbf{u}_i^{n+1} \tag{15}$$

The Fluid Implicit Particle FLIP Method

- Brackbill et al. [2] improved backwards transfer (Eq. 15) by updating the original particle velocity by the difference:

$$\mathbf{u}_{p,\mathrm{FLIP}}^{n+1} = \mathbf{u}_p^n + \sum_i w_{ip}^n (\mathbf{u}_i^{n+1} - \mathbf{u}_i^n). \tag{16}$$

- Minor adjustment to PIC eliminates energy dissipation resulting from transfering to and from the grid representation.

FLIP/PIC blending

- FLIP alone is too “splashy”
- PIC simulations tend not to be splashy enough due to the energy dissipation
- Zhu and Bridson [16] proposed the linear of the results of both methods, e.g.

$$\mathbf{u}_p^{n+1} = (1-\alpha)\mathbf{u}_{p,\text{PIC}}^{n+1} + \alpha \mathbf{u}_{p,\text{FLIP}}^{n+1} \tag{17}$$

- Exposes α which allows user to control “splashiness” of the fluid (little physical justification)[¶].
- Linear momentum conserved during transfers with FLIP and PIC, but
- Angular momentum given by $L_{\text{tot}} = \sum_i x \times m\mathbf{u}$ is not conserved during transfer.

[¶] Can be derived from numerical dissipation: $\alpha = \min\left(\frac{6\Delta t\nu}{\Delta x^2}, 1\right)$.

Affine Particle In Cell

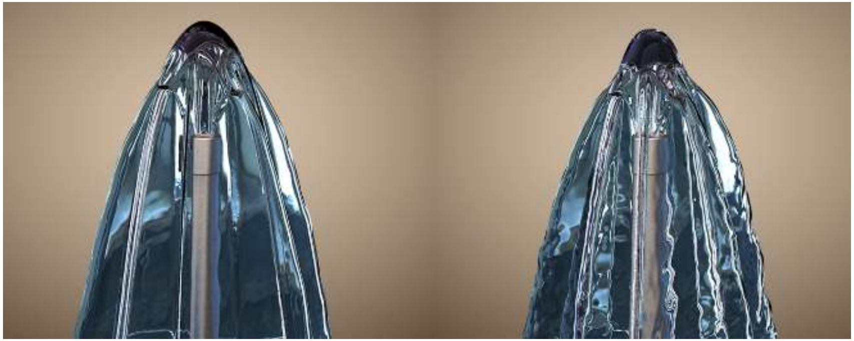
- Jiang et al. [8] stores two (2D) or three (3D) vectors at particles $\mathbf{c}_{pa}$, where a represents the x, y, z direction.
- The transfer from particles to the grid for the mass is as in Eq. 13, but to the faces is then achieved using:

$$m_{ai}^n u_{ai}^n = \sum_p m_p w_{aip}^n \left(\mathbf{e}_a^T \mathbf{u}_p^n + (\mathbf{c}_{pa}^n)^T (x_{ai} - x_p^n) \right) \tag{18}$$

  where $\mathbf{e}_a^T$ is the basis vector for axis a, and x_{ai} is the a-position of the grid node i.
- Ensures angular momentum is incorporated in grid velocity.
- The transfer of velocity from faces to particles is achieved using Eq. 15, but the vectors $\mathbf{c}_{pa}$ are updated using:

$$\mathbf{c}_{pa}^{n+1} = \sum_i \nabla w_{aip}^n \mathbf{u}_{ai}^{n+1}. \tag{19}$$

Affine Particle In Cell Results



- Minimal additional storage and performance overhead for significant quality improvement.
- FLIP/PIC is still used by VFX artists due to control they have over damping (the parameter α in Eq.17).
- Houdini [7] refers to FLIP as “splashy” and APIC as “swirly” to assist artists in identifying the appropriate tool.

Section 4

Implementation and Practicalities

The Volumetric Database

- Introduced by Museth [11] at DreamWorks. Academy Award in 2014 for wide range of industry applications.
- Near infinite volumes of 3D data, adaptive, hierarchical and cache coherent.
- $O(1)$ random access insertion, retrieval and deletion.
- Applied to many applications in Computer Graphics, including distributed PIC.

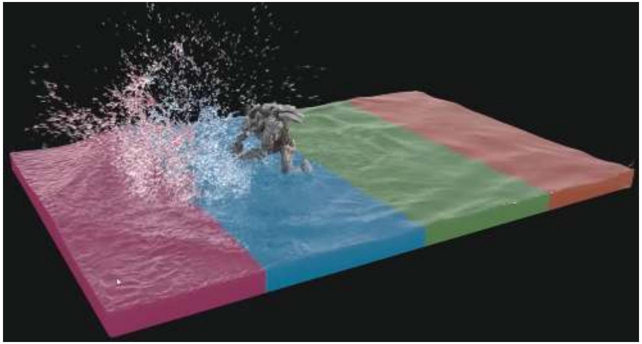


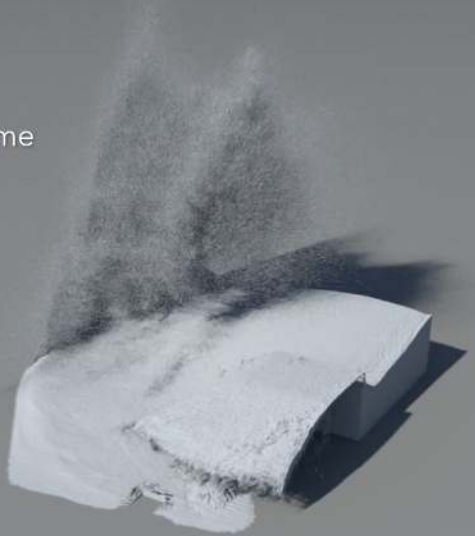
Figure: From SideFX [13].

Point Count: 1 billion

Peak Memory: 60GB

Performance: 10-15 mins/frame

Nodes: 1 machine



Render Time: 1 hour/frame

Memory: 11.0 GB

Insights from Jones [9]

- Artists work well with particles — PIC is naturally accepted part of fluid workflow.
- Adding and modifying particles has little risk of destabilising the simulation — important for *layering* effects.
- Whilst simulations are comparatively fast, they are dominated by the time to perform the *pressure projection*.
- Interaction and fluid behaviour happens on the scale of the background grid, so whilst particles are present in the model they do not represent any particular liquid mass.
- As the particles do not interact with each other directly, they are not guaranteed to be well-distributed.
- The simulations require manipulation of both point and volumetric data, and so artists may require ways to interact with both of these representations.

Insights from Others

- APIC has better visual quality than SPH in comparable time [15].
- "FLIP solvers do not need a high number of substeps" [compared with SPH] [12]. "The advantage of the FLIP Solver is that you run with only a few time steps per frame while SPH requires anywhere from 7 to 20 time steps or more per frame to stabilize." [7]
- "it is possible to add secondary fluids like splashes, foam, or mist" [12]. "You can introduce new particles at any time with little to no consequence. This opens up so many new work flows [. . .] that were simply not possible with SPH. For example, introducing splash particles with their own property attributes is now possible." [7]

Open Problems in Visual Effects

1. **Coupling of fluid solver with 2D ocean surface simulation**: It is impractical (and uncontrollable) to simulate the entire ocean with a full fluid solver. Linking these two so that the resulting simulation is seamless remains an open problem.
2. **Fluid authoring tools**: There is fundamentally a tradeoff between a simulation that is entirely physically plausible and one that is largely artistically controlled, and typically the amount of control varies depending on the specific shot.
3. **Adaptibility / multiresolution**: Currently DNEG can simulate 3 billion particles using distributed FLIP. Processing and storage requirements increase exponentially with particle numbers. Currently there is a requirement that fluid data can be visualised on a high end workstation, which limits the maximum resolution of any resulting fluid simulation. Multi-resolution would reduce hardware requirements significantly.
4. **Multiphase simulation**: Multiphase fluid simulation is a widely relevant problem, and also crops up sometimes in visual effects. A particularly common problem is the simulation of foam on ocean waves, but there are other examples from VFX.

Section 5

Appendices

Older method resolve FLIP/PIC pressure projection

The method described by [4] uses a relaxation method to eliminate velocity diffusion, which I include here for completeness. The Navier-Stokes equations can be decomposed into it's partial derivatives in each of the component dimensions (in $\mathbb{R}^2$):

$$\begin{aligned}
\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} &= -\frac{\partial p}{\partial x} + g_x + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\
\frac{\partial v}{\partial t} + \frac{\partial vu}{\partial x} + \frac{\partial v^2}{\partial y} &= -\frac{\partial p}{\partial y} + g_y + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \qquad (20)
\end{aligned}$$

where u, v are velocities in the x, y directions respectively, p is the local pressure, g is gravity and ν is the kinematic viscosity of the fluid. Terms on the left hand side of the equations account for changes in velocity due to local fluid acceleration and convection. The right hand terms take account of acceleration due to the force of gravity (or any body force g), acceleration due to the local pressure gradient (∇p) and drag due to kinematic viscosity (ν) or thickness of the fluid [4].

Older method resolve FLIP/PIC pressure projection contd

In Harlow and Welch [6], Brackbill et al. [2] and Foster and Metaxas [4], the discrete Navier-Stokes equations can be solved by finite differencing on the discretization:

$$\begin{aligned} u^{n+1}_{i+1/2,j} =& u_{i+1/2,j} + \Delta t\{(1/\Delta x)[u^2_{i,j} - u^2_{i+1,j}] \\ &+ (1/\Delta y)[(uv)_{i+1/2,j-1/2} - (uv)_{i+1/2,j+1/2}] + g_x \\ &+ (1/\Delta x)(p_{i,j} - p_{i+1,j}) \\ &+ (\nu/\Delta x^2)(u_{i+3/2,j} - 2u_{i+1/2,j} + u_{i-1/2,j}) \\ &+ (\nu/\Delta y^2)(u_{i+1/2,j+1} - 2u_{i+1/2,j} + u_{i+1/2,j-1}), \end{aligned} \tag{21}$$

yielding an explicit approximation of the updated velocity $u^{n+1}_{i,j}$ in terms of the projected pressure at cell centres and the projected velocities at staggered grid points.

Older method resolve FLIP/PIC pressure projection contd

The result is not divergence free / incompressible — Foster and Metaxas [4] determine the updated pressure field by solving the mass conservation equation $\nabla \mathbf{u} = 0$ by defining the fluid divergence (or "missing mass") for a cell (i, j) according to:

$$D_{i,j} = -\left(\frac{u_{i+1/2,j} - u_{i-1/2,j}}{\Delta x} + \frac{u_{i,j+1/2} - u_{i,j-1/2}}{\Delta y}\right) \tag{22}$$

with the change in pressure given by:

$$\Delta p_{i,j} = \frac{\beta_0}{2\Delta t}\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right) D_{i,j} \tag{23}$$

where β_0 is a relaxation coefficient within the range $[1, 2]$. The cell face velocities and cell pressure properties are then updated (see Foster and Metaxas [4] for the specifics), and the process is repeated. This iterative process converges in 3-6 iterations according to the paper.

Solving for viscosity

As stated previously, fluids are considered by default in VFX to be inviscid, but solutions to this are available. Batty and Bridson [1] solve viscosity as a separate step by the following PDE:

$$\hat{\mathbf{u}} = \frac{1}{\rho}\nabla \cdot \left(\nu \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right)\right) \tag{24}$$

which is discretized to give the following form:

$$\hat{\mathbf{u}} = \mathbf{u} + \frac{\Delta t}{\rho}\nabla \cdot \left(\nu \left(\nabla \mathbf{u}^* + (\nabla \mathbf{u}^\dagger)^T\right)\right) \tag{25}$$

where $\mathbf{u}^*$ and $\mathbf{u}^\dagger$ are used to denote whether the old or current version of $\mathbf{u}$ are used depending on the integration scheme. Once discretised this solution adds an additional linear solve to the system described above.

Solving for viscosity contd

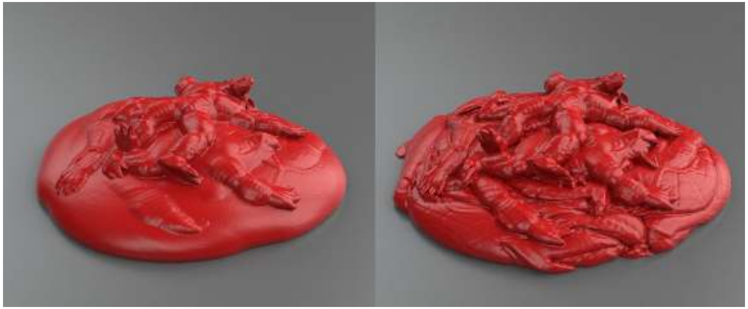


Figure: Collapsed piles of viscous armadillos. Boundary conditions of the method of Batty and Bridson [1] (left) reduce surface detail in comparison with the method of Larionov et al. [10]. Reproduced from Larionov et al. [10].

Larionov et al. [10] combine the viscous and pressure force solve into a single coupled system, and are able to address limitations in the decoupled approach of Batty and Bridson [1]. However the additional computational cost is not currently considered worth the qualitative improvement.

On the splitting of Ordinary Differential Equations

Bridson [3] demonstrates that ODE splitting is first order accurate using Taylor series expansion. Substituting Eq. 1 into Eq. 2 yields:

$$\begin{aligned}
q^{n+1} =& (q^n + \Delta t f(q^n)) + \Delta t g(q^n + \Delta t f(q^n)) \\
=& q^n + \Delta t f(q^n) + \Delta t (g(q^n) + O(\Delta t)) \\
=& q^n + \Delta t (f(q^n) + g(q^n)) + O(\Delta t^2) \\
=& q^n + \frac{dq}{dt}\Delta t + O(\Delta t^2).
\end{aligned}$$

Section 6

References

[1] Christopher Batty and Robert Bridson. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pages 219–228, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association. ISBN 978-3-905674-10-1. URL `http://dl.acm.org/citation.cfm?id=1632592.1632624`.

[2] J.U. Brackbill, D.B. Kothe, and H.M. Ruppel. Flip: A low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, 48(1): 25 – 38, 1988. ISSN 0010-4655. doi: https://doi.org/10.1016/0010-4655(88)90020-3. URL `http://www.sciencedirect.com/science/article/pii/0010465588900203`.

[3] Robert Bridson. *Fluid Simulation for Computer Graphics*. A K Peters, 2008.

[4] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471 – 483, 1996. ISSN 1077-3169. doi: https://doi.org/10.1006/gmip.1996.0039. URL `http://www.sciencedirect.com/science/article/pii/S1077316996900398`.

[5] Francis H. Harlow. Fluid dynamics in Group T-3 Los Alamos National Laboratory: (LA-UR-03-3852). *Journal of Computational Physics*, 195(2):414 – 433, 2004. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2003.09.031. URL `http://www.sciencedirect.com/science/article/pii/S0021999103005692`.

[6] Francis H. Harlow and J. Eddie Welch. Numerical Calculation of Time–Dependent Viscous Incompressible Flow of Fluid with Free Surface. *The Physics of Fluids*, 8(12):2182–2189, 1965. doi: 10.1063/1.1761178. URL `https://aip.scitation.org/doi/abs/10.1063/1.1761178`.

[7] SideFX Houdini. FLIP solver. URL `http://www.sidefx.com/docs/houdini/nodes/dop/flipsolver.html`.

[8] Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)*, 34(4):51, 2015. URL `https://dl.acm.org/citation.cfm?id=2766996`.

[9] Richard Jones. *Droplets, Splashes and Sprays: Turbulent Liquids in Visual Effects Production*. PhD thesis, Bournemouth University, 2019.

[10] Egor Larionov, Christopher Batty, and Robert Bridson. Variational stokes: A unified pressure-viscosity solver for accurate viscous liquids. *ACM Trans. Graph.*, 36(4):101:1–101:11, July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073628. URL http://doi.acm.org/10.1145/3072959.3073628.

[11] Ken Museth. Vdb: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph.*, 32(3):27:1–27:22, July 2013. ISSN 0730-0301. doi: 10.1145/2487228.2487235. URL http://doi.acm.org/10.1145/2487228.2487235.

[12] RealFlow. Glossary: Realflows liquid solver types. URL https://blog.realflow.com/technology/glossary-realflows-liquid-solver-types/.

[13] SideFX. Houdini 15 masterclass. URL https://www.youtube.com/watch?v=J86ycHU_ppc.

[14] Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-48560-5. doi: 10.1145/311535.311548. URL http://dx.doi.org/10.1145/311535.311548.

[15] Kiwon Um, Xiangyu Hu, and Nils Thuerey. Perceptual evaluation of liquid simulation methods. *ACM Trans. Graph.*, 36(4):143:1–143:12, July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073633. URL http://doi.acm.org/10.1145/3072959.3073633.

[16] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Trans. Graph.*, 24(3):965–972, July 2005. ISSN 0730-0301. doi: 10.1145/1073204.1073298. URL http://doi.acm.org/10.1145/1073204.1073298.