Towards An OpenFOAM-Based Framework For Meshless Simulations

Composable meshless multi-physics

Mohammed Elwardi Fadeli*, Holger Marschall - TU Darmstadt

Table of Content

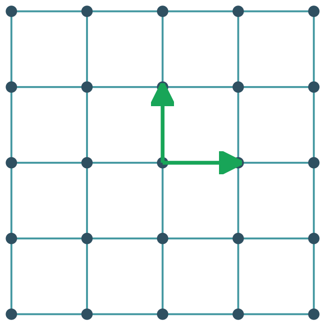
Motivating mesh-free methods for CFD

- Mesh-based methods come with some headache
  1. Excessive computational meshing costs
  2. Heavy dependence on mesh quality
  3. Some industries moving towards NURBS

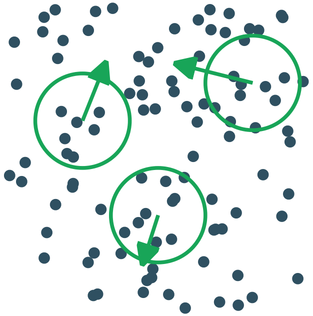
Motivating mesh-free methods for CFD

- Mesh-based methods come with some headache
  1. Excessive computational meshing costs
  2. Heavy dependence on mesh quality
  3. Some industries moving towards NURBS

- Common concerns about going meshless:
  1. Mesh connectivity eases gradient computations
  2. More sampling points required to match FVM/FEM accuracy



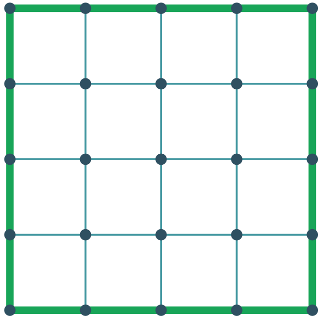
Neighbor indices are known/cached → gradients are "direct" and efficient.



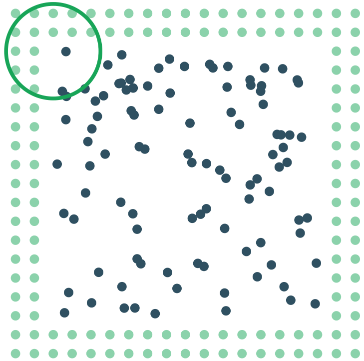
Neighbors must be searched & weighted → complex, less stable gradient estimation.

Motivating mesh-free methods for CFD

- Boundary Condition Complexity
  1. No natural "boundary faces" to apply constraints
  2. Many meshless software pieces will use ghost particles for boundary stability
  3. Or Lagrangian multipliers with penalty methods
- But this can be turned into strength when handling free-surface flows

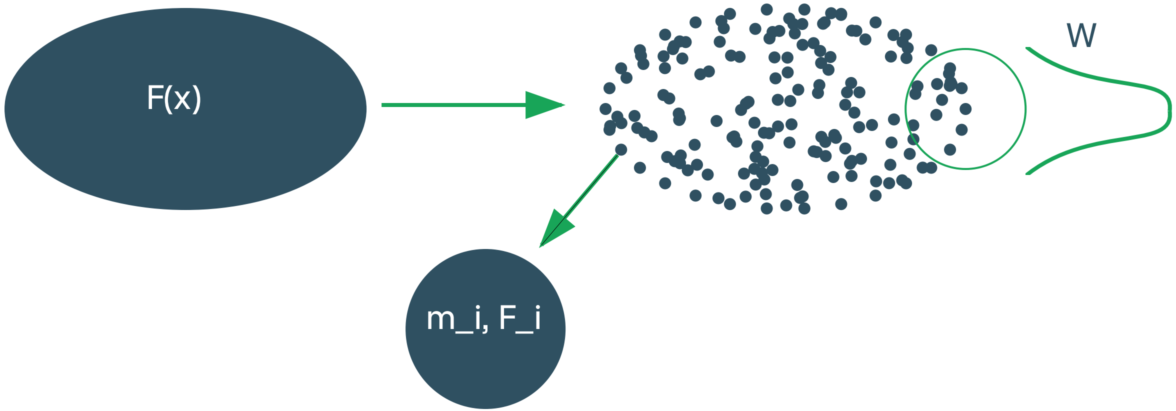


Clearly defined boundary elements → easy to apply BCs.



No natural boundary → ghost particles surround the domain to stabilize BCs.

Meshless methods 101



$$F(x_i) = \sum_j F_j \frac{m_j}{\rho_j} W_{ij}, \quad \rho(x_i) = \sum_j m_j W_{ij}$$

Approximating F derivatives shifts to the kernel function W in simple cases, although very unstable

MeshlessFlow: Framework overview

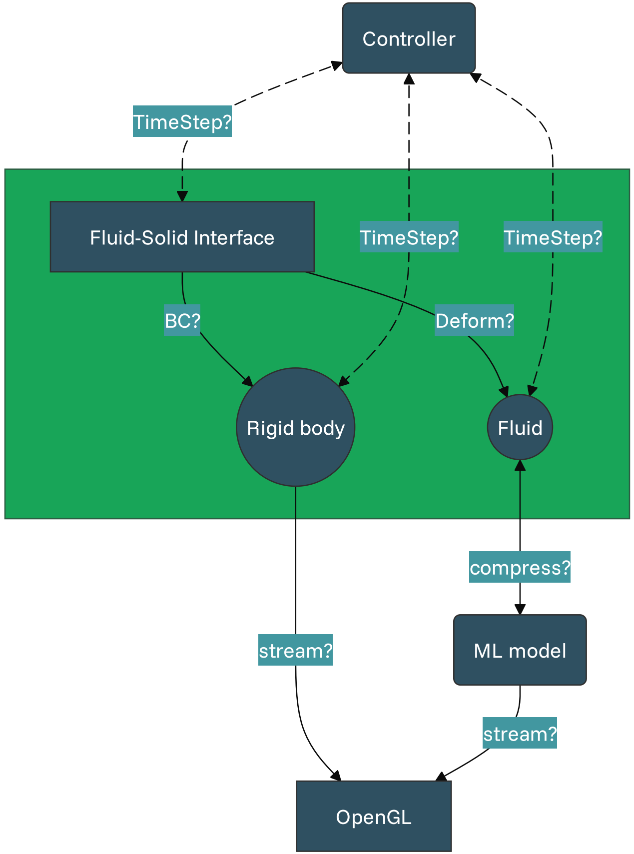
The **C**omponent-to-**C**omponent **I**nterface **P**rotocol

- Treating CFD simulations as compositions of independent components
- Each module interacts via **capabilities** negotiated through a language-agnostic protocol
  - Much like the Language Server Protocol in IDEs
  - Capabilities include:
    - MeshlessPDE: Can build and solve PDEs
    - Configurable: Communicates standard config
    - BoundingBox: Computes its bounding boxes

MeshlessFlow: Framework overview

The Component-to-Component Interface Protocol

- Treating CFD simulations as compositions of independent components
- Each module interacts via **capabilities** negotiated through a language-agnostic protocol
  - Much like the Language Server Protocol in IDEs
  - Capabilities include:
    - MeshlessPDE: Can build and solve PDEs
    - Configurable: Communicates standard config
    - BoundingBox: Computes its bounding boxes

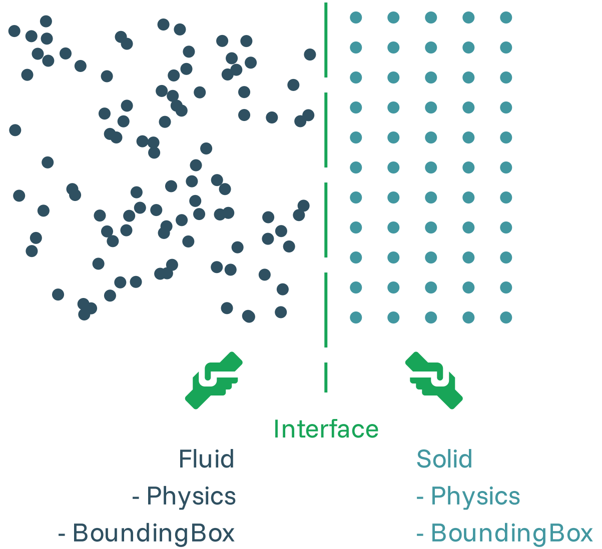


MeshlessFlow: Framework overview

The Component-to-Component Interface Protocol

- Capability-negotiation mechanism triggers at initialization phase
- A "Component" is either a process or an MPI group
- Efficient communication channels are then established (raw object streams, MPI***)
- Use what you already have! An FVM-based solver can behave as a component.

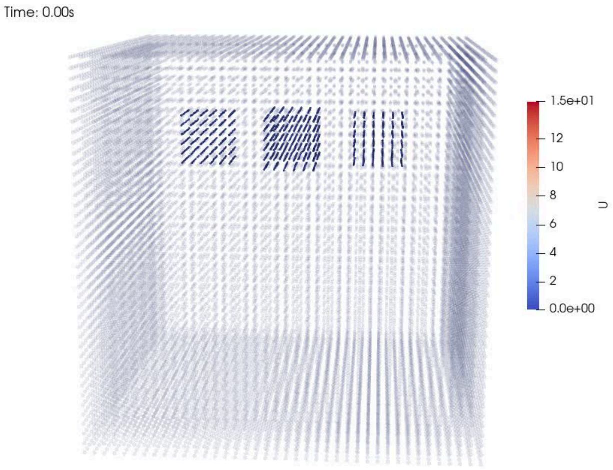
> CCIP is nothing but an abuse of the region-based methodology of multiRegionFoam [], initially designed for region-region convenient coupling



Loading citation...

Error loading citation: Unexpected token '<', "<!DOCTYPE "... is not valid JSON

A case study: Bouncing rigid bodies



- **Method:** Particle-based rigid body, using SPH infrastructure
- Contact Forces and Collisions -> Moment -> Motion
- Explicit time integration
- **Kernel Function:** Quintic Spline
- No global linear system solved

A case study: Bouncing rigid bodies

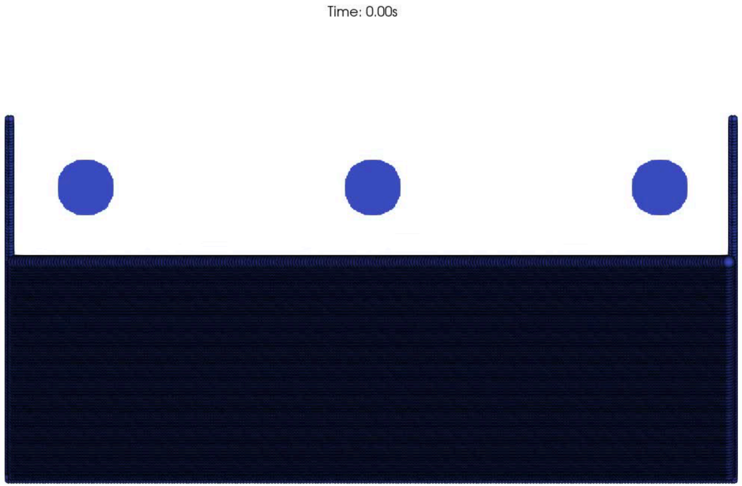
The Quintic Spline kernel

$$W(r,h) = \alpha_n \times \begin{cases} (3-\frac{r}{h})^5 - 6(2-\frac{r}{h})^5 + 15(1-\frac{r}{h})^5, & 0 \leq \frac{r}{h} < 1 \\ (3-\frac{r}{h})^5 - 6(2-\frac{r}{h})^5, & 1 \leq \frac{r}{h} < 2 \\ (3-\frac{r}{h})^5, & 2 \leq \frac{r}{h} < 3 \\ 0, & \frac{r}{h} \geq 3 \end{cases}$$

Notable properties:

- Large support radius
- Smoother pressure gradients (through C^4 continuity)
- Probably over-kill for this case…

A case study: Sinking rigid bodies (2D)

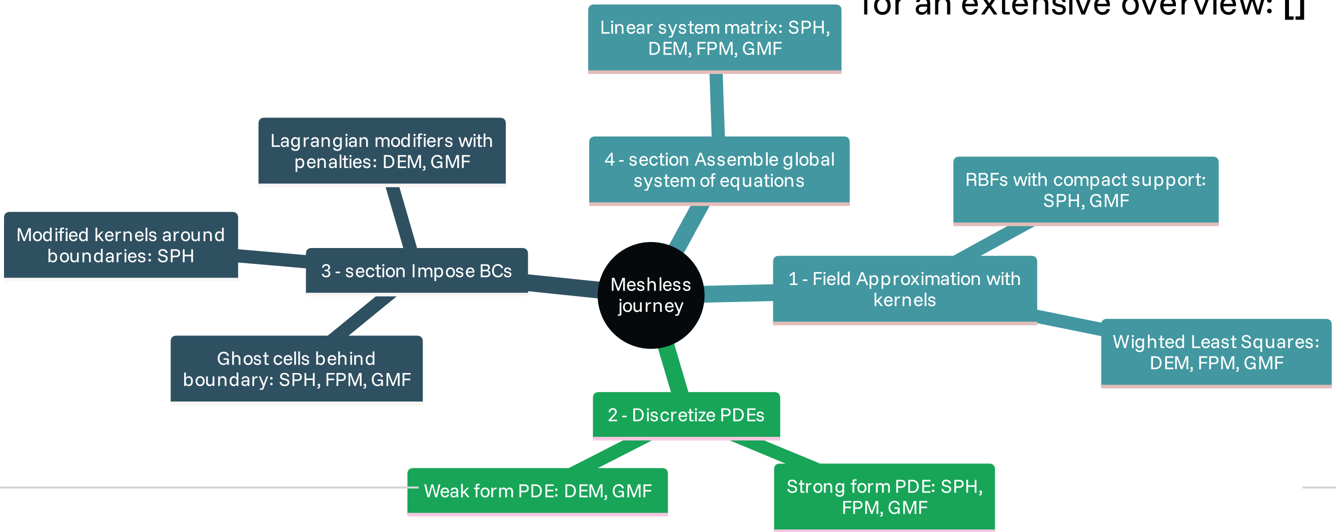


- **Method:** Weakly compressible SPH, Tait EOS, Momentum has artificial viscosity
- Coupling interface: Akinci Rigid-body Fluid coupling
- Explicit time integration
- **Kernel Function:** Quintic Spline
- No global linear system solved

Is it only about SPH?

Current implementation status for different meshless methods in MeshlessFlow

Strong form solutions mostly follow **[]**;
for an extensive overview: **[]**



Loading citation...

Error loading citation: Unexpected token '<', "<!DOCTYPE "... is not valid JSON

Loading citation...

Error loading citation: Unexpected token '<', "<!DOCTYPE "... is not valid JSON

What's next?

- More SPHERIC Grand challenges: Established validation cases
- Full Parallelization
- User Interface
- Coupling with OpenFOAM FVM-based solvers

Thank you for your attention

GitHub · More stuff like this

Technical notes

Loading citation...

Most important framework features:

Error loading citation: Unexpected token '<', "<!DOCTYPE "... is not valid JSON

Highly configurable, easily testable code (Shape discretization from **[]**)

```
// Type-agnostic, gets standard shape config, 0-runtime-cost
auto geoConfig = Reflect::schema<shape>("STLShape"); // STLShape is a "shape"
// Do stuff with geoConfig: GUI? LLM?
auto geo = shape::New(geoConfig);
// Works with more construction args too
auto discConfig = Reflect::schema<discretization>("Slak2019"); // STLShape is a "shape"
auto disc = discretization::New(discConfig, shape);
```

Symbolic assembly of matrices

```
// only a "meta" assembly
auto eq = exp::div(phi) + laplacian(D, phi) + src
// physics component is free to solve it,
// or send the meta form to other components
eq.assembleMatrix(disc).solve()
```