Smoothed Particle Hydrodynamics

Techniques for the Physics Based Simulation of Fluids and Solids

Part I

Introduction, Foundations, Neighborhood Search

Dan Koschier — UCL

Jan Bender — RWTH

Barbara Solenthaler — ETH zürich

Matthias Teschner — UNI FREIBURG

Speakers

Jan Bender

Barbara Solenthaler

Matthias Teschner

Dan Koschier

Dan Koschier

- Post-doctoral Researcher at UCL
  - Smart Geometry Processing Group lead by Niloy Mitra
- Research interests include
  - Physics-based simulation (deformable solids, fluids)
  - Modelling of interface phenomena
  - Cutting and fracture in solids
  - Machine learning enhanced simulation
- One of main contributors to SPlisHSPlasH
  - Developer of supporting libraries
    - CompactNSearch (compact hashing-based neighborhood search algorithm)
    - Discregrid (Parallel higher-order discretization on regular/adaptive grids

Jan Bender

- Professor at RWTH Aachen University
  - Head of computer animation group

- Research interests include
  - Physics-based simulation (rigid bodies, solids, fluids, ...)
  - Collision handling
  - Cutting and Fracture
  - Real-time visualization

- Founder and maintainer of SPlisHSPlasH
  - Open source project
  - C++ implementation of many! modern SPH-based simulation techniques
  - Supports fluids, deformables, and coupling with rigid bodies

Barbara Solenthaler

- Senior Research Scientist at ETH Zürich
  - Head of simulation and animation group

- Research interests include
  - Physics-based simulation
  - Artist-controllable techniques
  - Data-driven simulation

- Co-founder of Apagom AG
  - Commercial engine
  - Real-time fluid simulation

PHYSICS FORESTS
using MachineLearning

- More than 10 years of research on SPH-related topics

Matthias Teschner

- Professor at University of Freiburg
  - Head of computer graphics group

- Research interests include
  - Physics-based simulation (rigid bodies, solids, fluids, ...)
  - Rendering
  - Computational geometry
  - Cutting edge technology for fluid simulations in
    - Engineering, entertainment, art, medicine, and robotics

- Co-founder of spin-off FIFTY2 technologies
  - Concerned with development of commercial SPH solvers

- More than 10 years of research on SPH-related topics

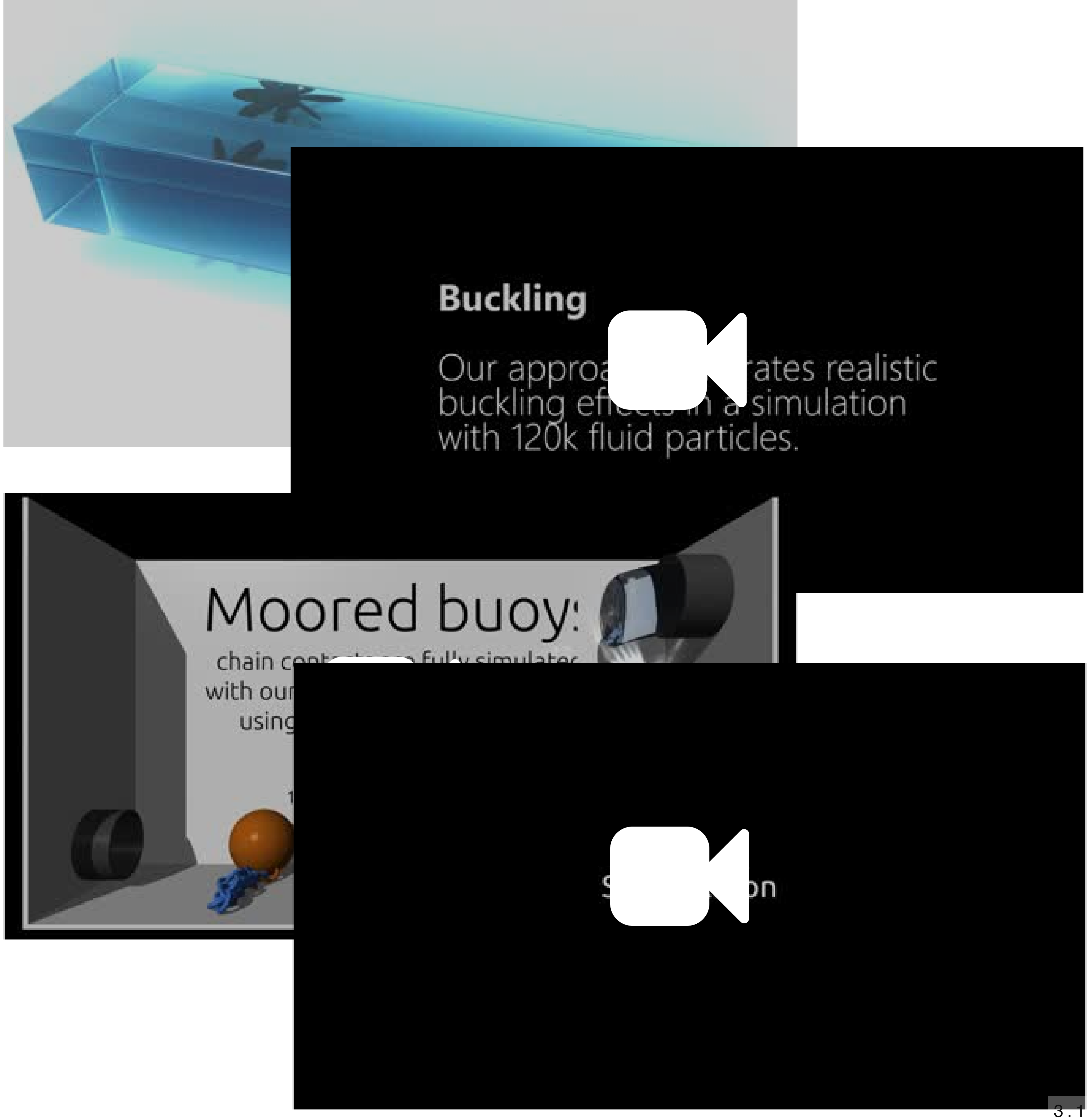
Main Goals of this Tutorial

- Explain the basic concept of SPH and its features
  - What is SPH? For what can it be used?
  - What is not mentioned in papers?
  - What are its strengths and weaknesses?
  - What can be done/has been done in simulation?

- Showcase
  - Generality and "beauty" of the concept
  - Potential
  - Wide range of applications (with examples)

- Make state-of-the-art methods tractable
  - Methods sound often more complex than they actually are

- <u>Motivate other researchers to "work on"/"use" SPH</u>



What can/can't be expected

CAN

- Practical introduction to SPH
- Brief introduction fluid/solid sim.
- Methods to realize physical phenomena
  - Modular/"as building blocks"
  - Videos/demos
- Clarification of "urban myths"
- Discussion of act. (C++) implementation
- (Some) state-of-the-art approaches
  - concepts, current challenges, ongoing trends

CAN'T

- Rigorous derivation of SPH concept
- Complete introduction to continuum mechanics
- Detailed discussion of results limitations of each approach
- Lecture on software architecture

"Urban myths"

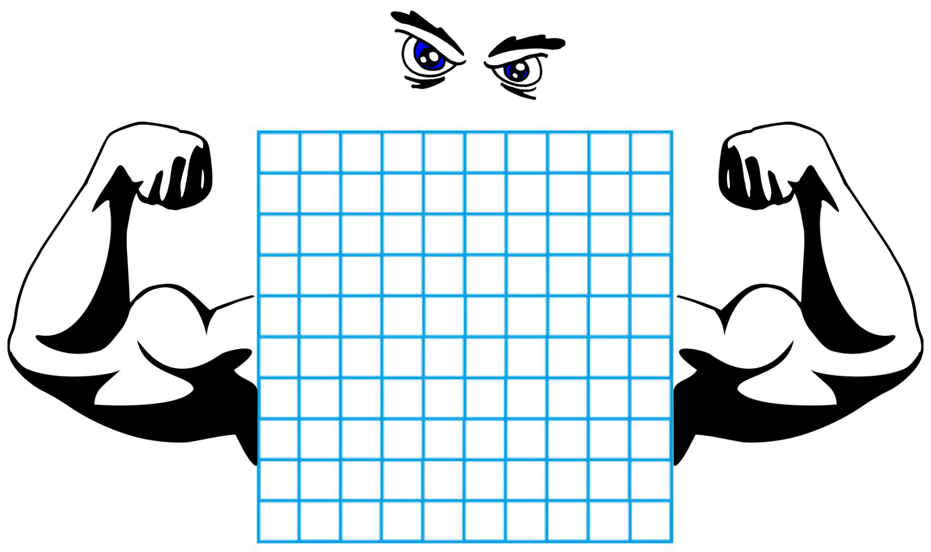
- An SPH particle represents
  - a physical atom/molecule
  - a droplet
  - a grain (e.g., in sand simulation)

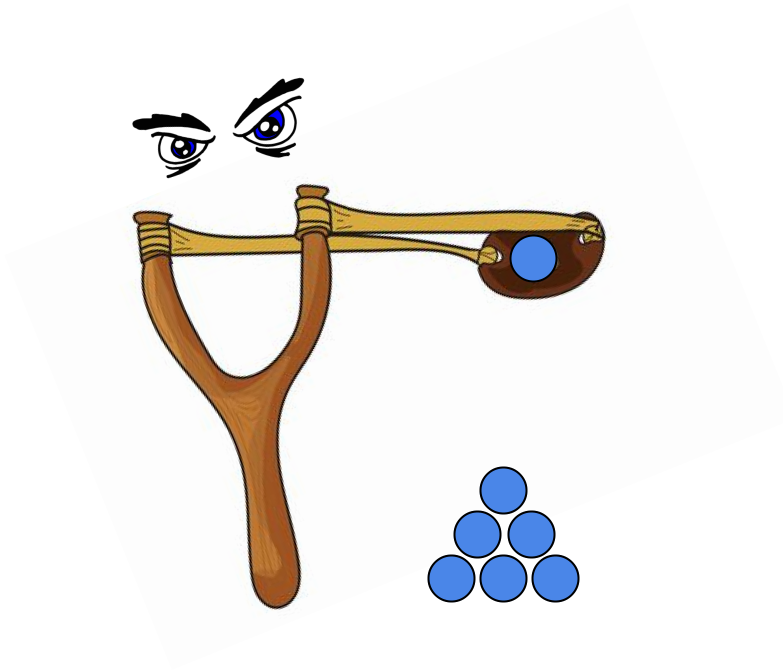
- SPH is better than Eulerian approaches
- Eulerian approaches are better than SPH

- "Proper" engineering CFD methods are grid-based

- SPH is (only) 0th-order consistent

- ...





Outline

<u>Block 1</u> (9:00 - 10:30)

- Foundations of SPH
- Governing equations
- Time integration
- <u>Example</u>: Our first SPH solver
- Neighborhood Search

Coffee break (30min)

<u>Block 2</u> (11:00 - 12:30)

- Enforcing incompressibility
  - State equation solvers
  - Implicit pressure solvers
- Boundary Handling
  - Particle-based methods
  - Implicit approaches

Lunch break (60min)

<u>Block 3</u> (13:30 - 15:00)

- Multiphase fluids
- Viscosity
- Vorticity and turbulence
- <u>Demo</u>: SPlisHSPlasH

Coffee break (30min)

<u>Block 4</u> (15:30 - 17:00)

- Deformable solids
- Rigid body simulation
  - Dynamics and coupling
- Data-driven/ML techniques
- Summary and conclusion

Foundations of SPH

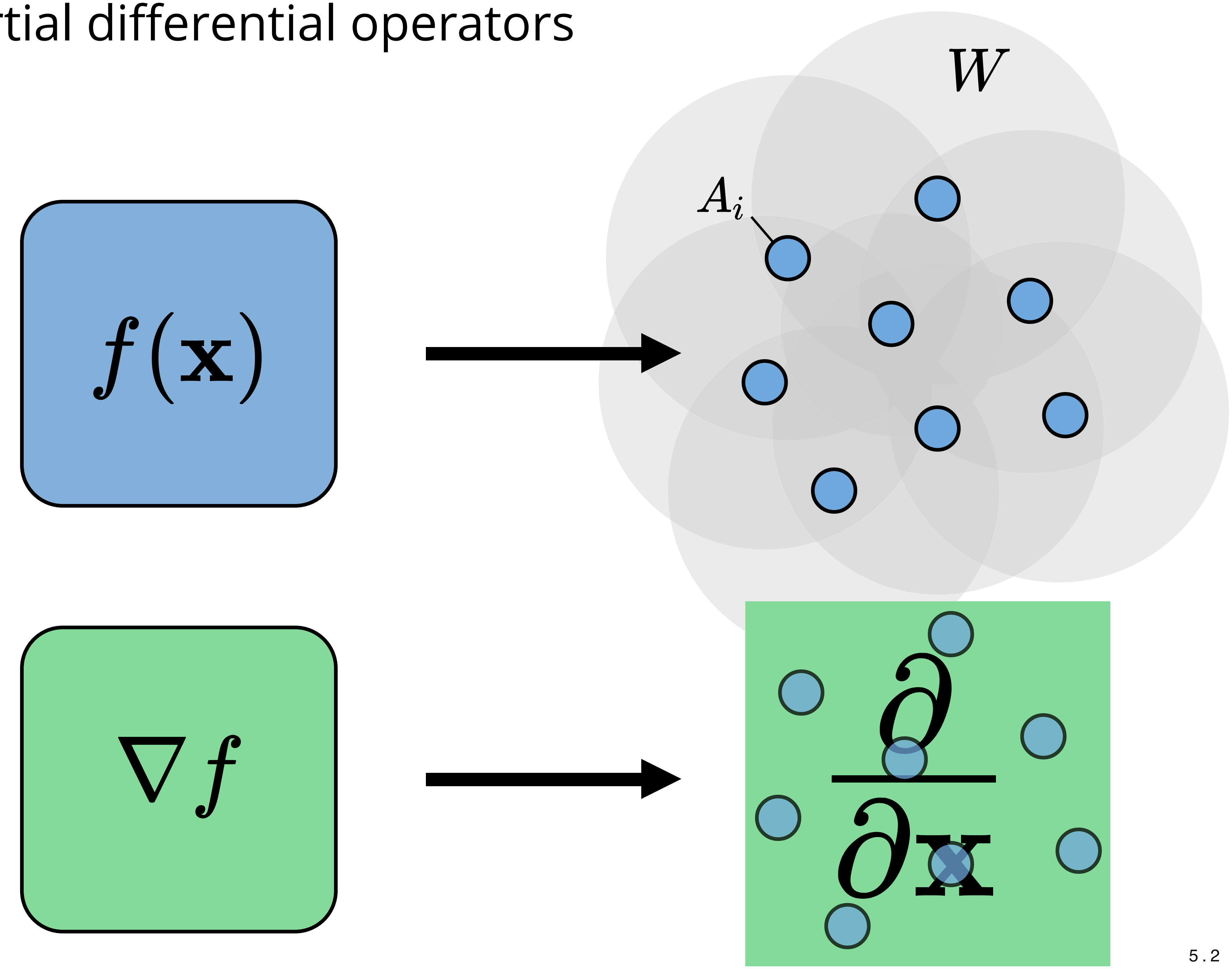
What is SPH?

" A mesh-free method for the discretization of "
functions and partial differential operators

- Functions are discretized into
  - Samples equipped with kernel function W
- Approximates/discretizes differential operators

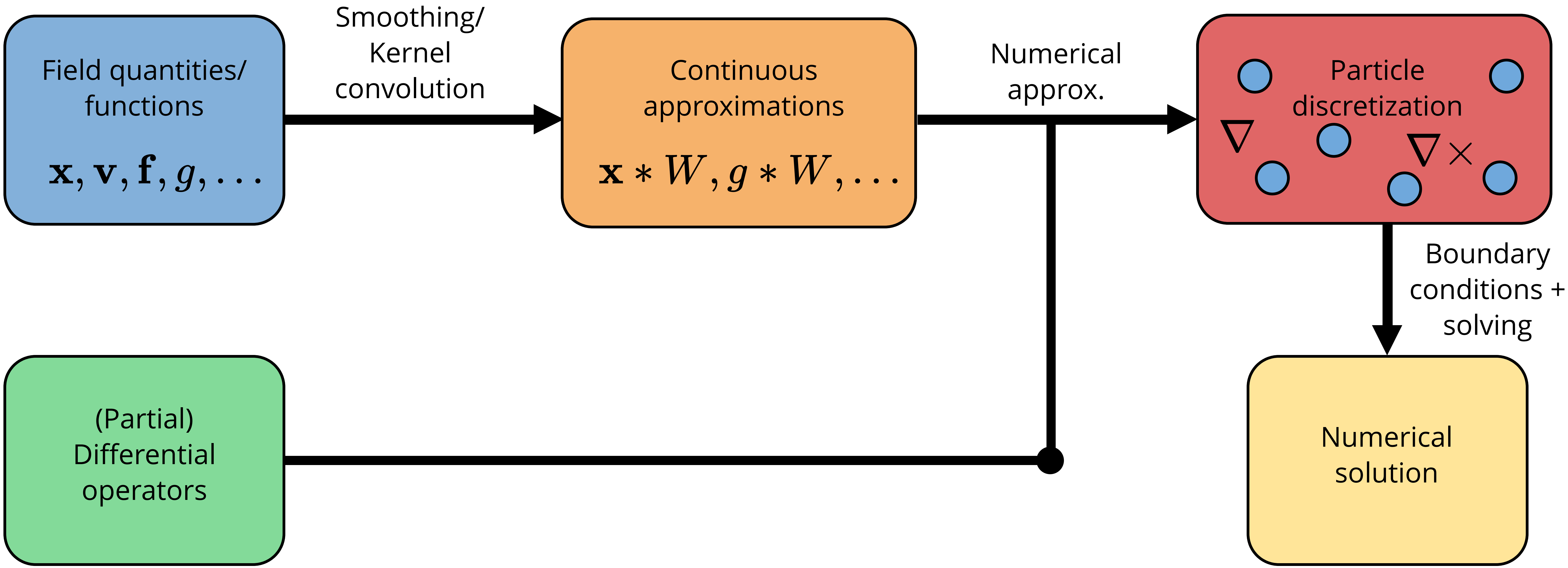
- Interpretation of SPH sample
  - Math.: Coefficients "controling" approx.
  - Phys.: Particle "carrying" quantities

- Useful to simulate continuum media
  - conservational properties
  - greatly handles topological changes
  - algorithms parallelize well
  - good for advection-type/transp. problems



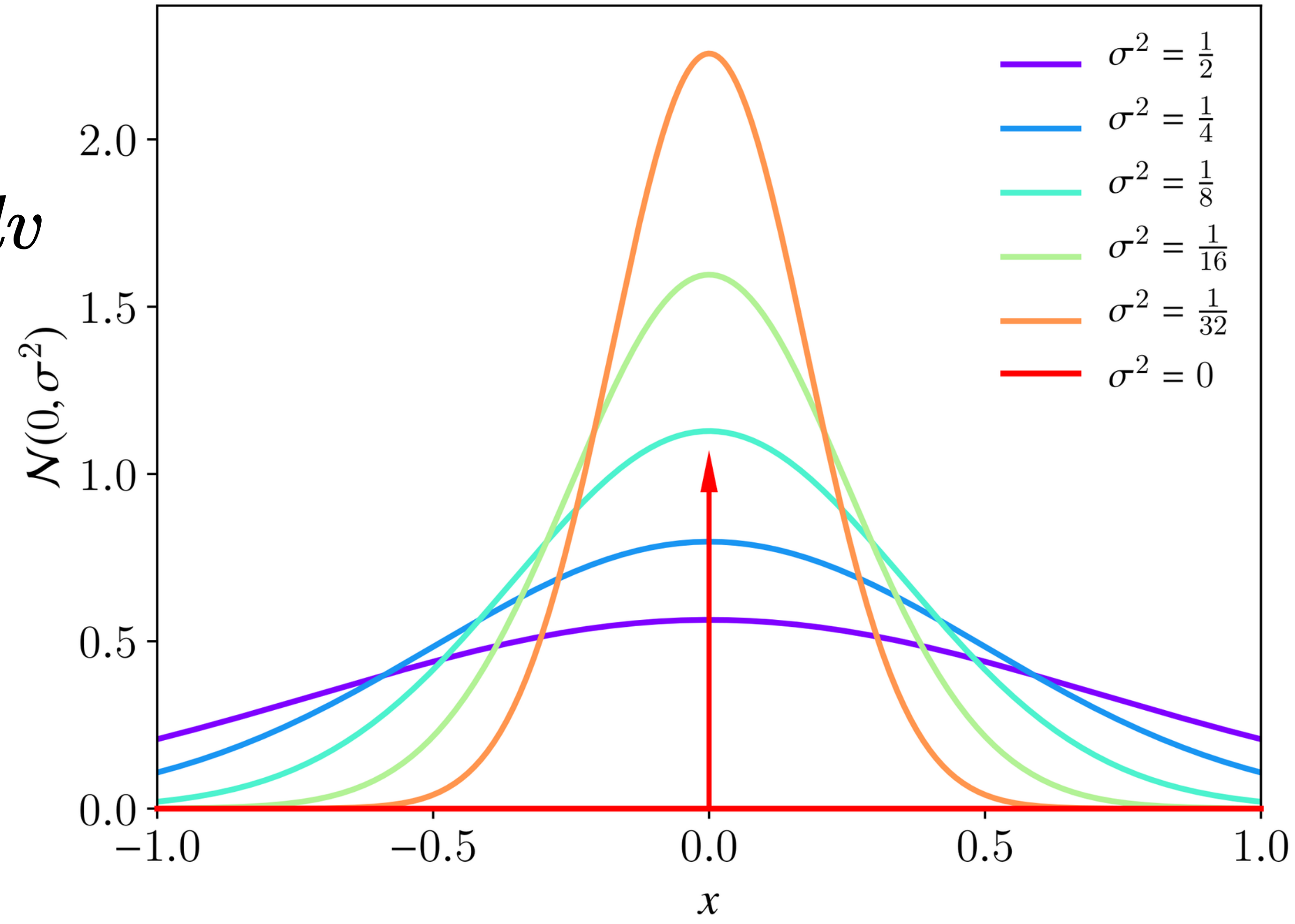
SPH Discretization Pipeline

PDE: $\nabla \times \mathbf{f} + \nabla g = \mathbf{h}(f, g)$



Continuous Approximation

- Dirac-δ function $$\delta(\mathbf{r}) = \begin{cases} \infty & \text{if } \mathbf{r} = \mathbf{0} \\ 0 & \text{otherwise} \end{cases}, \quad \int_{\mathbb{R}^d} \delta(\mathbf{x}) dv = 1$$
- Dirac-δ identity $$A(\mathbf{x}) = (A * \delta)(\mathbf{x}) = \int_{\mathbb{R}^d} A(\mathbf{x}')\, \delta(\mathbf{x} - \mathbf{x}') dv$$
- Approximation with Gaussian kernel $$\mathcal{N}(\mathbf{x}; \mu, \sigma^2), \ \lim_{\sigma \to 0} \mathcal{N}(\mathbf{x}; 0, \sigma^2) = \delta(\mathbf{x})$$
- Good choice because normalized, BUT: $\operatorname{supp}(\mathcal{N}) = \mathbb{R}^d$
- Instead use "some other" kernel: $W : \mathbb{R}^d \times \underbrace{\mathbb{R}^+}_{\text{Controls variance}} \to \mathbb{R}$ such that $$A(\mathbf{x}) \approx (A * W)(\mathbf{x})$$



Continuous Approximation - Kernel

- What's a good choice for the kernel $W : \mathbb{R}^d \times \mathbb{R}^+ \rightarrow \mathbb{R}$?

Desired properties

$$\int_{\mathbb{R}^d} W(\mathbf{r}', h) dv' = 1 \qquad \text{(normalization condition)}$$

$$\lim_{h' \to 0} W(\mathbf{r}, h') = \delta(\mathbf{r}) \qquad \text{(Dirac-}\delta\text{ condition)}$$

Essential for valid approximation $A(\mathbf{x}) \approx (A * W)(\mathbf{x})$

$$W(\mathbf{r}, h) \geq 0 \qquad \text{(positivity condition)}$$

→ (Optional) Ensures exclusively positive weighting, helps meeting physical constraints, e.g. $\rho \geq 0$

$$W(\mathbf{r}, h) = W(-\mathbf{r}, h) \qquad \text{(symmetry condition)}$$

→ (Optional) Allows 1st-order consistent approx.

$$W(\mathbf{r}, h) = 0 \text{ for } \|\mathbf{r}\| \geq \hbar, \qquad \text{(compact support condition)}$$

→ (Optional) Drastically improves efficiency.

- Kernel construction out of scope of this tutorial. See [LL10] for details.

Continuous Approximation - Kernel

- Cubic spline kernel; typical choice for $W : \mathbb{R}^d \times \mathbb{R}^+ \rightarrow \mathbb{R}$

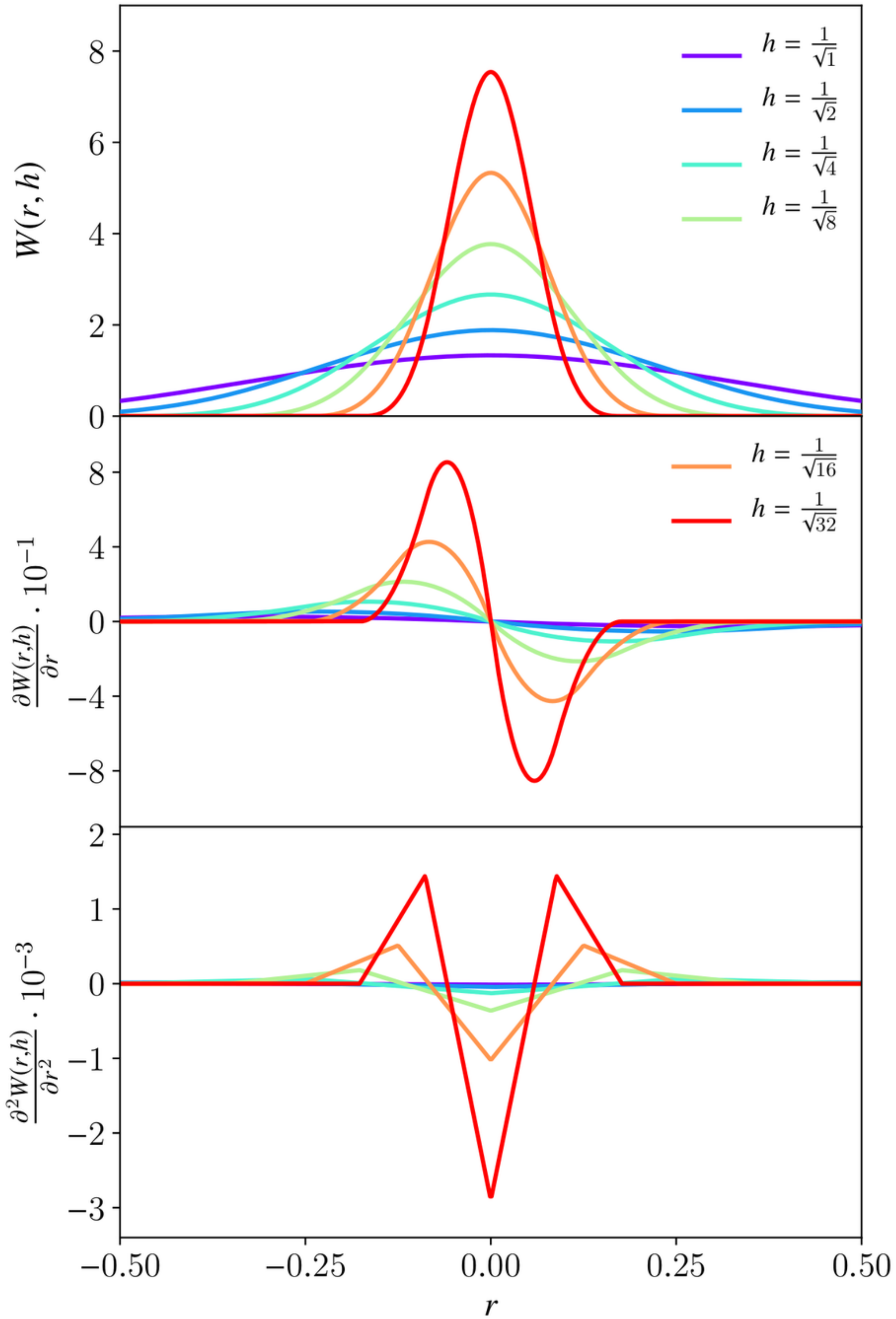
$$W(\mathbf{r}, h) = \sigma_d \begin{cases} 6(q^3 - q^2) + 1 & \text{for } 0 \leq q \leq \frac{1}{2} \\ 2(1-q)^3 & \text{for } \frac{1}{2} < q \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{with } q = \frac{1}{h}\|\mathbf{r}\|$$

- h denotes "smoothing length"
  - Controls support domain radius
- σ_d normalization constant
- C^2 -continuous

- Good choice? At least:

Kernel fulfills all conditions!



Continuous Approximation - Consistency

- How accurate is approximation?
- Polynomial error analysis:

$$(A * W)(\mathbf{x}) = \int \left[A(\mathbf{x}) + \nabla A|_{\mathbf{x}} \cdot (\mathbf{x}' - \mathbf{x}) + \frac{1}{2}(\mathbf{x}' - \mathbf{x}) \cdot \nabla\nabla A|_{\mathbf{x}} (\mathbf{x}' - \mathbf{x}) + \mathcal{O}(\|\mathbf{r}\|^3) \right] W(\mathbf{x} - \mathbf{x}', h) dv'$$

$$= A(\mathbf{x}) \underbrace{\int W(\mathbf{x} - \mathbf{x}') dv'}_{=1} + \nabla A|_{\mathbf{x}} \cdot \underbrace{\int (\mathbf{x} - \mathbf{x}') W(\mathbf{x} - \mathbf{x}') dv'}_{=0} + \mathcal{O}(\|\mathbf{r}\|^2)$$

If kernel normalized

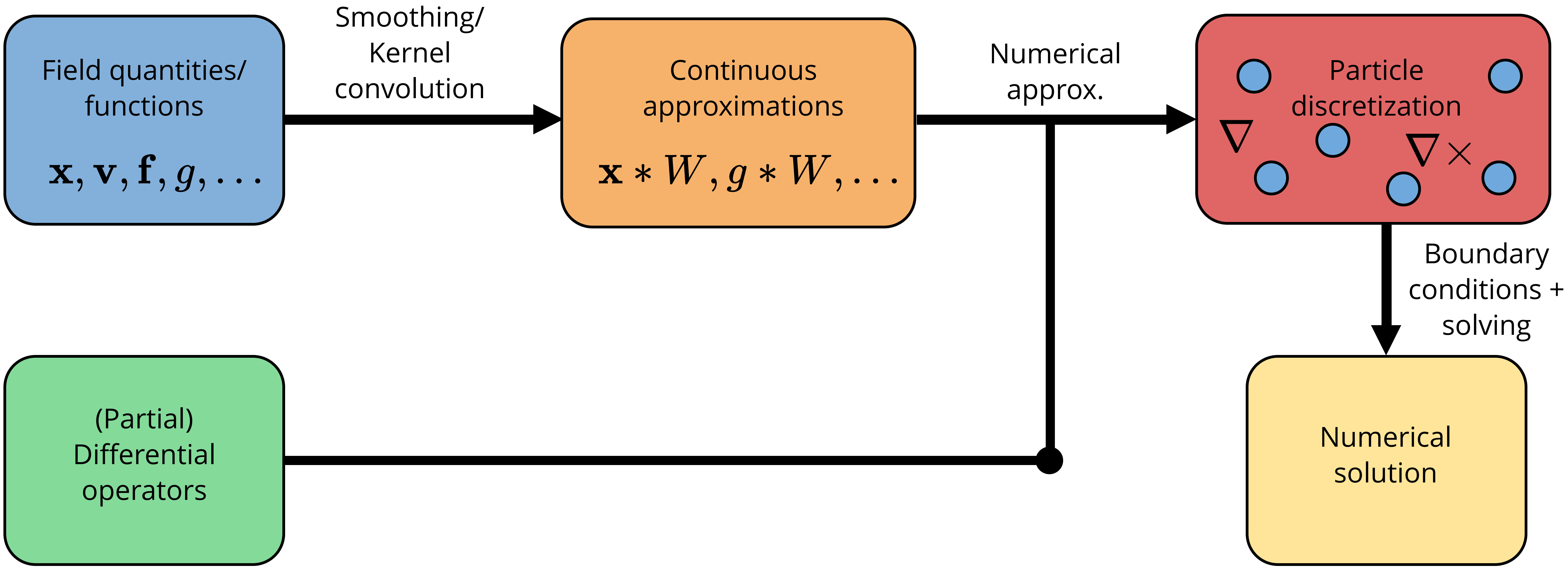
If kernel symmetric

- Normalized, symmetric kernels lead to (at least) 1st-order consistency

- Specialized kernels for higher-order consistency can be constructed

SPH Discretization Pipeline

PDE: $\nabla \times \mathbf{f} + \nabla g = \mathbf{h}(f, g)$



Field Discretization

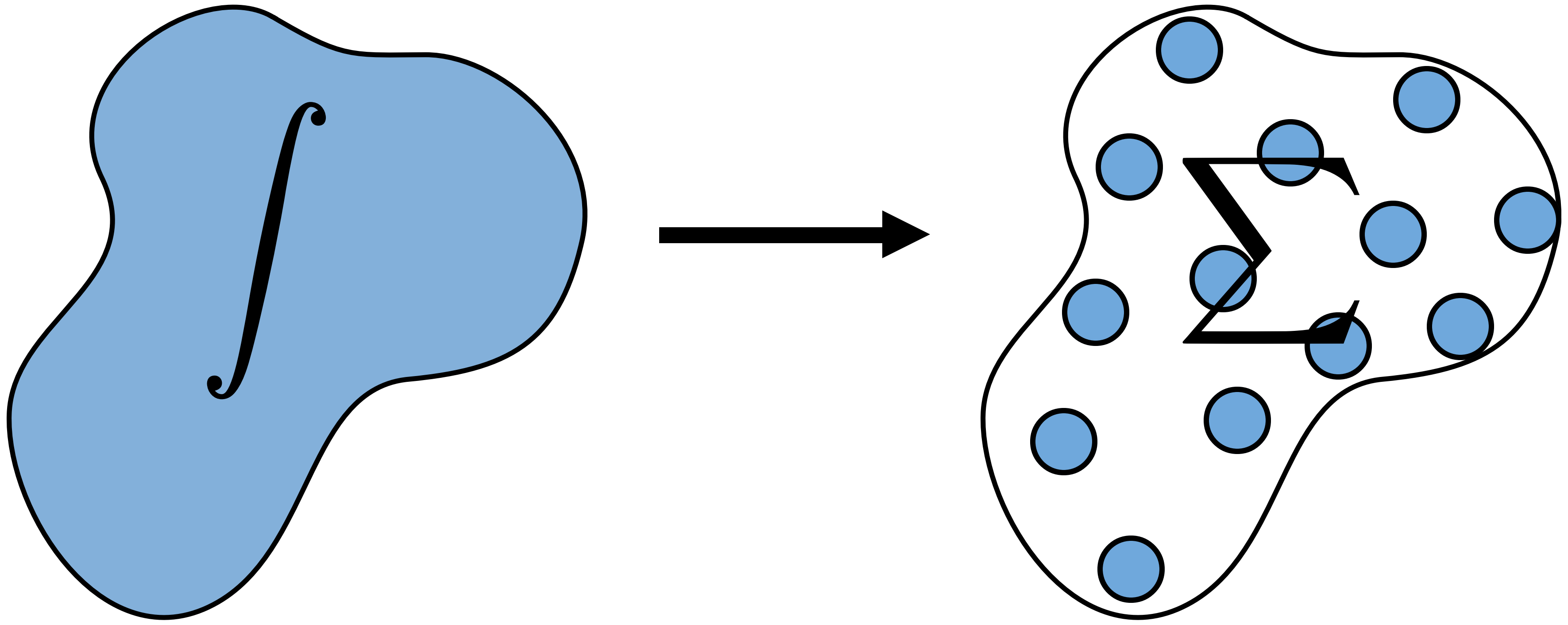
- Continuous approximation (convolution) involves integral
  - Analytic evaluation generally not possible
  - Numerical integration required

- Requires discretization

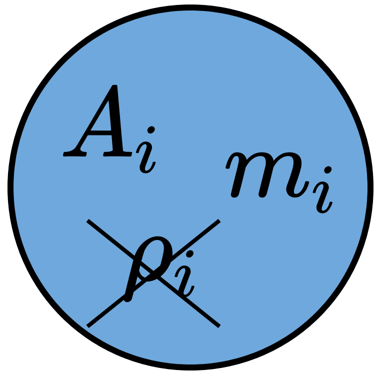
$$(A * W)(\mathbf{x}_i) = \int \frac{A(\mathbf{x}')}{\rho(\mathbf{x}')} W(\mathbf{x}_i - \mathbf{x}', h) \underbrace{\rho(\mathbf{x}') dv'}_{dm'}$$

$$\approx \sum_{j \in \mathcal{F}} A_j \frac{m_j}{\rho_j} W(\mathbf{x}_i - \mathbf{x}_j, h) := \langle A(\mathbf{x}_i) \rangle$$

- Monte-Carlo-like numerical integration

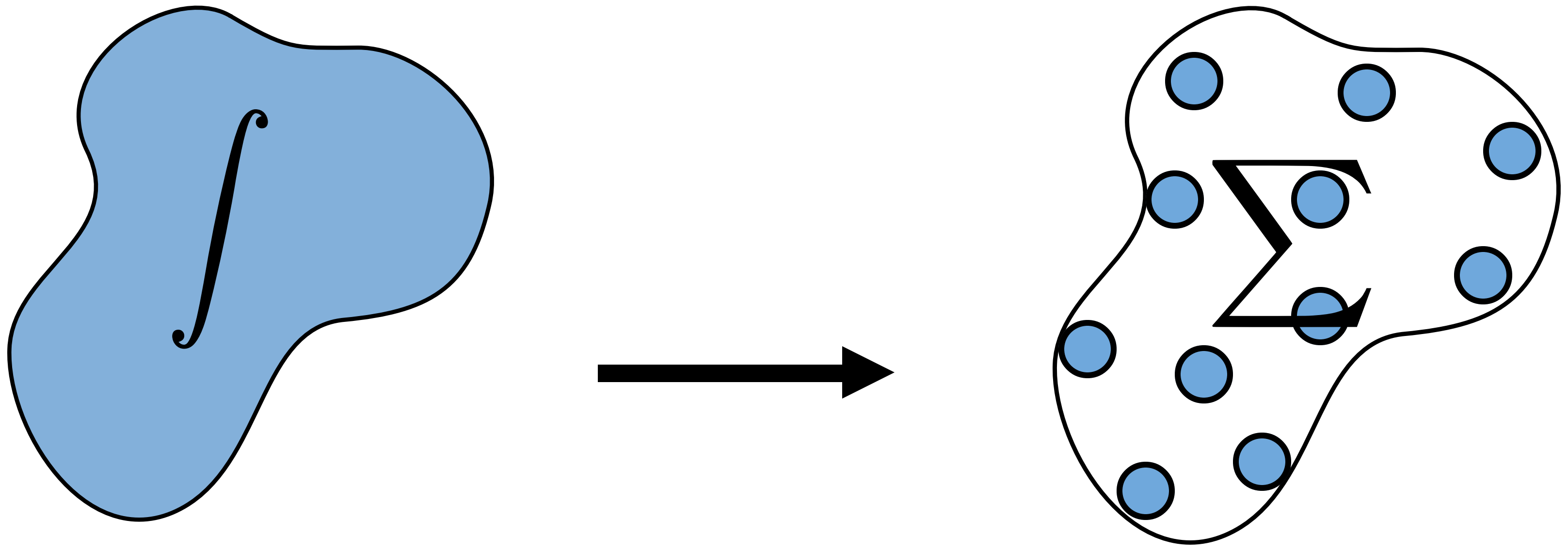


- Each particle carries
  - Field sample
  - Particle mass
  - Density <u>not</u> necessary



Field Discretization - Consistency

- What are we sacrificing?



$$\int \frac{A(\mathbf{x}')}{\rho(\mathbf{x}')} W(\mathbf{x}-\mathbf{x}', h)dm' \qquad \langle A \rangle$$

- Polynomial error analysis:

$$\langle A \rangle = A_i \boxed{\sum_j \frac{m_j}{\rho_j} W_{ij}} + \nabla A|_{\mathbf{x}_i} \cdot \boxed{\sum_j \frac{m_j}{\rho_j}(\mathbf{x}_j - \mathbf{x}_i) W_{ij}} + \mathcal{O}(\|\mathbf{r}\|^2)$$

- For 1st-order consistency:

$$\boxed{\sum_j \frac{m_j}{\rho_j} W_{ij} = 1} \qquad \boxed{\sum_j \frac{m_j}{\rho_j}(\mathbf{x}_j - \mathbf{x}_i) W_{ij} = \mathbf{0}}$$

Field Discretization - Consistency

- What are we sacrificing?

$$\sum_j \frac{m_j}{\rho_j} W_{ij} = 1$$

For 0th-order

$$\sum_j \frac{m_j}{\rho_j} (\mathbf{x}_j - \mathbf{x}_i) W_{ij} = \mathbf{0}$$

For 1st-order

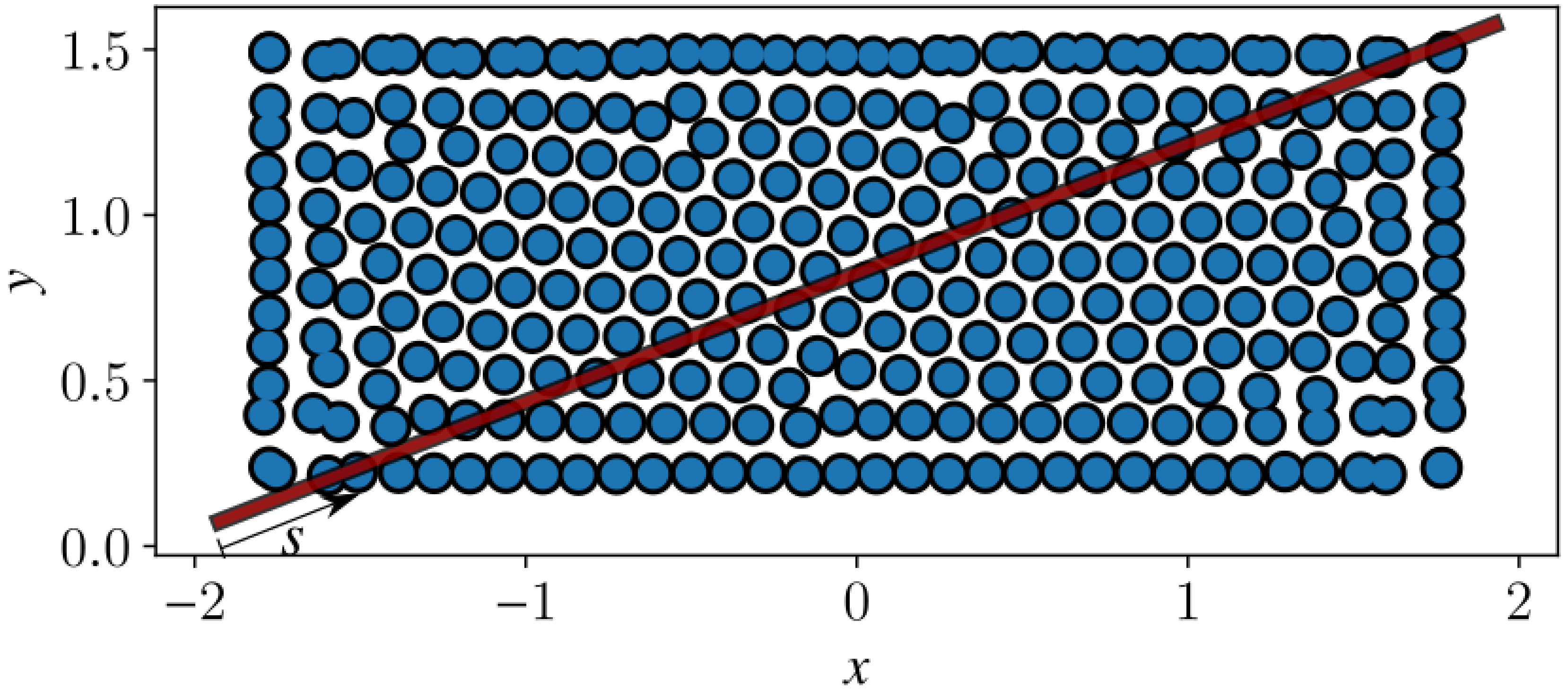
- => Particle ordering important
- => Almost never satisfied

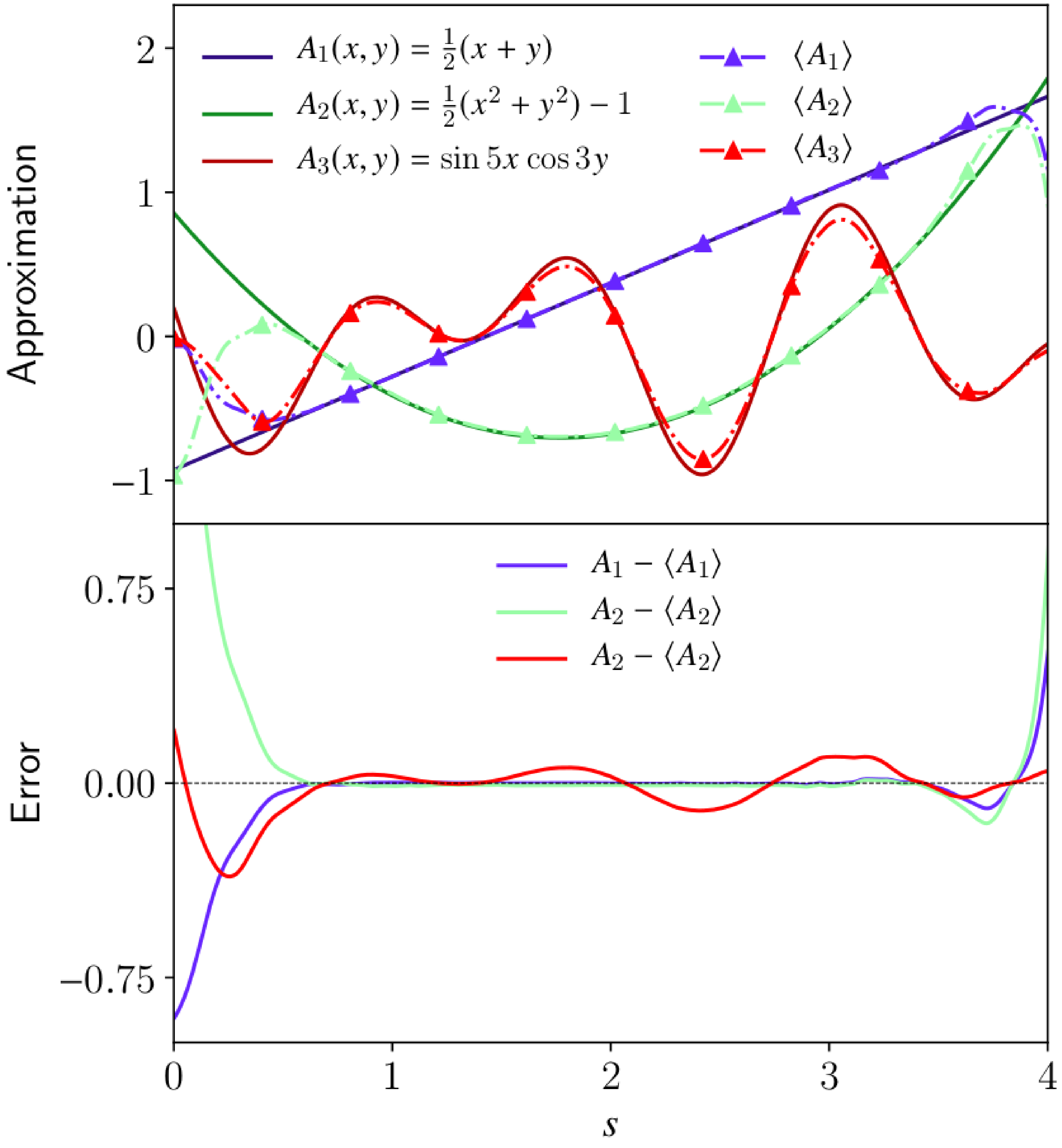
Consequence: Without further treatment we even lose 0th-order consistency

- Is this a problem?
  - **Not really!** Approximation accuracy usually still high!
  - Alternatively: order recovery (normalization, matrix-inversion)
- Common practice:
  - Graphics community: Often ignored. Visual quality still good.
  - Engineering community: Similar. Sometimes order recovery
  - Generally: Depends...
- Polynomial error is only half of the truth! We'll see that later

Field Discretization - Example

- Setting:
  - Rectangular domain discretized into particles
  - Test function sampled on particles
  - Discretization quality is tested along red line
- Results despite concistency order





Mass Density Estimation

- Density does not have to be carried by particles
  - Can be recovered/estimated

- Recall

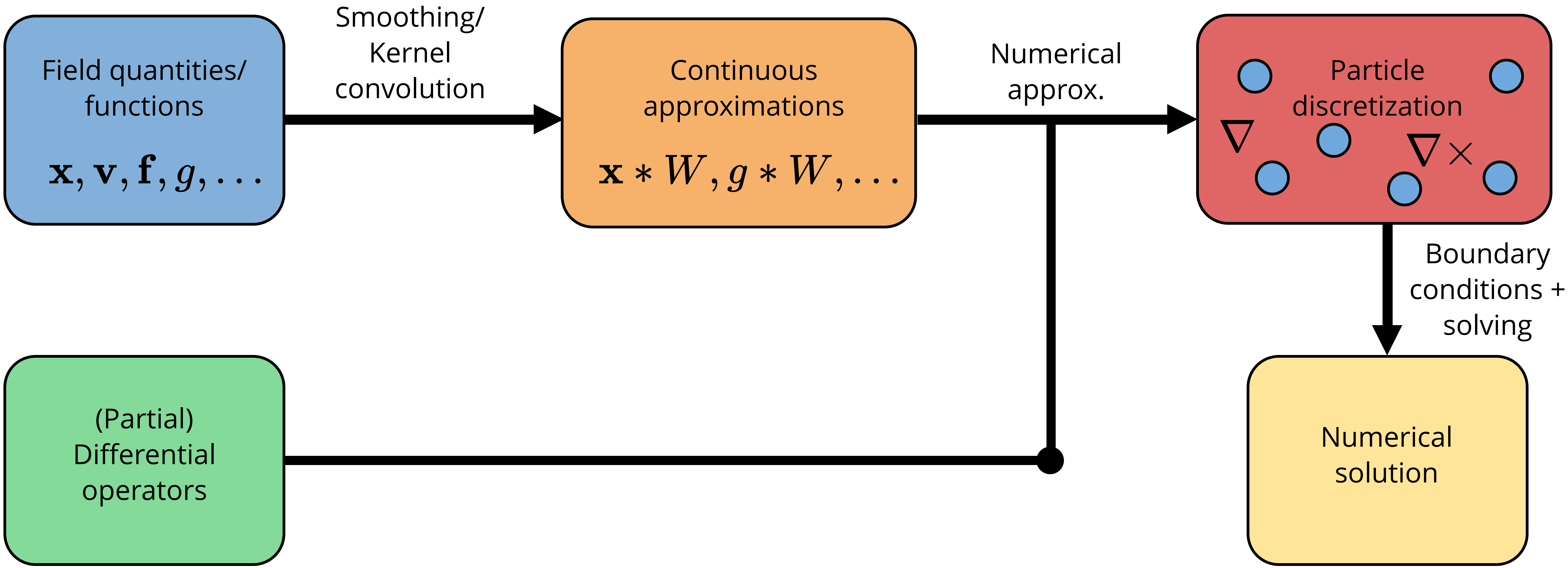
$$\langle A(\mathbf{x}_i) \rangle = \sum_{j \in \mathcal{F}} A_j \frac{m_j}{\rho_j} W_{ij}$$

- Plugging in density field directly yields:

$$\rho(\mathbf{x}_i) \approx \sum_{j \in \mathcal{F}} m_j W_{ij}$$

SPH Discretization Pipeline

PDE: $\nabla \times \mathbf{f} + \nabla g = \mathbf{h}(f, g)$



Discrete Differential Operators

- Direct discretization:
  - Derivative "shifts" to kernel function

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

$$\nabla \mathbf{A}_i \approx \sum_j \mathbf{A}_j \frac{m_j}{\rho_j} \otimes \nabla W_{ij}$$

$$\nabla \cdot \mathbf{A}_i \approx \sum_j \mathbf{A}_j \frac{m_j}{\rho_j} \cdot \nabla W_{ij}$$

$$\nabla \times \mathbf{A}_i \approx -\sum_j \mathbf{A}_j \frac{m_j}{\rho_j} \times \nabla W_{ij}$$

- Very simple
- Kernel gradient can be reused

!!! Direct method leads to unstable simulations !!!

- Improved variants:
  - Difference formula
  - Symmetric formula
  - ...

Difference Formula

- Direct gradient $$\nabla A_i \approx \sum_j A_j \frac{m_j}{\rho_j} \nabla W_{ij}$$
- Polynomial error analysis reveals:

$$\boxed{\langle \nabla 1 \rangle = \sum_j \frac{m_j}{\rho_j} \nabla W_{ij} = \mathbf{0}}$$

For 0th-order

$$\boxed{\sum_j \frac{m_j}{\rho_j} (\mathbf{x}_j - \mathbf{x}_i) \nabla W_{ij} = \mathbb{I}}$$

For 1st-order

- 0th-order can be recovered by subtracting first error term:

$$\nabla A_i \approx \langle \nabla A_i \rangle - A_i \langle \nabla 1 \rangle = \sum_j \frac{m_j}{\rho_j} (A_j - A_i) \nabla_i W_{ij}$$

(Difference formula)

- 1th-order can be recovered by small matrix inversion:

$$\nabla A_i \approx \mathbf{L}_i \left(\sum_j \frac{m_j}{\rho_j} (A_j - A_i) \nabla_i W_{ij} \right) \qquad \mathbf{L}_i = \left(\sum_j \frac{m_j}{\rho_j} \nabla_i W_{ij} \otimes (\mathbf{x}_j - \mathbf{x}_i) \right)^{-1}$$

Symmetric Formula

- Gradient derived from discrete Lagragian and density estimate in hydrodynamic systems:

$$\nabla A_i \approx \rho \left(\tfrac{A_i}{\rho_i^2} \langle \nabla \rho \rangle + \langle \nabla \left(\tfrac{A_i}{\rho_i} \right) \rangle \right)$$

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

(Symmetric formula)

- Approximation not even 0th-order consistent. **BUT**:
  - momentum conserving
  - error is guided by particle ordering (non-linear properties, symmetries, conservation, etc.)
  - => Leads to more stable simulations (at least in the hydrodynamic setting)
- Consequence:

Polynomial error analysis is only half the truth!

- More information: [Price 2012], Smoothed Particle Hydrodynamics and Magnetohydrodynamics
  - Sec. 5 "*Why a bad derivative leads to good derivatives: The importance of local conservation*"

Discrete Laplace Operator

- Direct Laplacian
  - Again: Rather bad approximation

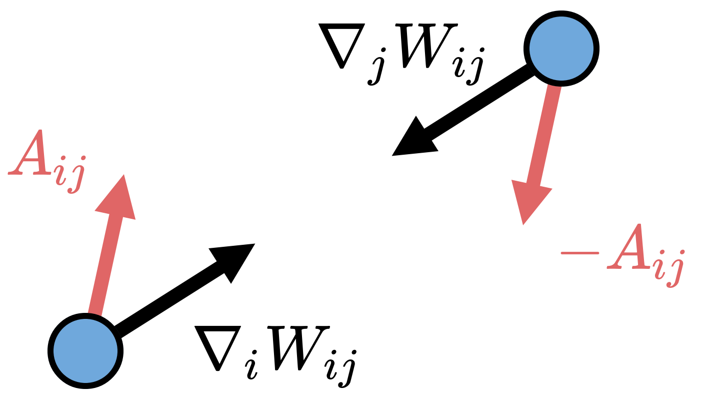
$$\nabla^2 A_i \approx \sum \frac{m_j}{\rho_j} A_j \nabla_i^2 W_{ij}$$

- Improved version by Brookshaw [Bro85]
  - Is of "difference type"

$$\nabla^2 A_i \approx -\sum_j \frac{m_j}{\rho_j} A_{ij} \frac{2\|\nabla_i W_{ij}\|}{\|\mathbf{r}_{ij}\|}$$

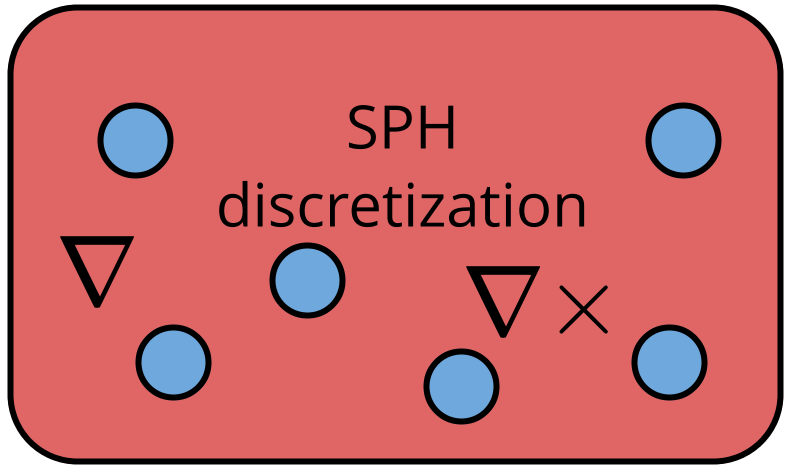
- Even improved version not optimal for vector Laplacians
  - Derived forces (e.g., viscosity force) not momentum conserving
  - If field is divergence-free momentum conserving approximation can be made!

$$\text{if } \nabla \cdot \mathbf{A} = 0 \qquad \Rightarrow \qquad \nabla^2 \mathbf{A}_i = 2(d+2) \sum_j \frac{m_j}{\rho_j} \frac{\mathbf{A}_{ij} \cdot \mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|^2} \nabla_i W_{ij}$$

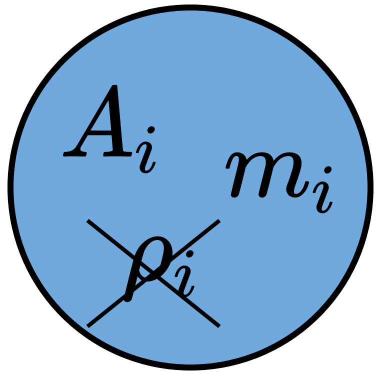


Quick Recap

- SPH discretizes
  - (spatial) functions
  - differential operators (PDEs)



- An SPH particle is
  - a coefficient in the approximation
  - a sample that "carries" a field quantity



- Symmetric, normalized kernels enforce <u>at least 1st-order consistency</u> in cont. approximation
- 0th-order consistency not guaranteed in particle discretiztion
  - Accuracy still good in practice
  - Local conservation properties sometimes more important than consistency orders

- Specialized gradient operators
  - Difference-type operator for 0th/1st-order consistency
  - Symmetric gradient for local conservation properties (useful for physical forces)

Continuum Mechanical Models

Governing Equations

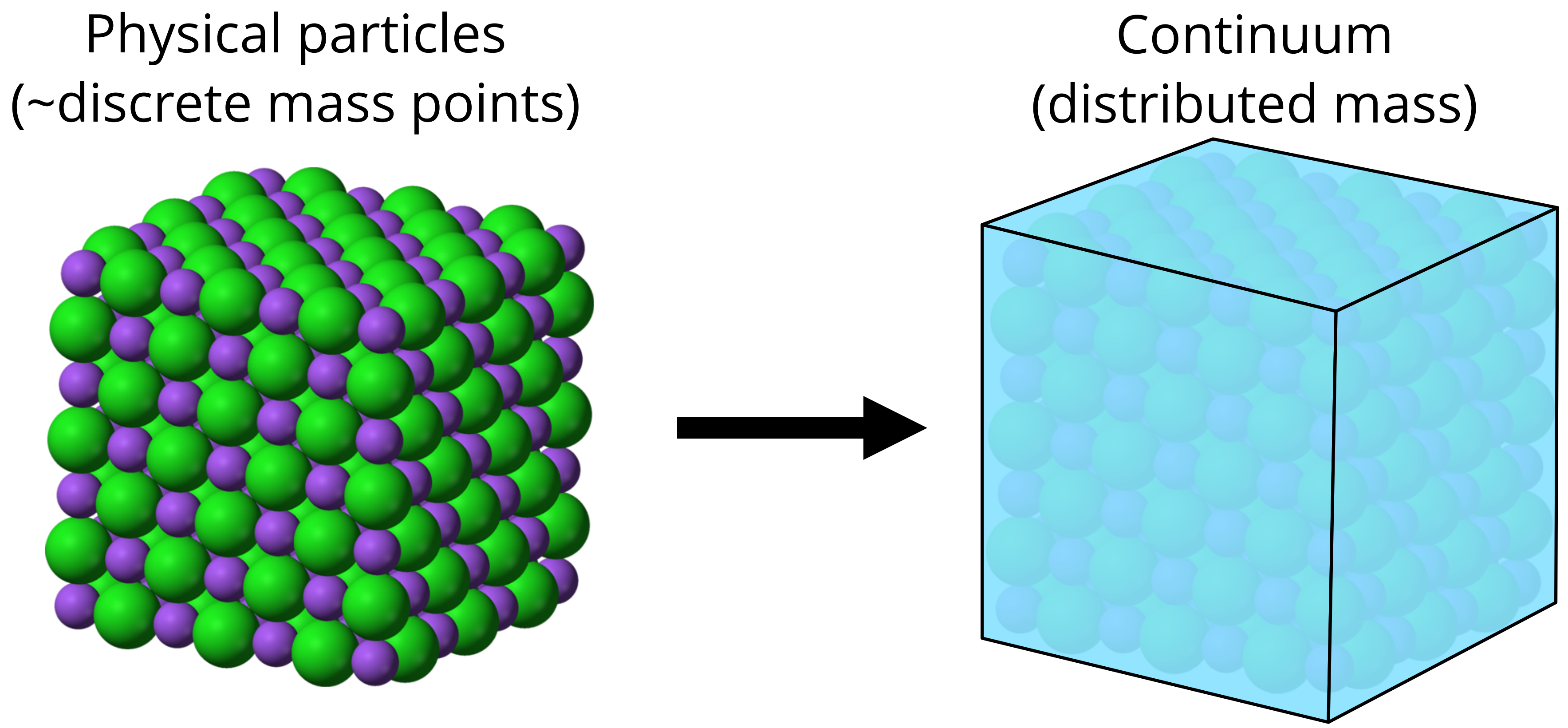
Governing Equations

- How to model fluids and solids?

- Graphics applications:
  - Visual appearance
  - Mostly governed by macroscopic behavior

- Typically continuum mechanical approach

- What is Continuum Mechanics?
  - Continuously distributed mass
  - Object can be infinitely often divided
  - Models physical phenomena as PDE
    - Linear elasticity, Navier-Stokes-Equations, etc.



Continuity Equation

- Describes evolution of object's mass density over time

$$\frac{D\rho}{Dt} = -\rho(\nabla \cdot \mathbf{v})$$

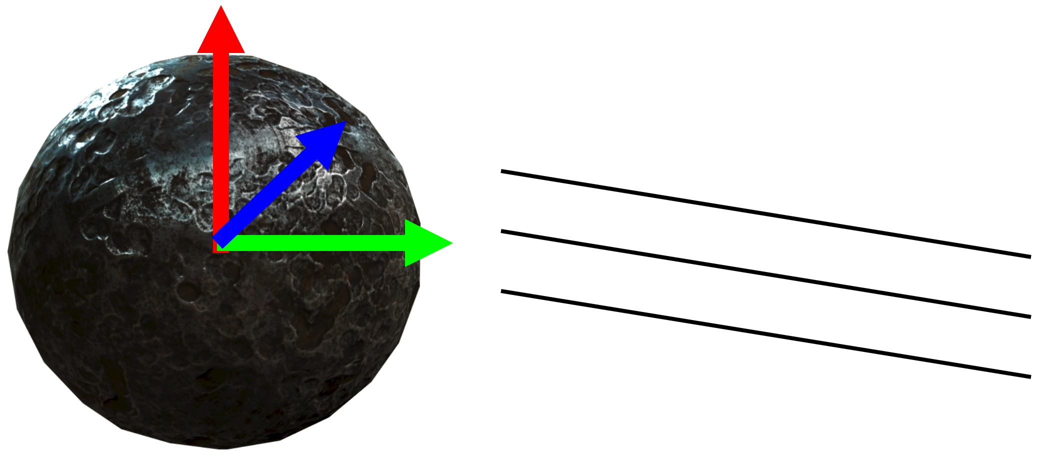
- $\frac{D}{Dt}(\cdot)$ denotes material derivative (more on the next slide)

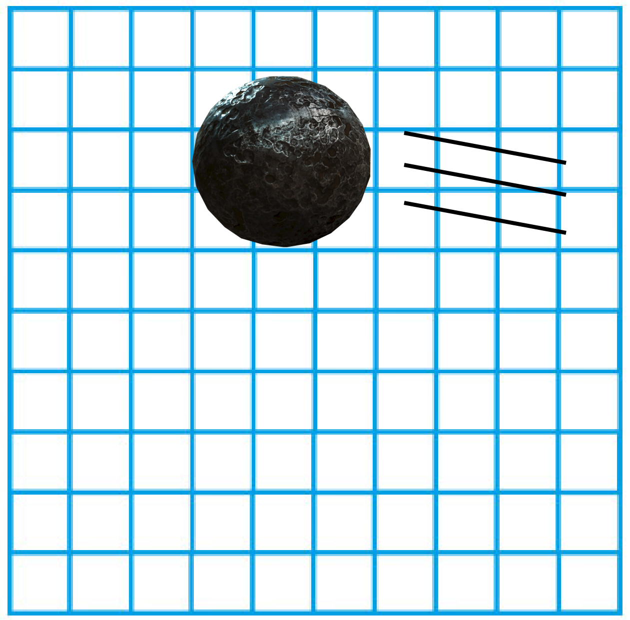
- Important for incompressible materials
  - Constraint:

$$\frac{D\rho}{Dt} = 0 \qquad \Leftrightarrow \qquad \nabla \cdot \mathbf{v} = 0$$

- Should be explicitly fulfilled in case of incompressible materials

Lagrangian vs. Eulerian Coordinates





Lagrangian Coordinates

- Identify (or label) a material of the fluid
- Track material particle as it moves
- Monitor change in its properties
- Field: $A_p^L(t)$

$$\frac{DA^L}{Dt} = \frac{\partial A^L}{\partial t}$$

<u>SPH advantageous in Lagrangian setting</u>

Eulerian Coordinates

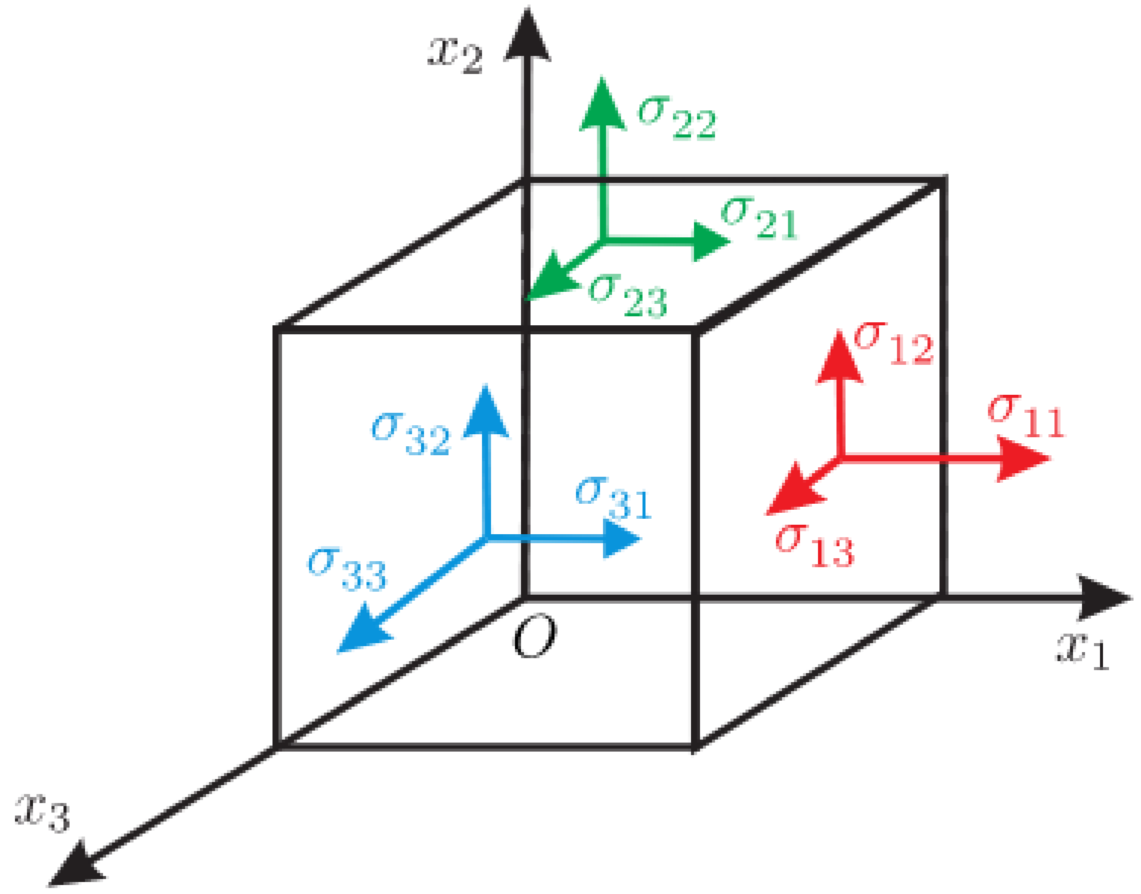
- Identify (or label) fixed location
- Observe fixed location (like sensor)
- Monitor change in properties during flow
- Field: $A^E(t, \mathbf{x})$

$$\frac{DA^E}{Dt} = \frac{\partial A^E}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} A^E$$

Linear Momentum Conservation

- Local balance law:

$$\rho \frac{D^2\mathbf{x}}{Dt^2} = \nabla \cdot \mathbf{T} + \mathbf{f}_{\text{ext}}$$



- $\mathbf{T}$ denotes Stress tensor
  - 2nd-order tensor [N/m^2]

- Often called: "Equation of Motion"
- Interpretation:
  - Generalization of Newtons 2nd law of motion for continua

- Completely material-independent
  - <u>Holds for fluids and solids</u> (and more)
  - Material behavior is encoded in stress tensor

Modelling Fluids

- How does stress tensor look like for a fluid?

- We need a so-called constitutive model!
  - Relates stress with strain, velocity, temperature, etc.
- **Newtonian fluid:**
  - Stress as a function of pressure and velocity

$$\mathbf{T} = -p\mathbb{I} + \mu(\nabla \mathbf{v} + \nabla \mathbf{v}^T)$$

- First term: resistance to compression
- Second term: viscosity

- Equation of motion for incompressible Newtonian fluid:

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{f}_{\text{ext}}$$

(Navier-Stokes equation)

Modelling (Elastic) Solids

- How does stress tensor look like for an elastic solid?
- Which constitutive model now?
  - We need relation between stress and deformation/strain!
- **Linear Elasticity:**
  - Stress as a linear function of strain measure

$$\mathbf{T} = \mathbf{C} : \boldsymbol{\varepsilon}$$

(Generalized Hooke's law)

- Can be interpreted as higher-dimensional spring
- In isotropic case:
  - Function of Young's modulus and Poisson ratio

$$\mathbf{C} = \mathbf{C}(E, \nu)$$

- More complex non-linear relations possible

Numerical Solving - Recipe

1. **Identify model - Constitutive model (fluid/solid/...), strain measure, incompressible?, ...**

2. **Split operators for simplification (see next slide)**

3. **Discretize fields and <u>spatial</u> differential operators using SPH**

4. **Discretize temporal derivatives (time integration)**

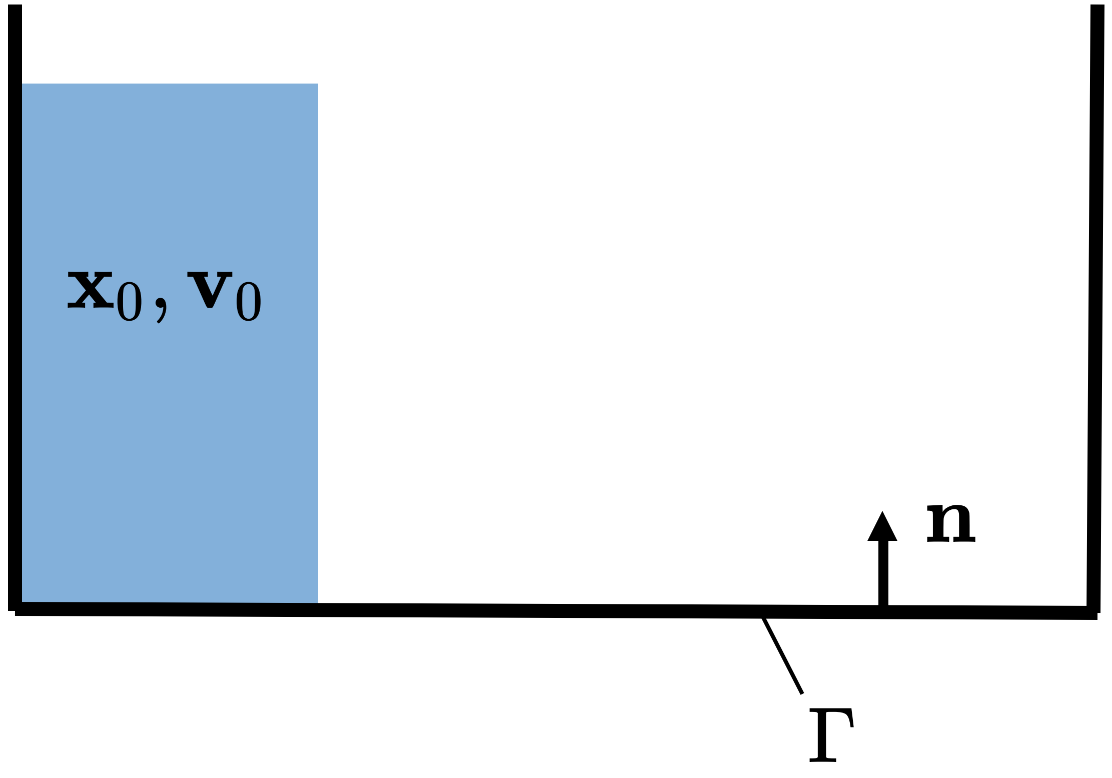
Solving - Step 1/2 - Model/Operator Split.

- Step 1, Final mixed initial-boundary value, e.g., weakly compressible Newtonian fluid

PDE	**Incompressibility**	**Initial conditions**	**Boundary conditions**
$\rho\frac{D\mathbf{v}}{Dt} = -\nabla p + \mu\nabla^2\mathbf{v} + \mathbf{f}_{\text{ext}}$ $\frac{D\mathbf{x}}{Dt} = \mathbf{v}$	$\frac{D\rho}{Dt} \approx 0$	$\mathbf{x}(t_0) = x_0$ $\mathbf{v}(t_0) = \mathbf{v}_0$	$\mathbf{v}\cdot\mathbf{n} \geq 0 \text{ on } \Gamma$

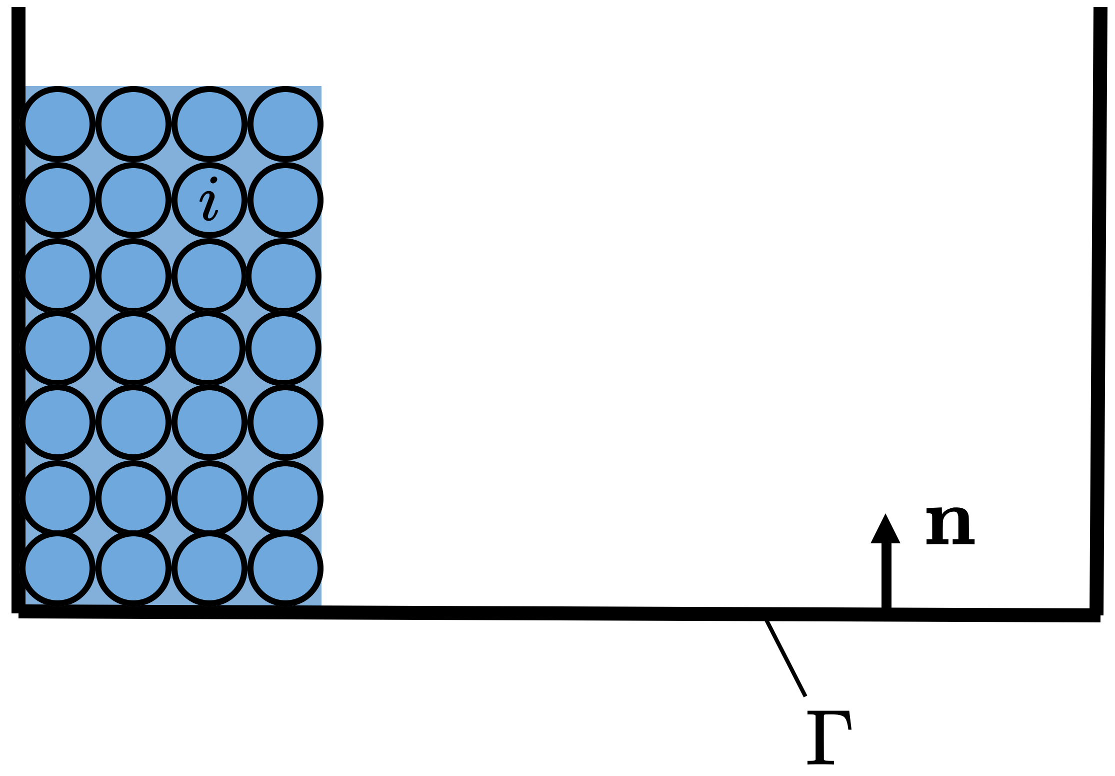
- Step 2, Operator splitting

1. Update $\mathbf{v}$ by solving $\frac{D\mathbf{v}}{Dt} = \frac{\mu}{\rho}\nabla^2\mathbf{v} + \frac{1}{\rho}\mathbf{f}_{\text{ext}}$
2. Determine ∇p using state equation $p = k(\rho - \rho_0)$
3. Update $\mathbf{v}$ by solving $\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p$
4. Update $\mathbf{x}$ by solving $\frac{D\mathbf{x}}{Dt} = \mathbf{v}$



Solving - Step 3 - Spatial Discretization

- Sample fields using SPH particles $\mathbf{x}, \mathbf{v}, m$
- Use reconstruction for density field $\rho_i = \sum_j m_j W_{ij}$
- Discretize (spatial) differential operators ∇, ∇^2



1. Update $\mathbf{v}_i$ by solving $\frac{D\mathbf{v}_i}{Dt} = \frac{\mu}{m_i} \sum_j \frac{m_j}{\rho_j} \mathbf{v}_{ij} \frac{2\|\nabla_i W_{ij}\|}{\|\mathbf{r}_{ij}\|} + \frac{1}{m_i}\mathbf{F}_{\text{ext}}$

2. Determine $\mathbf{F}_i^p$ using state equation $p_i = k(\rho_i - \rho_0)$ $\quad \mathbf{F}_i^p = \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla_i W_{ij}$

3. Update $\mathbf{v}_i$ by solving $\frac{D\mathbf{v}_i}{Dt} = -\frac{1}{m_i}\mathbf{F}_i^p$

4. Update $\mathbf{x}_i$ by solving $\frac{D\mathbf{x}_i}{Dt} = \mathbf{v}_i$

Solving - Step 4 - Time Discretization

- Decide for time integration scheme
  - Many schemes exist, e.g., Euler type, Runge-Kutta, BDF, Rosenbrock, Leapfrog, ...
- Let's use symplectic Euler

1. Update $\mathbf{v}_i$ by solving $\mathbf{v}_i^* = \mathbf{v}_i + \Delta t(\frac{\mu}{m_i} \sum_j \frac{m_j}{\rho_j} \mathbf{v}_{ij} \frac{2\|\nabla_i W_{ij}\|}{\|\mathbf{r}_{ij}\|} + \frac{1}{m_i}\mathbf{F}_{\text{ext}})$

2. Determine $\mathbf{F}_i^p$ using state equation $p_i = k(\rho_i - \rho_0)$ $\mathbf{F}_i^p = \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla_i W_{ij}$

3. Update $\mathbf{v}_i$ by solving $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^* - \frac{\Delta t}{m_i}\mathbf{F}_i^p$

4. Update $\mathbf{x}_i$ by solving $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t+\Delta t)$

Solving - Final algorithm

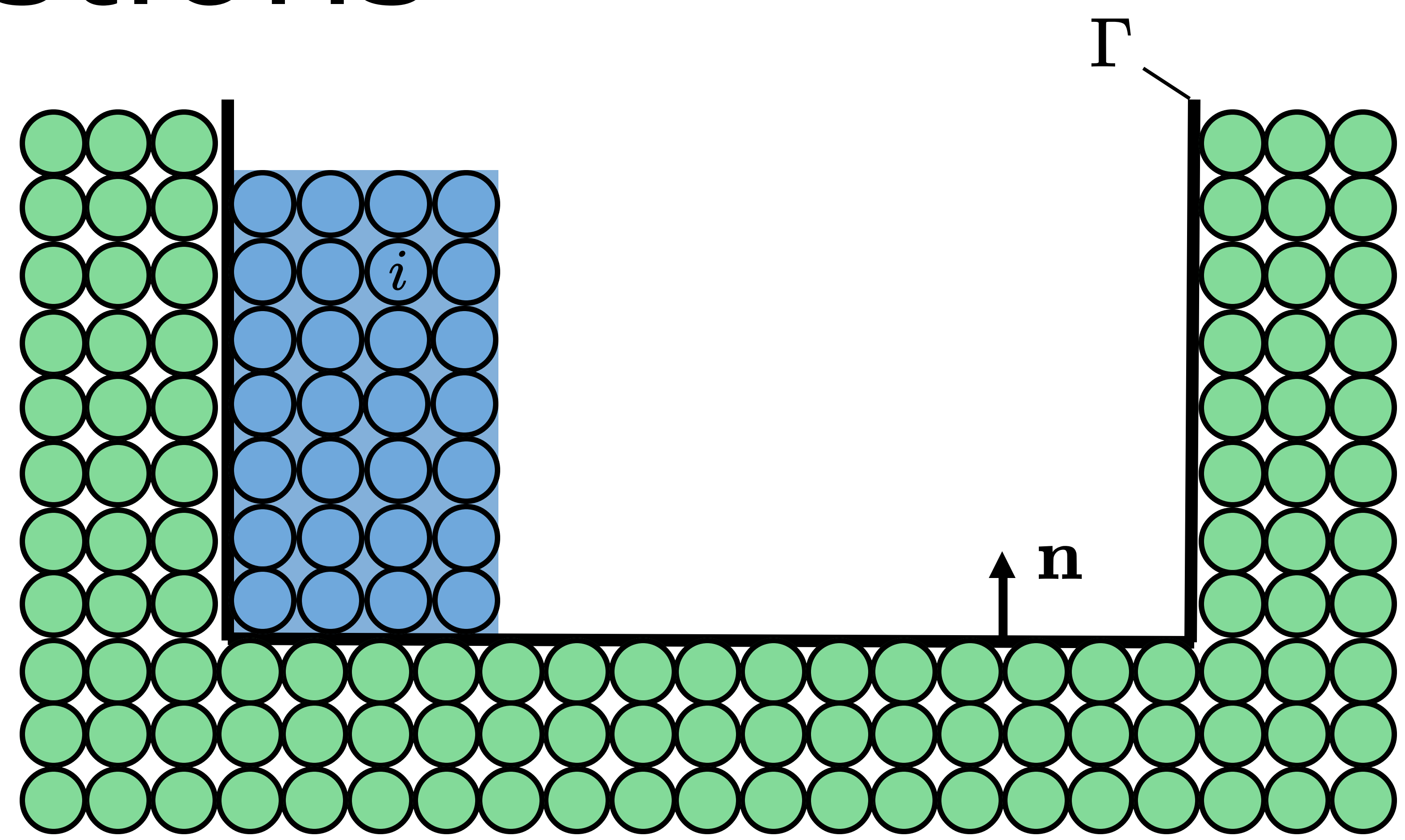
for all *particle i* **do**
  Reconstruct density ρ_i at $\mathbf{x}_i$ with Eq. (11)

for all *particle i* **do**
  Compute $\mathbf{F}_i^{\text{viscosity}} = m_i \nu \nabla^2 \mathbf{v}_i$, *e.g.*, using Eq. (23)
  $\mathbf{v}_i^* = \mathbf{v}_i + \frac{\Delta t}{m_i}(\mathbf{F}_i^{\text{viscosity}} + \mathbf{F}_i^{\text{ext}})$

for all *particle i* **do**
  Compute $\mathbf{F}_i^{\text{pressure}} = -\frac{1}{\rho}\nabla p$ using state eq. and Eq. (19)

for all *particle i* **do**
  $\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i^* + \frac{\Delta t}{m_i}\mathbf{F}_i^{\text{pressure}}$
  $\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i + \Delta t\,\mathbf{v}_i(t + \Delta t)$

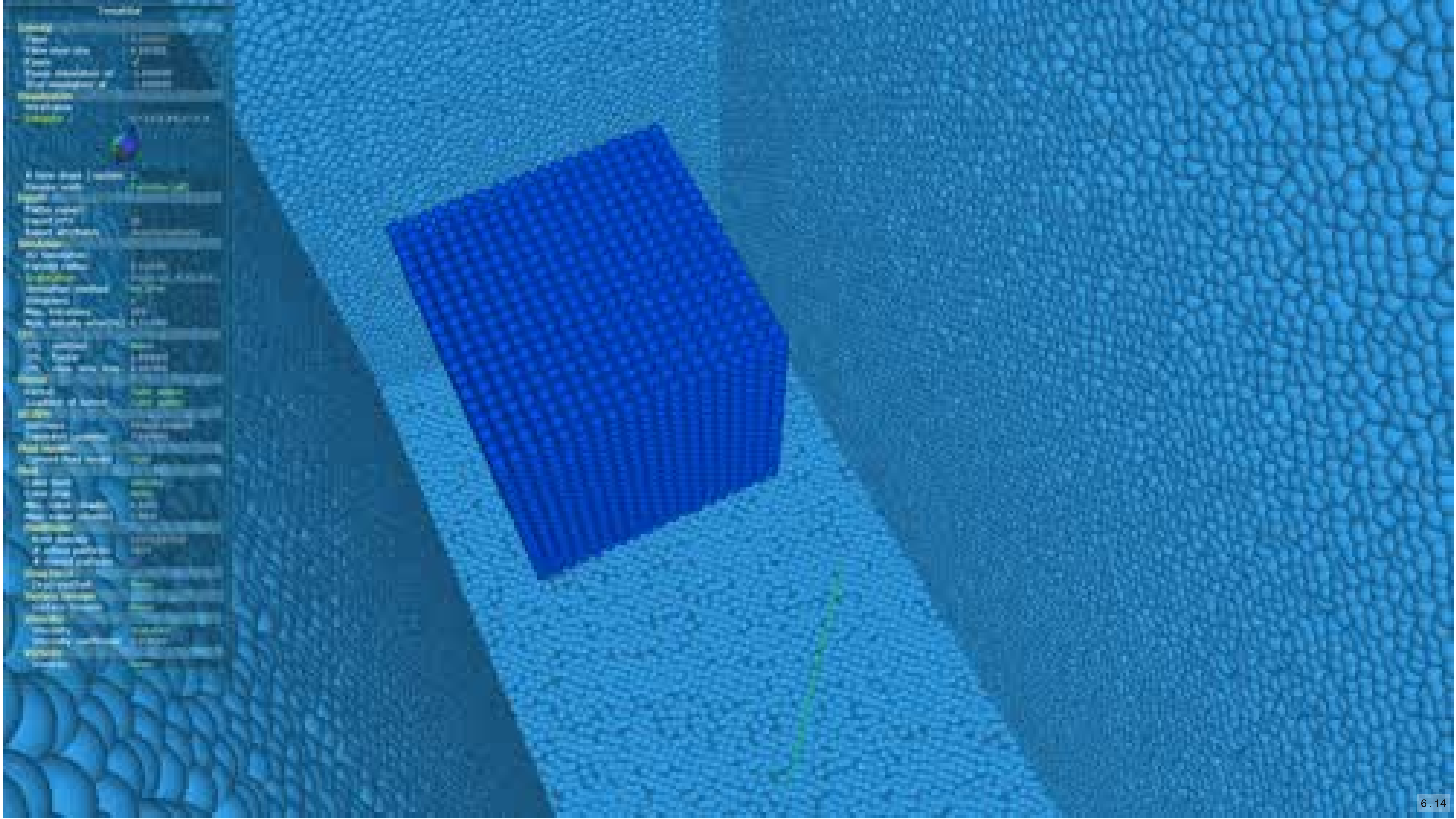
Solving - Remaining Questions



- What about boundary handling?! $\mathbf{v} \cdot \mathbf{n} \geq 0 \text{ on } \Gamma$
- Use "HACK" for now:
  - Use static fluid particles on boundary
  - Pressure force will "push" them inside

- What is a "good" time step size?
  - As large as possible for speed
  - As small as necessary for stability/accuracy
  - CFL condition as upper bound:
  - Scaling heuristically chosen $\lambda \approx 0.4$
  - Adapt time step during simulation

$$\Delta t \leq \lambda \frac{\text{particle diameter}}{\max_i \mathbf{v}_i}$$


6.14

Neighborhood Search

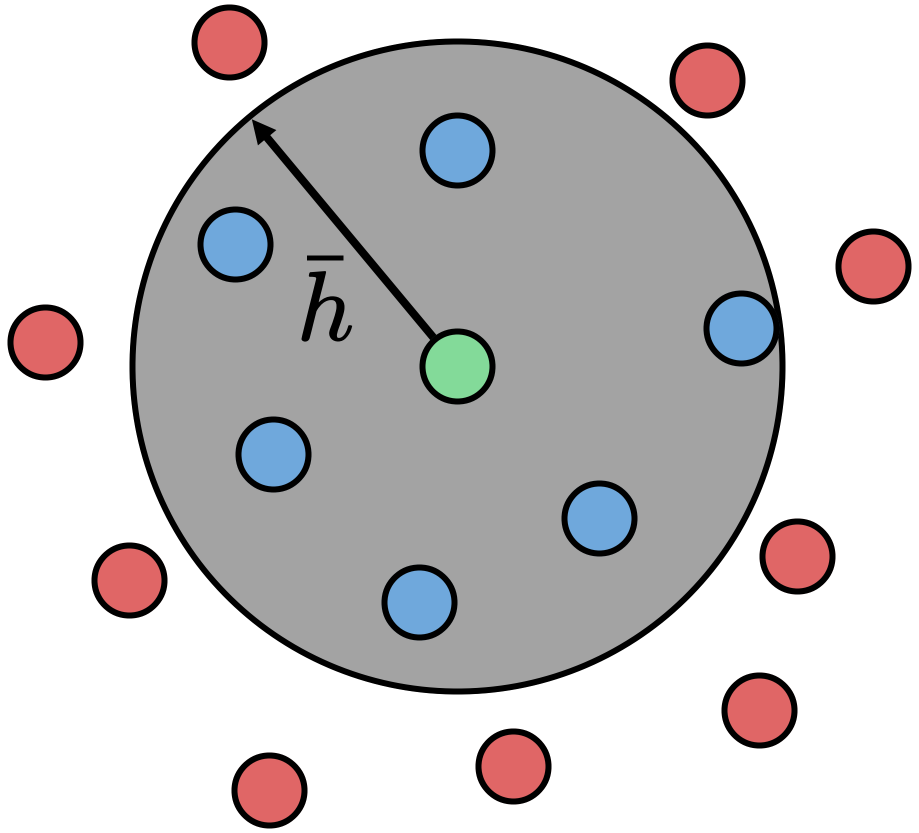
Motivation

- Lots of sums over particles of type $\sum_j (\cdot) W(\mathbf{x}_i - \mathbf{x}_j, h) \dots$
  - If computed for each particle $\Rightarrow \mathcal{O}(n^2)$
  - Can we optimize?
- Recap:
  - (Optional) Compact support kernel condition

$$W(\mathbf{r}, h) = 0 \text{ for } \|\mathbf{r}\| \geq \bar{h}$$

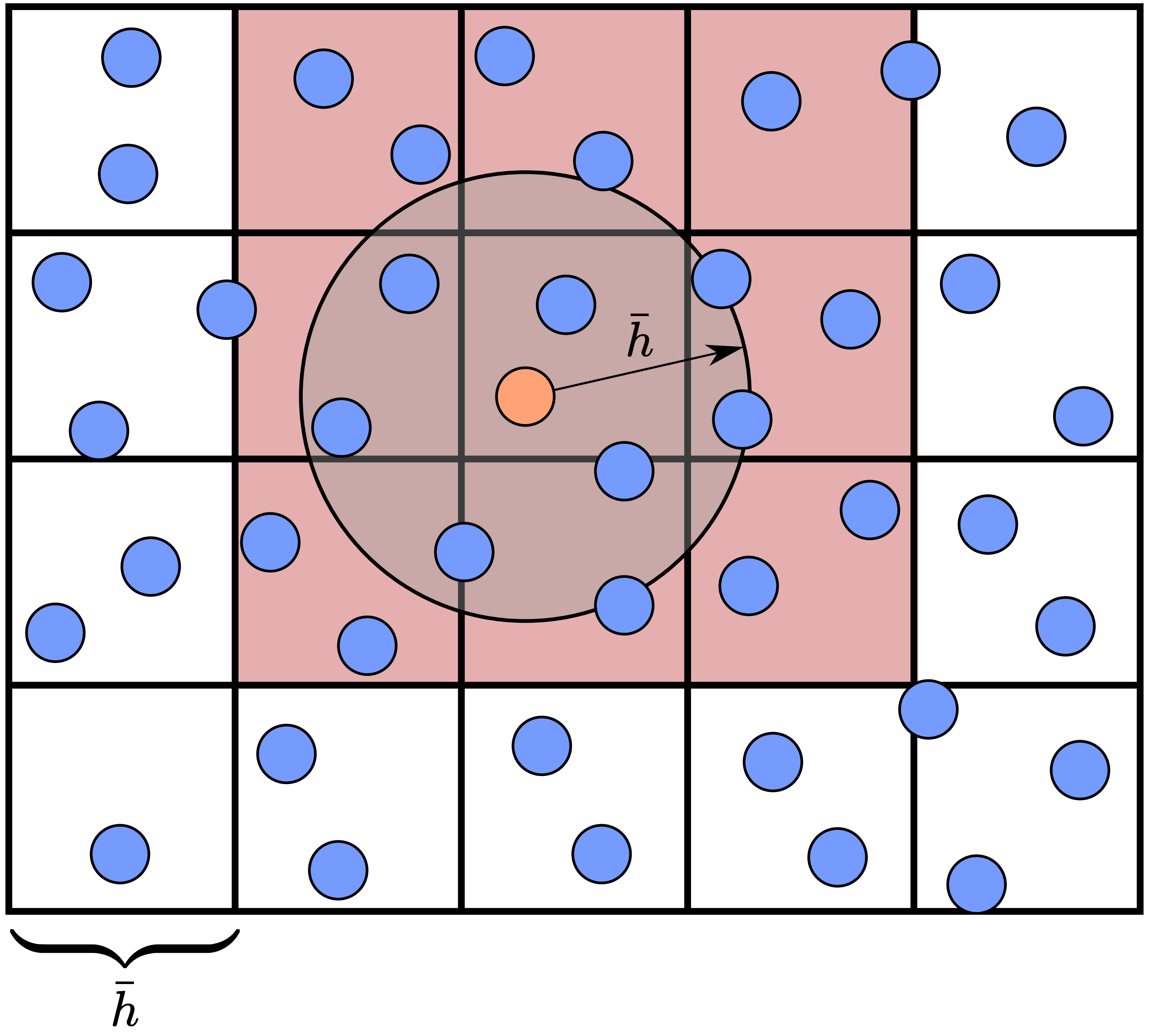
- Kernel (and derivatives) vanishes outside support radius
- Assuming a particle has on average k particles in radius $\bar{h}$
- If we know adjacencies

$\Longrightarrow$ Runtime complexity can be reduced to $\mathcal{O}(kn)$



Grid-based Neighborhood Search

- How can we efficiently find "neighbors"?
  - Compare all ? $\Rightarrow \mathcal{O}(n^2)$
- Idea: Place grid with m cells over domain
  - Choose cell size $\bar{h}$
  - Neighbors <u>must</u> lie within
    - same
    - or (26) neighboring cells
  - Sort particles into grid/find neighbors $\Rightarrow \mathcal{O}(m)$
- Many cells empty! => Memory intensive

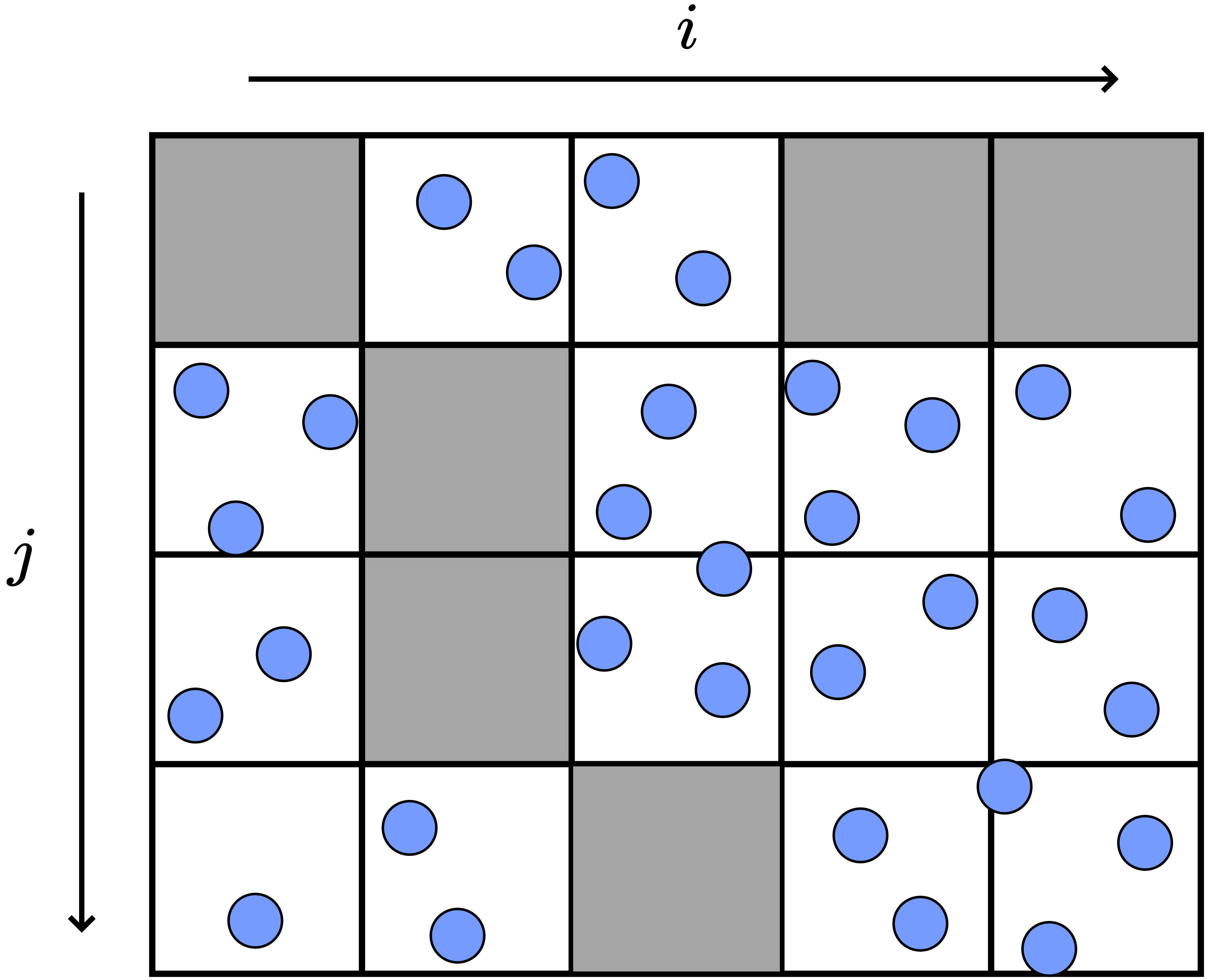


Spatial Hashing [THM*03]

- Sparse representation
  - Store only populated cells
  - Use hash map: (i,j,k) -> [p1, p2, ...]
  - Suitable hash function

$$hash(i,j,k) = [(p_1 i)\,\mathrm{XOR}\,(p_2 j)\,\mathrm{XOR}\,(p_3 k)]$$

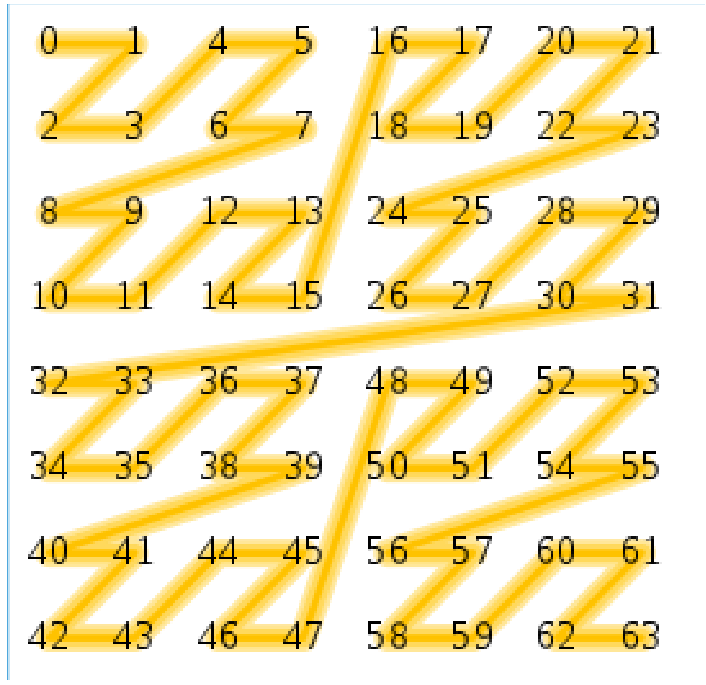
- $p_i :=$ large prime numbers



- => Much lower memory requirements
- Cache-hit rate suboptimal
  - Neighboring cells are not close in memory

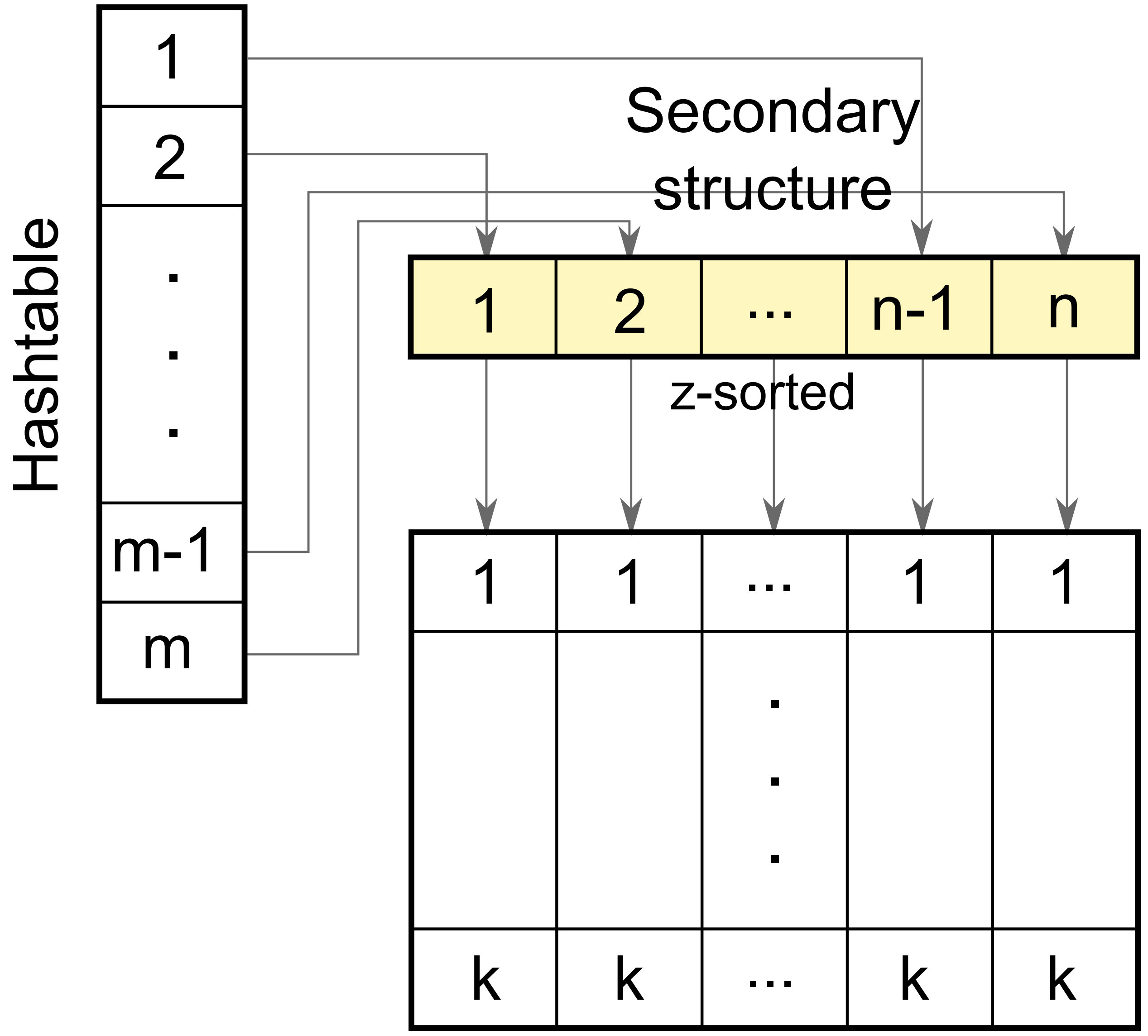
Compact Hashing [IABT11]

- Only store handles to secondary data structure in hash table
  - Sort cells in secondary structure
    - using space-filling z-curve



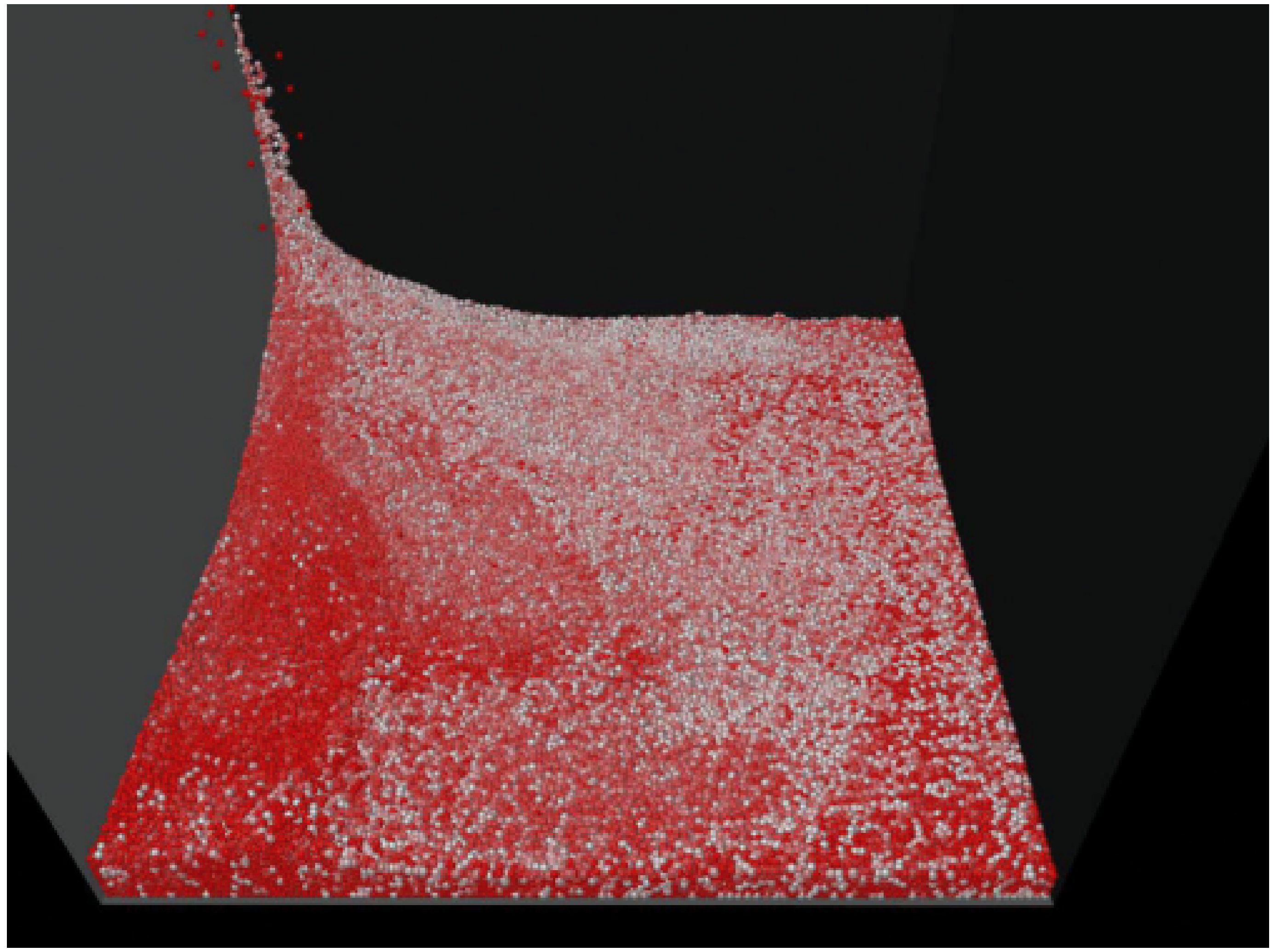
https://en.wikipedia.org/wiki/Z-order_curve

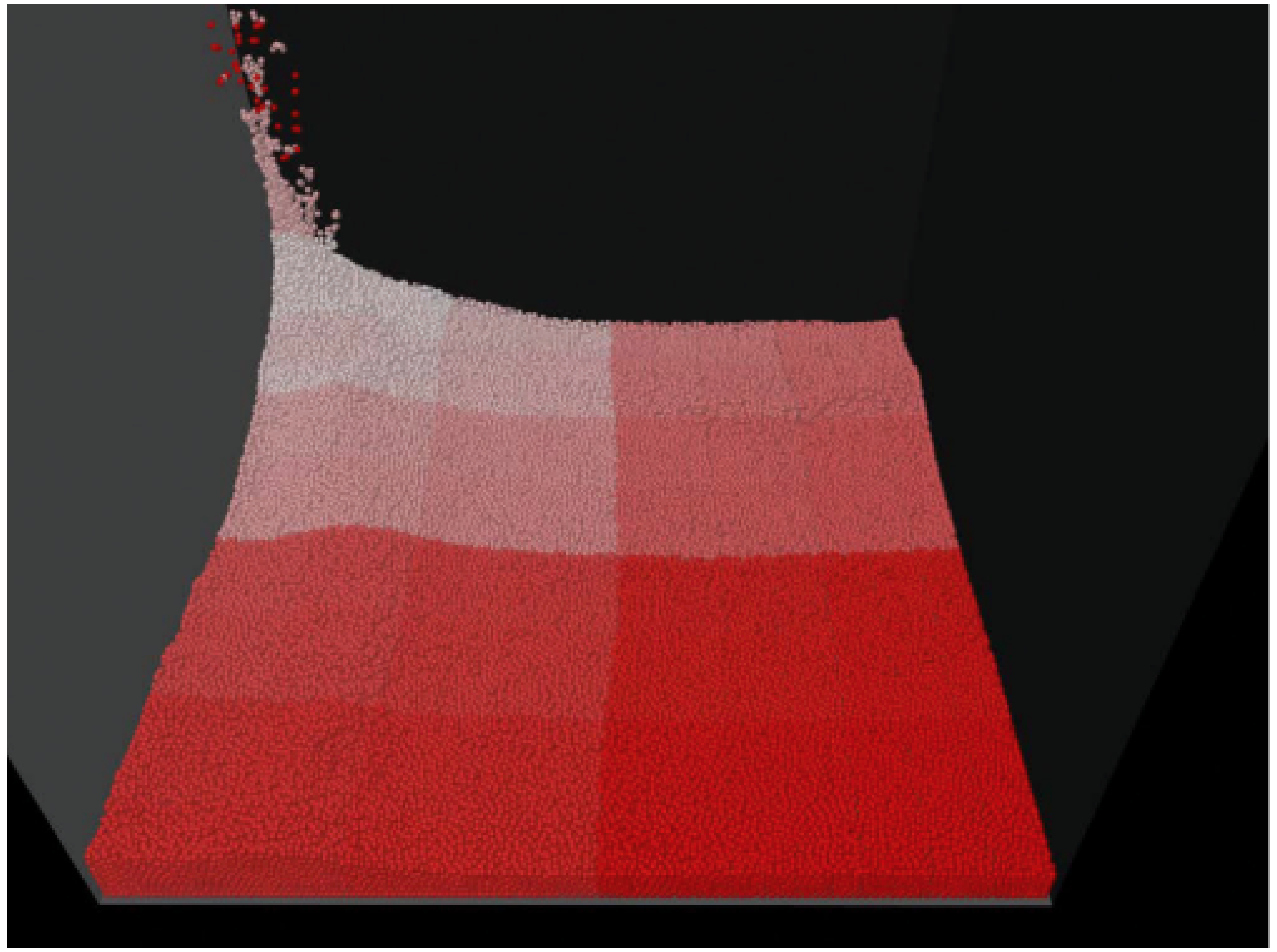
- => Higher cache efficiency



Compact Hashing [IABT11]

- Useful to sort particles along Z-curve as well
  - Sort expensive
  - Particles move "slowly"
  - Only sort every 1000th time step (or similar)





Compact Hashing [IABT11]

- Benchmark (130k particles)
- Times in ms

[IABT11]

method	construction	query	total
basic uniform grid	25.7 (27.5)	38.1 (105.6)	**63.8** (133.1)
index sort [Gre08]	35.8 (38.2)	29.1 (29.9)	**64.9** (77.3)
Z-index sort	16.5 (20.5)	26.6 (29.7)	**43.1** (50.2)
spatial hashing	41.9 (44.1)	86.0 (89.9)	**127.9** (134.0)
compact hashing	8.2 (9.4)	32.1 (55.2)	**40.3** (64.6)

Compact Hashing [IABT11]

- (Parallel) reference implementation: https://github.com/InteractiveComputerGraphics/CompactNSearch
  - supports independent point sets, activation table, z-ordering, ...

- Simple C++ implementation using STL hashmap:

```
struct SpatialHasher {
    std::size_t operator()(HashKey const &k) const {
        return 73856093 * k.k[0] ^ 19349663 * k.k[1] ^ 83492791 * k.k[2];
    }
};

class NeighborhoodSearch {
...
private:

    std::unordered_map<HashKey, CellID, SpatialHasher> m_map;
    std::vector<ParticleIDArray> m_particle_ids_per_cell;
};
```

Outline

Block 1 (9:00 - 10:30)

- Foundations of SPH
- Governing equations
- Time integration
- Example: Our first SPH solver
- Neighborhood Search

Coffee break (30min)

Block 2 (11:00 - 12:30)

- Enforcing incompressibility
  - State equation solvers
  - Implicit pressure solvers
- Boundary Handling
  - Particle-based methods
  - Implicit approaches

Lunch break (60min)

Block 3 (13:30 - 15:00)

- Multiphase fluids
- Viscosity
- Vorticity and turbulence
- Demo:

Coffee break (30min)

Block 4 (15:30 - 17:00)

- Deformable solids
- Rigid body simulation
  - Dynamics and coupling
- Data-driven/ML techniques
- Summary and conclusion