Simulation in Computer Graphics
Particle Fluids 1

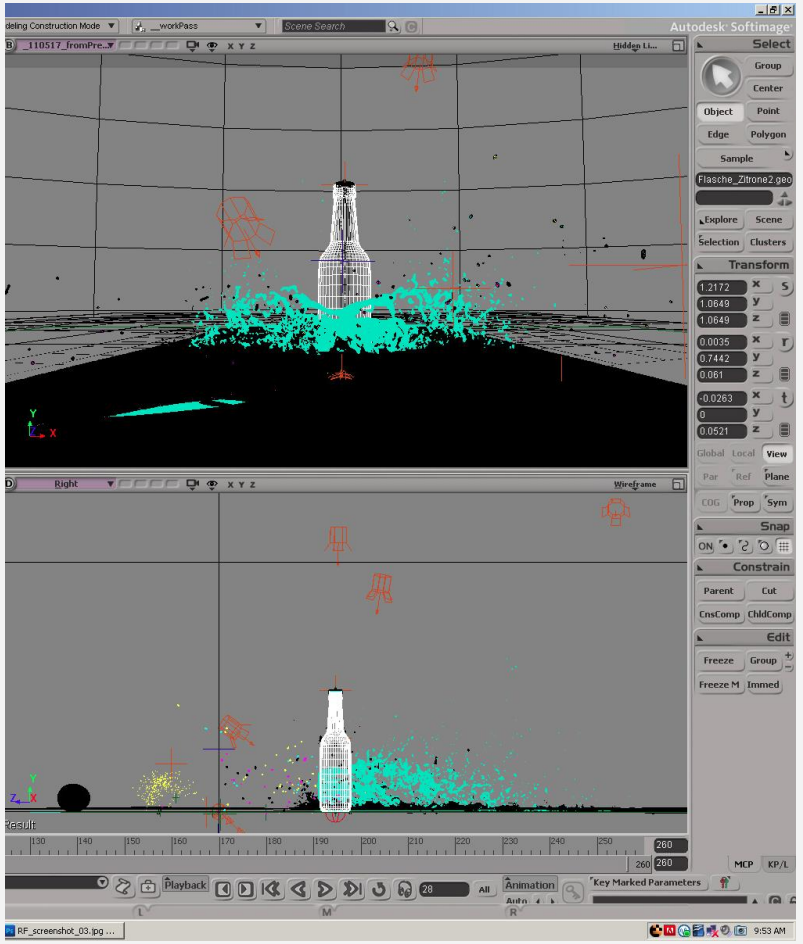
Matthias Teschner

Particle Fluids in Animation

Cooperation with
Pixar Animation Studios

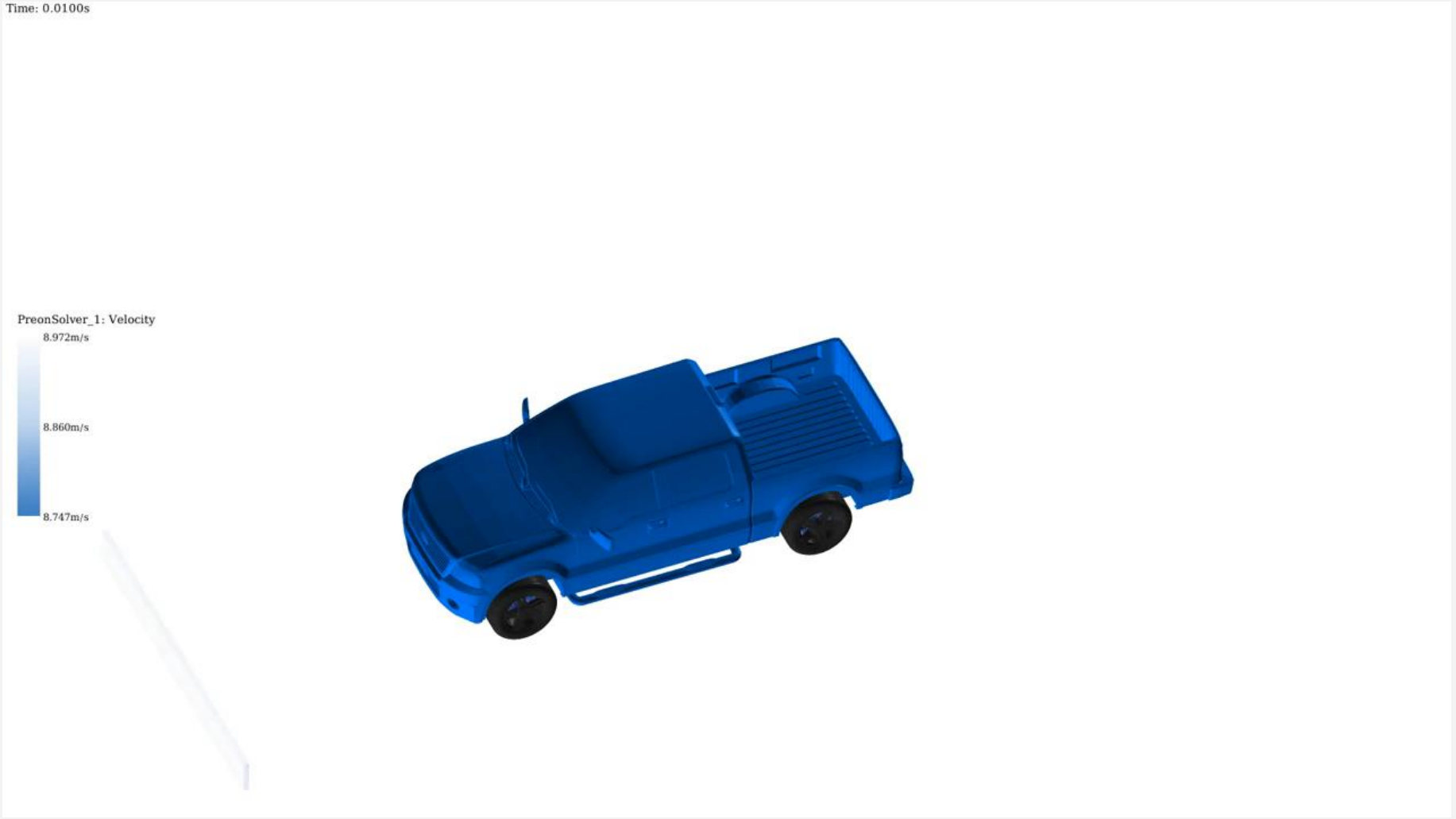
10 million fluid +
4 million rigid particles,
50 s simulated,
50 h computation time
on a 16-core PC,
www.youtube.com/cgfreiburg

Particle Fluids in Commercials



Copyright NHB Studios, Berlin, Hamburg, Dusseldorf

Particle Fluids in Engineering



PreonLab
FIFTY2 Technology

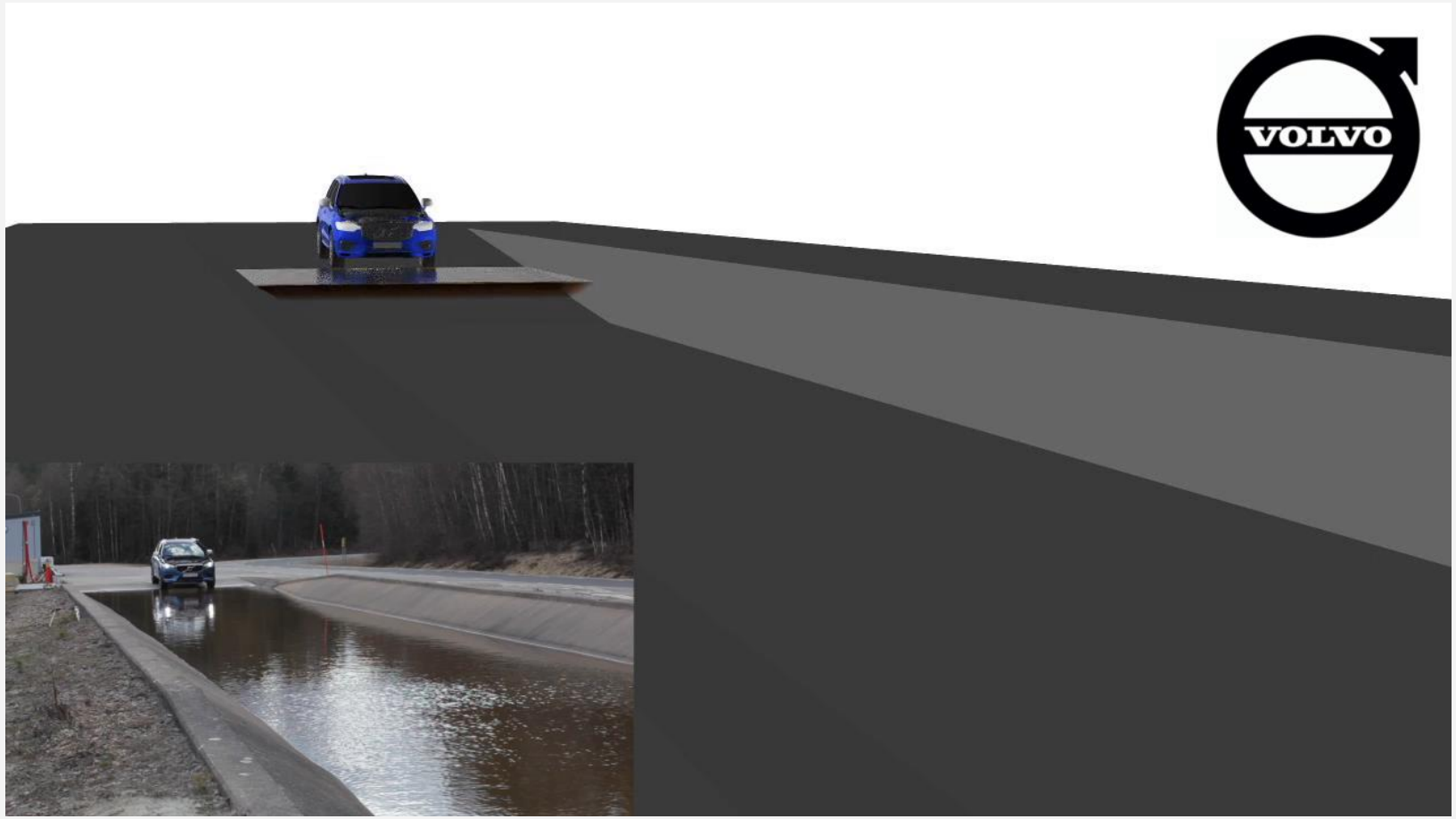
FORD F-150

Water wading

Validation of Particle Concepts

PreonLab
FIFTY2 Technology

CFD in Engineering



Johan Idoffsson
Chalmers University

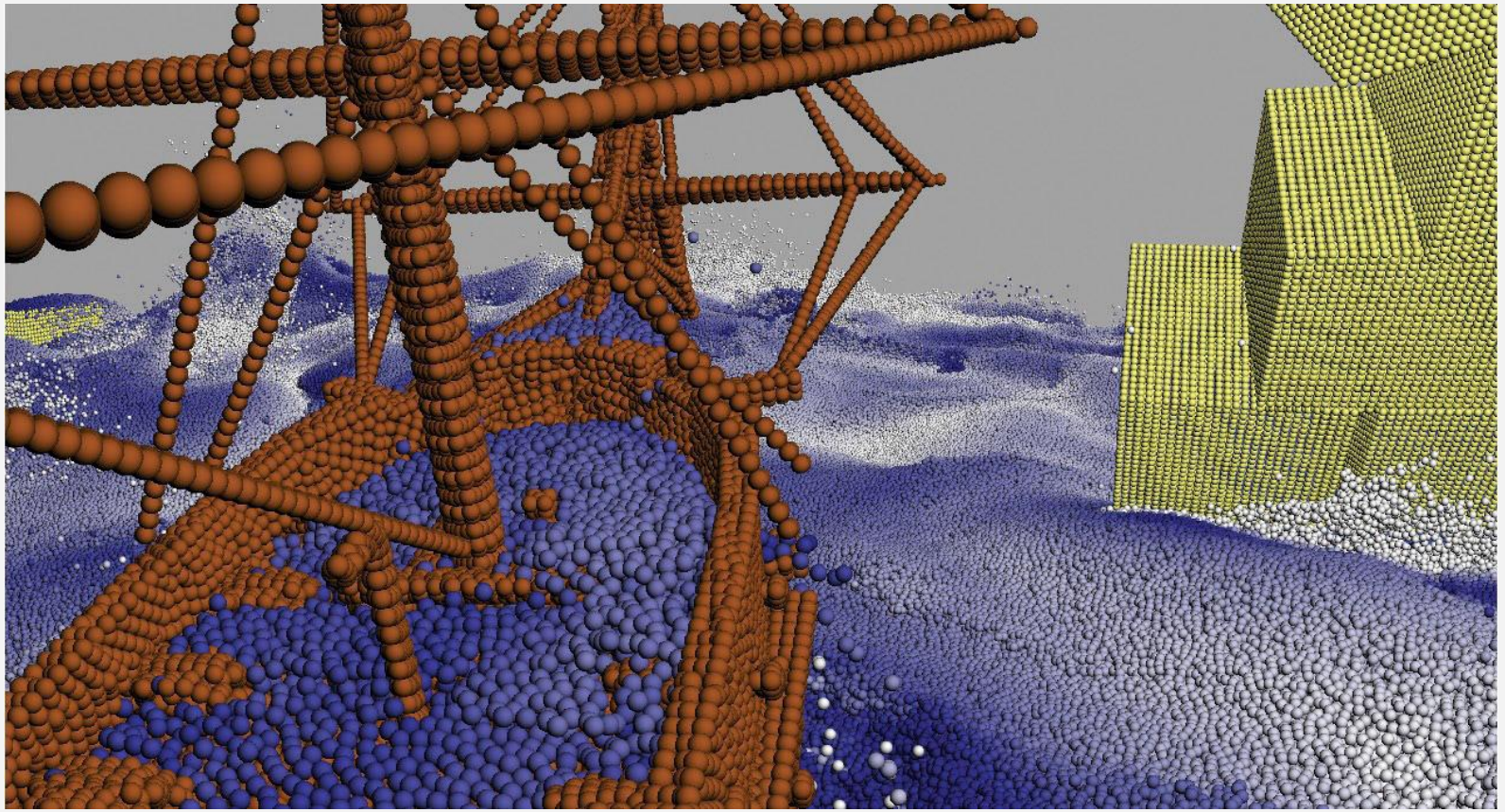
Volvo Cars

PreonLab
FIFTY2 Technology

Outline

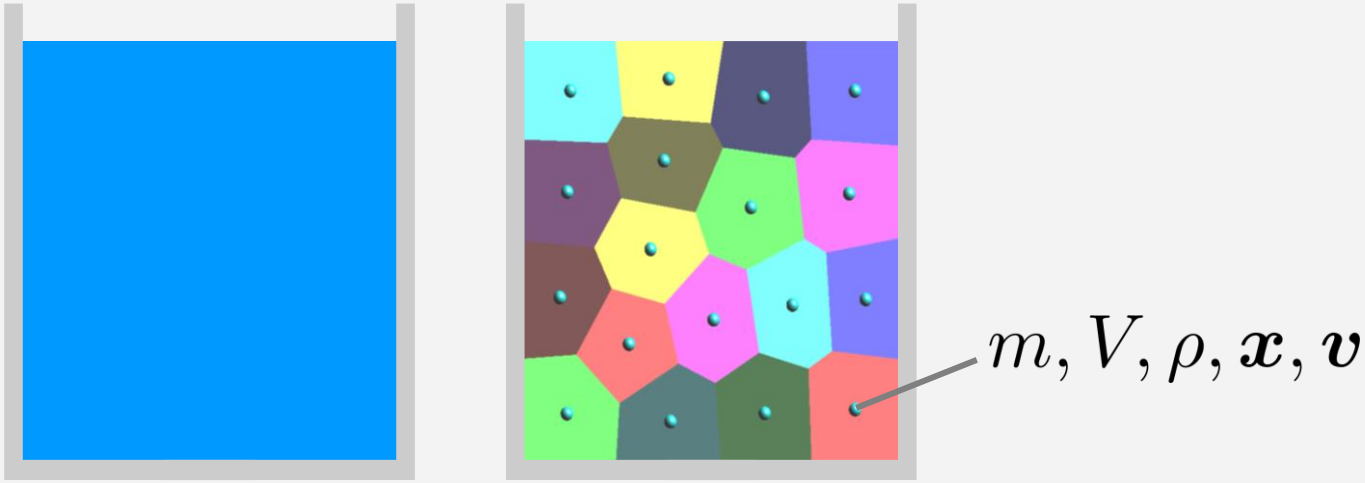
- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
- Boundary handling
- Incompressibility

Concept



Fluid Representation

– Fluid body is subdivided into small moving parcels, i.e. particles, with fluid properties

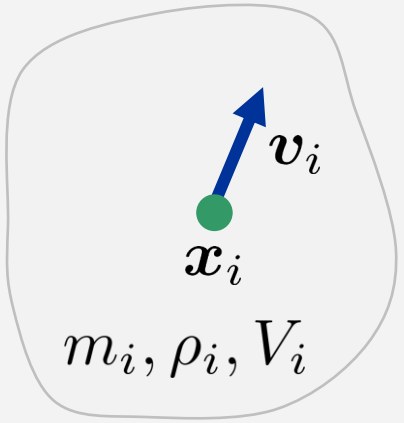


Fluid body

Set of fluid parcels

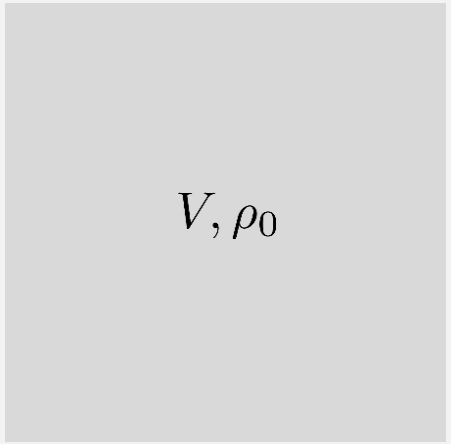
Particles / Fluid Parcels

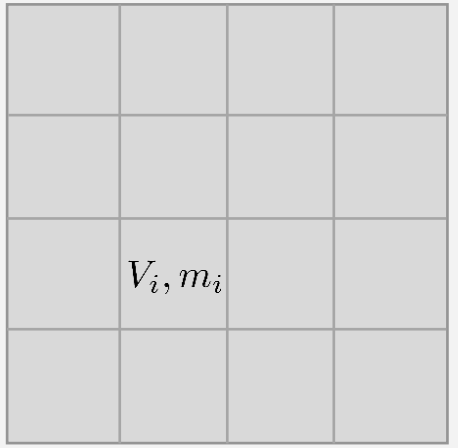
- Represent small fluid portions
- Are represented by a sample position $\boldsymbol{x}_i$
- Move with their velocity $\boldsymbol{v}_i$
- Have a fixed mass m_i
- Volume and density are related by $V_i = \frac{m_i}{\rho_i}$
  - Preservation of density / volume over time is one of the challenges of a fluid simulator
- Shape is not considered

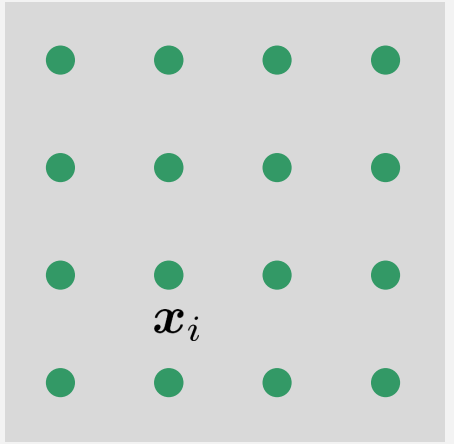


Typical Setup

- Define overall fluid volume V and fluid density ρ_0
- Define number n of particles $V_i = \frac{V}{n}$
- Compute particle mass as $m_i = \rho_0 \cdot V_i$
- Particles of uniform size
- Sample $\boldsymbol{x}_i$ represents a particle in the simulation

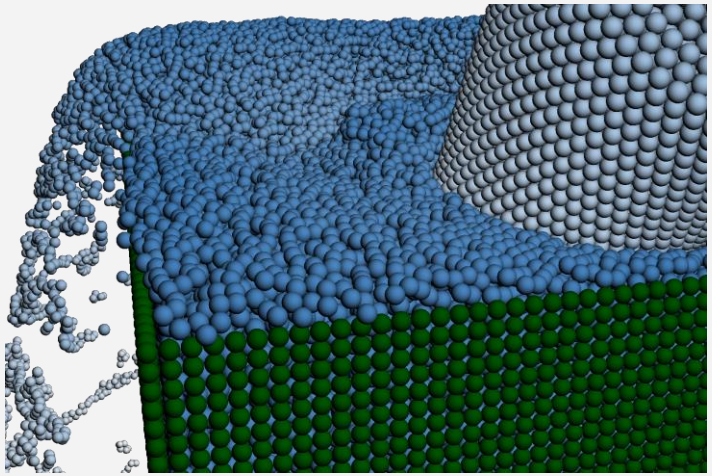




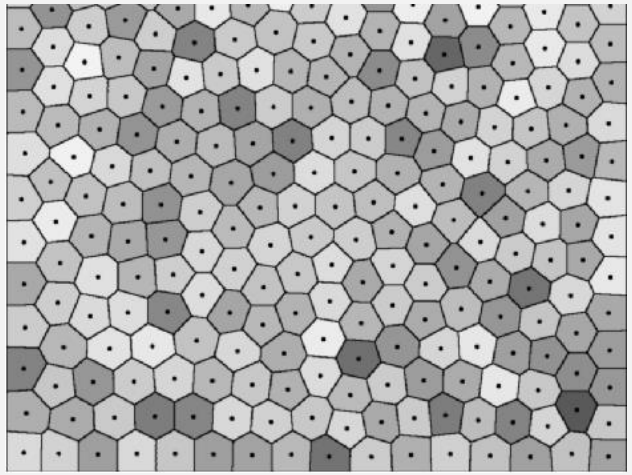


Particle Shape

- Typically initialized as a cube
- Implicitly handled as Voronoi cell by the simulation
- Typically visualized as a sphere



PreonLab, FIFTY2 Technology GmbH



Adrian Secord: Weighted Voronoi Stippling, NPAR 2002.

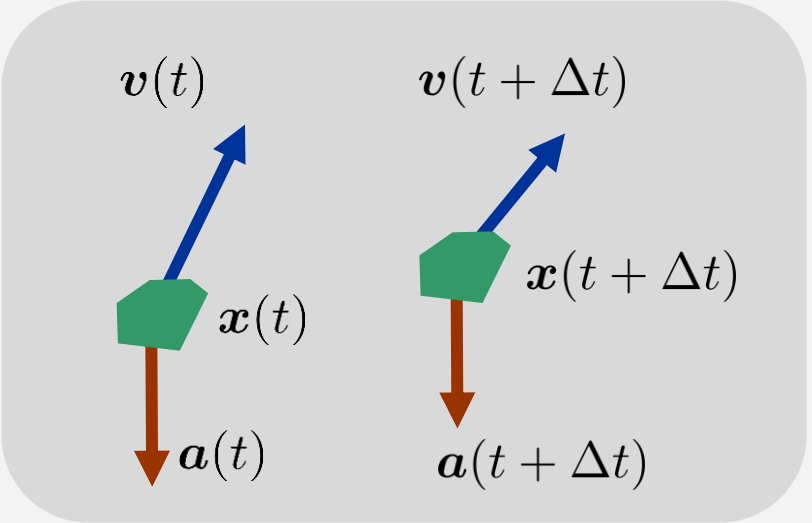
Fluid Simulation

- Computation of positions and velocities of fluid parcels over time
  - Velocity change from current time t to subsequent time $t + \Delta t$

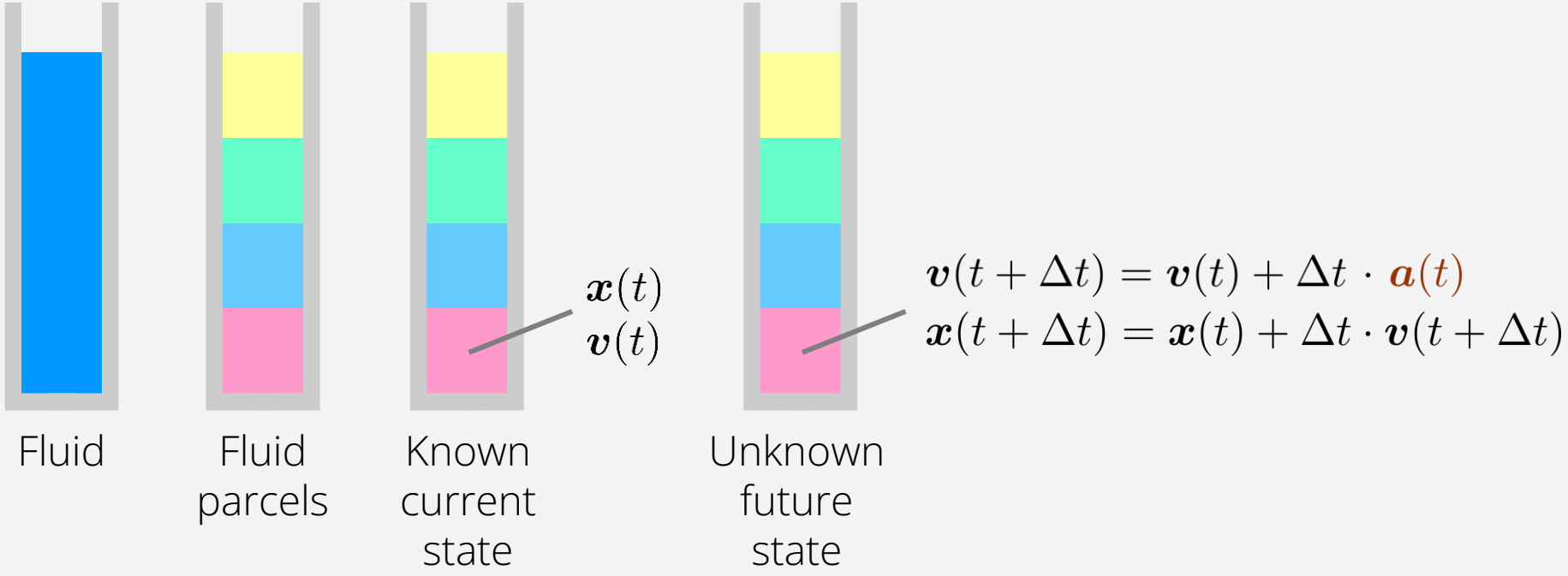
    $$\boldsymbol{v}(t + \Delta t) = \boldsymbol{v}(t) + \Delta t \cdot \boldsymbol{a}(t)$$

  - Position change

    $$\boldsymbol{x}(t + \Delta t) = \boldsymbol{x}(t) + \Delta t \cdot \boldsymbol{v}(t + \Delta t)$$



Example

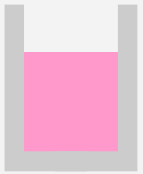

$\boldsymbol{x}(t)$
$\boldsymbol{v}(t)$
$\boldsymbol{v}(t+\Delta t) = \boldsymbol{v}(t) + \Delta t \cdot \boldsymbol{a}(t)$
$\boldsymbol{x}(t+\Delta t) = \boldsymbol{x}(t) + \Delta t \cdot \boldsymbol{v}(t+\Delta t)$
Fluid
Fluid parcels
Known current state
Unknown future state

Accelerations

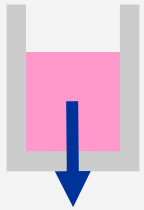
- Gravity $\boldsymbol{g}$
- Viscosity $\nu \nabla^2 \boldsymbol{v}$
  - Resistance to deformation
  - Accelerate parcel towards the average velocity of adjacent fluid parcels
- Pressure acceleration $-\frac{1}{\rho} \nabla p$
  - Prevent fluid parcels from density / volume changes

Simulation Step - Example

– Gravity and viscosity would change the parcel volume

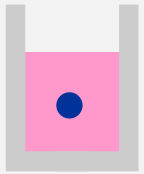


$$\boldsymbol{x}(t) = \mathbf{0}$$
$$\boldsymbol{v}(t) = \mathbf{0}$$



$\boldsymbol{g}$

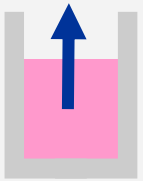
Gravity



$$\nu \nabla^2 \boldsymbol{v}(t) = \mathbf{0}$$

Viscosity

– Pressure acceleration avoids the volume/density change

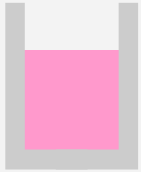


$$-\frac{1}{\rho}\nabla p = -\boldsymbol{g}$$

Pressure acceleration

Simulation Step - Example

– Current state

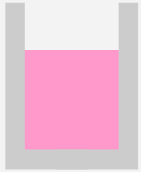


$$\boldsymbol{x}(t) = \mathbf{0}$$
$$\boldsymbol{v}(t) = \mathbf{0}$$

– Overall acceleration

$$\boldsymbol{a}(t) = \boldsymbol{g} + \nu\nabla^2\boldsymbol{v}(t) - \tfrac{1}{\rho}\nabla p$$
$$= \boldsymbol{g} + \mathbf{0} - \boldsymbol{g} = \mathbf{0}$$

– Subsequent state



$$\boldsymbol{v}(t + \Delta t) = \boldsymbol{v}(t) + \Delta t \cdot \boldsymbol{a}(t) = \mathbf{0}$$
$$\boldsymbol{x}(t + \Delta t) = \boldsymbol{x}(t) + \Delta t \cdot \boldsymbol{v}(t + \Delta t) = \mathbf{0}$$

Neighboring Parcels

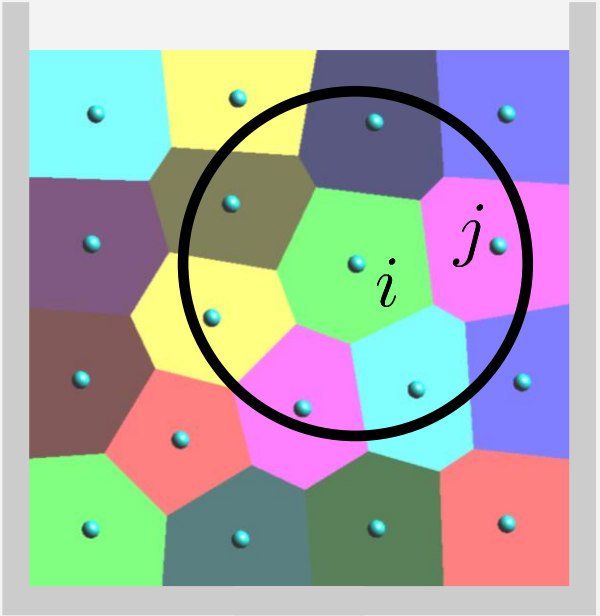
- Computations require neighboring parcels j
- Density or volume

$$\rho_i = \sum_j m_j W_{ij}$$

- Pressure acceleration

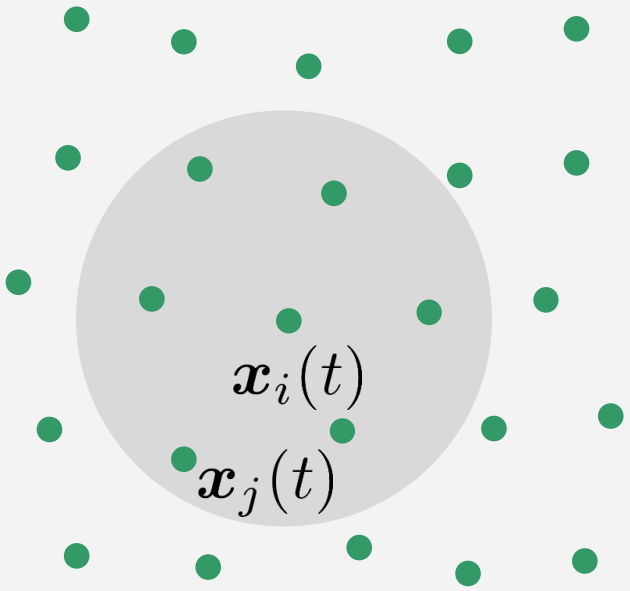
$$-\frac{1}{\rho_i}\nabla p_i = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij}$$

- Smoothed Particle Hydrodynamics SPH
  - Gingold and Monaghan, Lucy



Simulation Step - Implementation

- Determine adjacent particles / neighbors $\boldsymbol{x}_j(t)$ of particle $\boldsymbol{x}_i(t)$ ($\boldsymbol{x}_i(t)$ is neighbor of $\boldsymbol{x}_i(t)$!)
- Compute accelerations $\boldsymbol{a}_i(t) = \sum_j \ldots$ as sums of neighbors
- Advect the particles, e.g. Euler-Cromer
- Determine neighbors of particle $\boldsymbol{x}_i(t + \Delta t)$
- ...



Governing Equations

- Particles /sample positions $\boldsymbol{x}_i$ and the respective attributes are advected with the local fluid velocity $\boldsymbol{v}_i$

$$\frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} = \boldsymbol{v}_i$$

- Time rate of change of the velocity $\boldsymbol{v}_i$ of an advected sample is governed by the Lagrange form of the Navier-Stokes equation

$$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\boldsymbol{v}_i + \frac{\boldsymbol{F}_i^{\mathrm{other}}}{m_i}$$

Accelerations

- $-\frac{1}{\rho_i}\nabla p_i$: acceleration due to pressure differences
  - Preserves the fluid volume / density
  - Acts in normal direction at the surface of the fluid element
  - Small and preferably constant density deviations are important for high-quality simulation

Accelerations

- $\nu \nabla^2 \boldsymbol{v}_i$: acceleration due to friction forces between particles with different velocities
  - Friction forces act in tangential and normal direction at fluid elements
  - Kinematic viscosity $\nu \approx 10^{-6} \mathrm{m}^2 \cdot \mathrm{s}^{-1}$: larger friction is less realistic, but can improve the stability
  - Dynamic viscosity $\eta = \mu = \nu \cdot \rho_0$
- $\frac{\boldsymbol{F}_i^{\mathrm{other}}}{m_i}$: e.g., gravity

Accelerations

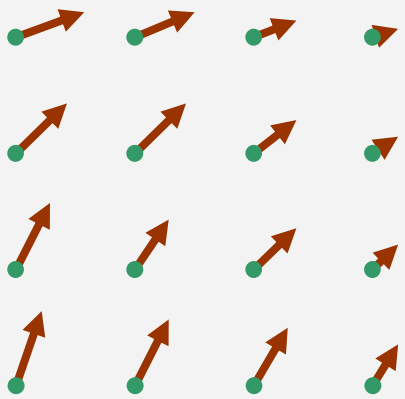
$$-\frac{1}{\rho}\nabla p = -\frac{1}{\rho}\begin{pmatrix}\frac{\partial p}{\partial x_x}\\ \frac{\partial p}{\partial x_y}\\ \frac{\partial p}{\partial x_z}\end{pmatrix} = -\frac{1}{\rho}\nabla\cdot\begin{pmatrix}p & 0 & 0\\ 0 & p & 0\\ 0 & 0 & p\end{pmatrix}$$

$$\nu\nabla^2\boldsymbol{v} = \nabla\cdot(\nu\nabla\boldsymbol{v}) = \nabla\cdot\nu\begin{pmatrix}\frac{\partial v_x}{\partial x_x} & \frac{\partial v_x}{\partial x_y} & \frac{\partial v_x}{\partial x_z}\\ \frac{\partial v_y}{\partial x_x} & \frac{\partial v_y}{\partial x_y} & \frac{\partial v_y}{\partial x_z}\\ \frac{\partial v_z}{\partial x_x} & \frac{\partial v_z}{\partial x_y} & \frac{\partial v_z}{\partial x_z}\end{pmatrix}$$

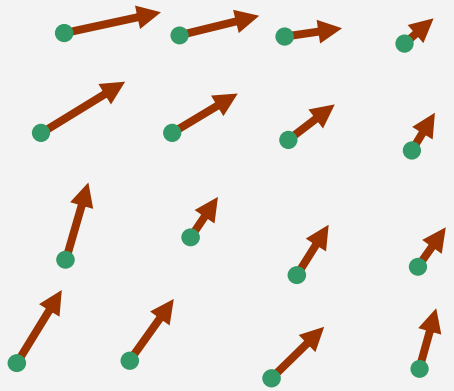
$$= \nu\begin{pmatrix}\frac{\partial^2 v_x}{\partial x_x^2} + \frac{\partial^2 v_x}{\partial x_y^2} + \frac{\partial^2 v_x}{\partial x_z^2}\\ \frac{\partial^2 v_y}{\partial x_x^2} + \frac{\partial^2 v_y}{\partial x_y^2} + \frac{\partial^2 v_y}{\partial x_z^2}\\ \frac{\partial^2 v_z}{\partial x_x^2} + \frac{\partial^2 v_z}{\partial x_y^2} + \frac{\partial^2 v_z}{\partial x_z^2}\end{pmatrix}$$

Lagrangian Fluid Simulation

- Fluid simulators compute the velocity field over time
- Lagrangian approaches compute the velocities for samples $\boldsymbol{x}_i$ that are advected with their velocity $\boldsymbol{v}_i$

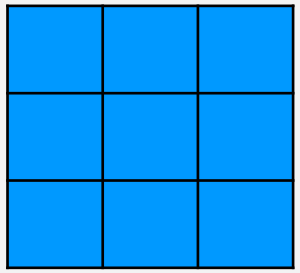


$$\boldsymbol{v}_i(x_i, y_i, z_i, t) = (u_i, v_i, w_i)$$
$$\boldsymbol{x}_i(t) = (x_i, y_i, z_i)$$



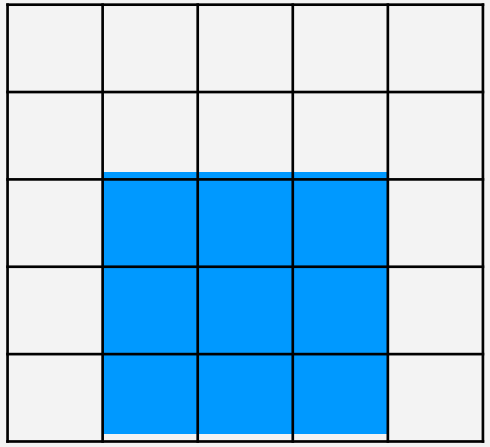
$$\boldsymbol{v}_i(x_i + \Delta t \cdot u_i, y_i + \Delta t \cdot v_i, z_i + \Delta t \cdot w_i, t + \Delta t)$$
$$\boldsymbol{x}_i(t + \Delta t) = (x_i + \Delta t \cdot u_i, y_i + \Delta t \cdot v_i, z_i + \Delta t \cdot w_i)$$

Moving Parcels vs. Static Cells



$$\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = \boldsymbol{g} + \nu\nabla^2\boldsymbol{v} - \frac{1}{\rho}\nabla p$$
$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{v}$$

Lagrangian: Acceleration of a moving parcel.



$$\frac{\partial\boldsymbol{v}}{\partial t} = \boldsymbol{g} + \nu\nabla^2\boldsymbol{v} - \frac{1}{\rho}\nabla p - (\boldsymbol{v}\cdot\nabla)\boldsymbol{v}$$

Eulerian: Acceleration at a static cell.

$$\frac{\mathrm{D}\boldsymbol{v}}{\mathrm{D}t} = \boldsymbol{g} + \nu\nabla^2\boldsymbol{v} - \frac{1}{\rho}\nabla p$$

$$\frac{\mathrm{D}\boldsymbol{v}}{\mathrm{D}t} = \frac{\partial\boldsymbol{v}}{\partial t} + (\boldsymbol{v}\cdot\nabla)\boldsymbol{v} \quad \text{or}$$
$$\frac{\mathrm{D}\boldsymbol{v}}{\mathrm{D}t} = \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} \quad \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{v}$$

Smoothed Particle Hydrodynamics

- Proposed by Gingold / Monaghan and Lucy (1977)
- SPH interpolates quantities at arbitrary positions and approximates spatial derivatives with a finite number of samples, i.e. adjacent particles

SPH for Fluids

- SPH in a Lagrangian fluid simulation
  - Fluid is represented with particles
  - Particle positions and velocities are governed by $\frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} = \boldsymbol{v}_i$ and $\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\boldsymbol{v}_i + \frac{\boldsymbol{F}_i^{\mathrm{other}}}{m_i}$
  - ρ_i, $-\frac{1}{\rho_i}\nabla p_i$, $\nu\nabla^2\boldsymbol{v}_i$ and $\frac{\boldsymbol{F}_i^{\mathrm{other}}}{m_i}$ are computed with SPH

SPH Interpolation

- Quantity A_i at an arbitrary position $\boldsymbol{x}_i$ is approximately computed with a set of known quantities A_j at sample positions $\boldsymbol{x}_j$: $A_i = \sum_j V_j A_j W_{ij} = \sum_j \frac{m_j}{\rho_j} A_j W_{ij}$
  - $\boldsymbol{x}_i$ is not necessarily a sample position
  - If $\boldsymbol{x}_i$ is a sample position, it contributes to the sum
- W_{ij} is a kernel function that weights the contributions of sample positions $\boldsymbol{x}_j$ according to their distance to $\boldsymbol{x}_i$
  - $W_{ij} = W\left(\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h}\right) = W(q)$
  - h is typically the particle size
  - $W(q) > 0$ for, e.g. $0 \leq q < 2$

Kernel Function

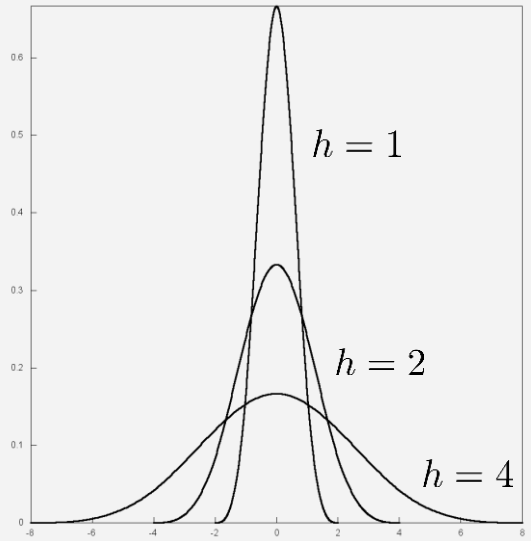
- Close to a Gaussian, but with compact support
  - Support typically between $2h$ and $5h$
- E.g. cubic spline (1D: $\alpha = \frac{1}{6h}$ 2D: $\alpha = \frac{5}{14\pi h^2}$ 3D: $\alpha = \frac{1}{4\pi h^3}$)

$$W(q) = \alpha \begin{cases} (2-q)^3 - 4(1-q)^3 & 0 \leq q < 1 \\ (2-q)^3 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases} \qquad q = \frac{\|\mathbf{x}_j - \mathbf{x}_i\|}{h}$$

- Number of considered neighbors depends on
  - Dimensionality, kernel support, particle spacing
  - E.g., 3D, cubic spline, support $2h$, particle spacing h results in 30-40 neighboring particles
  - Number of neighbors influences performance / accuracy

Kernel and Kernel Derivative in 1D

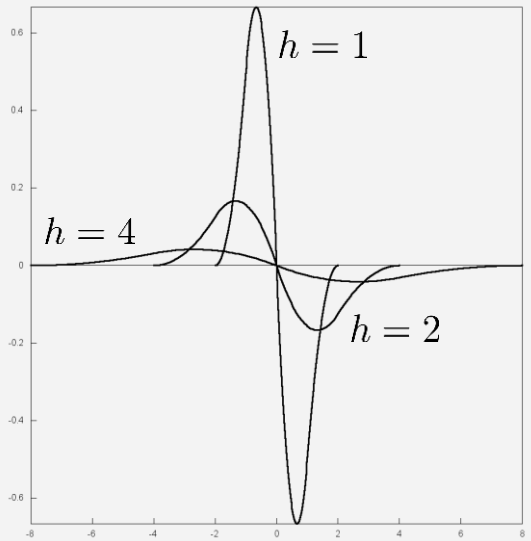
W



$x_j - x_i$

$$W(x_j - x_i) = \frac{1}{6h} \begin{cases} (2 - \frac{\|x_j - x_i\|}{h})^3 - 4(1 - \frac{\|x_j - x_i\|}{h})^3 & 0 \leq \frac{\|x_j - x_i\|}{h} < 1 \\ (2 - \frac{\|x_j - x_i\|}{h})^3 & 1 \leq \frac{\|x_j - x_i\|}{h} < 2 \\ 0 & \frac{\|x_j - x_i\|}{h} \geq 2 \end{cases}$$

∇W



$x_j - x_i$

$$\nabla W(x_j - x_i) = \frac{x_j - x_i}{6h^2 \|x_j - x_i\|} \begin{cases} -3(2 - \frac{\|x_j - x_i\|}{h})^2 + 12(1 - \frac{\|x_j - x_i\|}{h})^2 & 0 \leq \frac{\|x_j - x_i\|}{h} < 1 \\ -3(2 - \frac{\|x_j - x_i\|}{h})^2 & 1 \leq \frac{\|x_j - x_i\|}{h} < 2 \\ 0 & \frac{\|x_j - x_i\|}{h} \geq 2 \end{cases}$$

SPH for Fluids

- SPH in a Lagrangian fluid simulation
  - Fluid is represented with particles
  - Particle positions and velocities are governed by $\frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} = \boldsymbol{v}_i$ and $\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\boldsymbol{v}_i + \frac{\boldsymbol{F}_i^{\mathrm{other}}}{m_i}$
  - ρ_i, $-\frac{1}{\rho_i}\nabla p_i$, $\nu\nabla^2\boldsymbol{v}_i$ and $\frac{\boldsymbol{F}_i^{\mathrm{other}}}{m_i}$ are computed with SPH

Spatial Derivatives with SPH

- Original approximations

$$\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W_{ij}$$

$$\nabla^2 A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W_{ij}$$

- Currently preferred approximations

$$\nabla A_i = \rho_i \sum_j m_j \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2} \right) \nabla W_{ij}$$

preserves linear and angular momentum, when used for pressure forces

$$\nabla^2 A_i = 2 \sum_j \frac{m_j}{\rho_j} \frac{A_{ij} \cdot \boldsymbol{x}_{ij}}{\boldsymbol{x}_{ij} \cdot \boldsymbol{x}_{ij} + 0.01h^2} \nabla W_{ij}$$

more robust as it avoids the second derivative of W

$$\nabla \cdot \boldsymbol{A}_i = -\frac{1}{\rho_i} \sum_j m_j \boldsymbol{A}_{ij} \nabla W_{ij}$$

gives zero for constant A

$$A_{ij} = A_i - A_j \qquad \boldsymbol{A}_{ij} = \boldsymbol{A}_i - \boldsymbol{A}_j \qquad \boldsymbol{x}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$$

Density

- Explicit form
  - $\rho_i = \sum_j \frac{m_j}{\rho_j} \rho_j W_{ij} = \sum_j m_j W_{ij}$
  - Comparatively exact
  - Erroneous for incomplete neighborhood
- Differential update
  - Using the continuity equation
  - Time rate of change of the density is related to the divergence of the velocity field $\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = -\rho_i \nabla \cdot \boldsymbol{v}_i$
    $\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = \sum_j m_j \boldsymbol{v}_{ij} \nabla W_{ij}$
  - Drift

Pressure

- Quantifies fluid compression
  - E.g., state equation $p_i = \max\left(k(\frac{\rho_i}{\rho_0} - 1), 0\right)$
  - Rest density of the fluid ρ_0
  - User-defined stiffness k
- Pressure acceleration
  - $\boldsymbol{a}_i^{\mathrm{p}} = -\frac{1}{\rho_i}\nabla p_i = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij}$
  - Accelerates particles from high to low pressure, i.e. from high to low compression to minimize density deviation $\frac{\rho_i}{\rho_0} - 1$

Pressure values in SPH implementations should always be non-negative.

Simple SPH Fluid Solver

- Find neighbors of all particles
- Compute density
- Compute pressure
- Compute non-pressure accelerations, e.g. viscosity, gravity
- Compute pressure acceleration
- Update velocity and position

Contact handling, i.e. boundary handling is often realized as pressure acceleration.

Simple SPH Fluid Solver

for all *particle* i **do**
  find neighbors j

for all *particle* i **do**
  $\rho_i = \sum_j m_j W_{ij}$ Compute density
  $p_i = k(\frac{\rho_i}{\rho_0} - 1)$ Compute pressure

for all *particle* i **do**
  $\boldsymbol{a}_i^{\text{nonp}} = \nu \nabla^2 \boldsymbol{v}_i + \boldsymbol{g}$ Compute non-pressure accelerations
  $\boldsymbol{a}_i^{\text{p}} = -\frac{1}{\rho_i} \nabla p_i$ Compute pressure acceleration
  $\boldsymbol{a}_i(t) = \boldsymbol{a}_i^{\text{nonp}} + \boldsymbol{a}_i^{\text{p}}$

for all *particle* i **do**
  $\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i(t) + \Delta t \boldsymbol{a}_i(t)$
  $\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i(t + \Delta t)$

SPH Discretizations

- Density computation $\rho_i = \sum_j m_j W_{ij}$
- Pressure acceleration $-\frac{1}{\rho_i}\nabla p_i = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}$
- Viscosity acceleration $\nu\nabla^2 \boldsymbol{v}_i = 2\nu \sum_j \frac{m_j}{\rho_j} \frac{\boldsymbol{v}_{ij} \cdot \boldsymbol{x}_{ij}}{\boldsymbol{x}_{ij} \cdot \boldsymbol{x}_{ij} + 0.01h^2} \nabla W_{ij}$

Boundary Handling

- Boundaries can be represented with static fluid samples
- Computations incorporate boundary samples, e.g.

$$\rho_i = \sum_f m_f W_{if} + \sum_b m_b W_{ib}$$

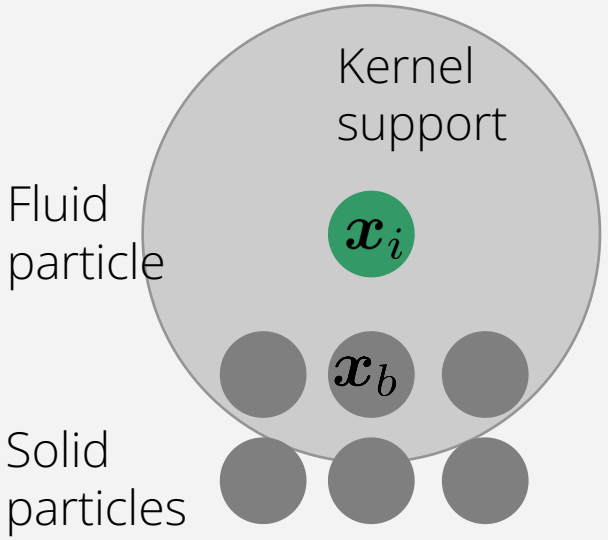
Fluid samples Boundary samples

$$-\frac{1}{\rho_i}\nabla p_i = -\sum_f m_f \left(\frac{p_i}{\rho_i^2} + \frac{p_f}{\rho_f^2}\right)\nabla W_{if} - \sum_b m_b \left(\frac{p_i}{\rho_i^2} + \frac{p_b}{\rho_b^2}\right)\nabla W_{ib}$$

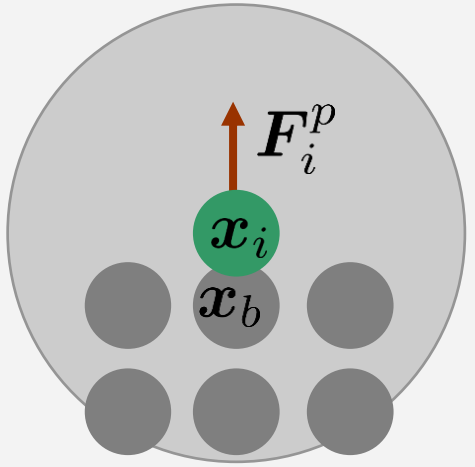
Fluid samples Boundary samples

- Fluid sample at boundary
  - Density and pressure increases
  - Pressure acceleration resolves contact

Boundary Handling



$$\rho_i < \rho_0$$
$$p_i = 0$$
$$p_b = 0$$
$$\boldsymbol{F}_i^{\mathrm{p}} = \mathbf{0}$$



$$\rho_i > \rho_0$$
$$p_i > 0$$
$$p_b > 0$$
$$\boldsymbol{F}_i^{\mathrm{p}} \neq \mathbf{0}$$

– Boundary handling: How to compute $\rho_i, p_i, p_b, \boldsymbol{F}_i^{\mathrm{p}}$?

Setting

- Kernel has to be defined, e.g. cubic with support of $2h$
- Particle mass m_i has to be specified
  - E.g., $m_i = h^3 \rho_0$ for a particle spacing of h
  - Spacing governs particle mass
  - Ratio of support vs. spacing governs the number of neighbors
- Numerical integration scheme
  - Semi-implicit Euler (symplectic Euler or Euler-Cromer) is commonly used

Setting

- Time step
  - Size is governed by the Courant-Friedrich-Levy (CFL) condition
  - E.g., $\Delta t \leq \lambda \frac{h}{\|\mathbf{v}^{\max}\|}$ with $\lambda = 0.1$ and particle spacing h
  - Motivation: For $\lambda \leq 1$, a particle moves less than its size / diameter per time step

Outline

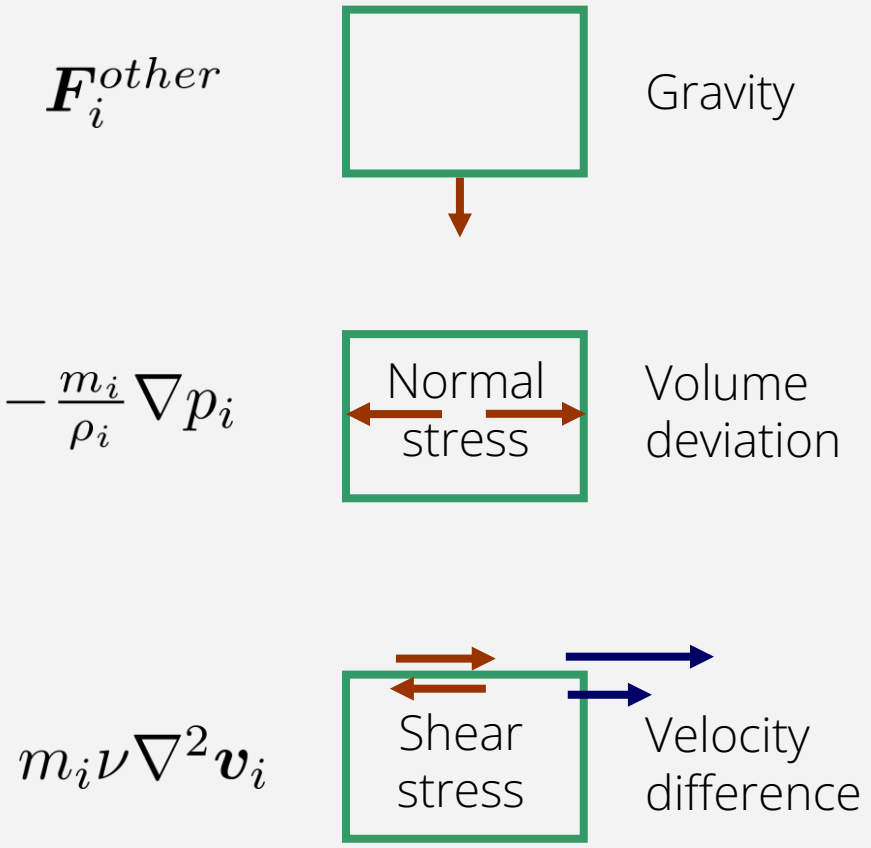
- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
- Boundary handling
- Incompressibility

Force Types

- Momentum equation

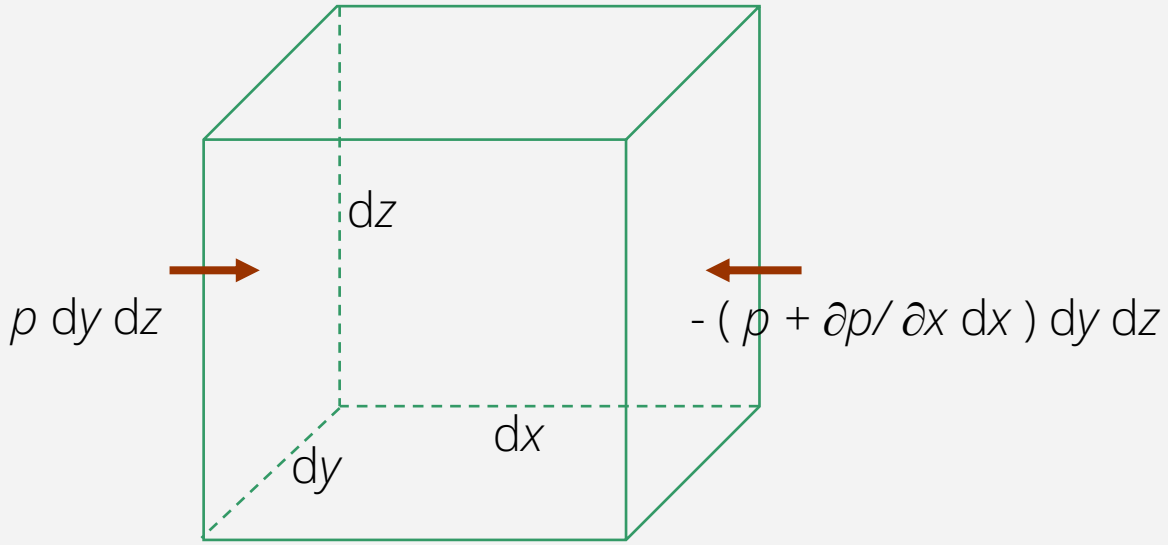
$$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\boldsymbol{v}_i + \frac{\boldsymbol{F}_i^{other}}{m_i}$$

- Body forces
- Surface forces
  - Normal stress related to volume deviation
  - Normal and shear stress related to friction due to velocity differences



Pressure Force in x-direction

- Pressure force acts orthogonal to the surface of the fluid element
- Resulting pressure force



$$\left(p - \left(p + \frac{\partial p}{\partial x}\,\mathrm{d}x\right)\right)\,\mathrm{d}y\,\mathrm{d}z = -\frac{\partial p}{\partial x}\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z = -\frac{\partial p}{\partial x}\,V$$

Overall Pressure Force

– Pressure force at particle i

$$\boldsymbol{F}_i^{\mathrm{p}} = -\begin{pmatrix} \frac{\partial p_i}{\partial x_{i,x}} \\ \frac{\partial p_i}{\partial x_{i,y}} \\ \frac{\partial p_i}{\partial x_{i,z}} \end{pmatrix} V_i = -\nabla p_i \; V_i = -\frac{m_i}{\rho_i} \nabla p_i$$

– Pressure acceleration

$$\boldsymbol{a}_i^{\mathrm{p}} = \frac{\boldsymbol{F}_i^{\mathrm{p}}}{m_i} = -\frac{1}{\rho_i} \nabla p_i$$

Cauchy Momentum Equation

- Lagrange form $\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = \frac{1}{\rho}\nabla \cdot \boldsymbol{\sigma} + \frac{\boldsymbol{F}^{other}}{m}$
- $\boldsymbol{\sigma}$ is the stress tensor (a 3x3 matrix in 3D) describing the pressure distribution at the surface of a fluid element $\boldsymbol{\sigma} = -p\boldsymbol{I_3} + \boldsymbol{\tau}$
- $\nabla \cdot \boldsymbol{\sigma}$ is the resulting force per volume
- $\boldsymbol{\tau}$ is the viscous stress tensor
- $\nabla \cdot \boldsymbol{\tau} = \nu\nabla^2\boldsymbol{v}$ is the resulting viscosity force per volume
- $\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\boldsymbol{v}_i + \frac{\boldsymbol{F}_i^{other}}{m_i}$

$$\boldsymbol{\tau} = \nu \begin{pmatrix} \frac{\partial u}{\partial x} + \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} & \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} + \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} \\ \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} & \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} + \frac{\partial w}{\partial z} \end{pmatrix}$$

Simulation in Computer Graphics Particle Fluids 2

Matthias Teschner

UNI FREIBURG

Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
- Boundary handling
- Incompressibility

Simple SPH Fluid Solver

for all *particle* i **do**
  find neighbors j

for all *particle* i **do**
  $\rho_i = \sum_j m_j W_{ij}$ Compute density
  $p_i = k(\frac{\rho_i}{\rho_0} - 1)$ Compute pressure

for all *particle* i **do**
  $\boldsymbol{a}_i^{\text{nonp}} = \nu \nabla^2 \boldsymbol{v}_i + \boldsymbol{g}$ Compute non-pressure accelerations
  $\boldsymbol{a}_i^{\text{p}} = -\frac{1}{\rho_i} \nabla p_i$ Compute pressure acceleration
  $\boldsymbol{a}_i(t) = \boldsymbol{a}_i^{\text{nonp}} + \boldsymbol{a}_i^{\text{p}}$

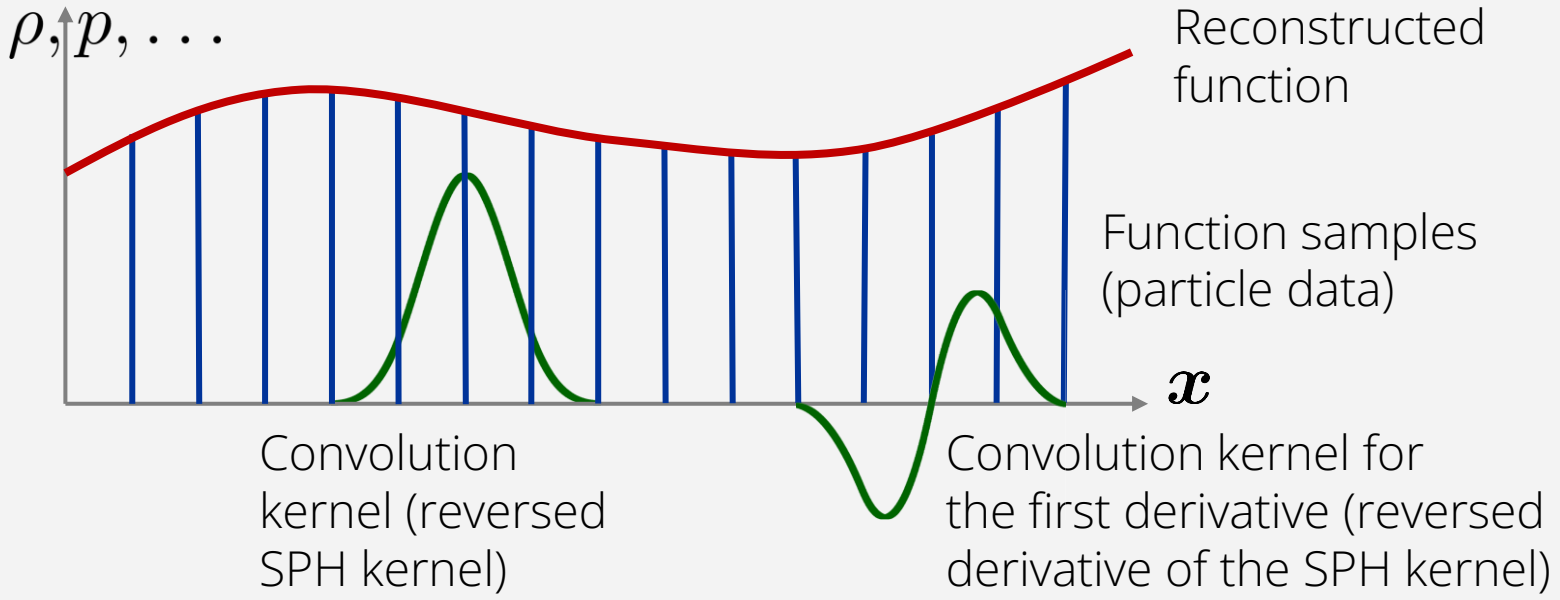
for all *particle* i **do**
  $\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i(t) + \Delta t \boldsymbol{a}_i(t)$
  $\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i(t + \Delta t)$

SPH Discretizations

- Density computation $\rho_i = \sum_j m_j W_{ij}$
- Pressure acceleration $-\frac{1}{\rho_i}\nabla p_i = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}$
- Viscosity acceleration $\nu\nabla^2 \boldsymbol{v}_i = 2\nu \sum_j \frac{m_j}{\rho_j} \frac{\boldsymbol{v}_{ij} \cdot \boldsymbol{x}_{ij}}{\boldsymbol{x}_{ij} \cdot \boldsymbol{x}_{ij} + 0.01h^2} \nabla W_{ij}$

SPH Concept

– Reconstruction of a function and its derivatives, e.g. $\rho, \nabla p, \nabla^2 \boldsymbol{v}$, from discrete samples
– Convolution of discrete samples with a filter / kernel



SPH Concept

- Quantity A at position $\mathbf{x}$ can be written as

  $A(\boldsymbol{x}) = (A * \delta)(\boldsymbol{x}) = \int_\Omega A(\boldsymbol{x}')\delta(\boldsymbol{x} - \boldsymbol{x}')\mathrm{d}\boldsymbol{x}'$
- Dirac delta $\delta(\boldsymbol{x}) = \delta(x)\delta(y)\delta(z)$ and $\delta(x) = \begin{cases} \infty & x = 0 \\ 0 & x \neq 0 \end{cases}$
- $\int_{-\infty}^{+\infty} \delta(x)\mathrm{d}x = 1$
- Dirac delta is approximated with a kernel function with limited local support, e.g. $2h$

  Particle size *h*

  $A(\boldsymbol{x}) = (A * W)(\boldsymbol{x}) = \int_\Omega A(\boldsymbol{x}')W(\boldsymbol{x} - \boldsymbol{x}', 2h)\mathrm{d}\boldsymbol{x}'$

  Reversed kernel
- Convolution: $(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)\mathrm{d}\tau$

First Derivative

- $\nabla A(\boldsymbol{x}) = \nabla(A * W)(\boldsymbol{x}) = (\nabla A * W)(\boldsymbol{x}) = (A * \nabla W)(\boldsymbol{x})$
- Follows from
  - $\mathcal{F}(\nabla(A * W)(\boldsymbol{x})) = i2\pi s(a(s) \cdot w(s))$
  - $\mathcal{F}((\nabla A * W)(\boldsymbol{x})) = i2\pi s \cdot a(s) \cdot w(s)$
  - $\mathcal{F}((A * \nabla W)(\boldsymbol{x})) = a(s) \cdot i2\pi s \cdot w(s)$

Fourier transform. See, e.g. Bracewell 1965.

- $\nabla A(\boldsymbol{x}) = \nabla(A * W)(\boldsymbol{x}) = (A * \nabla W)(\boldsymbol{x})$

$$= \int_\Omega A(\boldsymbol{x}')\nabla W(\boldsymbol{x} - \boldsymbol{x}', 2h)\mathrm{d}\boldsymbol{x}'$$

Reversed kernel derivative

Second Derivative

- $\nabla^2 A(\boldsymbol{x}) = \nabla^2 (A * W)(\boldsymbol{x}) = (\nabla^2 A * W)(\boldsymbol{x}) = (\nabla A * \nabla W)(\boldsymbol{x}) = (A * \nabla^2 W)(\boldsymbol{x})$
- Follows from
  - $\mathcal{F}(\nabla^2 (A * W)(\boldsymbol{x})) = (i2\pi s)^2 (a(s) \cdot w(s))$
  - $\mathcal{F}((\nabla^2 A * W)(\boldsymbol{x})) = (i2\pi s)^2 \cdot a(s) \cdot w(s)$
  - $\mathcal{F}((A * \nabla^2 W)(\boldsymbol{x})) = a(s) \cdot (i2\pi s)^2 \cdot w(s)$
- $\nabla^2 A(\boldsymbol{x}) = \nabla^2 (A * W)(\boldsymbol{x}) = (A * \nabla^2 W)(\boldsymbol{x})$

$$= \int_\Omega A(\boldsymbol{x}') \nabla^2 W(\boldsymbol{x} - \boldsymbol{x}', 2h) \mathrm{d}\boldsymbol{x}'$$

Reversed second kernel derivative

Kernel Function - Properties

- Integral should be normalized (unity condition) $\int_\Omega W(\boldsymbol{x}' - \boldsymbol{x}, 2h)\mathrm{d}\boldsymbol{x}' = 1$
- Support should be compact $W(\boldsymbol{x}' - \boldsymbol{x}, 2h) = 0 \text{ for } \|\boldsymbol{x}' - \boldsymbol{x}\| > 2h$
- Should be symmetric $W(\boldsymbol{x}' - \boldsymbol{x}, 2h) = W(\boldsymbol{x} - \boldsymbol{x}', 2h)$
- Should be non-negative $W(\boldsymbol{x}' - \boldsymbol{x}, 2h) \geq 0$
- Should converge to the Dirac delta for $h \to 0$
- Should be differentiable: $\nabla W, \nabla^2 W$ should exist

This is the actual parameterization of W.
x′ and x are reversed in SPH convolutions.

Particle Approximation

- $A(\boldsymbol{x}) = \int_\Omega A(\boldsymbol{x}')W(\boldsymbol{x}-\boldsymbol{x}', 2h)\mathrm{d}\boldsymbol{x}' = \int_\Omega \frac{A(\boldsymbol{x}')}{\rho(\boldsymbol{x}')}W(\boldsymbol{x}-\boldsymbol{x}', 2h)\rho(\boldsymbol{x}')\mathrm{d}\boldsymbol{x}'$
- Consider a limited number of samples / particles $\boldsymbol{x}_j$ representing a mass $m(\boldsymbol{x}_j) = \rho(\boldsymbol{x}_j)V(\boldsymbol{x}_j)$

  $A(\boldsymbol{x}_i) = \sum_j A(\boldsymbol{x}_j)W(\boldsymbol{x}_i-\boldsymbol{x}_j, 2h)V(\boldsymbol{x}_j)$

  $A(\boldsymbol{x}_i) = \sum_j \frac{m(\boldsymbol{x}_j)}{\rho(\boldsymbol{x}_j)}A(\boldsymbol{x}_j)W(\boldsymbol{x}_i-\boldsymbol{x}_j, 2h)$
- Typical notation

  $A_i = \sum_j \frac{m_j}{\rho_j}A_j W_{ij}$

Particle Approximation of Derivatives

- First derivative

$$\nabla A(\boldsymbol{x}_i) = \sum_j \frac{m(\boldsymbol{x}_j)}{\rho(\boldsymbol{x}_j)} A(\boldsymbol{x}_j) \nabla W(\boldsymbol{x}_i - \boldsymbol{x}_j, 2h)$$

$$\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W_{ij}$$

- Second derivative

$$\nabla^2 A(\boldsymbol{x}_i) = \sum_j \frac{m(\boldsymbol{x}_j)}{\rho(\boldsymbol{x}_j)} A(\boldsymbol{x}_j) \nabla^2 W(\boldsymbol{x}_i - \boldsymbol{x}_j, 2h)$$

$$\nabla^2 A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W_{ij}$$

Kernel Function - Example

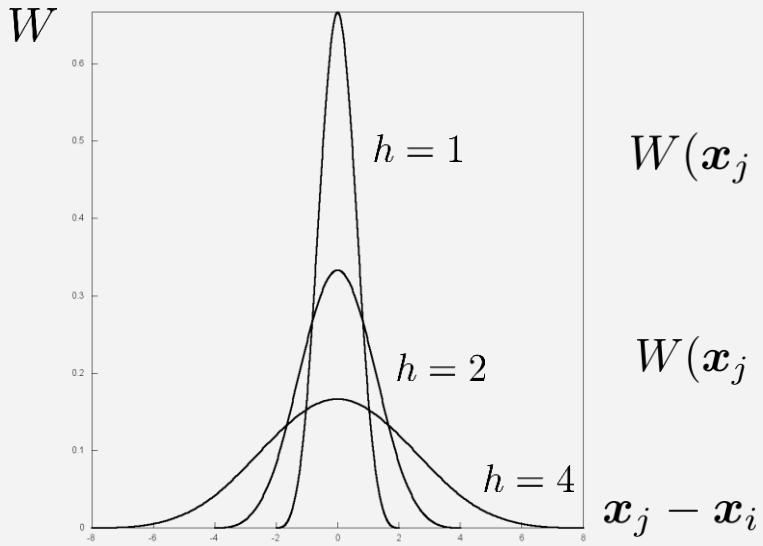
- Cubic spline

$$W(q) = \alpha \begin{cases} (2-q)^3 - 4(1-q)^3 & 0 \leq q < 1 \\ (2-q)^3 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases} \qquad q = \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h}$$

with $\alpha = \frac{1}{6h}$ (1D), $\alpha = \frac{5}{14\pi h^2}$ (2D), $\alpha = \frac{1}{4\pi h^3}$ (3D)

- Close to a Gaussian
  - Compact support between $2h$ and $5h$
- Number of considered samples depends on
  - Dimensionality, kernel support, particle spacing
  - Number of neighbors should not be too small

Kernel Function - Illustration



$$W(\boldsymbol{x}_j - \boldsymbol{x}_i) = \alpha \begin{cases} (2 - \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h})^3 - 4(1 - \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h})^3 & 0 \leq \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h} < 1 \\ (2 - \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h})^3 & 1 \leq \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h} < 2 \\ 0 & \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h} \geq 2 \end{cases}$$

$W(\boldsymbol{x}_j - \boldsymbol{x}_i) = W(\boldsymbol{x}_i - \boldsymbol{x}_j)$ depends on the distance between samples

Kernel Function - Implementation

– Reversed kernel function as used in SPH sums for the convolution

$$W(\boldsymbol{x}_i - \boldsymbol{x}_j) = \alpha \begin{cases} (2 - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h})^3 - 4(1 - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h})^3 & 0 \leq \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h} < 1 \\ (2 - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h})^3 & 1 \leq \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h} < 2 \\ 0 & \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h} \geq 2 \end{cases}$$

– Implementation

```
d  := distance(xi,xj)/h;
t1 := max(1-d,0);
t2 := max(2-d,0);
w  := alpha *(t2*t2*t2 - 4*t1*t1*t1);
```

First Kernel Derivative

- $\nabla W(\boldsymbol{x}_j - \boldsymbol{x}_i) = \left(\frac{\partial W}{\partial x_{j,x}}, \frac{\partial W}{\partial x_{j,y}}, \frac{\partial W}{\partial x_{j,z}} \right)^\mathsf{T} = \frac{\partial W(q)}{\partial q} \nabla q$
- Cubic spline

$$q = \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h} \qquad \nabla q = \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\| h}$$

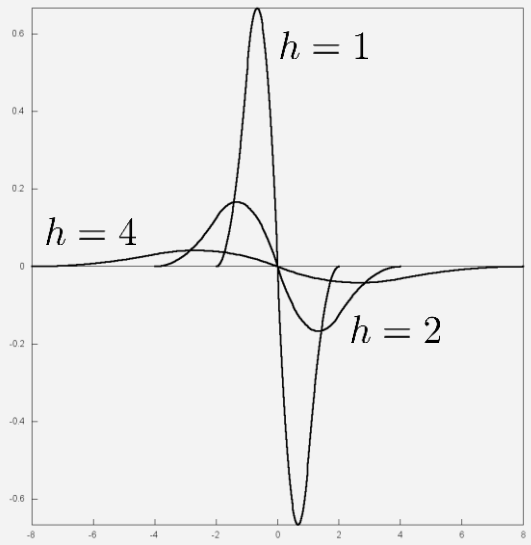
Derivative of q with respect to $\mathbf{x}_j$

$$\frac{\partial W(q)}{\partial q} = \alpha \begin{cases} -3(2-q)^2 + 12(1-q)^2 & 0 \leq q < 1 \\ -3(2-q)^2 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases}$$

$$\nabla W(\boldsymbol{x}_j - \boldsymbol{x}_i) = \alpha \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\| h} \begin{cases} -3(2-q)^2 + 12(1-q)^2 & 0 \leq q < 1 \\ -3(2-q)^2 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases}$$

Kernel Derivative - Illustration

∇W



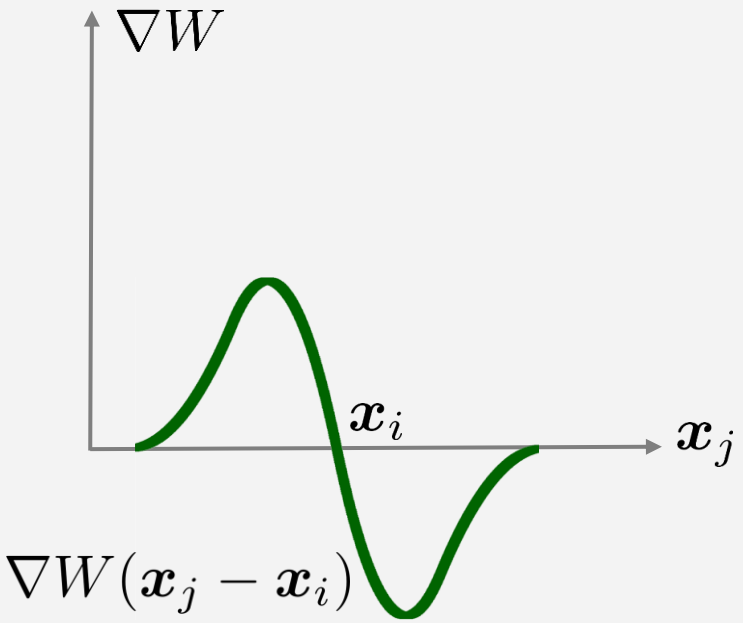
$$\nabla W(\boldsymbol{x}_j - \boldsymbol{x}_i) = \alpha \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\| h} \begin{cases} -3(2 - \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h})^2 + 12(1 - \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h})^2 & 0 \leq \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h} < 1 \\ -3(2 - \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h})^2 & 1 \leq \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h} < 2 \\ 0 & \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{h} \geq 2 \end{cases}$$

$$\nabla W(\boldsymbol{x}_j - \boldsymbol{x}_i) = -\nabla W(\boldsymbol{x}_i - \boldsymbol{x}_j)$$

$\boldsymbol{x}_j - \boldsymbol{x}_i$

Convolution with First Kernel Derivative

Kernel derivative



$$\nabla W(\boldsymbol{x}_j - \boldsymbol{x}_i) = \alpha \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{\|\boldsymbol{x}_{ji}\| h} \dots$$

SPH computes a convolution of A and ∇W to approximate ∇A. Therefore, the reversed kernel derivative $\nabla W(\boldsymbol{x}_i - \boldsymbol{x}_j)$ is used:

$$\nabla A(\boldsymbol{x}_i) = \sum_j A(\boldsymbol{x}_j) \nabla W(\boldsymbol{x}_i - \boldsymbol{x}_j) \frac{m(\boldsymbol{x}_j)}{\rho(\boldsymbol{x}_j)}$$

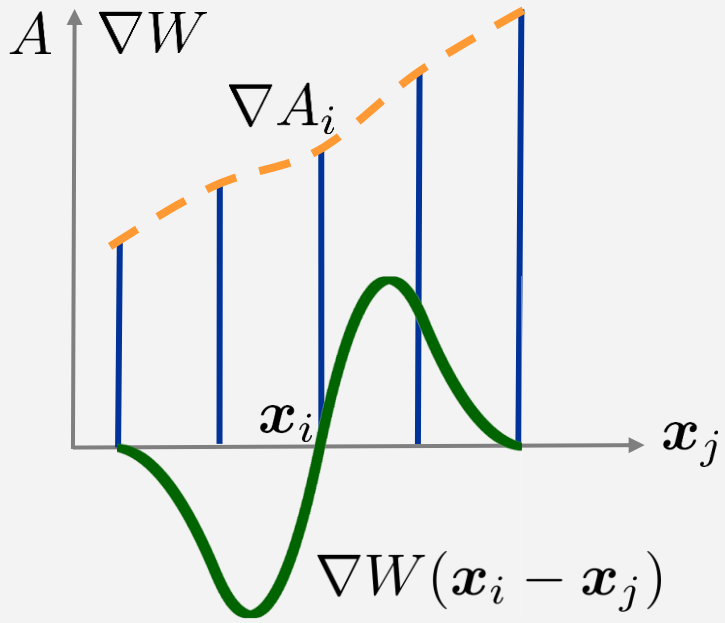
SPH notation:

$$\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W_{ij}$$

Convolution:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) \mathrm{d}\tau$$

Reversed kernel derivative



$$\nabla W(\boldsymbol{x}_i - \boldsymbol{x}_j) = \alpha \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\|\boldsymbol{x}_{ij}\| h} \dots$$

$$= -\nabla W(\boldsymbol{x}_j - \boldsymbol{x}_i) = \nabla W_{ij}$$

∇W is anti-symmetric.

Kernel Derivative - Implementation

- Reversed kernel derivative as used in SPH sums for the convolution

$$\nabla W(\boldsymbol{x}_i - \boldsymbol{x}_j) = \alpha \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\| h} \begin{cases} -3(2 - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h})^2 + 12(1 - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h})^2 & 0 \leq \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h} < 1 \\ -3(2 - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h})^2 & 1 \leq \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h} < 2 \\ 0 & \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h} \geq 2 \end{cases}$$

- Implementation

```
d  := distance(xi,xj)/h;
t1 := max(1-d,0);
t2 := max(2-d,0);
w1 := alpha * (xi-xj)/(d*h*h) * (-3*t2*t2 + 12*t1*t1);
```

Second Kernel Derivative

- $\nabla^2 W_{ji} = \nabla \cdot (\nabla W_{ji}) = \frac{\partial^2 W}{\partial x_{j,x}^2} + \frac{\partial^2 W}{\partial x_{j,y}^2} + \frac{\partial W^2}{\partial x_{j,z}^2}$

  $= \frac{\partial^2 W(q)}{\partial q^2}(\nabla q)^2 + \frac{\partial W(q)}{\partial q}(\nabla \cdot (\nabla q))$

- Cubic spline

  $q = \frac{\|\boldsymbol{x}_{ji}\|}{h} \quad (\nabla q)^2 = \frac{\boldsymbol{x}_{ji}}{\|\boldsymbol{x}_{ji}\|h} \cdot \frac{\boldsymbol{x}_{ji}}{\|\boldsymbol{x}_{ji}\|h} = \frac{\|\boldsymbol{x}_{ji}\|^2}{\|\boldsymbol{x}_{ji}\|^2 h^2} = \frac{1}{h^2}$

  $\nabla \cdot (\nabla q) = \frac{d-1}{h\|\boldsymbol{x}_{ji}\|}$ *d* is the dimensionality

  $$\frac{\partial W(q)}{\partial q} = \alpha \begin{cases} -3(2-q)^2 + 12(1-q)^2 & 0 \le q < 1 \\ -3(2-q)^2 & 1 \le q < 2 \\ 0 & q \ge 2 \end{cases} \qquad \frac{\partial^2 W(q)}{\partial q^2} = \alpha \begin{cases} 6(2-q) - 24(1-q) & 0 \le q < 1 \\ 6(2-q) & 1 \le q < 2 \\ 0 & q \ge 2 \end{cases}$$

- Symmetric $\nabla^2 W_{ji} = \nabla^2 W_{ij}$ Used in SPH convolutions

Design of a Kernel Function - 1D

– Definition of a shape, followed by normalization

$$\alpha\tilde{W}(\tfrac{\|\mathbf{x}_j-\mathbf{x}_i\|}{h}) = \alpha\tilde{W}(\tfrac{x}{h}) = \alpha\tilde{W}(q) = W(q) = \alpha \begin{cases} (2-q)^3 - 4(1-q)^3 & 0 \le q < 1 \\ (2-q)^3 & 1 \le q < 2 \\ 0 & q \ge 2 \end{cases}$$

$$2\int_0^{2h} \alpha\tilde{W}(x)\mathrm{d}x = 2\int_0^2 \alpha\tilde{W}(q)h\mathrm{d}q = 1$$ Integration by substitution

$$\alpha = \frac{1}{2\int_0^2 \tilde{W}(q)h\mathrm{d}q}$$

– 1D: integration over a line segment $2\int_0^2 \tilde{W}(q)h\mathrm{d}q =$

$$2\int_0^1 \left[(2-q)^3 - 4(1-q)^3\right] h\mathrm{d}q + 2\int_1^2 (2-q)^3 h\mathrm{d}q = 2\tfrac{11}{4}h + 2\tfrac{1}{4}h$$

$$\alpha = \frac{1}{6h}$$

Design of a Kernel Function – 2D, 3D

- 2D: Integration over the area of a circle

$$\int_0^{2\pi}\int_0^{2h}\tilde{W}(x)x\,\mathrm{d}x\mathrm{d}\phi = \int_0^{2\pi}\int_0^{2}\tilde{W}(q)hqh\,\mathrm{d}q\mathrm{d}\phi =$$

$$2\pi\int_0^1\left[q(2-q)^3-4q(1-q)^3\right]h^2\mathrm{d}q+2\pi\int_1^2 q(2-q)^3h^2\mathrm{d}q = 2\pi\tfrac{11}{10}h^2+2\pi\tfrac{3}{10}h^2$$

$$\alpha = \frac{5}{14\pi h^2}$$

- 3D: Integration over the volume of a sphere

$$\int_0^{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\int_0^{2h}\tilde{W}(x)x^2\sin\theta\,\mathrm{d}x\mathrm{d}\theta\mathrm{d}\phi = \int_0^{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\int_0^{2}\tilde{W}(q)(qh)^2h\sin\theta\,\mathrm{d}q\mathrm{d}\theta\mathrm{d}\phi =$$

$$4\pi\int_0^1\left[q^2(2-q)^3-4q(1-q)^3\right]h^3\mathrm{d}q+4\pi\int_1^2 q^2(2-q)^3h^3\mathrm{d}q = 4\pi\tfrac{19}{30}h^3+4\pi\tfrac{11}{30}h^3$$

$$\alpha = \frac{1}{4\pi h^3}$$

Derivatives – Original SPH Forms

- Original forms

$$\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W_{ij}$$
$$\nabla^2 A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W_{ij}$$

- However, resulting forces do not preserve momentum and are not necessarily zero for constant values $A_i = A_j$ in case of erroneous sampling

First Derivative - Anti-symmetric Form

- Momentum-preserving form $-\frac{1}{\rho_i}\nabla p_i = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij}$

  $$\nabla\left(\frac{A_i}{\rho_i}\right) = \frac{\rho_i \nabla A_i - A_i \nabla \rho_i}{\rho_i^2} = \frac{\nabla A_i}{\rho_i} - \frac{A_i \nabla \rho_i}{\rho_i^2}$$

  $$\nabla A_i = \rho_i \left(\nabla\left(\frac{A_i}{\rho_i}\right) + \frac{A_i \nabla \rho_i}{\rho_i^2}\right)$$

- SPH approximation

  $$\nabla A_i = \rho_i \left(\sum_j \frac{m_j}{\rho_j}\frac{A_j}{\rho_j}\nabla W_{ij} + A_i \sum_j \frac{m_j}{\rho_j}\frac{\rho_j}{\rho_i^2}\nabla W_{ij}\right)$$

  $$= \rho_i \sum_j m_j \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2}\right)\nabla W_{ij}$$

- Applied to pressure gradient, linear and angular momentum is preserved for arbitrary samplings

$$\boldsymbol{F}_i^{\mathrm{p}} = -m_i m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij} = m_j m_i \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2}\right)\nabla W_{ji} = -\boldsymbol{F}_j^{\mathrm{p}} \quad \nabla W_{ij} = -\nabla W_{ji}$$

First Derivative – Symmetric Form

- Term that vanishes for constant function values

$$\nabla(\rho_i A_i) = \rho_i \nabla(A_i) + A_i \nabla(\rho_i)$$

$$\nabla A_i = \frac{1}{\rho_i}\left(\nabla(\rho_i A_i) - A_i \nabla \rho_i\right)$$

- SPH approximation

$$\nabla A_i = \frac{1}{\rho_i}\left(\sum_j \frac{m_j}{\rho_j}\rho_j A_j \nabla W_{ij} - A_i \sum_j \frac{m_j}{\rho_j}\rho_j \nabla W_{ij}\right)$$

$$= \frac{1}{\rho_i}\sum_j m_j (A_j - A_i)\nabla W_{ij} = \frac{1}{\rho_i}\sum_j m_j A_{ji} \nabla W_{ij}$$

- Applied to velocity divergence, zero divergence for a constant velocity field is obtained for arbitrary samplings

Second Derivative with First Kernel Derivative

- Second derivative is error prone and sensitive to particle disorder
- Too few samples to appropriately approximate the second kernel derivative
- Therefore, the Laplacian is typically approximated with a finite difference approximation of the first derivative

$$\nabla^2 A_i = d \sum_j \frac{m_j}{\rho_j} \frac{A_{ij} \cdot \boldsymbol{x}_{ij}}{\boldsymbol{x}_{ij} \cdot \boldsymbol{x}_{ij} + 0.01h^2} \nabla W_{ij}$$

d is the dimensionality

$$A_{ij} = A_i - A_j \quad \boldsymbol{x}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$$

Derivatives - Summary

- Original approximations

$$\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W_{ij} \qquad \nabla^2 A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W_{ij}$$

- Currently preferred approximations
  - Robustness in case of particle disorder, i.e. $\sum_j \nabla W_{ij} \neq \mathbf{0}$

$$\nabla p_i = \rho_i \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij}$$

Preserves linear and angular momentum

$$\nu \nabla^2 \boldsymbol{v}_i = d \sum_j \frac{m_j}{\rho_j} \frac{\boldsymbol{v}_{ij} \cdot \boldsymbol{x}_{ij}}{\boldsymbol{x}_{ij} \cdot \boldsymbol{x}_{ij} + 0.01 h^2} \nabla W_{ij}$$

Avoids the second kernel derivative

$$\nabla \cdot \boldsymbol{v}_i = -\frac{1}{\rho_i} \sum_j m_j \boldsymbol{v}_{ij} \nabla W_{ij}$$

Zero for uniform velocity field

$$\boldsymbol{v}_{ij} = \boldsymbol{v}_i - \boldsymbol{v}_j \qquad \boldsymbol{x}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$$

Some Kernel Properties

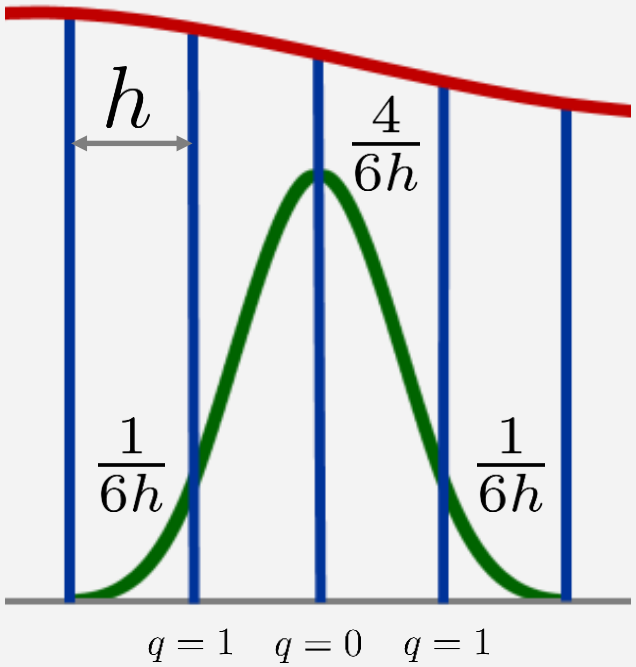
- In case of ideal sampling
- $\rho_i = \sum_j m_j W_{ij} = m_i \sum_j W_{ij} \quad m_i = m_j$
- $m_i \sum_j W_{ij} = \rho_i = \frac{m_i}{V_i} \quad \Rightarrow \quad \sum_j W_{ij} = \frac{1}{V_i} = \frac{\rho_i}{m_i}$
- $\nabla W_{ij} = -\nabla W_{ji} \quad \nabla W_{ij} = \alpha \frac{\boldsymbol{x}_{ij}}{\|\boldsymbol{x}_{ij}\| h} \dots \quad (\nabla W_{ii} = \mathbf{0})$
- $\sum_j \nabla W_{ij} = \mathbf{0}$
- $\sum_j (\boldsymbol{x}_i - \boldsymbol{x}_j) \otimes \nabla W_{ij} = -\frac{1}{V_i} \cdot \boldsymbol{I}$
- Can be used for test purposes

Kernel Illustration – 1D

$$W(q) = \frac{1}{6h} \begin{cases} (2-q)^3 - 4(1-q)^3 & 0 \leq q < 1 \\ (2-q)^3 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases} \qquad q = \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{h}$$

$$W(0) = \frac{1}{6h}\left((2-0)^3 - 4(1-0)^3\right) = \frac{4}{6h}$$
$$W(1) = \frac{1}{6h}(2-1)^3 = \frac{1}{6h}$$
$$W(2) = 0$$

$$\sum_j W_{ij} = W(0) + 2W(1) + 2W(2) = \frac{1}{h}$$

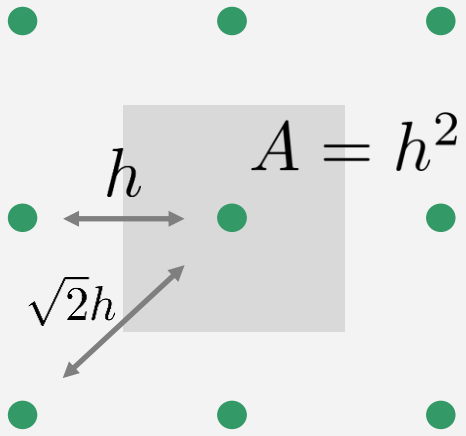


Kernel Illustration – 2D

$$W(q) = \frac{5}{14\pi h^2} \begin{cases} (2-q)^3 - 4(1-q)^3 & 0 \leq q < 1 \\ (2-q)^3 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases} \qquad q = \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{h}$$

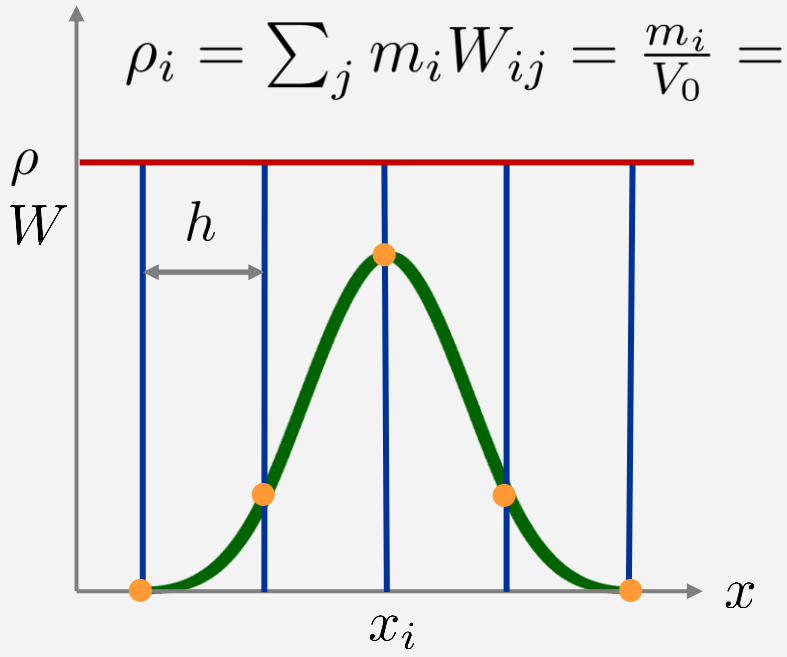
$$W(0) = \frac{5}{14\pi h^2}\left((2-0)^3 - 4(1-0)^3\right) = \frac{20}{14\pi h^2}$$
$$W(1) = \frac{5}{14\pi h^2}(2-1)^3 = \frac{5}{14\pi h^2}$$
$$W(\sqrt{2}) = \frac{5}{14\pi h^2}(2-\sqrt{2})^3 \approx \frac{1.005}{14\pi h^2}$$

$$\sum_j W_{ij} = W(0) + 4W(1) + 4W(\sqrt{2}) \approx \frac{1.001}{h^2}$$

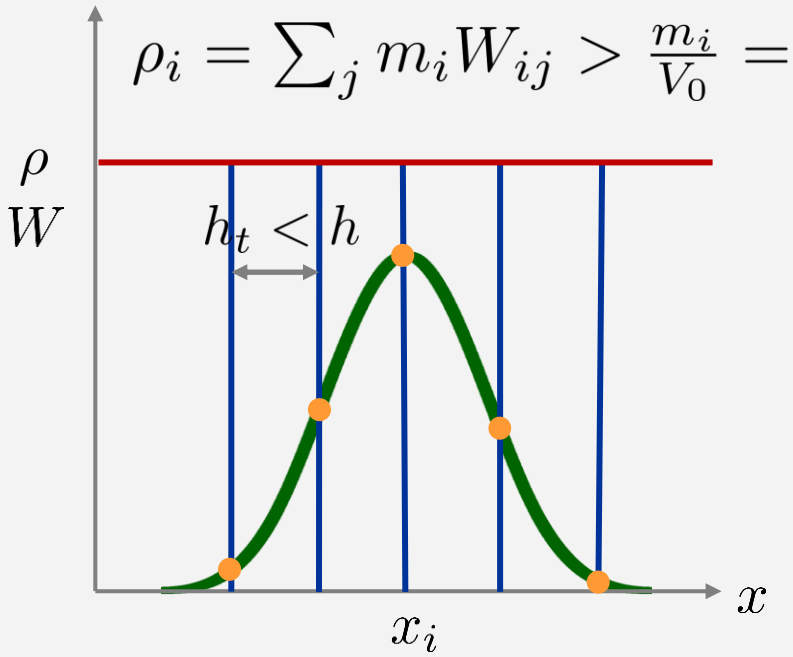


Kernel Illustration – Density Computation

– Is not a reconstruction of the function ρ, but detects erroneous sampling



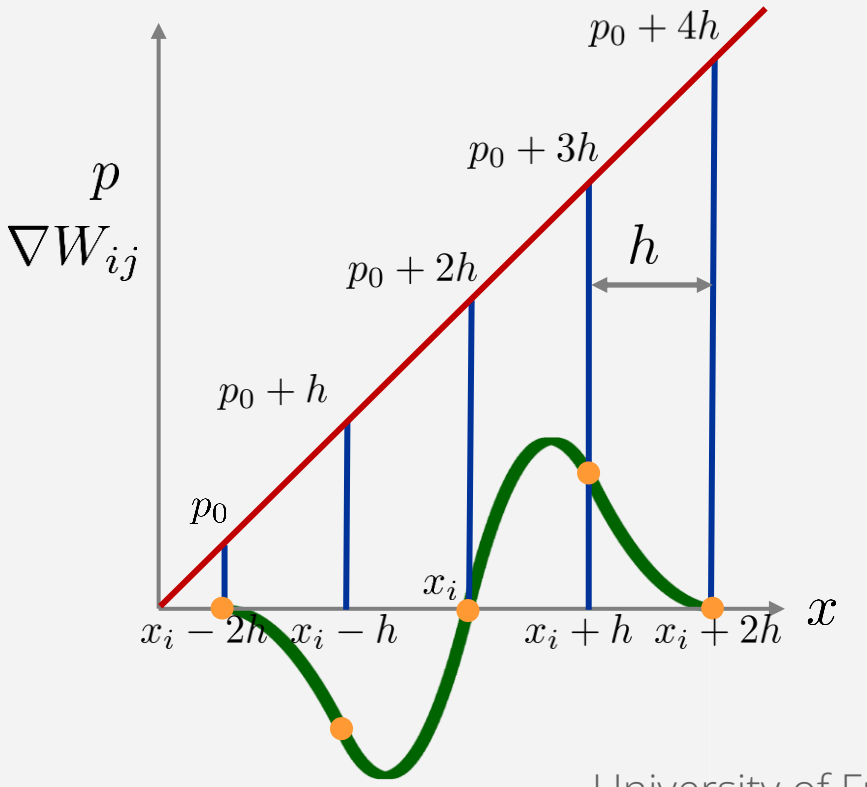
Correct sampling



Dense irregular sampling

Kernel Derivative Illustration – 1D

$$\nabla W_{ij} = \nabla W(x_i - x_j) = \frac{1}{6h}\frac{x_i - x_j}{\|x_i - x_j\| h} \begin{cases} -3(2 - \frac{\|x_i - x_j\|}{h})^2 + 12(1 - \frac{\|x_i - x_j\|}{h})^2 & 0 \leq \frac{\|x_i - x_j\|}{h} < 1 \\ -3(2 - \frac{\|x_i - x_j\|}{h})^2 & 1 \leq \frac{\|x_i - x_j\|}{h} < 2 \\ 0 & \frac{\|x_i - x_j\|}{h} \geq 2 \end{cases}$$



$$\nabla W(x_i - x_i) = 0$$

$$\nabla W(x_i - (x_i - 2h)) = \nabla W(x_i - (x_i + 2h) = 0$$

$$\nabla W(x_i - (x_i - h)) = -\nabla W(x_i - (x_i + h)) = \frac{1}{6h} \cdot \frac{1}{h} \cdot (-3)$$

$$\nabla p_i = \sum_j p_j \nabla W(x_i - x_j) h = -\frac{h}{2h^2}(p_0 + h) + \frac{h}{2h^2}(p_0 + 3h)$$

$$= \frac{(p_0 + 3h) - (p_0 + h)}{2h} = 1$$ central difference

Kernel Derivative Illustration – 2D Test

$$\nabla W(\boldsymbol{x}_i - \boldsymbol{x}_j) = \alpha \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\| h} \begin{cases} -3(2 - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h})^2 + 12(1 - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h})^2 & 0 \leq \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h} < 1 \\ -3(2 - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h})^2 & 1 \leq \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h} < 2 \\ 0 & \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h} \geq 2 \end{cases} \qquad \alpha = \frac{5}{14\pi h^2}$$

$$\nabla W((0,0) - (0,0)) = (0,0)$$

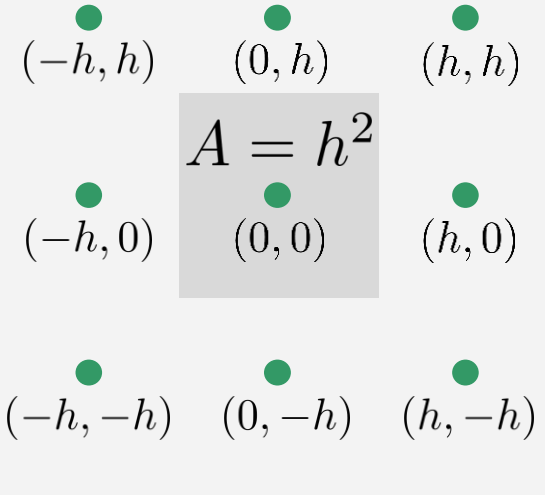
$$\nabla W((0,0) - (h,0)) = -\nabla W((0,0) - (-h,0)) = \alpha \frac{(0,0)-(h,0)}{h^2}(-3) = (\frac{3\alpha}{h}, 0)$$

$$\nabla W((0,0) - (0,h)) = -\nabla W((0,0) - (0,-h)) = \alpha \frac{(0,0)-(0,h)}{h^2}(-3) = (0, \frac{3\alpha}{h})$$

$$\nabla W((0,0)-(h,h)) = -\nabla W((0,0)-(-h,-h)) = \alpha \frac{(0,0)-(h,h)}{h^2\sqrt{2}}\beta = (-\frac{1}{h\sqrt{2}}\alpha\beta, -\frac{1}{h\sqrt{2}}\alpha\beta)$$

$$\nabla W((0,0)-(h,-h)) = -\nabla W((0,0)-(-h,h)) = \alpha \frac{(0,0)-(h,-h)}{h^2\sqrt{2}}\beta = (-\frac{1}{h\sqrt{2}}\alpha\beta, \frac{1}{h\sqrt{2}}\alpha\beta)$$

$$\beta = (-3)(2 - \sqrt{2})^2$$



Kernel Derivative Illustration – 2D Test

$$\nabla W((0,0)-(0,0)) = (0,0)$$

$$\alpha = \frac{5}{14\pi h^2}$$

$$\beta = (-3)(2-\sqrt{2})^2$$

$$\nabla W((0,0)-(h,0)) = -\nabla W((0,0)-(-h,0)) = \alpha\frac{(0,0)-(h,0)}{h^2}(-3) = (\frac{3\alpha}{h},0)$$

$$\nabla W((0,0)-(0,h)) = -\nabla W((0,0)-(0,-h)) = \alpha\frac{(0,0)-(0,h)}{h^2}(-3) = (0,\frac{3\alpha}{h})$$

$$\nabla W((0,0)-(h,h)) = -\nabla W((0,0)-(-h,-h)) = \alpha\frac{(0,0)-(h,h)}{h^2\sqrt{2}}\beta = (-\frac{1}{h\sqrt{2}}\alpha\beta,-\frac{1}{h\sqrt{2}}\alpha\beta)$$

$$\nabla W((0,0)-(h,-h)) = -\nabla W((0,0)-(-h,h)) = \alpha\frac{(0,0)-(h,-h)}{h^2\sqrt{2}}\beta = (-\frac{1}{h\sqrt{2}}\alpha\beta,\frac{1}{h\sqrt{2}}\alpha\beta)$$

$$(\boldsymbol{x}_i-\boldsymbol{x}_j)_x\cdot(\nabla W_{ij})_y = (\boldsymbol{x}_i-\boldsymbol{x}_j)_y\cdot(\nabla W_{ij})_x = 0+0+0+0+0+\frac{1}{\sqrt{2}}\alpha\beta+\frac{1}{\sqrt{2}}\alpha\beta-\frac{1}{\sqrt{2}}\alpha\beta-\frac{1}{\sqrt{2}}\alpha\beta = 0$$

$$(\boldsymbol{x}_i-\boldsymbol{x}_j)_x\cdot(\nabla W_{ij})_x = (\boldsymbol{x}_i-\boldsymbol{x}_j)_y\cdot(\nabla W_{ij})_y = 0-3\alpha-3\alpha+0+0+\frac{1}{\sqrt{2}}\alpha\beta+\frac{1}{\sqrt{2}}\alpha\beta+\frac{1}{\sqrt{2}}\alpha\beta+\frac{1}{\sqrt{2}}\alpha\beta = -6\alpha+4\frac{1}{\sqrt{2}}\alpha\beta$$

$$= -0.682\frac{1}{h^2} - 0.331\frac{1}{h^2} = -\frac{1.013}{h^2}$$

$$\Rightarrow \quad \sum_j(\boldsymbol{x}_i-\boldsymbol{x}_j)\otimes\nabla W_{ij} = -\frac{1}{A_i}\cdot\boldsymbol{I}$$

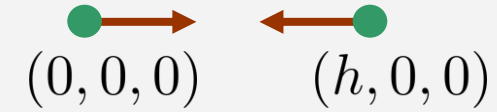
Kernel Derivative Illustration – 3D Sampling

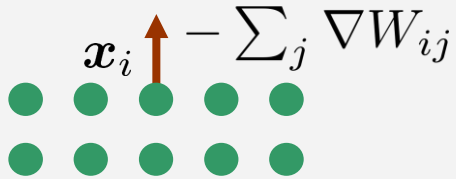
– Kernel derivative detects irregular samplings (vector from low to high sample concentration)

$$\nabla W(\boldsymbol{x}_i - \boldsymbol{x}_j) = \alpha \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\| h} \begin{cases} -3(2 - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h})^2 + 12(1 - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h})^2 & 0 \leq \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h} < 1 \\ -3(2 - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h})^2 & 1 \leq \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h} < 2 \\ 0 & \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{h} \geq 2 \end{cases} \quad \alpha = \frac{1}{4\pi h^3}$$

$$\nabla W((0,0,0) - (h,0,0)) = -3\alpha \frac{(0,0,0) - (h,0,0)}{h^2} = \frac{3\alpha}{h}(1,0,0)$$

$$\nabla W((h,0,0) - (0,0,0)) = -3\alpha \frac{(h,0,0) - (0,0,0)}{h^2} = -\frac{3\alpha}{h}(1,0,0)$$





Simulation in Computer Graphics Particle Fluids 3

Matthias Teschner

UNI FREIBURG

Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
- Boundary handling
- Incompressibility

Simple SPH Fluid Solver

for all *particle* i **do**
  find neighbors j

for all *particle* i **do**
  $\rho_i = \sum_j m_j W_{ij}$ Compute density
  $p_i = k(\frac{\rho_i}{\rho_0} - 1)$ Compute pressure

for all *particle* i **do**
  $\boldsymbol{a}_i^{\text{nonp}} = \nu \nabla^2 \boldsymbol{v}_i + \boldsymbol{g}$ Compute non-pressure accelerations
  $\boldsymbol{a}_i^{\text{p}} = -\frac{1}{\rho_i} \nabla p_i$ Compute pressure acceleration
  $\boldsymbol{a}_i(t) = \boldsymbol{a}_i^{\text{nonp}} + \boldsymbol{a}_i^{\text{p}}$

for all *particle* i **do**
  $\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i(t) + \Delta t \boldsymbol{a}_i(t)$
  $\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i(t + \Delta t)$

SPH Simulation Step With a State Equation (SESPH)

- For each particle do
  - Compute density
  - Compute pressure
- For each particle do
  - Compute accelerations
  - Update velocities and positions
- Density and acceleration computations
  process all neighbors of a particle

Neighbor Search

- For the computation of SPH sums in 3D, each particle needs to know at least 30-40 neighbors in each step
- Example setting
  - 30 million fluid particles
  - Up to 1 billion neighbors
  - 10000 simulation steps
  - Up to 10^{13} neighbors processed per simulation
- Efficient construction and processing of dynamically changing neighbor sets is essential

Performance – [Ihmsen et al. 2011]

Up to 30 million fluid particles,
11 s computation time for
neighbor search on a 16-core PC

Performance – [Band et al. 2019]

Million samples	Neighbor search [s]	Simulation step [s]	Memory [GB]	Cores
0.2	0.005	0.04	0.1	12
200	5	15	28	12
1300	33	355	172	24
7400	67	530	873	112

Characteristics

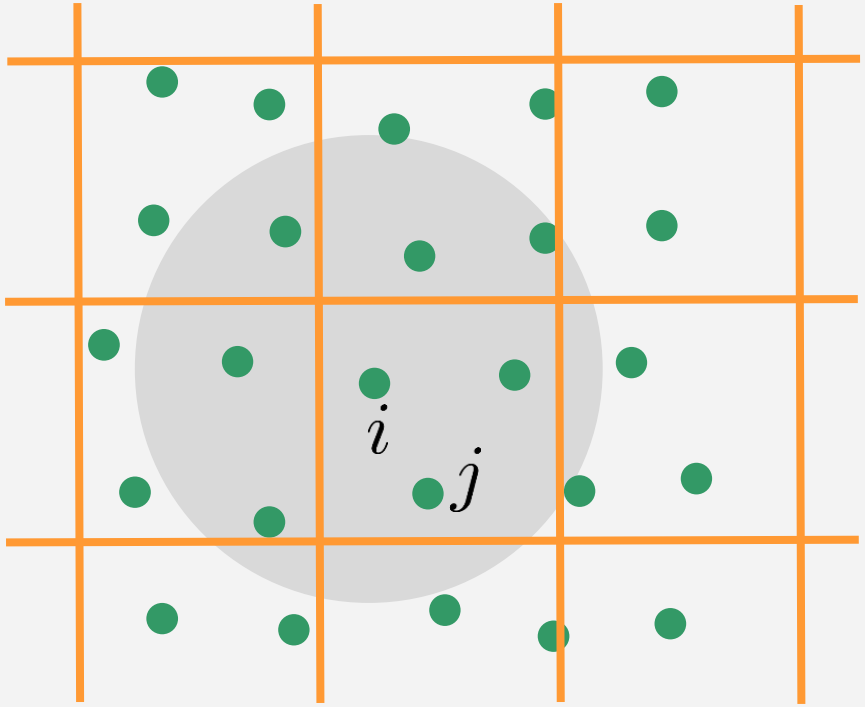
- SPH computes sums
  - Dynamically changing sets of neighboring particles
  - Temporal coherence
- Spatial data structures accelerate the neighbor search
  - Fast query
  - Fast generation (at least once for each simulation step)
  - Sparsely, non-uniformly filled simulation domain

Characteristics

- Space subdivision
  - Each particle is placed in a convex space cell, e.g. a cube
- Similarities to collision detection and intersection tests in raytracing
  - However, cells adjacent to the cell of a particle have to be accessed

Characteristics

- Hierarchical data structures are less efficient
  - Construction in $O(n \log n)$, access in $O(\log n)$
- Uniform grid is generally preferred
  - Construction in $O(n)$, access in $O(1)$



Characteristics

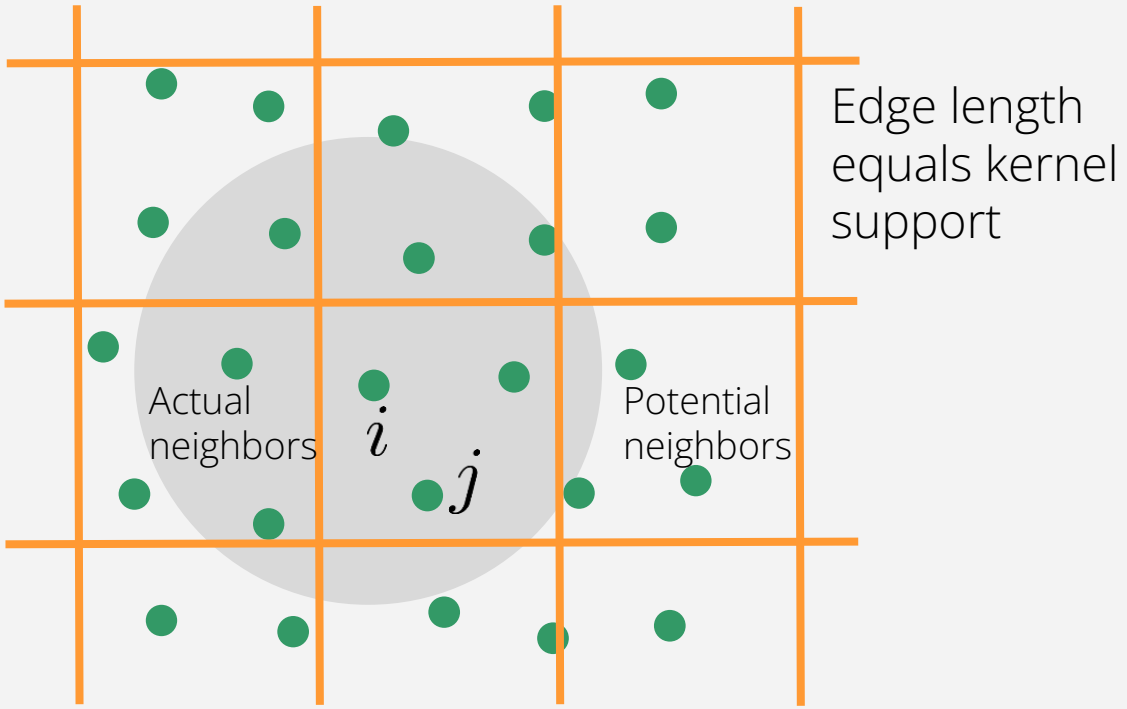
- Neighbor storage is generally expensive
  - Might be avoided for, e.g., a low number of neighbor queries per step or in case of very efficient computation
- Data structures have to process
  - Fluid particles of multiple phases, e.g. air
  - Rigid particles (static or moving)
  - Deformable particles

Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
  - Uniform grid
  - Index sort
  - Spatial hashing
  - Discussion
- …

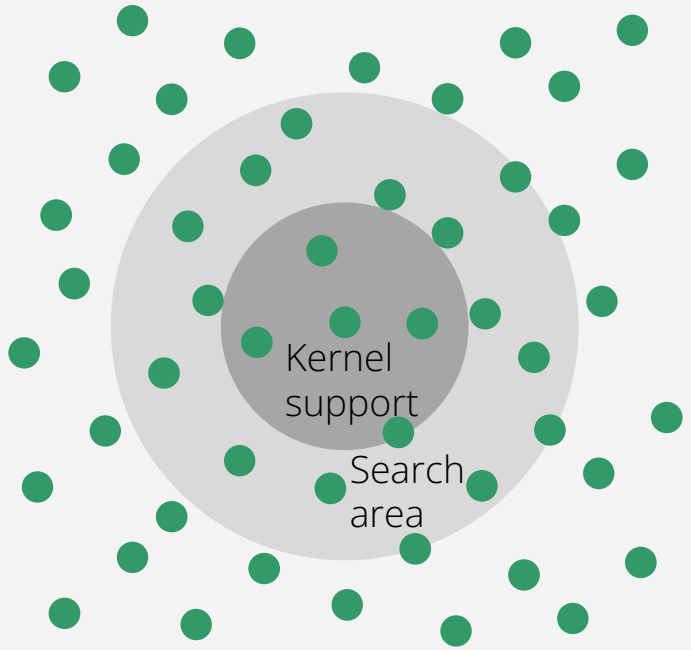
Concept

- Particle is stored in a cell
- In d-D, potential neighbors in 3^d cells are queried to estimate actual neighbors
- Cell size equals the kernel support of a particle
  - Smaller cells increase the number of tested cells
  - Larger cells increase the number of tested particles



Concept - Variant

- Verlet lists
  - Neighbor candidates are computed within a distance larger than the kernel support every n^{th} step
  - Actual neighbors are computed from neighbor candidates in each step
  - Neighbor candidates are valid for *n* steps
  - Motivated by temporal coherence: Particle does not move farther than its size in one step.



Concept - Variant

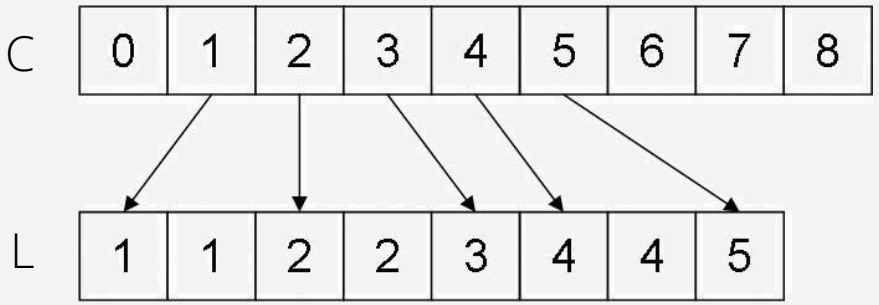
- Verlet lists
  - Proposed in 1967
  - Still popular in Lagrangian simulations
  - Acceleration data structure
    - Is only updated every n^{th} step
    - Is memory-intensive, requires storage of a comparatively large number of neighbor candidates

Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
  - Uniform grid
  - Index sort
  - Spatial hashing
  - Discussion
- …

Construction

- Compute cell index $c = k + l \cdot K + m \cdot K \cdot L$ for all particles with, e.g. $(k, l, m) = \lfloor \frac{x - x_{min}}{2h}, \frac{y - y_{min}}{2h}, \frac{z - z_{min}}{2h} \rfloor$
  - *K* and *L* denote the number of cells in *x* and *y* direction
- Particles are sorted with respect to their cell index
- Each grid cell (k, l, m) with index c stores a reference to the first particle in the sorted list

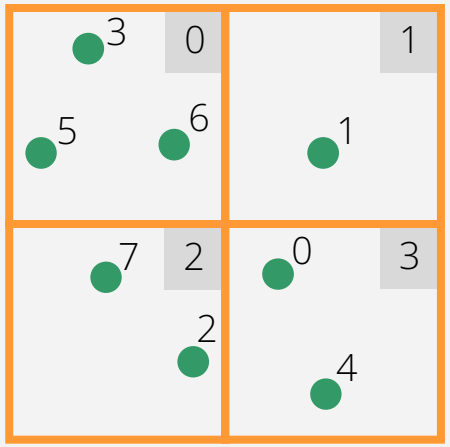


Cell indices of a linearized uniform grid

Cell indices of particles

Sorted with respect to cell index

Construction



Compute cell indices for particles and increment counter in C

C	0	1	2	3	4
counter	3	1	2	2	

Accumulate counters in C

C	0	1	2	3	4
accum	3	4	6	8	8

Associate particle i with cell j: L [-- C [j].counter].particle = i

C	0	1	2	3	4
index	3	4	6	7	8

L	0	1	2	3	4	5	6	7
particle								0

C	0	1	2	3	4
index	3	3	6	7	8

L	0	1	2	3	4	5	6	7
particle				1				0

C	0	1	2	3	4
index	3	3	5	7	8

L	0	1	2	3	4	5	6	7
particle				1		2		0

...

C	0	1	2	3	4
index	0	3	4	6	8

L	0	1	2	3	4	5	6	7
particle	6	5	3	1	7	2	4	0

Construction

C	0	1	2	3	4
index	0	3	4	6	8

L	0	1	2	3	4	5	6	7
particle	6	5	3	1	7	2	4	0

- Particles are sorted with respect to grid cell
- Index points to first particle in a cell
- Difference of two subsequent indices indicates the particle number of a grid cell

Construction

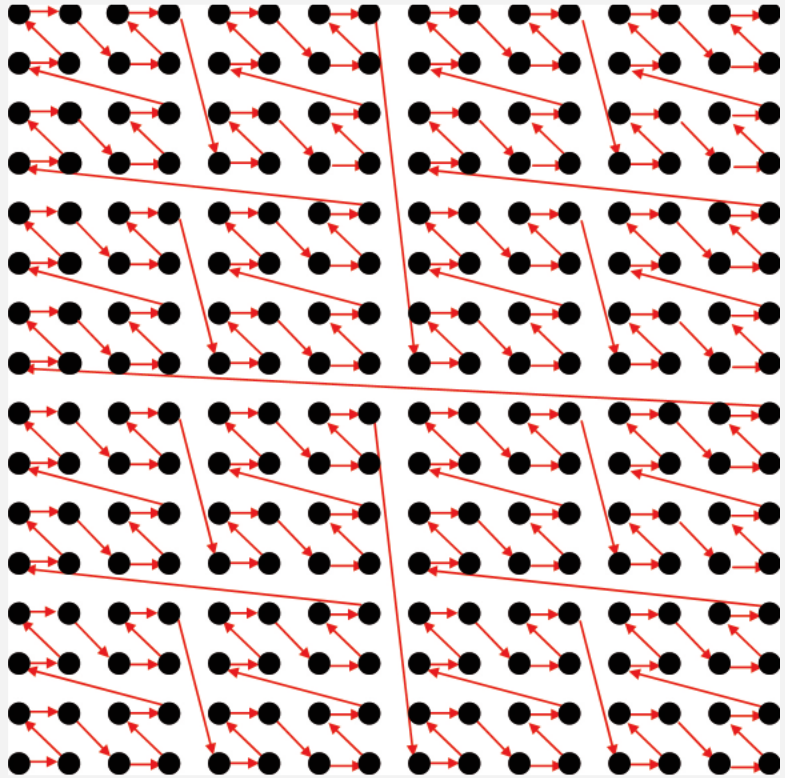
- Two iterations over particles
- One iteration over grid cells
- Entire simulation domain has to be represented
- Parallelizable
- Memory allocations are avoided
- Constant memory consumption

Query

- For a particle
  - Indices to grid cell and to adjacent cells are computed (Once for all particles in the same grid cell)
  - All particles in grid cell and adjacent cells are tested
- Parallelizable
- Improved cache-hit rate
  - Particles in the same cell are close in memory
  - Particles of neighboring cells are not necessarily close in memory

Space-filling Curves

- Alternative computation for grid cell indices
- E.g., particles are sorted with respect to a z-curve index
- Improved cache-hit rate
  - Particles in adjacent cells are close in memory
- Efficient computation of z-curve indices



z-curve

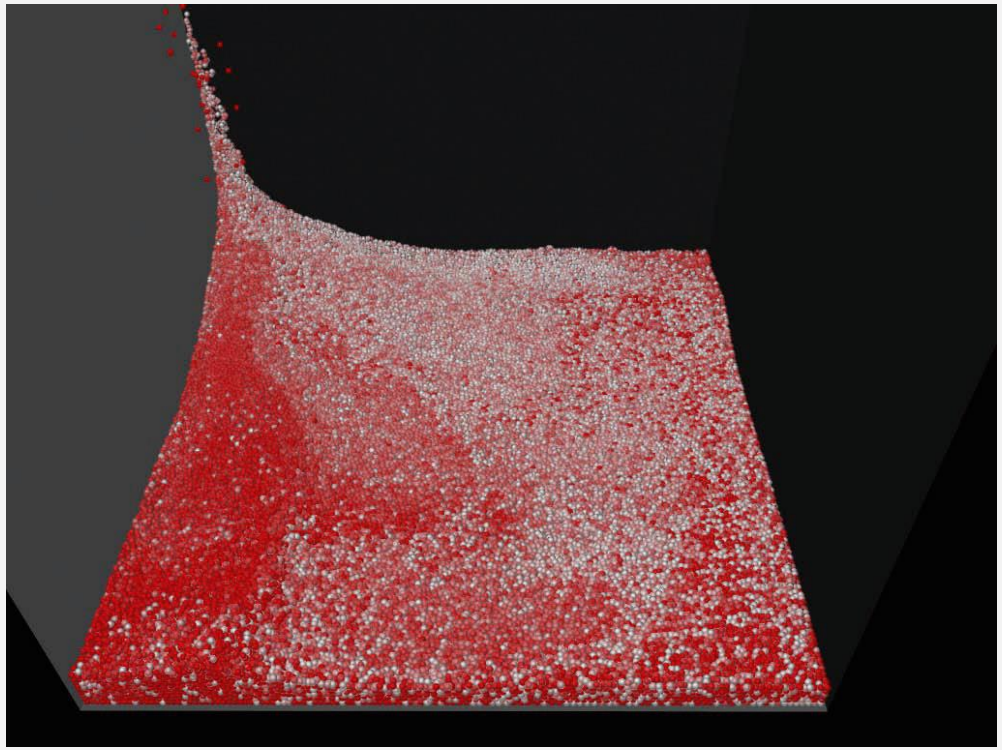
Sorting

- Particle attributes and z-curve indices can be processed separately
- Handles (particle identifier, z-curve index) are sorted in each time step
  - Reduced memory transfer
  - Spatial locality is only marginally influenced due to temporal coherence
- Attribute sets are sorted every n^{th} step
  - Restores spatial locality

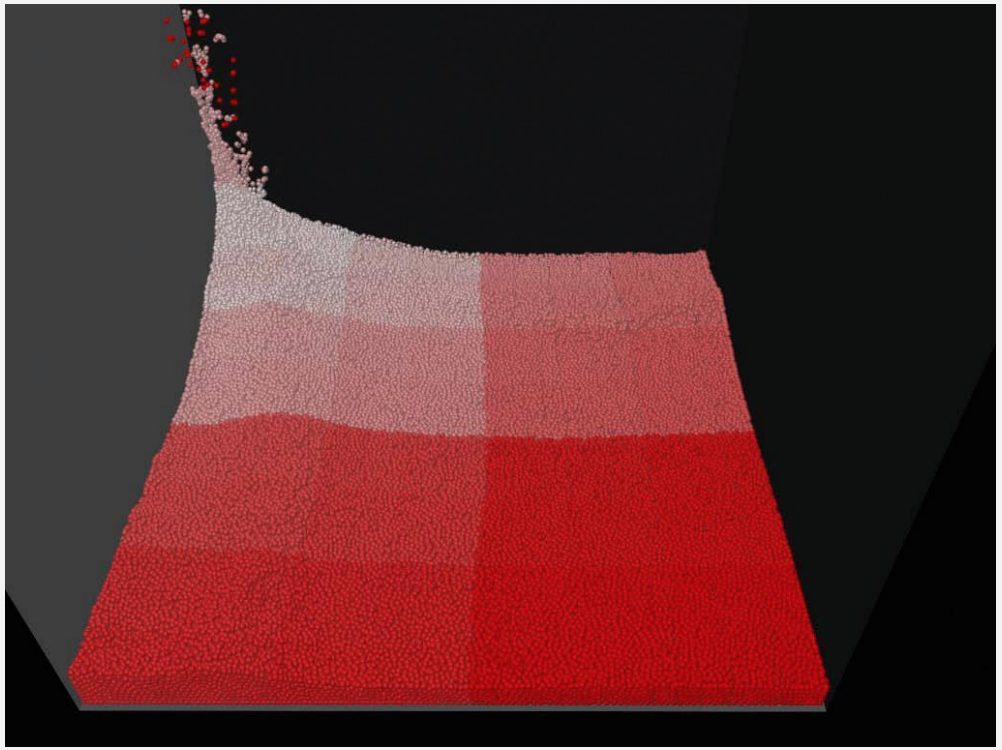
Sorting

- Radix sort or insertion sort can be employed
  - $O(n)$ for almost sorted arrays
  - Due to temporal coherence, a small percentage of all particles change their cell, i.e. z-curve index, in each step

Z-Index Sort - Reordering



Particle color indicates
memory location



Spatial compactness
using a z-curve

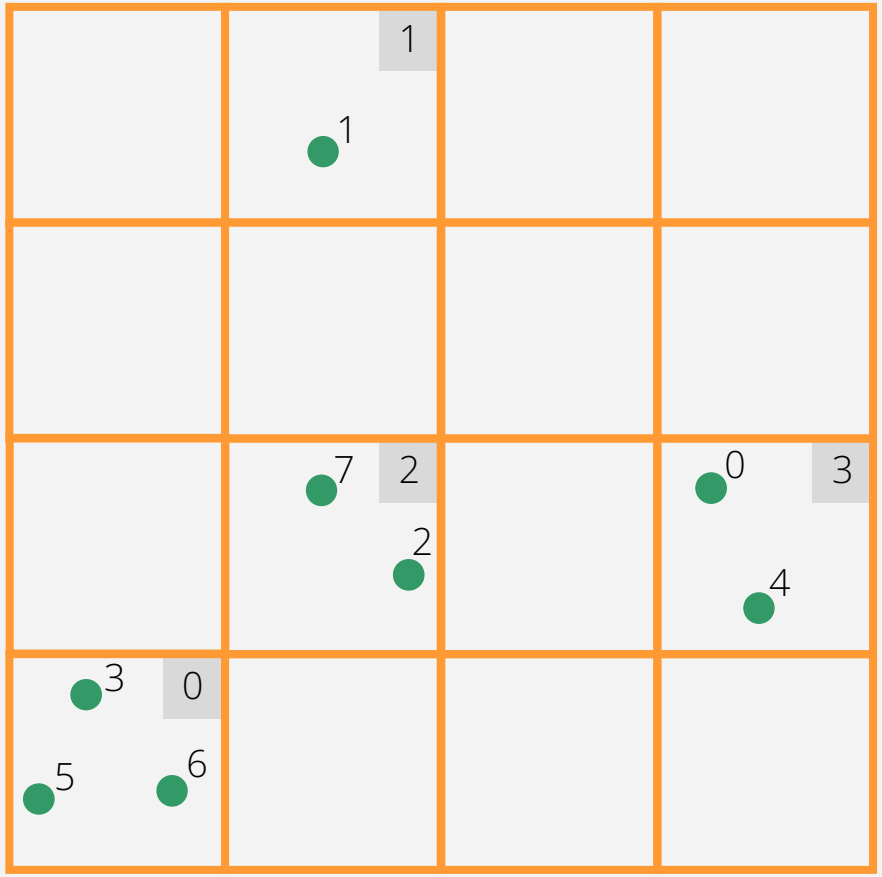
Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
  - Uniform grid
  - Index sort
  - Spatial hashing
  - Discussion
- …

Spatial Hashing

- Hash function maps a grid cell to a hash cell
  - 3D domain is mapped to a finite 1D list
  - Infinite domains can be handled
- Implementation
  - Compute a cell index c or a cell identifier (k, l, m) for a particle
  - Compute a hash function $i = h(c)$ or $i = h(k, l, m)$
  - Store the particle in a 1D array (hash table) at index i

Spatial Hashing



3D Grid

$i = h(c)$ Hash function

$3 = h(0)$

$1 = h(1)$

$4 = h(2)$

$7 = h(3)$

	0	1	2	3	4	5	6	7	8	...	n
particle		1		3	7			0			
				5	2			4			
				6							

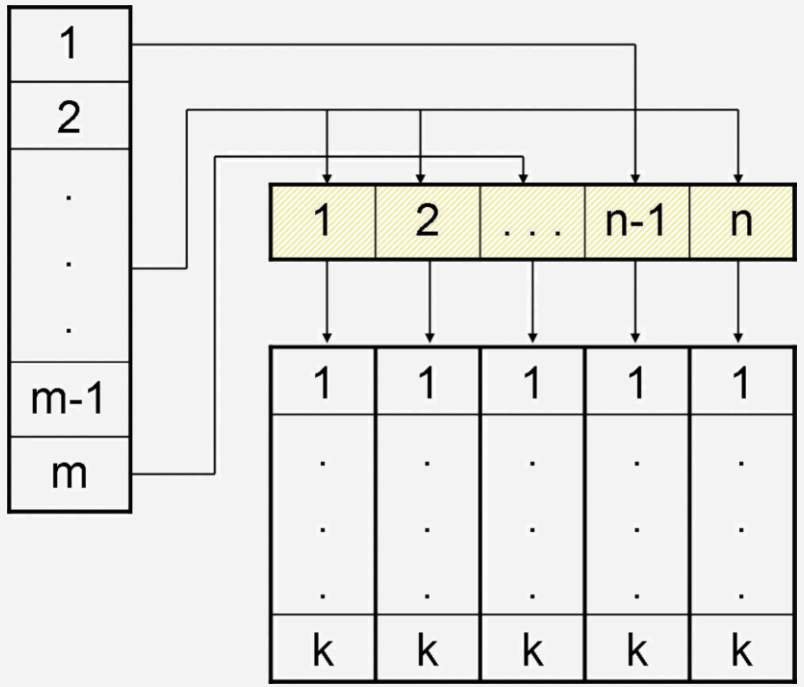
1D Hash map

Spatial Hashing

- Large hash tables reduce number of hash collisions
  - Different spatial cells with the same hash value cause hash collisions which slow down the query
- Reduced memory allocations
  - Memory for *m* entries is allocated for each hash cell
- Reduced cache-hit rate
  - Hash table is sparsely filled
  - Alternating filled and empty entries

Compact Hashing

- Hash cells store handles to a compact list of used cells
  - Elements in the used-cell list are generated, if a particle is placed in a new cell
  - Elements are deleted, if a cell gets empty
  - *k* entries are pre-allocated for each element in the list of used cells
- List of used cells is queried in the neighbor search



Compact Hashing - Construction

- Larger hash table compared to spatial hashing to reduce hash collisions
- Temporal coherence can be employed
  - List of used cells is not rebuilt, but updated
  - Particles with changed cell index are estimated
  - Particle is removed from the old cell and added to the new cell

Compact Hashing - Query

- Processing of used cells
  - Bad spatial locality
  - Used cells close in memory are not close in space
- Hash-collision flag
  - If there is no hash collision in a cell, hash indices of adjacent cells have to be computed only once for all particles in this cell

Compact Hashing - Query

- Particles are sorted with respect to a z-curve every n^{th} step
- After sorting, the list of used cells is rebuilt
- If particles are serially inserted into the list of used cells, the list is consistent with the z-curve
- Improved cache hit rate during the traversal of the list of used cells

Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
  - Uniform grid
  - Index sort
  - Spatial hashing
  - Discussion
- …

Comparison

Method	Construction	Query	Total
Basic grid	26	38	64
Index sort	36	29	65
Z-index sort	16	27	43
Spatial hashing	42	86	128
Compact hashing	8	32	40

- Measurements in ms for 130 k particles
- Ongoing research
  - Focus on sorting, parallelization and vectorization
  - Octrees, k-D trees, BVHs can also be realized with sorting

Summary

- Index sort
  - Fast construction based on sorting
  - Fast query
  - Particles are processed in the order of cell indices
- Z-index sort
  - Sorting with respect to a space filling curve improves cache-hit rate

Summary

- Spatial hashing
  - Less efficient query due to hash collisions and due to the traversal of the sparsely filled hash table
- Compact hashing
  - Fast construction (or update due to temporal coherence)
  - Fast query due to the compact list of used cells, due to the hash-collision flag and due to the z-curve

Simulation in Computer Graphics Particle Fluids 4

Matthias Teschner

UNI FREIBURG

Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
- Boundary handling
- Incompressibility

Simple SPH Fluid Solver

for all *particle* i **do**
  find neighbors j

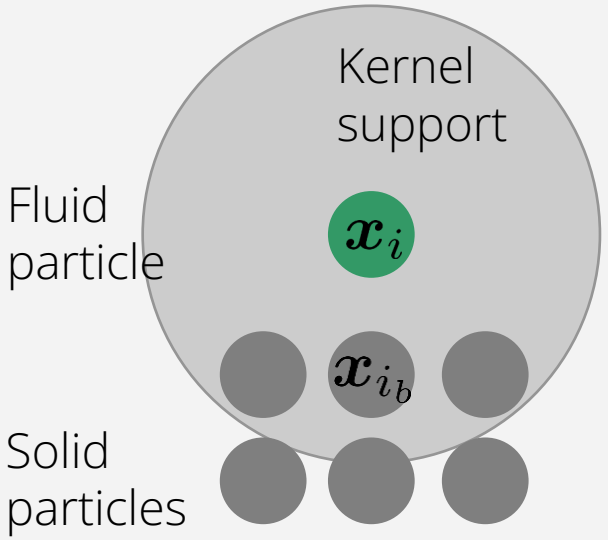
for all *particle* i **do**
  $\rho_i = \sum_j m_j W_{ij}$ Compute density
  $p_i = k(\frac{\rho_i}{\rho_0} - 1)$ Compute pressure

for all *particle* i **do**
  $\boldsymbol{a}_i^{\text{nonp}} = \nu \nabla^2 \boldsymbol{v}_i + \boldsymbol{g}$ Compute non-pressure accelerations
  $\boldsymbol{a}_i^{\text{p}} = -\frac{1}{\rho_i} \nabla p_i$ Compute pressure acceleration
  $\boldsymbol{a}_i(t) = \boldsymbol{a}_i^{\text{nonp}} + \boldsymbol{a}_i^{\text{p}}$

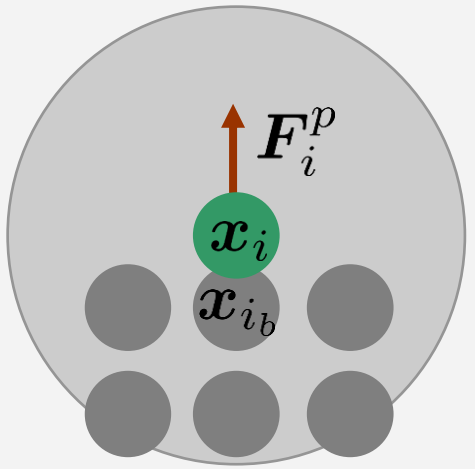
for all *particle* i **do**
  $\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i(t) + \Delta t \boldsymbol{a}_i(t)$
  $\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i(t + \Delta t)$

Concept

– Boundaries are sampled with particles that contribute to density, pressure and pressure acceleration of the fluid



$$\rho_i < \rho_0$$
$$p_i = 0$$
$$p_{i_b} = 0$$
$$\boldsymbol{F}_i^{\mathrm{p}} = \mathbf{0}$$

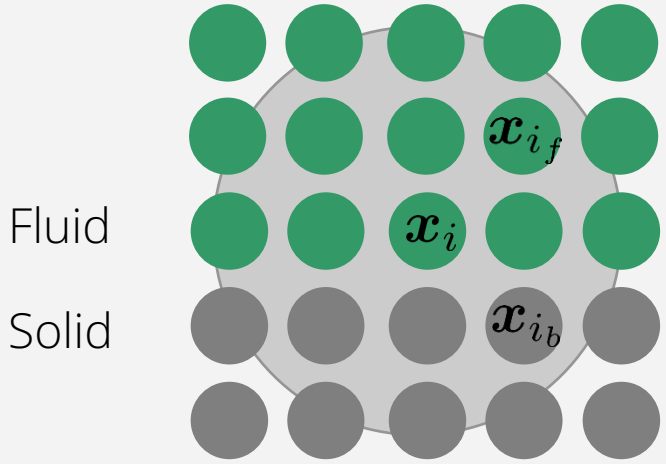


$$\rho_i > \rho_0$$
$$p_i > 0$$
$$p_{i_b} > 0$$
$$\boldsymbol{F}_i^{\mathrm{p}} \neq \mathbf{0}$$

– Boundary handling: How to compute $\rho_i, p_i, p_{i_b}, \boldsymbol{F}_i^{\mathrm{p}}$?

Several Layers with Uniform Boundary Samples

– Boundary particles are handled as static fluid samples



$$\rho_i = \sum_{i_f} m_{i_f} W_{ii_f} + \sum_{i_b} m_{i_b} W_{ii_b}$$

Boundary neighbors contribute to the density

$$m_i = m_{i_f} = m_{i_b}$$

All samples have the same size, i.e. same mass and rest density

$$\rho_i = m_i \sum_{i_f} W_{ii_f} + m_i \sum_{i_b} W_{ii_b}$$

$$p_i = k(\frac{\rho_i}{\rho_0} - 1)$$

– Pressure acceleration

$$\boldsymbol{a}_i^{\mathrm{p}} = -m_i \sum_{i_f} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2} \right) \nabla W_{ii_f} - m_i \sum_{i_b} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_b}}{\rho_{i_b}^2} \right) \nabla W_{ii_b}$$

All samples have the same size, i.e. same mass and rest density

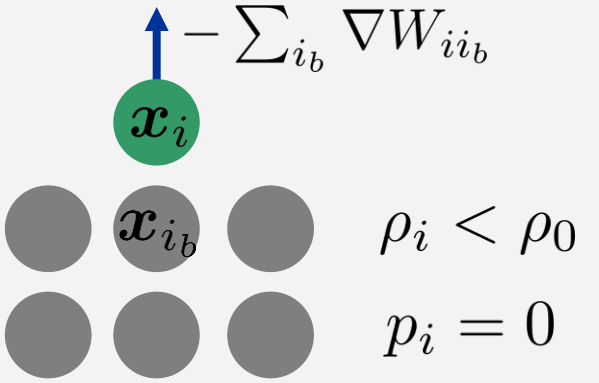
Contributions from fluid neighbors Contributions from boundary neighbors

Pressure at Boundary Samples

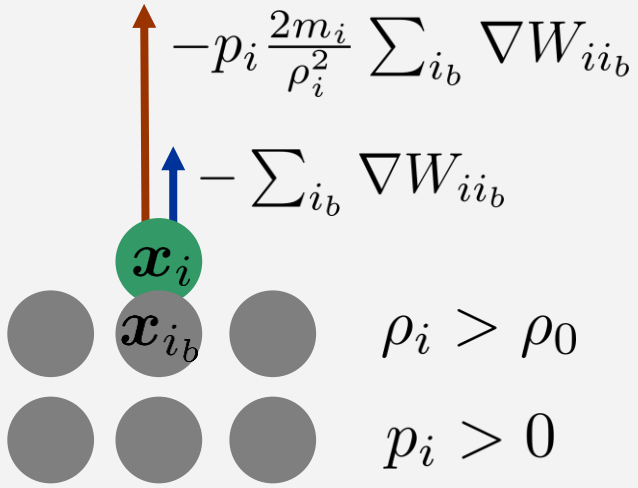
- Pressure acceleration at boundaries requires pressure at boundary samples
- Various solutions, e.g. mirroring, extrapolation, PPE
- Mirroring
  - Formulation with unknown boundary pressure p_{i_b}
  - $\boldsymbol{a}_i^{\mathrm{p}} = -m_i \sum_{i_f} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2} \right) \nabla W_{ii_f} - m_i \sum_{i_b} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_b}}{\rho_{i_b}^2} \right) \nabla W_{ii_b}$
  - Mirroring of pressure and density from fluid to boundary $p_{i_b} = p_i$
  - $\boldsymbol{a}_i^{\mathrm{p}} = -m_i \sum_{i_f} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2} \right) \nabla W_{ii_f} - m_i \sum_{i_b} \left(\frac{p_i}{\rho_i^2} + \frac{p_i}{\rho_i^2} \right) \nabla W_{ii_b}$

Boundary Contribution to Pressure Acceleration

$$\boldsymbol{a}_i^{\mathrm{p}} = - \ldots - m_i \sum_{i_b} \left(\frac{p_i}{\rho_i^2} + \frac{p_i}{\rho_i^2} \right) \nabla W_{ii_b} = - \ldots - p_i \frac{2m_i}{\rho_i^2} \sum_{i_b} \nabla W_{ii_b}$$



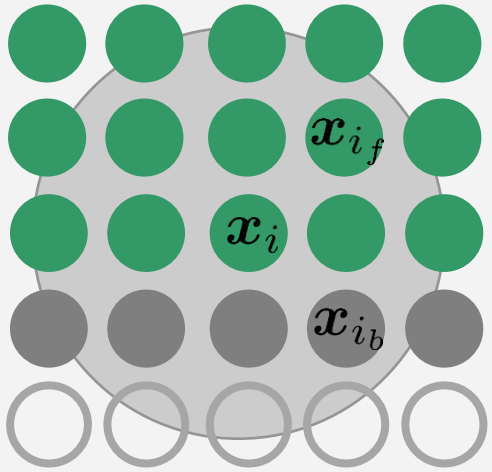
$$-p_i \frac{2m_i}{\rho_i^2} \sum_{i_b} \nabla W_{ii_b} = \mathbf{0}$$



$$-p_i \frac{2m_i}{\rho_i^2} \sum_{i_b} \nabla W_{ii_b} \neq \mathbf{0}$$

One Layer of Uniform Boundary Samples

– Contributions of missing samples have to be added



$$\rho_i = m_i \sum_{i_f} W_{ii_f} + m_i \sum_{i_b} W_{ii_b} + x$$

x is an approximation of the contribution from missing samples

$$\rho_i = m_i \sum_{i_f} W_{ii_f} + \gamma_1 m_i \sum_{i_b} W_{ii_b}$$

Offset typically implemented as scaling coefficient

$$\sum_{i_f} W_{ii_f} + \gamma_1 \sum_{i_b} W_{ii_b} = \frac{1}{V_i} \Rightarrow \gamma_1 = \frac{\frac{1}{V_i} - \sum_{i_f} W_{ii_f}}{\sum_{i_b} W_{ii_b}}$$

Kernel property

– Pressure acceleration

$$\boldsymbol{a}_i^{\mathrm{p}} = -m_i \sum_{i_f} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2} \right) \nabla W_{ii_f} - p_i \frac{2\gamma_2 m_i}{\rho_i^2} \sum_{i_b} \nabla W_{ii_b}$$

$$\sum_{i_f} \nabla W_{ii_f} + \gamma_2 \sum_{i_b} \nabla W_{ii_b} = \mathbf{0} \Rightarrow \gamma_2 = -\frac{\sum_{i_f} \nabla W_{ii_f} \cdot \sum_{i_b} \nabla W_{ii_b}}{\sum_{i_b} \nabla W_{ii_b} \cdot \sum_{i_b} \nabla W_{ii_b}}$$

Kernel gradient property
Pseudo inverse

Correction of Missing Contributions

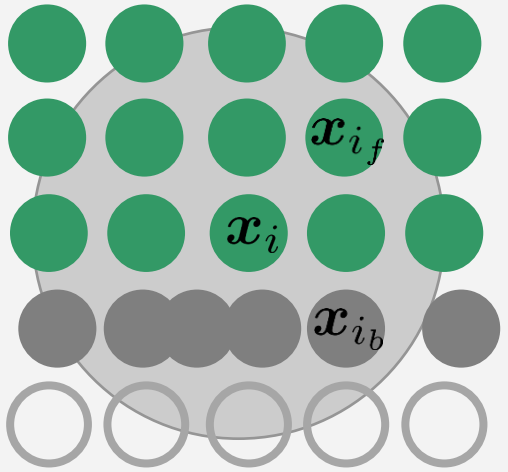
$$\rho_i = m_0(W_{00} + W_{01} + W_{02})$$
$$\boldsymbol{a}_i^{\mathrm{p}} = -p_i \frac{2m_i}{\rho_i^2}(\nabla W_{01} + \nabla W_{02})$$

$$\rho_i = \gamma_1 m_0(W_{00} + W_{01})$$
$$\boldsymbol{a}_i^{\mathrm{p}} = -p_i \frac{2\gamma_2 m_i}{\rho_i^2}\nabla W_{01}$$

– The motivation of γ_1 and γ_2 is to compensate contributions of missing samples to $\rho, p, \boldsymbol{a}^{\mathrm{p}}$

One Layer of Non-Uniform Boundary Samples

– Non-uniform contributions from boundary samples



$$\rho_i = m_i \sum_{i_f} W_{ii_f} + \sum_{i_b} m_{i_b} W_{ii_b}$$

$$V_{i_b}^0 = \frac{m_{i_b}}{\rho_0} = \frac{\gamma_1}{\sum_{i_{bb}} W_{i_b i_{bb}}}$$

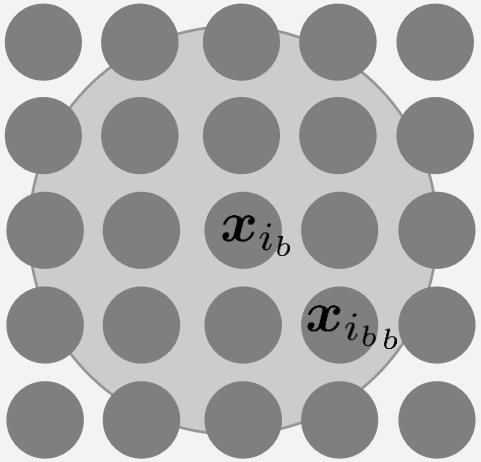
Non-uniform sizes, i.e. masses of boundary samples

Contribution, i.e. mass of a boundary sample is approximated from its boundary neighbors

– Pressure acceleration

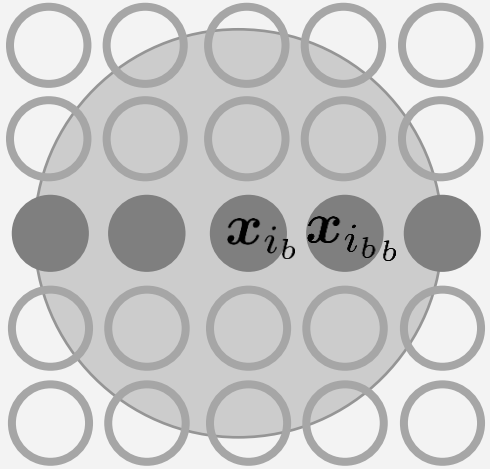
$$\boldsymbol{a}_i^{\mathrm{p}} = -m_i \sum_{i_f} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2} \right) \nabla W_{ii_f} - p_i \frac{2\gamma_2}{\rho_i^2} \sum_{i_b} m_{i_b} \nabla W_{ii_b}$$

One Layer of Non-Uniform Boundary Samples



$$V_{i_b}^0 = h^3 = \frac{1}{\sum_{i_{bb}} W_{i_b i_{bb}}}$$

For perfect sampling



$$V_{i_b}^0 = h^3 = \frac{\gamma_1}{\sum_{i_{bb}} W_{i_b i_{bb}}}$$

$$\Rightarrow \gamma_1 = h^3 \sum_{i_{bb}} W_{i_b i_{bb}}$$

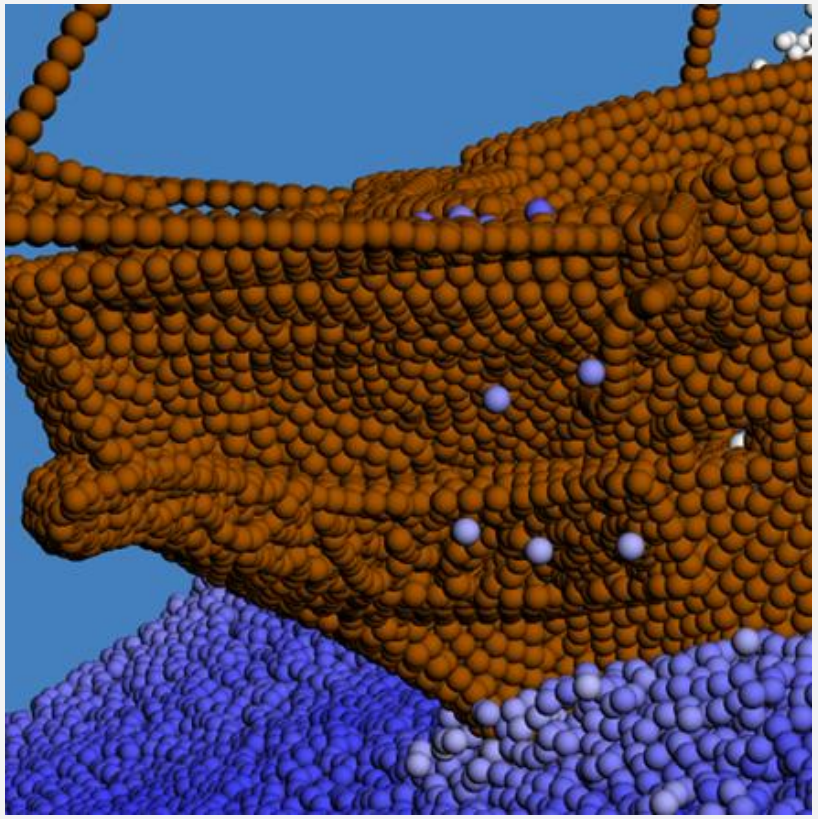
For perfect sampling

$$m_{i_b} = \rho_0 \frac{\gamma_1}{\sum_{i_{bb}} W_{i_b i_{bb}}}$$

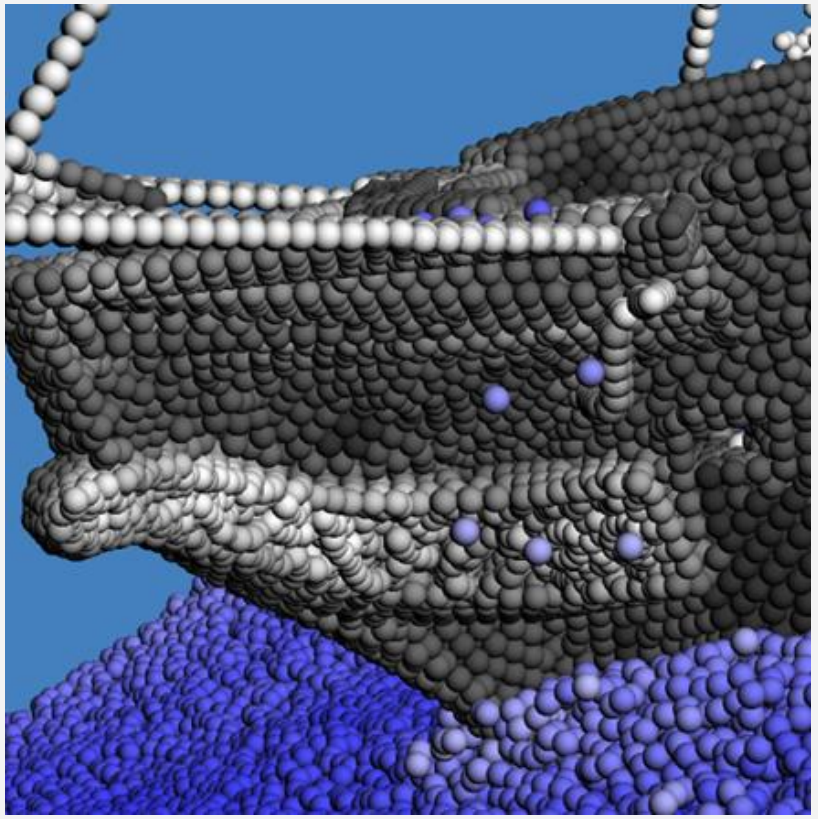
For arbitrary sampling

– In 3D, $\gamma_1 = 0.7$

Typical Boundary Representation

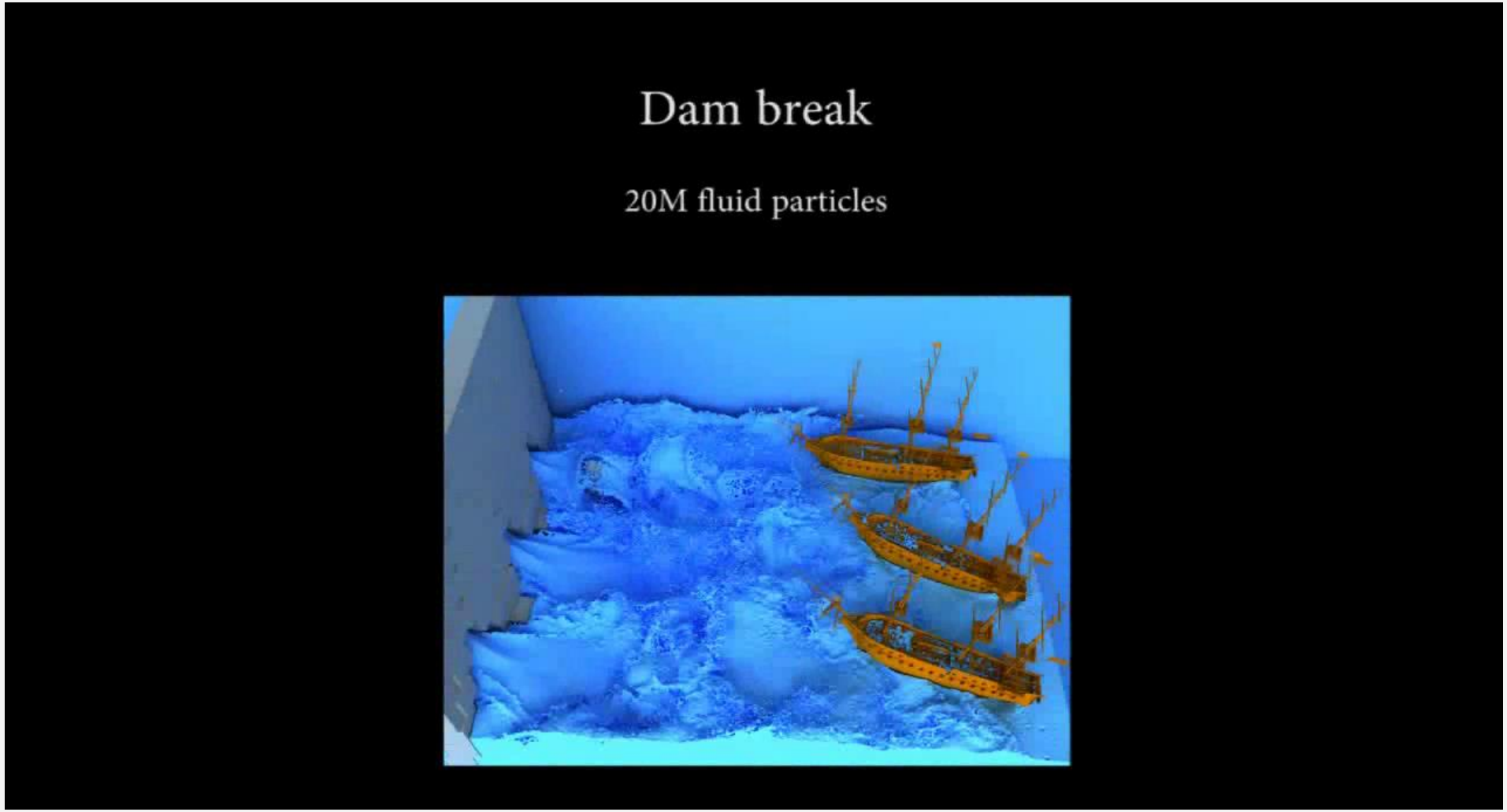


Boundary samples



Color-coded volume
of boundary samples

Rigid-Fluid Coupling



Rigid-Fluid Coupling

Summary

- Boundary is sampled with static fluid particles
- One layer of non-uniform samples
  - Arbitrary triangulated meshes can be used as boundary
  - Non-uniform boundary samples can be handled
  - Missing contributions to fluid density and pressure acceleration have to be corrected
  - Pressure is mirrored from fluid to boundary

Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
- Boundary handling
- Incompressibility

Incompressibility

- Is essential for a realistic fluid behavior
  - Less than 0.1% volume / density deviation in typical scenarios
- Inappropriate compression leads, e.g., to volume oscillations or volume loss
- Significant influence on the performance
  - Simple approaches require small time steps
  - Complex approaches work with large time steps

Approaches

- Minimization of density / volume errors
  - Measure difference of actual and desired density
  - Compute pressure and pressure accelerations that reduce density / volume deviations
- Minimization of velocity divergence
  - Measure the divergence of the velocity field
  - Compute pressure and pressure accelerations that reduce the divergence of the velocity field

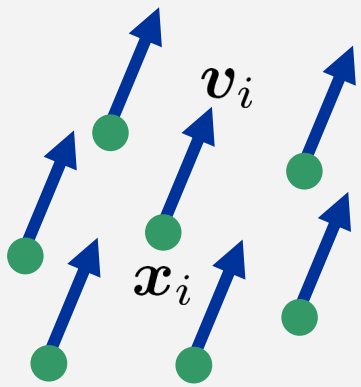
Approaches

- Velocity change per time step due to pressure acceleration and non-pressure acceleration
  $\frac{\mathrm{d}\boldsymbol{v}(t)}{\mathrm{d}t} = -\frac{1}{\rho}\nabla p(t) + \boldsymbol{a}^{\mathrm{nonp}}(t)$
- Predicted velocity after non-pressure acceleration
  $\boldsymbol{v}^* = \boldsymbol{v}(t) + \Delta t \boldsymbol{a}^{\mathrm{nonp}}(t)$
- Computation of pressure such that pressure acceleration either minimizes the divergence of $\boldsymbol{v}^*$ or the density error after advecting the samples with $\boldsymbol{v}^*$
- Final velocity $\boldsymbol{v}(t+\Delta t) = \boldsymbol{v}^* - \Delta t \frac{1}{\rho}\nabla p(t)$ with minimized divergence or minimized density error at advected samples

Density Invariance vs. Velocity Divergence

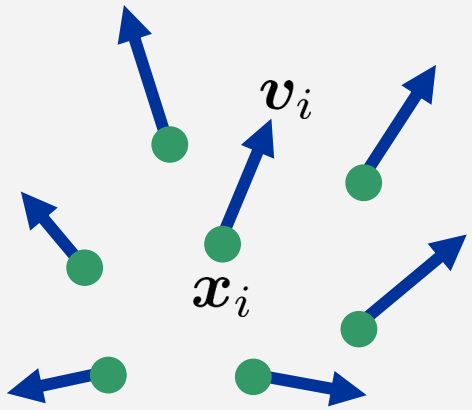
– Continuity equation: Time rate of change of the density is related to the divergence of the velocity

$$\frac{\mathrm{D}\rho_i}{\mathrm{D}t} = -\rho_i \nabla \cdot \boldsymbol{v}_i$$



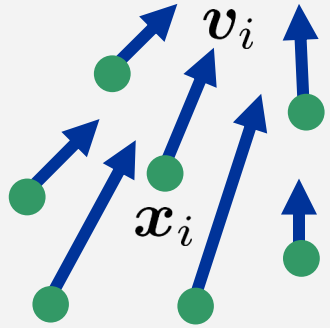
$$\frac{\mathrm{D}\rho_i}{\mathrm{D}t} = -\rho_i \nabla \cdot \boldsymbol{v}_i = 0$$

$$\nabla \cdot \boldsymbol{v}_i = 0$$



$$\frac{\mathrm{D}\rho_i}{\mathrm{D}t} = -\rho_i \nabla \cdot \boldsymbol{v}_i < 0$$

$$\nabla \cdot \boldsymbol{v}_i > 0$$



$$\frac{\mathrm{D}\rho_i}{\mathrm{D}t} = -\rho_i \nabla \cdot \boldsymbol{v}_i > 0$$

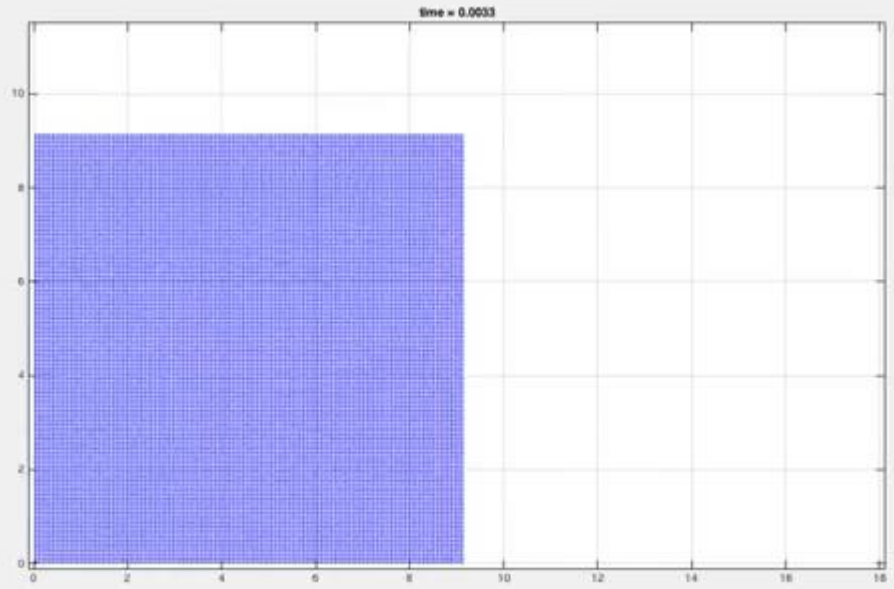
$$\nabla \cdot \boldsymbol{v}_i < 0$$

Density Invariance vs. Velocity Divergence

- Density invariance
  - Measure and minimize density deviations
- Velocity divergence
  - Measure and minimize the divergence of the velocity field
  - Zero velocity divergence corresponds to zero density change over time $-\rho_i \nabla \cdot \boldsymbol{v}_i = \frac{\mathrm{D}\rho_i}{\mathrm{D}t} = 0$, i.e. the initial density does not change over time
  - Notion of density is not required

Challenges

- Minimizing density deviations can result in volume oscillations
  - Density error is going up and down
  - Erroneous fluid dynamics
  - Only very small density deviations are tolerable, e.g. 0.1%



https://www.youtube.com/watch?v=hAPO0xBp5WU

Challenges

- Minimizing the velocity divergence can result in volume loss
  - Divergence errors result in density drift
  - No notion of actual density

Zhu, Lee, Quigley, Fedkiw, SIGGRAPH 2015

State Equations (EOS, SESPH)

- Pressure based on density deviations
- Pressure accelerations resolve compression induced by non-pressure accelerations
  - Density fluctuations / errors result in pressure
  - Pressure gradients result in pressure accelerations from high to low pressure to resolve density errors
- Simple computation
- Small time steps

State Equations (EOS, SESPH)

- Pressure is computed from density error
  - E.g. $p_i = k(\frac{\rho_i}{\rho_0} - 1)$ or $p_i = k(\rho_i - \rho_0)$
    - Referred to as compressible SPH
  - $p_i = k\left(\left(\frac{\rho_i}{\rho_0}\right)^7 - 1\right)$
    - Referred to as weakly compressible SPH
- Stiffness constant k does not govern the pressure, but the compressibility of the fluid
- Larger stiffness $\rightarrow$ less compressibility $\rightarrow$ smaller time step

Pressure values in SPH implementations should always be non-negative.

Simple SPH Fluid Solver

- **for all** *particle* i **do**
  find neighbors j

  for all *particle* i **do**
  $\rho_i = \sum_j m_j W_{ij}$
  $p_i = k(\frac{\rho_i}{\rho_0} - 1)$

  Compute pressure with a state equation

  for all *particle* i **do**
  $\boldsymbol{a}_i^{\text{nonp}} = \nu \nabla^2 \boldsymbol{v}_i + \boldsymbol{g}$
  $\boldsymbol{a}_i^{\text{p}} = -\frac{1}{\rho_i} \nabla p_i$
  $\boldsymbol{a}_i(t) = \boldsymbol{a}_i^{\text{nonp}} + \boldsymbol{a}_i^{\text{p}}$

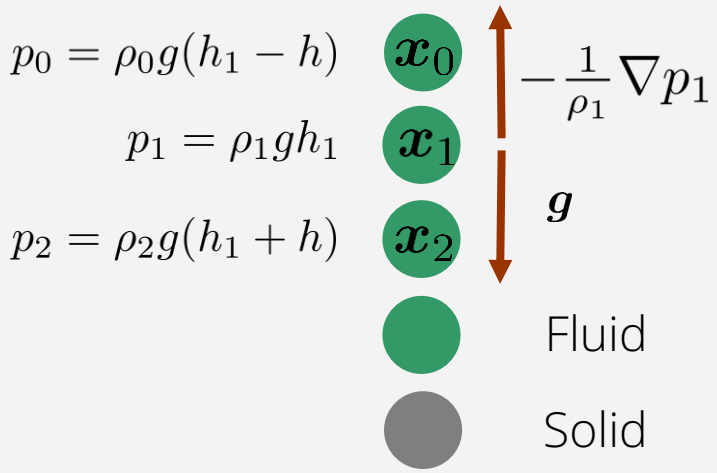
  for all *particle* i **do**
  $\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i(t) + \Delta t \boldsymbol{a}_i(t)$
  $\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i(t + \Delta t)$

Pressure - Illustration

- A fluid under gravity at rest
  - Gravity cancels pressure acceleration

$$\boldsymbol{g} = -\boldsymbol{a}_i^{\mathrm{p}} = \frac{1}{\rho_i}\nabla p_i = \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij}$$

$$= \sum_j m_j \left(\frac{k(\rho_i - \rho_0)}{\rho_i^2} + \frac{k(\rho_j - \rho_0)}{\rho_j^2}\right)\nabla W_{ij}$$



  - Differences between p_i and p_j are independent from k
  - Smaller k results in larger density error $\rho_i - \rho_0$ to get the correct pressure

SESPH with Splitting

- Split pressure and non-pressure accelerations
  - Non-pressure acceleration $\boldsymbol{a}_i^{\text{nonp}}$
  - Predicted velocity $\boldsymbol{v}_i^* = \boldsymbol{v}_i(t) + \Delta t \boldsymbol{a}_i^{\text{nonp}}$
  - Predicted position $\boldsymbol{x}_i^* = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i^*$
  - Predicted density $\rho_i^*(\boldsymbol{x}_i^*)$
  - Pressure p from predicted density ρ_i^*
  - Pressure acceleration $\boldsymbol{a}_i^{\text{p}}$
  - Final velocity and position $\boldsymbol{v}_i(t+\Delta t) = \boldsymbol{v}_i^* + \Delta t \boldsymbol{a}_i^{\text{p}} = \boldsymbol{v}_i^* - \Delta t \frac{1}{\rho_i^*} \nabla p_i$

    $\boldsymbol{x}_i(t+\Delta t) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i(t+\Delta t)$

SESPH with Splitting

- Motivation
  - Consider competing accelerations
  - Take effects of non-pressure accelerations into account when computing the pressure acceleration

Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
- Boundary handling
- Incompressibility

Simple SPH Fluid Solver

- **for all** *particle* i **do**
  find neighbors j

  for all *particle* i **do**
  $\rho_i = \sum_j m_j W_{ij}$
  $p_i = k(\frac{\rho_i}{\rho_0} - 1)$

  Compute pressure with a state equation

  for all *particle* i **do**
  $\boldsymbol{a}_i^{\text{nonp}} = \nu \nabla^2 \boldsymbol{v}_i + \boldsymbol{g}$
  $\boldsymbol{a}_i^{\text{p}} = -\frac{1}{\rho_i} \nabla p_i$
  $\boldsymbol{a}_i(t) = \boldsymbol{a}_i^{\text{nonp}} + \boldsymbol{a}_i^{\text{p}}$

  for all *particle* i **do**
  $\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i(t) + \Delta t \boldsymbol{a}_i(t)$
  $\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i(t + \Delta t)$

SESPH with Splitting

- **for all** *particle* i **do**
  find neighbors j

  for all *particle* i **do**
  $\boldsymbol{a}_i^{\text{nonp}} = \nu \nabla^2 \boldsymbol{v}_i + \boldsymbol{g}$
  $\boldsymbol{v}_i^* = \boldsymbol{v}_i(t) + \Delta t \boldsymbol{a}_i^{\text{nonp}}$

  for all *particle* i **do**
  $\rho_i^* = \sum_j m_j W_{ij} + \Delta t \sum_j m_j (\boldsymbol{v}_i^* - \boldsymbol{v}_j^*) \nabla W_{ij}$ Density at predicted positions
  $p_i = k(\frac{\rho_i^*}{\rho_0} - 1)$ Pressure at predicted positions

  for all *particle* i **do**
  $\boldsymbol{a}_i^{\text{p}} = -\frac{1}{\rho_i} \nabla p_i$

  for all *particle* i **do**
  $\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i^* + \Delta t \boldsymbol{a}_i^{\text{p}}$
  $\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i(t + \Delta t)$

Differential Density Update

- Density at advected positions is approximated without advecting the samples
- Continuity equation and time discretization

$$\frac{\mathrm{D}\rho_i}{\mathrm{D}t} = -\rho_i \nabla \cdot \boldsymbol{v}_i \qquad \frac{\rho_i^* - \rho_i(t)}{\Delta t} = -\rho_i \nabla \cdot \boldsymbol{v}_i^*$$

- Space discretization with SPH

$$\frac{\rho_i^* - \sum_j m_i W_{ij}}{\Delta t} = -\rho_i \left(-\frac{1}{\rho_i} \sum_j m_j (\boldsymbol{v}_i^* - \boldsymbol{v}_j^*) \nabla W_{ij} \right)$$

- Predicted density due to the divergence of $\boldsymbol{v}_i^*$

$$\rho_i^* = \sum_j m_i W_{ij} + \Delta t \sum_j m_j (\boldsymbol{v}_i^* - \boldsymbol{v}_j^*) \nabla W_{ij}$$

Approximate density at predicted positions: $\boldsymbol{x}_i^* = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i^*$

Iterative SESPH with Splitting

- Pressure accelerations are iteratively refined
  - Non-pressure acceleration $\boldsymbol{a}_i^{\text{nonp}}$
  - Predicted velocity $\boldsymbol{v}_i^* = \boldsymbol{v}_i(t) + \Delta t \boldsymbol{a}_i^{\text{nonp}}$
  - Iterate until convergence
    - Density from predicted position $\rho_i^*(\boldsymbol{x}_i, \boldsymbol{v}_i^*)$
    - Pressure from predicted density p_i
    - Pressure acceleration $\boldsymbol{a}_i^{\text{p}}$
    - Refine predicted velocity $\boldsymbol{v}_i^* = \boldsymbol{v}_i^* + \Delta t \boldsymbol{a}_i^{\text{p}}$
  - Final velocity and position $\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i^*$

$$\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i(t + \Delta t)$$

Iterative SESPH with Splitting

- Motivation
  - Iterative update is parameterized by a desired density error
  - Provides a fluid state with a guaranteed density error

Iterative SESPH with Splitting

- **for all** *particle* i **do**
  find neighbors j

  for all *particle* i **do**
  $\boldsymbol{a}_i^{\text{nonp}} = \nu \nabla^2 \boldsymbol{v}_i + \boldsymbol{g} \quad ; \quad \boldsymbol{v}_i^* = \boldsymbol{v}_i(t) + \Delta t \boldsymbol{a}_i^{\text{nonp}}$

  repeat

  for all *particle* i **do**
  $\rho_i^* = \sum_j m_j W_{ij} + \Delta t \sum_j m_j (\boldsymbol{v}_i^* - \boldsymbol{v}_j^*) \nabla W_{ij}$
  $p_i = k(\frac{\rho_i^*}{\rho_0} - 1)$

  for all *particle* i **do**
  $\boldsymbol{v}_i^* = \boldsymbol{v}_i^* - \Delta t \frac{1}{\rho_i^*} \nabla p_i$

  until $\rho_i^* - \rho_0 < \eta$ user-defined density error

  for all *particle* i **do**
  $\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i^* \quad ; \quad \boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i(t + \Delta t)$

Iterative SESPH - Variants

- Different quantities are accumulated
  - Velocity changes (local Poisson SPH)
  - Pressure (predictive-corrective SPH PCISPH)
    - Advantageous, if pressure is required for other computations
  - Distances (position-based fluids PBF)
    - $\Delta \boldsymbol{x}_i = -\frac{1}{\rho_0} \sum_j (\frac{p_i}{\beta_i} + \frac{p_j}{\beta_j}) \nabla W_{ij}$
- Different EOS and stiffness constants are used
  - $p_i = k(\rho_i - \rho_0)$ with $k = \frac{\rho_i^* r_i^2}{2\rho_0 \Delta t^2}$ in local Poisson SPH
  - $p_i = k(\rho_i - \rho_0)$ with $k = \frac{\rho_0^2}{2m_i^2 \cdot \Delta t^2 (\sum_j \nabla W_{ij}^0 \cdot \sum_j \nabla W_{ij}^0 + \sum_j (\nabla W_{ij}^0 \cdot \nabla W_{ij}^0))}$ in PCISPH
  - $p_i = k(\frac{\rho_i}{\rho_0} - 1)$ with $k = 1$ in PBF

Predictive-Corrective Incompressible SPH - PCISPH

- Goal: Computation of pressure accelerations $\boldsymbol{a}_i^{\mathrm{p}}$ that result in rest density ρ_0 at all particles
- Formulation: Density at the next step should equal the rest density

Desired density | Current density | Density change due to predicted velocity | Density change due to unknown pressure acceleration

$$\rho_i(t+\Delta t) = \rho_0 = \underbrace{\sum_j m_i W_{ij} + \Delta t \sum_j m_j (\boldsymbol{v}_i^* - \boldsymbol{v}_j^*) \nabla W_{ij}}_{\rho_i^*} + \Delta t \sum_j m_j (\Delta t \boldsymbol{a}_i^{\mathrm{p}} - \Delta t \boldsymbol{a}_j^{\mathrm{p}}) \nabla W_{ij}$$

Discretized continuity equation

PCISPH - Assumptions

- Simplifications to get one equation with one unknown:
  - Equal pressure at all neighboring samples

$$\boldsymbol{a}_i^{\mathrm{p}} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij} \approx -m_i \frac{2p_i}{\rho_0^2} \sum_j \nabla W_{ij}$$

$$\rho_0 - \rho_i^* = \Delta t^2 \sum_j m_j \left(-m_i \frac{2p_i}{\rho_0^2} \sum_j \nabla W_{ij} + m_j \frac{2p_j}{\rho_0^2} \sum_k \nabla W_{jk} \right) \nabla W_{ij}$$

Unknown pressures p_i and p_j

  - For sample *j*, only consider the contribution from *i*

$$\rho_0 - \rho_i^* = \Delta t^2 \sum_j m_j \left(-m_i \frac{2p_i}{\rho_0^2} \sum_j \nabla W_{ij} + m_i \frac{2p_i}{\rho_0^2} \nabla W_{ji} \right) \nabla W_{ij}$$

Unknown pressure p_i

$$\rho_0 - \rho_i^* = \Delta t^2 m_i^2 \frac{2p_i}{\rho_0^2} \sum_j \left(-\sum_j \nabla W_{ij} - \nabla W_{ij} \right) \nabla W_{ij} = -\Delta t^2 m_i^2 \frac{2p_i}{\rho_0^2} \left(\sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} + \sum_j (\nabla W_{ij} \cdot \nabla W_{ij}) \right)$$

PCISPH - Solution

– Solve for unknown pressure:

$$\rho_0 - \rho_i^* = -\Delta t^2 m_i^2 \frac{2p_i}{\rho_0^2} \left(\sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} + \sum_j (\nabla W_{ij} \cdot \nabla W_{ij}) \right)$$

$$p_i = \frac{\rho_0^2}{2\Delta t^2 m_i^2 (\sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} + \sum_j (\nabla W_{ij} \cdot \nabla W_{ij}))} (\rho_i^* - \rho_0) \qquad (\; p_i = k(\rho_i^* - \rho_0) \;)$$

Intuition: This pressure causes pressure accelerations that cause velocity changes that correspond to a divergence that results in rest density at the sample.

$$\rho(t + \Delta t) = \rho_0 = \rho_i^* + \Delta t \sum_j m_j (\Delta t \boldsymbol{a}_i^{\mathrm{p}} - \Delta t \boldsymbol{a}_j^{\mathrm{p}}) \nabla W_{ij}$$

PCISPH - Discussion

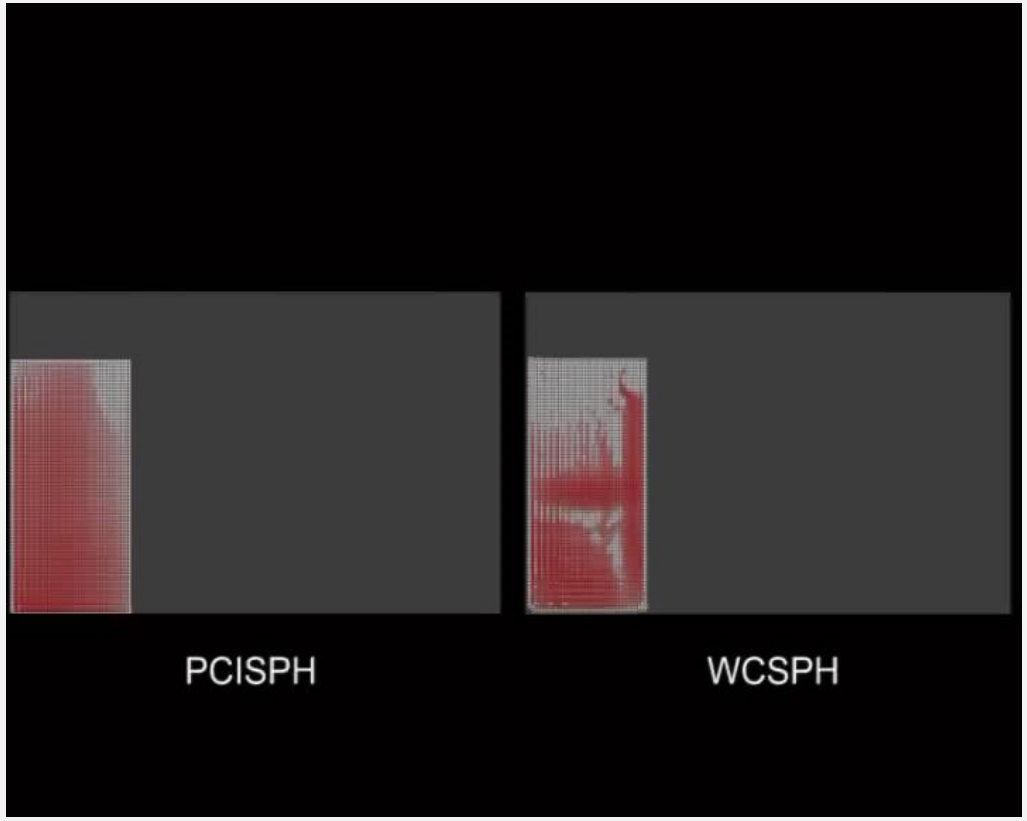
- Pressure is computed with a state equation $p_i = k(\rho_i^* - \rho_0)$
- k is not user-defined
- Instead, an optimized value k is derived and used
- Pressure is iteratively refined

PCISPH - Performance

- Typically three to five iterations for density errors between 0.1% and 1%
- Speed-up factor over non-iterative SESPH up to 50
  - More computations per time step compared to SESPH
  - Significantly larger time step than in SESPH
  - Speed-up dependent on scenario
- Non-linear relation between time step and iterations
  - Largest possible time step does not necessarily lead to an optimal overall performance

Comparison

- PCISPH [Solenthaler 2009]
  - Iterative pressure computation
  - Large time step
- WCSPH [Becker and Teschner 2007]
  - Efficient to compute
  - Small time step
- Computation time for the PCISPH scenario is 20 times shorter than WCSPH



Projection Schemes - Introduction

- Pressure causes pressure accelerations that cause velocity changes that cause displacements such that particles have rest density
- Projection schemes solve a linear system to compute the respective pressure field
  - PCISPH uses simplifications to compute pressure per particle from one equation. Solving a linear system is avoided.

Projection Schemes - Derivation

$$\frac{\mathrm{d}\boldsymbol{v}(t)}{\mathrm{d}t} = -\frac{1}{\rho}\nabla p(t) + \boldsymbol{a}^{\mathrm{nonp}}(t)$$

Velocity change per time step due to pressure acceleration and non-pressure acceleration

$$\boldsymbol{v}^* = \boldsymbol{v}(t) + \Delta t \boldsymbol{a}^{\mathrm{nonp}}(t)$$

Predicted velocity after non-pressure acceleration

$$\boldsymbol{v}(t+\Delta t) = \boldsymbol{v}^* - \Delta t \frac{1}{\rho}\nabla p(t)$$

Velocity after all accelerations

$$\boldsymbol{v}(t+\Delta t) - \boldsymbol{v}^* = -\Delta t \frac{1}{\rho}\nabla p(t)$$

Velocity change due to pressure acceleration

$$\nabla \cdot (\boldsymbol{v}^* - \boldsymbol{v}(t+\Delta t)) = \nabla \cdot \left(\Delta t \frac{1}{\rho}\nabla p(t)\right)$$

Divergence of the velocity change due to pressure acceleration

Projection Schemes - Derivation

$$\nabla \cdot (\boldsymbol{v}^* - \boldsymbol{v}(t + \Delta t)) = \nabla \cdot \left(\Delta t \frac{1}{\rho} \nabla p(t) \right)$$

$$\nabla \cdot \boldsymbol{v}^* - \nabla \cdot \boldsymbol{v}(t + \Delta t) = \nabla \cdot \left(\Delta t \frac{1}{\rho} \nabla p(t) \right)$$

Constraint: $\nabla \cdot \boldsymbol{v}(t + \Delta t) = 0$

Divergence of the final velocity field should be zero, i.e. no density change per time

$$\nabla \cdot \boldsymbol{v}^* = -\nabla \cdot (\Delta t \boldsymbol{a}^{\mathrm{p}})$$

Divergence of the velocity change due to pressure acceleration should cancel the divergence of the predicted velocity

$$\rho \nabla \cdot \boldsymbol{v}^* = \Delta t \nabla^2 p(t)$$

Pressure Poisson equation with unknown pressure

Density Invariance vs. Velocity Divergence

- Pressure Poisson equation PPE that minimizes the velocity divergence: $\Delta t \nabla^2 p(t) = \rho \nabla \cdot \boldsymbol{v}^*$
- PPE that minimizes the density error: $\Delta t \nabla^2 p(t) = \frac{\rho_0 - \rho^*}{\Delta t}$
  - Derivation:

$$\frac{\mathrm{D}\rho(t+\Delta t)}{\mathrm{D}t} + \rho(t+\Delta t)\nabla \cdot \boldsymbol{v}(t+\Delta t) = 0$$

Continuity equation at time $t + \Delta t$

Constraint: $\rho(t+\Delta t) = \rho_0$

$$\frac{\rho_0 - \rho(t)}{\Delta t} + \rho_0 \nabla \cdot \left(\boldsymbol{v}^* - \Delta t \frac{1}{\rho_0} \nabla p(t) \right) = 0$$

$$\frac{\rho_0 - (\rho(t) - \Delta t \rho_0 \nabla \cdot \boldsymbol{v}^*)}{\Delta t} - \Delta t \nabla^2 p(t) = 0$$

$$\rho^* = \rho(t) - \Delta t \rho_0 \nabla \cdot \boldsymbol{v}^*$$

Predicted density after sample advection with $\boldsymbol{v}^*$

PPE Forms - Interpretation

- Velocity divergence: $-\Delta t \frac{1}{\rho} \nabla^2 p = -\nabla \cdot \boldsymbol{v}^*$
  - Pressure p causes a pressure acceleration $-\frac{1}{\rho}\nabla p$ that causes a velocity change $-\Delta t \frac{1}{\rho}\nabla p$ whose divergence $\nabla \cdot (-\Delta t \frac{1}{\rho}\nabla p)$ cancels the divergence $\nabla \cdot \boldsymbol{v}^*$ of the predicted velocity, i.e. $\nabla \cdot \boldsymbol{v}^* + \nabla \cdot (-\Delta t \frac{1}{\rho}\nabla p) = 0$
- Density invariance: $-\Delta t \nabla^2 p = -\frac{\rho_0 - \rho^*}{\Delta t}$
  - The divergence $\nabla \cdot (-\Delta t \frac{1}{\rho}\nabla p)$ multiplied with density ρ is a density change per time that cancels the predicted density error per time $\frac{\rho_0 - \rho^*}{\Delta t}$, i.e. $\frac{\rho_0 - \rho^*}{\Delta t} + \rho \nabla \cdot (-\Delta t \frac{1}{\rho}\nabla^2 p) = 0$

PPE Solver

- Linear system with unknown pressure values $\boldsymbol{Ap} = \boldsymbol{s}$
  - One equation per particle $(\boldsymbol{Ap})_i = s_i \quad (\Delta t < \nabla^2 p_i >= \frac{\rho_0 - <\rho_i^*>}{\Delta t})$
- Iterative solvers
  - Conjugate Gradient
  - Relaxed Jacobi
- Fast computation per iteration
  - Few non-zero entries in each equation
  - Matrix-free implementations
  - Very few information per particle

<A> is a discretized form of *A*

PPE Solver

- Very large time steps
- Convergence dependent on the formulation
  - SPH discretization of $\nabla^2 p$
  - Source term (velocity divergence or density invariance)
- Accuracy issues
  - Volume drift for velocity divergence
  - Oscillations for density invariance

PPE Discretization

- Implicit incompressible SPH (IISPH) [Ihmsen et al. 2014]
  - PPE with density invariance as source term: $\Delta t^2 \nabla^2 p = \rho_0 - \rho^*$
  - Computation of ρ_i^*:
    $\rho_i^* = \rho_i + \Delta t \sum_j m_j \boldsymbol{v}_{ij}^* \nabla W_{ij}$ with $\boldsymbol{v}_i^* = \boldsymbol{v}_i + \Delta t \boldsymbol{a}_i^{\text{nonp}}$
  - Computation of $\Delta t^2 \nabla^2 p_i$:
    $\Delta t^2 \nabla^2 p_i = -\Delta t \rho_i \nabla \cdot (\Delta t \boldsymbol{a}_i^{\text{p}}) = \Delta t^2 \sum_j m_j \left(\boldsymbol{a}_i^{\text{p}} - \boldsymbol{a}_j^{\text{p}}\right) \cdot \nabla W_{ij}$
    with
    $\boldsymbol{a}_i^{\text{p}} = -\frac{1}{\rho_i} \nabla p_i = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}$

PPE System

- PPE $$\Delta t^2 \nabla^2 p_i = \rho_0 - \rho_i^*$$

  density change due to pressure accelerations — negative of the predicted density error

- Discretized PPE
  - System: $\boldsymbol{A}\boldsymbol{p} = \boldsymbol{s}$
  - Per particle: $\underbrace{\Delta t^2 \sum_j m_j \left(\boldsymbol{a}_i^{\mathrm{p}} - \boldsymbol{a}_j^{\mathrm{p}}\right) \nabla W_{ij}}_{(\boldsymbol{A}\boldsymbol{p})_i} = \underbrace{\rho_0 - \rho_i^*}_{s_i} \quad \boldsymbol{a}_i^{\mathrm{p}} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}$
  - Interpretation:

$$\Delta t \sum_j m_j \left(\Delta t \boldsymbol{a}_i^{\mathrm{p}} - \Delta t \boldsymbol{a}_j^{\mathrm{p}}\right) \nabla W_{ij} = \rho_0 - \rho_i^*$$
$$\Delta t \sum_j m_j \left(\boldsymbol{v}_i^{\mathrm{p}} - \boldsymbol{v}_j^{\mathrm{p}}\right) \nabla W_{ij} = \rho_0 - \rho_i^*$$
$$\Delta t \cdot \rho_i \cdot \nabla \cdot \boldsymbol{v}_i^{\mathrm{p}} = \rho_0 - \rho_i^*$$

Pressure accelerations causes a velocity change $\mathbf{v}^{\mathrm{p}}$ whose divergence causes a density change.

PPE Solver

- Relaxed Jacobi: $p_i^{l+1} = \max\left(p_i^l + \omega \frac{s_i - (\boldsymbol{A}\boldsymbol{p}^l)_i}{a_{ii}}, 0\right)$
  - For IISPH, typically $\omega = 0.5$
  - Diagonal element a_{ii}
    - Accumulate all coefficients of p_i in $\Delta t^2 \sum_j m_j \left(\boldsymbol{a}_i^p - \boldsymbol{a}_j^p\right) \nabla W_{ij}$
    - $a_{ii} = \Delta t^2 \sum_j m_j \left(-\sum_j \frac{m_j}{\rho_j^2} \nabla W_{ij}\right) \cdot \nabla W_{ij} + \Delta t^2 \sum_j m_j \left(\frac{m_i}{\rho_i^2} \nabla W_{ji}\right) \cdot \nabla W_{ij}$

PPE Solver - Implementation

- Initialization:

$$\rho_i = \sum_j m_j W_{ij} \quad a_{ii} = \dots$$
$$\boldsymbol{v}_i^* = \boldsymbol{v}_i + \Delta t \boldsymbol{a}_i^{\text{nonp}}$$
$$s_i = \rho_0 - \rho_i - \Delta t \sum_j m_j \boldsymbol{v}_{ij}^* \nabla W_{ij}$$
$$p_i^0 = \max\left(\omega \frac{s_i}{a_{ii}}, 0\right)$$

- Solver update in iteration l:
  - First loop:

$$(\boldsymbol{a}_i^{\text{p}})^l = -\sum_j m_j \left(\frac{p_i^l}{\rho_i^2} + \frac{p_j^l}{\rho_j^2}\right) \nabla W_{ij}$$

  - Second loop:

$$(\boldsymbol{A}\boldsymbol{p}^l)_i = \Delta t^2 \sum_j m_j \left((\boldsymbol{a}_i^{\text{p}})^l - (\boldsymbol{a}_j^{\text{p}})^l\right) \nabla W_{ij}$$

$$p_i^{l+1} = \max\left(p_i^l + \omega \frac{s_i - (\boldsymbol{A}\boldsymbol{p}^l)_i}{a_{ii}}, 0\right)$$

If a_{ii} not equal to zero

$$(\rho_i^{\text{error}})^l = (\boldsymbol{A}\boldsymbol{p}^l)_i - s_i$$

Continue until error is small

IISPH with Boundary Handling

- PPE: $\Delta t^2 \nabla^2 p_f = \rho_0 - \rho_f^* = \rho_0 - \rho_f + \Delta t \rho_0 \nabla \cdot \boldsymbol{v}_f^*$
- Discretized PPE: $\boldsymbol{A}\boldsymbol{p} = \boldsymbol{s}$

Index *f* indicates a fluid sample.
Index *b* indicates a boundary sample.
f_f indicates a fluid neighbor of *f*.
f_b indicates a boundary neighbor of *f*.

$$(\boldsymbol{A}\boldsymbol{p})_f = \Delta t^2 \sum_{f_f} m_{f_f} \left(\boldsymbol{a}_f^{\mathrm{p}} - \boldsymbol{a}_{f_f}^{\mathrm{p}} \right) \nabla W_{f f_f} + \Delta t^2 \sum_{f_b} m_{f_b} \boldsymbol{a}_f^{\mathrm{p}} \nabla W_{f f_b}$$

$$\boldsymbol{a}_f^{\mathrm{p}} = -\sum_{f_f} m_{f_f} \left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2} \right) \nabla W_{f f_f} - \gamma \sum_{f_b} m_{f_b} 2 \frac{p_f}{\rho_f^2} \nabla W_{f f_b}$$

$$s_f = \rho_0 - \rho_f - \Delta t \sum_{f_f} m_{f_f} \left(\boldsymbol{v}_f^* - \boldsymbol{v}_{f_f}^* \right) \nabla W_{f f_f} - \Delta t \sum_{f_b} m_{f_b} \left(\boldsymbol{v}_f^* - \boldsymbol{v}_{f_b}(t + \Delta t) \right) \nabla W_{f f_b}$$

IISPH with Boundary Handling

– Diagonal element

$$
\begin{aligned}
a_{ff} =& \Delta t^2 \sum_{f_f} m_{f_f} \left(- \sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}^2} \nabla W_{ff_f} - 2\gamma \sum_{f_b} \frac{m_{f_b}}{\rho_0^2} \nabla W_{ff_b} \right) \nabla W_{ff_f} \\
&+ \Delta t^2 \sum_{f_f} m_{f_f} \left(\frac{m_f}{\rho_f^2} \nabla W_{f_f f} \right) \nabla W_{ff_f} \\
&+ \Delta t^2 \sum_{f_b} m_{f_b} \left(- \sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}^2} \nabla W_{ff_f} - 2\gamma \sum_{f_b} \frac{m_{f_b}}{\rho_0^2} \nabla W_{ff_b} \right) \nabla W_{ff_b}
\end{aligned}
$$

IISPH with Boundary - Implementation

- Initialization:

$$\rho_f = \sum_{f_f} m_{f_f} W_{ff_f} + \sum_{f_b} m_{f_b} W_{ff_b} \qquad a_{ff} = \dots$$
$$\boldsymbol{v}_f^* = \boldsymbol{v}_f + \Delta t \boldsymbol{a}_f^{\text{nonp}}$$
$$s_f = \rho_0 - \rho_f - \Delta t \sum_{f_f} m_{f_f} \boldsymbol{v}_{ff_f}^* \nabla W_{ff_f} - \Delta t \sum_{f_b} m_{f_b} \boldsymbol{v}_{ff_b}^* \nabla W_{ff_b}$$
$$p_f^0 = \max\left(\omega \frac{s_f}{a_{ff}}, 0\right)$$

- Solver update in iteration *l*:
  - First loop:

$$(\boldsymbol{a}_f^{\text{p}})^l = -\sum_{f_f} m_{f_f} \left(\frac{p_f^l}{\rho_f^2} + \frac{p_{f_f}^l}{\rho_{f_f}^2}\right) \nabla W_{ff_f} - \gamma \sum_{f_b} m_{f_b} 2 \frac{p_f^l}{\rho_f^2} \nabla W_{ff_b}$$

  - Second loop:

$$(\boldsymbol{A}\boldsymbol{p}^l)_f = \Delta t^2 \sum_{f_f} m_{f_f} \left((\boldsymbol{a}_f^{\text{p}})^l - (\boldsymbol{a}_{f_f}^{\text{p}})^l\right) \nabla W_{ff_f} + \Delta t^2 \sum_{f_b} m_{f_b} (\boldsymbol{a}_f^{\text{p}})^l \nabla W_{ff_b}$$

$$p_f^{l+1} = \max\left(p_f^l + \omega \frac{s_f - (\boldsymbol{A}\boldsymbol{p}^l)_f}{a_{ff}}, 0\right)$$ If a_{ff} not equal to zero

$$(\rho_f^{\text{error}})^l = (\boldsymbol{A}\boldsymbol{p}^l)_f - s_f$$ Continue until error is small

PPE Solver - Comparison with PCISPH

- Breaking dam
  - 100k particles with diameter 0.05m
  - 0.01% average density error

	PCISPH			IISPH			PCISPH / IISPH		
		total comp. time [s]			total comp. time [s]		ratio		
Δt [s]	avg. iter.	pressure	overall	avg. iter.	pressure	overall	iterations	pressure	overall
0.0005	4.3	540	1195	2.2	148	978	2.0	3.6	1.2
0.00067	7.2	647	**1145**	2.9	149	753	2.5	4.3	1.5
0.001	14.9	856	1187	4.9	164	576	3.0	5.2	2.1
0.0025	66.5	1495	1540	18.4	242	410	3.6	**6.2**	3.8
0.004	-	-	-	33.5	273	**379**	-	-	-
0.005	-	-	-	45.8	297	383	-	-	-

- Largest possible time step does not necessarily result in the best performance

Simulation in Computer Graphics
Particle Fluids - Misc

Matthias Teschner

Parallel Scaling