CIS 6600
Advanced Topics in Computer Graphics and Animation

Dr. Stephen H. Lane
(shlane@cis.upenn.edu)
University of Pennsylvania

Spring 2024

Fluid Simulation

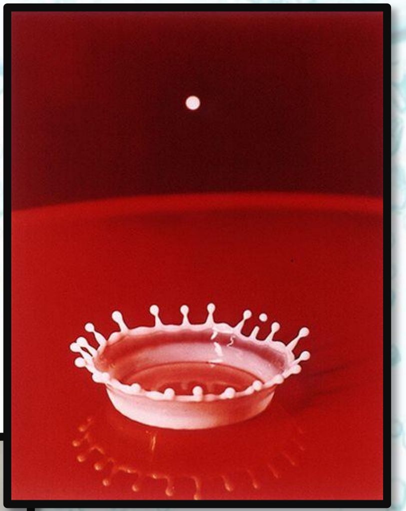
Overview

- Introduction
  - Applications
    - Film
    - Games
    - Scientific visualization
- Fluid Simulation Approaches
  - Height Field Methods
  - Physics-Based Approaches (Navier-Stokes equations)
    - Momentum equation (describe as more sophisticated form of f=ma)
    - Incompressibility equation (what is divergence free flow)
    - Typical boundary conditions (no slip and free slip)
    - Eulerian vs. Lagrangian Approaches (i.e. volumetric vs particle-based approaches)
    - Volumetric approaches (Eulerian)
    - MAC grid representation of pressure and velocity quantities
    - Simulation approach (based on steps in Stam paper)
  - Particle-based approaches (Lagrangian)
    - Smoothed Particle Hydrodynamics (SPH)
    - Basic Principles
    - Typical Kernel functions
    - Simulation approach
  - Hybrid methods
    - For example, PIC and FLIP
- Fluid Surface Representation
  - Marker particles
  - Level sets

What distinguishes fluids?





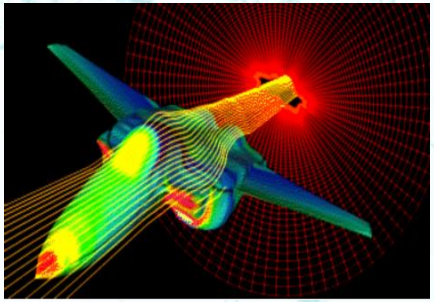


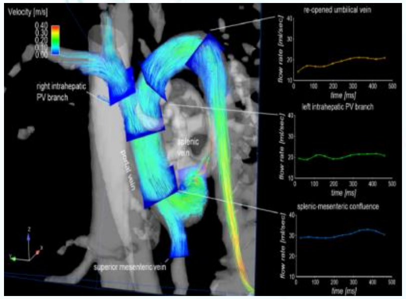
Intro

- What distinguishes fluids?
  - No “preferred” shape
  - Always flows when force is applied
  - Deforms to fit its container
  - Internal forces depend on velocities, not displacements

Intro

- Applications
  - Film
    - Avatar2
    - Croods
    - Moana
    - 2012
  - Games
    - Unity
    - Unreal
  - Scientific Visualization
    - Atmospheric Simulation
    - Hydrology (oceans and river flows)
    - Aerodynamics – laminar and turbulent flows
    - Astrophysics
    - Medical (blood flows)





Intro

- Fluid Equations of Motion
  - For physical simuations, we have Newton's second law:

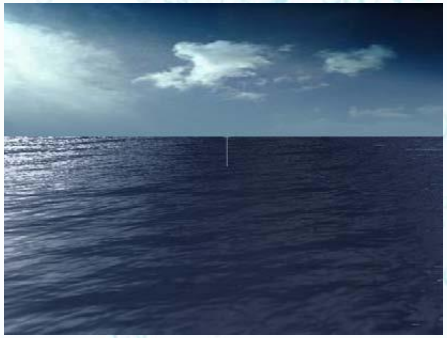
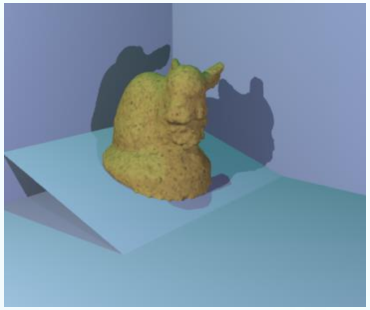
$$F = ma$$

  - Specialized for fluids:

"Navier-Stokes Equations"

Intro

- What can we achieve so far?
  - Newtonian fluid (water, oceans)
  - Smoke
  - Fire
  - Granular flow (Sand)
  - Non-Newtonian fluids
    - Blood, honey, goop, viscoelatic flow


Intro

- Fluid Characteristics
  - Basic properties
    - Pressure
    - Density
    - Viscosity (subject to shear stress)
    - Surface tension

Intro

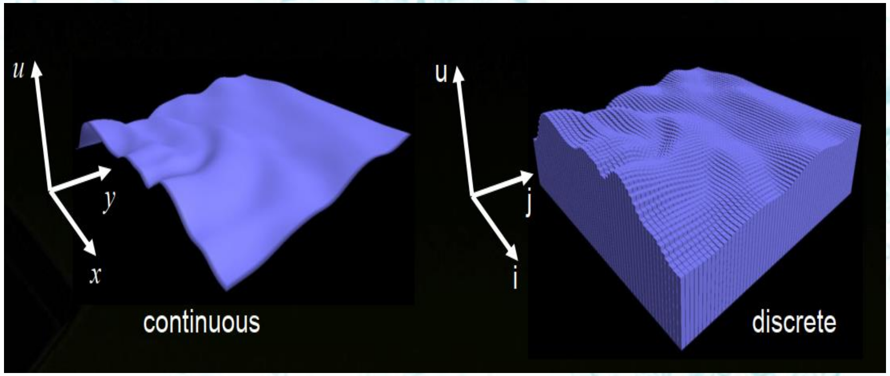
- Different types of fluids:
  - Incompressible (divergence-free) fluids: Fluids doesn’t change volume (very much).
  - Compressible fluids: Fluids change their significantly.
  - Viscous fluids: Fluids tend to resist a certain degrees of deformation
  - Inviscid (Ideal) fluids: Fluids don’t have resistance to the shear stress
  - Turbulent flow: Flow that appears to have chaotic and random changes
  - Laminar (streamline) flow: Flow that has smooth behavior

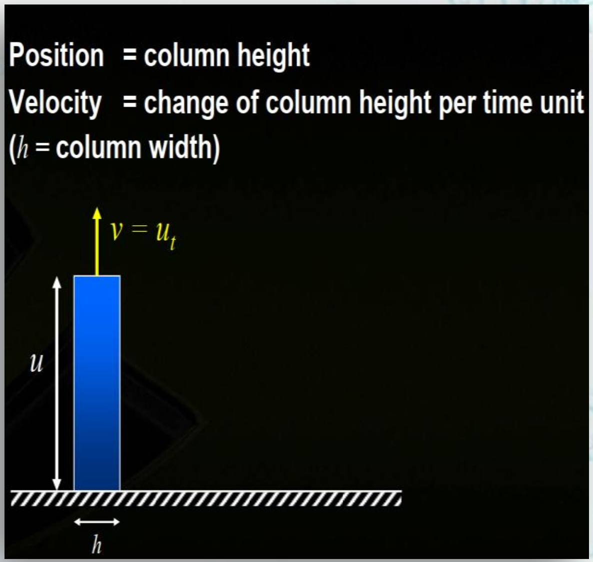
Intro

- Different types of fluids (Con’t):
  - Newtonian fluids: Fluids continue to flow, regardless of the force acting on it
  - Non-Newtonian fluids:
    - – Fluids that have non-constant viscosity
    - – Fluids may change physical behavior under different environmental conditions (i.e. phase transition)

Fluid Simulation Approaches

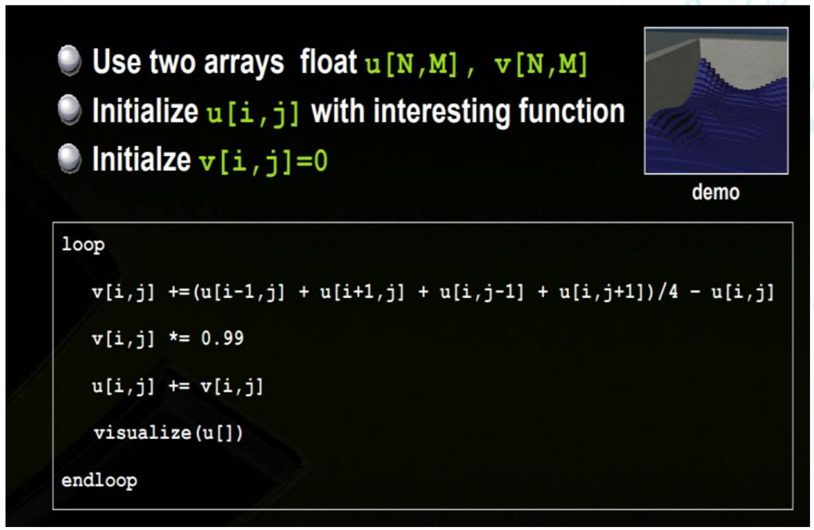
- Height Field Methods
  - Represent the water surface as a continuous 2D function u(x,y)
  - Discretize into grids
  - Imagine applying force to one column and integrating F=ma to get new height
  - Do the same for all columns in `u[i,j]`



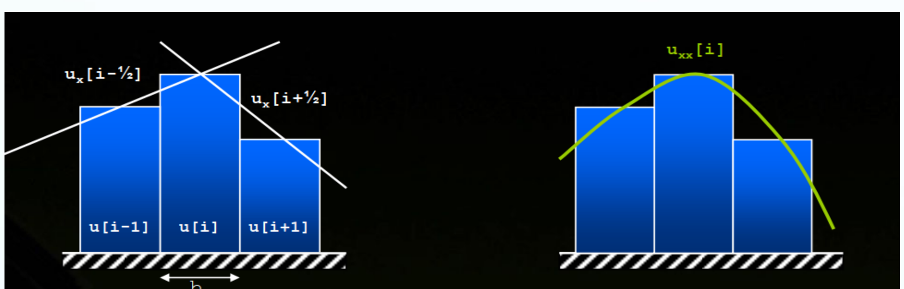


Fluid Simulation Approaches

- effect of force on height propagates to neighboring cells using a damped kernel function



- fit smooth surface to height field before rendering

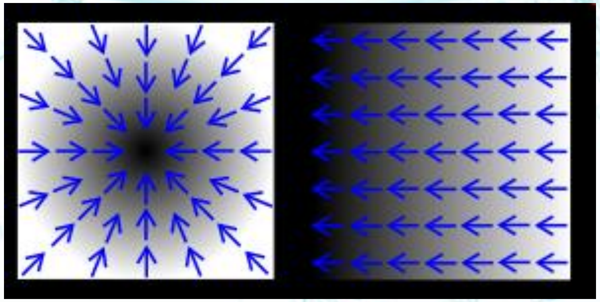


Quick Math Review

- Gradient (∇): A vector pointing in the direction of the greatest rate of increment

$$\nabla u = \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z} \right)$$

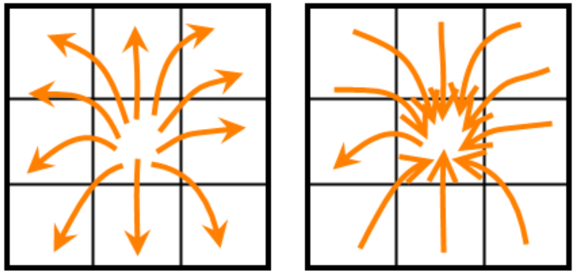
u can be a scalar or a vector



- Divergence ($\nabla \cdot$): Measure how the vectors are converging or diverging at a given location (volume density of the outward flux)

$$\nabla \cdot \boldsymbol{u} = \frac{\partial \boldsymbol{u}}{\partial x} + \frac{\partial \boldsymbol{u}}{\partial y} + \frac{\partial \boldsymbol{u}}{\partial z}$$

u can only be a vector



Source, Div(**u**) > 0

Sink, Div(**u**) < 0

Quick Math Review

- Laplacian (Δ or ∇^2): Divergence of the gradient

$$\nabla \cdot \nabla u = \nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

u can be a scalar or a vector

- Finite Difference: Derivative approximation

$$\frac{\partial u}{\partial x} = \frac{u_{i+1} - u_i}{x_{i+1} - x_i}$$

The Basic Equations

Computational Fluid Dynamics

- The Navier-Stokes Equations (developed around 1842 -1850)
  - Precise mathematical model of physical fluid flow in nature
  - Complicated, can be solved analytically only in very simple cases
  - No progress until 1950 when numerical algorithms started to appear
  - Can be very accurate (airplane aerodynamics, …)
  - Complex (difficult to implement)
  - Computational intensive (i.e. slow)
- For graphics applications
  - Simulation needs to look convincing
    - (does not always have to be physically accurate)
  - fast
  - simple to implement
  - stable (i.e. never “blows up”)

Notation

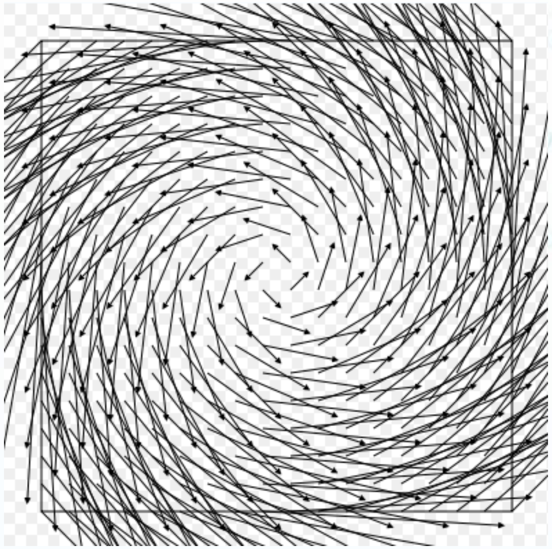
- $\vec{u}$: velocity with components (u,v,w)
- ρ: fluid density
- p: pressure
  - force per unit area that the fluid exerts on anything.
- $\vec{g}$: acceleration due to gravity or animator
- μ: dynamic viscosity

Navier-Stokes Equations

- Important Concepts
  - Conservation of Mass
  - Conservation of Momentum
  - Conservation of Volume
  - Internal and External forces
  - Viscosity
  - Boundary Conditions

Navier-Stokes Equations

- The state of the fluid in any point of time is modeled as velocity vector field:
  - a function that assigns velocity vector to any point in space



- The Navier-Stokes Equations:
  - precise description of evolution of velocity field over time

Navier-Stokes Equations

- "Momentum Equation"

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{g} + \nu \, \nabla \cdot \nabla \vec{u}$$

- "Incompressibility condition"

$$\nabla \cdot \vec{u} = 0$$

$\vec{u}$: Velocity

t: Time

ρ: Density

p: Pressure

$\vec{g}$: Gravity

ν: Kinematic Viscosity

The Momentum Equation

The Momentum Equation

- Just a specialized version of $\vec{F} = m\vec{a}$
- Let's build it up intuitively
- Imagine modeling a fluid with a bunch of particles (e.g. blobs of water)
  - A blob has a mass *m*, a volume *V*, and velocity $\vec{u}$
  - We'll write the acceleration $\vec{a}$ as $\frac{D\vec{u}}{Dt}$ (the "material derivative")

$$m\vec{a} = \vec{F}$$

$$m\frac{D\vec{u}}{Dt} = \vec{F}$$

  - What are the forces $\vec{F}$ that act on the fluid blob?

The Momentum Equation

- Forces on Fluids

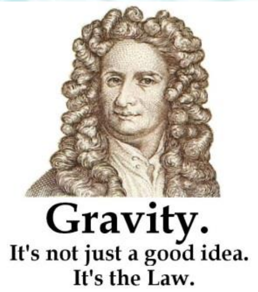
$$m\vec{a} = \vec{F}$$

$$m\frac{D\vec{u}}{Dt} = \vec{F}$$

$$m\frac{D\vec{u}}{Dt} = \vec{F}_{gravity} + \vec{F}_{pressure} + \vec{F}_{viscosity}$$

Forces on Fluids: Gravity

- Gravity: $F_{gravity}$



$$m\frac{D\vec{u}}{Dt} = m\vec{g} + \dots$$

- And a blob of fluid also exerts contact forces on its neighboring blobs (i.e. pressure)…

Forces on Fluids: Pressure

- Pressure: $F_{pressure}$
  - The “normal” contact force is pressure (force/area)
    - How blobs push against each other, and how they stick together
  - If pressure is equal in every direction, net force is zero… Important quantity is **pressure gradient**:

$$m\frac{D\vec{u}}{Dt} = m\vec{g} - V\nabla p + \ldots$$

    - We integrate the pressure over the blob’s volume
(which is equivalent to integrating the pressure over blob’s surface)

Forces on Fluids: Viscosity

- Viscosity: $F_{viscosity}$
  - What characterizes a viscous liquid?
    - "Thick", damped behavior
    - Higher resistance to flow

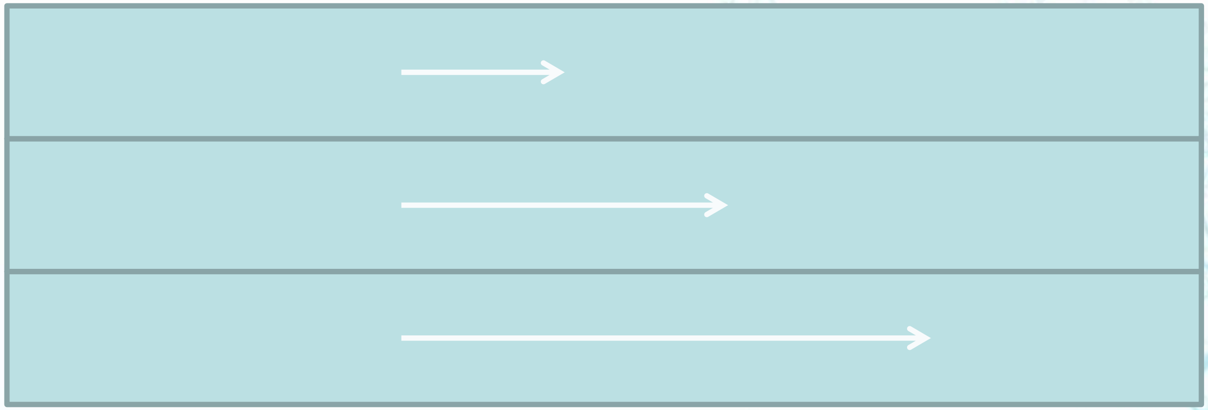
Forces on Fluids: Viscosity

- Viscosity: $F_{viscosity}$
  - Think of it as frictional part of contact force:
    a sticky (viscous) fluid blob resists other blobs moving past it
  - Viscous fluid resists deforming
  - Force that make particles move at average speed
  - For now, simple model is that we want velocity to be blurred/diffused/…
  - Blurring in PDE form comes from the Laplacian: $\nabla^2 \vec{u} = \nabla \cdot \nabla \vec{u}$

$$m \frac{D\vec{u}}{Dt} = m\vec{g} - V\nabla p + V\mu\nabla \cdot \nabla \vec{u}$$

Forces on Fluids: Viscosity

Loss of energy due to internal friction between molecules moving at different velocities.

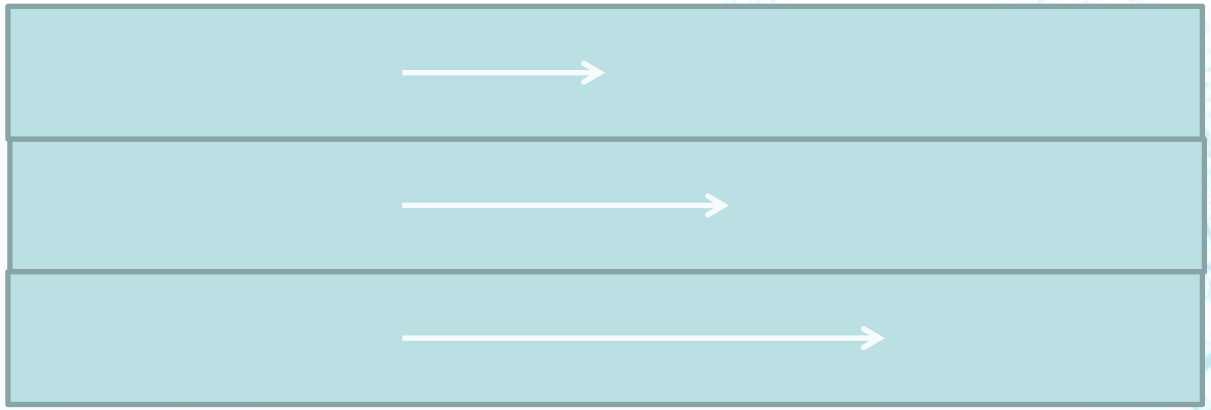


Interactions between molecules in different layers causes *shear stress* that…

- acts to oppose *relative* motion.
- causes an exchange of momentum

Forces on Fluids: Viscosity

Loss of energy due to internal friction between molecules moving at different velocities.

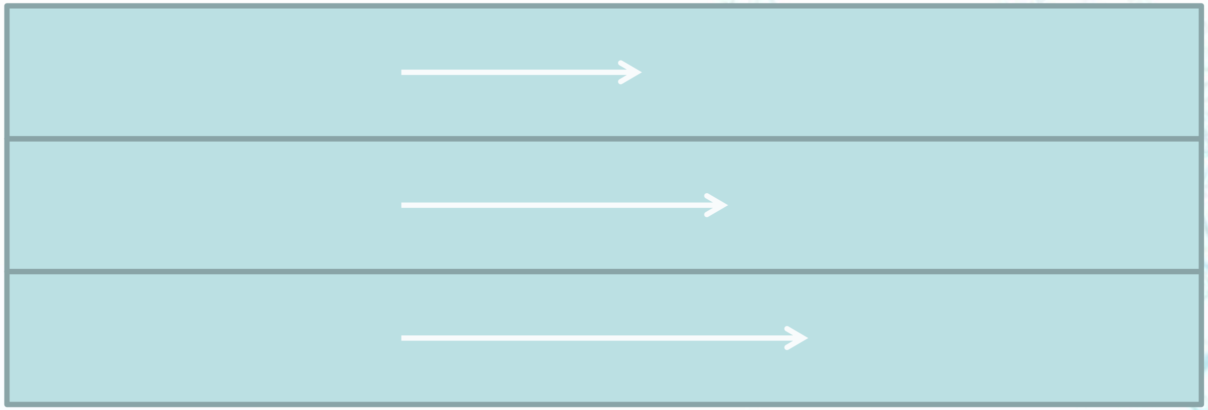


Interactions between molecules in different layers causes *shear stress* that…

- acts to oppose *relative* motion.
- causes an exchange of momentum

Forces on Fluids: Viscosity

Loss of energy due to internal friction between molecules moving at different velocities.

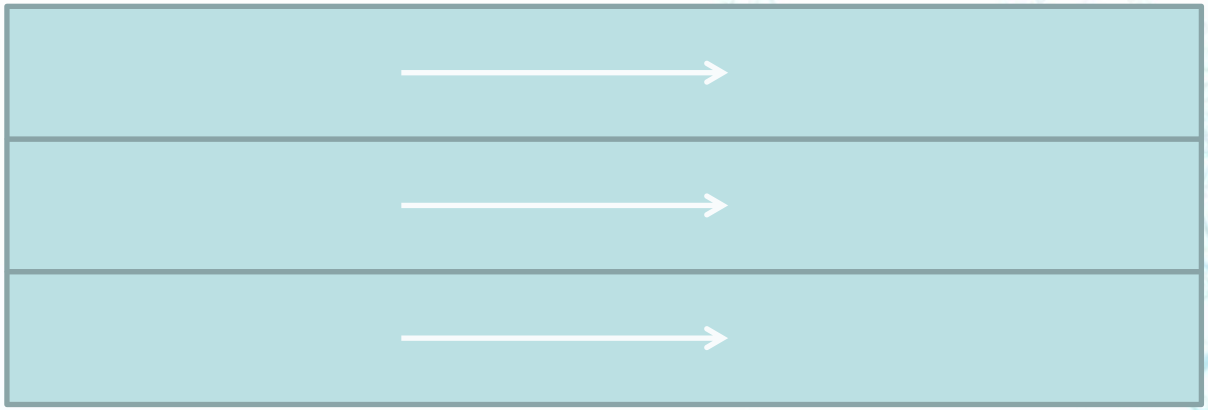


Interactions between molecules in different layers causes *shear stress* that…

- acts to oppose *relative* motion.
- causes an exchange of momentum

Viscosity

Loss of energy due to internal friction between molecules moving at different velocities.



Interactions between molecules in different layers causes *shear stress* that…

- acts to oppose *relative* motion.
- causes an exchange of momentum

Forces on Fluids: Viscosity

Imagine fluid particles with general velocities.

Each particle interacts with nearby neighbours, exchanging momentum.

Forces on Fluids: Viscosity

Amount of momentum exchanged is proportional to:

- Velocity gradient, ∇u.
- Viscosity coefficient, μ

For any closed region, net in/out flow of momentum:

$$\int_{\partial\Omega} \mu \nabla u \cdot n = \iint \mu \nabla \cdot \nabla u$$

So the viscosity force is: $\vec{F}_{viscosity} = V \mu \nabla \cdot \nabla \vec{u}$

Forces on Fluids: Viscosity

- The end result is a smoothing of the velocity.
  - This is exactly the action of the Laplacian operator

$$\nabla \cdot \nabla = \nabla^2$$

  - For scalar quantities, the same operator is used to model diffusion

The Continuum Limit

$$m\frac{D\vec{u}}{Dt} = m\vec{g} - V\nabla p + V\mu\nabla\cdot\nabla\vec{u}$$

- Model the world as a continuum:
  - # particles $\rightarrow\infty$
    Mass and volume $\rightarrow 0$
- In the limit we want $m\frac{D\vec{u}}{Dt} = \vec{F}$ to be more than 0 = 0:
  - Divide by mass

$$\frac{D\vec{u}}{Dt} = \vec{g} - \frac{V}{m}\nabla p + \frac{V}{m}\mu\nabla\cdot\nabla\vec{u}$$

The Continuum Limit – Con't

$$\frac{D\vec{u}}{Dt} = \vec{g} - \frac{V}{m}\nabla p + \frac{V}{m}\mu\nabla \cdot \nabla\vec{u}$$

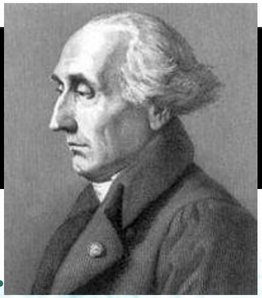
- The fluid density is $\rho = \dfrac{m}{V}$

$$\frac{D\vec{u}}{Dt} = \vec{g} - \frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla \cdot \nabla\vec{u}$$

$$\frac{D\vec{u}}{Dt} = \vec{g} - \frac{1}{\rho}\nabla p + \nu\,\nabla \cdot \nabla\vec{u} \qquad \nu = \frac{\mu}{m} \quad \Rightarrow \text{dynamic viscosity}$$

- The only weird thing is $\dfrac{D\vec{u}}{Dt}$ …

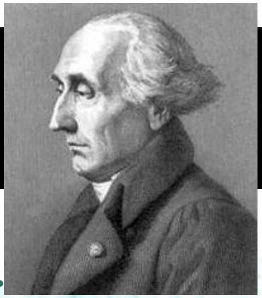
Lagrangian vs. Eulerian

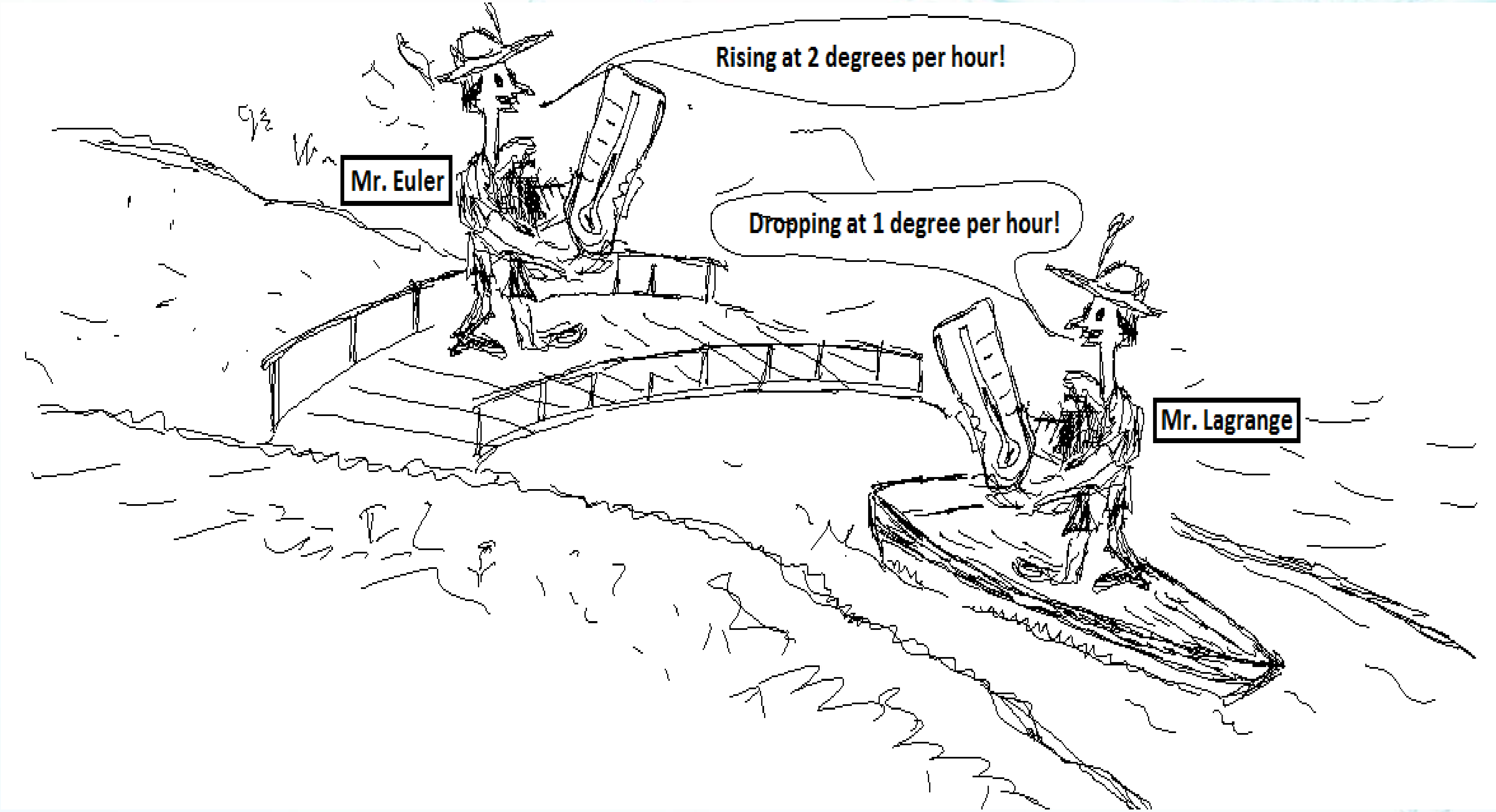


- **Lagrangian viewpoint**:
  - Treat the world like a particle system
  - Label each speck of matter, track where it goes (velocity, accel, etc.)
  - Point of reference moves with the material
- **Eulerian viewpoint**:
  - Point of reference is stationary
  - Measure stuff as it flows past
- **Example**: Measuring temperature of wind
  - Lagrangian: weather balloon, floating with the wind

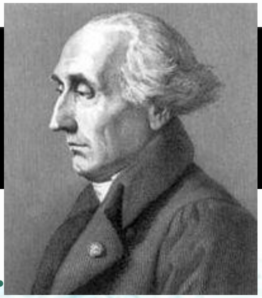
  - Eulerian: weather station on ground, wind blows past

Eulerian vs. Lagrangian

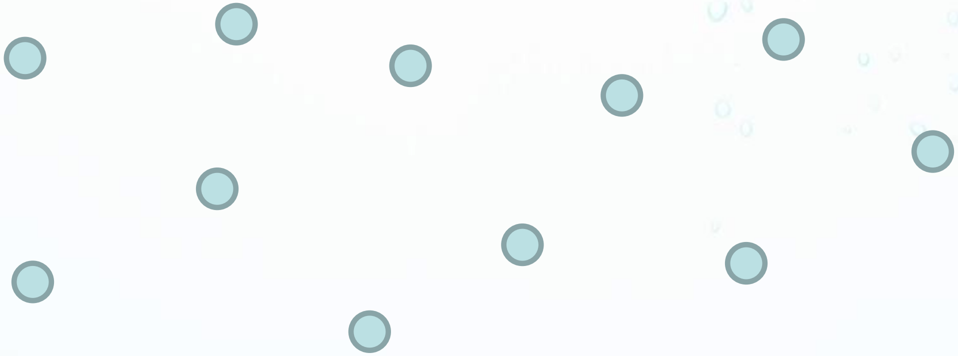




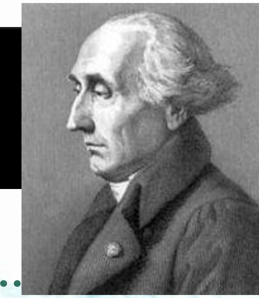
Eulerian vs. Lagrangian



- **Lagrangian**: Wind comprised of a set of *moving particles*, each with a temperature value at a particular point in space



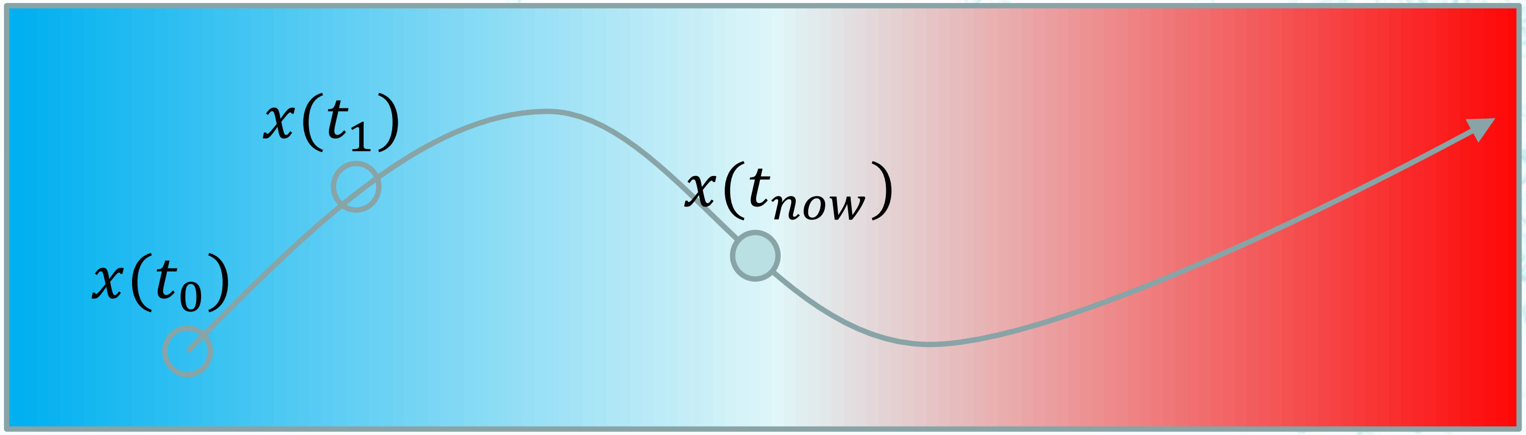
Eulerian vs. Lagrangian



- **Eulerian:** A fixed grid of temperature (sensor) values, that the wind *flows through.*

Relating Eulerian and Lagrangian

Consider the temperature $T(x,t)$ at a point following a given path, $x(t)$.



Two ways temperature changes:

1. Heating/cooling occurs at the current point.
2. Following the path, the point moves to a cooler/warmer location.

Time Derivatives

Mathematically:

$$\frac{D}{Dt}T(x,t) = \frac{\partial T}{\partial t} + \frac{\partial T}{\partial x}\frac{\partial x}{\partial t}$$

Chain rule!

$$= \frac{\partial T}{\partial t} + \nabla T \cdot \frac{\partial x}{\partial t}$$

Definition of ∇

$$= \frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T$$

Choose $\frac{\partial x}{\partial t} = u$

Material Derivative

$\frac{D}{Dt}$ is called the ***Material Derivative***

$$\overbrace{\frac{DT}{Dt}}^{\text{Change at a point moving with the velocity field.}} = \overbrace{\frac{\partial T}{\partial t}}^{\text{Change due to movement.}} + \underbrace{\boldsymbol{u} \cdot \nabla T}_{\text{Change at the current (fixed) point.}}$$

Material Derivative

- General Case
  - We have fluid moving in a velocity field u
  - It possesses some quality (i.e. property) q
  - At an instant in time t and a position in space x, the fluid at x has property value q(x,t)
  - How fast is that blob of fluid's q changing w.r.t time?
  - Answer:
    - the Material Derivative: $\frac{Dq}{Dt}$

Material Derivative

- Writing D/Dt Out
  - We can explicitly write it out from components:

$$\frac{Dq}{Dt} = \frac{\partial q}{\partial t} + \vec{u} \cdot \nabla q$$

$$= \frac{\partial q}{\partial t} \quad + \quad u\frac{\partial q}{\partial x} + v\frac{\partial q}{\partial y} + w\frac{\partial q}{\partial z}$$

Material Derivative

- This holds even if the vector field is velocity itself:

$$\frac{D\vec{u}}{Dt} = \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla u$$

$$\begin{bmatrix} Du/Dt \\ Dv/Dt \\ Dw/Dt \end{bmatrix} = \begin{bmatrix} \partial u/\partial t + \vec{u} \cdot \nabla u \\ \partial v/\partial t + \vec{u} \cdot \nabla v \\ \partial w/\partial t + \vec{u} \cdot \nabla w \end{bmatrix}$$

- Nothing different about this, just that the fluid blobs are moving at the velocity they're carrying.

Momentum Equation

- Replacing the material derivative in the previous Navier Stokes equation

$$\frac{D\vec{u}}{Dt} = \vec{g} - \frac{1}{\rho}\nabla p + \nu\, \nabla \cdot \nabla \vec{u}$$

with $$\frac{D\vec{u}}{Dt} = \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u}$$

- Yields the standard form of the Momentum Equation

$$\boxed{\frac{\partial \vec{u}}{\partial t} = -\vec{u} \cdot \nabla \vec{u} + \vec{g} - \frac{1}{\rho}\nabla p + \nu\, \nabla \cdot \nabla \vec{u}}$$

The Incompressibility Condition

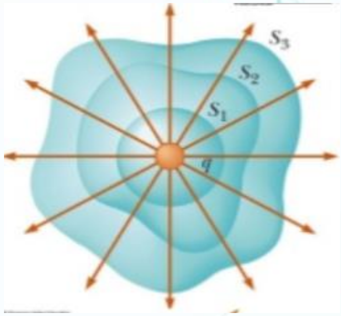
Compressibility

- Real fluids are compressible
- Shock waves, sound waves, pistons…
  - Note: liquids change their volume as well as gases, otherwise there would be no sound underwater
- But this is nearly irrelevant for animation
  - Shocks move too fast to normally be seen (easier/better to hack in their effects)
  - Acoustic waves usually have little effect on visible fluid motion
  - Pistons compressing gas in a cylinder is not of interest

Incompressibility

- Rather than having to simulate acoustic and shock waves, eliminate them from our model: assume fluid is **incompressible**
  - Turn stiff system into a constraint, just like rigid bodies!
- If you fix your eyes on any volume of space, volume of fluid in = volume of fluid out:

$$\iint_{\partial\Omega} \vec{u} \cdot \hat{n} = 0$$



Incompressibility

Intuitively, to be incompressible net flow into/out of a given region is zero (i.e. no sources or sinks)

Integrate the flow across the boundary of a closed region:

$$\int_{\partial\Omega} \boldsymbol{u} \cdot \boldsymbol{n} = 0$$

Divergence

- Let's use the divergence theorem:

$$\frac{dV}{dt} = \int_{\partial\Omega} \vec{u} \cdot \hat{n} = \iint_{\Omega} \nabla \cdot \vec{u} = 0$$

- So for any region, the integral of $\nabla \cdot \vec{u}$ is zero
  - Therefore, for it to be zero everywhere:

$$\boxed{\nabla \cdot \vec{u} = 0}$$

- Incompressible Flow
  - Density stays constant
  - Divergence: Net flow in or out of a volume
  - When divergence = 0, no sources or sinks

Inviscid Fluids

Dropping Viscosity

- In most fluid scenarios, viscosity term is small
- As a result, convenient to drop it from the equations:
  - Zero viscosity: called "inviscid"
  - Inviscid Navier-Stokes = "Euler equations"
- Numerical simulation typically makes errors that resemble physical viscosity, so we have the visual effect of it anyway
  - Called "numerical dissipation"
  - For animation: often numerical dissipation is larger than the true physical viscosity!

Aside: Some values of interest

- Air
  - Dynamic viscosity of air: $\mu_{air} \approx 1.8 \times 10^{-5}\ Ns/m^2$
  - Density of air: $\rho_{air} \approx 1.3\, kg/m^3$

- Water
  - Dynamic viscosity of water: $\mu_{water} \approx 1.1 \times 10^{-3}\ Ns/m^2$
  - Density of water: $\rho_{water} \approx 1000\, kg/m^3$

- The ratio, μ/ρ ("kinematic viscosity") is what's important for the motion of the fluid…

  … air is 10 times more viscous than water!

Inviscid Navier Stokes equations

- a.k.a. the incompressible Euler equations:

$$\frac{\partial \vec{u}}{\partial t} = -\vec{u} \cdot \nabla \vec{u} + \vec{g} - \frac{1}{\rho} \nabla p$$

$$\nabla \cdot \vec{u} = 0$$

Boundary Conditions

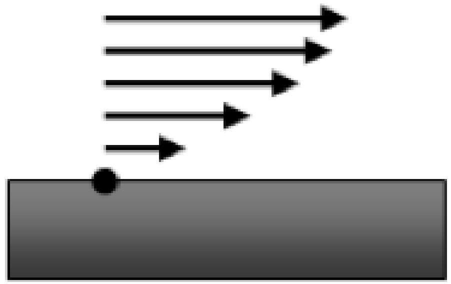
Boundary Conditions

- We know what’s going on inside the fluid: what about at the surface?
- Three types of surface
  - Solid wall: fluid is adjacent to a solid body
  - Free surface: fluid is adjacent to nothing
    (e.g. water is so much denser than air, might as well forget about the air)
  - Other fluid: possibly discontinuous jump in quantities (density, …)

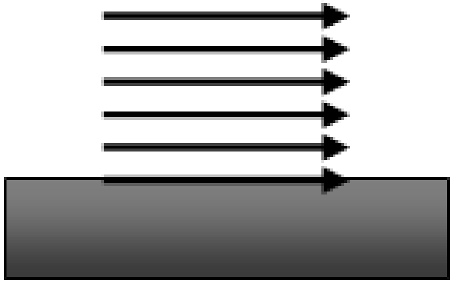
Boundary Conditions

- Solid walls - in contact with solid
  - Fluid should not be flowing into or out of it
  - So, normal component of velocity should be 0
  - "No-slip” or “free-slip” condition

- Free surfaces
  - Where we stop modeling the fluid
  - Set pressure to 0
  - Don't control velocity in any particular way



No-slip



Free-slip

Solid Wall Boundaries

- No fluid can enter or come out of a solid wall:

$$\vec{u} \cdot \hat{n} = \vec{u}_{solid} \cdot \hat{n}$$

- For common case of $\vec{u}_{solid} = 0$: $\vec{u} \cdot \hat{n} = 0$
  - Sometimes called the “no-stick” condition, since we let fluid slip past tangentially
- For viscous fluids, can additionally impose “no-slip” condition:

$$\vec{u} = \vec{u}_{solid}$$

Free Surface

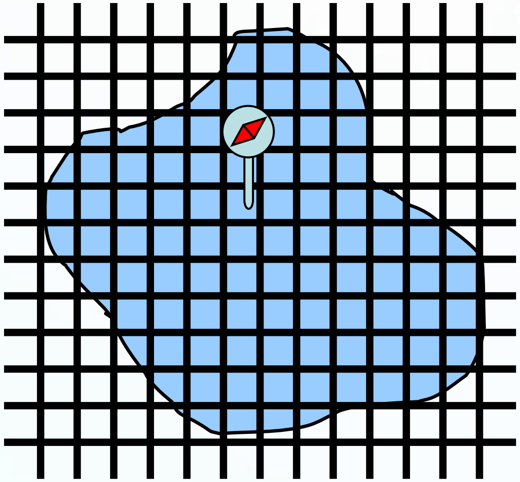
- Neglecting the other fluid, we model it simply as pressure = constant
  - Since only pressure gradient is important, we can choose the constant to be zero:

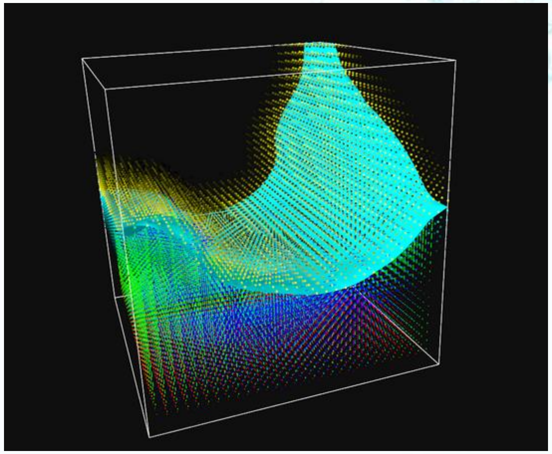
$$p = 0$$

Numerical Simulation Overview

Eulerian Approach

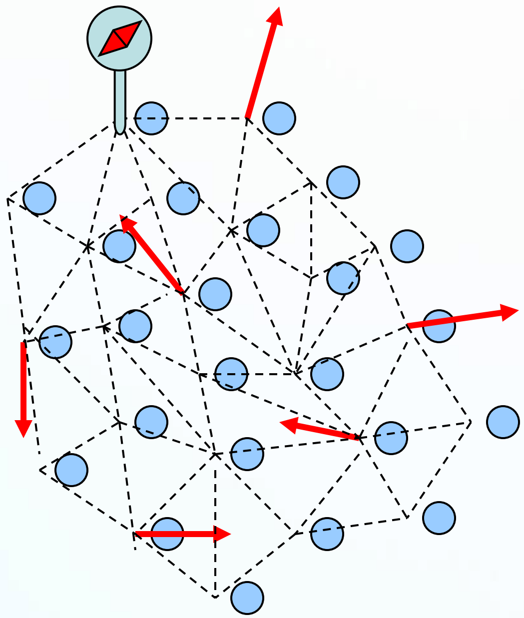
- Discretize the domain using finite differences
- Define scalar & vector fields on the grid
- Use the operator splitting technique to solve each term separately

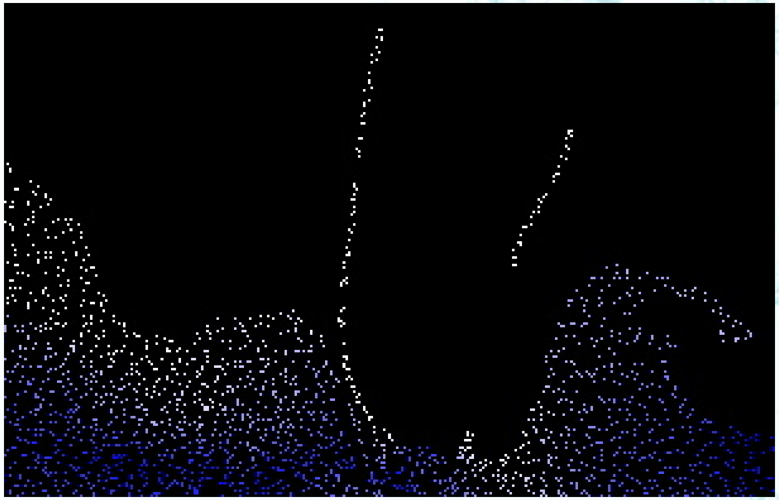




Lagrangian Approach

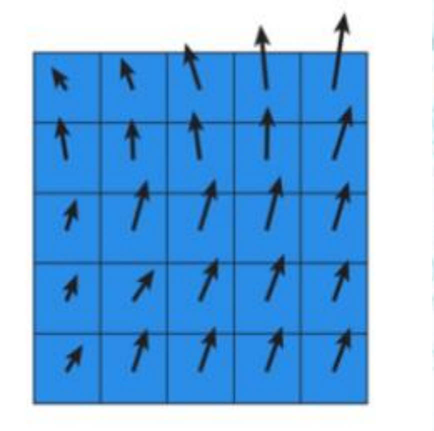
- Treat the fluid as discrete particles
- Apply interaction forces (i.e. pressure/viscosity) according to certain pre-defined smoothing kernels



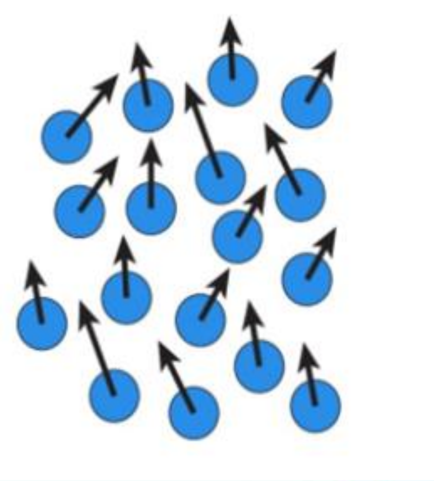


Eulerian vs Lagragian - Tradeoffs

- **Eulerian**
  - Fast, regular computation
  - Easy to represent, e.g. smooth surfaces
  - Simulation “trapped” in grid
  - Grid causes “numerical dissapation (i.e. diffusion)
  - Need to understand Navier-Stokes PDEs



- **Lagrangian**
  - Conceptually easy (like polygon soup)
  - Resolution/domain not limited by grid
  - Good particle distribution can be tough
  - Finding neighbors can be expensive



Splitting

- We have lots of terms in the momentum equation: a pain to handle them all simultaneously

$$\frac{\partial \vec{u}}{\partial t} = -\vec{u} \cdot \nabla \vec{u} + \vec{g} - \frac{1}{\rho} \nabla p$$

- Instead we split up the equation into its terms, and integrate them one after the other
  - Makes for easier software design too:
    a separate solution module for each term
- First order accurate in time

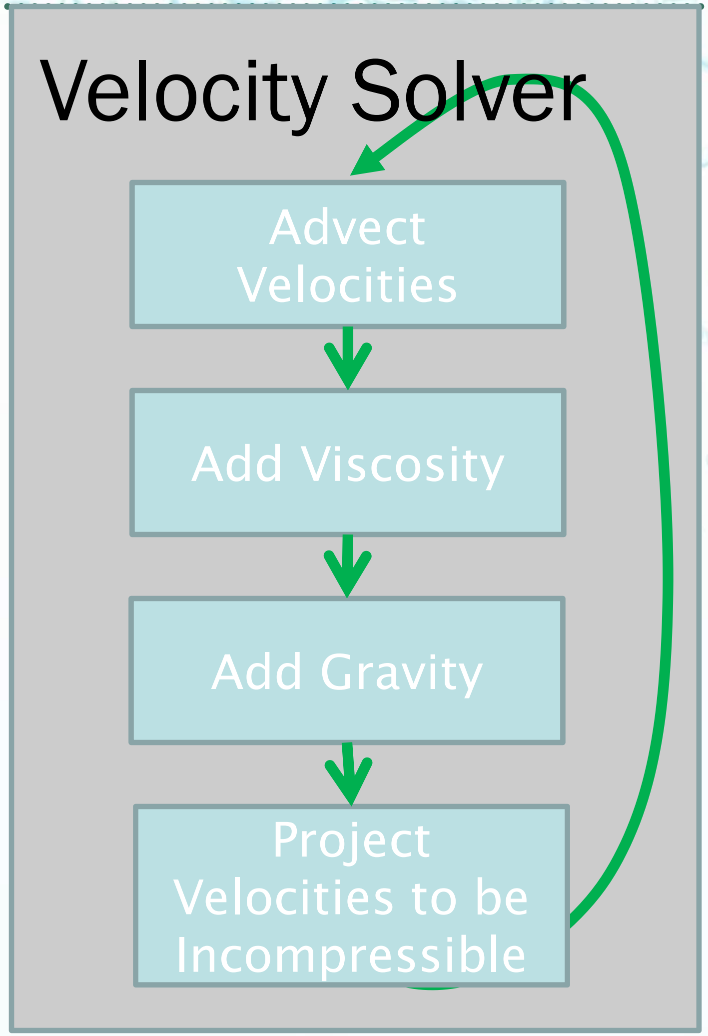
A Splitting Example

- Say we have a differential equation

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

- We can solve the component parts:
  - SolveF(q,Δt) solves $\frac{dq}{dt} = f(q)$ for time Δt
  - SolveG(q,Δt) solves $\frac{dq}{dt} = g(q)$ for time Δt
- Then put them together to solve the original equation:
  - $q^* = \text{SolveF}(q^n, \Delta t)$
  - $q^{n+1} = \text{SolveG}(q^*, \Delta t)$

The Big Picture



Splitting Momentum

- We have three terms: $$\frac{\partial \vec{u}}{\partial t} = -\vec{u} \cdot \nabla \vec{u} + \vec{g} - \frac{1}{\rho} \nabla p$$
- First term: **advection** $$\frac{\partial \vec{u}}{\partial t} = -\vec{u} \cdot \nabla \vec{u}$$
  - Moves the fluid through its velocity field
- Second term: **gravity** $$\frac{\partial \vec{u}}{\partial t} = \vec{g}$$
- Final term: **pressure update** $$\frac{\partial \vec{u}}{\partial t} = -\frac{1}{\rho} \nabla p$$
  - Need to compute pressure to make the fluid incompressible: $$\nabla \cdot \vec{u} = 0$$

Pressure Projection - Derivation

- Updating velocity using pressure term: $\frac{\partial \vec{u}}{\partial t} = -\frac{1}{\rho}\nabla p$
  - Also requires new velocity satisfy incompressible condition: $\nabla \cdot \vec{u} = 0$
- Discretize $\frac{\partial \vec{u}}{\partial t} \cong \frac{(\vec{u}_{new} - \vec{u}_{old})}{\Delta t} = -\frac{1}{\rho}\nabla p$

  rearranging $\vec{u}_{new} = \vec{u}_{old} - \frac{\Delta t}{\rho}\nabla p$
- Plugging $\vec{u}_{new}$ into $\nabla \cdot \vec{u}_{new} = 0$

  yields

$$\nabla \cdot \left(\vec{u}_{old} - \frac{\Delta t}{\rho}\nabla p \right) = 0$$

Pressure Projection

Implementation:

1) Solve the following linear system on the grid for the pressure *p:*

$$\nabla \cdot \left(\vec{u}_{old} - \frac{\Delta t}{\rho} \nabla p \right) = 0 \qquad \boxed{\Rightarrow \quad \frac{\Delta t}{\rho} \nabla \cdot \nabla p = \nabla \cdot \vec{u}_{old}}$$

2) Update grid velocity with:

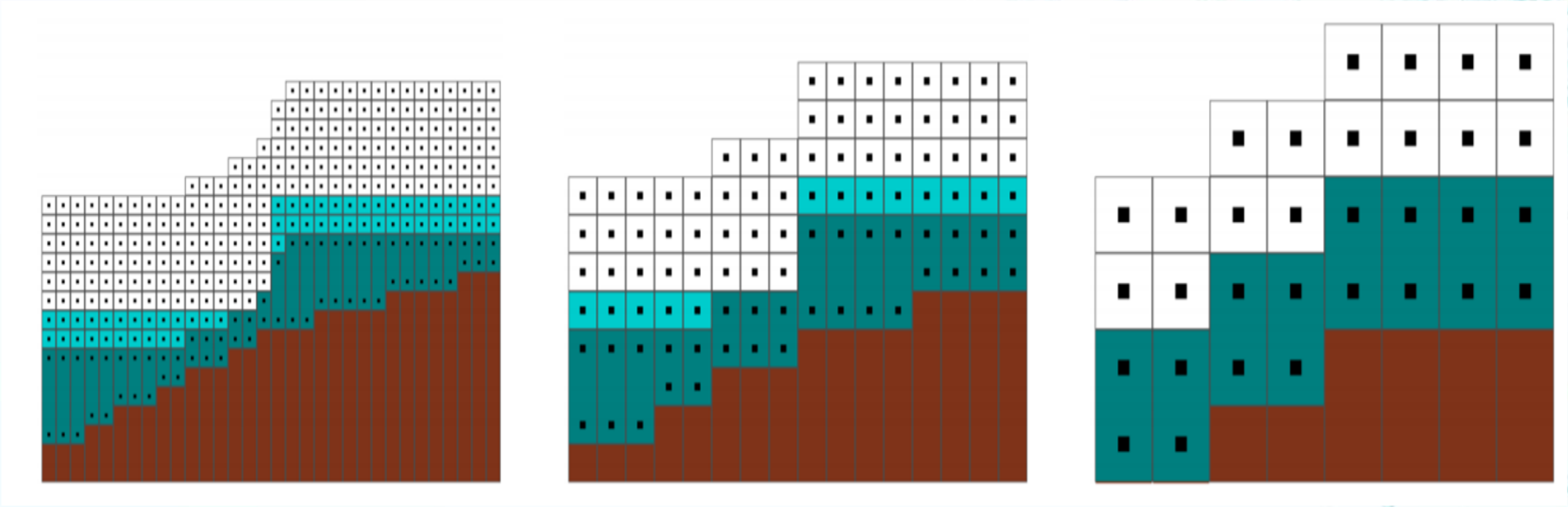
$$\boxed{\vec{u}_{new} = \vec{u}_{old} - \frac{\Delta t}{\rho} \nabla p}$$

Eulerian Approach

- That’s our general strategy in time; what about space?
- We’ll begin with a fixed Eulerian Approach
  - Trivial to set up
  - Easy to approximate spatial derivatives
  - Particularly good for the effect of pressure
- Disadvantage: advection doesn’t work so well
  - Later: particle methods that fix this

Eulerian Grid

Used to track properties and attributes at fixed points inside the fluid.



A Simple Grid

- We could put all our fluid variables at the nodes of a regular grid
- But this causes some major problems
- In 1D: incompressibility means: $\frac{\partial u}{\partial x} = 0$
- Central difference approximation at a grid point:

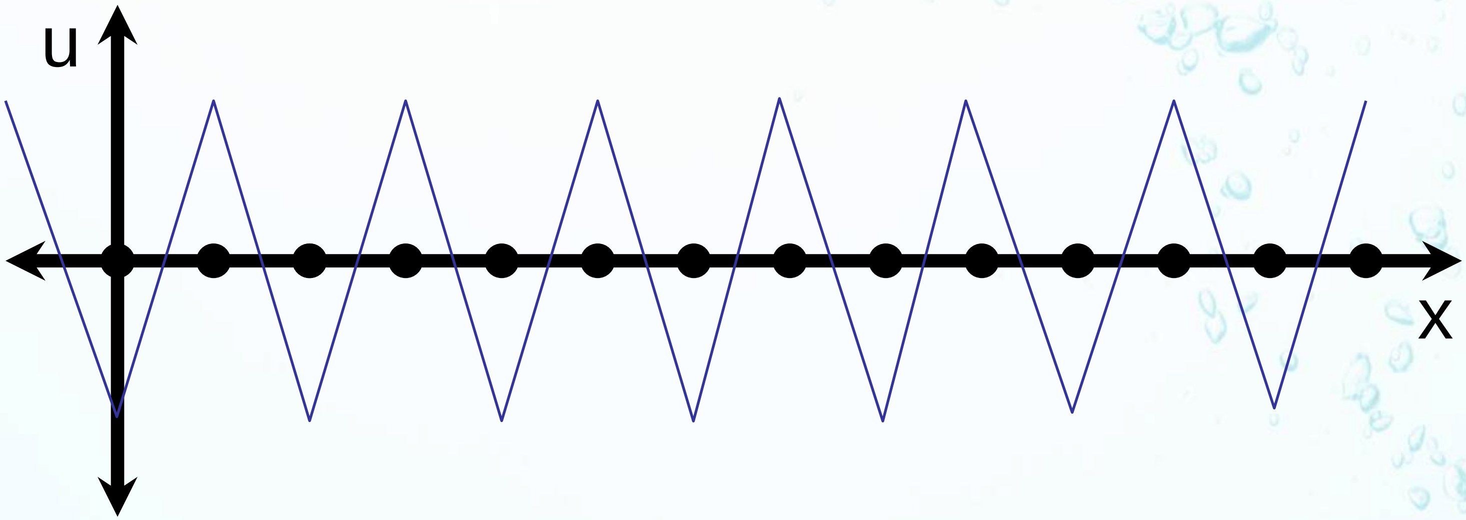
$$\frac{u_{i+1} - u_{i-1}}{2\Delta x} = 0$$

- Note the velocity at the grid point isn't involved!

A Simple Grid Disaster

- The only valid solution to $\dfrac{\partial u}{\partial x} = 0$ is u = constant

- But our numerical approximation can generate other solutions:



Staggered Grids

- Problem is solved if we don't skip over grid points
- To make it unbiased, we **stagger** the grid:
  - put velocities halfway between grid points
- In 1D, we estimate divergence at a grid point as:

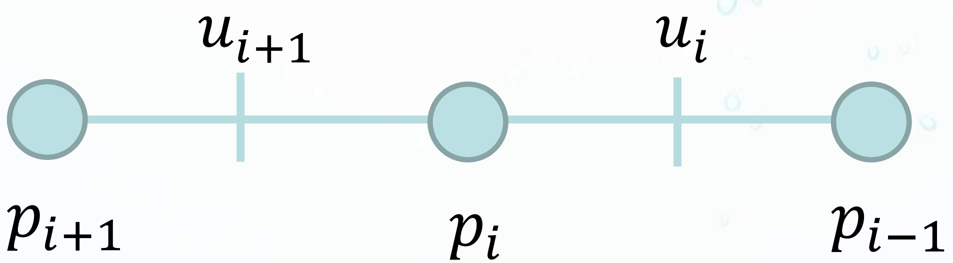
$$\frac{\partial u}{\partial x}(x_i) \approx \frac{u_{i+\frac{1}{2}} - u_{i-\frac{1}{2}}}{\Delta x}$$

- Problem solved!

Incompressibility

Pressure Projection: $$\frac{\Delta t}{\rho} \nabla \cdot \nabla p = \nabla \cdot \vec{u}_{old}$$

Discretize with finite differences:



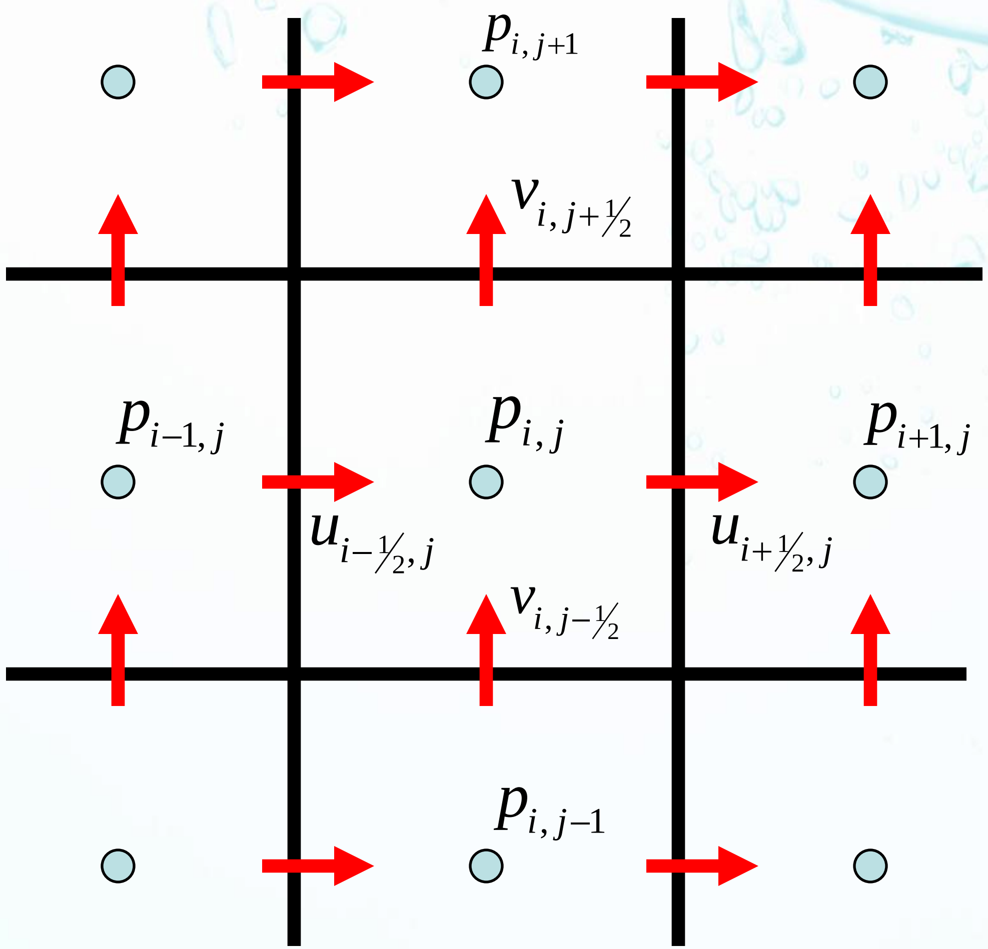
e.g., in 1D:

$$\frac{\Delta t}{\rho}\left(\frac{\frac{p_{i+1}-p_i}{\Delta x}-\frac{p_i-p_{i-1}}{\Delta x}}{\Delta x}\right)=\frac{u_{i+1}{}^{old}-u_i{}^{old}}{\Delta x}$$

The MAC Grid

- From the Marker-and-Cell (MAC) method [Harlow&Welch'65]
- A particular staggering of variables in 2D/3D that works well for incompressible fluids:
  - Grid cell (i,j,k) has pressure $p_{i,j,k}$ at its center
  - x-part of velocity $u_{i+1/2,jk}$ in middle of x-face between grid cells (i,j,k) and (i+1,j,k)
  - y-part of velocity $v_{i,j+1/2,k}$ in middle of y-face
  - z-part of velocity $w_{i,j,k+1/2}$ in middle of z-face

MAC Grid in 2D



The MAC Grid

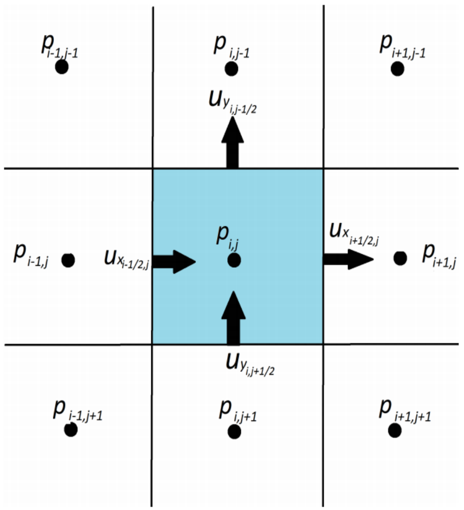


Figure 3: *Two Dimensional MAC Cell*

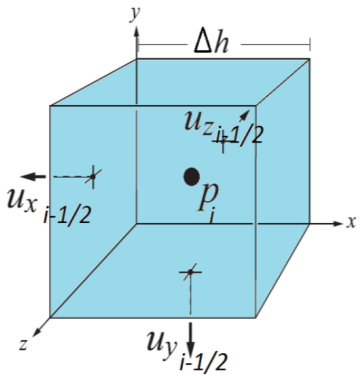
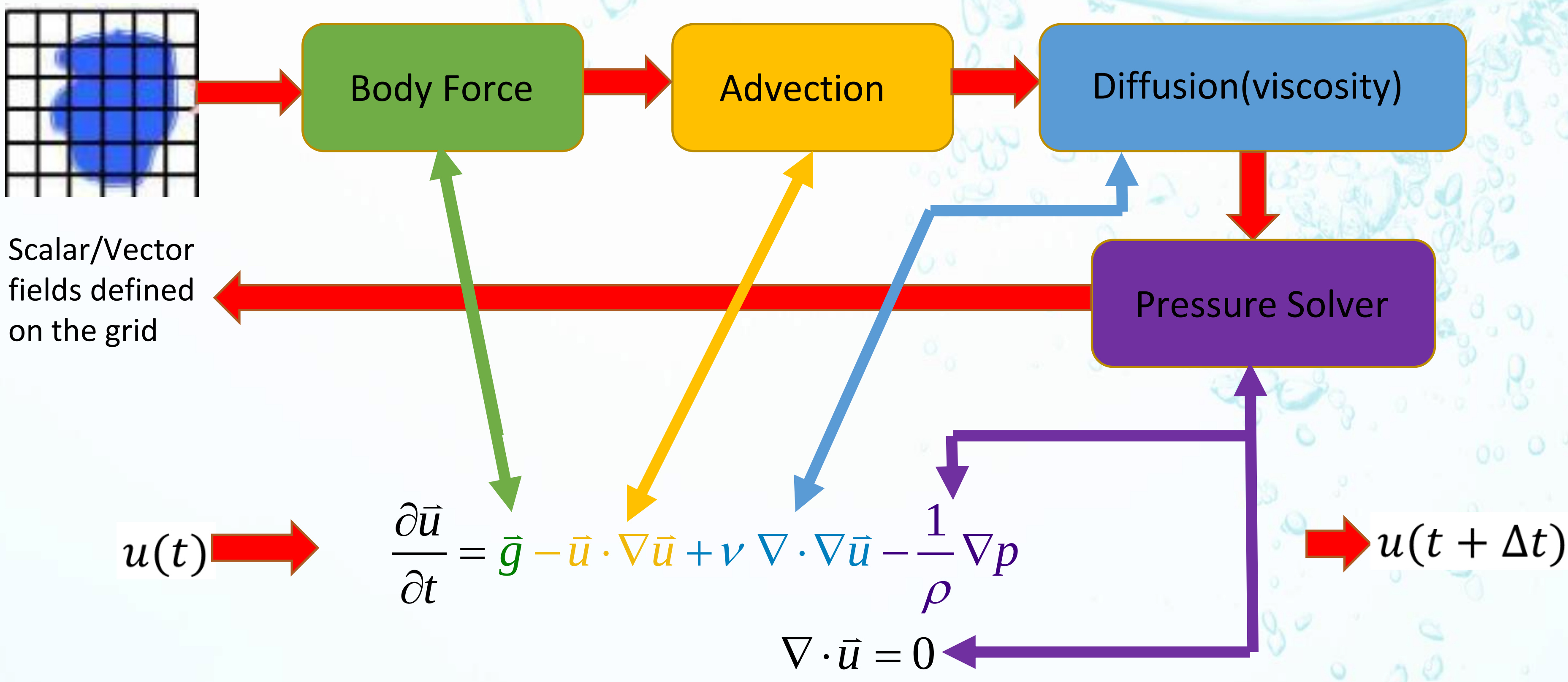


Figure 4: *Three Dimensional MAC Cell*

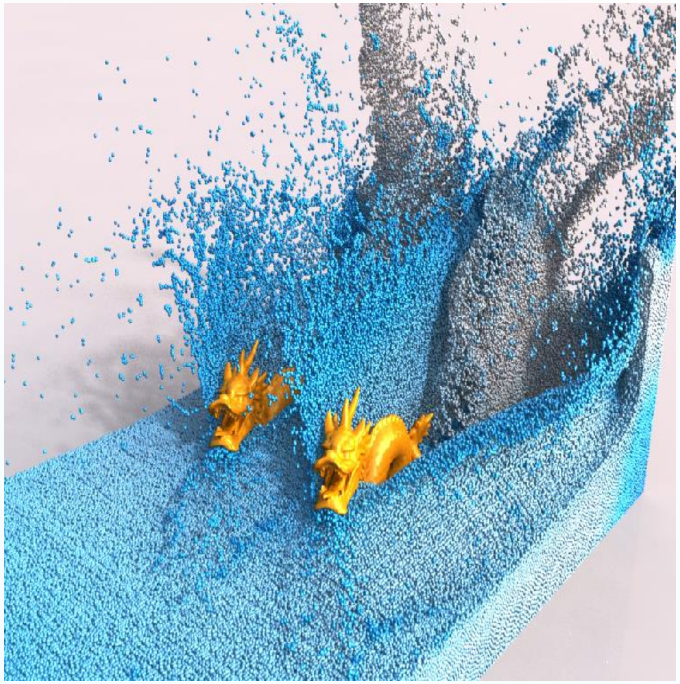
$$\frac{\partial w}{\partial x}_i \approx \frac{w_{i+\frac{1}{2}} - w_{i-\frac{1}{2}}}{2\Delta x}$$

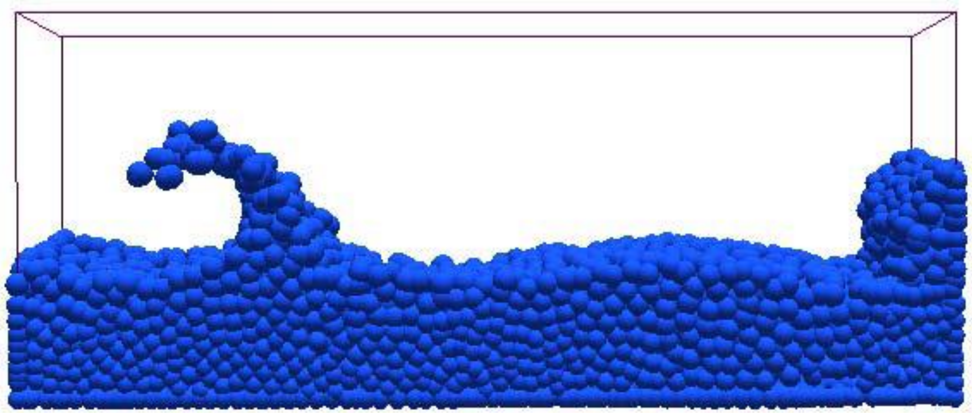
Eulerian Simulation – Main Loop



Lagrangian Approach

- Particle-Based
  - Simulate fluid as discrete particles





Lagrangian Approach

- Spherical Particle Hydrodynamics (SPH)

For all particles P_i:

$$\frac{\partial \vec{V}}{\partial t}_i = \vec{A}_i^{pressure} + \vec{A}_i^{viscosity} + \vec{A}_i^{gravity} + \vec{A}_i^{external}$$

- $\vec{X}$ Position
- $\vec{V}$ Velocity
- M Mass
- d Density
- ρ Pressure
- $\vec{C} =< C_{red}, C_{green}, C_{blue} >$ Color
- $\vec{F}$ Force

- Initialize all particles
- Set $t = 0$
- Choose a Δt
- for i from 0 to n

    for j from 1 to $numparticles$

        Get list L_j of neighbors for P_j

        Calculate $Density_j$ for P_j using L_j

        Calculate $Pressure_j$ for P_j using L_j

        Calculate acceleration A_j for P_j using $Density_j$ and $Pressure_j$

        Move P_j using A_j and Δt using Euler step

    $t = t + \Delta t$

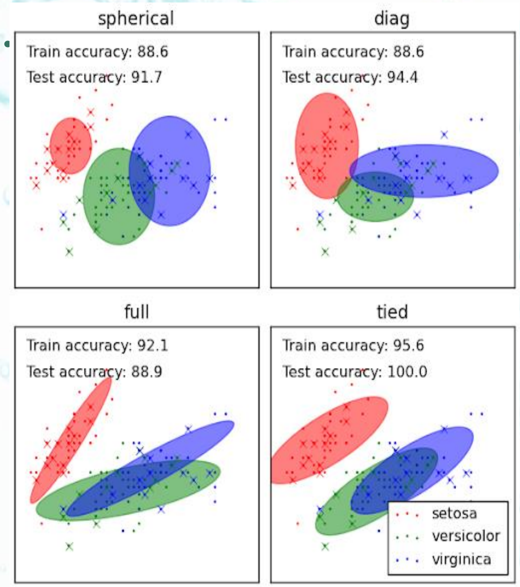
- Cleanup all data structures
- Exit

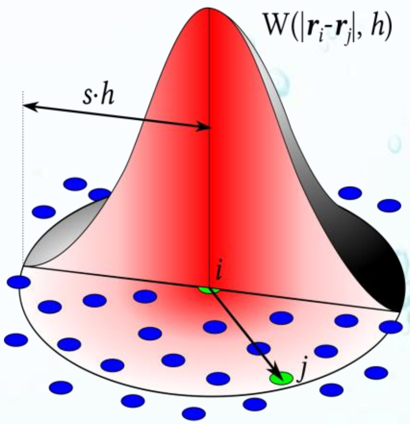
Lagrangian Approach

- Kernel Function

$$\sum_{j \neq i}^{n} M_j W_{R_{ij}}$$

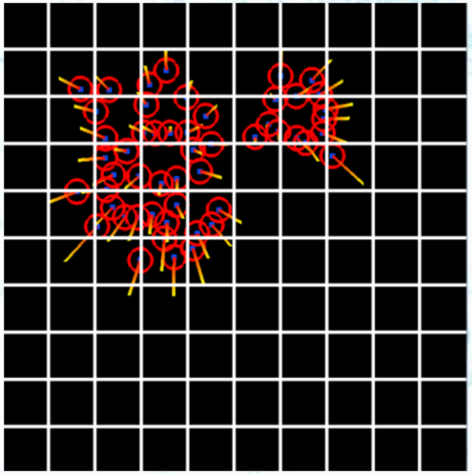
$$W(d) = \frac{1}{\pi^{\frac{3}{2}} h^3} \exp\left(\frac{r^2}{h^2}\right)$$



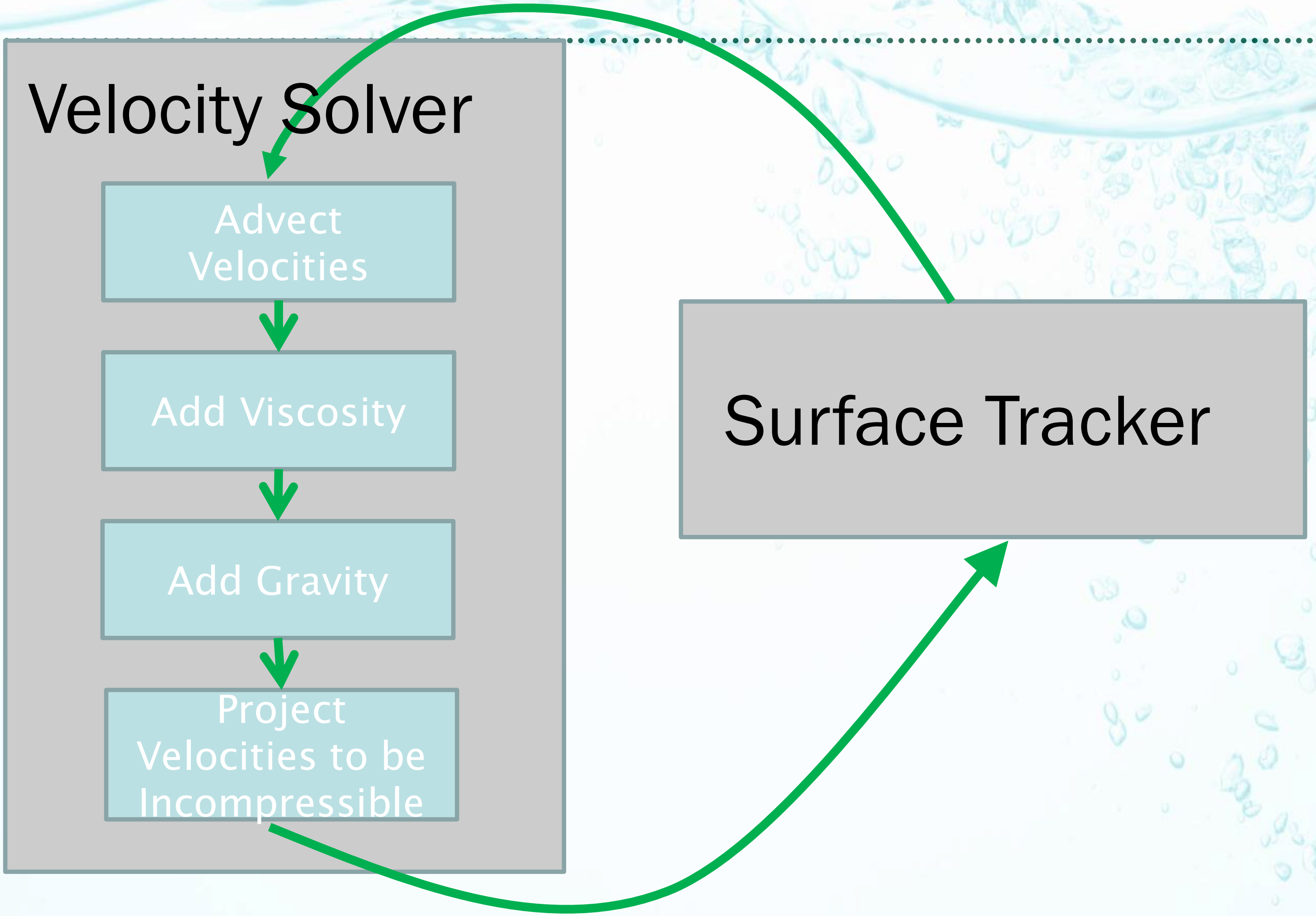


Lagrangian Approach

- Finding neighboring particles
  - We can divide our simulation space in 2D or 3D grid.
  - One must only examine 9 grid cells in the 2D case, or 27 grid cells in the 3D case.
  - For any grid cells far enough away, our kernel function will evaluate to 0 and their contributions will not be included on the current particle.
  - Each particle can be simulated in a separate thread with relative ease., therefore high performance SPH implementations are done on the GPU.



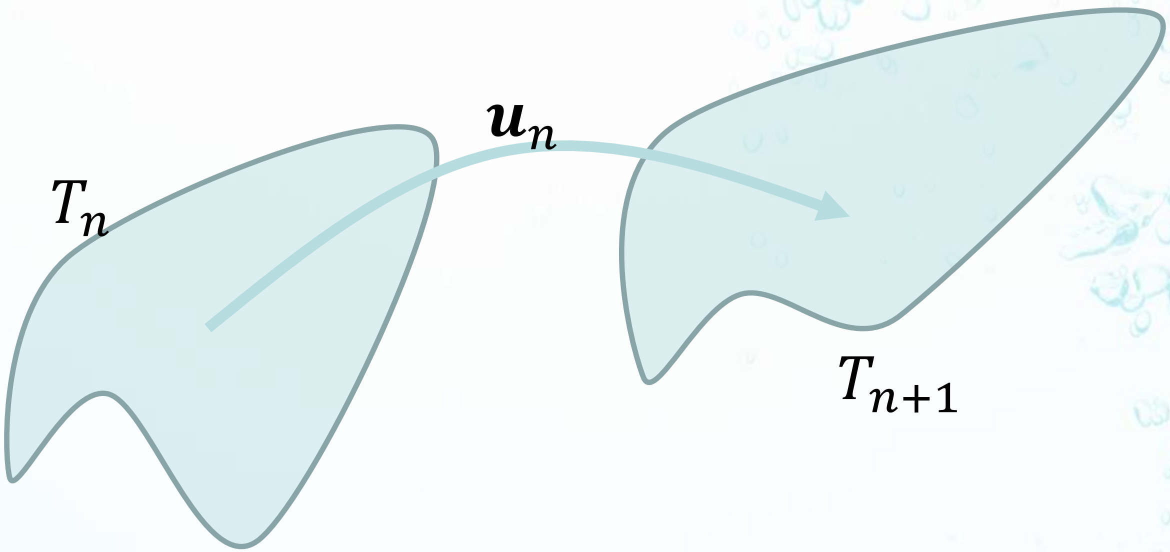
What about liquids?



Surface Tracker

Given: liquid surface geometry, velocity field, timestep

Compute: new surface geometry by advection.



Surface Tracker

Ideally:

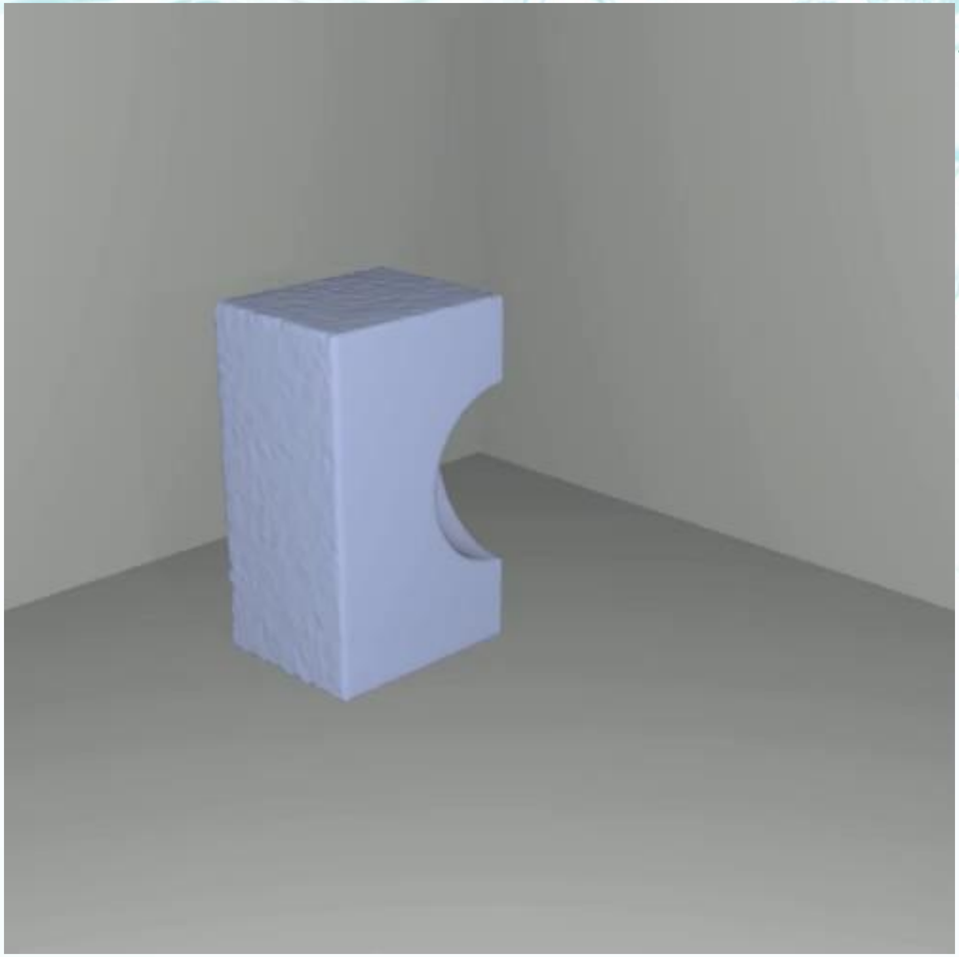
- Efficient
- Accurate
- Handles merging/splitting (topology changes)
- Conserves volume
- Retains small features
- Smooth surface for rendering
- Provides convenient geometric operations
- Easy to implement…

Very hard (impossible?) to do all of these at once.

Surface Tracking Options

1. Marker Particles
2. Level sets
3. Triangle meshes
4. Hybrids (many of these)

Particles

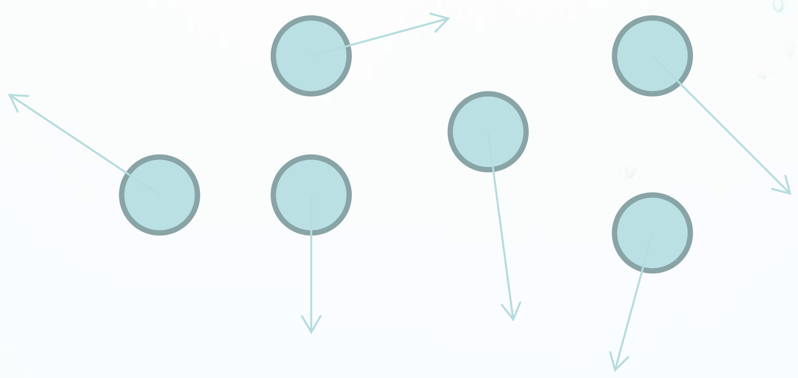


[Zhu & Bridson 2005]

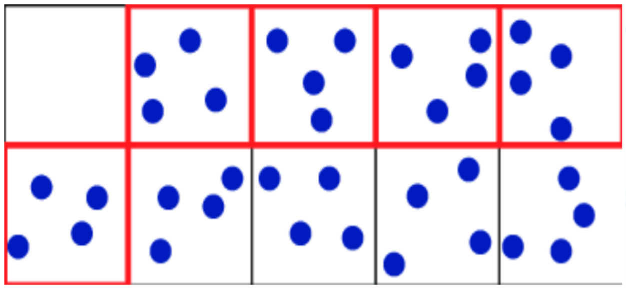
Particles

Perform passive Lagrangian advection on each particle.

For rendering, need to reconstruct a surface.

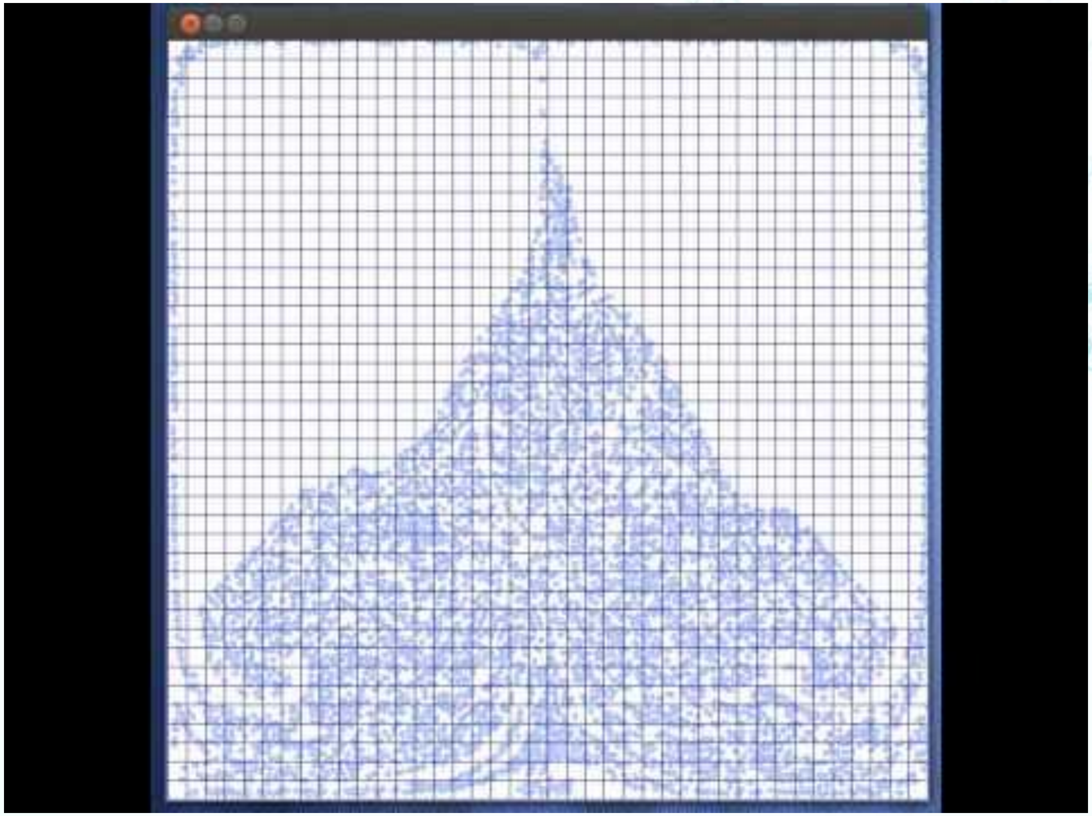


Marker Particles



- Any cell not containing marker particles (blue dots) is identified as an empty cell.
- Cells with at least one marker particle and at least one common boundary with an empty cell are the interface cells (marked in red).
- Cells accommodating at least one marker particle and surrounded only by other cells containing marker particles are marked as fluid cells.

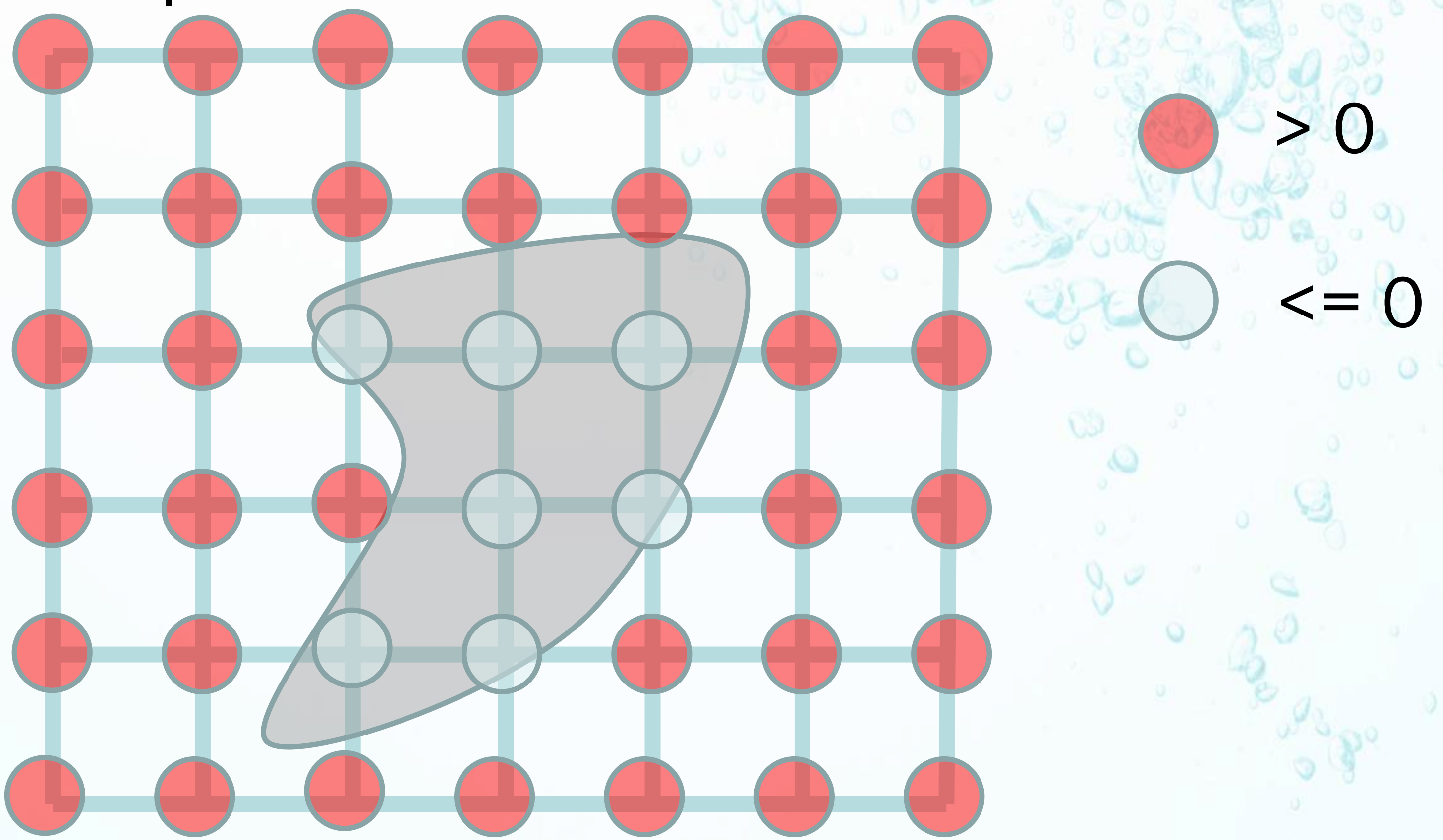
Marker Particles



Level sets

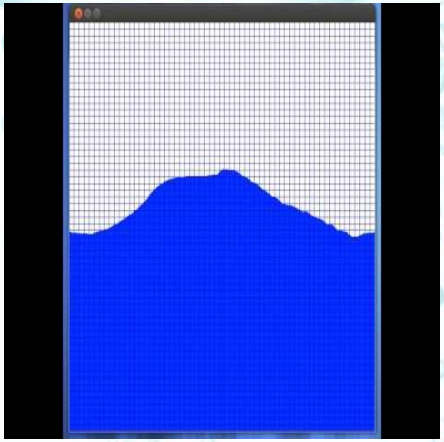
Each grid point stores *signed* distance to the surface (inside <= 0, outside > 0).

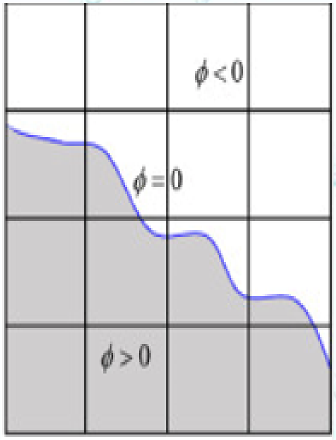
Surface is interpolated zero isocontour.



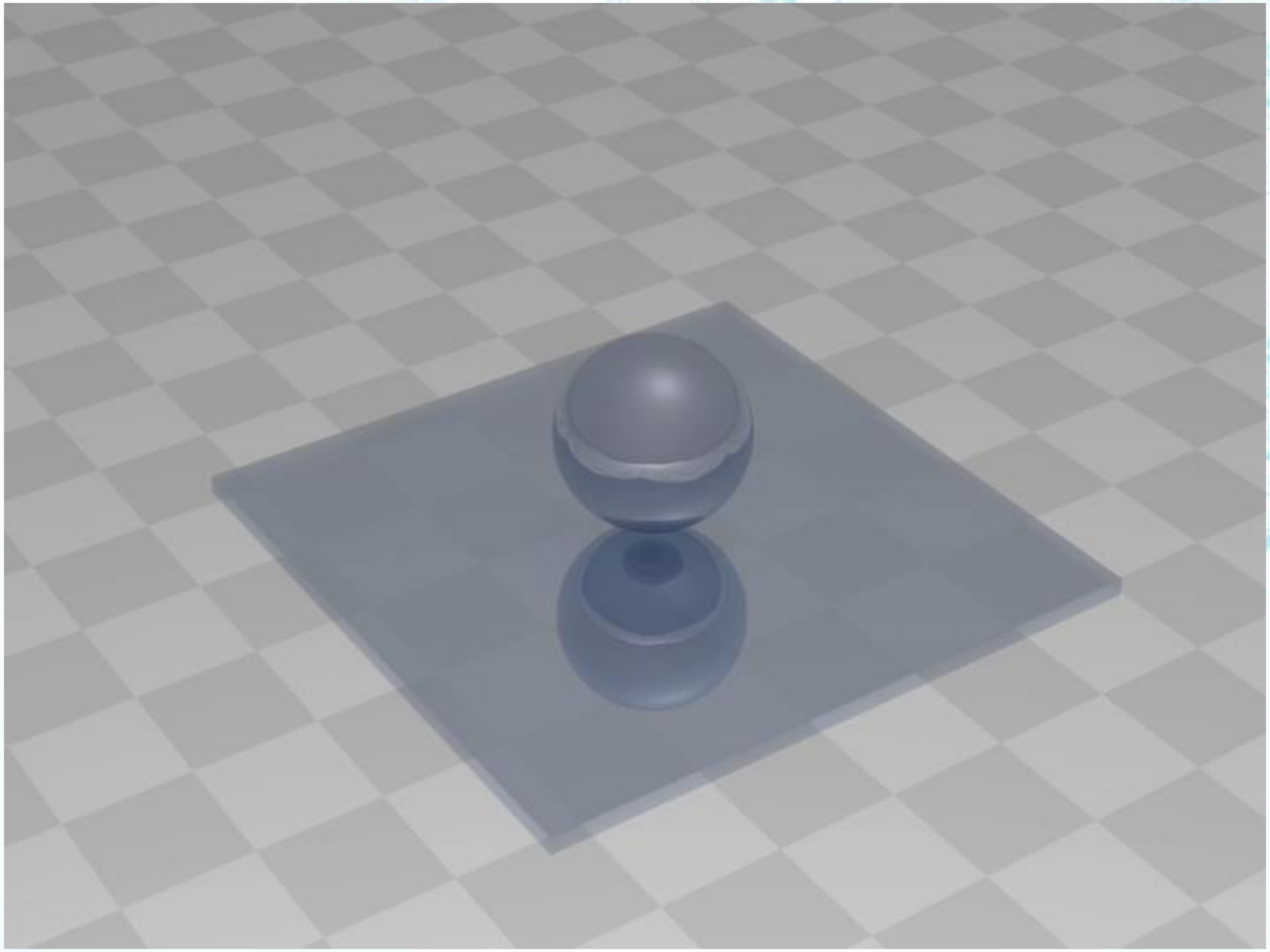
Level sets

- A numerical technique for tracking moving surfaces, curves.
- Returns a contour of the field.
- State of the art
- Define implicit surface function: $\phi(i, j, k)$
- Tri or Bilinear Interpolation can be used to estimate $\phi(\vec{x})$ between cell centers.
- Surface is taken where $\phi(\vec{x}) = 0$





Meshes

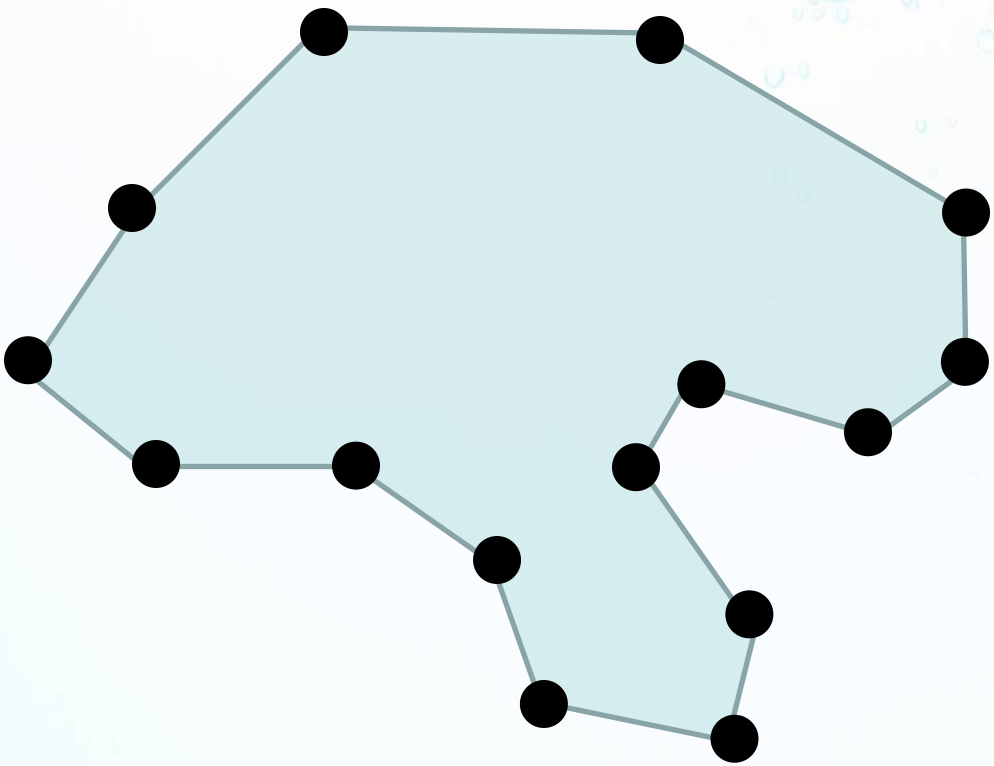


[Brochu et al 2010]

Meshes

Store a triangle mesh.

Advect its vertices, and correct for collisions.

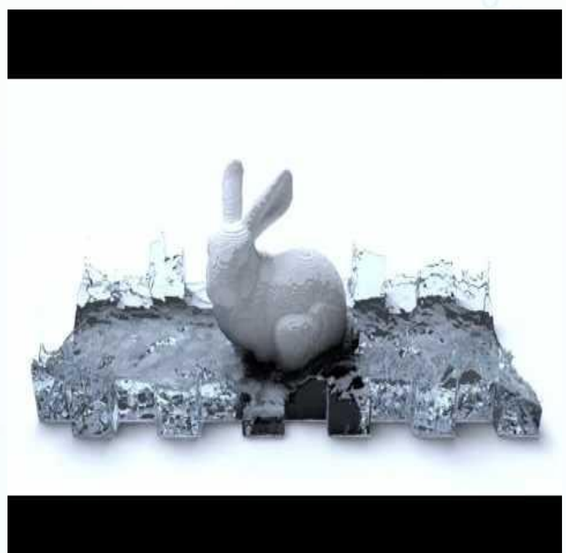


PIC and FLIP Fluids (Hybrid Methods)

- PIC stands for (Particle-In-Cell)/FLIP stands for (Fluid-Implicit-Particle).
- Is a hybrid method which uses both Lagrangian and Eulerian methods
- Mixes the perspectives of solving the system from a particle point of view and solving the system from a grid point of view (Eulerian).

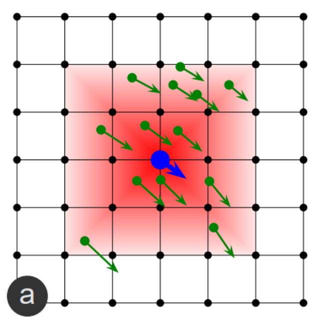
Why - Easier to solve for forces such as pressure on a uniform grid and track particle attributes such as position and velocity on the particles themselves.

Advantages - Fast simulation speed and acceptable accuracy for visual effects.



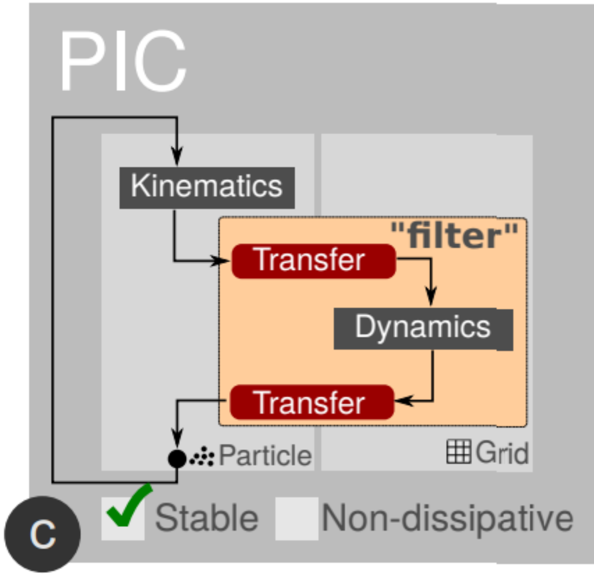
Hybrid Methods - Particle in Cell (PIC)

- Particle in Cell (PIC) transfers particle mass and velocities to grid



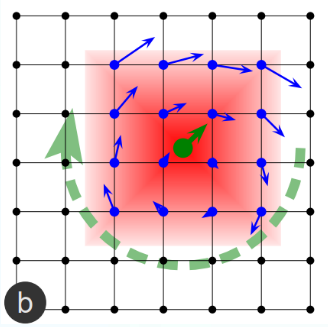
$$m_i^n = \sum_p w_{ip}^n m_p,$$

$$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p \mathbf{v}_p^n,$$



using the neighborhood (kernel) weighting function: $w_{ip}^n = N(\mathbf{x}_p^n - \mathbf{x}_i)$

- Next PIC updates grid velocities using grid-based pressure and force values
- Finally, PIC transfers the updated grid velocities back to the particles



$$\text{Grid evolution: } \mathbf{v}_i^n \to \tilde{\mathbf{v}}_i^{n+1}$$

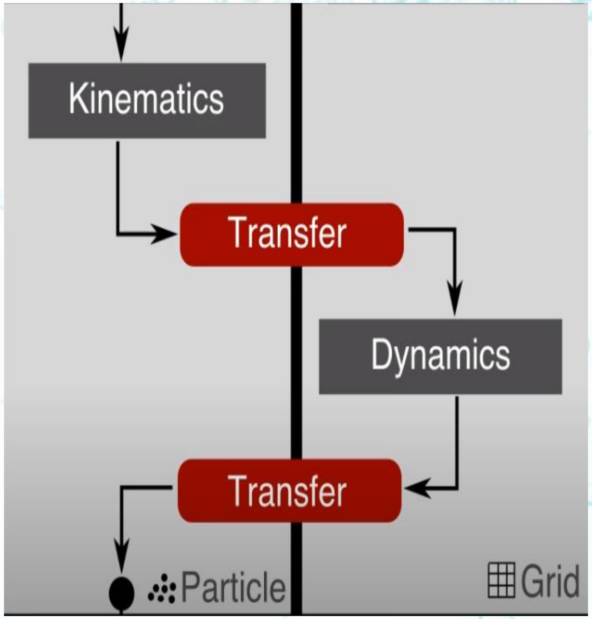
$$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1}$$

.

Hybrid Methods - PIC Summary

Advection handled with particles, but everything else computed on the grid

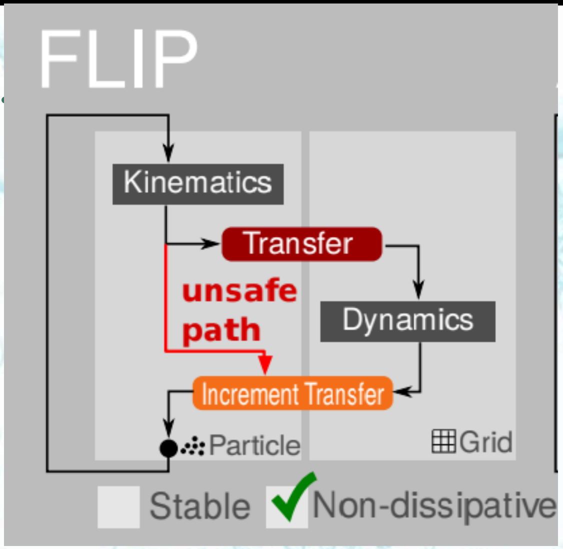
- Fluid velocity at a grid point are initialized as average of particles in the neighborhood
- Fluid velocity updated on the grid using the non-advection part of the NS equations
- New particle velocities computed by interpolating updated grid values
- This results in particles moving through space according to the grid velocity field



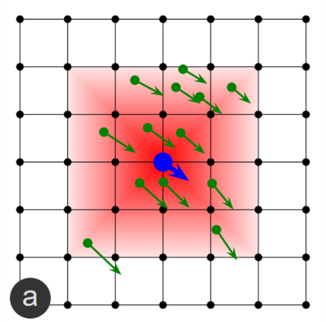
⇒ A major problem with PIC is that repeatedly averaging and interpolating the fluid variables causes numerical dissipation, which smooth out fluid details and motions.

⇒ As a result, it severely dampens rotational motion.

Hybrid Methods – Fluid Implicit Particle (FLIP)



- Fluid Implicit Particle (FLIP) transfers particle mass and change in velocities to grid

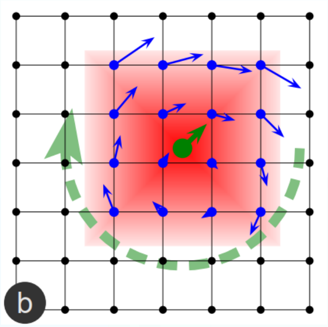


$$m_i^n = \sum_p w_{ip}^n m_p,$$

$$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p \mathbf{v}_p^n,$$

using the neighborhood (kernel) weighting function: $w_{ip}^n = N(\mathbf{x}_p^n - \mathbf{x}_i)$

- Next PIC updates grid velocities using grid-based pressure and force values
- Finally, PIC transfers the **changes in updated grid velocities** back to the particles



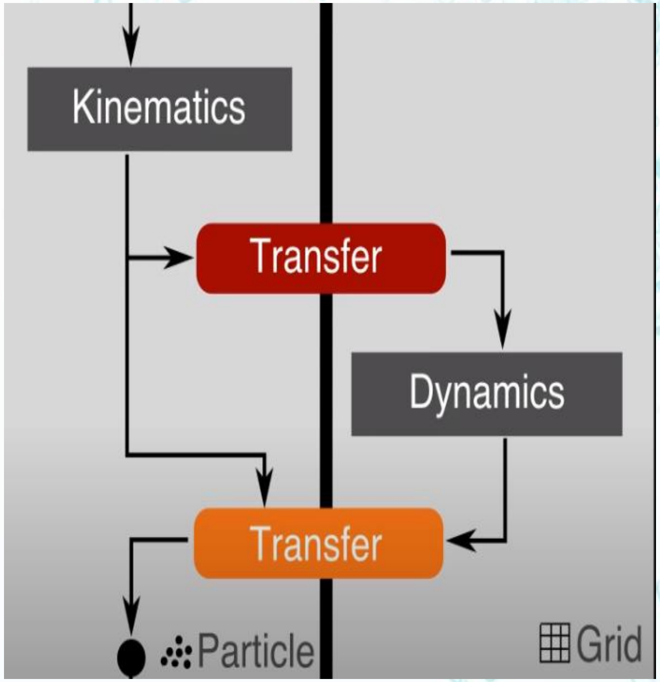
Grid evolution: $\mathbf{v}_i^n \rightarrow \tilde{\mathbf{v}}_i^{n+1}$

$$\mathbf{v}_p^{n+1} = \mathbf{v}_p^n + \sum_i w_{ip}^n (\tilde{\mathbf{v}}_i^{n+1} - \mathbf{v}_i^n)$$

.

Hybrid Methods – FLIP Summary

- Achieved almost total absence of numerical dissipation
  - Makes particles the fundamental representation of the fluid
  - Use the auxiliary grid simply to increment the particle variables according to the change computed on the grid



- Develops Noise

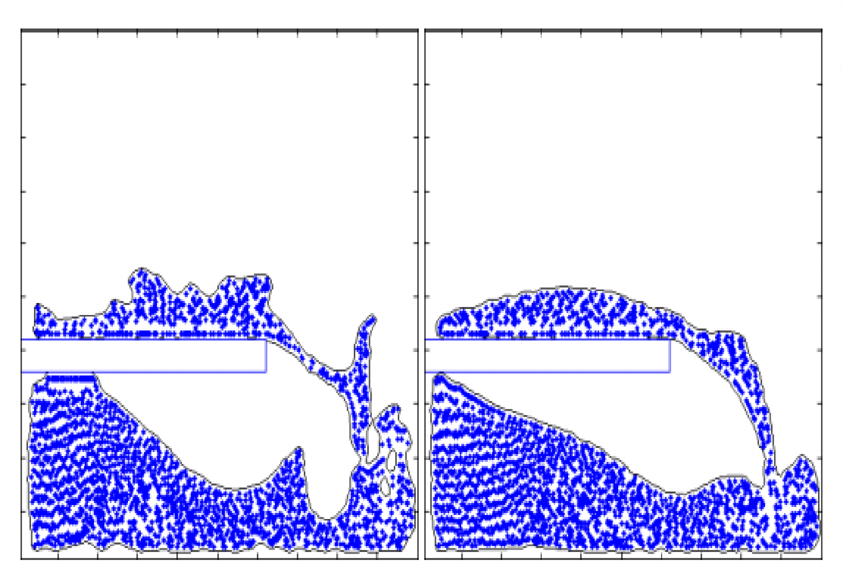
PIC and FLIP Implementation Summary

- Particle and Grid Update Equations

PIC	FLIP
$m_i^n = \sum_p w_{ip}^n m_p$	$m_i^n = \sum_p w_{ip}^n m_p$
$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p \mathbf{v}_p^n$	$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p \mathbf{v}_p^n$
Grid evolution: $\mathbf{v}_i^n \rightarrow \tilde{\mathbf{v}}_i^{n+1}$	Grid evolution: $\mathbf{v}_i^n \rightarrow \tilde{\mathbf{v}}_i^{n+1}$
$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1}$	$\mathbf{v}_p^{n+1} = \mathbf{v}_p^n + \sum_i w_{ip}^n (\tilde{\mathbf{v}}_i^{n+1} - \mathbf{v}_i^n)$
$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t \tilde{\mathbf{v}}_i^{n+1}$	$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t \tilde{\mathbf{v}}_i^{n+1}$
$\mathbf{x}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1}$	$\mathbf{x}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1}$

PIC vs FLIP

- PIC(Particle-in-Cell) viscous flows, such as sand
- FLIP(fluid Implicit-Particle) inviscid flows, such as water
- Use weighted average to tune viscosity



References

- Seminal Works
  - *Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface*, Francis Harlow, Eddie Welch (1965)
  - *Stable Fluids*, Jos Stam (1999)
  - *Practical Animation of Liquids*, Nick Foster, Ronald Fedkiw (2001)
  - *Animation and Rendering of Complex Water Surfaces,* Douglas Enright, Stephen Marschner, Ronald Fedkiw (2002)
  - "Real-Time Stable Fluid Dynamics for Games" [Stam, 2003]

- Other Sources
  - *Fluid Simulation,* Robert Bridson, Matthias Muller (SIGGRAPH 2007 Course 31)
  - *Real-Time Simulation and Rendering of 3D Fluids*， Keenan Crane， Ignacio Llamas， Sarah Tariq （2008, *GPU Gems3*, chapter30）
  - *Real-Time Water Rendering*, Claes Johanson (Master of Science thesis 2004)
  - *Real-Time Eulerian Water Simulation Using a Restricted Tall Cell Grid*, Nuttapong Chentanez, Matthias Muller (2011)

- Robert Bridson's SIGGRAPH 2007 Fluid Course notes: http://www.cs.ubc.ca/~rbridson/fluidsimulation/

Additional References

Fluid Simulation for Computer Graphics - Robert Bridson. 2008

Animating fluid sediment mixture in particle-laden flows | ACM Transactions on Graphics

Microsoft PowerPoint - GDC2008.ppt (ubm-twvideo01.s3.amazonaws.com) Fast Water Simulation using Height Fields

http://plaza.ufl.edu/ebrackear/ MAC Grid Representation

https://cg.informatik.uni-freiburg.de/intern/seminar/gridFluids_fluid-EulerParticle.pdf MAC Grid Representation

An overview of smoothed particle hydrodynamics for simulating multiphase flow - ScienceDirect SPH

https://abaqus-docs.mit.edu/2017/English/SIMACAEANLRefMap/simaanl-c-sphanalysis.htm SPH

LS_Models.pptx (drexel.edu) Level Set Surfaces

A multiple marker level-set method for simulation of deformable fluid particles - ScienceDirect Marker Particles