Control and Path Planning Strategies of a Micro-swimmer

Student: Antoine Ruch

Supervisors: Lucas Palazzolo, Céline Van Landeghem

August 24, 2025

Table of Contents

Context

Overview of micro-swimming

- Micro-swimmers: Tiny artificial/biological agents swimming through biological fluid.
- Unique swimming strategies encountered at microscale: *Life at Low Reynold's Number*, Purcell [4] 1977.
- Research interests in mathematics, bio-medicine, and micro-robotics.
- Application perspectives to targeted drug delivery, environment cleaning - minimally invasive treatments.

Context

Internship

- Mandatory two-months internship for the CSMI master's program. Hosted by Cemosis (IRMA), collaboration with INRIA Sophia Antipolis.
- Continuation of second semester coursework "Projet" - *Trajectory Planning for a Micro-swimmer in a Low-Reynolds Number Regime*.
- NEMO ANR project - numerical methods for the simulation of micro-swimmers.

Objectives

Part I

- Implementation of a path-planning procedure of a rigid magnetic head in Stokes fluid.
- Simulations within complex 2D fluid environments.

Part II

- Bayesian optimization for the control of a rigid magnetic head. Coupling `Feelpp` solver with SCBO.
- Validation on simple test cases.

Part III

- Implementation of numerical tools for magnetic swimmer swarms generation.
- Implementation of dipole-dipole interactions.

Table of Contents

Reynold's Number

Derivation of a dimensionless quantity characterizing the nature of flows: **Reynold's Number**.

Reynold's Number

$$\mathrm{Re} = \frac{\rho U L}{\mu} \tag{1}$$

- U and L problem dependent quantities, characteristic speed and length.

At low Re, flows tend to be *laminar* (parallel fluid layers), while at high Re they appear *turbulent* (chaotic and high contrast).

Steady & Unsteady Stokes

Micro-swimmers operate at low-Reynolds number, as small as 10^{-4} (bacteria). Taking Re $\rightarrow 0$ in the Navier-Stokes equation results in the vanishing of the inertial term:

Steady Stokes equation

$$\begin{aligned} -\nabla p + \mu \nabla^2 \mathbf{u} &= \rho \mathbf{a}, \\ \nabla \cdot \mathbf{u} &= 0. \end{aligned} \tag{2}$$

1. Virtually no momentum transfer to the surrounding fluid: balance of torques and forces.
2. A swimmer's net displacement depends solely on the movement's shape, not its speed.
3. Kinematic reversibility: reciprocal motion conveys no net-displacement.

Steady & Unsteady Stokes

Micro-swimming flows are not typically steady. Instead, we keep the time derivative and drop the convective inertial term:

Unsteady Stokes equation

$$\begin{aligned} \rho \partial_t \mathbf{u} + \nabla p - \nabla \cdot \sigma_{\mathcal{F}} &= f_{\mathcal{F}} \text{ on } \mathcal{F}^t, \\ \nabla \cdot \mathbf{u} &= 0 \text{ on } \mathcal{F}^t, \\ \mathbf{u} &= 0 \text{ on } \partial \mathcal{F}^t_D, \\ \sigma_{\mathcal{F}} \cdot \nu_{\mathcal{F}} &= 0 \text{ on } \mathcal{F}^t_N \end{aligned} \tag{3}$$

With $\sigma_{\mathcal{F}} = -pI + \mu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T \right)$.

Rigid body dynamics

We limit our study to the case of rigid body swimmers. Their dynamics can be described using point kinematics:

Newton equations

$$\begin{aligned} m_{\mathcal{S}} \frac{d\mathbf{U}}{dt} &= f_{\mathcal{S}}, \\ \frac{d}{dt}\left(RJ^* R^T \omega_{\mathcal{S}}\right) &= T_{\mathcal{S}}, \end{aligned} \tag{4}$$

with $\mathbf{U}$ the linear velocity, $m_{\mathcal{S}}$ the mass (assumed constant), $\omega_{\mathcal{S}}$ the angular velocity around the center of mass.

Magneto-swimmer model

Flagellated artificial swimmer comprising a magnetic head and elastic flagellum. We only consider the rigid magnetic part.

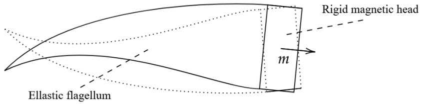


Figure: The full Magneto-swimmer model.

- $\mathbf{T}_M = m \times B$ - magnetic torque for orientation.
- $\mathbf{F}_\epsilon$ - fictitious linear force for self-propulsion. Handled by "*Adaptive linear force*" procedure.

Table of Contents

Second semester project

We identified 2 key objectives:

1. **Trajectory planning** - deciding on a prescribed feasible path to follow.
2. **Trajectory control** - strategies to ensure accurate control and minimize deviations.

Point-to-point startegy

Trajectory control - Implementation of a point-to-point strategy. The swimmer follows a series of ordered points in space.

Algorithm 1 Point-to-point strategy

Input : A PATH put together using a `preProcessingStep()`, e.g. the discretization of a curve, a tolerance δ and the index of the previous check-point n.
Output : A target angle θ giving the orientation of the magnetic field $\mathbf{B}$.

$X \leftarrow$ `getCurrentPosition()`
$\text{nextIndex} \leftarrow n$
$d \leftarrow \|X - \text{PATH}[\text{index}]\|_2$
if $d < \delta$ **then**
  $\text{nextIndex} \leftarrow \text{nextIndex} + 1$
end if
$\theta \leftarrow \arctan 2(\text{PATH}[\text{nextIndex}] - X)$

Distance function for path planning

Trajectory planning - Introduction of a path planning procedure using the distance function. Naturally accounts for obstacles and boundaries.

1. Let $F(x)$ the distance field over the domain $\mathcal{F}$.
2. Assemble the set of mesh elements falling on the level set $F(x) - d = 0$.
3. Determine a series of points to follow using the elements' barycenters.

Distance function

Built-in `Feelpp` method for computing the distance field of any given mesh region $\mathcal{R}^*$ from its boundary $\partial\mathcal{R}^*$ using the **Fast Marching Method** (FMM) Sethian [5]. Involves solving the **Eikonal equation:**

Eikonal equation

$$|\nabla T| = \frac{1}{f}, \quad T|_{\partial\mathcal{R}^*} = 0, \tag{5}$$

with T the arrival time of the boundary $\partial\mathcal{R}^*$ at $x \in \mathcal{R}^*$ moving at speed f in the inward direction.

For $f = 1$:

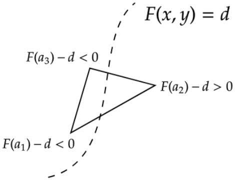
$$\boxed{T(x) = F(x) = \min_{y \in \partial\mathcal{R}^*} \|x - y\|_2}.$$

Implementation

Extracting mesh elements

Algorithm 2 Mesh element extraction

Input : The discretized fluid domain $\mathcal{F}_h$ and a distance d.
Output : A sample BARYCENTERS of the level set $F(x) = d$.
$F \leftarrow$ `FMM`(from: $\partial\mathcal{F}_h$) $-d$
for $e_h \in \mathcal{F}_h$ **do**
    if `checkSigns`(e_h) **then** BARYCENTERS`.push`(e_h`.barycenter`)
    end if
end for



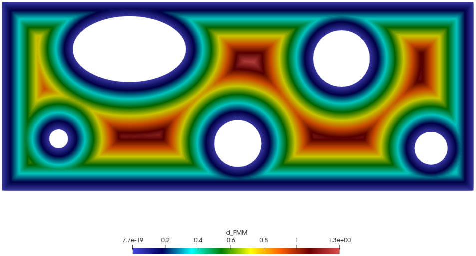
Implementation

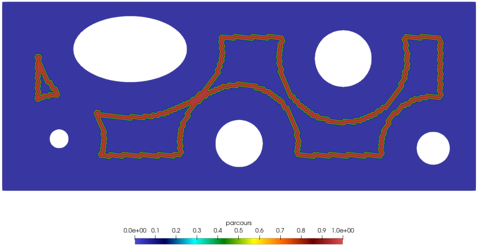
Assembling a sequence of check-points

Algorithm 3 Check-point assembly

Input : The BARYCENTERS container, and initial swimmer position X_{init}.
Output : A CHECKPOINT container of ordered points.

$p_0 \leftarrow$ `closest`(BARYCENTERS, X_{init})
BARYCENTERS.`pop`(p_0)
CHECKPOINT.`push`(p_0)
while BARYCENTERS $\neq \{\emptyset\}$ **do**
    $p \leftarrow$ `closest`(BARYCENTERS, p_0)
    BARYCENTERS.`pop`(p_0)
    CHECKPOINT.`push`(p_0)
    $p_0 \leftarrow p$
end while


d_FMM
7.7e-19
0.2
0.4
0.6
0.8
1
1.3e+00


parcours
0.0e+00
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0e+00

Considerations

Check-point assembly procedure: iterate from a position to the closest. Can lead to some difficulties:

- Non-connected level sets,
- Bypasses $\rightarrow$ unpredictability

Classic solution relevant to path-finding problems: from a point cloud to a geometric graph and path selection using **Dijkstra's algorithm**.

Geometric graph

We propose the `C++` implementation of a `Graph` structure: level-set sample $\rightarrow$ network of `Nodes`.

`Node`

id : $\mathbb{N}$ — unique identifier
subGraph_id : $\mathbb{N}$ — connected subgraph
(x,y,z) : $\mathbb{R}^3$ — coordinates

`Graph`

nbr_Nodes : $\mathbb{N}$ — graph size
nbr_subGraphs : $\mathbb{N}$ — connected subgraphs
data_Nodes : set of `Node`
dist_Map : $(i,j) \rightarrow \mathbb{R}$ — pairwise distances
connect_Map : $i \rightarrow j$ — connectivity

Geometric graph

- Graph is assembled using a depth first search in $O(|V| + |E|)$ Cormen et al. [1], determines connected sub-graphs.
- Depth-first search can be used to find a path, but suboptimal. Instead, implementation of Dijkstra's algorithm to find the shortest path.
- Uses INITIALIZE and GET_MIN sub-procedures.
- GET_MIN naive complexity $O(|V|^2 + |E|)$; we use a priority queue: $O(1)$ access to the smallest element, logarithmic cost insertion and deletion.

Dijkstra's algorithm

Algorithm 5 Djikstra's algorithm

Input: A geometric oriented graph G, starting and ending points s_1 and s_2.
Output: The optimal path from s_1 to s_2.

$S = \{\emptyset\}$
$Q = V$
INITIALIZE(`distance`, `predecessor`)
while Q $\neq \{\emptyset\}$ **do**
    $v \leftarrow$ GET_MIN(Q)
    Q.`pop`(v)
    S.`push`(v)
    for $u \in$ `get_neighbours`(v) **do**
        if `distance`[u] $>$ `distance`[v] $+ \omega(v, u)$ **then**
            `distance`[u] $=$ `distance`[v] $+ \omega(v, u)$
            `prededeccdor`[u] $=$ v
        end if
    end for
end while

Numerical experiments

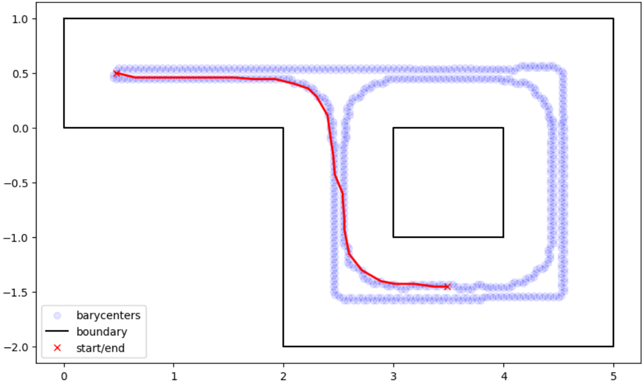


Figure: Simple Dijkstra test case.

Numerical experiments

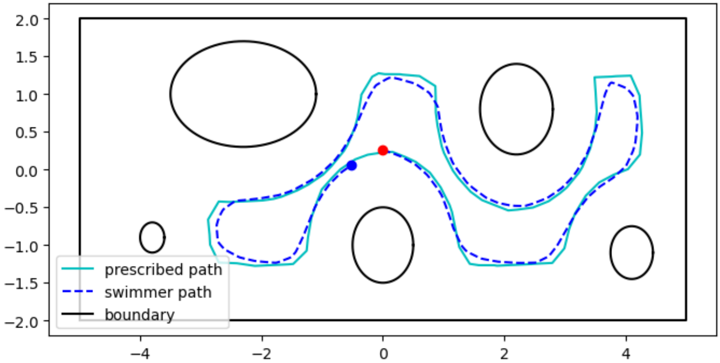


Figure: Magnetic control around a level curve.

Numerical experiments

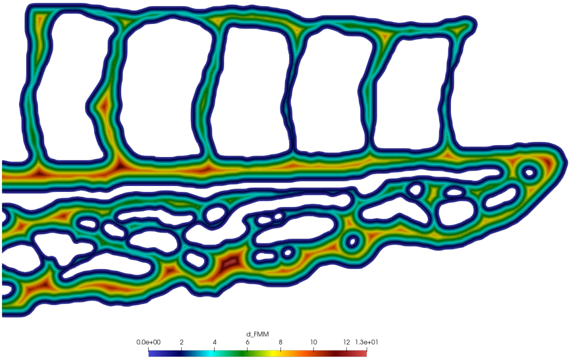


Figure: Zebra-fish tail distance field.

Numerical experiments

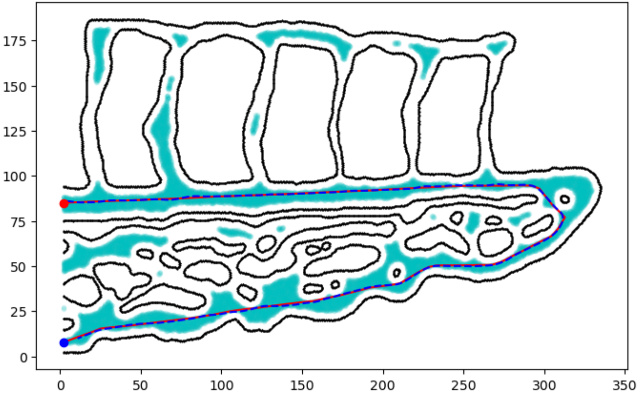


Figure: Magnetic control in a super-level-set.

Table of Contents

Gaussian processes

Objective: Formulate the trajectory control of a micro-swimmer as an optimal control problem.

Gaussian process

Let $(X_s)_{s\in\mathcal{S}}$ a family of random variables. It is said to be a Gaussian process if for any finite subset $A \subset \mathcal{S}$, $(X_s)_{s\in A}$ is a multivariate Gaussian.

$$\begin{aligned} &\phi : s \in \mathcal{S} \mapsto \phi(s) = X_s, \\ &\mu : s \in \mathcal{S} \mapsto \mu(s) = \mathbb{E}[X_s], \\ &K : s, t \in \mathcal{S} \mapsto K(s,t) = \mathrm{Cov}(X_s, X_t). \end{aligned}$$

K is a covariance function or *kernel*, assumed positive semi-definite.

GP regression

Gaussian process regression (w/ noise)

We assume the prior over the behavior of f:

$$f(x) = W(x) + \epsilon, \quad W \sim \mathcal{GP}(0, K) \text{ and } \epsilon \sim \mathcal{N}(0, gI_m)$$

We obtain the following posterior Gramacy [3]:

$$W(x) \mid \{(f(x_i) = y_i)\}_{i=1}^{k} \sim \mathcal{N}(\mu(x), \sigma^2(x)),$$

$$\mu(x) = K(x, \mathcal{X}) \left(\Sigma^{K,g}\right)^{-1} Y_k,$$

$$\sigma^2(x) = K(x, x) - K(x, \mathcal{X}) \left(\Sigma^{K,g}\right)^{-1} K(x, \mathcal{X})^T,$$

with $\Sigma_{i,j}^{K,g} = K(x_i, x_j) + g\delta_{i,j}$.

GP regression

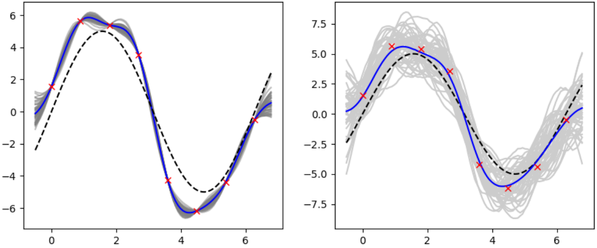


Figure: From left to right, interpolating and noisy GP regressions. The red points are the training data, a noisy sample of the function $\sin(5\pi x)$ which is shown in black here. The blue line is the mean of a generated 50 functions, here in gray.

Bayesian optimization

Objective: Find the global optimum of a continuous black box function f that is computationally expensive.

Bayesian optimization

$$\text{Find } x^* \text{ s.t. } x^* = \arg\min_{x \in \mathcal{S}} f$$

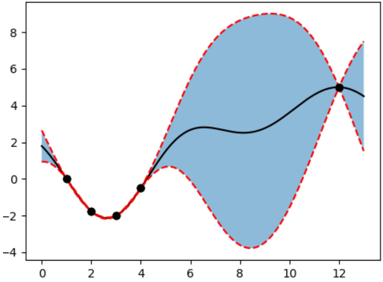
- Apply a prior over f.
- Fit a posterior distribution using a limited amount of training data.
- Construct an **acquisition function** to determine a promising batch to evaluate next.

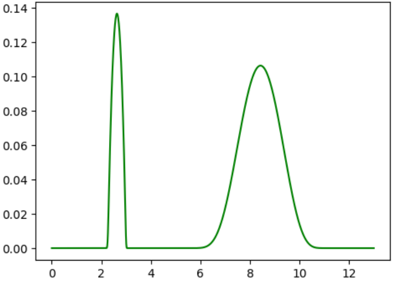
Acquisition function

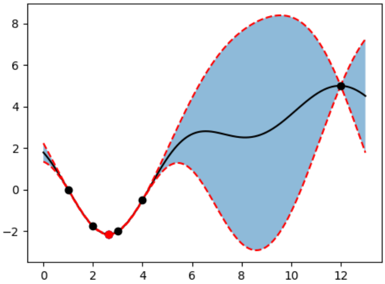
Example: the Expected Improvement (EI)

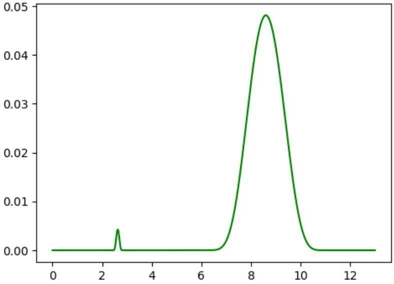
$$EI(x) = \mathbb{E}\left[\max\{(f_n^* - f[x]), 0\}\right], \quad f_n^* = \min_{m \leq n} f(x_m),$$

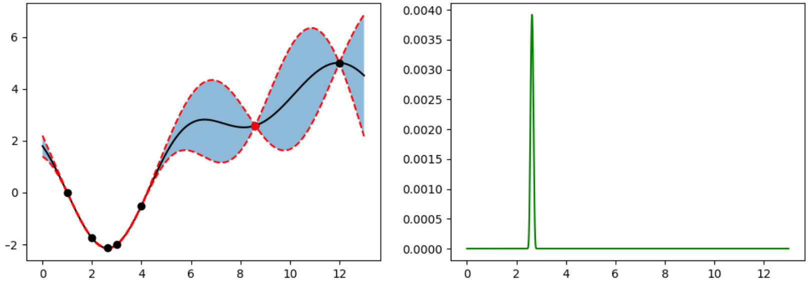
with $f[x]$ the surrogate of f constructed from $D_n = \{x_1, \ldots x_n\}$.
Maximizing the EI yields the best next point to evaluate relative to the current posterior.


8
6
4
2
0
−2
−4
0
2
4
6
8
10
12


0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8
10
12


8
6
4
2
0
−2
0
2
4
6
8
10
12


0.05
0.04
0.03
0.02
0.01
0.00
0
2
4
6
8
10
12


6
4
2
0
-2
0
2
4
6
8
10
12
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
2
4
6
8
10
12

SCBO

The **Scalable Constrained Bayesian Optimization** (SCBO) algorithm Eriksson and Poloczek [2]:

$$\arg\min_{x \in \mathcal{S}} f(x) \text{ s.t } c_1(x) \leq 0, \ldots c_m(x) \leq 0.$$

- Trust-region-based approach to acquisition.
- New points are sampled in the trust region to create a batch of q points to evaluate.
- The trust region is rescaled, and its center updated as better points are discovered.

Magneto-swimmer control

Objective: Control of a rigid magnetic head swimmer along a reference trajectory $X_{\text{ref}} : [0,1] \to \mathbb{R}^d$. The swimmer's path X_u is the solution of a control problem:

$$\begin{cases} \dot{X_u}(t) = f(t, X_u(t), u(t)), \\ X_u(0) = X_0. \end{cases} \qquad u : [0, T] \mapsto \mathbb{R}^m.$$

We seek to minimize the following objective function:

$$C(u,\gamma) = \int_0^T \begin{aligned} &\|X_u(t) - X_{\text{ref}}(\gamma(t))\|_Q \\ &+ \|X_u(T) - X_{\text{final}}\|_R \\ &+ \|u(t)\|_S \; dt \end{aligned}, \tag{6}$$

with Q, R and S the cost matrices, s.t.:

$$\|X - Y\|_Q = (X - Y)^T Q (X - Y).$$

We approach infinite dimension optimization spaces with B-Splines:

$$u \in \mathcal{U} := \{u \in L^2([0,T],\mathbb{R}^m) : m_j \leq u_j(t) \leq M_j\}$$

$$\gamma \in \Gamma := \{\gamma \in \mathcal{C}^0([0,T],[0,1]) : \dot{\gamma} > 0, \gamma(0) = 0, \gamma(T) = 1\}$$

$$\Downarrow$$

$$\tilde{\mathcal{U}} = \left\{ \left(\sum_{i=1}^{N_u^j} P_i^{u_j} B_{i,k}^{u_j}(t) \right)_{j \in \{1,\dots m\}} \text{ s.t. } m_j \leq P_i^{u_j} \leq M_j \right\}$$

$$\tilde{\Gamma} = \left\{ \sum_{i=1}^{N_\gamma} P_i^\gamma B_{i,k}^\gamma(t) \text{ s.t., for } \Delta P_i^\gamma = P_{i+1}^\gamma - P_i^\gamma : \begin{cases} \Delta P_i^\gamma \geq 0 \\ P_0^\gamma = 0 \\ 1 - \epsilon \leq \sum \Delta P_i^\gamma \leq 1 \end{cases} \right\}$$

Finally, the optimization problem we wish to solve is:

$$\boxed{\arg\min_{P_i^{\gamma}, P_i^{u_j}} C(\gamma, u) \text{ s.t. } \sum P_i^{\gamma} \leq 1, -\sum P_i^{\gamma} \leq \epsilon - 1.}$$

We use two control splines:

1. $u_1(t) = \sum_{i=0}^{9} P_i^{u_1} B_i^{u_1}(t)$ - the target velocity, input of the *Adaptive linear force* procedure.

$$P_i^{u_1} \in [0, 6]$$

2. $u_2(t) = \sum_{i=0}^{9} P_i^{u_2} B_i^{u_2}(t)$ - the angular velocity $\dot{\theta}(t)$ of the magnetic field $\mathbf{B}$ s.t.:

$$\mathbf{B}(t) = B \begin{pmatrix} \cos(\theta(t)) \\ \sin(\theta(t)) \end{pmatrix} \quad P_i^{u_2} \in [-2\pi, 2\pi].$$

As a first approach we do not optimize the reference path parametrization γ.

Test case n°1 - Reaching a target point

Optimization parameters

Q	R	n_{iter}	s_{batch}	C_{final}
0	I_3	250	10	7.46×10^{-2}

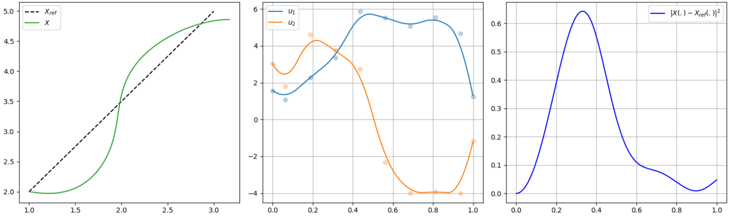


Figure: **Left:** The swimmer path in green, the reference path black. **Middle:** The spline curves and their coefficients. **Right:** The cost error between the reference path and the swimmer's trajectory.

Test case n°2 - Avoiding an obstacle

Optimization parameters

Q	R	n_{iter}	s_{batch}	C_{final}
$100 \times J_3$	$10 \times I_3$	250	10	4.31×10^2

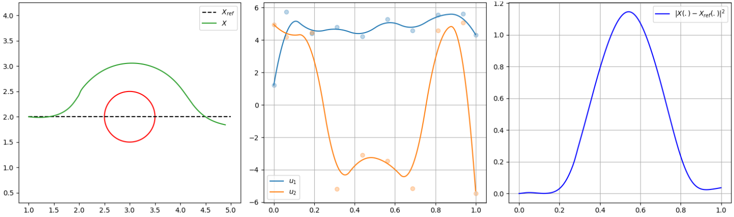


Figure: The obstacle is indicated in red on the **left**. It is a circle perforation of radius 0.5 centered at $(3, 2)$.

Test case n°3 - Following a curve

Optimization parameters

Q	R	n_{iter}	s_{batch}	C_{final}
J_3	0	250	10	7.18×10^{-2}

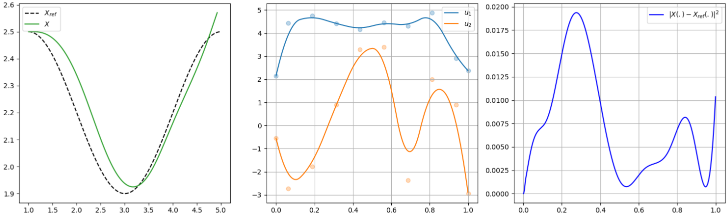


Figure: The prescribed path has the analytical expression of a sine wave.

Test case n°4 - Overcoming boundary effects

Optimization parameters

Q	R	n_{iter}	s_{batch}	C_{final}
J_3	0	200	10	2.66×10^{-1}

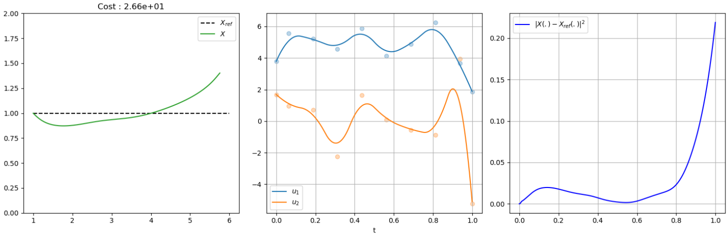


Figure: The swimmer is tasked to go in a straight line while close to the bottom boundary.

Table of Contents

Swarm generation tools

Objective: Propose numerical tools for swarm generation and implementation of multi-objects magnetic control in `Feelpp`. Observe collective behaviors with high-fidelity solver.

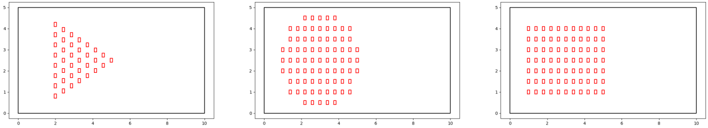


Figure: Customizable swarm formations and automatic generation of `Feelpp` configuration files (`.geo` + `.json`) with geometry feasibility assertions.

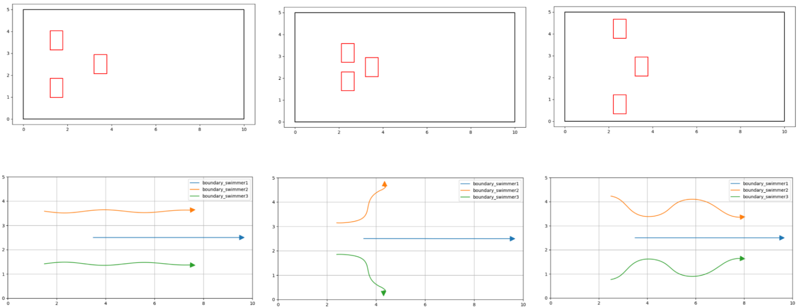


Figure: Spread out 3-swimmers in triangle configuration with no magnetic field.

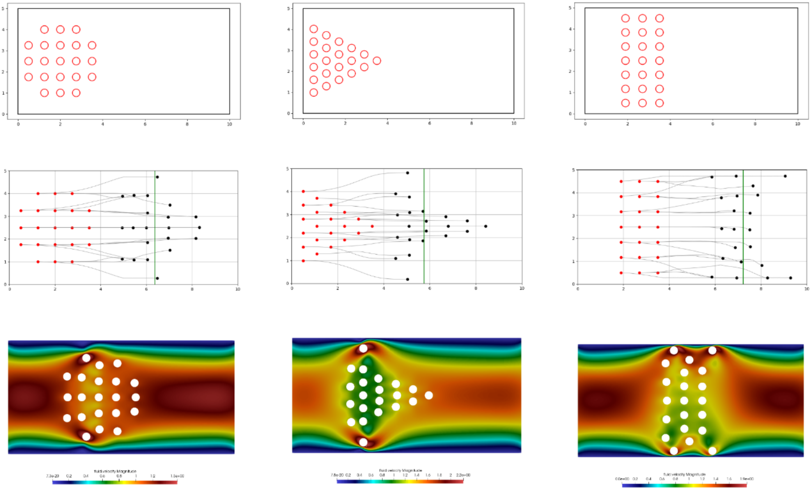


Figure: 21 disk-swimmers in different configurations with no magnetic field.

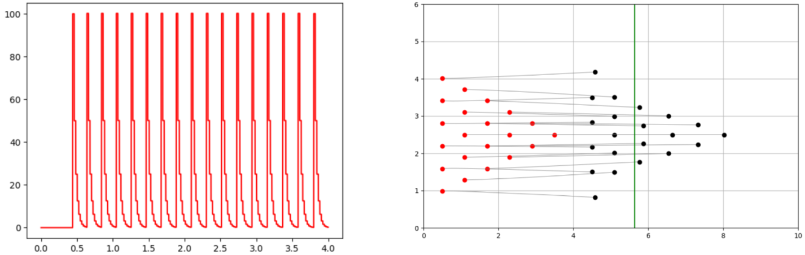


Figure: Triangle formation with periodically activating magnetic field to correct deviations.

Modelling

We propose the implementation of dipole-dipole forces and torques from the interaction between multiple magnetic particles. Let $\{\mathbf{r}_1, \ldots, \mathbf{r}_n\}$ particle positions and $\{\mathbf{m}_1, \ldots, \mathbf{m}_n\}$ magnetic moments.

- The torque experinced by i following its interaction with j:

$$\mathbf{T}_i^j = \mathbf{m}_i \times \mathbf{B}_j(\mathbf{r}_i),$$

with:

$$\mathbf{B}_j(\mathbf{r}_i) = \frac{\mu_0}{4\pi|\mathbf{r}_{ij}|^3}\left[3(\mathbf{m}_j \cdot \hat{\mathbf{r}}_{ij})\hat{\mathbf{r}}_{ij} - \mathbf{m}_j\right].$$

Modelling

- The force produced by j onto i:

$$\begin{aligned}\mathbf{F}_{ij} &= \nabla_{\mathbf{r}_i}(\mathbf{m}_i \cdot \mathbf{B}_j) \\ &= \frac{3\mu_0}{4\pi|\mathbf{r}_{ij}|^4}(\mathbf{m}_j(\mathbf{m}_i \cdot \hat{\mathbf{r}}_{ij}) + \mathbf{m}_i(\mathbf{m}_j \cdot \hat{\mathbf{r}}_{ij}) + \hat{\mathbf{r}}_{ij}(\mathbf{m}_i \cdot \mathbf{m}_j) \\ &+ 5\hat{\mathbf{r}}_{ij}(\mathbf{m}_j \cdot \hat{\mathbf{r}}_{ij})(\mathbf{m}_i \cdot \hat{\mathbf{r}}_{ij})),\end{aligned}$$

with:

$$\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j, \quad \hat{\mathbf{r}}_{ij} = \frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|}.$$

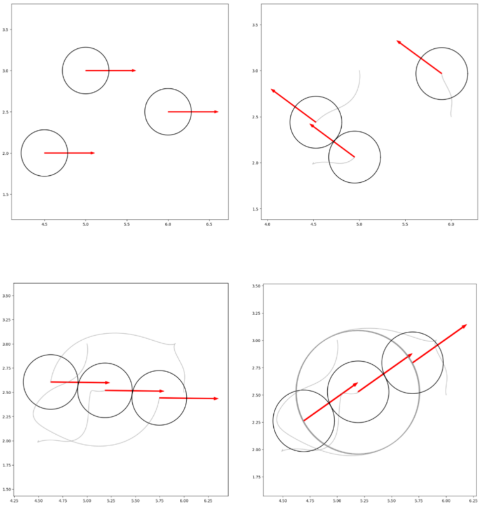


Figure: 3 dipoles at different simulation times with rotating magnetic field $\mathbf{B}(t) = B(\cos(2\pi t), \sin(2\pi t))$

Table of Contents

Conclusion

- We obtained satisfying results for simple test cases of the Bayesian optimization.
- Bayesian optimization finds purpose when the cost function is expensive to evaluate, and the high fidelity solver allows for arbitrary complex configurations.
- However, the computational burden of each simulation turned out to be a limitation.
- Future work should focus on reducing computational load: parallelize batch evaluation, or use a low fidelity solver to converge quickly and use it as a starting batch.

References

[1] Thomas H. Cormen et al. *Introduction to Algorithms*. 4th. The MIT Press, 2009.

[2] David Eriksson and Matthias Poloczek. "Scalable Constrained Bayesian Optimization". In: *CoRR* abs/2002.08526 (2020).

[3] Robert B. Gramacy. *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Boca Raton, Florida: Chapman Hall/CRC, 2020.

[4] E. M. Purcell. "Life at low Reynolds number". In: *American Journal of Physics* 45 (Jan. 1977), pp. 3–11. DOI: 10.1119/1.10903.

[5] J A Sethian. "A fast marching level set method for monotonically advancing fronts.". In: *Proceedings of the National Academy of Sciences* 93 (1996), pp. 1591–1595. DOI: 10.1073/pnas.93.4.1591.