Cemosis, IRMA
Université de Strasbourg

Mathematical and computational frameworks for simulating and optimizing micro-swimmers

18th May 2021

Luca Berti, Vincent Chabannes, Laetitia Giraldi(Inria), Christophe Prud'homme

Some Projects at Cemosis

- **IBat** with Synapse-Concept, Febus Ingenierie / Zohra, Vincent, Romain, Yannick
- **Mordicus** FUI with EDF, Safran, Phimeca, Sorbonne U, ... / Vincent, Romain, Christophe Trphime
- **Eye2brain** with U Paris, U Missouri and Inria / Romain, Vincent, Philippe and Thomas
- **Hifimagnet** with LNCMI / Christophe T, Vincent, Romain V. and Romain
- **Swimmer** with Inria Sophia / Laetitia, Luca and Vincent

Extend, capitalize, reuse the work done over the years on Feel++

Feel++

- A large range of numerical methods to solve (parametrized) partial differential equations: cG, dG, hdG, crb, ... in 1D, 2D and 3D
- Powerful interpolation tools in 2D/3D
- Support for high performance computing up to thousands of cores for linear and non-linear problems using PETSc/SLEPc and InHouse solution strategies. (MPI + MT, HPC and Cloud)
- Various toolboxes that capitalize our work over the years: CFD, CSM, Heat, Electric, Maxwell and the coupling between them FSI(ALE and LS), heat and fluid, thermoelectric ...
- New toolbox CFPDEs: arbitrary number of (coupled) linear and non-linear possibly time-dependent PDEs, automatic differentiation, DSL to handle PDE description

Outline

Micro-swimming framework in Feel++

Mesh adaptation

Examples of swimmers

Perspectives

Micro-swimming framework in Feel++

Mathematical modelling

The swimming problem we consider is defined in $\mathcal{F}_t \subseteq \mathbb{R}^d$

$$\begin{cases} -\mu\Delta\mathbf{u} + \nabla p = f, & \text{in } \mathcal{F}_t \\ \nabla \cdot \mathbf{u} = 0, & \text{in } \mathcal{F}_t \\ \mathbf{u} = U + \Omega \times (x - x^{CM}(t)) + u_d(t) \circ \mathcal{A}_t^{-1}, & \text{on } \partial S_t \\ m\dot{U} = F_{ext} - F_{fluid}, \\ J\dot{\Omega} = M_{ext} - M_{fluid}. \end{cases}$$

- the swimmer S is completely immersed in fluid
- U and Ω are the translational and angular velocities of the body
- u_d is the time derivative of the body's displacement
- $\mathcal{A}_t$ is the map $\mathcal{F}_0 \to \mathcal{F}_t$, $x \mapsto x + \phi(x)$, where ϕ is an extension of the boundary displacement

Swimmer motion in the fluid system

$$\begin{cases} -\mu\Delta u + \nabla p = 0, & \text{in } \mathcal{F}_t, \\ \nabla \cdot u = 0, & \text{in } \mathcal{F}_t, \\ u = U + \Omega \times (x - x^{CM}) + u_d \circ \mathcal{A}_t^{-1}, & \text{on } \partial\mathcal{F}_t \cap \partial S, \\ m\dot{U} = F = F_{ext} - F_{fluid}, \\ J\dot{\Omega} = M = M_{ext} - M_{fluid}. \end{cases}$$

Weak formulation $\rightarrow$ test functions $(\tilde{u}, \tilde{p}, \tilde{U}, \tilde{\Omega})$ such that $\tilde{u} = \tilde{U} + \tilde{\Omega} \times (x - x^{CM})$ on ∂S

$$2\mu \int_{\mathcal{F}} D(u) : D(\tilde{u})\, dx - \int_{\mathcal{F}} p \nabla \cdot \tilde{u}\, dx + mU \cdot \tilde{U} + J\Omega \cdot \tilde{\Omega} = F_{ext} \cdot \tilde{U} + M_{ext} \cdot \tilde{\Omega}.$$

B. Maury, *Direct Simulations of 2D Fluid-Particle Flows in Biperiodic Domains*. Journal of Computational Physics, 156, 1999.

Numerical techniques

- spatial discretization $\rightarrow$ conforming Lagrange finite elements
- moving fluid domain $\rightarrow$ Arbitrary-Lagrangian-Eulerian frame
- rigid body velocity $\rightarrow$ constrain fluid velocity $\mathbf{u}$ to the subspace

$$V_R = \{\mathbf{u} \in [H^1(\mathcal{F})]^d \,|\, \mathbf{u}|_{\partial S} = U + \Omega \times (x - x^{CM}) \text{ for } U, \Omega \in \mathbb{R}^d\}$$

- mesh adaptation $\rightarrow$ harmonic extension and remeshing

Algebraic form (I)

We need to enforce the condition $\tilde{u} = \tilde{U} + \tilde{\Omega} \times (x - x^{CM})$ **on** ∂S on the test functions.

- We build the system matrix A. No coupling for the moment is enforced between fluid and swimmer.

$$A = \begin{bmatrix} A_{II} & A_{I\Gamma} & 0 & 0 & B_I^T \\ A_{\Gamma I} & A_{\Gamma\Gamma} & 0 & 0 & B_\Gamma^T \\ 0 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & R & 0 \\ B_I & B_\Gamma & 0 & 0 & 0 \end{bmatrix}$$

- We build a coupling matrix $\mathcal{P}$ such that $(u_I, u_{\partial S}, U, \Omega, p)^T = \mathcal{P}(u_I, u_{\partial S}, U, \Omega, p)^T$

$$\mathcal{P} = \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & 0 & \tilde{P}_U & \tilde{P}_\Omega & 0 \\ 0 & 0 & I & 0 & 0 \end{bmatrix}$$

Algebraic form (II)

$$\mathcal{P}^T \begin{bmatrix} A_{II} & A_{I\Gamma} & 0 & 0 & B_I^T \\ A_{\Gamma I} & A_{\Gamma\Gamma} & 0 & 0 & B_\Gamma^T \\ 0 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & R & 0 \\ B_I & B_\Gamma & 0 & 0 & 0 \end{bmatrix} \mathcal{P}$$

$$= \begin{bmatrix} A_{II} & \mathbf{0} & A_{I\Gamma}\tilde{P}_U & A_{I\Gamma}\tilde{P}_\Omega & B_I^T \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \tilde{P}_U^T A_{\Gamma I} & \mathbf{0} & \tilde{P}_U^T A_{\Gamma\Gamma}\tilde{P}_U + T & \tilde{P}_U^T A_{\Gamma\Gamma}\tilde{P}_\Omega & \tilde{P}_U^T B_\Gamma^T \\ \tilde{P}_\Omega^T A_{\Gamma I} & \mathbf{0} & \tilde{P}_\Omega^T A_{\Gamma\Gamma}\tilde{P}_U & \tilde{P}_\Omega^T A_{\Gamma\Gamma}\tilde{P}_\Omega + R & \tilde{P}_\Omega^T B_\Gamma^T \\ B_I & \mathbf{0} & B_\Gamma\tilde{P}_U & B_\Gamma\tilde{P}_\Omega & 0 \end{bmatrix}.$$

Comments on the resulting system

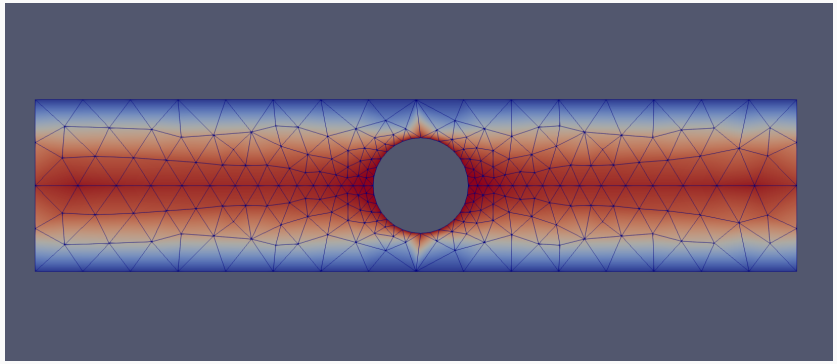
Solve

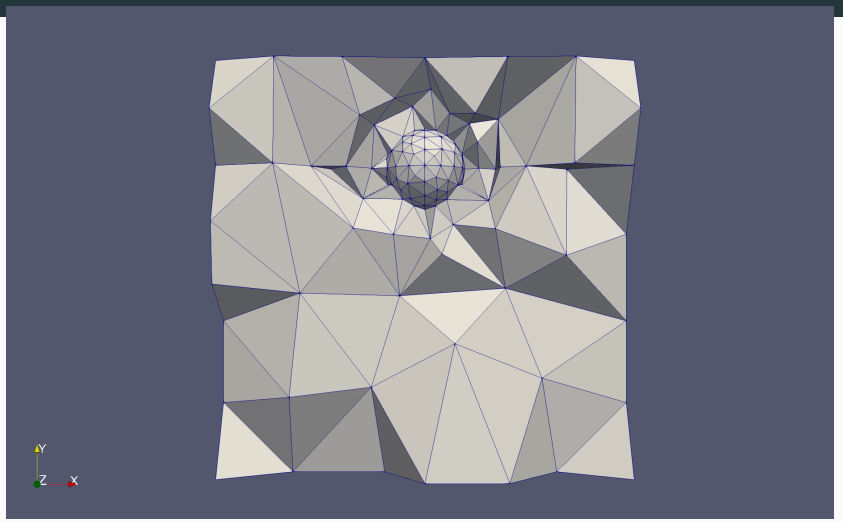
$$\mathcal{P}^T A \begin{bmatrix} u_I \\ u_{\partial S} \\ U \\ \Omega \\ p \end{bmatrix} = \mathcal{P}^T A \mathcal{P} \begin{bmatrix} u_I \\ u_{\partial S} \\ U \\ \Omega \\ p \end{bmatrix} = \mathcal{P}^T \begin{bmatrix} F_I \\ 0 \\ F_{\text{ext}} \\ M_{\text{ext}} \\ 0 \end{bmatrix} - \mathcal{P}^T A \begin{bmatrix} 0 \\ u_d \circ \mathcal{A}_t^{-1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{1}$$

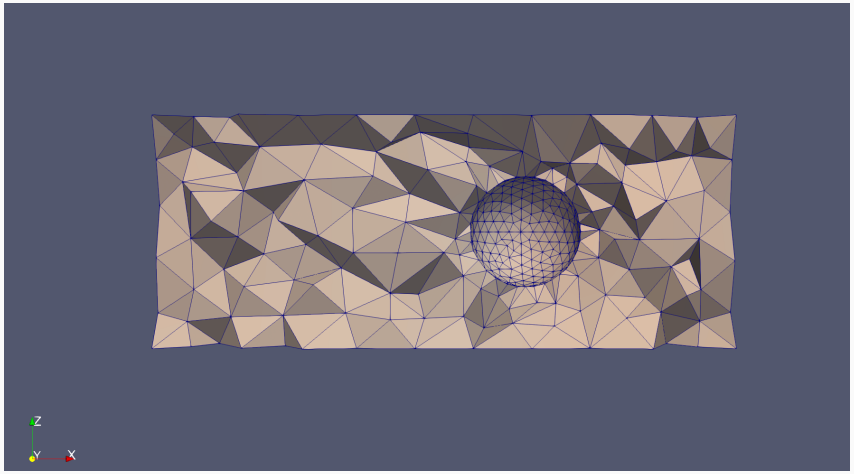
- Requires efficient implementation of $\mathcal{P}^T A \mathcal{P}$ specially in parallel. The matrix is actually assembled.
- Use a block preconditioner of type PCD or PMM depending on the regime by grouping together velocity unknowns.
- Velocity blocks are split further between fluid velocity and rigid body velocities and use an (inexact) gauss seidel preconditioner

Mesh adaptation

Remeshing in Feel++ using (Par)MMG



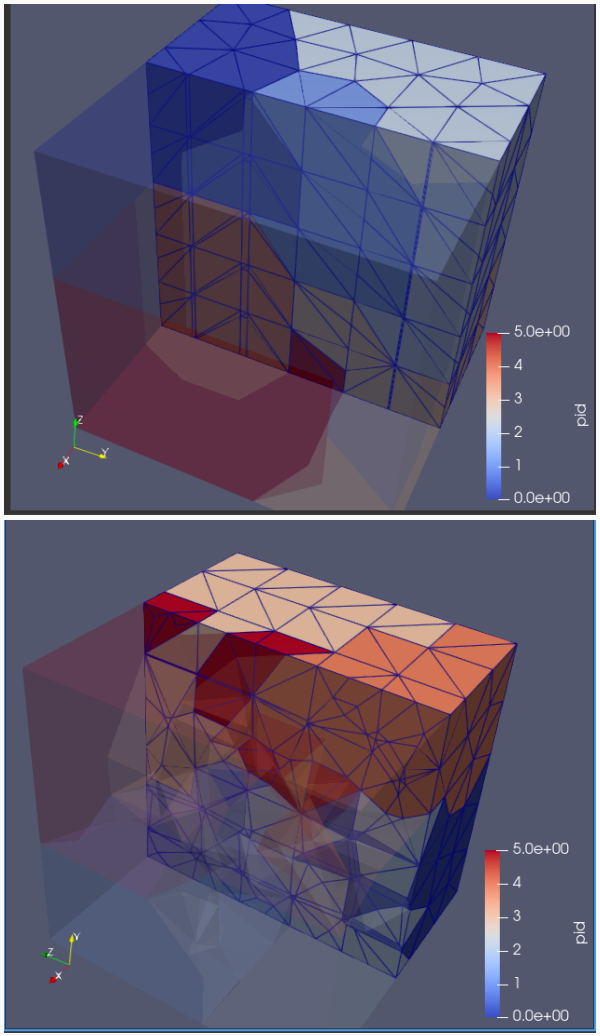




- Edge split
- Edge collapse
- Edge swap
- Node relocation
- Local size function h
- lots of ways to control the metric

Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and

Parallel remeshing



- Parallel remeshing is a challenge and MMG provides an initial version last Autumn
- Iterative strategy: remesh in subdomains while keeping interprocess faces fixed and iterate after repartitioning.

Remeshing metric

$$h_{new}(x) = \begin{cases} h_{avg} & \text{if } dist(x, \partial S_t) \leq nh_{avg} \\ h_{avg} * c_{\text{far}} & \text{if } dist(x, \partial S_t) > nh_{avg} \end{cases}$$

where

- $h_{new}(x)$ is the mesh size passed to MMG, other recipes can be applied depending on context
- h_{avg} is the average mesh size of the swimmer
- $dist$ is the distance to the swimmer computed using a (parallel) fast marching algorithm
- nh_{avg} provides a distance threshold away from the swimmer
- c_{far} is a coefficient greater than 1 to get larger elements far away
- Gradation mechanism of MMG smoothes out the edge length increase

Parent-son relationship for meshes

To evaluate expressions $u_d \circ \mathcal{A}_t^{-1}$ in the swimmer's reference frame, the initial reference mesh must be stored.

- define the elements/faces that remain unvaried across remeshing
- create a bijective correspondence between these elements/faces in the two meshes
- evaluation of expressions is now realized on the parent mesh
- there is no localization at all necessary in the interpolation process.
- This is implemented thanks to the required element feature of MMG with a twist to build the relation.

Remeshing and interpolation

Using ALE to handle moving domains is possible under the condition that movements are not too large.

Available mesh quality indices

$$q_{2D} = \frac{4A^2}{|e_1||e_2||e_3|2p}, \; q_{2D-BX} = \frac{4\sqrt{3}A}{|e_1|^2 + |e_2|^2 + |e_3|^2}, \; q_{3D} = \frac{2^3 3^8 \sqrt{3} V^4}{(\sum_{i=1}^{4} A_1^2)^3}$$

We constrain:

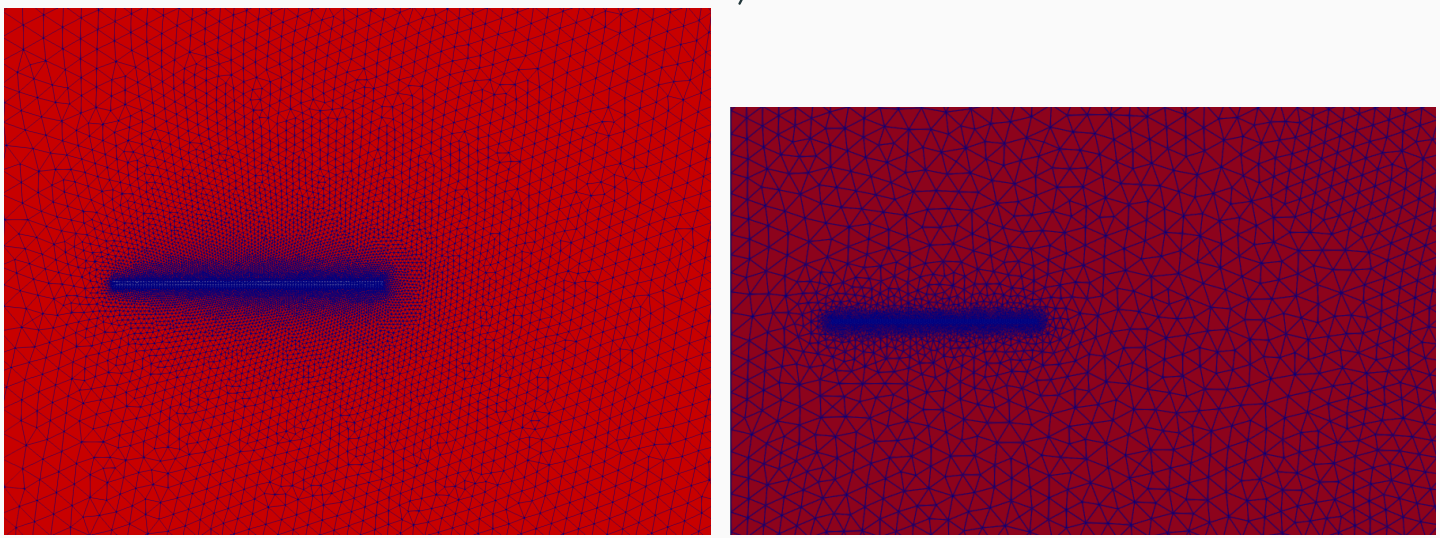
- Interfaces, that must be kept,
- Interface discretization, that should be remain unvaried as well (in order to maintain all the properties linked to areas and volumes of the enclosed region),

We interpolate the solution from the old to the new mesh, in order to restart computations.

Mesh adaptation for swimming simulations

We individuated three remeshing procedures that corresponded to different needs of our model swimmers:

- Initial mesh size distribution $\neq$ Desired mesh size distribution

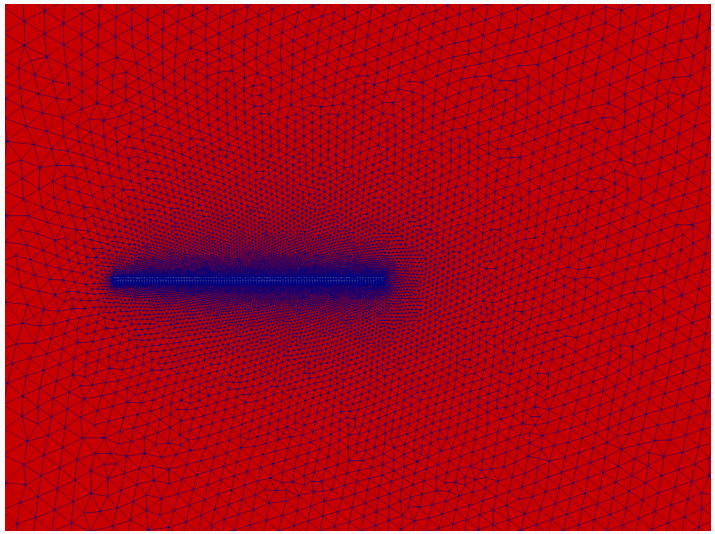


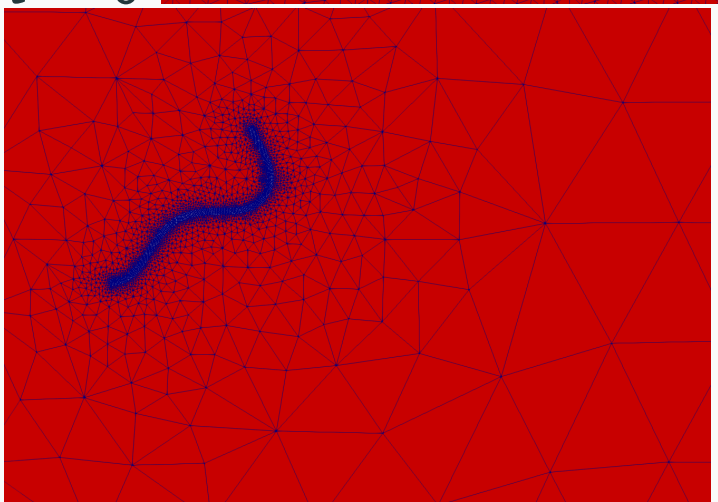
  - Define the mesh size distribution in the fluid
  - Remesh by keeping fixed the elements in the swimmer

Mesh adaptation for swimming simulations

We individuated three remeshing procedures that corresponded to different needs of our model swimmers:

- Reference swimmer configuration $\neq$ Swimmer configuration at $t = 0$

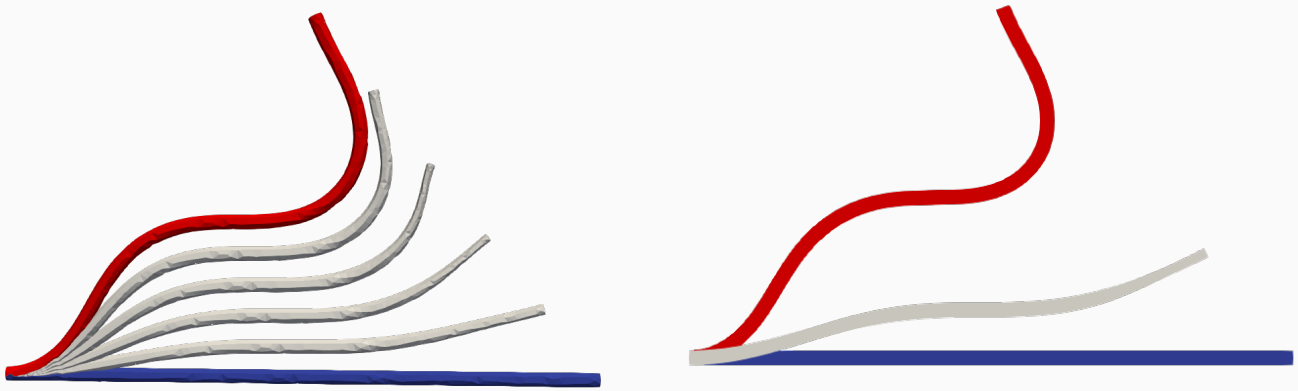




Mesh adaptation for swimming simulations

We individuated three remeshing procedures that corresponded to different needs of our model swimmers:

- Reaching the desired configuration by sub-steps (3d and 2d)



  - Given the displacement η_{n+1}, fraction it in sub-steps
  - Alternate harmonic extensions and remeshing steps until the position at $t = t_{n+1}$ is reached

Examples of swimmers

Articulated swimmers

Figure 1: Example of articulated swimmers with translational constraints

- Analytical computations and benchmarking
- Slight modification of independent rigid bodies formulation

Procedure:

- Identify B_n as the reference body
- $\mathbf{U}_i$ of all the other bodies B_i, $i = 1 \dots n-1$, are expressed as functions of $\mathbf{U}_n$ via constraints of the form

$$\mathbf{U}_i = \mathbf{U}_n + \mathbf{W}_{in}(t), \quad i = 1 \dots n-1,$$

where $\mathbf{W}_{in}(t)$ represents the relative velocity between B_i and B_n.

Translational constraints - Lagrange multipliers

Lagrange multipliers:

$$\dot{\mathbf{U}}_i \cdot \tilde{\mathbf{U}}_i + \alpha_i \cdot \tilde{\mathbf{U}}_i = 0 \quad i = 1 \dots n-1,$$

$$\dot{\mathbf{U}}_n \cdot \tilde{\mathbf{U}}_n - \sum_{i=1}^{n-1} \alpha_i \cdot \tilde{\mathbf{U}}_n = 0$$

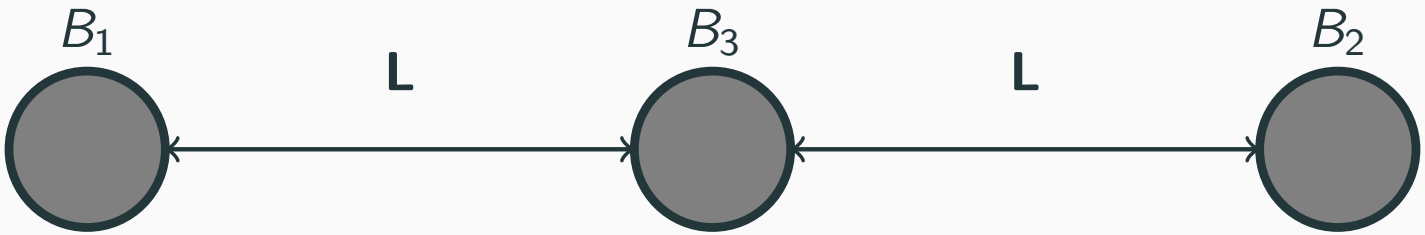
$$\alpha_i \cdot (\mathbf{U}_i - \mathbf{U}_n) = \alpha_i \cdot \mathbf{W}_{in}, \quad i = 1 \dots n-1.$$

The addition of Lagrange multipliers entails further constraints on the space V_R, which are reflected in the coupling operator between $\mathbf{u}$ and $\mathbf{U}_i$.

Three sphere swimmer

It is made of three sphere of radius R. The lateral spheres are separated from the central one by a distance of L.

This object swims by retracting and elongating its arms in a non-reciprocal fashion.



Analytical results describing its motion are available. [1]

[1] A. NAJAFI AND R. GOLESTANIAN, Simple swimmer at low reynolds number: Three linked spheres, Physical review. E, Statistical, nonlinear, and soft matter physics, 69 (2004)

Results

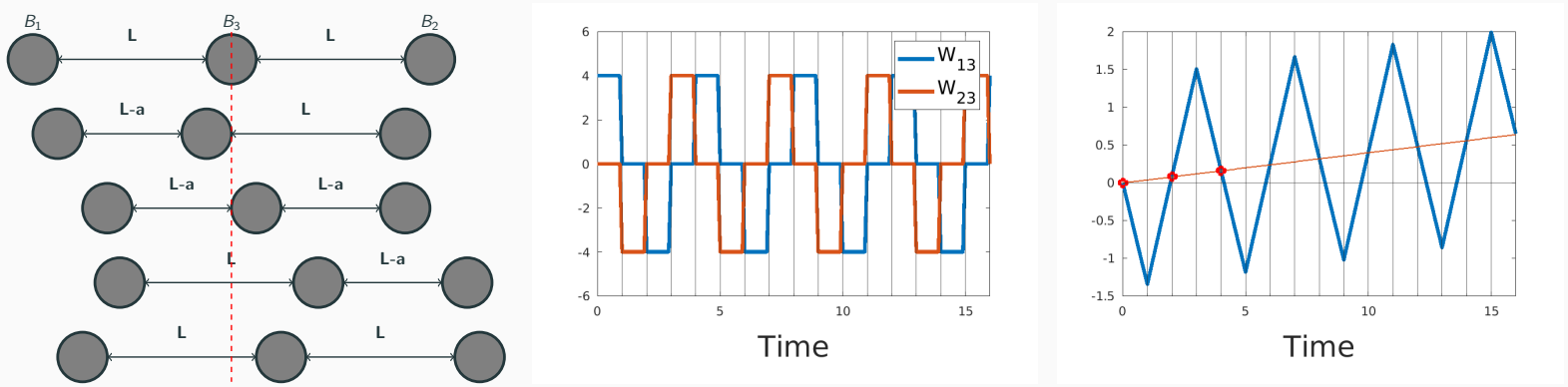
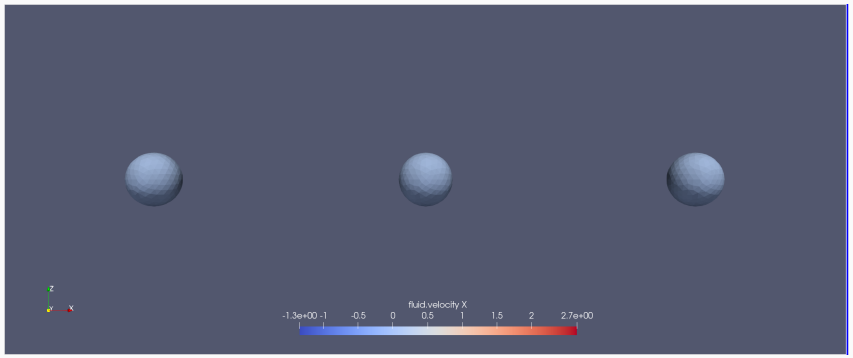


Figure 2: Three-sphere swimmer and its swimming gait.



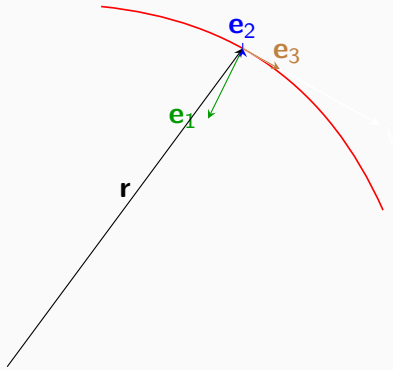
[2]

[2] Luca Berti, Vincent Chabannes, Laetitia Giraldi, Christophe Prud'Homme. Modeling and finite element simulation of multi-sphere swimmers. 2020.

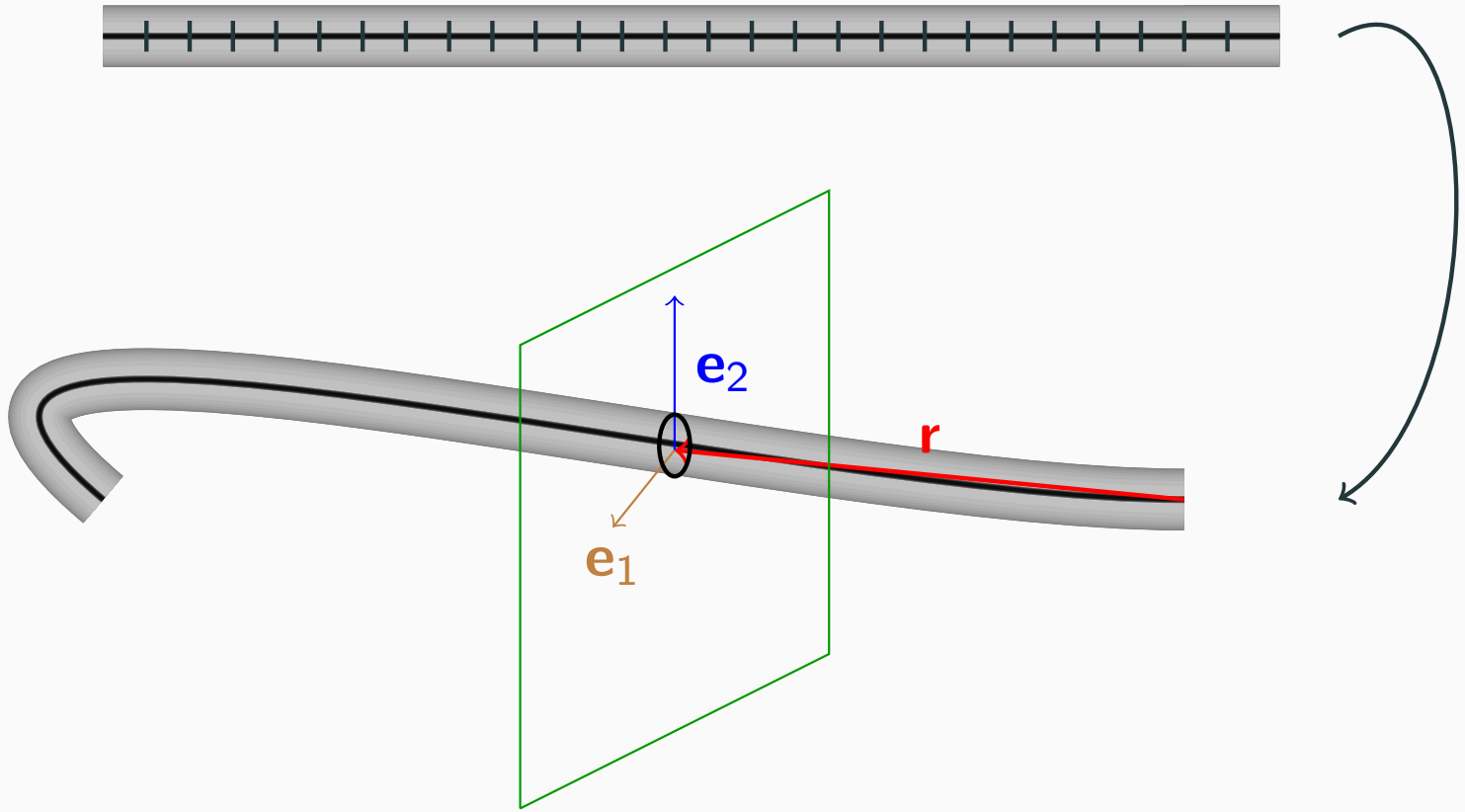
Centreline motion from ODEs

- Position vector $\mathbf{r}$
- Cosserat reference frame $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$
- arclength coordinate l
- torsion parameter τ_f
- curvature function κ_f

$$\begin{cases} \dfrac{d\mathbf{r}}{dl} = -\mathbf{e}_3, \\ \dfrac{d\mathbf{e}_1}{dl} = -\kappa_f \mathbf{e}_3 + \tau_f \mathbf{e}_2, \\ \dfrac{d\mathbf{e}_2}{dl} = -\tau_f \mathbf{e}_1, \\ \dfrac{d\mathbf{e}_3}{dl} = \kappa_f \mathbf{e}_1. \end{cases}$$



Create the 2D/3D mesh from the centerline



- parallel transported frame (to avoid rotating sections along the tail)
- linear interpolation of position between nodes of the 1D model
- extend the 1D motion to the orthogonal section via $\mathbf{e}_1, \mathbf{e}_2$

Position Vs Velocity formulation

Via automatic differentiation, we can obtain

$$\frac{d\mathbf{r}}{dt}, \frac{d\mathbf{e}_1}{dt}, \frac{d\mathbf{e}_2}{dt}, \frac{d\mathbf{e}_3}{dt}.$$

We can recover $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ by time integration

Figure 3: Left: Implicit Euler scheme (blue) and position (red). Right: Runge-Kutta 4 scheme (green) vs Implicit Euler scheme (blue).

Spermatozoon 2D and 3D

- Analytical displacement velocity 2D and 3D
- various parameters (like amplitude) allowing to test the robusteness of the framework and benchmarking with Comsol possible in 2D

Perspectives

Fluid-solid interaction

- Couple a moving hyper-elastic swimmer to the fluid (externally powered)
- Propose two active-elasticity models to be used in the framework (M1 projects)
  Visible elastic effects $\rightarrow$ passive and active components

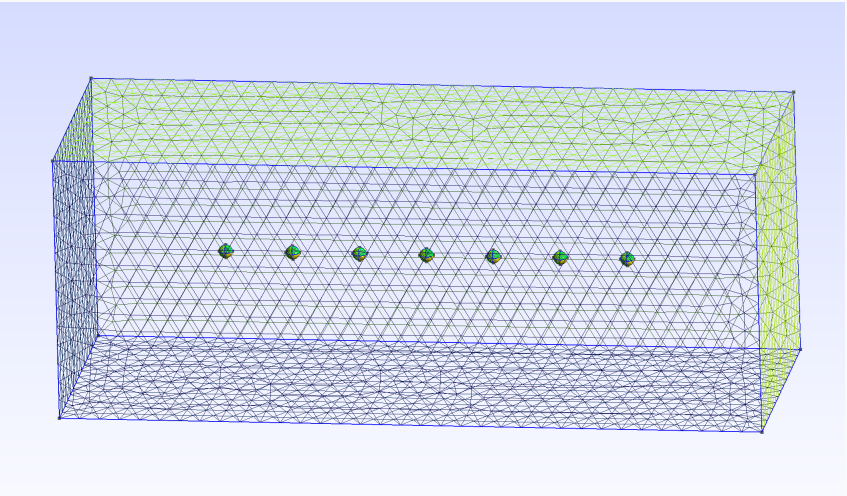
$$\Sigma_p = \Sigma_{visible} - \Sigma_a e_a \otimes e_a \qquad \text{(Additive model)}$$

$$F_p = F_{visible} F_a^{-1} \qquad \text{(Multiplicative model)}$$

- Managing mesh adaptation and motion with Lagrangian description of elasticity
- Add rigid-body velocity in the FSI coupling conditions

Learning to swim

Learn optimal swimming strategies for multi-sphere swimmers via Q-learning (M2 stage).



- steady fluid
- background flow
- next to boundaries