INTERNSHIP REPORT
Control and Path Planning Strategies of a Micro-swimmer

Student: Antoine Ruch
Supervisors: Lucas Palazzolo, Céline Van Landeghem

August 21, 2025

Contents

1 Introduction

1.1 Micro-swimming

Micro-swimming refers to the scientific field that studies the motion of biological and synthetic objects through a fluid environment. Biological micro-swimmers, such as bacteria or sperm cells, evolve within physical constraints vastly different from what we typically experience at macro-scale Berg and Berry [4]. Edward Purcell, in his paper *Life at Low Reynolds Number* Purcell [24], was one of the first to highlight the unique challenges encountered by swimming organisms at the microscopic scale.

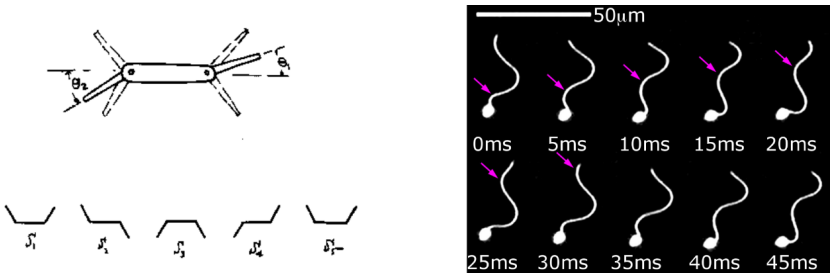

Figure 1: On the left an artificial swimmer from Purcell [24] and on the right sperm-cells shape deformation from Elgeti, Winkler, and Gompper [12]

Since then the study of micro-swimming has sparked researchers' interests in diverse fields of study, from mathematics to bio-medicine and micro-robotics. Recent developments have brought forward concievable solutions for the manufacturing and design of bio-inspired artificial swimmers, driven by perspectives in targeted drug delivery and environment cleaning for minimally invasive medical interventions Bunea and Taboryski [8].

Of course, laboratory experiments are expensive and time-consuming to set up. Mathematical modelling and numerical simulation thus play a crucial role in understanding a micro-swimmer's physical constraints and behavior within complex biological environments.

1.2 Context

The present work was carried out as part of a mandatory two-months internship for the CSMI Master's program. The topic was first introduced during the second semester coursework "Project" for which a report titled *Trajectory Planning for a Micro-swimmer in a Low-Reynolds Number Regime* was submitted. This internship thus acts as the continuity of said project and was completed within the scope of the NEMO ANR project, which focuses on developing numerical methods for the simulation and control of magnetically activated swimmers called **magneto-swimmers**.

In the context of an ongoing collaboration between IRMA [1] Strasbourg and INRIA [2] Sophia-Antipolis, this internship was supervised by PhD students *Lucas Palazzolo* (INRIA) and *Céline Van Landeghem* (IRMA), both former CSMI students. I would like to express my most sincere thanks to them for their patience and benevolence. My grattitude also goes to *Christophe Prud'homme* (IRMA) for allowing me to continue my semester work during this internship, and *Laetitia Giraldi* (INRIA) for her guidance and supervision during this project.

[1] Institut de Recherche en Mathématiques Avancées
[2] Institut National de Recherche en Informatique et Automatique

1.3 Objectives

This internship can be divided into 2 main segments. The first part was supervised *Céline Van Landeghem*, who was also in charge of supervising the second semester project. Her PhD thesis focuses on numerical methods for micro-swimming simulations. The second part was supervised and based on a project proposed by *Lucas Palazzolo*, whose PhD work focuses on control methods for micro-swimming optimizations.

Part 1 The first part of the internship picks up where we left off for the second semester project, during which we experimented different control strategies of a rigid magnetic head within simple fluid environments. We propose a path planning procedure using the distance function, along with a few simple test cases. Using a point-to-point guiding procedure we also present simulations of the rigid magnetic head control within complex fluid environments.

Part 2 This second part of the internship investigates the viability of a Bayesian Optimization algorithm for the control of the rigid magnetic head in a Stokes fluid. We first propose a coupling of a provided `Python` implementation of a Bayesian optimization algorithm with the `Feelpp` solver used during the second semester project to simulate micro-swimming, along with the control procedures developed at the time. We then validate our application on different test cases including trajectory tracking and obstacle avoidance.

In this report we also introduce numerical tools for the simulation of magnetic micro-swimmer swarms, or **magneto-swarms**.

1.4 Numerical framework

Feel++ The numerical solvers used in this internship were provided by and implemented using the `Feelpp` open-source library Feel++ [14]. `Feelpp` is scientific computing library written in `C++` which can be used to solve a large range of differential equations using the finite-element method. The library integrates meshing solutions as well as `Python` binders for a smoother experience. `Feelpp` also offers a flexible parametrization of its applications using a `json` file.

Python This project uses `Python` together with `Feelpp` binders and other common utilities, most notably `Botorch` for Bayesian optimization, `Scipy` for handling B-Splines, `Numpy` for linear algebra and `Matplotlib`, which is used for most of the plots showcased in this report.

ParaView ParaView is an open-source data visualizing application. `Feelpp` post-processing components allow for flexible and reliable visualization of simulation output using ParaView.

GMSH GMSH is an open-source finite element mesh generator with built-in post-processing features. It is used throughout this project to design the swimmer's body and its environment.

GitHub GitHub is a powerful version control application and a core component of collaborative development. A well maintained and documented GitHub repository is part of the final deliverables for this project.

2 Modeling of micro-swimmers

2.1 Navier-Stokes equations & Reynold's number

Organisms swimming at microscopic scales experience unique physical constraints referred to as low Reynolds number regimes, after the dimensionless quantity constructed from nondimensionalizing the **Navier-Stokes equations**:

$$\rho\left(\partial_t \mathbf{u} \ + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \rho\mathbf{a}, \tag{1}$$

where $\mathbf{u}$ is a velocity field, a vector quantity, defined over a fluid domain $\mathcal{F} \subset \mathbb{R}^d, d = 2, 3$. p is the pressure field, a scalar quantity, defined over $\mathcal{F}$. ρ is the fluid mass density, and μ the dynamic viscosity of the fluid, both of which we will suppose to be constant over time. Finally, $\mathbf{a}$ is the body accelerations acting on the continuous fluid, e.g. gravity. This equation describes the motion of viscous incompressible Newtonian fluids, with the addition of a zero divergence velocity field:

$$\nabla\cdot\mathbf{u} = 0$$

Nondimensionalizing helps to gain insights into the relative importance of each quantity. Scaling each term yields a dimensionless quantity referred to as **Reynold's number**:

$$\mathrm{Re} = \frac{\rho U L}{\mu}, \tag{2}$$

where U and L are called characteristic velocity and lengths, which are problem dependent quantities that encapsulate the flow's relative size and speed. In our case we will typically make a rough estimate and express Reynold's number as $\|\mathbf{u}\|/\mu$, taking $\rho = 1$ and $L = 1$, considering we can not practically simulate microscopic environments. Low Reynolds number flows are said to be laminar, as opposed to turbulent - for microorganisms, this number is typically of the order of 10^{-5}. Taking $\mathrm{Re} \to 0$, we obtain the so-called **Stokes system**:

$$\begin{cases} -\nabla\mathbf{p} + \mu\nabla^2\mathbf{u} = \rho\mathbf{a}, \\ \nabla\cdot\mathbf{u} = 0. \end{cases} \tag{3}$$

Reynolds number is often introduced as being the ratio between inertial and viscous forces. In this case, inertia's contribution vanishes, completely overcame by the shear resistance of the adjacent fluid. As a result, a swimmer cruising in a low Reynolds fluid transfer virtually no momentum to its surroundings and reaches its terminal velocity almost instantly. This translates to the balance of external and fluid forces and torques in Newton's law on the swimmer Lauga and Powers [19]:

$$\mathbf{F}_{\text{ext}} + \mathbf{F} = 0, \quad \mathbf{T}_{\text{ext}} + \mathbf{T} = 0.$$

The above equations are part of necessary boundary conditions to ensure proper coupling of body-fluid equations. We also impose no-slip conditions on the swimmer's boundary, enforcing zero fluid velocity relative to the surface of the swimmer.

Another key take away from the description of Stokes flow is the kinematic reversibility property, stating that any time reversed motion causing the swimmer to deform in a non-reciprocal way will convey no net displacement. A vivid illustration of this property is given by Purcell's **Scallop Theorem**, considering the scallop, by opening and closing its valves, performs a reciprocal motion and thus would do quite poorly as a Stokes swimmer.

2.2 Fluid & rigid body coupling

Unsteady stokes The Stokes equation, although an approximation of low Reynolds fluid dynamics, actually provides a good estimate for $\mathrm{Re} < 0.1$ Proudman and Pearson [23]. However, the expression provided here (3) is of a steady Stokes flow, which is not typically the case in micro-swimming. Instead, we keep the time derivative term $\rho\partial_t\mathbf{u}$ but drop the convective inertial term $(\mathbf{u}\cdot\nabla)\mathbf{u}$, giving rise to a model called the unsteady Stokes equation:

$$\begin{aligned} \rho_{\mathcal{F}}\partial_t\mathbf{u} - \nabla p + \mu_{\mathcal{F}}\nabla^2\mathbf{u} &= f_{\mathcal{F}} \text{ on } \mathcal{F}^t, \\ \nabla\cdot\mathbf{u} &= 0 \text{ on } \mathcal{F}^t, \\ \mathbf{u} &= 0 \text{ on } \partial\mathcal{F}_D^t, \\ \sigma_{\mathcal{F}^t}\cdot\nu_{\mathcal{F}^t} &= 0 \text{ on } \mathcal{F}_N^t, \end{aligned} \tag{4}$$

where $\mathcal{F}^t$ is the domain occupied by the fluid at time t. We assume constant fluid density $\rho\mathcal{F}$ and viscosity $\mu_{\mathcal{F}}$ throughout this report. We call $\sigma_{\mathcal{F}^t}$ the stress tensor defined as $-pI + \mu_{\mathcal{F}}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right)$. Finally, $\partial\mathcal{F}_N^t$ and $\partial\mathcal{F}_D^t$ are portions of the fluid domain boundary over which we enforce respectively Neumann and Dirichlet conditions.

In the following section, we provide only a fraction of the theoretical elements underlying the simulation of micro-swimmers, which is developed in great detail in Landeghem [18]

Rigid body model Let $\mathcal{S}^t$ the swimmer domain inscribed within the fluid domain. In the present work we only consider rigid bodies, the dynamics of which can be described using point kinematics and Newton's second law:

$$\begin{aligned} m_{\mathcal{S}^t}\frac{d\mathbf{U}}{dt} &= \mathbf{f}_{\mathcal{S}^t}, \\ \frac{d}{dt}\left(RJ^*R^T\omega_{\mathcal{S}^t}\right) &= \mathbf{T}_{\mathcal{S}^t}, \end{aligned} \tag{5}$$

where U is the body's linear velocity and $m_{\mathcal{S}^t}$ its mass, we suppose to remain constant over time. $\omega_{\partial\mathcal{S}^t}$ is the angular velocity around the center of mass $\mathbf{X}_{\mathrm{cm}}$, J^* is the moment of inertia expressed is the swimmer's frame of reference and R is the rotation matrix from the initial fluid domain to the swimmer's frame of reference.

Coupling conditions We mentioned earlier that for the swimmer's dynamics to be accurately described within a fluid environment we need to ensure proper coupling of the Stokes and Newton equations through the continuity of the fluid velocity at interfaces, i.e.,no-slip conditions on the swimmer's surface:

$$\mathbf{u} = \mathbf{U} + \omega_{\mathcal{S}^t}\times(\mathbf{X} - \mathbf{X}_{\mathrm{cm}}), \quad \mathbf{X}\in\partial\mathcal{S}^t. \tag{6}$$

We also need to ensure proper coupling of forces and torques acting on the swimmer. If $f_{\mathcal{S}^t}$ and $T_{\mathcal{S}^t}$ respectively the forces and torques acting on the swimmer's body we must have:

$$\begin{cases} \mathbf{f}_{\mathcal{S}^t} = \mathbf{f}_G + \mathbf{f}_L + \mathbf{f}_H, \\ \mathbf{T}_{\mathcal{S}^t} = \mathbf{T}_G + \mathbf{T}_L + \mathbf{T}_H, \end{cases} \tag{7}$$

With L standing for lubrication, G for gravitational and H for hydrodynamic forces and torques.

Collision model Micro-swimmers move through complex fluid domains potentially involving obstacles and walls. In this project we are also led to simulate swarms of micro-swimmers, hence the need for a collision handling procedure. Approaches found in the literature mainly rely on collision and lubrication forces, the latter acting as a corrective force during near collision events Wachs et al. [28]. In this internship we use a contact-avoidance approach (Glowinski [16]), using lubrication forces and torques, implemented and generalized to complex shaped bodies in Landeghem [18]. The collision detection phase also relies on a narrowband **Fast Marching Method** proposed in Sethian [26].

In order to model lubrication forces and torques, a short range force is added to the balance of forces as soon as a collision is detected, i.e, when the distance d falls below a given threshold ρ_L. Let $X_{\mathcal{S}^1}$ and $X_{\mathcal{S}^2}$ the contact points between two rigid complex shaped bodies. The lubrication force and torque acting on $\mathcal{S}^1$ are given by:

$$\begin{aligned} \mathbf{F}_L^1 &= \frac{1}{\epsilon}\left(\mathbf{X}_{\mathcal{S}^1} - \mathbf{X}_{\mathcal{S}^2}\right)(\rho_L - d)^2 \mathbf{1}_{l \leq \rho_L}, \\ \mathbf{T}_L^1 &= -(\mathbf{X}_{\mathcal{S}^1} - \mathbf{X}_{\mathcal{S}^1,\mathrm{cm}}) \times \mathbf{F}_L^1. \end{aligned} \tag{8}$$

The same goes for $\mathcal{S}^2$.

2.3 Numerical resolution

The simulation of micro-swimmers allows for different numerical strategies based on how complex we intend the fluid domain's geometry and dynamic to be. Advantages, shortcomings as well as implementation techniques and optimizations of these approaches are thoroughly discussed in Berti [5], Landeghem [18]. One of the most common method found in the literature is the Resistive Force Theory, which allows for an ODE description of the micro-swimming problem. Although fast this technique falls short when it comes to modelling swimming in bounded fluid domains. More expensive techniques such as the Boundary Element Method offer more robustness and generalize well to complex swimmer shapes and fluid models.

The technique used in this internship is the **Finite Element Method**, which implies the discretization of the fluid domain into geometrical elements, in our case triangles. Meshes are especially convenient for solving problems formulated in an Eulerian frame of reference, meaning we are interested in the evolution of a certain quantity over a fixed domain, at specific spatial coordinates. This naturally clashes with the Lagrangian description of the swimmer's dynamics, where the reference frame is attached to the material domain. To allow for the deformation of domains within an Eulerian description of fluid mechanics, the Arbitrary Lagrangian-Eulerian (ALE) method is employed, where a bijective map from the initial fluid domain $\mathcal{F}^{t=0}$ to the current one $\mathcal{F}^t$ is defined:

$$\mathcal{A} : (0, T] \times \mathcal{F}^0 \mapsto \mathcal{F}^t.$$

Contributions from Landeghem [18] to the `Feelpp` library allow us to use an ALE method to simulate micro-swimming. Using this method on a discretized domain however, involves the displacement of mesh elements according to the computed fluid dynamics. A periodic remeshing is thus necessary to prevent invalid elements from occurring. Moreover, the ALE method requires the mesh's topology not to be altered, hence the need for a collision-avoidance scheme as mentioned in section 2.2.

2.4 Artificial swimmers

We mentioned earlier that for a micro-swimmer to move through a Stokes flow required its body to deform in a non-reciprocal fashion. Moreover, the resulting trajectory is essentially geometric, manifesting from a sequence of shapes the swimmer's body assumes over time, the speed of execution only impacting the amount of time at which a given distance is covered. These sequence of shapes, called "strokes", are studied in the literature for their controllability and efficiency, relating the problem of micro-swimming with the mathematical theory of optimal control, see in Alouges, F. [2]. Several artificial swimmer designs have been studied for their controllability, most notably Purcell's three-link swimmer which consists of three rigid arms chained together. An optimal link-length ratio was derived in Giraldi, Martinon, and Zoppello [15]. We also mention the three-sphere swimmer composed of three spheres attached together that moves by shrinking and extending its arms, proposed by Najafi and Golestanian [20].

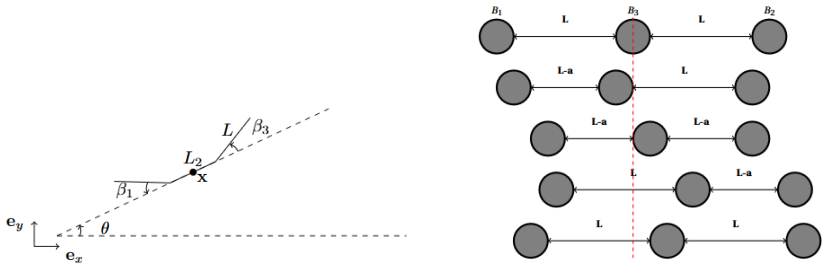

Figure 2: On the left the Purcell's 3-link swimmer Giraldi, Martinon, and Zoppello [15] and on the right Najafi and Golesanian's 3-sphere swimmer Berti et al. [6]

The scale at which micro-swimmer's are supposed to operate makes built-in powering sources difficult to implement, that is why most of artificial swimmers found in the literature rely on external energy sources, such as magnetic or acoustic Rao et al. [25]. The swimmer we are interested in is one of the first iteration of a magnetically activated micro-swimmer, called the *Magneto-swimmer*.

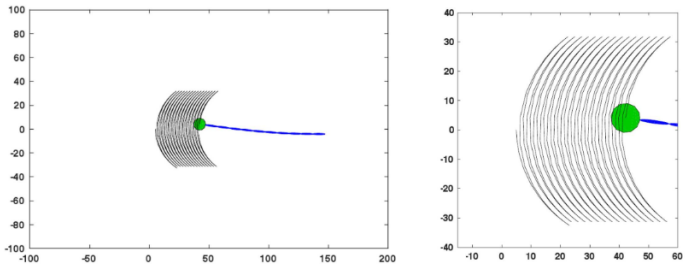

Figure 3: Magnetically actuated sperm-cell-like swimmer in Zoppello et al. [31]

The Magneto-swimmer model The original model was proposed in Dreyfus et al. [10], introducing a bio-inspired design composed of a chain of magnetic particles linked by DNA and attached to a red blood cell. Filaments were aligned using an external uniform magnetic field and actuated by an oscillating transverse one. This sperm-cell-like design has been analyzed for its controllability along curved path in Zoppello et al. [31] and for its viability in Alouges et al. [1]. For this internship, we used the modelling work done in Landeghem [18] of a flagellated micro-swimmer. Its main components include a magnetic head to which is attached an elastic flagellum.

The swimmer's motion is initiated by the activation of an oscillating magnetic field, taking advantage of the surrounding viscous forces acting on the swimmer's flagellum to produce a net displacement. During the second semester project however, we only considered the rigid magnetic part - a simplified version of the magneto-swimmer we continued working with during the internship. The rigid part of the swimmer possesses a magnetic moment $\mathbf{m}$ aligned with its orientation θ:

$$\mathbf{m} = m \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}$$

We are able to control the orientation of the magnetic head using an external uniform magnetic field $\mathbf{B}$, which produces a torque around the swimmer's center of mass as it interacts with the magnetic moment:

$$\mathbf{T}_M = \mathbf{m} \times \mathbf{B}$$

As a result the balance of torques acting on the rigid body are modified to take into account the addition of a magnetic torque:

$$\mathbf{T}_{\mathcal{S}^t} = \mathbf{T}_G + \mathbf{T}_L + \mathbf{T}_F + \mathbf{T}_M$$

In order to model the swimmer's self-propulsion capacities will also add a fictitious linear force $\mathbf{F}_\epsilon$ to the balance of forces:

$$\mathbf{f}_{\mathcal{S}^t} = \mathbf{f}_G + \mathbf{f}_L + \mathbf{f}_F + \mathbf{F}_\epsilon$$

Another way of inducing the translational motion of the swimmer domain $\mathcal{S}$ is to impose Dirichlet type boundary conditions on $\partial\mathcal{S}$ such that the swimmer is advected linearly in the direction of its current orientation. However, the way boundary conditions are handled in our `Feelpp` solver makes this approach incompatible with lubrication forces, which will simply not be enforced during near contact events. We thus rely on the fictitious force $\mathbf{F}_\epsilon$ to initiate linear motion on the magnetic head. This force is aligned with the swimmer's orientation, and its magnitude can give us some insights on the thrust necessary to move an object through a Stokes fluid. During the second semester project we presented a procedure to dynamically adapt this amplitude such that the swimmer reaches and maintains a given velocity. We will refer to this procedure as the **Adaptive linear force** procedure throughout this report.

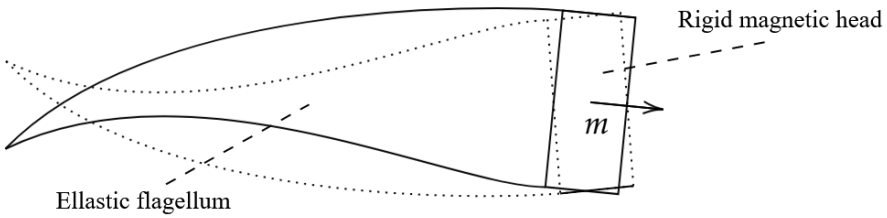

Figure 4: The full Magneto-swimmer model.

3 Distance function strategy

3.1 Review of second semester project

The primary objective of the second semester project was to gain insights into the challenges of micro-swimming numerical simulations, especially during close encounters with boundaries and obstacles. We identified two key components of micro-swimming simulations:

1. **Trajectory planning** - or deciding on a prescribed feasible path for the swimmer to follow. This is assuming no significant change to the swimmer's environments for the time of the simulation.

2. **Trajectory control** - implementing and tracking control strategies such that the swimmer responds to the surrounding hydrodynamics forces and minimizes its deviation from the prescribed path.

Of the control strategies implemented for the project, the **Point-to-point strategy** yielded the most promising results. During the trajectory planning phase we decide on a series of ordered points in space, which could be obtained from the discretization of a curve. The trajectory control phase is then relatively straight forward: the swimmer is tasked with going through each point, one after the other. The first version of this strategy's implementation read:

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

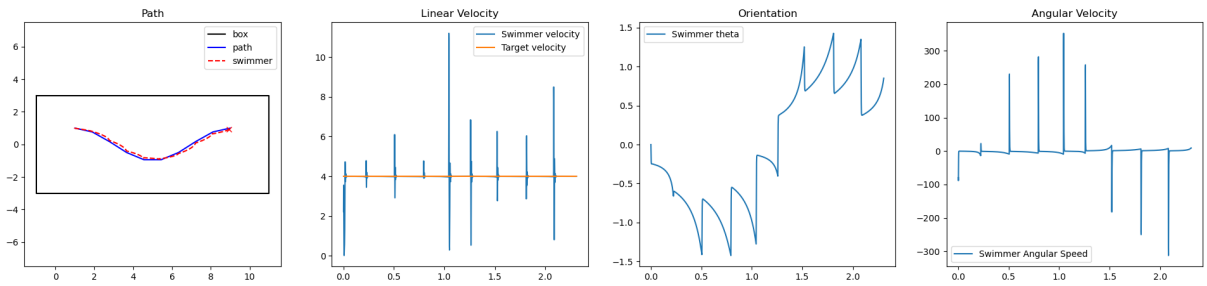

Figure 5: Check-point strategy experiment for a discretized cosine

Although the swimmer in this configuration appears to behave as expected, we notice peaks of high angular velocity around the check points, which could suggest instabilities in the control strategy. For the internship we implemented a revised version of this procedure. In particular, we opted for a higher threshold of at least 2 times the mean distance between each point. The index also keeps being incremented until the distance is higher than the threshold.

3.2 Trajectory planning

During the second semester project we introduced a pre-processing step involving the distance function for trajectory planning. However, computing the distance field over a 2 or 3 dimensional not necessarily convex region is a difficult task. Thankfully, the `Feel++` library offers us a built-in method for computing the distance field of any given mesh region $\mathcal{S}^*$ from its boundary $\partial\mathcal{S}^*$. The provided implementation is based on the **Fast Marching Method** (FMM) we mentioned earlier (2.2), which involves solving the **Eikonal equation**:

$$|\nabla T| = \frac{1}{f}, \quad T(x) = 0 \text{ on } \partial\mathcal{S}^*, \tag{9}$$

with T the arrival time at $x \in \mathcal{S}^*$ of the moving boundary $\partial\mathcal{S}^*$ at speed f in the inward normal direction, enforcing an entropy condition to discriminate weak solutions, similarly to Burgers' equation for which one can construct infinitely many weak solutions due to the formation of shock waves. For $f = 1$ the solution corresponds to the distance field F:

$$F(x) = \min_{y \in \partial\mathcal{S}^*} \|x - y\|_2. \tag{10}$$

Using the distance field of the fluid region $\mathcal{F}$ we can extract specific mesh elements e_h of the discretization region $\mathcal{F}_h$ located a given distance d away from the boundary. This amounts to deciding whether an element falls on the level set $F(x) - d = 0$ based on the following criterion:

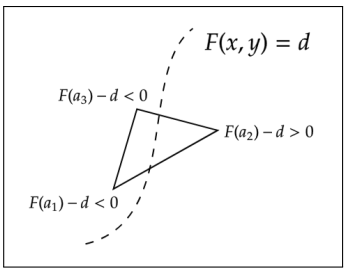

Figure 6: Mesh element condition

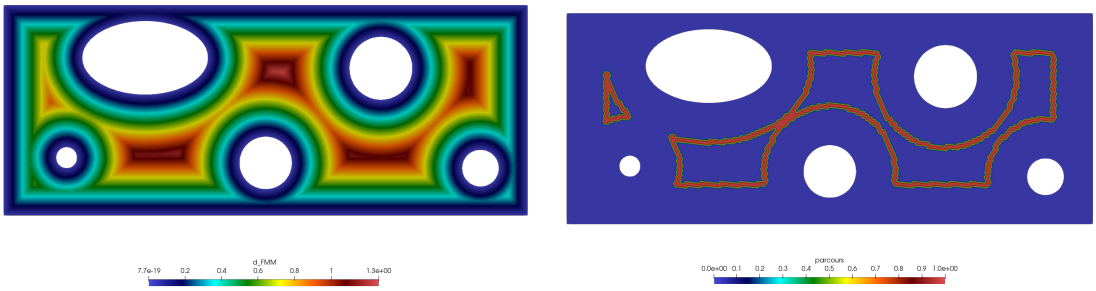

Figure 7: On the left the distance field of a perforated box, on the right the selected element vertices set to 1, and 0 everywhere else.

An element is thus selected if two of its vertices are of opposite signs. We check for this using a simple `checkSigns` method. The barycenters of the selected elements are then stored in a random access container, providing a sample of the level set of the distance function - the precision of which depends on the mesh size h.

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

The BARYCENTERS[3] container is an unordered array of points that is not yet usable as is by the Point-to-point strategy. We propose a second procedure that assembles an ordered sequence of check-points from BARYCENTERS using the swimmer's position as a starting point, going from one position to the closest.

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

We are now able to use the Check Point control strategy procedure to guide the swimmer along the assembled sequence of points. Using a level set of the distance function allows us to enforce a specific distance from the boundary at every instant of the swimmer's trajectory, naturally accounting for walls and obstacles. However, notice on the visual on the right of figure 7 that the selected level set is not connected. Depending on the swimmer's initial position, the check point assembly procedure 3 may not yield expected results.

Other considerations, such as bypasses, motivate the need for robust trajectory planning methods from the sample of a level set. We may also want to consider giving our procedure the option of finding a path along the level set between two given points.

3.3 Geometric graph implementation

We introduce a `C++` implementation of a `Graph` structure, transforming the sample level set into a network of `Nodes`.

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

[3]For dynamic sized containers we use the methods `.push` for insertion and `.pop` for deletion.

Graph

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

Each barycenter is turned into a `Node` and connected within a `Graph`. We will say that two nodes are connected if their euclidean distance falls below a certain threshold ϵ depending on the mesh size h. A conservative approach is to take $\epsilon = 3h$, such that the level set structure is preserved and no corners are cut. Each pair of nodes s_1 and s_2 is also associated with a weight $\omega(s_1, s_2)$, taken to be the euclidean distance between them. In order to determine whether two nodes belong to the same connected sub-graph, and in particular whether they may be connected, a procedure similar to a depth-first search is implemented.

This implementation uses a LIFO type data structure available in the `C++` standard library called a stack. Since every node in enqueued exactly once and every node's adjacent vertices are at most all visited the time complexity of the depth-first procedure is $O(|V| + |E|)$ for V the set of vertices and E the set of edges. Although the depth-first procedure could be used to find a path between two points of the graph, it is not generally optimal. Finding the optimal path not only allows us to deal with bypasses, but also provides smoother paths which are preferable for the Check Point strategy.

Algorithm 4 Connected subgraph search

```
Input: The connection map of a graph G.connect_Map.
Output: Initializes G.nbr_subGraphs and each node's subGraph_id attribute.
unmarkedNodes ← G.data_Nodes
while unmarkedNodes ≠ {∅} do
    root ← getFirstUnmarked(unmarkedNodes, G.data_Nodes)
    unmarkedNodes.pop(root)
    stack.push(root)
    while stack ≠ {∅} do
        next ← stack.top()
        next.subGraph_id = G.nbr_subGraphs
        stack.pop()
        for s ∈ get_neighbours(next) do
            if s ∈ unmarkedNodes then
                unmarkedNodes.pop(s)
                s.subGraph_id = G.nbr_subGraphs
                stack.push(s)
            end if
        end for
        G.nbr_subGraphs++
    end while
end while
```

3.3.1 Dijkstra's algorithm

Let $G = (V, E)$ an oriented graph with positive weight function ω. Dijkstra's algorithm allows us to find the optimal path between two nodes $s_1, s_2 \in V$ with a total cost of $\delta(s_1, s_2)$. We will call s_1 the source term and the initial position of the path.

Let `distance` the mapping from $v \in V$ to a higher bound estimate of the optimal cost between s_1 and v, i.e.:

$$\delta(s_1, v) \leq \texttt{distance}[v].$$

We also set `predecessor` the mapping from v to its predecessor on the estimated optimal path from s_1 to v. The algorithm works by initializing $S = \{\emptyset\}$ and iterating on V until $S = V$. Every time a node v is added to S, we can show that:

$$\texttt{distance}[\text{s}] = \delta(s_1, v).$$

Once `distance` maps every node to its optimal path cost from s_1 we can assemble the full path from s_2 to its predecessor until we reach s_1.

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 = {∅}
Q = V
INITIALIZE(distance, predecessor)
while Q ≠ {∅} do
    v ← GET_MIN(Q)
    Q.pop(v)
    S.push(v)
    for u ∈ get_neighbours(v) do
        if distance[u] > distance[v] + ω(v, u) then
            distance[u] = distance[v] + ω(v, u)
            predeceddor[u] = v
        end if
    end for
end while
```

The INITIALIZE procedure builds the `distance` and `predecessor` maps, setting:

$$\texttt{predecessor}[v] = s_1 \text{ if } v == s_1, \text{ else None},$$
$$\text{and } \texttt{distance}[v] = 0 \text{ if } v == s_1, \text{ else } \infty.$$

The bulk of Dijkstra's time complexity comes from the GET_MIN procedure which determines from `distance` the closest point in Q to s_1. A naive implementation would yield a time complexity of $O(|V|^2 + |E|)$. Instead, our implementation uses a priority queue to find the smallest element in a set in constant $O(1)$ complexity at the expense of logarithmic cost insertion and deletion.

3.3.2 Numerical results

The implementation of Dijkstra's algorithm is validated on several test cases, all obtained from a level set sample. A simple condition check also makes sure beforehand that the two nodes we attempt to connect belong to the same connected subgraph, as it is not necessarily the case within a level set (see figure 7).

If we do not want to necessarily go through the most optimal path we also have the option of interpolating a few points to form a complete loop around the level curve. The following visualization is of a magnetic head swimming simulation around the same level curve that of figure 7.

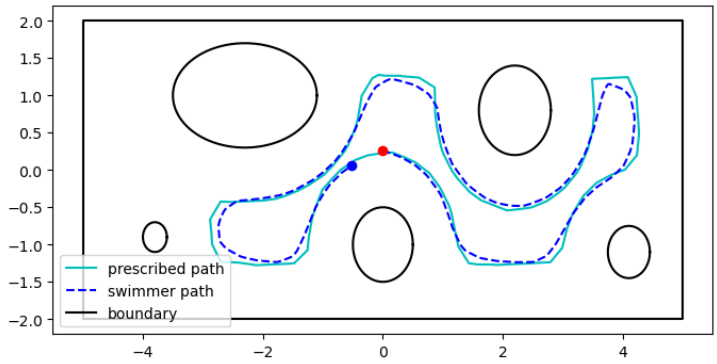

Figure 8: Magnetic control around a level curve. The cyan line is the path found by the Dijkstra interpolation method, and the blue line is the swimmer's path. Red • is the starting position and blue • the final position.

We also tested the performance of Dijkstra's path finding and the **Point-to-point strategy** on a more complex environment. Using the **Body insertion algorithm** developed by PhD. student *Céline Van Landeghem* we were able to place a rigid magnetic head into the discretized fluid domain of a zebra-fish-tail vascular system. Within such irregular bounded fluid environments we might prefer using a super-level set rather than a level set of the distance field.

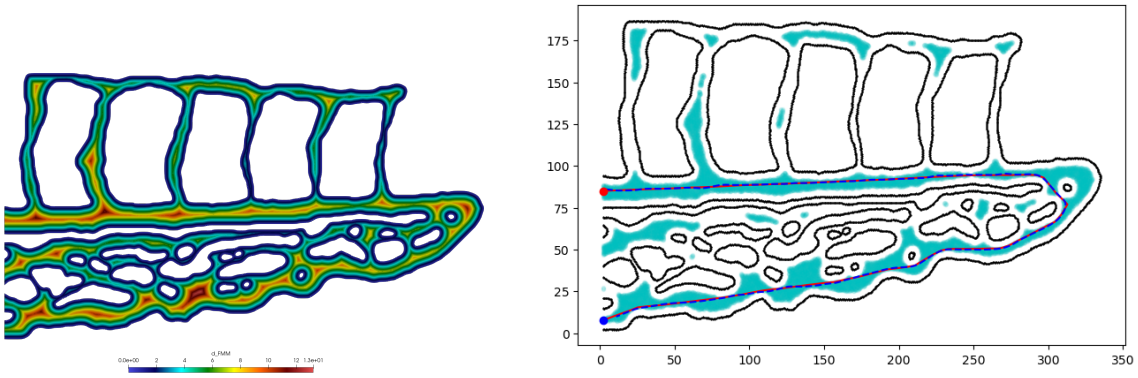

Figure 9: On the **left** the distance field of the zebra-fish tail's 2D side profile. On the **right** a simulation of swimmer control in the zebra-fish tail. The cyan cloud represents the distance super-level set $F(x) \geq 6$. The red line is the prescribed path, while the blue dashed line is the swimmer's actual path.

The swimmer is placed within a Stokes fluid with dynamical viscosity $\mu_{\mathcal{F}} = 10000$. We set a constant uniform magnetic field interacting with the swimmer's magnetic moment $\mathbf{m}$ to produce a torque around its center of mass. The target orientation of the swimmer is given by the **Point-to-point strategy** procedure, enforcing the swimmer to follow the prescribed ordered set of points. At the outlet we impose homogeneous Neumann boundary conditions (no-traction) while on the remaining walls we enforce homogeneous Dirichlet conditions (no-slip).

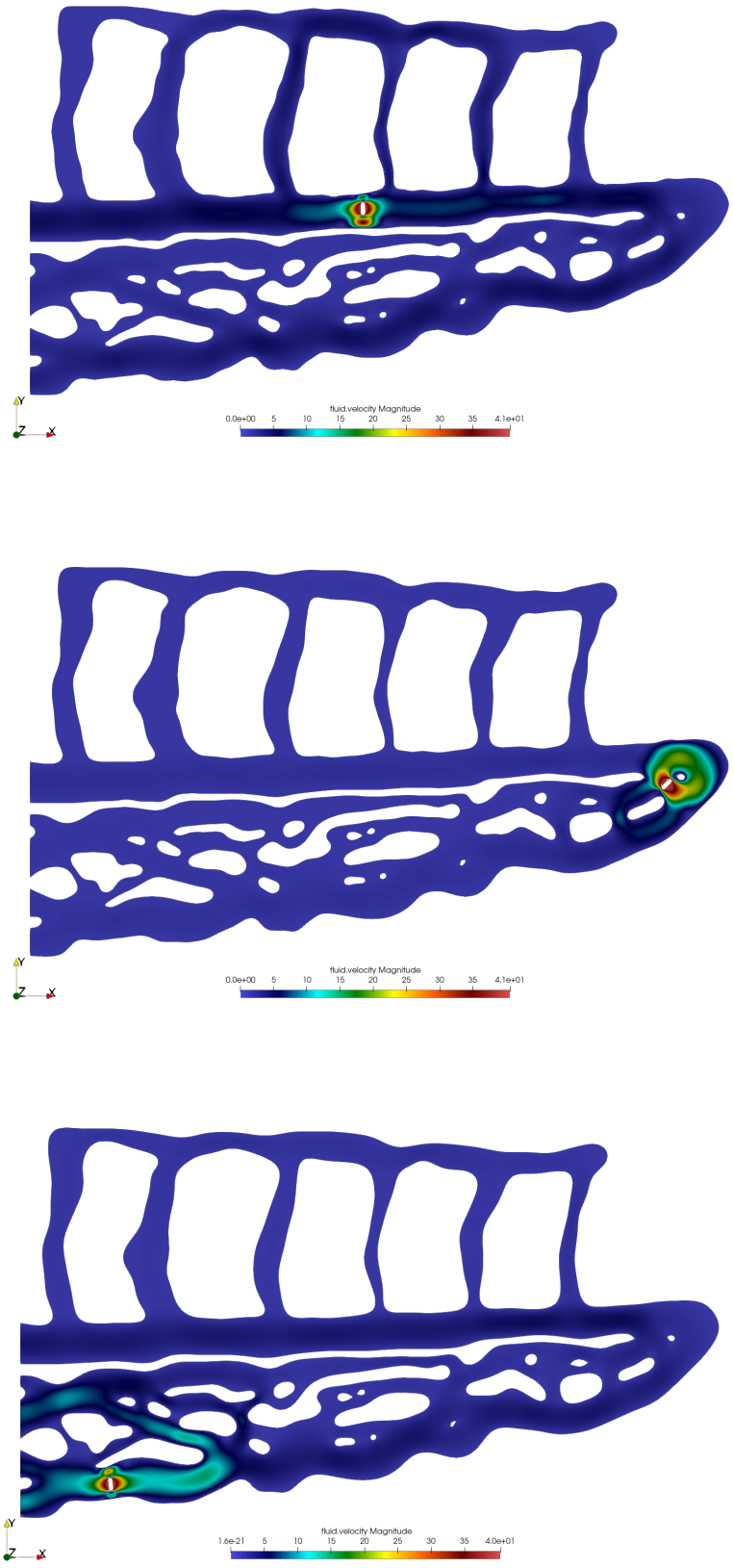

Figure 10: Velocity field Swimmer at different time steps of the simulation, from top to bottom: $t = 800$, $t = 1500$, $t = 3000$.

4 Trajectory control optimization

We previously mentioned how micro-swimming relates to the mathematical theory of *optimal control* and *control theory* in general. In section (2.4) we presented references tackling the controllability of artificial swimmer designs, i.e., can a given swimmer achieve any prescribed position by performing a certain sequence of body deformations, or strokes. Once controllability is established one can ask whether a given displacement could be obtained at a minimal energetic cost, which is a question relevant to *optimal control*. The controllability of magnetic flagellated artificial swimmers is investigated is Alouges et al. [1] using an actuating magnetic field as a control mechanism. An optimum of said control was found and tested within a laboratory setting in El Alaoui-Faris [11]. Optimization of swimmer performances also involves finding optimal body shapes for maximum average speed and efficiency. We reviewed the literature around the 3-Link and 3-Sphere swimmers, but shape improvements for flagellated swimmers are also investigated in Berti [5]. In Palazzolo et al. [21] optimal shapes for helical micro-swimmers were found using *free-form deformation*, a geometric deformation modelling technique, and *Bayesian optimization*, a probabilistic approach to global optimization of high-dimension black-box functions. In the following segment of this report, we provide elements of Bayesian optimization theory. We then introduce an optimal control problem for the trajectory control of the partial magneto-swimmer model (see 2.4) within a simple canal-like geometry to validate the coupling of a `Feelpp` solver and a Bayesian optimization algorithm.

4.1 Bayesian optimization

In this section we describe the surrounding notions of Bayesian optimization, including Gaussian processes and Bayesian regression. We then provide the basic outline of a Bayesian optimization algorithm, before introducing a trust-region based version abbreviated SCBO.

Gaussian Process A *stochastic process* is a family of random variables $(X_s)_{s\in\mathcal{S}}$ indexed by a continuous variable s often interpreted as time. Stochastic processes are often introduced as random functions, generalizing the concept of random vectors. A stochastic process is said to be *Gaussian* (GP) if for any finite subset $A \subset \mathcal{S}$, $(X_s)_{s\in A}$ is a multivariate Gaussian. We also adopt the following notations:

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

Zero-mean Gaussian processes can be completely defined by their covariance function K, which is assumed to be positive semi-definite. Common covariance functions, also called kernels, include:

1. *The Squared Exponential Kernel:*

$$K^l_{\exp}(s,t) = \exp\left(-\frac{\|s-t\|^2}{2l^2}\right).$$

2. *The Matérn Kernel:*

$$K^{\nu,l}_{\mathrm{matern}}(s,t) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\|s-t\|\sqrt{\frac{2\nu}{l}}\right)K_\nu\left(\|s-t\|\sqrt{\frac{2\nu}{l}}\right),$$

with K_ν the modified Bessel function and $\Gamma(\nu)$ the gamma function.

3. *The Rational Quadratic Kernel:*

$$K_{RQ}^{l,\alpha}(s,t) = \left(1 + \frac{\|s-t\|^2}{2\alpha l}\right)^{-\alpha}.$$

All the kernels mentioned above are continuous. As a result, any sample of a Gaussian process, i.e., a deterministic function on $\mathcal{S}$, is also continuous. Moreover, different kernels will offer varying smoothness; for instance, any sample from a Gaussian process with a Rational Quadratic kernel $\mathcal{GP}(0, K_{RQ})$ will produce a $\mathcal{C}^\infty$ function, as opposed to Matérn kernels which can only guarantee continuity, but offer more numerical stability.

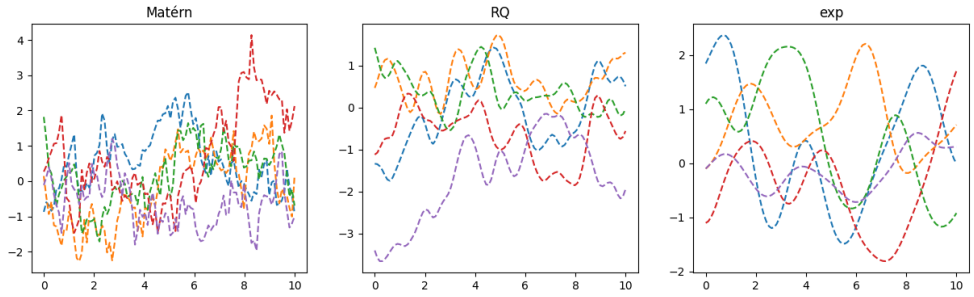

Figure 11: From left to right, 5 samples of zero-mean Gaussian processes with Matérn, Rational Quadratic and Exponential kernels on $\mathcal{S} = [0, 10]$.

To generate instances of random functions as shown in figure 11, we assume $\{x_1, \dots, x_n\}$ a mesh grid discretization of $\mathcal{S}$ and generate a vector $\{y_1, \dots, y_n\}$ sampled from Y a Gaussian random vector such that:

$$Y = [Y_1, \dots, Y_n] \sim \mathcal{N}(0, \Sigma^K), \quad (\Sigma^K)_{i,j} = K(x_i, x_j).$$

This is motivated by the fact that $Y_i = Y_{s=x_i}$ and Y is thus a finite collection from $(Y_s)_{s\in\mathcal{S}} \sim \mathcal{GP}(0, K)$.

GP Regression Let f a function defined over a compact and convex set $\mathcal{S} \in \mathbb{R}^N$. We anticipate subsequent developments and assume f is also computationally expensive to evaluate. We only have access to a finite and restricted set of evaluations $D_k = \{(x_i, y_i)\}_{i=1}^k$ with $x_i \in \mathcal{S}$ and $f(x_i) = y_i$. GP regression starts with declaring a prior distribution over a family of functions f would supposedly belong to. We assume for now:

$$f \sim \mathcal{GP}(0, K),$$

with K an appropriate covariance function. For simplicity, we chose a zero-mean prior, although previous knowledge on the behavior of f could indicate a different non-zero mean function. Given D_n we can now propose a posterior distribution over the family of functions from which those data might have originated:

$$f(x) \mid \{(f(x_i) = y_i)\}_{i=1}^k.$$

Let $\Sigma^K \in \mathcal{M}_{k,k}(\mathbb{R})$ such that $\Sigma^K_{i,j} = K(x_i, x_j)$ and $K(x, \mathcal{X}) = (K(x, x_1), \dots, K(x, x_n)) \in \mathcal{M}_{1,k}(\mathbb{R})$. We can show that Gramacy [17]:

$$f(x) \mid \{(f(x_i) = y_i)\}_{i=1}^k \sim \mathcal{N}(\mu(x), \sigma^2(x)), \tag{11}$$

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

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

Any sample over this function distribution will verify $f(x_i) = y_i$, meaning the training data is interpolated. Interpolation is in general not a good sign and suggests sensitivity to noise and propensity to overfitting. This is due to the fact that we considered up until now a prior estimation without any noise, resulting in the interpolation of the training data and poor generalization. A more robust approach would be to propose the following prior distribution:

$$f(x) = W(x) + \epsilon.$$

With $W \sim \mathcal{GP}(0, K)$ and $\epsilon \sim \mathcal{N}(0, gI_N)$. The hyperparameter g is called the nugget; it is usually obtained via likelihood maximization. We obtain the following posterior Gramacy [17]:

$$W(x) \mid \{(f(x_i) = y_i)\}_{i=1}^{k} \sim \mathcal{N}(\mu(x), \sigma^2(x)), \tag{12}$$

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

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

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

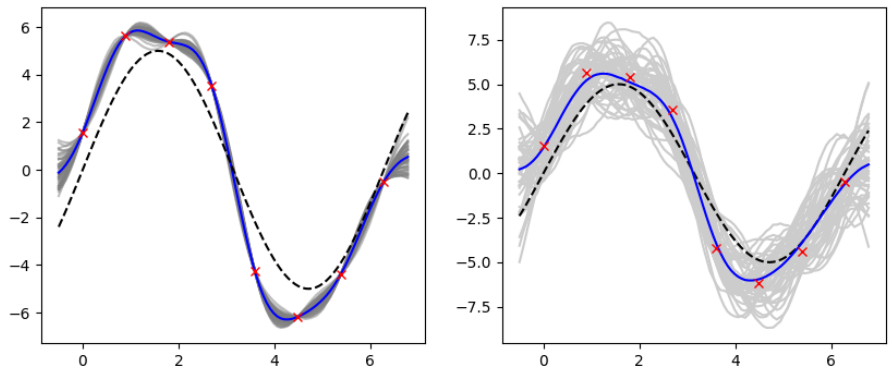

Figure 12: From left to right, interpolating and noisy GP regressions. The red points are the training data, a noisy sample of the function $\sin(5\pi x)$ which is shown in black here. The blue line is the mean of a generated 50 functions, here in gray.

Optimization *GP optimization*, or *Bayesian optimization*, aims at finding the global optimum of a continuous function f we will assume to act as a black-box and to be computationally expensive to evaluate:

$$\text{Find } x^* \text{ s.t } x^* = \arg\min_{x \in \mathcal{S}} f \tag{13}$$

$\mathcal{S}$ is in general a simple but high-dimensional compact set, such as a hypercube. Bayesian optimization works by applying a prior over f, which is then fitted to a posterior distribution using a limited amount of evaluations, acting as training data. The posterior is used to construct an acquisition function that determines a promising batch of points to evaluate next.

Example: We wish to determine the minimum of f a given function. One common acquisition function for the n-th addition to a training data set $D_{n-1} = \{x_1, \dots x_{n-1}\}$ is the maximization of the Expected Improvement (EI), which is defined as:

$$EI(x) = \mathbb{E}\left[\max\{(f_n^* - f[x]), 0\}\right], \quad f_n^* = \min_{m \leq n} f(x_m), \tag{14}$$

with $f[x]$ the posterior distribution constructed from D_{n-1}, hence $f[x] \sim \mathcal{N}(\mu(x), \sigma^2(x))$. Considering we are using GPs as prior distributions, the Expected Improvement can be written as (see *proof*):

$$EI(x) = (f_n^* - \mu(x))\Phi\left(\frac{f_n^* - \mu(x)}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{f_n^* - \mu(x)}{\sigma(x)}\right), \tag{15}$$

with ϕ and Φ respectively the normal standard probability density function and cumulative distribution function. Maximizing the Expected Improvement thus yields the best next point relative to the current posterior distribution. We provide a few iterations of the method below.

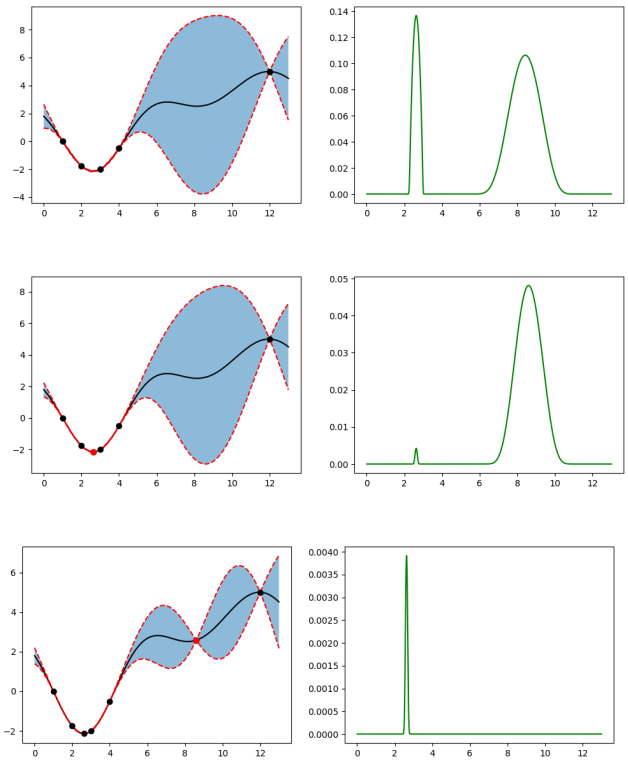

Figure 13: On the left in black the original function and its GP surrogate in blue. On the right in green the Expected Improvement.

Algorithm 6 Bayesian optimization

Input: A function f, n_0 initial points $\in \mathcal{S}$ a hypercube, and N the maximum amount of evaluations.
Output: An estimate of $\arg\min_{x \in \mathcal{S}} f(x)$
$n \leftarrow n_0$
$D_n \leftarrow \{(x_i, y_i)\}_{i=1}^{n}$
while n < N **do**
  Fit $f[x]$ on D_n
  $x_{n+1} \leftarrow$ `acquisitionProcedure()`
  Evaluate $f(x_{n+1}) = y_{n+1}$
  $D_{n+1} = \{(x_i, y_i)\}_{i=1}^{n+1}$
  $n \leftarrow n + 1$
end while

4.2 Scalable Constrained Bayesian Optimization

The Scalable Constrained Bayesian Optimization SCBO Eriksson and Poloczek [13] algorithm aims at solving constrained optimization problems while minimizing evaluations of both the objective and the constraint functions:

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

The acquisition function of SCBO relies on a trust region approach moving through the domain $\mathcal{S}$ as better points are discovered.

1. Trust region sampling A trust region is a hypercube of sides lengths L initially set to L_{init} and centered at a point x^* of maximum utility, meaning:

$$x^* = \begin{cases} \arg\min_{x \in F_c} f(x) & \text{if } F_c \neq \emptyset, \\ \arg\min_{x \in X_0} \sum_{j=1}^{m} \max\{c_j(x), 0\} & \text{if } F_c = \emptyset, \end{cases} \tag{16}$$

with $F_c = \{x_i \in X_0 : c_j(x_i) \leq 0, \forall j : 1, \dots m\}$ and $X_0 = \{x_1, \dots x_k\}$ a set of initial points. We then sample r candidate points within the trust region, using for instance a Sobol sampler.

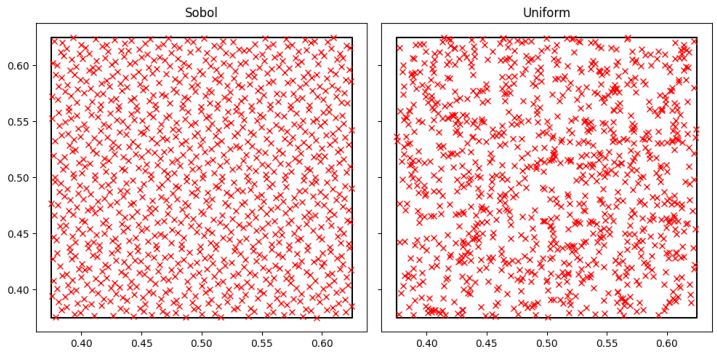

Figure 14: A comparison between Sobol (left) and uniform (right) sampling of a 2D box. Sobol sampling becomes especially relevant for high dimension sampling.

2. Posterior Sampling GP posteriors are fitted onto the available training data D_n, for both the objective and the constraint functions. Initially we have:

$$D_0 = \{x_i, f(x_i), c_1(x_i), \dots, c_m(x_i)\}_{i=1}^{k}.$$

We will note $\hat{f}, \hat{c}_1, \dots, \hat{c}_m$ the GP surrogates. Every iteration we aim at generating a batch of q points to evaluate. For each of these points we sample a realization $\{(\hat{f}(x_i), \hat{c}_1(x_i), \dots, \hat{c}_m(x_i)) : 1 \leq i \leq r\}$ and chose the point of maximum utility. We are left with a batch of q points to evaluate the objective and constraint functions on, which are then added to the training data D_n to create D_{n+1}. This method of sampling is an enhanced version of Thompson sampling Thompson [27] for black-box constraints.

3. Updating the Trust Region We must choose a new trust region to sample using the information we previously gathered. We pick the point of maximum utility in D_{n+1} for the new center. If the new center is better (in terms of utility) than the previous one, we count this as a success and increment n_s the number of consecutive successes. Otherwise, we register a fail and increment n_f, for number of fails. In both cases the other variable is set to 0. We would also have beforehand decided on a fixed number of admissible successes τ_s and fails τ_f. Let L_n be the current length of the trust region.

1. If $n_s == \tau_s$, there is little chance the optimum belongs to the current region, thus we extend its size to cover more ground:

$$L_{n+1} = \min\{2L_n, L_{\max}\}, \quad n_s = 0.$$

2. If $n_f = \tau_f$, there is a pretty good chance the optimum is close to the current center, and we shrink the region's size:

$$L_{n+1} = L_n/2, \quad n_f = 0.$$

The algorithm terminates once the current size of the trust region falls below a certain threshold $L_{\min}$, or once we have depleted our maximum number of iterations. The `Python` implementation of `SCBO` used in this project was provided by *Lucas Palazzolo* and adapted from BoTorch [7].

4.3 Control of a magneto-swimmer

In this section we present a flexible model for the optimal control of the partial magneto-swimmer model. The objective is to validate the coupling of our magnetic head in a Stokes fluid `Feelpp` solver and the provided implementation of `SCBO`. Let us start by defining a reference path $X_{\text{ref}} : [0,1] \mapsto \mathbb{R}^d$, in our case, $d = 2$, with $X_{\text{final}} = X_{\text{ref}}(1)$. The swimmer's trajectory, X_u, is the solution of a control problem:

$$\begin{cases} \dot{X}_u(x) = f(t, X_u(t), u(t)), \\ X_u(0) = X_0. \end{cases} \quad u : [0,T] \mapsto \mathbb{R}^m \tag{17}$$

We seek to minimize the following objective function, which is computationally expensive to evaluate as it amounts to solving the complete trajectory of the swimmer at each evaluation:

$$\begin{aligned} C(u,\gamma) = \int_0^T & \|X_n(t) - X_{\text{ref}}(\gamma(t))\|_Q \\ & + \|X_u(T) - X_{\text{final}}\|_R \\ & + \|u(t)\|_S \; dt \end{aligned} \tag{18}$$

The cost matrix Q, R and S can be adjusted depending on the problem at hand. The function u is a certain control we wish to obtain an optimum of in order to minimize the objective function:

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

We also have the option of optimizing the parametrization of the reference path γ, we chose to be forward only:

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

We are left with an infinite dimension optimization problem of a particularly computationally expensive function. In order to numerically solve it we approach our optimization spaces with finite dimension B-Splines.

B-Splines B-Splines are linear combinations of positive piece-wise polynomials defined over a time interval $[t_0, t_n]$ with $\mathcal{T} = \{t_0, \ldots, t_n\}$ a non-decreasing sequence of time points. Let $B_{i,k}(t)$ the i-th B-Spline with $i \in [0, n-k-1]$, a piece-wise k degree polynomial defined by the following recurrence relation:

$$\begin{aligned} B_{i,0}(t) &= \begin{cases} 1 & \text{if } t_i \leq t \leq t_{i+1}, \\ 0 & \text{else,} \end{cases} \\ B_{i,k}(t) &= \omega_{i,k}(t) B_{i,k-1} + (1 - \omega_{i+1,k}(t)) B_{i+1,k-1}, \\ \omega_{i,p}(t) &= \begin{cases} \frac{t - t_i}{t_{i+p} - t_i} & \text{if } t_{i+p} \neq t_i, \\ 0 & \text{else.} \end{cases} \end{aligned}$$

We provide a few properties of B-Splines Piegl and Tiller [22]:

1. For every $k \geq 0$:
$$\sum_{i=0}^{n-k-1} B_{i,k} = 1, \quad \forall t \in [t_k, t_{n-k}).$$
2. $B_{i,k}$ is reduced to zero outside $[t_i, t_{i+k+1})$.
3. $B_{i,k}$ is $\mathcal{C}^\infty$ over (t_i, t_{i+1}), and $\mathcal{C}^{p-s}$ at every point t_j with s the multiplicity of t_j in $\{t_i, \ldots, t_{i+p+1}\}$.

Example: Let $[0, 5]$ a time interval. We set the following time nodes:
$$\mathcal{T} = \{0, 0, 0, 1, 2, 3, 4, 5, 5, 5\}.$$

$\mathcal{T}$ is such that we can define 7 B-Splines $\{B_{i,2}\}_{i=0\ldots6}$ that are piece-wise polynomials of degree 2 and $\mathcal{C}^1$ at every node except 0 and 5 on which they equal to 0 if $i \neq 0$ or 6 and 1 otherwise. This property implies the interpolation of the first and last coefficients.

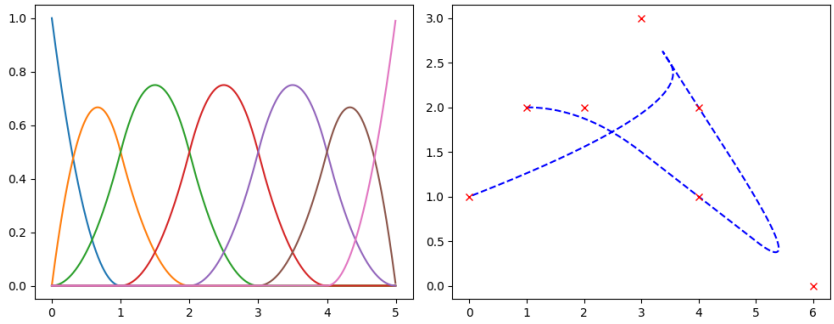

Figure 15: On the left the 7 B-Splines and the on the right a random 2D linear combination of them in blue with the coefficients in red.

γ and u will be approached by B-Splines defined over a uniform time discretization $\mathcal{T}_u$ and $\mathcal{T}_\gamma$:
$$\begin{cases} u_j \approx \sum_{i=1}^{N_u^j} P_i^{u_j} B_{i,k}^{u_j}(t), \\ \gamma \approx \sum_{i=1}^{N_\gamma} P_i^\gamma B_{i,k}^\gamma(t). \end{cases}$$

The optimization spaces thus become:
$$\tilde{\mathcal{U}} = \left\{ \left(\sum_{i=1}^{N_u^j} P_i^{u_j} B_{i,k}^{u_j}(t) \right)_{j \in \{1,\ldots m\}} \text{ s.t. } m_j \leq P_i^{u_j} \leq M_j \right\},$$
$$\tilde{\Gamma} = \left\{ \sum_{i=1}^{N_\gamma} P_i^\gamma B_{i,k}^\gamma(t) \text{ s.t., for } \Delta P_i^\gamma = P_{i+1}^\gamma - P_i^\gamma : \begin{cases} \Delta P_i^\gamma \geq 0 \\ P_0^\gamma = 0 \\ 1 - \epsilon \leq \sum \Delta P_i^\gamma \leq 1 \end{cases} \right\}.$$

The condition on $\tilde{\mathcal{U}}$ enforces $m_j \leq u_j \leq M_j$, while the condition on $\tilde{\Gamma}$ ensures γ is a bijection from $[0, 1]$ to $[0, T]$. Finally, the optimization problem we aim to solve reads:
$$\underset{P_i^\gamma, P_i^{u_j}}{\arg\min}\, C(\gamma, u) \text{ s.t. } \sum P_i^\gamma \leq 1, -\sum P_i^\gamma \leq \epsilon - 1. \tag{19}$$

We use a computational trick to avoid optimizing with equality constraints by setting a small ϵ.

4.4 Contribution

The `Python` implementation of `SCBO` provided by *Lucas Palazzolo* comes with full B-Spline support to easily optimize over spline coefficients. The objective here was to implement the coupling of the `Feelpp` solver used in the first part of this report to simulate, within a full PDE setting, the swimming of a rigid body in a Stokes fluid. In this section we go through the practical objectives and challenges encountered during this process.

Python binding The `Feelpp` development suite provides `Python` binders to easily integrate its toolboxes within a more user-friendly framework. Using a custom branch of the `Feelpp` GitHub repository Feel++ [14] acting as an external library we were able to provide flexible access to the magnetic swimmer solver from our `SCBO` application. This custom branch includes additional control and data processing methods written in `C++`, which provide for instance access to the swimmer's trajectory and control input management.

Geometry generation `Feelpp` applications are fully configurable using a `.json` input file, which in turn points to a `.geo` or `.msh` file describing the geometry of the domain. In our application, configuration and geometry files are automatically generated and customizable by the user, from boundary conditions to the swimmer's shape and mesh size.

Swimmer problem An instance of the already provided `Problem` class was also implemented for the magneto-swimmer model, which provides the objective function and other necessary set-up methods for trajectory optimization using `SCBO`. Additionally, a post-processing pipeline is proposed for easy visualization.

Reproducibility To validate the coupling we propose 4 test cases of trajectory control within simple geometries, inspired by the experiments carried out during the second semester project. Each test case comes with its own parameter file and an introductory Markdown note.

4.5 Numerical results

Control splines As mentioned previously, we propose 4 test cases for the optimization of the control of a magnetic head swimmer (see partial magneto-swimmer model 2.4). For all 4 of these simulations we approach the optimal controls using B-Splines by optimizing with `SCBO` the coefficients of a 2 dimensional control spline in the basis $(\{B_{i,2}^{u_j}\}_{i=0}^{9})_{j=1}^{2}$, such that:

$$u_1(t) \approx \sum_{i=0}^{9} P_i^{u_1} B_i^{u_1}(t) \text{ and } u_2(t) \approx \sum_{i=0}^{9} P_i^{u_2} B_i^{u_2}(t).$$

with u_1 the target velocity, which is an input quantity of the **Adaptive linear force** procedure, and u_2 a target angular velocity, which in the case of a magnetically controlled object corresponds the angular velocity of the uniform external magnetic field $\mathbf{B}$, provided by:

$$\mathbf{B}(t) = B \begin{pmatrix} \cos(\theta(t)) \\ \sin(\theta(t)) \end{pmatrix}.$$

Since we do not have direct control over the fields actual angular velocity $\dot{\theta}$, we integrate it using a simple Euler scheme:

$$\theta(t_{n+1}) = \theta(t_n) + dt\ u_2(t).$$

Simulations are carried out over the time interval $[0, 1]$, and we chose for now not to optimize the reference path parametrization γ.

Based on previous knowledge of what angular and linear speed a swimmer is able to assume without encountering numerical instabilities, we set the following conditions:

$$P_i^{u_1} \in [0, 6], \quad P_i^{u_2} \in [-2\pi, 2\pi].$$

As a result the optimization space becomes:

$$\mathcal{S} = [0, 6]^{10} \times [-2\pi, 2\pi]^{10}.$$

Cost parameters Each experiment is accompanied by a table summarizing the main optimization parameters: the cost matrices R and Q, the number of iterations n_{iter}, the batch size s_{batch} and the final cost C_{final}. We chose to set the cost matrix S to 0, focusing on trajectory accuracy rather than energy efficiency. The swimmer's state at time t is represented by a 3 dimensional vector $X(t)$ with:

$$X(t) = (x(t), y(t), \theta(t))^T.$$

Thus, the norm associated with a cost matrix Q is given by:

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

Geometry Additionally, we suppose a rectangular swimmer of lengths 0.5×1 immersed in a fluid bounded by a box, enforcing homogeneous Dirichlet conditions (no-slip) on all 4 sides of the domain.

4.5.1 Reaching a target point

We first aim at minimizing the distance between the swimmer and a given position in (x_1, y_1). We also enforce periodic orientation such that $\theta_1 = \theta_0 = 0$.

Optimization parameters 4.5.1

Q	R	n_{iter}	s_{batch}	C_{final}
0	$100 \times I_3$	250	10	7.46

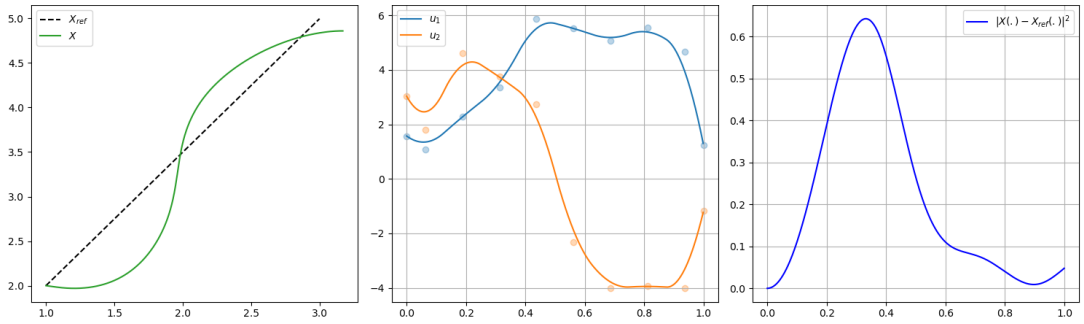

Figure 16: **Left:** The swimmer path in green, the reference path black. **Middle:** The spline curves and their coefficients. **Right:** The cost error between the reference path and the swimmer's trajectory.

The initial state $X_0 = (x_0, y_0, \theta_0)$ is given by $(1, 2, 0)$ and the target state is set to $X_{\text{ref}}(1) = (3, 5, 0)$. The fluid domain is a box of lengths 5×7. Overall, the optimization algorithm approaches reasonably well the expected configuration in less than 2500 simulations ($n_{\text{iter}} \times s_{\text{batch}}$).

4.5.2 Avoiding an obstacle

The objective here is similar to the first case, except this time we place an obstacle in the direct way of the swimmer. The fluid domain is box of lengths 7×5.

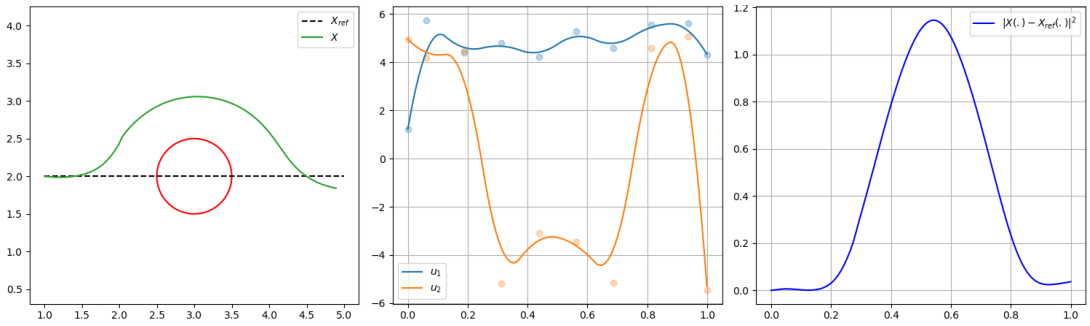

Figure 17: The obstacle is indicated in red on the **left**. It is a circle perforation of radius 0.5 centered at $(3, 2)$.

Optimization parameters 4.5.2

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

`SCBO` successfully finds a path around the obstacle put in the way of the swimmer. In this case we do not simply try to minimize the distance between two end states: we found results to be more satisfactory if the objective function also comprises the L2 error between the swimmer trajectory and the reference path. In the above table J_3 indicates the following matrix:

$$J_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

4.5.3 Following a curve

This time we aim at minimizing the distance between the swimmer and a given prescribed path with moderate curvature.

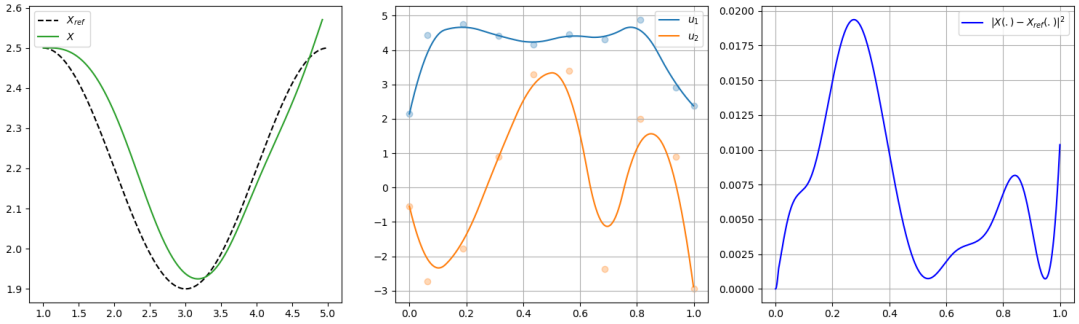

Figure 18: The prescribed path has the analytical expression of a sine wave.

Optimization parameters 4.5.3

Q	R	n_{iter}	s_{batch}	C_{final}
$100 \times J_3$	0	250	10	7.18

Once again our Bayesian optimization algorithm yields quite satisfactory controls and the swimmer follows the overall shape of the path. However, we stress the relatively low curvature of the prescribed trajectory, we purposely not raised too much in order to avoid numerical instabilities and mesh singularities.

4.5.4 Overcoming boundary effects

During the second semester project, we noticed the effects of boundaries on a swimmer's trajectories. A swimmer was placed in a deliberately asymmetrical environment, and when driven by a linear force aligned with its orientation, tended to deviate from its expected trajectory towards the boundary it was the closest to. We wish to see if Bayesian optimization can determine control able to overcome these effects. In this simulation the swimmer is placed 1 spatial length away from bottom boundary, a distance we have seen to be enough for boundary effects to manifest.

Optimization parameters 4.5.4

Q	R	n_{iter}	s_{batch}	C_{final}
$100 \times J_3$	0	200	10	2.66×10

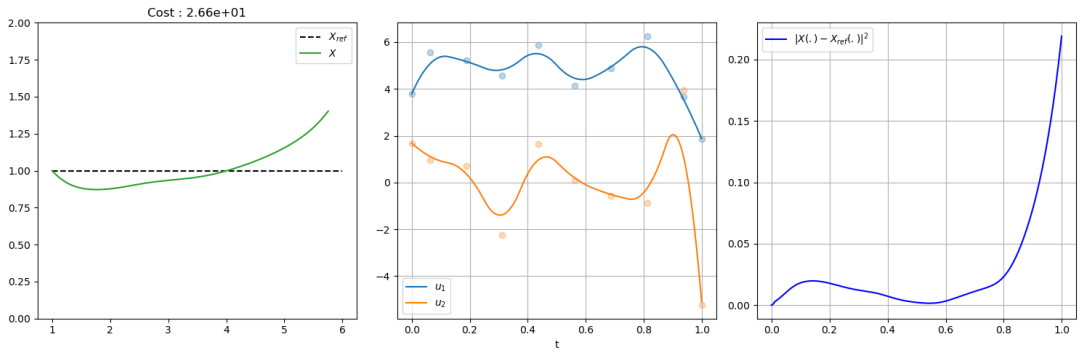

Figure 19: The swimmer is tasked to go in a straight line while close to the bottom boundary.

Although the swimmer does not end up deviating towards the bottom boundary, the control determined by `SCBO` seems to have overcorrected the swimmer's orientation.

Conclusion & Perspectives Overall, the current implementation of a Stokes swimmer solver coupled with a Bayesian optimization algorithm provided satisfying results for simple trajectory control problems. Bayesian optimization finds purpose specifically when the cost function is computationally expensive to evaluate, and where most classic methods fail. Although most test cases presented above did end up fulfilling our expectations, our model still lacks robustness and predictability. Moreover, due to time constraints and technical difficulties, we were not able to conduct a full performance study of the implementation. In particular, the computational burden of each simulation run turned out to be a major limitation: a single optimization procedure could require several days of computation, which prevented us from exploring the parameter space in sufficient depth.

Future work should therefore focus on reducing the computational load of the coupled framework. One possibility would be to parallelize each batch evaluation, reducing the cost of an iteration to one simulation. Another option is to use alternative numerical methods, like the Resistive Force Theory, to converge quickly towards an optimum, and use the latter as a starting batch for a more accurate but expensive solver. A more general approach is to consider multi-fidelity solvers Do and Zhang [9], combining high and low accuracy physical models to construct GP based multi-fidelity surrogates.

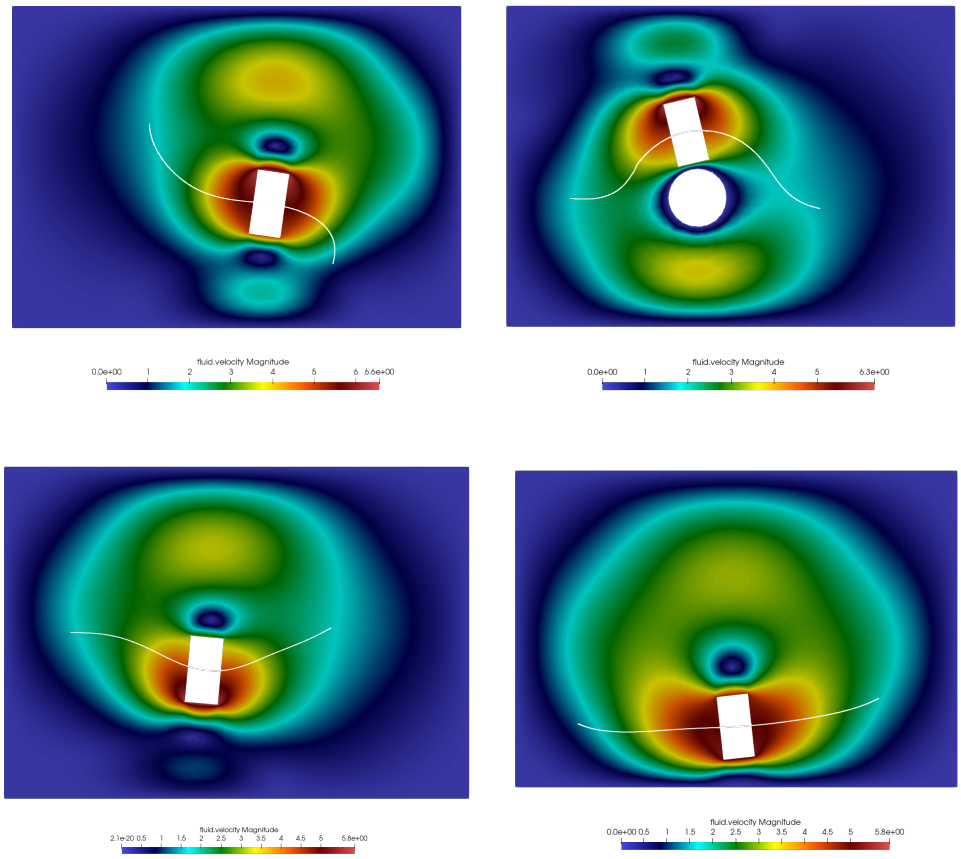

Figure 20: Simulations using the optimal controls at time $t = 0.5$ - **top-left** (a) (rotated 90°), **top-right** (b), **bottom-left** (c), **bottom-right** (d). The swimmer and its trajectory are shown white.

5 Collective swimmer motion

Understanding the collective motion of particles in a fluid is a critical aspect of designing effective micro-swimming solutions to industrial or medical problems. There is only so much a single swimmer can do before being limited by its small size and volume, and large swarms of microscopic robots have proven to be viable solutions, outperforming single swimmer models for tasks such as delivery, cleaning or manipulation. Most microscopic swarm models involve a collection of self-propelling particles, or active particles, often disregarding hydrodynamic interactions for the benefit of a simpler, more intuitive model, delivering good results in terms of collective emergent behaviors Bechinger et al. [3]. A common and well documented instance of active particles immersed in a Stokes fluid is the squirmer model, which supposes a two-dimensional system of N self-propelled spherical particles. We also mention the passive particle model, which does not make the assumption of self-propulsion. Instead, the particles are advected using external power sources, such as a magnetic field. For instance, in the paper Yang et al. [30], researchers used a swarm of black iron oxide (Fe_3O_4) magnetic nanoparticles to investigate different autonomy levels, from manual navigation using an actuation field, to fully autonomous navigation, featuring a CNN type architecture trained to output a swarm distribution from an image of the swarm's surrounding environment.

Owning to time constraints, the objective of this part of the internship remained rather modest. We propose a generalization of the magnetic control implementation of a single magneto-swimmer to a swarm of magnetic objects, with the addition of numerical tools for swarm generation and dipole-dipole interactions.

5.1 Swarm generation & magnetic control

The first step was to extend the current magnetic control implementation in `Feelpp` to multiple rigid bodies. Each body marker is associated with its own translational force $\mathbf{F}_\epsilon$ to simulate self-propulsion. Orientation control using an external magnetic field however is common to the swarm. Collisions between swimmers are handled by the `Feelpp` library and generalized to complex shaped bodies in Landeghem [18]. While studying swarm motion we can realistically expect to simulate hundreds of swimmers with the considerate amount of precision allowed by the Finite Element Method, which we should mention is not the most effective approach to model very large swarm behaviors. The generation of configuration files thus becomes rapidly tedious, which is why we propose `Python` routines for swarm generation, along with data visualization pipelines.

Swarm formations We propose 3 swimmer formation methods with customizable parameters.

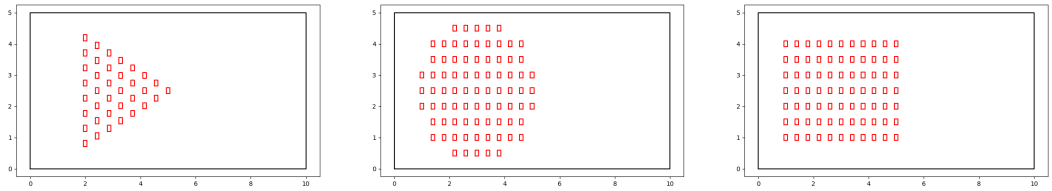

Figure 21: Swarm formations triangle, circle, and rectangle.

Appropriate `.json` and `.geo` files are generated with parameters easily configurable, from the swimmers' shape - ellipsoid, disk, rectangle - to the space between them and the environment's dimensions.

Numerical experiments We first observe the behavior of a simple 3 swimmers configuration, where 3 rigid bodies of rectangular shapes are positioned in a horizontal symmetry, with a canal-like fluid environment: homogeneous Dirichlet on the bottom and top walls, homogeneous Neumann on the right and left walls. The swimmers are set to a reach and maintain a velocity of 1 using the **Adaptive linear force** procedure, with no uniform magnetic source to correct their trajectories. Their initial orientations are set to $\theta_0 = 0$.

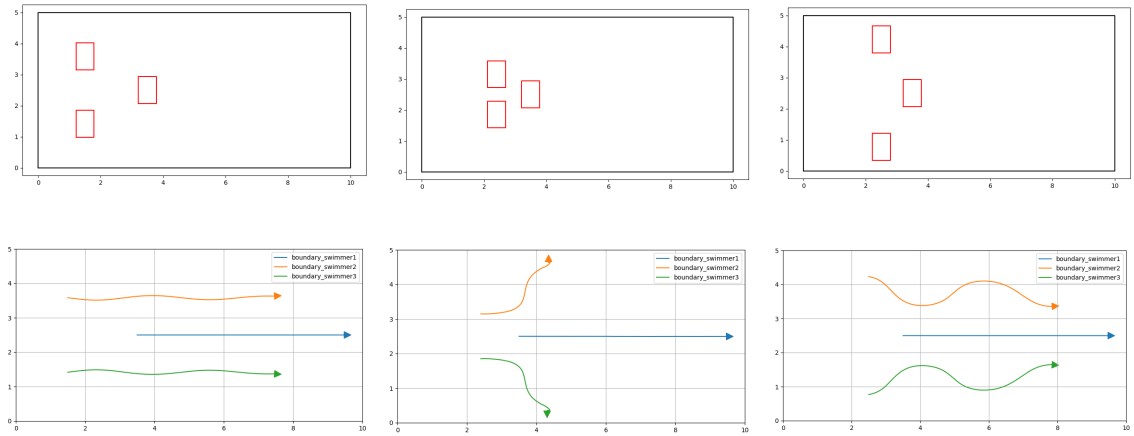

Figure 22: We test 3 configurations varying in spread. The tight layout appears to be the least stable with the top and bottom swimmers ending up faces to the walls, while the other two manage to keep a relative formation integrity with some oscillations.

We then examine the performance of different swarm formations within a simple canal-like geometry. Each swarm is composed of 21 disk swimmers of uniform sizes and densities. No magnetic field is used to correct the swimmers orientations, and we use an alternative procedure to enforce a specific "directional" speed of amplitude 1. Up until now the **Adaptive linear force** procedure would only helps reach and maintain a specific velocity amplitude, in any direction. For swarms, we found that using this directional speed yielded more satisfactory results.

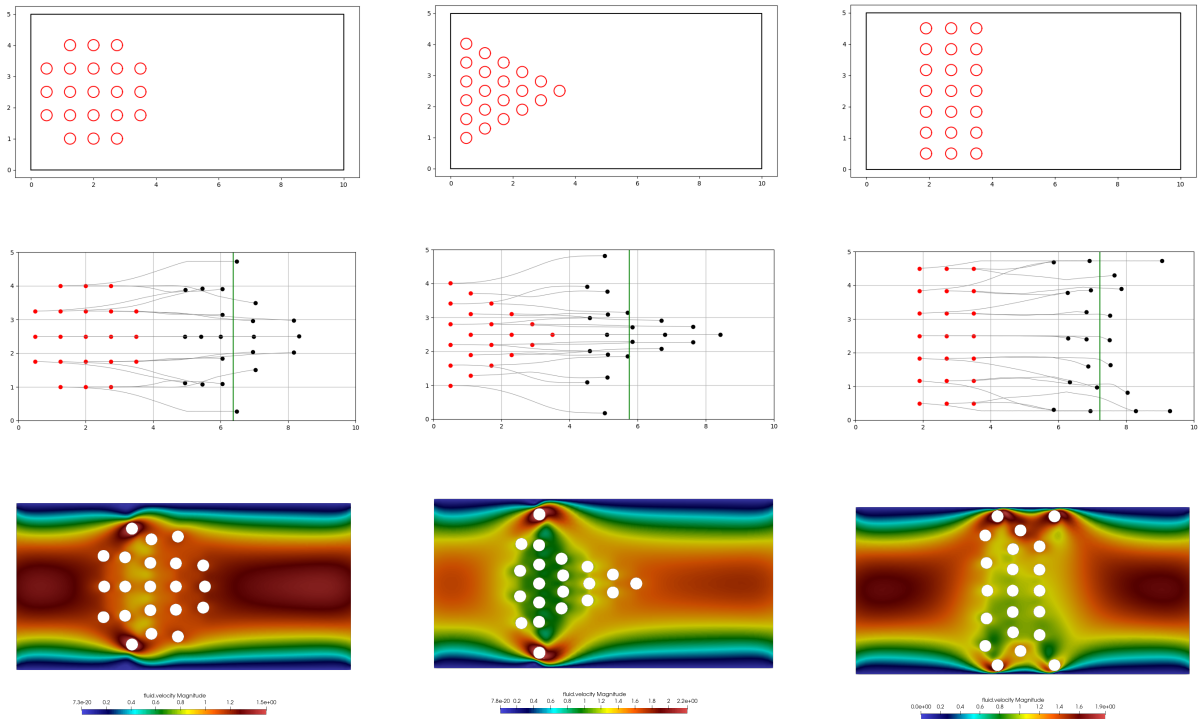

Figure 23: **First row:** The initial configuration of disk swimmers in boxes 10 by 5. All formations are aligned at $x = 3.5$. **Second row:** The trajectory in gray of each swimmer, their starting points in red and end points in black. The mean distance travelled in the horizontal direction is indicated in green. **Third row:** Velocity fields mid-simulation ($t = T/2$).

Algorithm 7 Directional Velocity

Input: The time step dt, current and previous positions X_n and X_{n+1}, current and previous orientations θ_n and θ_{n+1}.

Output: The directional velocity (in θ_n) v.

$$R \leftarrow \begin{pmatrix} \cos(\theta_n) & -\sin(\theta_n) \\ \sin(\theta_n) & \cos(\theta_n) \end{pmatrix}$$

$D_n \leftarrow R(X_{n+1} - X_n)$

$v \leftarrow D_n\,[0]\,/dt$

Notice how these configurations struggle to keep their original shape. In particular, swimmers on the outskirt of the swarm appear more susceptible of deviating from the collective trajectory. In order to mitigate these tendencies we introduce an external magnetic field, that activates and increases its amplitude as more swimmers leave an expected circular sector.

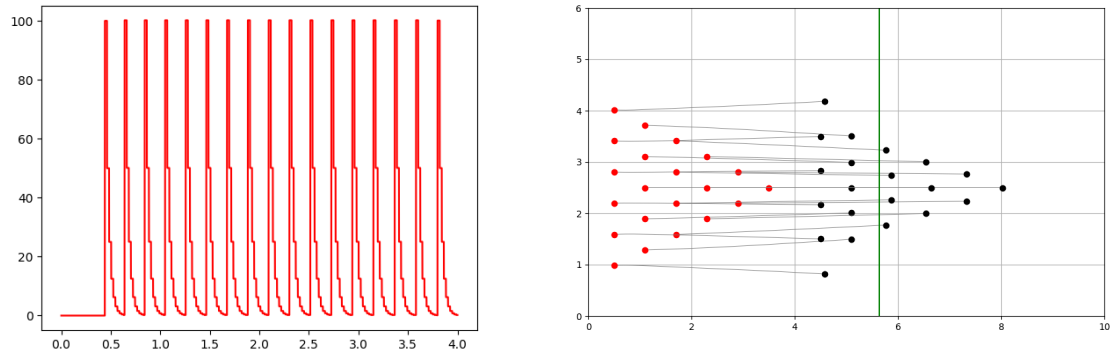

Figure 24: On the **left**, the amplitude of a periodically activating magnetic field. On the **right**, a triangle formation with the same configuration as in 23, adjusted by the magnetic field on the left.

5.2 Dipole-dipole Interactions

Physical model If we wish to model our swimmers as a swarm of magnetic particles we need to consider the magnetic forces and torques resulting from their mutual interactions. Whether these dynamics are negligible or not in the presence of external magnetic sources however, is beyond the scope of this report. In the following section we provide a simple physical model for modelling these magnetic interactions, inspired by Xu et al. [29].

Let $\{\mathbf{r}_1, \dots, \mathbf{r}_n\}$ the positions of a swarm of n magnetic particles provided with magnetic moments $\{\mathbf{m}_1, \dots, \mathbf{m}_n\}$. We will note:

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

If $\mathbf{B}_j$ is the magnetic field produced by the j-th magnetic particle, we have:

$$\mathbf{B}_j(\mathbf{r}_i) = \frac{\mu_0}{4\pi|\mathbf{r}_{ij}|^3}\left[3(\mathbf{m}_j \cdot \hat{\mathbf{r}}_{ij})\hat{\mathbf{r}}_{ij} - \mathbf{m}_j\right], \tag{20}$$

the expression of $\mathbf{B}_j$ at $\mathbf{r}_i$ assuming $\mathbf{r}_{ij}$ is not too small, with μ_0 the magnetic constant. As a result, the torque experienced by i following its interaction with j is given by:

$$\mathbf{T}_i^j = \mathbf{m}_i \times \mathbf{B}_j.$$

The magnetic force produced by the dipole j onto i is:

$$\mathbf{F}_{ij} = \nabla_{\mathbf{r}_i}(\mathbf{m}_i \cdot \mathbf{B}_j), \tag{21}$$

which gives us when developed (see *proof*):

$$\mathbf{F}_{ij} = \frac{3\mu_0}{4\pi|\mathbf{r}_{ij}|^4}\left(\mathbf{m}_j(\mathbf{m}_i \cdot \hat{\mathbf{r}}_{ij}) + \mathbf{m}_i(\mathbf{m}_j \cdot \hat{\mathbf{r}}_{ij}) + \hat{\mathbf{r}}_{ij}(\mathbf{m}_i \cdot \mathbf{m}_j) + 5\hat{\mathbf{r}}_{ij}(\mathbf{m}_j \cdot \hat{\mathbf{r}}_{ij})(\mathbf{m}_i \cdot \hat{\mathbf{r}}_{ij})\right). \tag{22}$$

Numerical experiment We test our implementation on a simple case comprising 3 rigid disk swimmers placed in canal-like fluid environment, with their initial orientations set to 0. All 3 of them are provided with a magnetic moment of magnitude 100 aligned with their x coordinate. For this simulation, we do not assume self-propulsion; external magnetic fields could be used to produce a collective net displacement, such as the ones used in Yang et al. [30]. In our case however, we focus on collective behaviors under uniform magnetic fields - we use a rotating one with constant angular velocity:

$$\mathbf{B}(t) = 500 \begin{pmatrix} \cos(2\pi t) \\ \sin(2\pi t) \end{pmatrix}$$

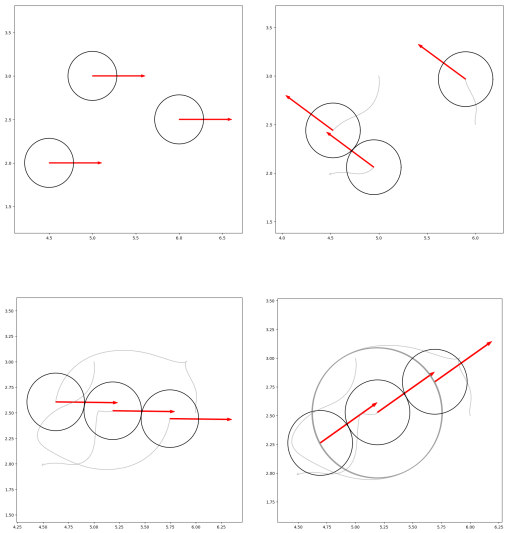

Figure 25: Snapshots at different simulation times - **top-left:** $t = 0$, **top-right** $t = 0.05T$, **bottom-left**: $t = 0.1T$, **bottom-right**: $t = T$. Swimmers are indicated in black disks, their trajectories in gray and their orientations with red arrows. Dipoles appear to end up in a stable configuration, rotating as a whole with not net displacement.

6 Conclusion

During this internship, we reviewed several challenges of micro-swimming simulations using a high-fidelity finite element solver. Specifically, we investigated the performances of a path planning strategy based on the distance function, which yields for any point in a mesh the distance from the fluid boundary. This allowed us to extract mesh elements and nodes to construct optimal paths using Dijkstra's algorithm with boundary distance constraints, and simulate the control of a magnetic rigid swimmer within complex environments. We then proposed an implementation for the coupling of a `Feelpp` solver with a Bayesian optimization algorithm called SCBO, of which we tested the performance using 4 simple test cases for the control of a magnetic micro-swimmer. Despite an important computational cost, the optimization algorithm managed to find promising controls. Perspectives are discussed in 4.5.4. Finally, we introduced numerical tools in `Feelpp` for modelling swarms of magneto-swimmers, including an implementation of dipole-dipole interactions.

Appendix A Proofs

1. Proof of Expected Improvement

$$
\begin{aligned}
EI(x) &= \mathbb{E}\left[\max\{0, f_n^* - f[x]\}\right] \\
&= \int_{-\infty}^{+\infty} \max\{0, f_n^* - t\} \frac{1}{\sigma(x)\sqrt{2\pi}} \exp\left(-\frac{(t-\mu(x))^2}{2\sigma(x)^2}\right)\, dt \\
&= \int_{-\infty}^{f_n^*} (f_n^* - t) \frac{1}{\sigma(x)\sqrt{2\pi}} \exp\left(-\frac{(t-\mu(x))^2}{2\sigma(x)^2}\right)\, dt \\
&= \int_{-\infty}^{\frac{f_n^*-\mu(x)}{\sigma(x)}} (f_n^* - \sigma(x)u - \mu(x))\phi(u)\, du \qquad u = \frac{t-\mu(x)}{\sigma(x)}, \sigma(x) > 0 \\
&= (f_n^* - \mu(x))\Phi\left(\frac{f_n^* - \mu(x)}{\sigma(x)}\right) - \sigma(x) \int_{-\infty}^{\frac{f_n^*-\mu(x)}{\sigma(x)}} u\phi(u)\, du \\
&= (f_n^* - \mu(x))\Phi\left(\frac{f_n^* - \mu(x)}{\sigma(x)}\right) + \sigma(x)\left[\frac{1}{\sqrt{2\pi}} \exp\left(\frac{-u^2}{2}\right)\right]_{-\infty}^{\frac{f_n^*-\mu(x)}{\sigma(x)}} \\
&= \boxed{(f_n^* - \mu(x))\Phi\left(\frac{f_n^* - \mu(x)}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{f_n^* - \mu(x)}{\sigma(x)}\right)}
\end{aligned}
$$

2. Proof of dipole-dipole force We use the following identities:

$$
\begin{aligned}
\nabla_r \frac{1}{|r|^n} &= -\frac{nr}{|r|^{n+2}} \\
\nabla_r (m \cdot r) &= m
\end{aligned}
$$

We thus have:

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

We also have:

$$
\begin{aligned}
\nabla_{\mathbf{r}_i} \left[\frac{(\mathbf{m}_j \cdot \mathbf{r}_{ij})(\mathbf{m}_i \cdot \mathbf{r}_{ij})}{|\mathbf{r}_{ij}|^5}\right] &= (\mathbf{m}_i \cdot \mathbf{r}_{ij}) \nabla_{\mathbf{r}_i} \left(\frac{(\mathbf{m}_j \cdot \mathbf{r}_{ij})}{|\mathbf{r}_{ij}|^5}\right) + \frac{(\mathbf{m}_j \cdot \mathbf{r}_{ij})}{|\mathbf{r}_{ij}|^5} \nabla_{\mathbf{r}_i} (\mathbf{m}_i \cdot \mathbf{r}_{ij}) \\
&= \mathbf{m}_j \frac{(\mathbf{m}_i \cdot \mathbf{r}_{ij})}{|\mathbf{r}_{ij}|^5} + 5\mathbf{r}_{ij} \frac{(\mathbf{m}_j \cdot \mathbf{r}_{ij})(\mathbf{m}_i \cdot \mathbf{r}_{ij})}{|\mathbf{r}_{ij}|^7} + \mathbf{m}_i \frac{(\mathbf{m}_j \cdot \mathbf{r}_{ij})}{|\mathbf{r}_{ij}|^5}
\end{aligned}
$$

As a result:

$$
\begin{aligned}
\mathbf{F}_{ij} &= \frac{3\mu_0}{4\pi} \left(\mathbf{m}_j \frac{(\mathbf{m}_i \cdot \mathbf{r}_{ij})}{|\mathbf{r}_{ij}|^5} + 5\mathbf{r}_{ij} \frac{(\mathbf{m}_j \cdot \mathbf{r}_{ij})(\mathbf{m}_i \cdot \mathbf{r}_{ij})}{|\mathbf{r}_{ij}|^7} + \mathbf{m}_i \frac{(\mathbf{m}_j \cdot \mathbf{r}_{ij})}{|\mathbf{r}_{ij}|^5} + \mathbf{r}_{ij} \frac{(\mathbf{m}_i \cdot \mathbf{m}_j)}{|\mathbf{r}_{ij}|^5}\right) \\
&= \boxed{\frac{3\mu_0}{4\pi|\mathbf{r}_{ij}|^4} \left(\mathbf{m}_j(\mathbf{m}_i \cdot \hat{\mathbf{r}}_{ij}) + \mathbf{m}_i(\mathbf{m}_j \cdot \hat{\mathbf{r}}_{ij}) + \hat{\mathbf{r}}_{ij}(\mathbf{m}_i \cdot \mathbf{m}_j) + 5\hat{\mathbf{r}}_{ij}(\mathbf{m}_j \cdot \hat{\mathbf{r}}_{ij})(\mathbf{m}_i \cdot \hat{\mathbf{r}}_{ij})\right)}
\end{aligned}
$$

References

[1] François Alouges et al. “Can Magnetic Multilayers Propel Artificial Microswimmers Mimicking Sperm Cells?” In: *Soft Robotics* 2 (2015), pp. 117–128.

[2] Alouges, F. “Low Reynolds number swimming and controllability”. In: *ESAIM: Proc.* 41 (2013), pp. 1–14. DOI: 10.1051/proc/201341001.

[3] Clemens Bechinger et al. “Active Particles in Complex and Crowded Environments”. In: *Reviews of Modern Physics* 88.4 (Nov. 2016). DOI: 10.1103/revmodphys.88.045006.

[4] Howard Berg and Richard Berry. “E. Coli in motion”. In: *Physics Today* 58 (Feb. 2005), pp. 64–65. DOI: 10.1063/1.1897527.

[5] Luca Berti. “Numerical methods and optimization for micro-swimming”. Theses. Université de Strasbourg, Dec. 2021.

[6] Luca Berti et al. “Modelling and finite element simulation of multi-sphere swimmers”. en. In: *Comptes Rendus. Mathématique* 359.9 (2021), pp. 1119–1127. DOI: 10.5802/crmath.234.

[7] BoTorch. *Scalable Constrained Bayesian Optimization (SCBO) Tutorial*. https://botorch.org/docs/tutorials/scalable_constrained_bo/. Accessed: 2025-08-13; Last updated: August 12, 2025. 2025.

[8] Ada-Ioana Bunea and Rafael Taboryski. “Recent Advances in Microswimmers for Biomedical Applications”. In: *Micromachines* 11 (2020). ISSN: 2072-666X. DOI: 10.3390/mi11121048.

[9] Bach Do and Ruda Zhang. “Multifidelity Bayesian Optimization: A Review”. In: *AIAA Journal* 63.6 (2025), pp. 2286–2322. DOI: 10.2514/1.J063812.

[10] Rémi Dreyfus et al. “Microscopic Artificial Swimmers”. In: *Nature* 437 (Nov. 2005), pp. 862–5. DOI: 10.1038/nature04090.

[11] Yacine El Alaoui-Faris. “Modélisation et contrôle optimal de micro-nageurs magnétiques”. Thèse de doctorat dirigée par Pomet, Jean-Baptiste et Regnier, Stéphane Mathématiques Université Côte d’Azur 2020. PhD thesis. 2020.

[12] J Elgeti, R G Winkler, and G Gompper. “Physics of microswimmers—single particle motion and collective behavior: a review”. In: *Reports on Progress in Physics* 78.5 (Apr. 2015), p. 056601. DOI: 10.1088/0034-4885/78/5/056601.

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

[14] Feel++. *Feel++: Finite Element Embedded Language and Library in C++*. https://github.com/feelpp/feelpp. Accessed: 2025-08-15. 2011–2025.

[15] Laetitia Giraldi, Pierre Martinon, and Marta Zoppello. “Optimal design of Purcell’s three-link swimmer”. In: *Phys. Rev. E* 91 (Feb. 2015), p. 023012. DOI: 10.1103/PhysRevE.91.023012.

[16] Roland Glowinski. “Finite element methods for incompressible viscous flow”. In: *Numerical Methods for Fluids (Part 3)*. Vol. 9. Handbook of Numerical Analysis. Elsevier, 2003, pp. 3–1176. DOI: https://doi.org/10.1016/S1570-8659(03)09003-3.

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

[18] Céline Van Landeghem. “Micro-natation dans des environnements complexes”. Manuscrit de thèse de doctorat, non publié, Université de Strasbourg, 2025. 2025.

[19] Eric Lauga and Thomas R Powers. “The hydrodynamics of swimming microorganisms”. In: *Reports on Progress in Physics* 72.9 (Aug. 2009), p. 096601. DOI: 10.1088/0034-4885/72/9/096601.

[20] Ali Najafi and Ramin Golestanian. “Simple swimmer at low Reynolds number: Three linked spheres”. In: *Phys. Rev. E* 69 (June 2004), p. 062901. DOI: 10.1103/PhysRevE.69.062901.

[21] Lucas Palazzolo et al. “Parametric shape optimization of flagellated microswimmers using Bayesian techniques”. In: *Phys. Rev. Fluids* 10 (3 Mar. 2025). DOI: 10.1103/PhysRevFluids.10.034101.

[22] Les Piegl and Wayne Tiller. *The NURBS book*. Berlin, Heidelberg: Springer-Verlag, 1995. ISBN: 3540550690.

[23] Ian Proudman and J. R. A. Pearson. “Expansions at small Reynolds numbers for the flow past a sphere and a circular cylinder”. In: *Journal of Fluid Mechanics* 2 (1957), pp. 237–262. DOI: 10.1017/S0022112057000105.

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

[25] K. Jagajjanani Rao et al. “A Force to Be Reckoned With: A Review of Synthetic Microswimmers Powered by Ultrasound”. In: *Small* 11.24 (2015), pp. 2836–2846. DOI: https://doi.org/10.1002/smll.201403621.

[26] 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.

[27] William R. Thompson. “On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples”. In: *Biometrika* 25.3/4 (1933), pp. 285–294. (Visited on 08/20/2025).

[28] Anthony Wachs et al. “7 - Modeling of short-range interactions between both spherical and non-spherical rigid particles”. In: *Modeling Approaches and Computational Methods for Particle-Laden Turbulent Flows*. Ed. by Shankar Subramaniam and S. Balachandar. Academic Press, 2023, pp. 217–264. ISBN: 978-0-323-90133-8. DOI: https://doi.org/10.1016/B978-0-32-390133-8.00019-0.

[29] Zhiqiang Xu et al. “Simulation Study on the Motion of Magnetic Particles in Silicone Rubber-Based Magnetorheological Elastomers”. In: *Mathematical Problems in Engineering* 2019.1 (2019), p. 8182651. DOI: https://doi.org/10.1155/2019/8182651.

[30] Lidong Yang et al. “Autonomous environment-adaptive microrobot swarm navigation enabled by deep learning-based real-time distribution planning”. In: *Nature Machine Intelligence* 4.5 (May 2022), pp. 480–493. DOI: 10.1038/s42256-022-00482-8.

[31] Marta Zoppello et al. “Modeling and steering magneto-elastic micro-swimmers inspired by the motility of sperm cells”. In: *Atti della Accademia Peloritana dei Pericolanti - Classe di Scienze Fisiche, Matematiche e Naturali* 96.S3 (2018).