Towards An OpenFOAM-Based Framework For Meshless Simulations

Composable meshless multi-physics

Mohammed Elwardi Fadeli*, Holger Marschall - TU Darmstadt

Table of Content

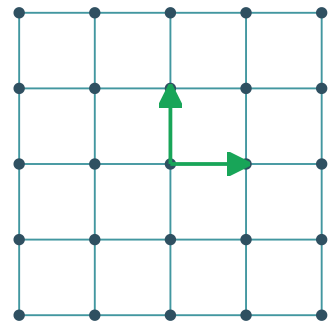
Motivating mesh-free methods for CFD

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

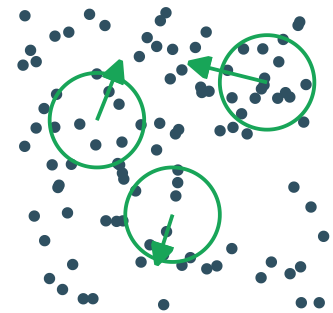
Motivating mesh-free methods for CFD

- 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



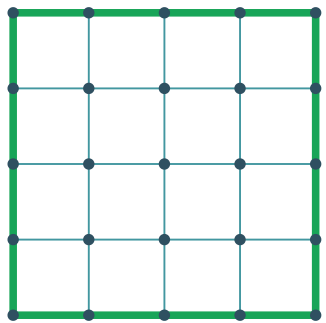
Neighbor indices are known/cached → gradients are "direct" and efficient.



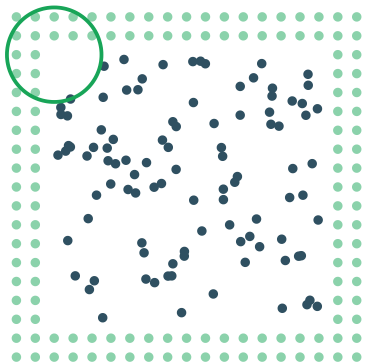
Neighbors must be searched & weighted → complex, less stable gradient estimation.

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

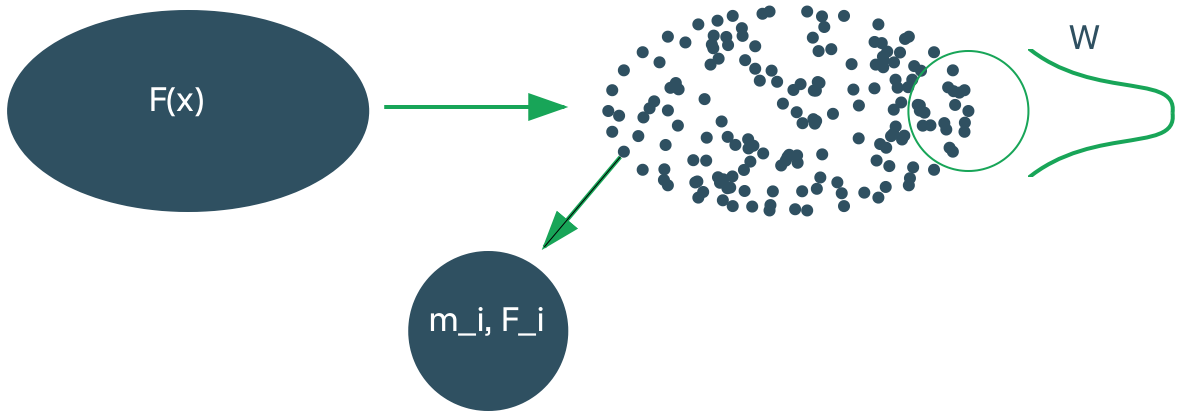


Clearly defined boundary elements → easy to apply BCs.



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

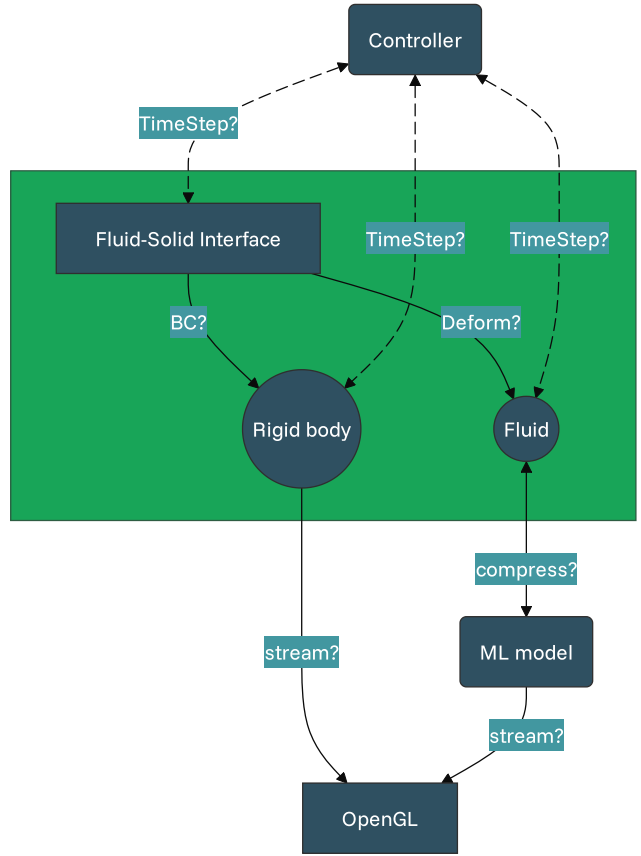
The **C**omponent-to-**C**omponent **I**nterface **P**rotocol

- 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 **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

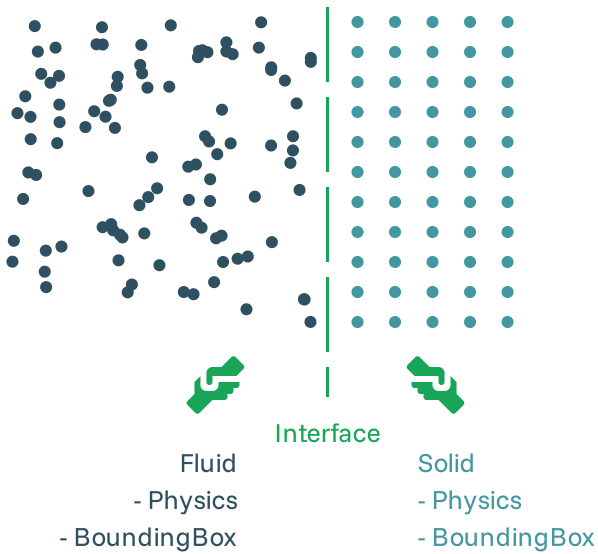


MeshlessFlow: Framework overview

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

- 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 **[Alkafri, 2024]**, initially designed for region-region convenient coupling



[Alkafri, 2024] Alkafri, H., Habes, C., Fadeli, M. E., Hess, S., Beale, S. B., Zhang, S., Jasak, H., & Marschall, H. (2024). multiRegionFoam: a unified multiphysics framework for multi-region coupled continuum-physical problems. Engineering with Computers, 41(2), 1051–1084. https://doi.org/10.1007/s00366-024-01974-4

A case study, striding on water

By Turnstone Videos.
Strider locomotion has been described by **[Hu, 2003]**,

[Hu, 2003] Hu, D. L., Chan, B., & Bush, J. W. M. (2003). The hydrodynamics of water strider locomotion. Nature, 424(6949), 663–666. https://doi.org/10.1038/nature01793

A case study, striding on water

Fluid region, with Weakly Compressible SPH

- EOS: $p = p_0 + B((\frac{\rho}{\rho_0})^\gamma - 1)$
- Navier-Stocks for density and momentum
- Equations are integrated and solved for; separately from other components
- Only the maximal `TimeStep length` is dictated by a controller component

A case study, striding on water

A few words on some WCSPH properties

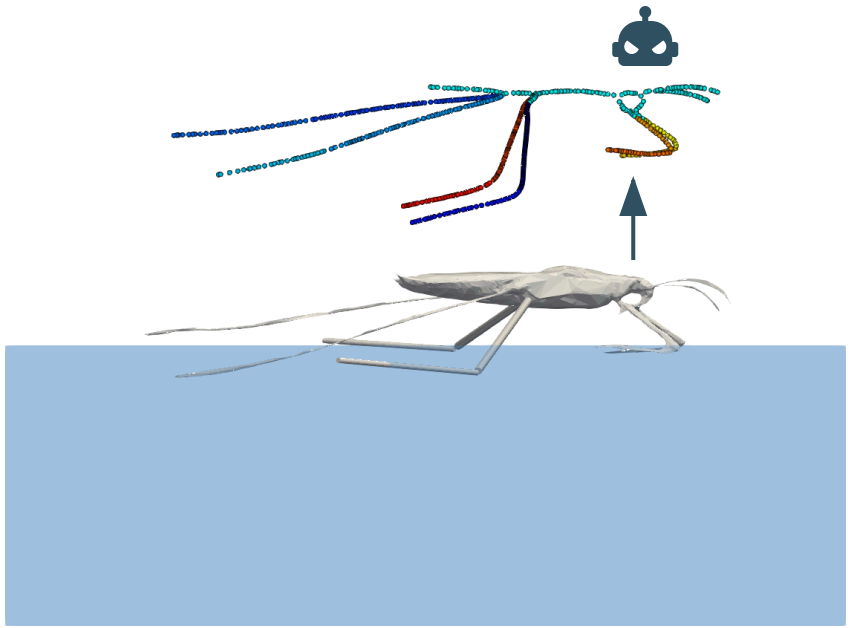
The Quintic Spline kernel:

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

Notable properties:

- Large support radius
- Smoother pressure gradients (through C^4 continuity)

A case study, striding on water



Rigid body moments and motions are integrated with an RK2 algorithm.

Using a **skeletonized** strider body to ease controlling the strider through a behavioral tree algorithm.

- Animate middle legs following a prescribed motion.
- With a prescribed velocity, that mimics a realistic strider stroke.

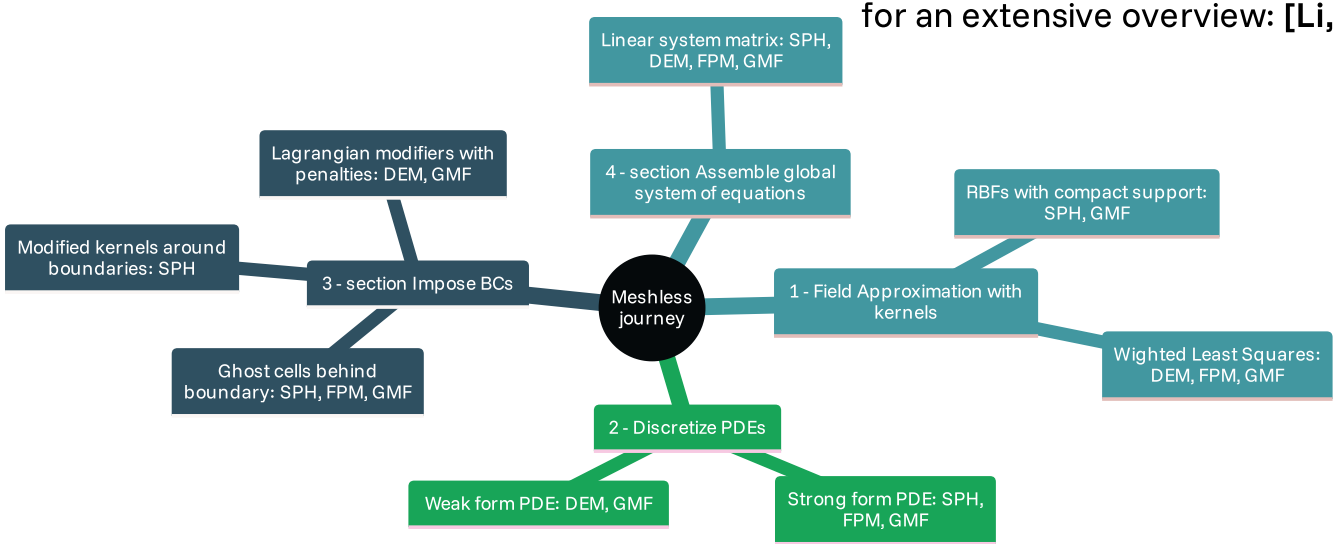
An **FSI interface** component

- Applies hydrophobic properties to strider legs
- Transmits fluid forces on strider body to the skeleton (still in development)

Is it only about SPH?

Current implementation status for different meshless methods in MeshlessFlow

Strong form solutions mostly follow **[Slak, 2021]**; for an extensive overview: **[Li, 2013]**



[Slak, 2021] Slak, J., & Kosec, G. (2021). Medusa: a C++ library for solving PDEs using strong form mesh-free methods. ACM Transactions on Mathematical Software, 47(3), 1–25. https://doi.org/10.1145/3450966

[Li, 2013] Li, H., & Mulay, S. S. (2013). Meshless methods and their numerical properties. CRC Press. https://doi.org/10.1201/b14492

Thank you for your attention

GitHub · More stuff like this

Technical notes

Most important framework features:

[Slak, 2019] Slak, J., & Kosec, G. (2019). On generation of node distributions for meshless PDE discretizations. SIAM Journal on Scientific Computing, 41(5), A3202–A3229. https://doi.org/10.1137/18m1231456

Highly configurable, easily testable code (Shape discretization from **[Slak, 2019]**)

```
// 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()
```