Université de Strasbourg

UNIVERSITÉ DE STRASBOURG

msii ÉCOLE DOCTORALE

ÉCOLE DOCTORALE MATHÉMATIQUES, SCIENCES DE L'INFORMATION ET DE L'INGÉNIEUR

Institut de Recherche Mathématique Avancée

THÈSE présentée par :

Luca BERTI

soutenue le : **13 décembre 2021**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline/ Spécialité : MATHÉMATIQUES APPLIQUÉES

Numerical methods and optimization for micro-swimming

THÈSE dirigée par :

M. PRUD'HOMME Christophe	Professeur, Université de Strasbourg

RAPPORTEURS :

M. MAÎTRE Emmanuel	Professeur, Université de Grenoble-Alpes
Mme SALMON Stéphanie	Professeure, Université de Reims

AUTRES MEMBRES DU JURY :

M. BEC Jérémie	Directeur de recherches, CNRS
M. BERGMANN Michel	Chargé de recherches, Inria Bordeaux Sud-Ouest
M. FORMAGGIA Luca	Professeur, Politecnico di Milano
Mme GIRALDI Laetitia	Chargée de recherches, Inria Sophia-Antipolis Méditerranée

---

INVITÉE :

Mme. SZOPOS Marcela	Professeure, Université de Paris

A Mirko e Wanda.

Acknowledgments

First of all, I would like to thank my advisors: Christophe Prud'homme and Laetitia Giraldi. Thank you for giving me the opportunity to work with you on such an interesting subject and for your constant support.

Merci Christophe pour tes conseils, ton aide tout au long de ma thèse, la liberté que tu m'as laissé et les opportunités d'encadrer des étudiants. Merci Laetitia pour tes conseils, tes propositions de recherche et de m'avoir inclus dans plusieurs collaborations et projets. Je tiens à remercier aussi Vincent Chabannes pour les discussions que nous avons pu avoir, le support informatique constant avec *Feel++*, et pour avoir permis aux idées de cette thèse de devenir réalisables.

I would like to thank Stéphanie Salmon and Emmanuel Maître for accepting to review this thesis. I thank as well Jérémie Bec, Michel Bergmann, Luca Formaggia and Marcela Szopos for being part of the examination committee.

Je remercie aussi Mickael Binois pour son soutien et sa collaboration dans une partie des mes travaux.

Je tiens à remercier François Alouges et Matthieu Aussal, qui m'ont accueilli à l'École Polytechnique pendant quelques semaines lors de ma première année de thèse, pour m'avoir accompagné à la découverte de la méthode des éléments de frontières.

I would like to thank Kenta Ishimoto, who welcomed Laetitia and myself for a two week workshop in Kyoto, and for the numerous video-calls that followed. Thank you for your insights, your patience and your support.

I would like to thank the colleagues of the equipe MoCo, especially Yannick, Clémentine, Joubine, Victor and Emmanuel, for organising seminars and other activities that maintain the research group active.

Je veux remercier aussi tous les doctorants (anciens ou non) du département de mathématiques, et en particulier François, Philippe, Thibault, Fréd, Marie, Lorenzo, Lukas, Laura, Pierre, Marie, Romain, JB, Djibril, Sonia, pour leur bienveillance et leur support.

I would also like to thank the amazing team of the Addal association and the PhD associative life in Strasbourg, Monica, Dimitra, Hanine, Lesia, Annabela, Yuvna, Merveille, Riccardo, Filipa.

I would like to thank the good friends that made the stay in France less lonely, Gabriele, Thao, Alejandro, Valeriia, Michèle, Léon, Garance, Claire and Clémence.

Grazie agli amici di sempre, Giovanni, Caterina, Nicola, Laura, Sara, Daniele, Silvio, Giorgia, Giorgio. Grazie per essermi stati vicini in tutti questi anni e per le lunghe chiacchierate.

Grazie alla mia famiglia per il sostegno perenne in tutte le mie scelte.

Et pour terminer, merci à toi, Tanguy, pour m'avoir accompagné pendant ces années de thèse.
Merci pour ta créativité, tes attentions et ton soutien constant (et pour les figures dans l'introduction).

Contents

Résumé

L'étude de la natation à la micro échelle (micro-natation) a subi une évolution majeure dans les dernières années. Son but était initialement de comprendre le mouvement de micro-organismes flagellés (comme les spermatozoïdes, ou les bactéries) et extraire les principes qui décrivent la nage à cette échelle et qui la différencient de la nage basée sur les contributions inertielles. Maintenant, à ces derniers, des investigations basées sur la théorie de la commande optimale et l'investigation de la forme de ces nageurs ont été établies. Ces dernières sont motivées par l'application médicale des micro-robots nageurs, qui est déjà une réalité dans des laboratoires de recherche grâce à des études en vitro et vivo. De nombreux défis sont encore présents et motivent la recherche dans le domaine, tels que le contrôle de ces nageurs, leur comportement dans des endroits confinés, la meilleure forme pour les robots. La simulation numérique est un outil qui peut aider à résoudre ces questions, et elle peut agir complémentairement aux expériences de laboratoire, qui peuvent à leur tour en valider les résultats.

Pendant cette thèse nous nous sommes concentrés à la fois sur la description numérique de la micro-natation et sur ses interactions avec les problèmes susmentionnés de forme optimale. Pour ce faire, nous avons considéré deux méthodes numériques pour résoudre les équations fluides, à savoir la méthode des éléments finis et la méthode des éléments de frontière. Ces méthodes ont été appliquées à des nageurs flagellés différents, en tenant compte de leur stratégie propulsive et de l'influence de cette dernière sur un domaine fluide éventuellement maillé. En fait, parmi les nageurs à flagelle, il est possible d'identifier deux types de propulsions, qui se basent sur la propagation d'ondes hélicoïdales ou d'ondes planaires. Le premier type de nageur, lors de la rotation de ses flagelles, produirait une déformation du maillage fluide qui comporterait un remaillage très fréquent, tandis que dans le deuxième cas le remaillage pourrait être fait moins souvent. De plus, en termes de modélisation, la flagelle du premier type de nageurs est souvent considérée comme une hélice rigide, alors que le deuxième cas est modélisé avec une flagelle déformable. C'est donc pour cela que le premier type de nageurs a été simulé à l'aide des éléments de frontière et le deuxième à l'aide des éléments finis, car seulement cette dernière méthode requiert le maillage du domaine fluide.

Voici le contexte dans lequel cette thèse est développée et les contributions qui peuvent être mises en avant:

- L'étude de la forme optimale de nageurs à inspiration bactérienne, multi-flagellés, qui en optimise la vitesse de propulsion tout en favorisant la nage dans une direction prescrite. La méthode numérique utilisée est la méthode des éléments de frontière pour la résolution des équations de Stokes couplées avec les contraintes d'auto-propulsion. Cette méthode est basée sur la fonction de Green et la formulation intégrale des équations de Stokes et sur le fait qu'on peut obtenir la solution du problème fluide en discrétisant juste la frontière du nageur. Les systèmes algébriques qui sont obtenus par cette méthode sont denses (car la fonction de Green couple chaque nœud du maillage avec chaque autre nœud) et de petite

taille comparés avec les systèmes creux produits par une discrétisation aux éléments finis. Cela implique que différentes techniques de stockage et de résolution sont requises pour cette méthode. Différents cas test issus de la littérature sont mis en œuvre pour valider la méthode numérique de résolution du problème fluide. Le problème d'optimisation est ensuite attaqué en utilisant la méthode d'optimisation Bayesienne, qui n'a jamais été utilisée dans le cadre de l'optimisation de forme en micro-natation. Cette méthode procède en construisant un meta-modèle du problème d'optimisation de forme en se basant sur un budget fini d'évaluations, car la résolution du problème fluide reste coûteuse et augmente avec le nombre de flagelles. Le principe de cette méthode est d'utiliser les évaluations de la fonction coût pour construire, à l'aide de la régression Gaussienne, un modèle probabiliste de cette fonction, dont moyenne et variance sont mises à jour chaque fois qu'une nouvelle mesure est disponible. De plus, grâce à ces estimations, il est possible de choisir le point successif où la fonction coût sera évaluée dans les régions où la variance, l'incertitude du modèle, est majeure.

- L'étude du mouvement de nageurs à inspiration bactérienne, bi-flagellés, pour lesquels on modélise l'élasticité des jonctions à l'aide de ressorts de torsion. Ici nous utilisons la Resistive Force Theory (RFT), une méthode approchée qui est tout de même valable dans l'étude d'organismes mono-flagellés, sous la contrainte de négliger la contribution des interactions entre les différents parties qui composent le nageur, à savoir queue et corps cellulaire. Elle permet d'arriver à des formulations analytiques qui clarifient l'impact des différents paramètres qui décrivent le nageur et elle a été utilisée pour des études de commande optimale de nageurs robotiques validés avec des expériences de laboratoire. Dans une première partie nous nous concentrons sur la modélisation d'un nageur à deux flagelles à l'aide de la RFT pour mettre en valeur l'effet que la position des jonctions élastiques et la forme du corps cellulaire ont sur la vitesse de propulsion du nageur. Cette approximation est valable lorsque les queues sont éloignées et les effets hydrodynamiques entre elles sont négligeables. Dans un deuxième temps, nous considérons un modèle BEM dans lequel nous laissons les queues libres de se réorienter suite à l'interaction entre le fluide et les jonctions élastiques. Cette dernière partie présente la modélisation du problème et un exemple de validation. Le but futur sera de comparer les résultats du modèle BEM avec les résultats issus de la Resistive Force Theory.

- L'utilisation de la librairie *Feel++* pour la simulation aux éléments finis du problème de la micro-natation. La méthode des éléments finis est rarement utilisée dans l'étude de la micro-natation car souvent lourde et coûteuse pour les études normalement menés dans ce domaine. Par contre, si l'objectif est d'étudier des fluides biologiques complexes (à partir du sang, jusqu'à la matrice extracellulaire) et leurs interactions avec des micro-nageurs, ou bien modéliser l'élasticité des nageurs en utilisant un modèle décrit par des équations aux dérivées partielles, la méthode des éléments finis regagne son intérêt aussi pour de la micro-natation. La méthode choisie pour décrire l'interaction fluide-nageur est la méthode Arbitrary-Lagrangian-Eulerian, qui découple en partie l'évolution du milieu fluide de l'évolution du domaine computationnel. Cette méthode, déjà présente dans la librairie grâce au travail de V. Chabannes pour la simulation des écoulements sanguins, a été augmentée en ajoutant la partie qui décrit le mouvement d'un corps rigide immergé dans le fluide, nécessaire pour décrire la position du nageur à tout moment. Cette description de la nage oblige aussi à considérer les problèmes liés au déplacement du maillage et à sa reconstitution lors d'une déformation importante. C'est dans cette optique que l'évaluation de la qualité du maillage et l'inclusion du remaillage à l'aide des librairies *MMG* et *parMMG* ont été considérées. Dans cette partie de la thèse nous avons considéré différents méthodes

qui peuvent être utilisées pour décrire le mouvement d'un nageur en nous basant sur la littérature: soit on connaît la forme exacte du nageur à chaque instant temporel, grâce à une description analytique de sa forme; soit cette forme est décrite par un modèle simplifié (par exemple des ODEs qui, une fois résolues, donnent la forme du nageur à chaque instant); soit la forme dérive d'un modèle basé sur des EDPs. Chacun de ces cas demande un traitement particulier, spécialement si un remaillage est déclenché, et on les a considérés dans les développements proposés.

Introduction

The study of swimming at the micro-scale (micro-swimming) has undergone a major evolution in the last few years. Its aim was initially to understand the motion of swimming microorganisms (like spermatozoa, or bacteria) in order to extract the principles that describe swimming at low Reynolds number [104, 45, 49, 76]. More recently, micro-swimming has been studied using the tools of optimal control theory [6, 8, 48, 9, 50] and the shapes of swimmers have been investigated [107, 128, 100, 129, 79]. These studies are motivated by the medical application of swimming micro-robots, which are already a reality in research laboratories [135, 79, 61]. Many challenges are still present and motivate research in the field, such as the control of these robotic swimmers, their behaviour in confined areas, their shape. Numerical simulations are a tool that can help study these questions, and they often act as complement to laboratory experiments. This thesis aligns with this research direction, by studying the problem of swimming at the micro-scale, with emphasis on the shape of micro-swimmers and control of their motion, using different numerical methods tailored to the cases under study.

We begin our discussion by presenting the peculiarities of low Reynolds number flows and how they affect the swimming behaviour of micro-organisms. We continue by presenting the different classes of swimmers that are capable of propulsion at low Reynolds number and specify the ones we concentrate on. Later, we introduce the mathematical and numerical models that have been developed to study (micro-)swimming and highlight the ones we focus on. Finally, the contributions of this thesis are detailed.

Low Reynolds number flow

At the microscopic scale, the propulsion of swimming organisms is essentially due to viscous forces [121, 104]. This means that micro-swimmers take only advantage of viscous drag to propel themselves, and that the effective propulsion techniques are fundamentally different from those experienced by larger swimmers, for which inertial forces overpower viscosity effects [27]. It is possible to highlight the difference between these swimming regimes by considering the Navier-Stokes equations, that describe the space-time evolution of a Newtonian fluid. If we define by $\mathcal{F} \subseteq \mathbb{R}^d$ the fluid domain, $d = 2, 3$, $\rho > 0$ the fluid density, $\mu > 0$ the fluid viscosity, $p : \mathcal{F}\times]0, T] \to \mathbb{R}$ and $u : \mathcal{F}\times]0, T] \to \mathbb{R}^d$ the fluid pressure and velocity, $f : \mathcal{F}\times]0, T] \to \mathbb{R}^d$ the external volume forces, Navier-Stokes equations in the Eulerian frame read as

$$\begin{cases} \rho\partial_t u + \rho(u \cdot \nabla u) = -\nabla p + \mu\Delta u + f, & \text{in } \mathcal{F}, \\ \nabla \cdot u = 0, & \text{in } \mathcal{F}. \end{cases}$$

Fluid flow in Navier-Stokes equations can be driven by at least two factors: by prescribing a non-zero external force f or via suitable boundary conditions on $\partial\mathcal{F}$ that enforce non-zero

surface tensions or fluid velocities. In the following, we suppose that $f = 0$ and that fluid flow is produced by prescribing boundary conditions on $\partial\mathcal{F}$.

In order to measure the importance of viscous and inertial forces, we proceed with the non-dimensionalisation of the Navier-Stokes equations by using the characteristic parameters of a typical micro-swimmer. The non-dimensional version of the fluid variables is obtained by scaling time, velocity and pressure using the characteristic time scale T, speed U and length L of an average micro-organism. We use $T = \frac{U}{L}$, i.e. the ratio of the characteristic speed U and length L of the swimmer, to scale the time variable. The scaling procedure gives the non-dimensional variables

$$\bar{t} = \frac{t}{T}, \quad \bar{u} = \frac{u}{U}, \quad \bar{p} = \frac{p}{\frac{\mu U}{L^2}},$$

where $\frac{\mu U}{L^2}$ is the pressure scaling factor for viscosity-dominated flows. The non-dimensional version of Navier-Stokes equations reads as

$$\begin{cases} Re(\partial_{\bar{t}}\bar{u} + \bar{u} \cdot \nabla\bar{u}) = -\nabla\bar{p} + \Delta\bar{u}, & \text{in } \mathcal{F}, \\ \nabla \cdot \bar{u} = 0, & \text{in } \mathcal{F}, \end{cases}$$

where Re is the Reynolds number

$$Re = \frac{\rho U L}{\mu}.$$

Reynolds number gives a measure of the relative intensity of inertial and viscous effects in a Newtonian fluid, as it can be expressed as the ratio of their characteristic values, $\rho U^2 L^2$ and $\mu U L$ respectively

$$Re = \frac{\rho U^2 L^2}{\mu U L}.$$

If we choose realistic values for $U \approx 30\,\mu\text{m} \cdot \text{s}^{-1}$ and $L \approx 1\,\mu\text{m}$ [104], based on a bacterium propelling in water, where $\rho \approx 10^3\,\text{kg} \cdot \text{m}^{-3}$ and $\mu \approx 10^{-3}\,\text{kg} \cdot \text{m}^{-1} \cdot \text{s}^{-1}$, we obtain $Re \approx 3 \cdot 10^{-5}$. This value of Re confirms what we stated in the beginning of the section, i.e. at the micro-scale inertial contributions are negligible when compared to viscous forces. Such a small Reynolds number motivates to push further the simplification of the fluid model and to consider the limiting case where $Re \to 0$ in the fluid equations. The resulting equations for Newtonian fluids are the Stokes equations

$$\begin{cases} -\nabla p + \mu\Delta u = 0, & \text{in } \mathcal{F}, \\ \nabla \cdot u = 0, & \text{in } \mathcal{F}. \end{cases}$$

Representation of Stokes solution in unbounded domain

Stokes equations are linear in fluid pressure and velocity, and the absence of time derivatives makes them time-reversible. This means that any sum of solutions to the equations is still a solution, and that it is possible to have a fluid flow as well as its time inversion obeying the Stokes equations. A consequence of the linearity of Stokes equations in terms of modelling is that the superposition principle holds, making it possible to express any solution of the Stokes equations as a superposition of fundamental solutions. In particular, there exist boundary integral representations relying on the existence of a Green kernel and the superposition principle, that allow to express a complex flow as a combination of several fundamental solutions obtained

with point forcing terms [68, 49]. For example, in free-space, the fundamental solution of Stokes problem is given in terms of the Green kernel

$$G(x,y) = \frac{1}{8\pi\mu}\left(\frac{\mathbb{I}}{\|x-y\|} + \frac{(x-y)\otimes(x-y)}{\|x-y\|^3}\right),$$

where $\mathbb{I}$ is the identity tensor and $\otimes$ denotes the Kronecker product $(a\otimes b)_{ij} = a_i b_j$. Based on the superposition principle, and the related integral representation formulas, there exists a number of approximate descriptions of flows at low Reynolds number that are generated by thin beating filaments, like Resistive Force theory [49, 82] and Slender Body theory [66]. Using this methodology, one can also describe the flow surrounding moving ellipsoidal bodies [68].

Due to the invariance of Stokes equations by time inversion, micro-swimming gaits must satisfy a "non-reciprocity" constraint to ensure propulsion, as highlighted by Purcell in his seminal talk [104]. Reciprocal motions produce fluid fields that are characterised by successive concatenations of Stokes flows and their time inversions. By taking as an example the stroke pattern of a scallop, which is reciprocal due to its only degree of freedom, Purcell showed that a swimmer is incapable to propel if the non-reciprocity constraint is not satisfied. This observation, now known as the Scallop theorem, shows the fundamental difference between inertia-based and low Reynolds number swimming: while in the first case the acceleration of body deformation plays a role in the propulsion, in the second case only the sequence of geometric shapes is determinant for motion. Despite this constraint on the flow field that micro-swimmers produce, there exists a plethora of shapes and deformation strategies that ensure net propulsion, as we will show in the next section.

Swimming at low Reynolds number

Biological swimmers

Biological micro-swimmers are cells that are able to move in a fluid medium by continuously changing their shape. Most of them developed thin appendices, i.e. flagella and cilia, where the deformation strategy is concentrated. The propagation of waves along these deformable appendices is the most common propulsion method, and depending on the number of flagella, their density on the cell's body, the shape of the cell body and the presence of external obstacles, different travelling waveforms are preferred [37].

Starting with mono-flagellated micro-organisms, i.e. swimmers having one flagellum, one can see that different propulsion mechanisms can be present and that these swimmers can switch between deformation strategies according to external conditions. The first examples of mono-flagellated micro-swimmers are spermatozoa, which show two types of tail beatings, planar and three-dimensional, depending on their proximity to a solid boundary. The first behaviour is shown when spermatozoa are swimming close to solid surfaces [45] while the second one appears when they swim in bulk fluid [62]. In the first case, the cell follows a circular path, and at the same time it is attracted towards the surface due to hydrodynamical effects (see figure 1, right). This behaviour is functional to reproduction, as the sperm cell is trapped in the proximity of the egg cell and progressively approaches it to fuse its nucleus with the egg cell's one [36]. This behaviour is initially triggered by chemotactic interaction with the egg cell [28] that accentuates the asymmetry of the tail beating. In fact, when swimming in bulk fluid, in presence of a chemoattractant, spermatozoa follow an helical path whose axis is aligned with the gradient of the chemical cue. The curvature of the swimming path results from their tail beating in a

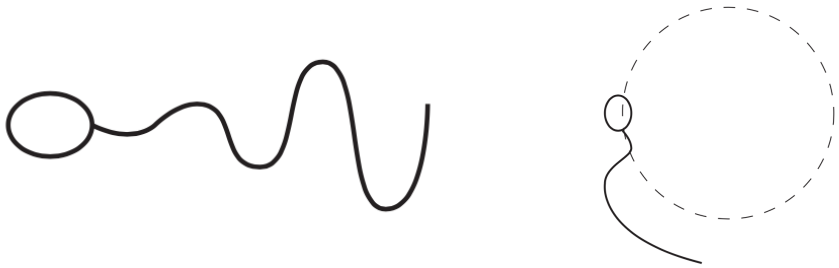


Figure 1: On the left, a stylized representation of a spermatozoon propagating a planar wave along its tail. On the right, a stylized representation of the circular path that sperm cells follow when swimming in proximity of a surface.

non-planar fashion [62]. Sperm cells of other species show planar sinusoidal beating [132] (see figure 1, left) and other mono-flagellated swimmers, with larger cell body and shorter tail like *Trypanosomes*, show yet different planar beating patterns [46].

In the case of multi-flagellated micro-organisms, i.e. swimmers with several flagella and possibly cilia, variability in the propulsion strategy is larger than before. In the *E. Coli* case, a multi-flagellated bacterium, flagella are attached to the main body via a flexible hook, and the propulsion is guaranteed by a combination of helical waves propagating along the flagella and flagellar bundling phenomena [65]. Bundling is also accentuated by the fact that, at low Reynolds number, the cell body counter-rotates with respect to the flagellar bundle. In this micro-organism, each flagellum can rotate independently from the others: when flagella rotate in the same direction they entangle and form a tail bundle, allowing the bacterium to swim in a precise direction; when one or more flagella counter-rotate in the bundled state, the tail tangle is undone and the bacterium reorients (see figure 2). This *run and tumble* behaviour is justified by the influence of chemotactic cues on the bacterium's swimming direction. While *E. Coli* forms a unique tail bundle, other bacteria can form multiple tail bundles due to their conformation. As an example, a recent study considered a magnetotactic marine bacterium presenting two separate flagellar bundles [136, 118]. In its case, a unique bundle was not formed because the flagella are localised in two points of its cellular body and sheathed in two separate groups. The study shows that the two separate flagellar bundles generate faster propulsion than one bundle and that, similarly to *E. Coli*, the propulsion of this bacterium is guaranteed by the counter-rotation of the cell body with respect to the flagella. In addition to chemical cues, magnetotactic bacteria are able to follow the lines of the geomagnetic field as para-magnetic beads are naturally included in their cell bodies.

The swimming direction of the biological micro-swimmers we presented can be influenced by chemical interactions or magnetism, while their propulsion is fundamentally due to the exchange of mechanical energy with the fluid by deforming their bodies. The same principles are exploited in the manufacturing of artificial micro-swimmers, as the next paragraph shows.

Artificial swimmers

A large class of man-made micro-swimmers is composed of biomimetic swimmers inspired by bacteria and self-propelling cells [13]. In this family, [34] reported the very first artificially constructed micro-swimmer by using DNA filaments as connecting material between a haemoglobin cell and small magnetic spheres that composed the beating flagellum. In the study, flagellar

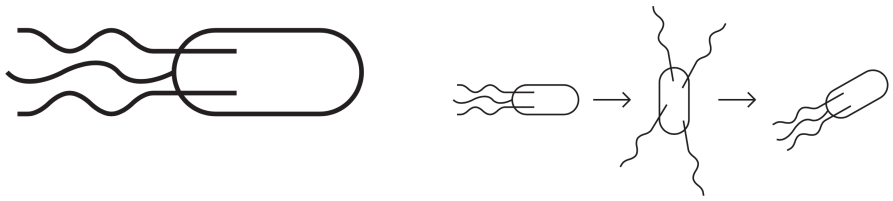


Figure 2: On the left, a stylized representation of an *E. Coli* bacterium. On the right, a stylized representation of the run-and-tumble swimming strategy of *E. Coli*.

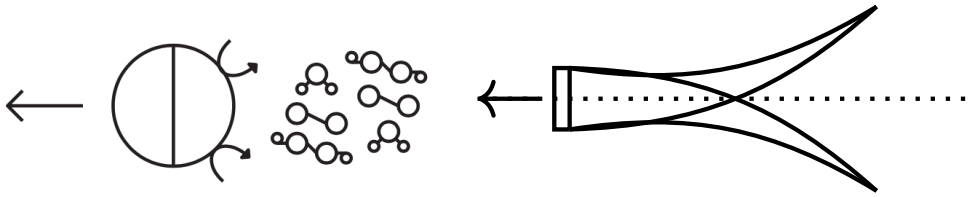


Figure 3: On the left, a stylized representation of a Janus particle propelling itself thanks to chemical reactions. On the right, a stylized representation of an artificially propelled micro-swimmer, inspired by [96].

motion was controlled by an external magnetic field and a non-reciprocal deformation was imposed allowing the propeller to swim. In this category of bio-hybrid entities, one can also couple a biological micro-swimmer to artificial matter in order to improve its propulsion, its control properties or to add functionalities to it. Spermatozoa trapped in hollow tubes [3] or *Spirulina* coated with magnetite [135] are two examples of bio-hybrid swimmers that combine the shape or propulsion characteristics of the biological organism with the controllability properties of the added artificial material: in fact, the coupled entity can then respond to external magnetic fields and be steered in the desired direction. A steering mechanism based on the alignment with a magnetic field is also used in the controlled propulsion of other artificial micro-swimmers [128].

Propulsion can also arise from non-mechanical effects. For example, some studies consider propulsion strategies which are of chemical origin (Janus particles, functionalized surfaces, see figure 3, left). In this case, the artificial swimmer is coated with specific chemical compounds that can react with the surrounding fluid and produce the necessary energy to propel itself [89]. Others focus on propulsion via acoustic micromanipulation techniques, as these methods are biocompatible and precise for sensitive micro-agents like cells. These techniques focus sound waves to trap and manipulate the particles, which can move passively (if the particle is dragged by the acoustic wave tweezers) or actively (if the particle self-propels as a result of the interaction with the tweezers at its resonant frequency). This latter category comprises artificial micro-swimmers with air cavities or with flagella that vibrate in resonance with the external sound waves [88]. Low Reynolds number swimming can be extended even to situations where swimmers are not microscopic: these studies are based on scaled-up laboratory experiences where the gravity force is compensated and the fluid density and viscosity are appropriately tuned to reproduce the

conditions of low Reynolds number flows [96] (see figure 3, right, for the representation of this scaled-up swimmer). This kind of setup allows focusing on the shape of the swimmer and on the control of its trajectory, neglecting the effects of Brownian motion or random fluctuations that appear when swimmers are microscopic.

Presently, the study of micro-swimmers continues to flourish as new possibilities arise: high-precision flagella visualization for a range of biological micro-swimmers [45], force measurements thanks to optical tweezers [92], miniaturization and control of artificial micro-swimmers [133] are just few among them. The development of targeted drug delivery micro-robots or non-invasive surgical ones is a joint effort for the scientific community due to the large number of challenges that must be faced [125], and modelling and simulation have their place in this as they provide valuable insights by focusing the attention on specific problems. In fact, numerical simulations can provide *in silico* experiments that may complement, explain or even substitute biological observations and be helpful for robot design and (numerical) prototyping.

Modelling and simulation of micro-swimming

The analysis of the previous swimmers relies on different theories that stemmed from the study of fluid mechanics at low Reynolds number. These theories exploit the linearity of the flow equations and their Green kernels to give an approximation of the propulsion speeds and forces exerted by a swimmer in low Reynolds number flows. Due to its prominence in micro-swimming strategies, the propagation of travelling waves on slender filaments has been investigated numerically and analytically by many authors. In [121], Taylor studied the propagation of sinusoidal travelling waves on a non-deformable infinite sheet and computed its propulsion velocity. He showed that the resulting speed, at first order, depends quadratically on the amplitude and the wavelength of the wave. In [1] the authors focused on helical travelling waves, and using the approximate hydrodynamic effects of slender filaments proposed by Gray and Hancock [49], they analysed the motion of such swimmers and investigated the optimal head-tail ratio guaranteeing faster propulsion. Also, [66] used an approximated model to study the propulsion generated by slender filaments, propagating planar or helical waves. Their model is based on Slender Body Theory, which computes the effect of flagellar motion thanks to a line distribution of fundamental solutions of Stokes equations along the centerline of the flagellum. Resistive Force Theory (RFT) [49, 82] and Slender Body Theory (SBT) [66] are quite accurate models for mono-flagellated swimmers with thin flagella, as their accuracy depends on the ratio between the thickness and the length of the swimming appendices. Numerical simulations of flagellated micro-organisms based on RFT, have been proposed in two and three dimensions: by discretizing the flagellum in N segments, [5] recovered the circular paths followed by sperm cells swimming close to a surface, while [35] used the RFT formalism to simulate a magnetic micro-robot with a thicker elastic flagellum. These approximate models focus on the hydrodynamical effects of each component of the swimmer, and neglect all the contributions that arise from close-range interactions among swimmer's parts. Even if approximate models can be used successfully for multi-flagellated swimmers, as in [59] to study how the hydrodynamical instability of rotating flagella generates propulsion, the mutual interaction of flagella need to be weak (or knowingly neglected). The role of close-range hydrodynamical interaction among the beating flagella is fundamental in the bundling of flexible flagella, in *E. Coli* for instance. Also the rotation of the cell's body influences bundling, but while this can be described with the help of resistive force theory [101], the effect of near-field interactions on bundle formation can not be captured by approximate hydrodynamical theories like RFT or SBT due to their hypotheses on weakly

interacting structures, and more accurate methods must be used [65, 112]. Accurate numerical methods take into account the two or three dimensional nature of the problem at hand and usually require a discretisation of the domains occupied by the fluid and/or the swimmer. A numerical method that is widely used in the micro-swimming community is the Boundary Element method, that relies on the integral formulation of the Stokes equations to determine the fluid velocity field [102]. This formulation allows the solution of swimming problems in three-dimensional domains by discretizing only the two-dimensional surfaces that bound the fluid. This allows significant savings in terms of storage of the geometry of the problem as the fluid domain (potentially unbounded) is not discretized. The boundary integral formulation that we evoked previously expresses the fluid velocity at the boundary of the swimmer as a function of fluid stresses σ_{ik} and velocities u_i, via the fundamental solutions G_{ij} and T_{ijk}

$$c_{ij}(x_0)u_i(x_0) = -\frac{1}{8\mu\pi}\int_{\partial\mathcal{F}} \sigma_{ik}(x)n_k(x)G_{ij}(x,x_0)\,\mathrm{d}S(x) + \frac{1}{8\pi}\int_{\partial\mathcal{F}}^{PV} u_i(x)T_{ijk}(x,x_0)n_k(x)\,\mathrm{d}S(x), \tag{1}$$

where $x_0 \in \partial\mathcal{F}$, n_k is the normal to $\partial\mathcal{F}$ pointing inside the fluid domain and $c_{ij}(x)$ is a function that depends on the geometry of $\partial\mathcal{F}$ [18]. The superscript PV means principal value, and it corresponds to the value of the integral when $x_0 \in \partial\mathcal{F}$. Fundamental solutions G_{ij} and T_{ijk} become singular as $x \to x_0$, hence a regularisation procedure needs to be employed for the numerical solution of (1). Two solutions are possible: a regularisation of the numerical approximation of the singular integrals using a semi-analytic procedure [55] or the usage of regularised kernels G^{ε}_{ij} and T^{ε}_{ijk}, that are computed using a regularised impulse function depending on a small parameter ε. Boundary element method has been used to investigate the hydrodynamics of micro-swimmers in unbounded media, in presence of a wall and other swimmers. In [100] the authors analysed the shape of micro-swimmers composed of an ellipsoidal head and a single helical flagellum and they conducted a parametric study to identify the values of the parameters ensuring maximal propulsion efficiency. In [127] this method was used to study the behaviour of the bacterium *Leishmania mexicana* in presence of rigid boundaries. In this study, the boundary element method was employed for two reasons: first, the shape of the bacterium (short flagellum and large cell body) was not adapted to an analysis based on RFT; second, the presence of the wall and its effects on the micro-swimmer needed to be accurately studied in order to explain how the bacterium approaches and is deflected from the solid boundary. Also [118] used BEM to study the swimming behaviour of a bacterium close to rigid boundaries, this time focusing on a bi-flagellated bacterium. In [126] the hydrodynamics of two swimming sperm cells was studied, in terms of stability, synchronism, and collaborative advantages. Boundary elements can be used for non-linear problems as well, like Navier-Stokes equations, but in this case the formulation is no longer defined on the boundary alone. For example, the internal domain needs to be meshed if a pseudo-body force approach is chosen to approximate the convective term [69]. In [29] the regularised kernel method (based on G^{ε}_{ij} and T^{ε}_{ijk}) was applied to the computation of Stokes flow, its numerical analysis was performed and its implementation was benchmarked against cases from the literature. Later, [95] used the regularised kernel method to study the hyper-activation of sperm flagella by coupling the fluid model to a biochemical one, taking into account the dynamics of the calcium ions that are responsible of the switching in swimming behaviour. In [119] the regularised kernel method is used to perform a fluid-structure coupling between the viscous fluid and an elastic filament: the flow is generated by the elastic forces of the filament, while the filament's shape and position are obtained by integrating the fluid velocity field. The regularised kernel method was recently used to propose a higher-fidelity model for bacteria interaction in [58], showing that the complex flow generated by the relative rotation of cell body and flagellum is determinant to describe the interactions and behaviours of bacterial active matter.

Method	Characteristics	Successful applications	Limitations
Resistive force theory/Slender body theory	Approximate hydrodynamical effects via expansion of the Green function	Motion of swimmers with slender flagella [1, 66, 5, 49]	Only short-range interactions [112]
Boundary element method/Regularised kernel method	Surface geometry discretization Boundary integral formulation	Swimmer in infinite domain [95, 119, 58] Presence of walls and short flagellum [127, 118] Swimmer-swimmer interaction [126]	Not adapted when the fluid model is nonlinear
Finite element method	Geometry discretization PDE discretization	General fluid and elasticity models [31, 33, 109, 115]	Infinite fluid domain Expensive 3D simulations

Table 1: Summary of methods used for modelling and simulation of micro-swimming.

Another accurate numerical method able to account for the full, non-approximate description of the swimming problem is the finite element method. This numerical method is very versatile and it has been used in a wide variety of contexts and physical problems. An interesting feature is its easy application, if compared with the boundary element method, to couplings between different PDEs, as when coupling fluid and elasticity equations, or when the Newtonian fluid tensor is substituted by a complex fluid one [115]. The literature on finite elements and micro-swimming is quite reduced. To our knowledge, publications in this field are mainly coming from the usage of commercial finite element software. Using the finite element method and the *Comsol* software, an analysis of the swimming behaviour of 2d sperm cells was performed [109], varying the parameters that describe the geometry and the gait of the swimmers. In [33], using the same software, the authors studied the behaviour of helical swimmers in a channel by varying the parameters describing the geometry of the helix. If one does not restrict to micro-swimming, one finds that a number of fluid-structure interaction and swimming problems at higher Reynolds numbers have been addressed using this method. For example, in [113] the authors simulated a swimming medusa by using snapshot of its gait, in order to study the mechanisms underlying its propulsion. In [31] the authors considered the coupling of Navier-Stokes and active elasticity equations to simulate the swimming behaviour of a carangiform fish propelled by its muscular force. Table 1 proposes a summary of the methods discussed in this section and what they are able to capture.

Optimisation and micro-swimming

In order to understand biological micro-swimmers and their motion strategies, one often needs to think in terms of propulsion efficiency or optimality with respect to a certain cost function. Moreover, control and shape optimisation of artificial micro-swimmers are crucial for applications. For these reasons, questions involving micro-swimming and optimisation in its largest sense have been addressed in recent literature.

In [100] a parametric study of the shape of a helical mono-flagellated swimmer is performed, where propulsion speed and efficiency are considered. The authors remark that the minima in the two cost functions usually do not coincide, which makes it interesting to investigate the two different optimal swimmers. In [56] a genetic algorithm is used to analyse the optimal beating pattern of a sperm flagellum in terms of its swimming efficiency. In this case, the

optimisation technique is inspired by the biological process of natural selection, by searching the "fittest" swimmer from a set of offspring solutions. This optimisation technique is of black-box type, meaning that it does not have a priori information about the problem at hand (like a gradient descent would have) other than evaluations of the cost function at different points. This family of methods, also called derivative-free since they do not use information about the derivative of the cost and constraint functions, are suitable for the optimisation of models that are computationally expensive [110, 75]. It is also possible to use theoretical shape optimisation to optimise the shape of swimmers: in [129] the authors study the optimal shape of rigid, magnetic, helical micro-swimmers both analytically and numerically, and compute the solution of the fluid problem via BEM. Experimental studies are very frequent in this domain: [107] analysed the shape of helical, scaled-up swimmers with multiple rigid flagella, while [128] investigated the shapes of helical microscopic swimmers to determine which tail length provides the largest propulsion speed.

Together with shape optimisation, the question of controlling the gait of swimmers and determining the optimal one is a relevant topic in recent literature. An optimal control problem for a flagellated scaled-up micro-robot has recently been addressed in [35] from the mathematical and experimental point of view. The objective of this study was to find a suitable periodic magnetic field that propelled the swimmer faster than a planar sinusoidal field. By using a limited number of Fourier modes, the authors provided a magnetic field that improved swimming speed by enforcing a three-dimensional beating pattern. Reinforcement learning has also been considered for path-planning and control of micro-swimmers. In these studies, propulsion and net advancement are rephrased in terms of a reward function, often the displacement in the desired direction, or the reach of a predetermined objective. Swimmers are then modelled as agents who can access a set of actions and want to optimise their expected reward. In [2] Q-learning is used to find the optimal strategy for a point-like swimmer immersed in a turbulent flow and moving between two points. In [122] reinforcement learning is used to find the stroke of a multi-sphere swimmer ensuring the fastest propulsion in different fluid media. The authors also show that, in Stokes flow, the optimal stroke of a three-sphere swimmer coincides with the one proposed in [91], the first paper to propose this swimmer model.

Meta-modelling and reduced-order modelling

The numerical solution of an optimisation problem relies on algorithms that evaluate the cost and constraint functions several times. When the cost function or the constraints depend on the solution of PDE-based problems, the optimisation can become extremely expensive. One possibility is to approximate the effects described by the PDE using a simpler model. We have extensively talked about approximate models of low Reynolds swimming, and we mention once more [35] as an example of using RFT coupled with optimisation. However, if this approximation is not possible and a full model must be used, it is still possible to reduce the computational cost of the optimisation by using meta-models and reduced-order models. Meta-modelling creates a simplified surrogate of the costly model by sampling the feasible region, which is then used for the optimisation process [67, 70]. In [47] the authors considered two algorithms for reduced order modelling of Stokes equations with parametrized domain, and computed the velocity and pressure field generated by a push-me-pull-you micro-swimmer [11], that is composed of two inflatable spheres of constant total volume moving on a line. The domain parametrization is based on the radius of one of the spheres and the distance between their centers. Using reduced-order modelling, it could be possible to assess the propulsion speed of the swimmer and look for the optimal swimming strategy by evaluating the effects of sphere inflation and distance on

the overall swimmer's motion. The usage of meta-modelling in the field of micro-swimming is quite unexplored, but techniques like kriging [114] and physics-based machine learning models for fluid mechanics [71, 21] could be exploited to address the shape or gait optimisation problems and optimal control of microscopic swimmers.

Main contributions of the thesis

The complexity and abundance of swimmers' gait and motion, together with the necessity to consider more complex and realistic swimming environments, require the development of a flexible framework, capable of accurate numerical simulation and open to optimisation. This thesis presents two instances of this framework, based on different numerical methods. Part I proposes a computational framework based on the boundary element method. Using this numerical approach, we modelled the complex hydrodynamical effects that originate from interacting objects, like flagella and cell bodies in a bacterium, without the need to discretise the fluid domain. This same approach was exploited for the parametric shape optimisation of micro-swimmers inspired by flagellated bacteria, where the absence of a discretised fluid domain avoided the meshing problems arising from rotating flagella. A further extension to include elasticity effects at the junctions has been outlined, in order to consider micro-swimmers which are closer to biological ones. Part II proposes a computational framework based on the finite element method. This second approach, despite being more costly than the previous one, was chosen for its extensibility to complex environments and fluid models. In the present work, however, we focused on the modelling and numerical aspects of the swimming problem in a classical Newtonian fluid. We addressed the computational problems arising from the choice of the Arbitrary-Lagrangian-Eulerian formulation to describe the swimming problem, like mesh adaptation and choice of the appropriate preconditioning strategies. Parts III and IV present some computational aspects of this second approach and a gallery of swimmers that were addressed within the framework: multi-sphere swimmers, flagellated sperm cells and elastic bodies. In this last part, a reinforcement learning algorithm was also interfaced with the simulation framework to study the the gait optimisation of multi-body swimmers in two dimensions.

Thesis content and outline

As we have shown, the accurate numerical simulation of swimming micro-organisms is important to capture their mutual interactions, their interaction with boundaries and get insight in their behaviour. For this reason, in this thesis we focus on the mathematical modelling and numerical simulation of swimming micro-organisms via finite and boundary element method. At the same time, our interest goes to shape improvement of these bodies and the more general problem of fluid-elastic interaction.

In part I, we model the fluid flow via the integral formulation of Stokes equations, and we simulate it via Boundary Element method. In chapter I.1 the swimming problem is detailed by adding to the Stokes equations the self-propulsion constraints, ensuring zero net force and torque on the swimmer. The discretization of the problem is presented, as well as various benchmarks for the method. In chapter I.2, this method is applied to a bi-flagellated swimmer whose head and tails are non deformable, that propels by rotating its helical tails. This swimmer was studied using BEM over the finite element method because the relative motion between head and tails would have required frequent remeshing of the fluid domain, if it were discretized. In

this chapter we focus on the swimmer's shape optimisation in two ways: first, we optimise its parameters successively, showing that funnel-like tails guarantee stronger propulsion and prolate cell bodies are preferred when few rigidly rotating flagella are present. After that, we consider Bayesian optimisation to construct a meta-model of the shape optimisation problem, allowing all parameters to vary at the same time. The two optimisation methods that are employed are of derivative-free type, since information on the gradient of the cost function is not available. Chapter I.3 contains the embryo of a study comparing Resistive Force Theory and Boundary Element method on a variation of the bio-inspired bi-flagellated bacterium of chapter I.2. In this chapter, the helical tails are allowed to bend at the head-tail junctions, where the flexible hooks are modelled as torque springs. In the first part of the chapter, the geometrical modelling and benchmarks of the RFT are reported. These are followed by the results of RFT on the swimmer: the interaction between the elastic junctions and the fluid forces lead the tails to settle at an equilibrium angle depending on the cell's body shape. Subsequently, the BEM modelling of the problem is addressed and a benchmark is proposed.

In part II, the Finite Element method is used to study deformable micro-swimmers. In this thesis, the Arbitrary-Lagrangian-Eulerian formalism is employed to model the fluid problem in moving domain. In chapter II.1 the mathematical modelling of the swimming problem is presented. A formulation for the motion of rigid-bodies in a fluid, based on the work of [85], is discussed, together with mathematical models of passive and active elasticity. In this case, the coupling conditions between the fluid and the elastic body are also discussed. Chapter II.2 is dedicated to the discretization of the swimmer problem and the numerical solution strategies that are employed. Chapter II.3 is dedicated to validation and verification of the method.

In part III we collect and detail the developments in the *Feel++* library that were contributed during this thesis. The implementation of mesh adaptation and rigid-body motion are described via the most important functions and data structures.

Part IV presents numerical experiments based on finite elements. A formulation to simulate the three-sphere swimmer using finite elements, or more generally bodies whose components move relatively to each other, is proposed and validated. A section on the gait optimisation of these swimmers is presented, where the optimisation was carried out via reinforcement learning. Models of sperm cells are also investigated in this chapter to validate our computational model when a predetermined boundary displacement is imposed.

The thesis closes with the collection of oral and written communications realised during this doctoral work.

The computational framework

In this thesis we use two numerical methods to perform our investigations, namely the finite and boundary element methods. Our developments rely on two libraries, i.e. the *Feel++* library for finite elements and the *Gypsilab* library for boundary elements.

The *Feel++* library. The Finite Element Embedded Library in C++ (*Feel++*) [103] is a computational framework providing a wide variety of advanced numerical methods for the solution of partial differential equations. Its core assets are Galerkin methods (continuous, discontinuous, spectral, hybrid-discontinuous) as well as reduced basis and domain decomposition methods. The library provides a number of tools ranging from polynomial interpolation to mesh

adaptation via a domain-specific embedded language mimicking the mathematical terminology of variational formulations. *Feel++* allows seamless parallel computations for solving large scale problems, for 1D, 2D, 3D and surface problems. Its interfaces with Boost, Ublas, Eigen3 and PETSc/SLEPc allow efficient data storage and solution of linear systems, in sequential and parallel, while its interface with *MMG* and *parMMG* libraries allows the usage of remeshing and mesh adaptation techniques in sequential and parallel as well. *Feel++* follows the C++17 standard (more recently updated to C++20) and uses the Boost C++ libraries Parameter, Fusion among others, to enhance the readability and conciseness of its code. *Feel++* provides also applications that solve physical problems (issued from fluid mechanics, solid mechanics, heat diffusion) or coupled problems (thermo-fluid, fluid-structure interaction, multifluid) built from libraries included in the framework [23, 87, 84].

The Gypsilab library. The Gypsilab library is an open source Matlab library for Boundary Element/Finite Element simulations [4]. It relies on a number of classes that handle different parts of the simulation process: mesh, finite element spaces, Green kernels, regularisation techniques and quadrature formulas. Its syntax is close to the mathematical variational formulation of the problems. It also contains a toolbox for storing the system matrices in H-matrix form, i.e. a compressed format that neglects matrix entries giving small contributions, hence allowing larger size problems to be treated. Regularisation techniques are based on semi-analytical integrations, that allows to correct the inaccuracy of numerical quadrature close to singularities via analytical integration [55].

Perspectives

Starting from the present work, numerous research opportunities open in the modelling and computational domains. First, the fluid model can be extended to more biologically relevant fluids, where the viscoelastic effects are taken into consideration, to obtain more realistic, complex and multi-layered domains where swimmers can be studied. Examples of these fluids are available in the boundary integral case [57] and in the differential case [115]. Comparison with other numerical methods could be considered, for example with immersed boundary methods.

The study of the shape optimisation problem could be cast in a more rigorous framework, using the tools of shape derivative and adjoint fluid problem. This study has been performed in the case of a still body in Stokes flow, in order to find the shape that minimizes the viscous drag effects [16]. A similar shape optimisation problem was numerically solved in the case of Navier-Stokes equations in [32]. However, at the best of our knowledge, the study of swimming shapes using this technique has been considered only once in [129] for the study of helical and magnetically propelled swimmers.

A last research direction could be the control of swimmers in complex fluid environments, where the presence of obstacles or walls invalidates the control strategies that are currently used in bulk fluid. Recently, machine learning has been introduced in the field to solve these control problems, and reinforcement learning algorithms [122] have been introduced to optimise the swimmer's gait or to solve path-planning problems [81]. This direction, for instance, could be explored in presence of obstacles in the fluid.

Part I

Boundary element framework for flagellated bacteria: modelling and optimisation

This part focuses on modelling, simulation and shape optimisation of flagellated bacteria swimming in a Stokes fluid. In chapter I.1, the Boundary Integral formulation of the Stokes equations is used to model the system, which is subsequently simulated via the Boundary Element Method. In chapter I.2, two optimisation techniques are explored to analyse the parametric shape optimisation of the swimmers: the derivative-free Nelder-Mead algorithm on one hand, and Bayesian optimisation on the other hand. In chapter I.3 the RFT and BEM methods are used to simulate a bi-flagellated bacterium with elastic head-tail junctions and in chapter I.4 some implementation aspects of the framework are presented.

Chapter I.1

Boundary Element Method

In section I.1.1 the mathematical and numerical modelling are presented: in subsection I.1.1.1 the integral form of Stokes equations is detailed, together with the self-propulsion constraints that encode the dynamical laws guaranteeing swimming at the micro-scale; in subsection I.1.1.2 the numerical details of BEM are given, and validation tests are presented in subsections I.1.1.3 and I.1.1.4; in subsections I.1.1.5 and I.1.1.6 a solver of rigid body motion based on quaternions and its validation are proposed.

I.1.1 Mathematical modelling

I.1.1.1 Integral formulation of Stokes equations

Stokes equations are a system of partial differential equations, linear in velocity and pressure. The integral form of the three-dimensional Stokes equations is based on the existence of a tensorial Green kernel G_{ij}, for $i, j \in \{1, 2, 3\}$, that collects the i-th component of the fluid velocity when the system is impulsively forced in the j-th direction. Let us denote by $\mathcal{F} \subseteq \mathbb{R}^3$ the unbounded fluid domain, complement in $\mathbb{R}^3$ of the region occupied by the swimmer, $u = u(t, x)$ the flow velocity and $p = p(t, x)$ the fluid pressure, $\bar{g} = \bar{g}(x, t)$ the known Dirichlet data to be imposed on $\partial\mathcal{F}_D$, μ the viscosity of the flow. The system reads as

$$\begin{cases} \nabla p - \mu \Delta u = 0 & \text{on } \mathcal{F}, \\ \nabla \cdot u = 0 & \text{on } \mathcal{F}, \\ u = \bar{g} & \text{on } \partial\mathcal{F}_D, \\ \|u\|, p \to 0 & \text{as } \|x\| \to \infty. \end{cases} \tag{I.1.1}$$

Using Fourier transform it is possible to find the fundamental solution $G_{ij}(x, x_0)$, for $i, j \in \{1, 2, 3\}$, of singularly forced Stokes equations, which reads

$$G_{ij}(x, x_0) = \frac{1}{8\pi\mu}\Big(\frac{\delta_{ij}}{\|x - x_0\|} + \frac{(x_i - x_{0i})(x_j - x_{0j})}{\|x - x_0\|^3}\Big), \tag{I.1.2}$$

and provides the fluid velocity at point x as a result of an impulsive force located in x_0. The fundamental solution for the pressure reads

$$p_j(x, x_0) = \frac{1}{4\pi}\frac{(x_j - x_{0j})}{\|x - x_0\|^3},$$

and provides the pressure value at x as a result of an impulsive force located in x_0 [102]. Generally speaking, the mathematical expression of G depends on the shape and geometry of the domain, and whenever these are simple enough, an analytical expression of the Green kernel is available (see [102]). The pressure fundamental solution is seldom used: in its place the stress tensor

$$T_{ijk} = -\delta_{ik} p_j(x, x_0) + \frac{\partial G_{ij}}{\partial x_k}(x, x_0) + \frac{\partial G_{kj}}{\partial x_i}(x, x_0),$$

is preferred as it allows the imposition of Neumann boundary conditions.. Its expression, in free space, is

$$T_{ijk}(x, x_0) = -6 \frac{(x_i - x_{0i})(x_j - x_{0j})(x_k - x_{0k})}{\|x - x_0\|^5}. \tag{I.1.3}$$

In the general case, the fluid domain $\mathcal{F}$ can have one or more disjoint boundaries, representing the fluid-solid interfaces where either Dirichlet either Neumann boundary conditions can be imposed. Using the Lorentz reciprocal identity [102], the velocity field in $\mathcal{F}$ can be described by the boundary integral equation

$$u_j(x_0) = -\frac{1}{8\mu\pi} \int_{\partial\mathcal{F}} \sigma_{ik}(x) n_k(x) G_{ij}(x, x_0) \, \mathrm{d}S(x) + \frac{1}{8\pi} \int_{\partial\mathcal{F}} u_i(x) T_{ijk}(x, x_0) n_k(x) \, \mathrm{d}S(x) \tag{I.1.4}$$

when $x_0 \in \mathring{\mathcal{F}}$, or by

$$c_{ij}(x_0) u_i(x_0) = -\frac{1}{8\mu\pi} \int_{\partial\mathcal{F}} \sigma_{ik}(x) n_k(x) G_{ij}(x, x_0) \, \mathrm{d}S(x) + \frac{1}{8\pi} \int_{\partial\mathcal{F}}^{PV} u_i(x) T_{ijk}(x, x_0) n_k(x) \, \mathrm{d}S(x) \tag{I.1.5}$$

when $x_0 \in \partial\mathcal{F}$, where $\int^{PV}$ indicates the principal value of the integral and $c_{ij}(x)$ a coefficient that depends on the problem. The first integral on the right-hand side of (I.1.5) is sometimes denoted as "single layer", while the second one by "double layer".

If the fluid flows around a still solid, the fluid velocity on that interface will be zero: this will imply that also the contribution of the double layer, when considering the surface of such solid, will be zero. If the immersed object is experiencing a rigid body motion, its velocity can be written as $u(x) = U + \omega \times (x - x^{CM})$, where x^{CM} is the centroid of the body, and substituting it into the second integral of (I.1.5) leads to

$$\frac{1}{4\pi} \int_{\partial\mathcal{F}}^{PV} u_i(x) T_{ijk}(x, x_0) n_k(x) \, \mathrm{d}S(x) = 0. \tag{I.1.6}$$

Hence, when considering rigid body motion, the double-layer integral gives zero contribution and the boundary integral representation of the flow is given by the single-layer potential alone, according to

$$u_j(x_0) = -\frac{1}{8\mu\pi} \int_{\partial\mathcal{F}} \sigma_{ik}(x) n_k(x) G_{ij}(x, x_0) \, \mathrm{d}S(x), \tag{I.1.7}$$

for $x_0 \in \partial\mathcal{F}$.

Equation (I.1.7) provides the velocity on the boundary of the solid once the fluid stresses are known. It can also be inverted and used to obtain the stresses from the fluid velocity on the boundary of the solid [18]. If the solid self-propels or there are external forces F_{ext} or torques T_{ext} that power its motion, (I.1.7) must be complemented by a set of two vector equations closing the system. These equations are

$$\begin{aligned} \int_{\partial\mathcal{F}} \sigma_{ij}(x) n_j(x) \, \mathrm{d}S(x) &= F_{i,ext}, \\ \int_{\partial\mathcal{F}} \sigma_{ij}(x) n_j(x) \times (x - x^{CM}) \, \mathrm{d}S(x) &= T_{ext}, \end{aligned} \tag{I.1.8}$$

stating that the net fluid forces and torques over the swimmer are balanced by the external ones. If the external forces and torques are zero, these balance equations are called *self-propulsion constraints*. The solution of system (I.1.7)-(I.1.8) provides the translational and rotational speeds of the swimmer (U, ω), together with the normal fluid stress distribution σn on its boundary.

I.1.1.2 BEM for Stokes equations and the swimming problem

The Boundary Element method (BEM) is a mesh based numerical method. In our case, the computational domain is a simplicial mesh which discretizes the surface of the swimming object. A conforming finite element space is defined over the discrete surface and an appropriate quadrature formula is chosen, depending on the polynomial degree of approximation. We now detail the numerical formulation of problem (I.1.7)-(I.1.8).

In the following, let us denote by f the surface tensions σn. The components of surface tensions are expanded as $f_j(y) = \sum_{l=1}^{N} f_j^l \phi^l(y)$, where $\{\phi^l\}_{l=1}^N$ span the scalar piecewise-linear continuous finite element space over $\partial\mathcal{F}$, and each component of equation (I.1.7) is projected onto this space, giving

$$\begin{aligned} &\int_{\partial\mathcal{F}} U_i \phi^k(x)\,\mathrm{d}x - \int_{\partial\mathcal{F}} ((x - x_S)\times\omega)_i \phi^k(x)\,\mathrm{d}x + \\ &\int_{\partial\mathcal{F}} \left(\int_{\partial\mathcal{F}} G_{ij}(x,y) \sum_{l=1}^{N} f_j^l \phi^l(y)\,\mathrm{d}y \right) \phi^k(x)\,\mathrm{d}x = 0 \quad \text{for } k = 1,\dots,N, \end{aligned} \tag{I.1.9a}$$

$$\int_{\partial\mathcal{F}} \sum_{l=1}^{N} f_j^l \phi^l(y)\,\mathrm{d}y = F_{j,ext} \quad \text{for } l = 1,\dots,N, \tag{I.1.9b}$$

$$\int_{\partial\mathcal{F}} (y - x^{CM}) \times \sum_{l=1}^{N} [f_1^l, f_2^l, f_3^l] \phi^l(y)\,\mathrm{d}y = -T_{ext} \quad \text{for } l = 1,\dots,N. \tag{I.1.9c}$$

The integrals are evaluated numerically via Gaussian quadrature, and the singular Green kernels are regularized when the singularity and the evaluation points approach one another. A possible regularization method consists in using semi-analytic integration for singular kernels to correct the result of Gaussian quadrature. This technique expresses the Green kernels in polar coordinates (r, θ), as it is possible to separate the singular part in r from the non-singular part in θ. Then, instead of performing Gaussian quadrature on the Cartesian form of the Green kernels, the singular part is analytically integrated, while the non-singular one is numerically integrated [55].

If the boundary of the fluid domain $\partial\mathcal{F}$ can be written as a sum of disjoint sets ∂F_i, for $i = 1,\dots,n_T$, the previous integrals can be split accordingly, and the resulting system matrix will have a block structure. According to the number of disjoint sets n_T, the block G that corresponds to the kernel discretization will have $(n_T)^2$ sub-blocks: each of these will be obtained by integrating $\int_A \int_B G_{ij}(x,y)\phi^l(y)\phi^k(x)$, where $A, B \in \{\partial F_1, \dots, \partial F_{n_T}\}$. Thanks to the symmetry of the Green kernel (I.1.2) with respect to its arguments, only the upper-diagonal blocks need to be computed. A n_T sub-block structure is present in both of the blocks J and K implementing the self-propulsion constraints (I.1.8). In summary, the matrix form of equations (I.1.9) reads as

$$\begin{matrix} 3\times N\times n_T \\ 3 \\ 3 \end{matrix} \left\{ \begin{matrix} \\ \\ \\ \end{matrix} \right. \begin{bmatrix} \mathrm{G} & J^T & K^T \\ J & 0 & 0 \\ \underbrace{K}_{3\times N\times n_T} & \underbrace{0}_{3} & \underbrace{0}_{3} \end{bmatrix} \begin{bmatrix} f \\ U \\ \omega \end{bmatrix} = \begin{bmatrix} 0 \\ F_{ext} \\ -T_{ext} \end{bmatrix}.$$

The definition of G, J, K is postponed to section I.2.6.

The implementation is done through the Matlab BEM library *Gypsilab*[1].

I.1.1.3 Convergence study and verification

The testcase we selected for the validation of the method was presented in [52, p.119, formulas 4-17.17 and 4-17.18]: it consists in a vertically translating rigid sphere with prescribed velocity on the boundary $u = [0; 0; \bar{U}_z]$.

The exact solution in [52] is given in spherical coordinates to exploit the symmetry of the configuration, but one can resort to its Cartesian representation by using the tangent map of the change of coordinates [52, p.507, formula A-15.29].

Let N denote the number of mesh nodes. Denote by G the matrix coming from the discretization of the single layer, and by M the mass matrix whose entries are

$$\mathrm{G}_{lk}^{\{ij\}} = \int_{\partial\mathcal{F}} \int_{\partial\mathcal{F}} G_{ij}(x,y)\phi^l(y)\phi^k(x), \qquad M_{lk}^{\{ii\}} = \int_{\partial\mathcal{F}} \phi^l(x)\phi^k(x), \qquad i,j \in \{1,2,3\},$$

for the $\{ij\}$ and $\{ii\}$ sub-blocks, respectively. Then, boundary stresses f are found via matrix inversion

$$f = -\mathrm{G}^{-1} M U,$$

where U is the vector of size $3N$ whose components are

$$U = \begin{bmatrix} 0_{N\times 1} \\ 0_{N\times 1} \\ (\bar{U}_z)_{N\times 1} \end{bmatrix}.$$

The velocity field U_{val} is then evaluated on a set of preassigned points by multiplying the stresses f with a radiation matrix G_{eval}, computed in a similar way to G,

$$U_{eval} = -M^{-1}\mathrm{G}_{eval} f.$$

U_{val} is then compared with the exact solution coming from the literature. In our case we evaluated U_{val} and the exact solution on a sphere whose radius was twice the radius of the translating sphere.

For the testcase, we consider a translating sphere of unit radius, fluid viscosity $\mu = 1$, a translational velocity of magnitude $\bar{U}_z = 2$ and a number of mesh points growing as 2^p, $6 \leq p \leq 11$, with a maximum number of 2048 mesh points. In figure I.1.1 we present the convergence plot for the L^2 norm without the usage of H-matrices storage, while in figure I.1.2 we present the convergence plot for the L^2 norm with the usage of H-matrices. In this second case, we considered again a translating sphere of unit radius, fluid viscosity $\mu = 1$, a translational velocity of value $\bar{U}_z = 2$. However, thanks to the storage technique, a larger number of mesh points could be used. They again grew as 2^p, with $6 \leq p \leq 13$, and a maximum number of 8192 mesh points was used. The convergence profiles in figures I.1.1 I.1.2 are of order 2, as expected from $\mathbb{P}^1$ finite element theory. In [4] it is also shown that hierarchical compression ensures a faster assembly step when the size of the problem grows, as well as an improvement in the solution time if compared to the computation of the full matrix associated to the Green kernel of the problem (acoustic scattering, in their case).

[1] https://github.com/matthieuaussal/gypsilab

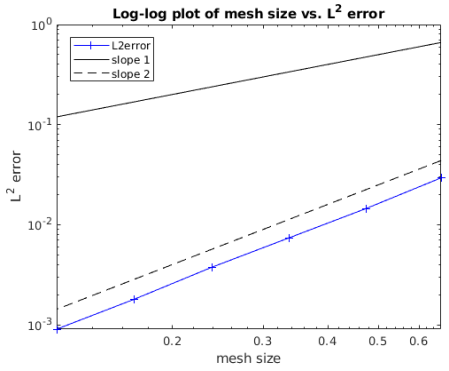


Figure I.1.1: L^2 error convergence without H-matrices.

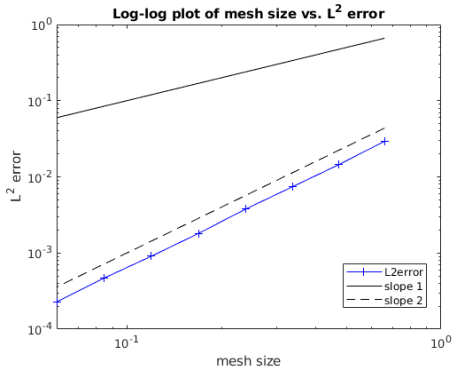


Figure I.1.2: L^2 error convergence with H-matrices.

In order to verify the correct implementation of (I.1.7)-(I.1.8), we tested our code on a sphere subject to a translating force and a rotating torque. Thanks to Stokes formulas for viscous fluid force $F_{ext} = -6\pi R\mu U$ and torque $T_{ext} = -8\pi R^3\mu\omega$ for a sphere of radius R, we are able to recover the imposed velocities U and ω with relative errors, as collected in table I.1.1.

Velocity	Imposed value	Computed value	Relative error (%)
U_x	9.0	9.0173	< 0.2
U_y	5.0	5.0096	< 0.2
U_z	1.0	1.0019	< 0.2
ω_x	34.0	34.1964	< 0.6
ω_y	6.0	6.0347	< 0.6
ω_z	3.0	3.0172	< 0.6

Table I.1.1: Velocity values and relative errors in the forced translating and rotating sphere testcases.

I.1.1.4 Validation of the swimming system

After testing the example of the translating sphere, we reproduced some cases issued from micro-swimming literature where the BEM method was used to solve the self-propulsion problem. The

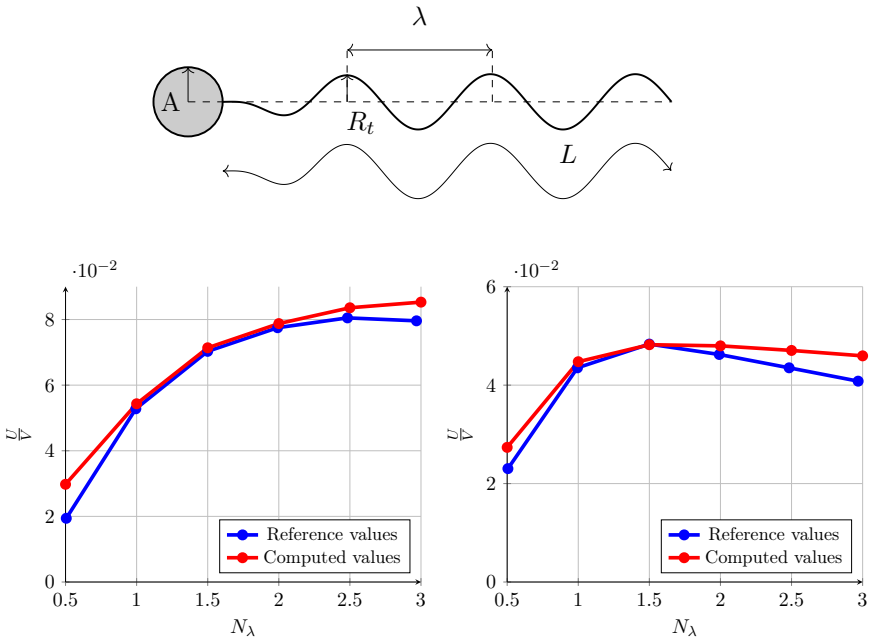


Figure I.1.3: On the top figure we report the geometry of the swimmer considered in [100]: the radius of the head A, the radius of the helix and its wavelength λ, the total length L of the tail are shown. On the bottom figures we show the comparison of our simulation results (red) with the values sampled from [100] (blue). On the left, the results for $L/A = 10$ are reported while on the right the results for $L/A = 5$ are reported.

first case we considered is issued from [100]: our aim was to reproduce figure 2 of the above mentioned paper, where the non-dimensional mean swimming speed U/V, with $V = \omega_F/k_E$ the ratio of the tail's rotation speed and shrinkage coefficient k_E (see I.2.2 for the role of k_E in the shape of the helix), was plotted as a function of the number of wavelength of the helical tail N_λ. Figure I.1.3(top) shows the helical swimmer under consideration and some of its geometrical parameters. The number of wavelengths N_λ is related to the total tail length L by $L = \int_0^{\lambda N_\lambda} |ds/dx|\, dx$, where $|ds/dx|$ is the norm of the tangent to the centerline of the helix. The ratio between the radius A of the spherical head of the swimmer and the thickness a of the flagellum is 0.02 in this case, and the product $R^t 2\pi/\lambda = 1$, where R^t is the helix radius and λ is the wavelength of the helix. The figure in the paper was sampled using a visualisation tool, and the obtained values are compared to the results of our simulation in figure I.1.3. In figure I.1.3 two tail lengths L are considered for comparison: $L = 10A$ and $L = 5A$. The second comparison we considered was with [118], where a bi-flagellated bacterium was considered. In this case, the non-dimensionalisation of U and ω was performed with the angular ω_0 and linear speeds U_0 of a mono-flagellated bacterium having the same tail and head shapes. Figure I.1.4 shows the comparison between our results and figure 2 of the paper. The values of the geometric parameters that were employed and the shape of the swimmers are detailed in table I.2.1 and in section I.2.

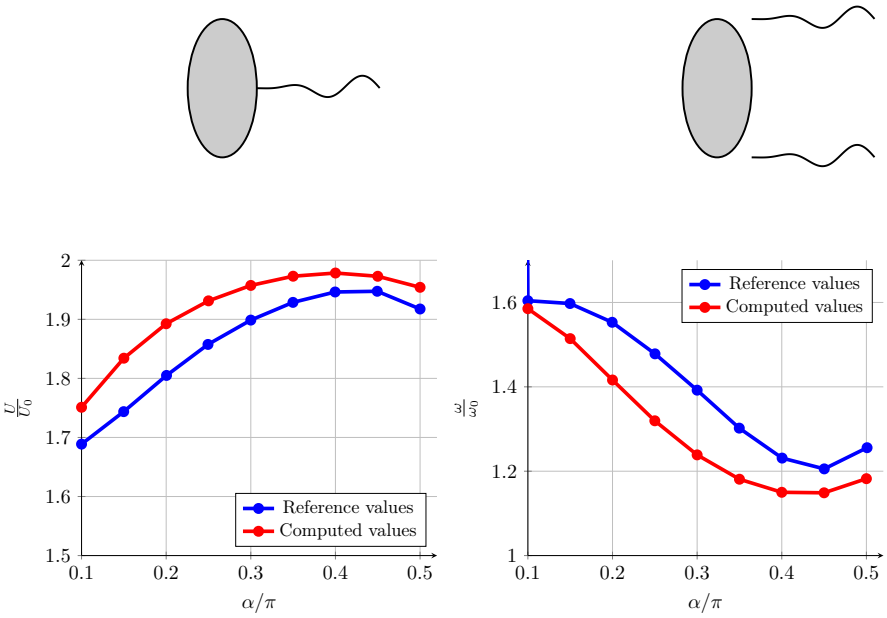


Figure I.1.4: On the top figures we report a stylized representation of the swimmers considered for the comparison with [118]. The velocities U_0 and ω_0 are those the mono-flagellated bacterium, while U and ω are those of the bi-flagellated bacterium. On the bottom figures, we compare our simulation results (red) with the values sampled from [118] (blue).

I.1.1.5 Rigid-body solver

In order to update the position and orientation of the swimmer, we implemented a rigid body solver in Matlab using the quaternion representation for the angular description of the body. Unit quaternions are a good option to represent the orientation of a rigid object, since they avoid parametrization issues that can affect other representations, like the gimbal lock for Euler angles [40, 74].

A quaternion q is defined by a scalar q_0 and a 3-dimensional vector $q_v = (q_{v_1}, q_{v_2}, q_{v_3})$, $q = (q_0, q_v)$. The conjugate of q will be denoted by $\bar{q} = (q_0, -q_v)$ and the quaternion product is defined as $q * r = (q_0 r_0 - q_v \cdot r_v \, , \, q_0 r_v + q_v r_0 + q_v \times r_v)$. The norm of a quaternion is defined as $||q|| = \sqrt{q\bar{q}} = \sqrt{q_0^2 + q_{v_1}^2 + q_{v_2}^2 + q_{v_3}^2}$. Each quaternion of unit norm represents a reflection in 3D space, and together with its conjugate concurs to describe a rotation in space. The rotation matrix linked to a unit quaternion $q = (q_0, q_v)$ is given by

$$R(q) = \begin{bmatrix} q_0^2 + q_{v_1}^2 - q_{v_2}^2 - q_{v_3}^2 & 2(q_{v_1}q_{v_2} - q_0 q_{v_3}) & q_{v_1}q_{v_3} + q_0 q_{v_2} \\ 2(q_{v_1}q_{v_2} + q_0 q_{v_3}) & q_0^2 - q_{v_1}^2 + q_{v_2}^2 - q_{v_3}^2 & 2(q_{v_2}q_{v_3} - q_0 q_{v_1}) \\ 2(q_{v_1}q_{v_3} - q_0 q_{v_2}) & 2(q_{v_2}q_{v_3} + q_0 q_{v_1}) & q_0^2 - q_{v_1}^2 - q_{v_2}^2 + q_{v_3}^2 \end{bmatrix}.$$

In what follows we present how to get the swimmer frame dynamics in the laboratory frame. In particular, we derive the equations describing the evolution of the quaternion associated to the orientation of the swimmer. Let $X \in \mathbb{R}^3$ denote the coordinates of a point in the body frame and $[0, X]$ the corresponding quaternion where $q_0 = 0$ and $q_v = X$. In the laboratory frame, the coordinates $x \in \mathbb{R}^3$ of the same point are obtained by multiplication of $[0, X]$ by q, the unit quaternion describing the orientation of the swimmer, and its conjugate: $[0, x] = q * [0, X] * \bar{q} = [0, R(q)X]$. Performing time differentiation of the previous relation, we get

$$\frac{\mathrm{d}[0,x]}{\mathrm{d}t} = \frac{\mathrm{d}q}{\mathrm{d}t} * [0,X] * \bar{q} + q * [0,X] * \frac{\mathrm{d}\bar{q}}{\mathrm{d}t} = \frac{\mathrm{d}q}{\mathrm{d}t} * \bar{q} * [0,x] + [0,x] * q * \frac{\mathrm{d}\bar{q}}{\mathrm{d}t}.$$

It is possible to prove that $-q * \frac{\mathrm{d}\bar{q}}{\mathrm{d}t} = \frac{\mathrm{d}q}{\mathrm{d}t} * \bar{q}$ are of the form $[0, w]$. Hence, by performing the product of $[0, 2w] * [0, x]$, one gets

$$\frac{\mathrm{d}[0,x]}{\mathrm{d}t} = [0, 2w \times x].$$

The relationship $\frac{\mathrm{d}x}{\mathrm{d}t} = 2w \times x$ relates the linear velocity of point x with its vector product with $2w$, i.e. the angular velocity of the body seen from the laboratory frame. This means that, if we denote by ω the angular velocity of the body in the body frame, its coordinates in the laboratory frame are $R(q)\omega$. Hence, $2\frac{\mathrm{d}q}{\mathrm{d}t} * \bar{q} = [0, R(q)\omega]$. Then, multiplying both sides of the equality by q gives the following dynamics for the quaternion

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{1}{2}[0, R(q)\omega] * q. \tag{I.1.10}$$

Once the velocities of the swimmer U and ω are recovered by solving (I.1.7)-(I.1.8) in its reference frame, we can determine the position and orientation of the swimmer in the laboratory reference frame by solving a set of ODEs of the form [40]

$$\begin{cases} \dfrac{\mathrm{d}q}{\mathrm{d}t} = \dfrac{1}{2}[0, R(q)\omega] * q, \\ \dfrac{\mathrm{d}X}{\mathrm{d}t} = R(q)U, \end{cases} \tag{I.1.11}$$

where q is the unit quaternion representing the orientation of the body in the laboratory fixed frame, $R(q)$ is the rotation matrix associated to the unit quaternion q, $*$ denotes the quaternion product and X is the position of the swimmer's centroid in the fixed reference frame. Appropriate initial conditions for the ODEs have to be chosen. The discretization of the time derivatives is based on the fourth-order Nystrom scheme [17].

I.1.1.6 Validation of rigid body solver

In order to validate the rigid body solver, we considered the results presented in [83]. The solution to

$$\frac{\mathrm{d}X}{\mathrm{d}t} = R(q)U, \tag{I.1.12a}$$
$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{1}{2}[0, R(q)\omega] * q, \tag{I.1.12b}$$
$$\frac{\mathrm{d}U}{\mathrm{d}t} = F/m, \tag{I.1.12c}$$
$$I\frac{\mathrm{d}\omega}{\mathrm{d}t} + \omega \times I\omega = T, \tag{I.1.12d}$$

complemented with adequate initial conditions, is searched.

Let us remark that the exactness of the translational part of the rigid body motion is easy to check. As a result, we will consider only equations (I.1.12b) and (I.1.12d).

We validate the rotational part of the rigid body motion by extracting a set of sample points from the graphics of [83] and compare such results with ours. The case we will consider is characterised by a time-varying torque, defined in the body frame, given by

$$T = \begin{bmatrix} 1.0 + 2.7 \times 10^{-2}t - 2.4 \times 10^{-4}t^2 + 5.7 \times 10^{-7}t^3 \\ -1.5 - 9.0 \times 10^{-3}t + 1.2 \times 10^{-4}t^2 - 3.0 \times 10^{-7}t^3 \\ 13.5 \end{bmatrix} N \cdot m.$$

The body reference frame at $t = 0$ coincides with the laboratory reference frame, which in terms of quaternions is expressed as $q(0) = (1, 0, 0, 0)$. The components of the angular velocity ω at $t = 0$ along the three principal axes are $(0, 0, 0.329)\, rad/s$, while the inertia matrix is given by

$$I = \begin{bmatrix} 2985 & 0 & 0 \\ 0 & 2729 & 0 \\ 0 & 0 & 4183 \end{bmatrix} kg \cdot m^2$$

Since in [83] the results about the frame orientation are given in terms of Euler angles (ϕ_x, ϕ_y, ϕ_z), we use the following relationship to obtain them from a quaternion $q = (q_0, q_1, q_2, q_3)$:

$$\begin{bmatrix} \phi_x \\ \phi_y \\ \phi_z \end{bmatrix} = \begin{bmatrix} \arctan \frac{2(q_0q_1 + q_2q_3)}{1-2(q_1^2+q_2^2)} \\ \arcsin(2(q_0q_2 - q_3q_1)) \\ \arctan \frac{2(q_0q_3 + q_1q_2)}{1-2(q_2^2+q_3^2)} \end{bmatrix}$$

Figure I.1.5 shows that our results are in agreement with the numerical results in [83].

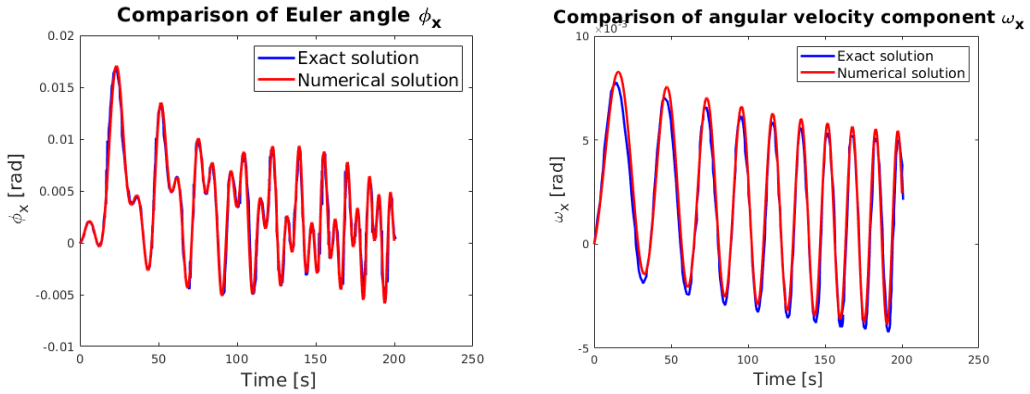


Figure I.1.5: Comparison of the sampled solution from [83] and our numerical solution from equations (I.1.12). On the left, the results on Euler angle ϕ_x are compared. On the right, the results on the component ω_x of the angular velocity are compared.

Chapter I.2

Shape improvement of a multi-flagellated bacterium

In this chapter we use the Boundary Element method to study the shape of multi-flagellated micro-organisms inspired by magnetotactic bacteria MO-1 [78, 136, 118]. We exploit the Nelder-Mead algorithm and Bayesian optimisation to optimise the shapes of these micro-swimmers and find the fastest propelling ones. In section I.2.1 the swimmer model is presented and in section I.2.2 the discrete swimming system and optimisation algorithms are detailed. Section I.2.3 collects the results of subsequent steps of shape optimisation, each concerning only part of the swimmer's parameters, via the Nelder-Mead algorithm, while section I.2.4 contains the results of simultaneous optimisation of all the swimmer's parameters, via Bayesian optimisation. Sections I.2.5 reports the comparison of the two approaches and section I.2.6 contains a detailed formulation for the BEM matrix entries.

I.2.1 Mathematical modelling

The model swimmer S is composed of non-deformable parts: an ellipsoidal head, and n_T helical tails, where $n_T \in \{1, 2, 4\}$ varies according to the micro-swimmer in consideration (see figure I.2.1 for a graphical representation of MO-1). The ellipsoidal head H is described by the equation

$$H = \{(x, y, z) : \frac{x^2}{(R_1^h)^2} + \frac{y^2}{(R_2^h)^2} + \frac{z^2}{(R_3^h)^2} = 1\}, \tag{I.2.1}$$

where R_1^h is the semi-axis in the propulsion direction and R_2^h, R_3^h are the two orthogonal semi-axes. The tails, denoted by F_i, $i = 1, ...n_T$, are tubes of radius r, having as centerline the curve of total length L described by

$$\begin{aligned} x(s) &= s, \\ y(s) &= R^t(1 - e^{-k_E^2 s^2}) \cos(2\pi s/\lambda), \\ z(s) &= R^t(1 - e^{-k_E^2 s^2}) \sin(2\pi s/\lambda), \end{aligned} \tag{I.2.2}$$

where R^t is its maximal radius, λ is its wavelength and k_E is a shrinkage coefficient [54]. Tails are separated from the cell body by a small gap l, measured along the normal to the ellipsoid, and are symmetrically distributed and rotated with respect to the propulsion direction. The gap

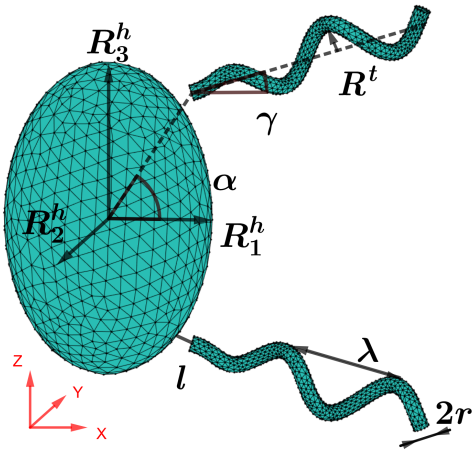


Figure I.2.1: Three-dimensional mesh model of the MO-1 bacterium ($n_T = 2$). In this picture the gaps l between the cell body and the tails are visible. The head is an ellipsoid with major axis R_3^h and minor axes $R_1^h = R_2^h$. The tail arc-length is L, its sectional radius is r. The helix wavelength is λ and its maximal radius is R^t. The analytical expression for the cell body and the tail's centerline are given in (I.2.1) and (I.2.2), respectively.

between the tails and the head corresponds to the space occupied by a flexible hook, which allows the flagella to change their orientation. In this chapter we neglect flexibility effects and suppose that the flagella are not able to change their orientation as they interact with the surrounding fluid. The latitude of the tail junctions is denoted by α while their inclination angle with respect to the horizontal is indicated by γ. The previous notations are presented in Figure I.2.1, in the case of a bi-flagellated swimmer ($n_T = 2$). In order to swim, the helices rotate around their axes at speed $\omega_F = -2\pi$, mimicking bacteria propagating helical waves along their tail. This modelling was already employed in [118] for the bi-flagellated swimmer, and in [100] for the mono-flagellated swimmer.

The fluid is modelled via Stokes equations, due to the small value of the Reynolds number for micro-swimmers. Fluid velocity and pressure, denoted by (u, p), satisfy the following Dirichlet boundary value problem when the swimmer is composed of one head H and several flagella F_i, $i \in \{1, 2, 4\}$

$$\begin{cases} \nabla p - \mu \Delta u = 0 & \text{on } \mathbb{R}^3 \setminus S \\ \nabla \cdot u = 0 & \text{on } \mathbb{R}^3 \setminus S \\ u = U + \omega \times (x - x^{CM}) & \text{on } \partial H \\ u = U + \omega \times (x - x^{CM}) + \omega_F \vec{e}_1^{F_i} \times (x - x^{F_i}) & \text{on } \partial F_i \end{cases}$$

where $S = H \cup F_1 \cup \cdots \cup F_{n_T}$, x^{CM} is the center of mass of the cell, $\vec{e}_1^{F_i}$ is the axis direction of the $i-$th tail, for $i \in \{1, \ldots, n_T\}$, x^{F_i} is the $i-$th tail's junction $i \in \{1, \ldots, n_T\}$, $U \in \mathbb{R}^3$ and $\omega \in \mathbb{R}^3$ are the linear and angular velocity of the swimmer. We remark that the Dirichlet boundary conditions are composed of two distinct parts: the term $\omega_F \vec{e}_1^{F_i} \times (x - x^{F_i})$ depends on the rotation rate of the helical tail, that is a known datum, while term $U + \omega \times (x - x^{CM})$ contains the linear and angular velocities that result from the interaction between the swimmer and the fluid, which are unknown. Using the integral representation formula (1), the linear and

angular velocities (U, ω) and the surface tensions f can be determined via

$$U - (x - x^{CM}) \times \omega + \int_{\partial S} G(x,y) f(y)\,\mathrm{d}y = (x - x^F) \times \omega_F \vec{e}_1^F, \tag{I.2.3a}$$

$$\int_{\partial S} f(y)\,\mathrm{d}y = 0, \tag{I.2.3b}$$

$$\int_{\partial S} (y - x^{CM}) \times f(y)\,\mathrm{d}y = 0. \tag{I.2.3c}$$

Equation (I.2.3a) derives from (I.1.7), where we exploited the fact that on the boundary of a rigid body the velocity writes as $u(x) = U + \omega \times (x - x^{CM})$ and that reduces the second integral in (1) to 0 when $x \in \partial S$ [102]. Equations (I.2.3b)-(I.2.3c) respectively indicate that net forces and torques over the swimmer are zero, i.e. the body moves thanks to internal forces and body deformations. Using (U, ω) obtained from (I.2.3), the resulting trajectory of the swimmer can be computed via the rigid-body motion solver based on quaternions that we presented in subsection I.1.1.5.

In the case of our interest, the shape of the swimmer depends on a finite number of parameters $p \in \mathcal{P}$, where we suppose $\mathcal{P} \subset \mathbb{R}^d$ to be a compact set and $d \in \mathbb{N}^*$ to be the number of variable parameters. In order to optimise the propulsion speed, we define the cost function $j(\bar{U})$ to be a function of the swimmer's average velocity $\bar{U} = \frac{1}{T}\int_0^T U(t)\,\mathrm{d}t$ over the tail's rotation period T. The volumes of the head $|H|$ and tail $\sum_{i=1}^{n_T} |F_i|$ are fixed and equal to vol_H and vol_F. Thus, the general form of the optimisation problem reads as

$$\max_{\substack{p \in \mathcal{P}, \\ |H| = vol_H, \\ \sum_{i=1}^{n_T} |F_i| = vol_F}} \quad j(\bar{U}). \tag{I.2.4}$$

I.2.2 Numerical methods

I.2.2.1 The swimming problem

We briefly recall the discretization procedure for the boundary integral equations. As it was previously done, the components of the surface tensions are expanded as $f_j(y) = \sum_{l=1}^N f_j^l \phi^l(y)$, where $\{\phi^l\}_{l=1}^N$ span the scalar piecewise linear finite element space over ∂S, and each component of equation (I.2.3a) is projected on this space, giving

$$\int_{\partial S} U_i \phi^k(x)\,\mathrm{d}x - \int_{\partial S} ((x - x^{CM}) \times \omega)_i \phi^k(x)\,\mathrm{d}x + \int_{\partial S} \left(\int_{\partial S} G_{ij}(x,y) \sum_{l=1}^N f_j^l \phi^l(y)\,\mathrm{d}y \right) \phi^k(x)\,\mathrm{d}x = \int_{\partial S} (x - x^F) \times \omega_F \vec{e}_1^F \phi^k(x)\,\mathrm{d}x \quad \text{for } k, l = 1, \ldots, N, \tag{I.2.5a}$$

$$\int_{\partial S} \sum_{l=1}^N f_j^l \phi^l(y)\,\mathrm{d}y = 0 \quad \text{for } l = 1, \ldots, N, \tag{I.2.5b}$$

$$\int_{\partial S} (y - x^{CM}) \times \sum_{l=1}^N [f_1^l, f_2^l, f_3^l]^T \phi^l(y)\,\mathrm{d}y = 0 \quad \text{for } l = 1, \ldots, N. \tag{I.2.5c}$$

The matrix form of equations (I.2.5) reads as

$$\begin{matrix} 3\times N\times(1+n_T)\ \{ \\ 3\ \{ \\ 3\ \{ \\ \end{matrix} \begin{bmatrix} G & J^T & K^T \\ J & 0 & 0 \\ \underbrace{K}_{3\times N\times(1+n_T)} & \underbrace{0}_{3} & \underbrace{0}_{3} \end{bmatrix} \begin{bmatrix} f \\ U \\ \omega \end{bmatrix} = \begin{bmatrix} I(\omega_F) \\ 0 \\ 0 \end{bmatrix}.$$

We remark that, in this specific case, (I.2.5a) contains the tails' angular velocities that are responsible for the swimmer's propulsion.

I.2.2.2 The optimisation algorithms

Two approaches are discussed to solve the parametric shape optimisation problem of the multi-flagellated micro-swimmer. In the first case, parameters are optimised successively, one by one, while in the second approach they are all optimised simultaneously. In both cases, black-box optimisation algorithms are considered as evaluating the cost function is computationally expensive, and there is no information about the differentiability of the cost function with respect to the optimisation variables.

The algorithm we considered for the first approach is the Nelder-Mead simplex method [93], that is available in the Matlab *fminsearch* routine and is an unconstrained minimization algorithm. It consists in evolving a set of $n+1$ linearly independent points, where n is the dimension of the parameter space, in order to reach the minimum of the given function. The underlying idea of the Nelder-Mead algorithm is to eventually shrink the simplex onto the optimal point. At every iteration, the new point is searched on the line connecting the centroid of the best n vertices with the worst vertex x_{n+1}. If no improvement is obtained in this way, a shrinkage of the simplex is realized, and all the vertices but x_1 are recomputed and re-evaluated. The routine is detailed in Algorithm 1. One can remark that this algorithm only exploits the current $n+1$ points to evaluate the descent direction, and according to [93] it requires on average $48(n+1)^{2.11}$ evaluations to converge to the minimum of the cost function. Convergence results about this algorithm are proved in [72], showing linear convergence in one dimension to the minimizer, in the strictly convex case. This optimization method can be used to minimize non-differentiable or discontinuous functions, and it is often used successfully by practitioners. However, it may fail on certain functions for which the contraction step of the algorithm is applied repeatedly [86], converging to a non-stationary point. We have decided to use this method nevertheless for the low number of function evaluations necessary to converge to the minimum of the cost function and its applicability to discontinuous functions, as we don't have an explicit knowledge of the dependence of the cost on the geometric parameters of the swimmer.

The second approach is realized via Bayesian optimisation [44, 67], which also belongs to the category of black-box optimisation methods and is available in the Matlab routine *bayesopt*. In Bayesian optimisation the cost function is treated as a random variable, and its minimization is based on the construction of a surrogate model, less expensive to evaluate. This model is built from the cost function evaluations via Gaussian process regression, it is progressively updated as new evaluations are performed and it influences the choice of the next evaluation points.

Given n points $x_1, x_2, ..., x_n \in \mathbb{R}^d$ and the corresponding values of the cost function $f(x_1), f(x_2)$, $..., f(x_n)$, the prior distribution/surrogate function built from these evaluations is the normal distribution

$$f(x_{1,..,n}) \sim \mathcal{N}(\mu(x_{1,..,n}), \Sigma(x_{1,...,n}, x_{1,...,n})), \tag{I.2.6}$$

Algorithm 1: Nelder-Mead simplex algorithm

Initialize $\rho, \gamma, \sigma, \chi$. (standard values are $\rho = 1, \gamma = \frac{1}{2}, \sigma = \frac{1}{2}, \chi = 2$);
while *not converged* **do**
  Order the $n+1$ points to satisfy $f(x_1) \leq f(x_2) \leq ... \leq f(x_{n+1})$;
  Compute $x_r = (1+\rho)\sum_{i=0}^{n} x_i/n - \rho x_{n+1}$; **Evaluate** $f(x_r)$;
  if $f(x_1) \leq f(x_r) < f(x_n)$ **then**
    $x_{n+1} \leftarrow x_r$
  else if $f(x_r) < f(x_1)$ **then**
    Compute $x_e = (1+\rho\chi)\sum_{i=0}^{n} x_i/n - \rho\chi x_{n+1}$; **Evaluate** $f(x_e)$;
    if $f(x_e) < f(x_r)$ **then**
      $x_{n+1} \leftarrow x_e$
    else
      $x_{n+1} \leftarrow x_r$
  else if $f(x_n) \leq f(x_r) < f(x_{n+1})$ **then**
    Compute $x_c = (1+\rho\gamma)\sum_{i=0}^{n} x_i/n - \gamma\rho x_{n+1}$; **Evaluate** $f(x_c)$;
    if $f(x_c) < f(x_r)$ **then**
      $x_{n+1} \leftarrow x_c$
    else
      Substitute x_i, $i = 2, ..., n+1$ with $v_i = x_1 + \sigma(x_i - x_1)$
  else if $f(x_r) \geq f(x_{n+1})$ **then**
    Compute $x_c = (1-\gamma)\sum_{i=0}^{n} x_i/n + \gamma x_{n+1}$; **Evaluate** $f(x_c)$;
    if $f(x_c) < f(x_{n+1})$ **then**
      $x_{n+1} \leftarrow x_c$
    else
      Substitute x_i, $i = 2, ..., n+1$ with $v_i = x_1 + \sigma(x_i - x_1)$

where the mean μ and covariance Σ are computed from previously chosen mean and covariance functions, as the sample mean and the Màtern kernel respectively. Using the Bayes rule, it is possible to infer the value of the cost function at a point x and construct the posterior distribution $f(x)|f(x_{1,...,n}) \sim \mathcal{N}(\mu_{1,..,n}(x), \Sigma_{1,...,n}(x))$ where

$$\begin{aligned} \mu_{1,..,n}(x) &= \Sigma(x, x_{1,...,n})\Sigma(x_{1,...,n}, x_{1,...,n})^{-1}(f(x_{1,...,n}) - \mu(x_{1,..,n})) + \mu(x), \\ \Sigma_{1,...,n}(x) &= \Sigma(x,x) - \Sigma(x, x_{1,...,n})\Sigma(x_{1,...,n}, x_{1,...,n})^{-1}\Sigma(x_{1,...,n}, x), \end{aligned} \tag{I.2.7}$$

are the updated mean and covariance. Based on this posterior probability distribution, one chooses the next evaluation point according to an acquisition function, like the expected improvement. The expected improvement is defined as the conditional expectation

$$EI_n(x) = \mathbb{E}\Big[[\min f(x_{1,...,n}) - f(x)]^+ | x_{1,...,n}, f(x_{1,...,n})\Big], \tag{I.2.8}$$

where $[\min f(x_{1,...,n}) - f(x)]^+$ denotes the positive part of $[\min f(x_{1,...,n}) - f(x)]$, and the next evaluation point can be chosen as $x_{n+1} = \arg\max EI_n(x)$. The maximum of $EI_n(x)$ is to be found in regions where the posterior mean and standard deviation are high. In more generality, this maximization is a trade-off between exploring regions where little information is known and exploiting points were the expected posterior mean is larger than the cost function at the previous best point.

The previous model works when the optimisation variables belong to simple domains, like hyper-rectangles, but extensions of it allow the treatment of constrained problems where the constraints are also expensive to evaluate (typically if they depend on the black-box outputs) [60]. In this case, the expected improvement function will contain a term that weights the probability that

point x is feasible. Derivative-free optimisation algorithms have already proved effective in micro-swimming problems, showing that helical propulsion is more efficient in the case of larger cell bodies for sperm cells [56].

Details on the optimisation

When using the Nelder-Mead algorithm, we considered a non-convex cost function $j(\bar{U}, \kappa) = \bar{U}_1^2 - \kappa(\bar{U}_2^2 + \bar{U}_3^2)$, and we maximize it under the constraint of constant tail arc-length. The parameter κ can be used to penalize transversal speeds as they can affect the forward propulsion. We expect that using this type of cost function, where transversal speeds are further penalised, will give slower but more stable swimmers. The procedure we follow depends on the micro-swimmer under consideration, but it is essentially based on subsequent optimisation of its parameters. When using Bayesian optimisation, the cost function is the first component of the average velocity, $j(\bar{U}) = \bar{U}_1$. We consider additional constraints on $\bar{U}_2, \bar{U}_3$ defined by

$$|\bar{U}_2|, |\bar{U}_3| \leq \varepsilon, \tag{I.2.9}$$

where ε is a tolerance to be determined. In this case, the parameters of each swimmer are optimised all at once.

In our simulations, the number of propulsion speed evaluations in each average is four, and they are obtained when the tail is rotated around its axis by angles of $0.5\pi t$, for $t \in \{0, 1, 2, 3\}$. Using $k = 4$ evaluations instead of 6, 8 or 12 does not change the speed values in the propulsion direction, nor the swimmer angular velocity in that same direction. The transversal velocities change, but they remain of the same order they had in the 4 evaluations case. We limit the number of function evaluations of the Nelder-Mead algorithm to 30. Such value is a compromise between the accuracy of results and the time required for getting them. In figure I.2.2 an evaluation on optimal tails' parameters for bi-flagellated swimmers shows that, between 20 and 30 evaluations, the difference in radii and wavelengths is significant (ranging from 40% to 20%), while these same differences between 30 and 45 evaluations amounted to 5% only. For what concerns Bayesian optimisation, the function evaluation budget ranges from 200 to 250. We will denote a quantity as "optimised" anytime it is the output of one of the previous optimisation procedures.

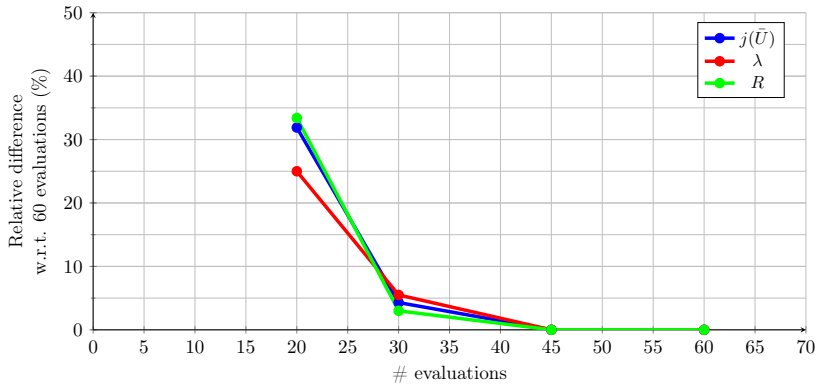


Figure I.2.2: This figure justifies the choice of 30 function evaluations in the optimisation process: performing 45 or 60 evaluations leads to values that differ of about 5%.

Symbol	Value (dimensionless)	Value (dimensional)
R_1^h	0.874	0.65 µm
R_2^h	0.874	0.65 µm
R_3^h	$1.5R_1^h$	0.975 µm
L	3.0	2.2 µm
r	0.067	50 nm
R^t	0.2	0.15 µm
λ	1.0	0.74 µm
k_E	$0.333 \cdot 2\pi/\lambda = 2.09$	$2.8\,\mu m^{-1}$
l	$2r = 0.134$	100 nm

Table I.2.1: Parameters describing the body and tail shape of the bacterium. The length-scale for the non-dimensionalisation is $\lambda = 0.74\,\mu m$. This table is taken from [118].

I.2.3 Results for the first approach

In this section we address the optimisation of several swimmers. We first consider a mono-flagellated micro-swimmer and optimise, successively, its tail wavelength and radius, its head, and its tail section. Secondly, a bi-flagellated swimmer is considered, and the placement of the tails and the head shapes are successively investigated. A symmetry constraint is imposed on the shape in order to ensure a motion along the x-axis.

Mono-flagellated swimmer. The optimisation of the mono-flagellated swimmer is addressed in this section, according to the cost function $j(\bar{U}, \kappa)$. In order to validate our method, we reproduced [100, fig. 6]. This graph represents the non-dimensional mean swimming speed of a mono-flagellated micro-swimmer as a function of the ratio between its tail length and the radius of its spherical head. The comparison is reported in figure I.2.3; blue squares are the samples of [100, fig. 6] while green circles are the result of our algorithm combining BEM and Nelder-Mead optimisation. The red triangle denotes the optimum found by our algorithm, which shows that the optimisation process finds the same optimum as in the previous study. The velocity value that is found is not exactly the same, and it might be attributed to the different domain discretization (coarse in [100] and fine here) or the boundary elements (linear in our case and piecewise constant in [100]). The Nelder-Mead method is applied to few parameters at once (first two tail parameters, then two head parameters given the previous optimal tail) to limit the number of function evaluations necessary to increase the propulsion speed. Choosing this sequence of steps and not the opposite (optimize first the head, then the tail) was an arbitrary choice, but we expect that the optimal shape arising from this second approach will not have a significantly worse performance than the one found here.

Tail optimisation. Firstly, fixing the values of head parameters, tail section and arc-length as in table I.2.1, the helix radius R^t and the wavelength λ are optimised. In figure I.2.4, three optimal tails are presented for swimmers with *prolate* ($R_1^h > R_3^h$), spherical and *oblate* heads ($R_1^h < R_3^h$) of equal volume. This first analysis returns as fastest swimmer the one with a *prolate* head, depicted in figure I.2.4A, followed by the one with a spherical head, depicted in figure I.2.4B. We notice that the tail shape is qualitatively very similar in the three cases. In figure I.2.5 we compare the initial tail, as introduced in [118], and the optimal tail found by the optimisation method using $\kappa = 10$ and an *oblate* head. The resulting tail has larger radius and larger wavelength than the tail in [118], our initial guess. As a result, the optimal tail is not

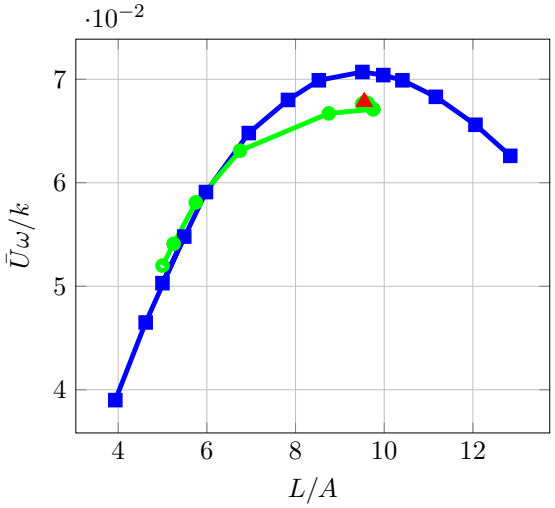
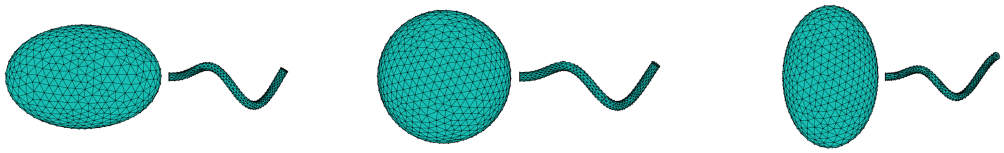


Figure I.2.3: Comparison of [100, Fig. 6] (blue thick line) and our results (green circles and red triangle). On the x-axis the ratio L/A is represented, where L is the tail length and A is the radius of the spherical head. On the y-axis the ratio $\bar{U}\omega/k$ is represented, where $\omega = 2\pi$, $k = 1.5$ is the wave parameter such that $k/k_E = 1$, and $\bar{U}$ is the mean propulsion speed. The optima roughly coincide, and the function values differ by 5% maximum.

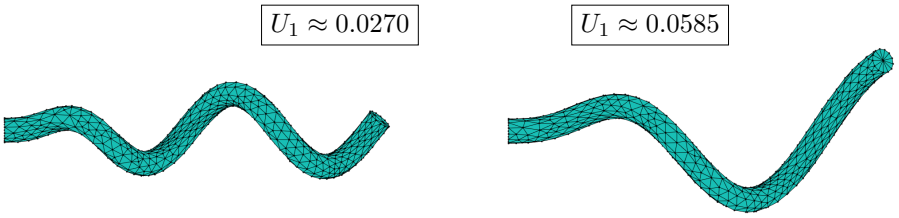
forming a complete turn of radius R^t anymore, and it rather resembles to a conical helix than to a cylindrical one. The figure reports the values of the swimming speeds, that is doubled with respect to its initial value.

Head optimisation. Starting from the swimmer in figure I.2.4C the head shape is optimised under the constraint of constant volume. The resulting head has the same volume as the initial one (i.e. the head from [118]; see Table I.2.1 for the values). Figure I.2.6B shows the optimised head, obtained with a maximal number of function evaluations equal to 30. The optimal head shape is a *prolate* ellipsoid, with $R_2^h \approx R_3^h$ and $R_1^h \approx 3R_3^h$. In other words, the sections orthogonal to the propulsion direction are circular, and the sections containing the tail's axis are ellipses with an aspect ratio of 3.



(A). *Prolate* head. (B). Spherical head. (C). *Oblate* head.

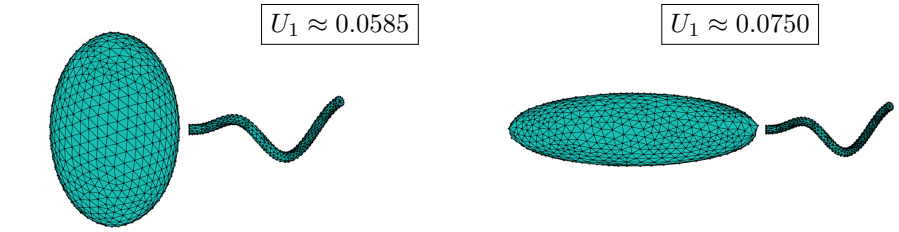
Figure I.2.4: Comparison of optimal tails in the case of three head types with the same volume. The fastest swimmer is on the left and the slowest on the right. We used the cost function $j(\bar{U}, \kappa)$ with $\kappa = 10$, implementing further penalisation of lateral velocities to stabilise the forward motion.



(A). Initial tail from [118]. R^t =0.15 µm and λ =0.74 µm.

(B). Optimised tail. R^t =0.30 µm and λ =1.26 µm.

Figure I.2.5: In sub-figure (A) the tail of [118] is presented. In sub-figure (B), the optimised tail for the mono-flagellated swimmer is represented. The latter was obtained by optimising R^t and λ according to $j(\bar{U}, \kappa)$, with initial shape (A), with 30 evaluations in the Nelder-Mead algorithm. The optimised tail has larger radius and larger wavelength than the tail in [118], our initial guess. The optimised shape is not forming a complete turn of radius R^t anymore, and it rather resembles to a conical helix than to a cylindrical one. The value of the propulsion velocity U_1 is given in the two cases.



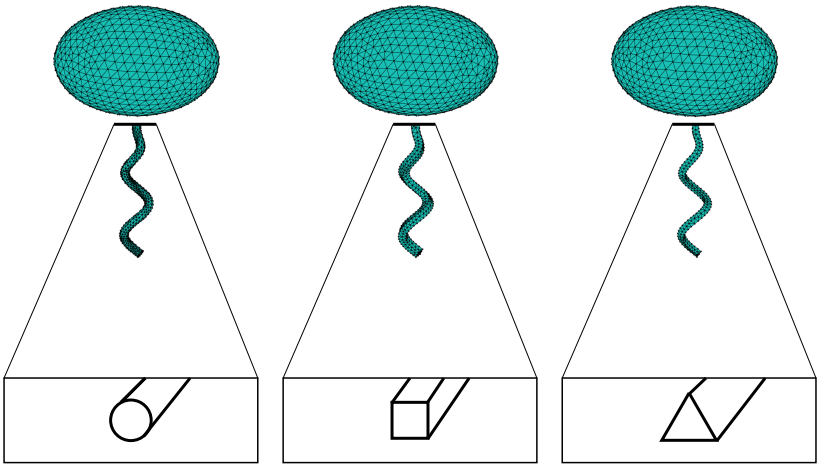
(A). Initial head from [118] with optimal tail. $R_1^h = R_2^h$ =0.65 µm, R_3^h =0.975 µm.

(B). Optimised head with optimal tail. R_1^h =1.52 µm, $R_2^h = R_3^h$ =0.5 µm.

Figure I.2.6: Comparison between the initial and the optimal head in the mono-flagellated case. The optimal head shape is a *prolate* ellipsoid, with $R_2^h \approx R_3^h$ and $R_1^h \approx 3R_3^h$. The sections orthogonal to the propulsion direction are circular, and the sections containing the tail's axis are ellipses with an aspect ratio of 3. Sub-figure (A) shows the head from [118] and the optimised tail previously computed, while sub-figure (B) shows at the same time the optimised head and tail. The value of the propulsion velocity U_1 is given in the two cases.

Tail section optimisation. This paragraph is devoted to studying the effects that tail section has on the propulsion speed. Three geometries are compared, under the constraint of constant sectional area, and therefore constant tail volume. The helical tail is built by transporting the section at $s = 0$ parallel to itself along the centerline, i.e. instead of using the Frenet-Serret reference frame for the curve, we use here a parallel transported frame (see [51]). A parallel transported frame prevents the section to twist around the centerline as it is displaced, while in the Frenet-Serret frame case the orientation of the section depends on the torsion of the centreline.

Once more, we are interested in selecting the tail which ensures the fastest propulsion. We compare the mean speeds for three geometrical forms of the tail section: the circle, the triangle and the square. As it is reported in figure I.2.7, the triangular section guarantees a larger displacement speed. This result is explained by considering that, even if the area is constant, the



Section	Propulsion speed	Perimeter
Circle	0.0271	$2\pi r \approx 6.28r$
Square	0.0290	$4\sqrt{\pi}r \approx 7.09r$
Triangle	0.0302	$6\sqrt{\pi}r/\sqrt[4]{3} \approx 8.08r$

Figure I.2.7: Comparison between the propulsion speeds of different tails sections. The sections have the same area πr^2 but they are different in shape. The parameters are the same as in [118], and in particular $r = 0.067$. The tail with a triangular section produces a larger propulsion speed, followed by the square one and the circular one.

perimeter changes when passing from circular to polygonal sections. Larger sectional perimeter leads to larger contact surface with the fluid, which enhances the propulsion. Thus, being the triangle the geometrical figure with the larger perimeter among the ones we considered, it shows a larger propulsion speed. These results are in line with the velocity pattern shown by the experiments in [108]. Since we wanted to test the dependence of propulsion speed with respect to the fluid-tail contact surface, we used the oblate head and initial tail shape to perform the simulations as the same conclusion would have been true for other head and tail shapes.

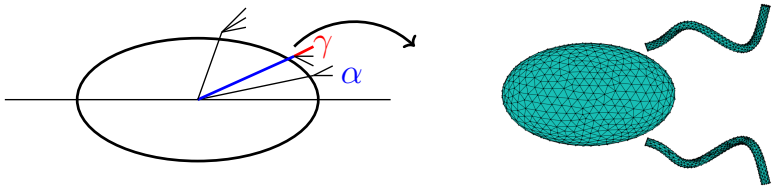


Figure I.2.8: Schematization of (α, γ) variability in the study. For each of the three latitudes α, from two to three tail attachment angles γ are chosen. In the figure, an example is reproduced where $(\alpha, \gamma) = (0.3\pi, 0.1\pi)$.

Bi-flagellated swimmer This section focuses on the head optimisation for a swimmer with two flagella, inspired by MO-1 (see [118]), using the same procedure we have introduced in section I.2.3. As it is schematically depicted in figure I.2.8, eight tail configurations are tested. During the optimisation process, the tail positions were constrained to be symmetric with respect to the horizontal plane. Finally, the head volume was kept constant and equal to the initial volume, calculated from table I.2.1. In figure I.2.9, the optimised heads are depicted according to the latitude of the junctions and their angle of attachment: on the columns, the latitude angle α is indicated, while on the rows the tail orientation angle γ is reported. Figure I.2.9 shows that a larger mean speed is reached mainly in the case of a *prolate* head. Only two situations lead to an optimised *oblate* head shape, and they are reported in red. Finally, the best configuration in terms of head shape, orientation and position of the tails, is reached for a swimmer having a *prolate* head and angle of attachment $\gamma = 0$ (see the blue swimmer in figure I.2.9).

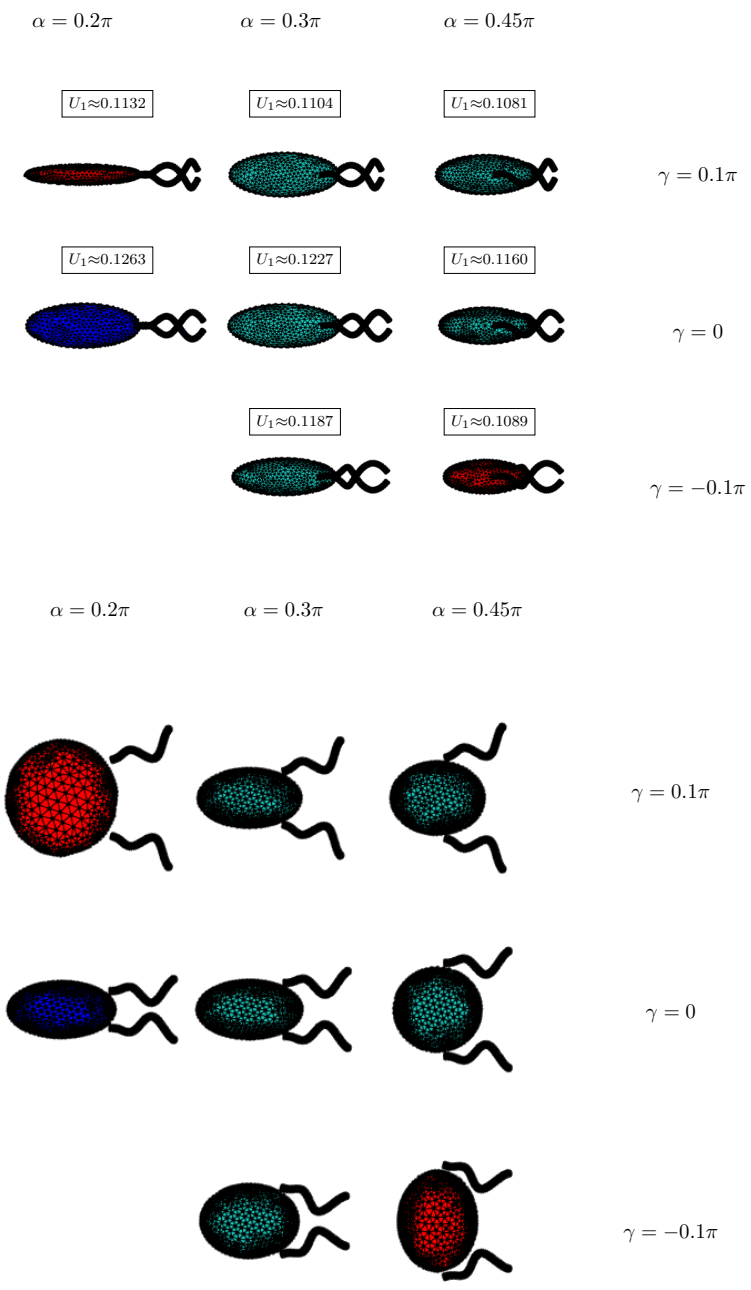


Figure I.2.9: Optimal heads in the bi-flagellated case as function of the tail junctions' latitude α and attachment angles γ. The image above shows the xy projections of the swimmers (radii $R_1^h - R_2^h$), while the image below shows the xz projections (radii $R_1^h - R_3^h$). Leftmost swimmers have $\alpha = 0.2\pi$, middle swimmers have $\alpha = 0.3\pi$, rightmost swimmers have $\alpha = 0.45\pi$. Green and blue swimmers have a *prolate* optimal head, red swimmers have an *oblate* optimal head. The blue swimmer is the fastest propelling one. The value of the propulsion velocity U_1 is given.

Parameter	Min	Max	Parameter	Min	Max
R^t	0.1	1	R_1^h	0.5	1.5
λ	0.3	4	R_3^h	0.5	1.5
α	0	0.5π	γ	-0.3π	0.3π
β	0	π	δ	-0.5π	0.5π

Table I.2.2: Parameter intervals for the Bayesian optimisation algorithm.

I.2.4 Results of the second approach

In this section we address the shape optimisation of bacteria-inspired swimmers using Bayesian optimisation. In this section the parametrisation of the swimmers included further variables in order to represent all tail positions and orientations. We suppose the swimmer to be composed of an ellipsoidal cell body (a volume constraint is prescribed also in this case) and n_T helical tails as in (I.2.2). If $n_T \geq 2$, one of the tails is identified as a reference and four angles are introduced to describe its position and orientation; the remaining tails are symmetrically arranged with respect to the propulsion direction. As in the previous chapter, α denotes the latitude of the tail junction and γ the inclination angle with respect to the horizontal plane xy; angle β denotes the longitude of the junction with respect to the xz meridian and δ the inclination angle with respect to the vertical plane xz.

The parameter space of the optimisation is a hyper-rectangle $\mathcal{P} \subseteq \mathbb{R}^d$, and its extrema are collected in table I.2.2. The optimisation problem reads

$$\max_{\substack{p\in\mathcal{P},\\ |H|=vol_H,\\ \sum_{i=1}^{n_T}|F_i|=vol_F\\ |\bar{U}_2|\leq\varepsilon,|\bar{U}_3|\leq\varepsilon}} -\bar{U}_1. \tag{I.2.10}$$

The shape optimisation of swimmers was initialised with 50 shapes. One of these shapes was provided manually, using the values that were obtained with the Nelder-Mead optimisation, while the 49 remaining ones were obtained via Latin hypercube sampling on $\mathcal{P}$.

It is possible that the optimisation algorithm proposes a set of parameters that leads to tail-tail intersection or tail-body intersection during its exploration of the parameter space. In order to detect these unphysical cases and avoid the solution of the BEM system, we compute the convex envelopes of tails and head, using the `alphaShape` routine in *Matlab*. For each pair of tail-tail or body-tail envelopes, we look for intersections by checking if points of one envelope are contained in the other. Considering the convex envelope of the helical tails allows us to also check if any intersection will happen during the rotation of the tail around its axis. If an intersection is detected, the corresponding set of parameters is declared unfeasible.

Mono-flagellated swimmer. Different values ε were tested to reduce the transversal motion of the swimmer, and the value that was kept is $\varepsilon = 0.01$. The optimal shape that was obtained for the mono-flagellated swimmer is reported in figure I.2.10: letting the placement of the tail junction free, the optimal shape resulted in a prolate head and a tail which is eccentric with respect to the propulsion direction. The optimal parameters are reported in table I.2.3 and none of them is located on the boundary of the feasible region. The average propulsion velocity in this case is $U_1 \approx 0.089$, which is larger than the propulsion velocities obtained in the first case.

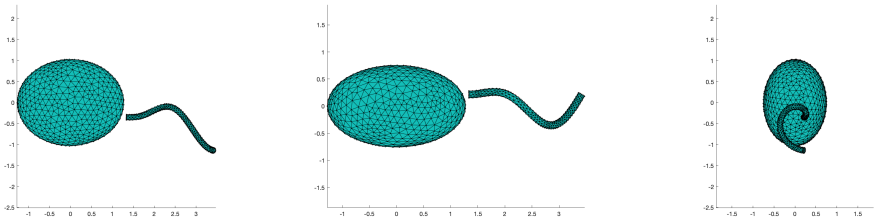


Figure I.2.10: Optimal mono-flagellated swimmer. From the left: swimmer $R_1 - R_3$ plane, $R_1 - R_2$ plane and $R_2 - R_3$ plane.

Parameter	Optimal value	Parameter	Optimal value
R^t	0.819	R_1^h	1.278
λ	2.792	R_3^h	1.034
α	-0.301	γ	-0.019
β	1.303	δ	0.001

Table I.2.3: Parameters describing the geometry of the mono-flagellated optimal swimmer in figure I.2.10. The optimal speed in this case is $U_1 \approx 0.089$.

Bi-flagellated swimmer. The presence of two flagella increases the number of parameters of the problem, as the position and orientation of each flagellum is described by four angles. In order to investigate whether four angular parameters per tail are necessary, we fixed the position of the tails to be symmetric with respect to the propulsion direction and let the angles $\gamma_1, \delta_1, \gamma_2, \delta_2$ responsible for the orientation, free to change. We observed that the optimal configuration in this case was obtained by a swimmer whose tail orientations were also symmetric with respect to the direction of propulsion. For this reason, only a set of four angles is used to describe the position and orientation of the tails. Starting from the reference tail, whose configuration is dictated by $\alpha, \beta, \gamma, \delta$, the other tail is created by a rotation of 180° around the propulsion direction. The optimal swimmer is depicted in figure I.2.9, and it has an oblate head. The tails, rotating around their axes, are very close to the body of the swimmer and the helices they form are very elongated. All the parameters, whose values are collected in table I.2.4, are in the interior of $\mathcal{P}$ and among them δ is the closest one to one extremum of the feasible interval. The propulsion velocity for this swimmer is extremely large, $U_1 \approx 1$, larger that the other propulsion speeds of two orders of magnitude.

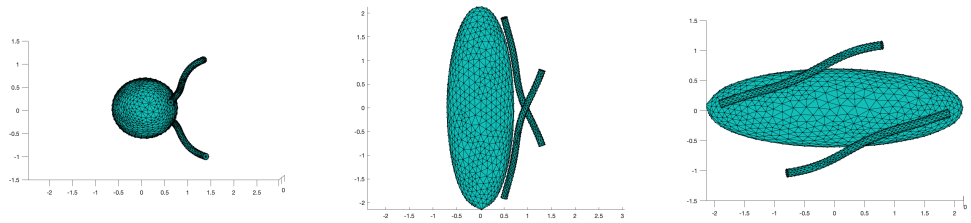


Figure I.2.11: Optimal swimmer in the bi-flagellated case. From the left: swimmer $R_1 - R_3$ plane, $R_1 - R_2$ plane and $R_2 - R_3$ plane.

Parameter	Optimal value	Parameter	Optimal value
R^t	0.104	R_1^h	0.709
λ	3.133	R_3^h	0.654
α	0.102	γ	0.328
β	2.465	δ	1.316

Table I.2.4: Parameters describing the geometry of the optimal bi-flagellated bacterium in figure I.2.11. The optimal speed in this case is $U_1 \approx 1$.

Tetra-flagellated swimmer. In the tetra-flagellated swimmer case, the four flagella are located at an angular distance of $\frac{\pi}{2}$ from each other, ensuring a symmetric body configuration as in the bi-flagellated case. The optimal shape of the tetra-flagellated swimmer is depicted in figure I.2.12. in this case the head of the swimmer is flattened in the propulsion direction and the tails are similar to the optimal tails found in the bi-flagellated case. Table I.2.5 contains the values of the optimal parameters describing the tetra-flagellated swimmer. The propulsion speed of this swimmer is $U_1 \approx 1.42$, which is larger than the bi-flagellated swimmer's velocity but not linearly increasing with the number of tails.

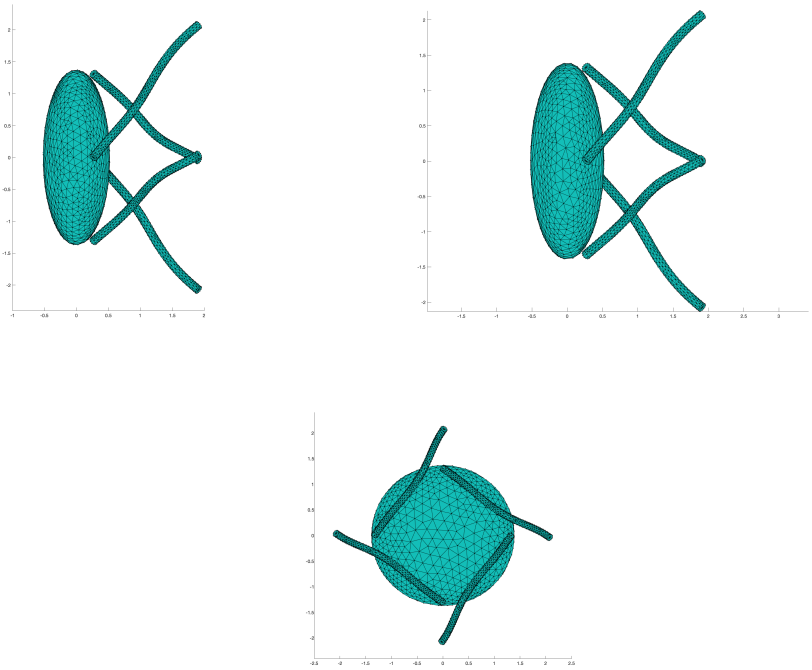


Figure I.2.12: Optimal swimmer in the tetra-flagellated case. From the left: swimmer $R_1 - R_3$ plane, $R_1 - R_2$ plane and $R_2 - R_3$ plane.

Parameter	Optimal value	Parameter	Optimal value
R^t	0.11201	R_1^h	0.52329
λ	3.4106	R_3^h	1.3729
α	1.152	γ	-0.74737
β	2.6615	δ	-0.72203

Table I.2.5: Parameters describing the geometry of the optimal bacterium in figure I.2.12. The optimal speed in this case is $U_1 \approx 1.42$.

I.2.5 Discussion

The shape optimisation of multiple micro-swimmers has been addressed, choosing to maximize the advancement speed in the $-x$ direction. Using the first approach, tail shape, head shape and tail section shape optimisation were addressed in the mono-flagellated case. All these results are in line with the literature: in the optimisation process, the number of waves in the optimal tail approaches unity, which was seen to be optimal for swimmers with shorter tails in [54, 100]. The number of turns is also in agreement with the experimental findings in [128], who find (for ferromagnetic robots with thicker flagella, whose head and tails are integral) the value of about 1.1 as the optimal one. The tail section optimisation fits the experimental results in [108]: triangular tail sections have a larger perimeter and allow larger propulsion speeds since they offer a wider contact surface to the surrounding fluid. Regarding the bi-flagellated case, the head shape was optimised with respect to the different tail configurations. The *prolate* heads appears to be the hydrodynamical best shape in a wide number of tail configurations, while *oblate* heads, as the one of MO-1 bacteria, appear to be optimal in a restricted subset of tail configurations.

When using the second optimisation approach and allowing the tails to orient outside the plane, different optimal shapes are obtained. In the multi-flagellated cases, propulsion speeds are also significantly larger than the ones obtained via the first approach, perhaps due to the increased symmetry of the swimmer's tail-body configuration. The presence of multiple tails does not increase the propulsion speed proportionally to their number, but rather sub-linearly. Moreover, the cell's body seems to elongate in the directions of the tails' junctions: *prolate* in the mono-flagellated case, *oblate* in the bi-flagellated case, flat and symmetric in the tetra-flagellated case.

In Table I.2.6 the parameters corresponding to the optimal shape of the swimmers are compared. The parameters found with the Nelder-Mead approach are normalised using the initial value of $\lambda = 0.74\,\mu\text{m}$, as results of Bayesian optimisation are. It is clear that the optimal shapes are not the same, because in the Bayesian optimisation all parameters were optimized at once and more shapes could be visited by allowing the angles $\alpha, \beta, \gamma, \delta$ (needed to localize the tail on the ellipsoid and define its orientation) to vary as well. Even though it is Bayesian optimization which ensured faster propelling swimmers in both cases, the optimal shapes found with the Nelder-Mead approach fell in the same categories (prolate head and funnel-like tail for the mono-flagellated; oblate head when tails have negative γ in the bi-flagellated case) of the other method.

Swimmer	Parameter	Optimal value (Nelder-Mead)	Optimal value (Bayesian)
Mono-flagellated	R^t	0.405	0.819
	λ	1.702	2.792
	R_1^h	2.054	1.278
	R_3^h	0.675	1.034
Bi-flagellated	R^t	0.405	0.104
	λ	1.702	3.133
	R_1^h	1.704	0.709
	R_3^h	0.885	0.654

Table I.2.6: Comparison of the optimal swimmer shape parameters found with Nelder-Mead and Bayesian optimization.

I.2.6 Explicit formulas for matrices in (I.2.5)

The matrix form of equations (I.2.5) is

$$\begin{matrix} {\scriptstyle 3\times N}\{ \\ {\scriptstyle 3}\{ \\ {\scriptstyle 3}\{ \end{matrix} \begin{bmatrix} G & J^T & K^T \\ J & 0 & 0 \\ \underbrace{K}_{3\times N} & \underbrace{0}_{3} & \underbrace{0}_{3} \end{bmatrix} \begin{bmatrix} f \\ U \\ \Omega \end{bmatrix} = \begin{bmatrix} I(\omega) \\ 0 \\ 0 \end{bmatrix}.$$

Sub-matrix G is composed of $(1+n_T)^2$ sub-matrices $G^{\{A,B\}}$, for $A, B \in \{\partial H, \partial F_i\}$. Let us define as N_A and N_B the cardinality of the scalar finite element subspaces corresponding to the degrees of freedom over A and B, respectively. Each of the $G^{\{A,B\}}$ is subdivided into $N_A \times N_B$ sub-matrices of size 3×3, named $G^{\{A,B\}}_{ij,lk}$ defined as

$$G^{\{A,B\}}_{ij,lk} := \int_A \int_B G_{ij}(x,y)\phi_l(x)\phi_k(y)\,\mathrm{d}x\mathrm{d}y, \tag{I.2.11}$$

for $i,j = 1,2,3$ and $l = 1,\dots,N_B$, $k = 1,\dots,N_A$.

Sub-matrix J is composed of $(1+n_T)$ sub-matrices J^A for $A \in \{\partial H, \partial F_i\}$. Each of the J^A has size $3\times 3N_A$, it is block diagonal and it components $J^A_{il,j}$ are $3\times N_A$ sub-matrices defined as

$$J^A_{il,j} := \int_A \vec{e}_i \phi_l(x)\,\mathrm{d}x, \tag{I.2.12}$$

for $i = 1,2,3$ and $l = 1,\dots,N_A$ inside each block and $j = 1,2,3$ denoting the diagonal block.

Sub-matrix K is composed of $(1+n_T)$ sub-matrices K^A for $A \in \{\partial H, \partial F_i\}$. Each of the K^A has size $3\times 3N_A$, and its components $K^A_{ij,l}$ are defined as

$$K^A_{ij,l} := \int_A [(x - x^{CM}) \times \vec{e}_i]_j \phi_l(x)\,\mathrm{d}x, \tag{I.2.13}$$

for $i,j = 1,2,3$ and $l = 1,\dots,N_A$.

Vector $I(\omega)$ is composed of (n_T) non-zero sub-vectors $I(\omega)^A$ for $A \in \{\partial F_i\}$, and one zero sub-vector for $A = \partial H$. Each of the non-zero sub-vectors $I(\omega)^{A_i}$ is subdivided into N_A sub-vectors of size 3, named $I(\omega)^{A_i}_{j,l}$ defined as

$$I(\omega)^{A_i}_{j,l} = \int_{\partial A_i} [(x - x^{A_i}) \times \omega \vec{e}_1^{A_i}]_j \phi_l(x)\,\mathrm{d}x, \tag{I.2.14}$$

for $j = 1,2,3$ and $l = 1,\dots,N_A$.

Chapter I.3

Swimming of a bi-flagellated bacterium with elastic junctions

In this chapter we consider the MO-1 bacterium model to study its swimming behaviour under the hypothesis of elastic tail-body junctions. While in the previous section the tails were rotating at a fixed angle, here they are free to bend at the junctions as the elastic effects are modelled via two torque springs at each junction. Since the two tail bundles are sufficiently far apart, we first study an RFT model describing the steady swimming of the bacterium in section I.3.1. In section I.3.2 we include elasticity at the junctions in the BEM numerical model and present an initial testcase.

I.3.1 The RFT model

I.3.1.1 Geometry

The swimmer being addressed in this section is composed of a spheroidal head (i.e. among the three radii that define an ellipsoid, two of them are equal) and of two tails. The tails are Higdon helices of equations (I.2.2), given in the tail's reference frame $Oxyz$. Here, the origin O coincides with $(0,0,0)$, the first axis is tangent to the curve at $s=0$, and the other two span the perpendicular plane to the tangent vector. The parameter s takes values in $[0, X_M]$, R^t is the helix radius, λ is the tail's wavelength and $k_E = 0.33 * 2\pi/\lambda$. In this parametrization, the s variable is not the arc-length, hence X_M does not coincide with the length of the tail, but rather with the maximal value taken in the local X direction. The arc-length of the line is given by

$$a(s) = \int_0^s \sqrt{1 + (\frac{dy(r)}{dr})^2 + (\frac{dz(r)}{dr})^2} dr, \tag{I.3.1}$$

from which one can compute the derivative with respect to s

$$\frac{da(s)}{ds} = \sqrt{1 + (\frac{dy(s)}{ds})^2 + (\frac{dz(s)}{ds})^2}, \tag{I.3.2}$$

that in turn can be rewritten, calling $E(s) = (1 - e^{-k_E^2 s^2})$, as

$$\frac{da(s)}{ds} = \sqrt{1 + (R^t \frac{dE}{ds}(s))^2 + (2\pi R^t/\lambda E(s))^2}. \tag{I.3.3}$$

I.3.1.2 The balance of forces and torques

The absence of inertial effects in the low Reynolds number regime simplifies Newton's equations to a balance of forces and torques where the inertial term is absent. In the following, we make the assumption that the bending at the junctions is constrained on a plane (the one containing the angle γ, cf. figure (I.2.1)). The balance of viscous drag and propulsion forces for the bi-flagellated swimmer, projected in the propulsion direction and expressed in the laboratory frame, is

$$[C_D + 2K(\gamma_{eq})]U_x - 2F_{prop}\cos(\gamma_{eq}) = 0. \tag{I.3.4}$$

Here, C_D is the head resistance matrix, $K(\gamma_{eq})$ is the tail resistance matrix, F_{prop} is the propulsion force vector, γ_{eq} indicates the bending angle at which the torque balance at the junction is realized and U_x denotes the translational velocity in the propulsion direction. The prefactor 2 indicates that the contributions coming from the two flagella add up. The balance of torques over the flagellum reads as

$$L(\gamma_{eq})U_x - G(k, \gamma_{eq} - \bar{\gamma}) = 0, \tag{I.3.5}$$

where $L(\gamma_{eq})$ is the tail resistance matrix related to torque effects, G is a function of k, the torque spring elasticity, and $\gamma_{eq} - \bar{\gamma}$ the angle between the tail's axis in its equilibrium position and the tail's axis in its rest position. The function $G(k, \gamma_{eq} - \bar{\gamma})$ can be either $G(k, \gamma_{eq} - \bar{\gamma}) = k(\gamma_{eq} - \bar{\gamma})$ (if we want to use the linear torque spring model), or $G(k, \gamma_{eq} - \bar{\gamma}) = k\tan(\gamma_{eq} - \bar{\gamma})$. This latter model is preferable since it behaves linearly for small displacements, but penalises large deviations with stronger recovery forces. In particular, it can prevent unphysical interpenetrations between tails and cell body. For the reasons just given, we use the second model for simulations, while the first one will be useful for testcases.

I.3.1.3 The resistance matrices and propulsion force

The resistance matrix C_D is computed according to [68, pag.64-68] for prolate-oblate spheroids, in the head reference frame. In the prolate and oblate case, the drag force F has a similar form to Stokes' drag law

$$C_D = X_A \vec{d} \otimes \vec{d} + Y_A(\mathbb{I} - \vec{d} \otimes \vec{d}) \tag{I.3.6}$$

where X_A and Y_A have different expressions depending on the shape of the spheroid and $\vec{d}$ denotes the symmetry axis of the spheroid. We remark that the definition of eccentricity for X_A and Y_A is $e = \sqrt{a_K^2 - c_K^2}/a_K$ where a_K is the larger semi-axis of the spheroid [68].

The resistance matrix $K_f(\gamma_{eq})$ is computed in the tail's reference frame by integrating $dF = -[c_\parallel \hat{t} \otimes \hat{t} + c_\perp(\mathbb{I} - \hat{t} \otimes \hat{t})]\, ds$ over the flagellum, where $c_\parallel = (4\mu\pi)/(\ln(2\lambda/r) + 0.5)$ and $c_\perp = (2\mu\pi)/(\ln(2\lambda/r) - 0.5)$, λ is the tail wavelength, r the tail's thickness. Here ds is the arc-length, $\hat{t}$ is the normalised tangent vector to the helix, expressed using the arc-length parametrization. So,

$$K_f(\gamma_{eq}) = \int_0^L dF = -\int_0^{X_M} [c_\parallel \hat{t}(x) \otimes \hat{t}(x) + c_\perp(\mathbb{I} - \hat{t}(x) \otimes \hat{t}(x))] \frac{ds}{dx}\, dx, \tag{I.3.7}$$

using the parametrization in (I.2.2), and (I.3.3) for ds/dx.

The resistance matrix $L_f(\gamma_{eq})$ is computed in the tail's reference frame as well, by integrating $x \times dF = -x \times [c_\parallel \hat{t} \otimes \hat{t} + c_\perp(\mathbb{I} - \hat{t} \otimes \hat{t})]\, ds$ over the flagellum

$$L_f(\gamma_{eq}) = \int_0^L x \times (dF \cdot (e_i \times x)) = -\int_0^{X_M} x \times ([c_\parallel \hat{t}(x) \otimes \hat{t}(x) + c_\perp(\mathbb{I} - \hat{t}(x) \otimes \hat{t}(x))] \cdot (e_i \times x)) \frac{ds}{dx}\, dx, \tag{I.3.8}$$

again using the parametrization in (I.2.2), and (I.3.3) for ds/dx.

The propulsion force vector is F^f_{prop}, in the tail reference frame, is computed as follows

$$F^f_{prop} = \int_0^{X_M} dF \cdot (\omega_{tail} \times x)\, dx = -\int_0^{X_M} [c_\parallel \hat{t}(x) \otimes \hat{t}(x) + c_\perp(\mathbb{I} - \hat{t}(x) \otimes \hat{t}(x))] \cdot (\omega_{tail} \times x)\, \frac{ds}{dx}\, dx, \tag{I.3.9}$$

where ω_{tail} is the imposed angular velocity, which is directed along the tail's axis.

These integrals are computed in the tails' reference frame, and are successively expressed in the laboratory frame by multiplication with the rotation matrix

$$R = \begin{bmatrix} \tilde{e}_1 & \tilde{e}_2 & \tilde{e}_3 \end{bmatrix}, \tag{I.3.10}$$

where $\tilde{e}_i$, $i \in \{1, 2, 3\}$ is the tail local basis expressed in the laboratory reference frame. Hence, in the laboratory frame, we get $K(\gamma_{eq}) = RK_f(\gamma_{eq})R^T$, $L(\gamma_{eq}) = RL_f(\gamma_{eq})R^T$ and $F_{prop} = RF^f_{prop}$.

In the following simulations, we fix $\bar{\gamma}$ to be the angle between laboratory x axis and the normal to the head at the tail's attachment point. Such normal is given by $\vec{n} = (x/(R^H_1)^2, y/(R^H_2)^2, z/(R^H_3)^2)$. Oblate spheroids with high eccentricity will have $\bar{\gamma}$ close to 0, while prolate spheroids with high eccentricity will have $\bar{\gamma}$ close to $\pi/2$. We fix the tail length by choosing $X_M = 3$. This value of X_M, however, is not big enough to prevent the asymmetry of the helix to affect F_{prop}, $L(\gamma_{eq})$ or $K(\gamma_{eq})$. Since the asymmetry is averaged over a tail rotation period, we take the projection of F^f_{prop} along the helix axis before applying the rotation matrix R (cf. figures I.3.2-I.3.4 and the following subsection). To produce these figures we fixed the tails' latitude angle $\alpha = 0.4\pi$.

The value of the elasticity constant k will be varied to see the effects it has on the equilibrium angle γ_{eq} and on the propulsion velocity U_x.

I.3.1.4 Validation of the RFT model

The RFT model was validated against analytical results coming from the literature. We consider a cylindrical helix of pitch angle θ, where $\tan\theta = 2\pi R^t/\lambda$, and we compute the fluid drag K, the fluid torque L and the propulsion force F_{prop} for different values of the length-wavelength ratio L/λ and the wavelength-radius ratio λ/R^t. Our results are compared to [111], who computed F_{prop}, L, K for the cylindrical helix

$$\begin{aligned} F_{prop} &= (\Omega R^t)(C_n - c_t)\sin\theta\cos\theta\frac{L}{\cos\theta}, \\ \mathrm{L} &= \left(\Omega (R^t)^3\right)(C_n\cos^2\theta + C_t\sin^2\theta)\frac{L}{\cos\theta}, \\ K &= U(C_n\sin^2\theta + C_t\cos^2\theta)\frac{L}{\cos\theta}. \end{aligned} \tag{I.3.11}$$

In figures I.3.1 we report the results of our benchmark for different values of the ratio λ/R. The reference values for the fluid drag, torque and propulsion force, computed with equation (I.3.11), are shown in yellow, while the values we computed are shown in red. We remark that the two curves coincide, as the relative difference between our results and the benchmark is smaller than $1e-10$. In blue we report the same coefficients, computed in the case of the helix of our interest. Similar results were obtained when computing the RFT coefficients for different values of the ratio L/λ, and for such reason those figures were not included.

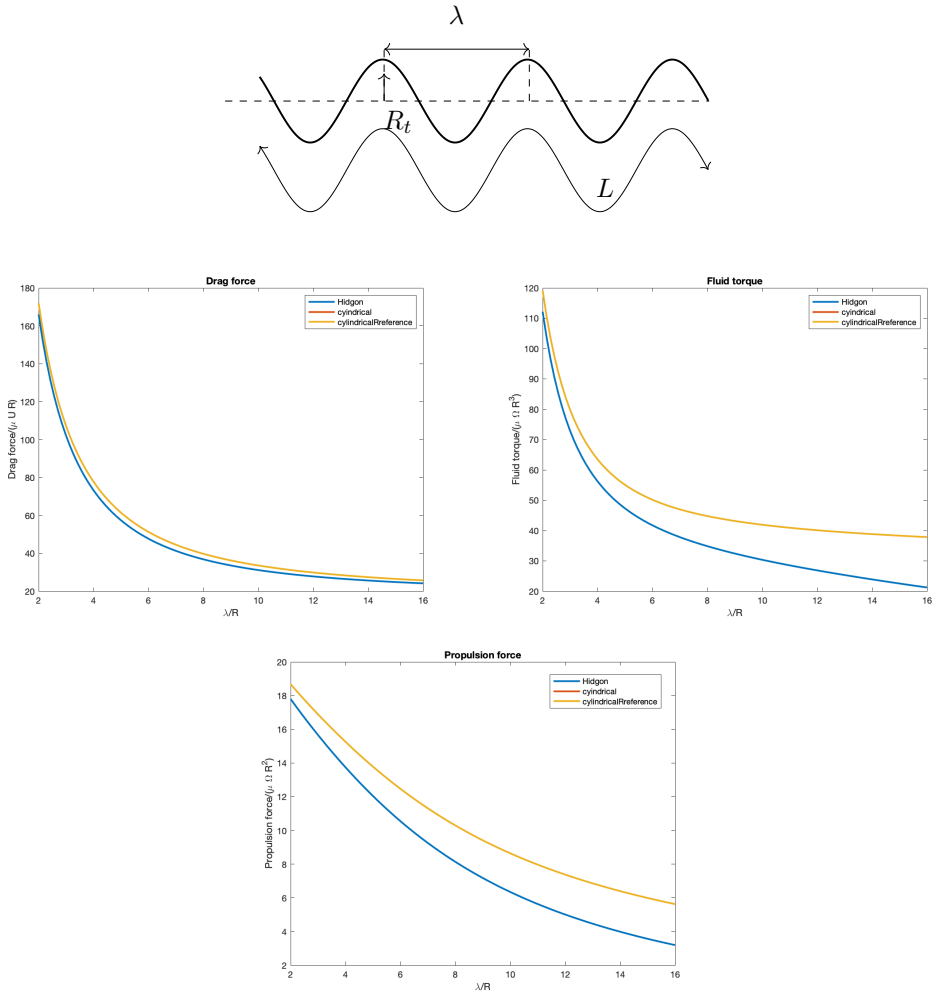


Figure I.3.1: The top figure gives a schematic representation of the cylindrical helix and its parameters. The bottom figures present the comparison between the RFT coefficients for a cylindrical helix as described in (I.3.11) and the ones computed by our code (in yellow and red, superposed), matching with a relative tolerance of 10^{-10}. In blue, we present the curves for the helix of our interest, described by equations (I.2.2).

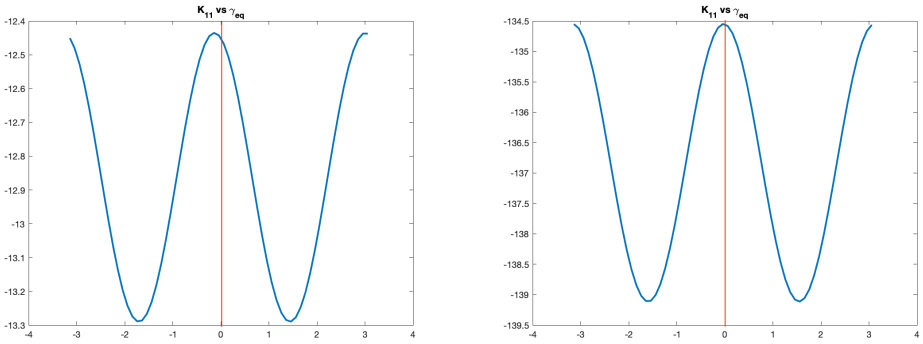


Figure I.3.2: Comparison of $K(\gamma_{eq})$ for $X_M = 3$ (left) and $X_M = 30$ (right). We can see the influence of finite size effects on $K(\gamma_{eq})$ from the asymmetry of the graph with respect to $\gamma_{eq} = 0$ axis.

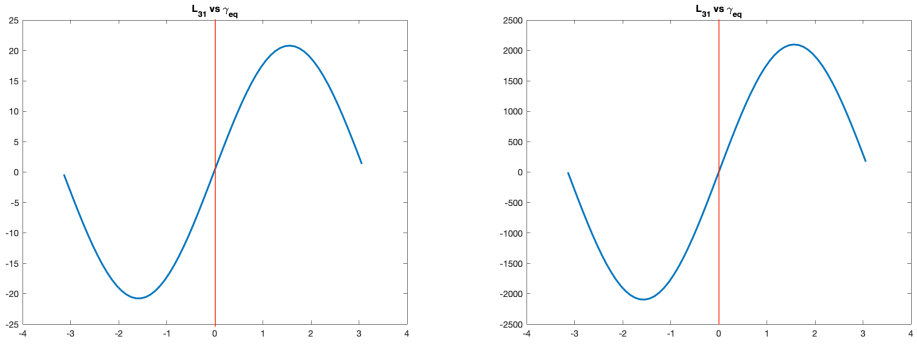


Figure I.3.3: Comparison of $L(\gamma_{eq})$ for $X_M = 3$ (left) and $X_M = 30$ (right). We can see the influence of finite size effects on $L(\gamma_{eq})$ from the asymmetry of the graph with respect to the origin.

I.3.1.5 Results of the model

In figures I.3.2-I.3.4 we see the effect of finite size that a shorter and longer helix have on the quantities K, L and F_{prop}. This effect is reduced if a long tail is considered, because the terminal portions of the tail are proportionally less important. This is shown by looking at the symmetry of components K_{11}, L_{31} and $F_{prop,1}$ when the helix axis is aligned with $[\cos(\gamma_{eq}), \sin(\gamma_{eq})]$ for positive γ_{eq} or negative γ_{eq}. The ideal situation, in absence of finite size effects, would be a perfect even/odd symmetry (depending on the coefficients). When considering the propulsion force, we can mitigate these finite size effects by considering the average force over a period, directed in the direction of the tail's axis, that is we directly take $F^f_{prop} = [F^f_{prop,1}\, 0\, 0]$ in the tail's frame.

The equilibrium angle γ_{eq} and the propulsion speed vary with the elasticity constant, the head shape and the tails' attachment angle α. In figures I.3.5-I.3.6 we show all these variations at once, fixing the elasticity constant. The top figures in I.3.5-I.3.6 show the reference angles $\bar{\gamma}$ in black (they vary because the normal varies with the attachment point and with the head shape) and the corresponding coloured γ_{eq}, as the attachment angles α vary from 0.1 to 1.1472, as a

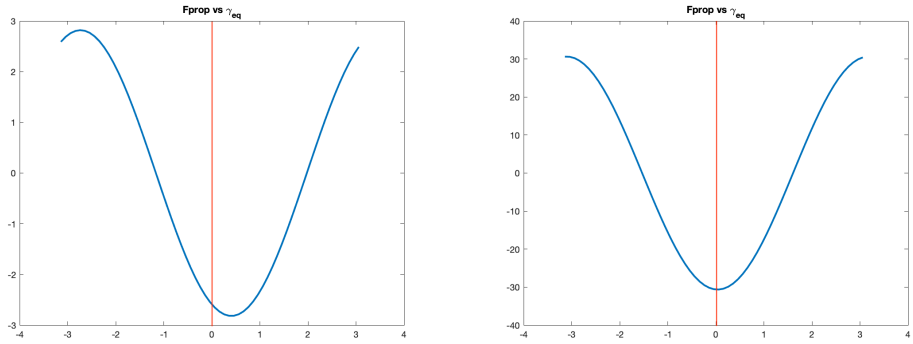


Figure I.3.4: Comparison of F_{prop} for $X_M = 3$ (left) and $X_M = 30$ (right). As before, We can see the influence of finite size effects on F_{prop} from the asymmetry of the graph with respect to $\gamma_{eq} = 0$ axis.

function of the head eccentricity (in the abscissa). The bottom figures display the propulsion velocity as the attachment angles α vary, as a function of the head eccentricity. In figure I.3.5 the elasticity constant is $k = 100$ and in figure I.3.6 the elasticity constant is $k = 10$. It is possible to see that lower values of the elasticity constant allow for larger deviations from the equilibrium position of the tails, as expected, since the elastic recovery momentum $G(k, \gamma_{eq} - \bar{\gamma})$ in equation (I.3.5) is weaker. This encourages the alignment of tails with the propulsion direction, which improves the propulsion speed for prolate heads, as we can see by comparing the two red curves in the bottom figures of I.3.5-I.3.6. It is possible to see that, when tails are closer to the equator of the spheroid, the propulsion velocity is higher for prolate head shapes (positive x). On the other hand, when tails are located far from the equator, larger propulsion speeds are experienced for oblate heads.

I.3.2 The BEM model

The swimmer being considered has an ellipsoidal head and two helical tails. The tails junction points are located symmetrically over the cell body, at the latitude provided by the angle α and with an inclination with respect to the horizontal given by the angle γ. We indicate as $S \subset \mathbb{R}^3$ the region occupied by the swimmer, and by ∂S its surface. H and F_i $(i = 1, 2)$ will be used to indicate the cell body and the tails, respectively; x^{CM} will denote the center of mass of the cell body and x_{F_i} the junction point between the $i-$th flagellum and cell. We model the elasticity effects at the head-tail junctions using torque springs, which are defined by their elasticity constant k. A non-zero elastic torque will induce a rotation at the junction, which will be described by two angular velocities ω_i around the e_2 and e_3 axes of the flagellum's local reference frame. A more general expression of the elastic torque can be given by

$$M_{elast} = -k \frac{\tilde{e}_3^{eq} \times \tilde{e}_3}{||\tilde{e}_3^{eq} \times \tilde{e}_3||} \tan(\arcsin(\frac{||\tilde{e}_3^{eq} \times \tilde{e}_3)||}{||\tilde{e}_3^{eq}||\,||\tilde{e}_3||}), \tag{I.3.12}$$

where $\tilde{e}_3$ is the axis vector of the tail in its rest position (seen in the lab frame) and $\tilde{e}_3^{eq}$ is the axis vector of the tail in its equilibrium position (seen in the lab frame).

As before, we employ the single layer formulation for the Stokes flow field (the constant prefactor

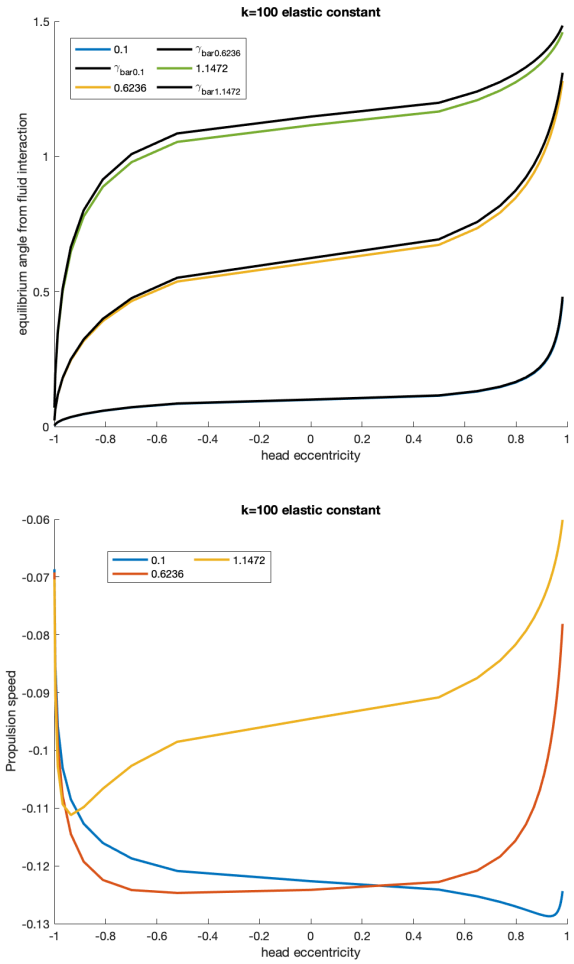


Figure I.3.5: Equilibrium angle (top) and propulsion speed (bottom) as a function of the head eccentricity. The different curves are parametrized by the angle α which corresponds to the latitude of the tails on the spheroid. In this case, the elasticity constant is equal to 100 and the elastic recovery momentum maintains the tails close to their initial position.

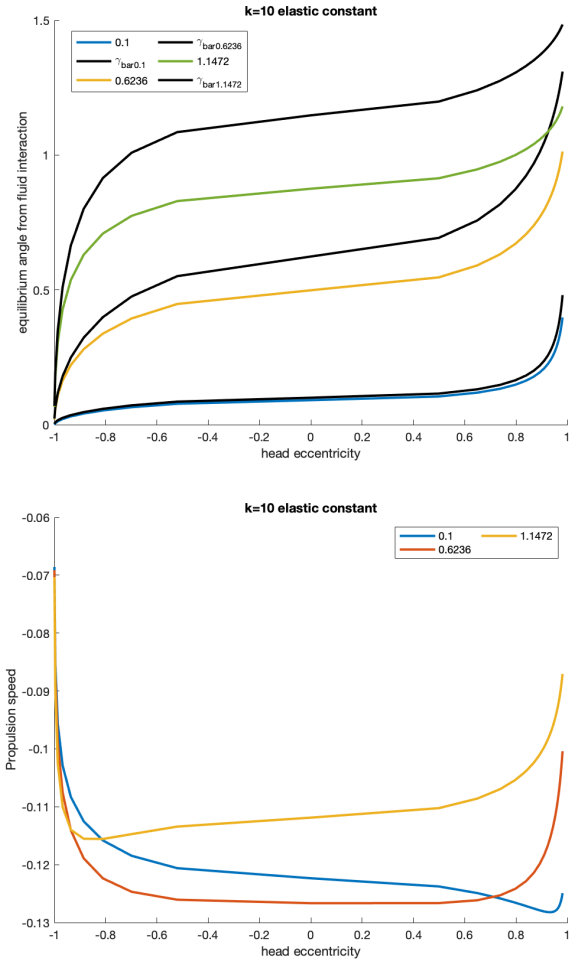


Figure I.3.6: Equilibrium angle (top) and propulsion speed (bottom) as a function of the head eccentricity. The different curves are parametrized by the angle α. In this case, the elasticity constant is equal to 10 and the elastic recovery momentum is weaker than the previous case. This encourages the alignment of tails with the propulsion direction, which improves the propulsion speed for prolate heads.

is inside $\mathbf{G}$),

$$u(x) = -\int_{\partial S} \mathbf{G}(x,y)\mathbf{f}(y)\, dS(y), \tag{I.3.13}$$

and complement it with self propulsion constraints

$$\begin{aligned} \int_{\partial S} f\, dS = 0, \\ \int_{\partial S} (x - x^{CM}) \times f\, dS = 0. \end{aligned} \tag{I.3.14}$$

The velocity field on the surface of the swimmer has the following form

$$u(\mathbf{x}) = \begin{cases} U + \omega \times (x - x^{CM}), & x \in \partial H, \\ U + \omega \times (x - x^{CM}) + \mathbf{B}\omega_1 \times (x - x_{F_1}) + \mathbf{B}\mathbf{\Omega_{F_1}} \times (x - x_{F_1}), & x \in \partial F_1, \\ U + \omega \times (x - x^{CM}) + \mathbf{B}\omega_\mathbf{2} \times (x - x_{F_2}) + \mathbf{B}\mathbf{\Omega_{F_2}} \times (x - x_{F_2}), & x \in \partial F_2, \end{cases} \tag{I.3.15}$$

where $\mathbf{\Omega_{F_i}}$ is the imposed and known axial rotation of the flagellum, $\omega_\mathbf{i}$ is the angular velocity coming from the torque spring and $\mathbf{B}$ is the matrix expressing the change in coordinates from the body frame to the laboratory frame. $\omega_\mathbf{i}$ and $\mathbf{\Omega_{F_i}}$ are expressed in the cell body frame.

In order to recover $\omega_\mathbf{1}$ and $\omega_\mathbf{2}$, we complement the previous system with the following equations

$$\begin{aligned} \int_{\partial F_1} (x - x_{F_1}) \times f = -\mathbf{M}^{F_1}_{elast}, \\ \int_{\partial F_2} (x - x_{F_2}) \times f = -\mathbf{M}^{F_2}_{elast}, \end{aligned} \tag{I.3.16}$$

where $\mathbf{M}^{F_i}_{elast} = -k\tan(\phi)\mathbf{e}_\perp$, k is the elasticity constant, $\phi = \arcsin(|\mathbf{e}'_z \times \mathbf{e}''_z|)$ and $\mathbf{e}_\perp = (\mathbf{e}'_z \times \mathbf{e}''_z)/|\mathbf{e}'_z \times \mathbf{e}''_z|$, where $\mathbf{e}''_z$ is the axis vector of the tail in its rest position (seen in the lab frame) and $\mathbf{e}'_z$ s the the axis vector of the tail in its equilibrium position (seen in the lab frame). The elastic momentum has the same form as in (I.3.12), but for the sake of benchmark, we consider also an elastic momentum which is linear with respect to ϕ: $\mathbf{M}^{F_i}_{elast} = -k\phi\mathbf{e}_\perp$. Equations (I.3.16) impose the torque balance at the head-tail junctions by equalling the elastic and fluid momenta, with respect to the junction x_{F_i}. By expanding each component of f over the finite element basis $\{\phi^l\}_{l=1}^N$ defined over the flagella, we can write $f_j(y) = \sum_{l=1}^N f_j^l \phi^l(y)$, analogously as we did for the self-propulsion constraint on the torque. We can then write the variational formulation

$$\begin{aligned} \int_{\partial F_1} (x - x_{F_1}) \times \sum_{l=1}^N [f_1^l, f_2^l, f_3^l]^T \phi^l(x) = -\mathbf{M}^{F_1}_{elast}, \\ \int_{\partial F_2} (x - x_{F_2}) \times \sum_{l=1}^N [f_1^l, f_2^l, f_3^l]^T \phi^l(x) = -\mathbf{M}^{F_2}_{elast}. \end{aligned} \tag{I.3.17}$$

I.3.2.1 Validation of the elasticity model at the junctions

The implementation of the torsion spring in the BEM model is validated by considering the problem of tail relaxation, from a position where the elastic torque is non-zero to the rest position where the elastic torque is zero, and subsequent stillness of the flagellum around that position. We performed this check for different ellipsoidal shapes, and in all cases the relaxation

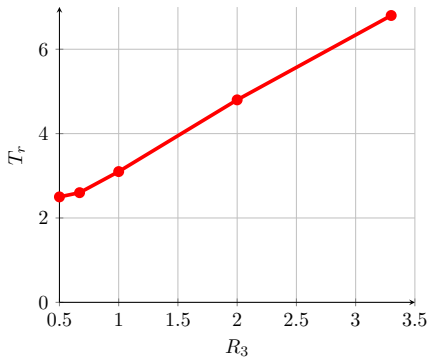
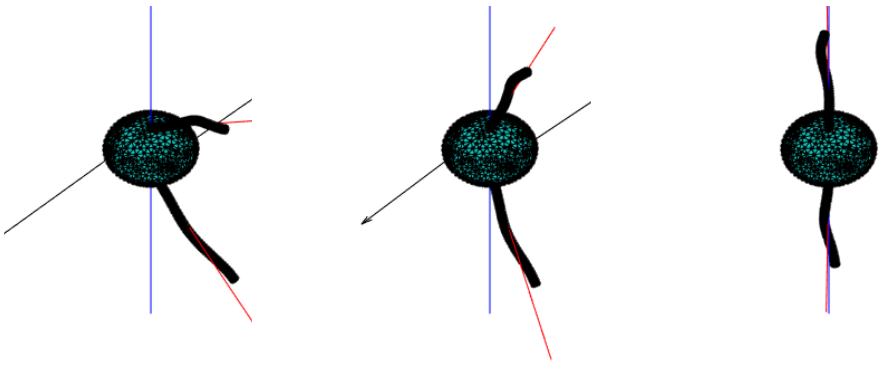


Figure I.3.7: Relaxation times for two tails whose junctions are located at the extremities $(0, 0, \pm R_3)$ of an ellipsoidal body of radii $R_1 = R_2 = 1$ and R_3. Smaller values of R_3 correspond to smaller and flatter spheroids.

position was attained. The relaxation time depended on the ellipsoid's shape, as shown in figure I.3.7. Figure I.3.8 shows the relaxation of the tails towards the vertical position, denoted in our case by a blue vertical vector.



(a) $t = 0.3$ (b) $t = 1.4$ (c) $t = 2.7$

Figure I.3.8: Relaxation of the tails at the junctions. The blue lines correspond to the relaxed position of the tails, the red lines to the current position and the black lines with the direction of the elastic moment. As the time passes, the black vectors shrink and the red vector aligns with the vertical blue one.

Chapter I.4

Software development

The results and simulations presented in this part have been realised in Matlab. Part of the simulation framework is based on the Matlab BEM library *Gypsilab*, namely domain discretisation, matrix assembly and numerical integration. Other parts are deferred to Matlab's own routines, like the optimisation algorithms and algebraic solution of the BEM system. The rest has been implemented using as much as possible Matlab classes and tested via Matlab Unit testing.

The rigid-body solver, based on quaternions, is programmed as a class which calls a module of functions. The module contains few functions implementing fundamental quaternion operations, like the product of two quaternions or the transformation of the unit quaternion into a rotation matrix. The class contains the *properties* which describe the position of the rigid body in space and the *methods* that are required to solve the system of rigid-body ODEs. The testcase presented in I.1.1.6 is implemented using the Unit Testcase framework of Matlab.

The scripts and functions implementing the convergence tests presented in I.1.1.3 are contained in the *Gypsilab* GitHub repository[1].

The RFT simulations are based on a class that computes the drag force, torque and propulsion force given the helical tail. The choice of the cylindrical helix allowed us to benchmark our results, as shown in the previous sections.

The BEM simulations are also based on few Matlab classes. The geometry of the swimmer is encoded in three classes: two handle the mesh generation of head and tail, while the third adds further information on the global geometry of the swimmer (like the angles α and γ) while inheriting the methods and properties of the first two. This class also contains a method that detects intersections between tails or between one tail and the head. Another class is responsible of the boundary element simulation during the tail revolution period around its axis: this class derives the geometry from the swimmer's geometry class and implements the construction of the Stokes problem using BEM over the boundary of the swimmer when no elasticity at the junction is considered. BEM simulations using elastic junctions are treated by another class, which builds on the previous BEM class and adds the terms including the elastic momentum and the torque springs at the junctions. Tests were run on these classes, and their results have been shown along this chapter.

[1] https://github.com/matthieuaussal/gypsilab/nonRegressionTest/stokes/translatingSphere

Part II

Finite element framework for micro-swimming simulation and optimisation

This part presents the mathematical and numerical modelling of the swimming problem in differential form, where the numerics is based on a conforming Arbitrary-Lagrangian-Eulerian fluid-swimmer finite element discretization. Chapter II.1 presents the mathematical modelling of the problem, focusing on different swimmer models and the Arbitrary-Lagrangian-Eulerian description of the fluid problem. Chapter II.2 contains the various aspects of the discrete problem: spatial and temporal discretisations are addressed, as well as preconditioning and mesh adaptation. Chapter II.3 validates the methods and models presented in the previous two chapters through testcases on mesh adaptation and fluid-body simulations.

Chapter II.1

Modelling

In this chapter we focus on the mathematical modelling of the fluid-structure problem. In section II.1.1 we present the different swimmers that we are able to model and simulate using the framework, and we focus on their deformation strategy and gait. The distinctions we made are also based on the mathematical and/or computational form of the swimming stroke. In section II.1.2 we address the formulation of the moving continua using the Arbitrary-Lagrangian-Eulerian formalism, which will then be used to express the fluid problem in moving domain; in section II.1.3 we present the fluid model we consider and show how we modelled the motion of an immersed rigid body; in section II.1.4 we present the coupling conditions that describe the interaction of the fluid and swimmers for the different swimmer models we discussed at the beginning of the chapter. The formulation is presented in terms of Navier-Stokes equations, to be as general as possible, hence section II.1.5 discusses the usage of Stokes equations and how the formulation changes with this model.

II.1.1 Swimmer models

The variety of mathematical and computational descriptions of swimmers' gaits led us to consider different swimmer models in order to include as many cases as possible in our micro-swimming framework. We grouped swimmer gaits and models into three categories. The first category is constituted by non-deformable swimmers or by collection of rigid bodies. As these swimmers are not able to deform their components, they move either by exploiting external actuations or by relative motion of their parts. The second category of models contains deformable swimmers whose body deformation is prescribed and known in advance. Here, the swimming gait can be modelled via an analytical formula or it can come from observations. The third category comprises elastic swimmers modelled by elasticity equations. In our case, non-linear elasticity is chosen as the hypothesis of small displacement gradients is generally not satisfied by the swimmers we aim to model, as their bodies bend significantly. In this last case, we consider also models of "active" elasticity, that allow to model bodies that can self-deform exploiting internal actuation mechanisms.

Let us denote by $\mathcal{S} \subset \mathbb{R}^d$ the region occupied by the swimmer and by $u_d(t, X) : [0, T] \times \partial\mathcal{S} \to \mathbb{R}^d$ the deformation velocity at the boundary of the swimmer encoding the swimming gait, defined on the swimmer's initial configuration. Let $X \in \mathcal{S}$ be the variable in the reference configuration, $\eta : \mathcal{S} \to \mathbb{R}^d$ the swimmer's displacement and $x \in (\mathbb{I}+\eta)(\mathcal{S})$ the variable in the current configuration of the swimmer's domain. The deformation velocity will have different forms depending on the

swimmer under consideration, and it will be part of the kinematic boundary condition at the fluid-swimmer interface. As we will see later, on $\partial\mathcal{S}$, one requires that the restriction of the fluid velocity $u(t,x)\Big|_{\partial\mathcal{S}}$ and the swimmer's velocity coincide. This latter can be split as the sum of two contributions: a rigid-body contribution $U + \omega \times (x - x^{CM})$, where U and ω are the translational and angular velocity of the swimmer and $x^{CM} = (\mathbb{I} + \eta)(X^{CM})$ its center of mass, and u_d, which denotes the deformation velocity of the object, that is the speed at which its shape changes or equivalently the velocity at which it's able to push the fluid at its boundary. Let $\rho_s(X)$ be the density of the swimmer: its mass m, center of mass X^{CM} and inertia matrix J are computed as

$$m = \int_{\mathcal{S}} \rho_s(X)\,\mathrm{d}X, \quad X^{CM} = \frac{1}{m}\int_{\mathcal{S}} X\rho_s(X)\,\mathrm{d}X, \quad J = \int_{\mathcal{S}} \rho_s(X)(X - X^{CM}) \otimes (X - X^{CM})\,\mathrm{d}X.$$

These quantities are defined in the swimmer's reference frame, as they are computed in the reference configuration $\mathcal{S}$ of the swimmer's body. As it will be shown in section II.1.3, the center of mass and the moment of inertia of the swimmer will evolve as the swimmer's local reference frame moves with respect to the laboratory frame, in which the fluid equations are cast. Also the expression of the deformation velocity u_d will be influenced by the change of reference frame necessary to express the swimmer's variables in the fluid's reference frame.

II.1.1.1 Rigid swimmers

A non-deformable micro-swimmer can be used to approximate ciliated micro-organisms via the "squirmer" model [37], or to simulate swimmers like helical micro-robots [129] or phoretic particles [89]. In the squirmer case, the swimming gait of the organism can be encoded in the function u_d by prescribing a velocity field tangent to $\partial\mathcal{S}_0$, modelling the cilia waving pattern on the surface of the body. For a circular squirmer moving in the direction $\vec{d}$, the magnitude of the surface velocity $|u_d| \propto \sin(\angle(\vec{d}, x - x^{CM}))$ depends on the swimming direction $\vec{d}$. For helical micro-robots and phoretic particles, $u_d = 0$ as those swimmers do not deform. However, their motion is coming from chemical or magnetic forces and torques that influence directly their translational and angular velocities U and ω.

When considering several rigid bodies, capable of moving relatively to each other, as in the case of multi-sphere swimmers [91, 6], one can prescribe a non-zero u_d to impose the relative velocity between these bodies. This possibility will be shown in chapter IV.1, where the three-sphere swimmer stroke proposed in [91] will be reformulated as a relative velocity prescription. Figure II.1.1 illustrates this example where the left and right arms are activated by the function of corresponding colour.

II.1.1.2 Swimmers with prescribed deformation

In the case of deformable swimmers, the function u_d denotes the deformation velocity of the body. Depending on the case under consideration u_d can be available in different forms: analytical [109], from reduced models based on ODEs [62] or from images [113]. Particular attention should be paid to the body configuration in which the analytical and reduced models are evaluated. Normally, they are evaluated on the initial configuration of the swimmer: for example, in the case of a sperm cell, often these quantities are computed considering the tail to be in horizontal position, as in

$$u_d(t,X) = \begin{bmatrix} A\cos(4\pi(t - X)) \\ B\cos(2\pi(t - X)) \end{bmatrix}, \tag{II.1.1}$$

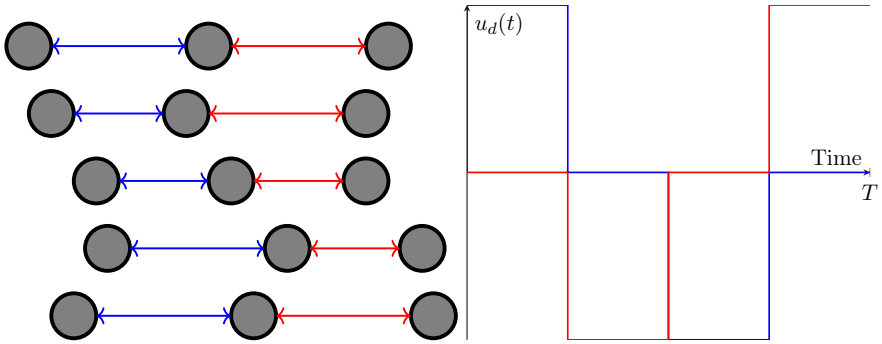


Figure II.1.1: The three sphere swimmer and its swimming gait are presented, as proposed by [91]. Next, the functional form of the relative velocity function coming from this gait is presented over one period T.

a simplified version of the deformation velocity employed in [109]. Another note to be considered is that, for reduced models, the displacement being computed might concern just a subset of the body, or lower dimensional approximation of it, for example the centerline of the tail as in

$$\begin{cases} \dfrac{d\mathbf{r}}{dl} = -\mathbf{e}_3, \\ \dfrac{d\mathbf{e}_1}{dl} = -\kappa_f \mathbf{e}_3 + \tau_f \mathbf{e}_2, \\ \dfrac{d\mathbf{e}_2}{dl} = -\tau_f \mathbf{e}_1, \\ \dfrac{d\mathbf{e}_3}{dl} = \kappa_f \mathbf{e}_1. \end{cases} \tag{II.1.2}$$

where $\mathbf{r}$ is the position vector of the centerline, $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ give the orientation of the local reference frame and $l \in [0, L]$. In this case, one should also devise a way to extend the displacement to the whole region occupied by the swimmer's body. In chapter IV.2.1 an example of this has been addressed.

II.1.1.3 Elastic swimmers

The last category of swimmers we considered is the category of elastic swimmers, that collects swimmers whose deformation is not known a priori or simplified with reduced models, but arises from the solution of elasticity and the fluid-swimmer interaction. In this case we differentiate among two elasticity models: passive elasticity is used to describe swimmers that deform as a result of external forces and torques, while active elasticity is used to model swimmers who can deform as a result of internal effects. The first case comprises flexible passive micro-robots, while the second one comprises flagella whose deformation is caused by internal molecular motors, or more generally muscular action for larger swimmers.

Passive elasticity

Let $F = \mathbb{I} + \nabla_X \eta$ be the spatial gradient of the position vector $x(t, X)$, encoding the geometric variations of the deformable body. Then, let $E = \frac{1}{2}(F^T F - \mathbb{I}) = \frac{1}{2}(\nabla_X \eta + \nabla_X \eta^T + \nabla_X \eta^T \nabla_X \eta)$ be the strain tensor, which depends non-linearly on the displacement η. Finally, let us define Σ

as the second Piola-Kirchhoff tensor, whose expression depends on the constitutive model of the elastic body and E. In this thesis we restrict to hyper-elastic materials, for which there exists an energy function $\mathcal{E}$ such that $\Sigma = \frac{\partial \mathcal{E}}{\partial E}$. One peculiarity of hyper-elastic materials is that the work done by the structure, when passing from one configuration to the other, depends only on the initial and final state, and not on the particular sequence of deformations. The easiest model one can consider is the Saint-Venant-Kirchhoff model, for which

$$\mathcal{E} = \frac{\lambda}{2} tr(E)^2 + \mu E : E, \qquad \Sigma = \frac{\partial \mathcal{E}}{\partial E} = \lambda tr(E)\mathbb{I} + 2\mu E, \tag{II.1.3}$$

where

$$\lambda = \frac{e\nu}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{e}{2(1+\nu)}$$

are the Lamé coefficients depending on the Young modulus e and Poisson ratio ν of the solid. The displacement $\eta :]0,T] \times \mathcal{S}_0 \to \mathbb{R}^d$ is the solution of the hyper-elasticity equations

$$\rho_s \frac{\partial^2 \eta}{\partial t^2} - \nabla \cdot (F\Sigma) = f \quad \text{on } \mathcal{S}_0 \times]0,T], \tag{II.1.4a}$$

$$\eta(0,X) = \bar{\eta}(X) \quad \text{on } \mathcal{S}_0 \times \{0\}, \tag{II.1.4b}$$

$$F\Sigma\vec{n} = g(t,X) \quad \text{on } \partial\mathcal{S}_N \times [0,T], \tag{II.1.4c}$$

$$\eta(t,X) = h(t,X) \quad \text{on } \partial\mathcal{S}_D \times [0,T], \tag{II.1.4d}$$

where ρ_s is the density of the solid and f is the density of external forces. In this formulation the boundary was divided in two disjoint subsets $\partial\mathcal{S}_N, \partial\mathcal{S}_D$ where the Neumann and Dirichlet boundary conditions are imposed via functions $g(t,X)$ and $h(t,X)$ respectively. The term $\nabla \cdot (F\Sigma)$ models the internal forces that arise from the body's deformations, while $\rho_s \frac{\partial^2 \eta}{\partial t^2}$ denotes the momentum contributions associated to the acceleration of the body.

If inertial effects are neglected, which is the case at low Reynolds number, the acceleration term can be dropped and equations II.1.4a reduce to

$$-\nabla \cdot (F\Sigma) = f, \quad \text{on } \mathcal{S}_0. \tag{II.1.5}$$

In order to later address the finite element discretization, let us derive the variational formulation of the non-linear elasticity equations. The variational problem is finding $\eta(t,\cdot) \in [H^1(\mathcal{S}_0)]^d$ such that, for all test functions $\phi \in [H^1(\mathcal{S}_0)]^d$,

$$\int_{\mathcal{S}_0} \rho_s \frac{\partial^2 \eta}{\partial t^2} \cdot \phi \, \mathrm{d}X + \int_{\mathcal{S}_0} F\Sigma : \nabla\phi \, \mathrm{d}X = \int_{\mathcal{S}_0} f \cdot \phi \, \mathrm{d}X + \int_{\partial\mathcal{S}_N} F\Sigma\vec{n} \cdot \phi \, \mathrm{d}S, \tag{II.1.6}$$

and $\eta(t,X) = h(t,X)$ on $\partial\mathcal{S}_D$. One way to solve the non-linear variational problem is using Newton method coupled with finite elements. We also derive the linearisation of the product $F\Sigma$ for a Saint Venant-Kirchhoff material to use it later in the Newton algorithm. Let $t > 0$ and h be the increment. The linearization of $F\Sigma$ at η is given by

$$\frac{d}{dt}(F\Sigma)(\eta + th)|_{t=0} = \frac{d}{dt}(F)(\eta + th)|_{t=0}\Sigma(\eta) + F(\eta)\frac{d}{dt}(\Sigma)(\eta + th)|_{t=0}, \tag{II.1.7}$$

where

$$\frac{d}{dt}(F)(\eta + th)|_{t=0} = \nabla h, \qquad \frac{d}{dt}(\Sigma)(\eta + th)|_{t=0} = \lambda tr(\frac{d}{dt}E|_{t=0})\mathbb{I} + 2\mu\frac{d}{dt}E|_{t=0}, \tag{II.1.8}$$

and

$$\frac{d}{dt}E|_{t=0} = \frac{1}{2}(\nabla h + \nabla h^T) + \frac{1}{2}(\nabla\eta^T\nabla h + \nabla h^T\nabla\eta). \tag{II.1.9}$$

Active elasticity

Active elasticity models are built on passive elasticity equations by adding activity functions or decomposing elastic effects in their active and passive components. For this reason, we use the same notations that we used in the previous paragraph on passive elasticity. One can find two models of active hyper-elasticity in the literature, namely active stress and active strain models [94]. The active stress model is based on an additive decomposition of the Piola-Kirchhoff stress tensor that separates the passive and active effects

$$\begin{aligned} \rho_s \frac{\partial^2 \eta}{\partial t^2} - \nabla \cdot (F\Sigma - F\Sigma_a) &= f, && \text{on } \mathcal{S}_0 \times [0,T], \\ \eta(0,X) &= \bar{\eta}(X) && \text{on } \mathcal{S}_0 \times \{0\} \\ F\Sigma\vec{n} - F\Sigma_a\vec{n} &= g(t,X) && \text{on } \partial\mathcal{S}_N \times [0,T], \\ \eta(t,X) &= h(t,X) && \text{on } \partial\mathcal{S}_D \times [0,T]. \end{aligned} \tag{II.1.10}$$

The equations hence solve the problem $\rho_s \partial_{tt}\eta - \nabla \cdot (F\Sigma_p) = f$ where $\Sigma_p = \Sigma - \Sigma_a$ is the passive component of the total stress of the body. A model for Σ_a corresponding to a travelling wave along a two dimensional elastic body is given by $\Sigma_a = [A \sin(2\pi f(X - v_a t))(Y_c - Y)]e_a \otimes e_a$ [123]. In the previous formula, e_a is the tangent unit vector to the muscular fibres, A is the amplitude of the deformation, f its frequency, v_a is the speed of the travelling wave, X and Y are the coordinates in the swimmer's reference frame, with X corresponding to the direction of propagation of the wave, and Y_c is the center-line coordinate.

The second model is based on a multiplicative decomposition of the deformation gradient $F = F_p F_a$ into a passive and active part and the contraction of the body is ensured by prescribing F_a [10]. Following what was done by [26, 30], we also consider this second approach. As before, since the elasticity equations only describe the passive effects of the deformation, they are expressed as a function of $\Sigma_p = \Sigma(F_p) = \lambda tr(E_p)\mathbb{I} + 2\mu E_p$.

$$\begin{aligned} \rho_s \frac{\partial^2 \eta}{\partial t^2} - \nabla \cdot (|\det(F_a)| F F_a^{-1} [\Sigma(F) - \Sigma(F_a)] F_a^{-T}) &= 0 && \text{on } \mathcal{S}_0 \times [0,T], \\ \eta(0,X) &= \bar{\eta}(X) && \text{on } \mathcal{S}_0 \times \{0\} \\ |\det(F_a)| F F_a^{-1} [\Sigma(F) - \Sigma(F_a)] F_a^{-T} \vec{n} &= g(t,X), && \text{on } \partial\mathcal{S}_N \times [0,T], \\ \eta(t,X) &= h(t,X) && \text{on } \partial\mathcal{S}_D \times [0,T]. \end{aligned} \tag{II.1.11}$$

Differently from the first model, it is now the *visible* strain tensor $E = \frac{1}{2}(F^T F - \mathbb{I})$ that can be decomposed as the sum of a time-varying distortion strain E_a, depending on the imposed muscular deformations, and of E_p, the *passive* part of elastic strain. In [30] the *active* strain E_a is prescribed as a function of the imposed transversal displacement of the swimmer's axis $d(X,t)$, for example

$$E_{a,XX}(X,Y,t) = -Y \frac{\partial^2 d(X,t)}{\partial X^2}, \quad E_{a,XY} = E_{a,YX} = E_{a,YY} = 0. \tag{II.1.12}$$

Since F_a is used in (II.1.11), one should recover the value of F_a from the imposition of E_a, by solving the normal equations $F_a^T F_a = 2E_a + \mathbb{I} = f(E_a)$. These equations can be solved by decomposing $f(E_a) = Q^T D Q$, where D is diagonal with positive entries and Q is orthogonal, which gives $F_a = \sqrt{D}Q$. The linearisation of the active elasticity models for the Newton method are done exactly as in the passive case, by linearising the product $F\Sigma$.

The wellposedness of the fluid-structure problem involving an active elastic body, the existence and uniqueness of the solution is studied in [123] in the active stress case. The active strain

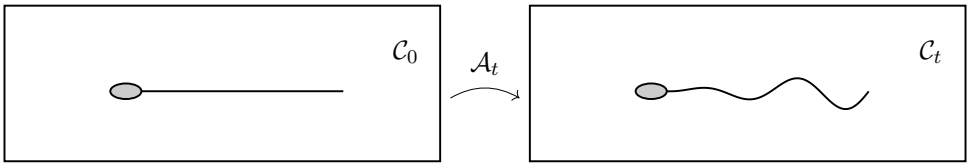


Figure II.1.2: Illustration of the evolution of the fluid computational domain $\mathcal{C}_t$ as described by the Arbitrary-Lagrangian-Eulerian map.

case was studied by [90], who considered the electromechanical coupling of the active strain model with the activation potential, which is responsible for cardiac contraction, and established the existence of weak solutions to a linearised version of the system. To our knowledge, the mathematical analysis of the fluid-structure problem where the active structure is modelled via the active strain approach has not been addressed yet.

II.1.2 Arbitrary-Lagrangian-Eulerian description of continua

The swimmers we consider in this thesis propel by interacting with the surrounding fluid medium and deforming their bodies. Hence, the mathematical formulation of the fluid problem is cast in a time-dependent domain. Consider $\mathcal{F}_0 \subseteq \mathbb{R}^d$, for $d = 2, 3$ and $\varphi_t : \mathcal{F}_0 \to \mathcal{F}_t$ a parametrized family of smooth bijective functions describing the motion of the fluid continuum. Here, for the sake of simplicity, the initial domain $\mathcal{F}_0$ will coincide with the reference and material domains, but we will not always assume this in the numerical cases we will treat. Navier-Stokes and Stokes equations are classically written in the Eulerian reference frame, meaning that they describe the evolution of the fluid variables for each "geometric" point in a fixed computational domain $\mathcal{C}$. This description is opposed to the Lagrangian formalism, which describes the evolution of the continuum by following the trajectories of each particle, typical of solid mechanics. As this second approach is classically employed in solid mechanics, it is necessary to establish a common description of the fluid and solid equations in order to formulate the fluid-solid interaction problem. Among the possible choices, we consider the Arbitrary-Lagrangian-Eulerian (ALE) formalism, which partially decouples the evolution of the computational domain from the evolution of the underlying continua.

Let $\mathcal{C}_t$ be the computational domain where the fluid equations are solved at time t. We define the ALE maps $\mathcal{A}_t : \mathcal{C}_0 \to \mathcal{C}_t$ as the one-parameter family of smooth bijective functions that describe the evolution of the computational domain (see Figure II.1.2 for an example). In our case, we consider $\mathcal{C} \equiv \mathcal{C}_0 \equiv \mathcal{F}_0$, but in general we have $\mathcal{C}_t \neq \mathcal{F}_t$ for $t > 0$. The time evolution of the boundary of the computational domain $\partial\mathcal{C}_t$, coincides with that of $\partial\mathcal{F}_t$, but it is extended to the interior of $\mathcal{C}_t$ via an arbitrary function that does not depend on the fluid evolution. Let $\bar{\phi}_t : \partial\mathcal{C}_0 \to \partial\mathcal{C}_t$ the function describing the time evolution of the boundary of the computational (and fluid) domain. The extension method we will be using throughout the thesis is based on the Laplace (or harmonic) smoothing and consists in solving the elliptic equation

$$\begin{cases} \nabla \cdot ((1 + \tau(X))\nabla\phi_t(X)) = 0, & \text{on } \mathcal{C}_0, \\ \phi_t = \bar{\phi}_t, & \text{on } \partial\mathcal{C}_0, \end{cases} \tag{II.1.13}$$

where $\tau(X)$ is a discontinuous function to be specified later, and reconstructing $\mathcal{A}_t(X) = X + \phi_t(X)$, for $X \in \mathcal{C}_0$. The change of reference frame, from Eulerian to ALE, requires particular care as additional terms appear due to the different evolution in time of the computational

domain and fluid continuum. Given an arbitrary function $a(t, X) : [0, T] \times \mathcal{C}_0 \to \mathbb{R}^d$ defined in the ALE frame over its reference domain $\mathcal{C}_0$, it is possible to express it in the Eulerian frame as $a(t, \mathcal{A}_t^{-1}(x)) : [0, T] \times \mathcal{C}_t \to \mathbb{R}^d$. The time derivative of a in the ALE frame can be compared to its expression in the Eulerian frame

$$\frac{\partial a}{\partial t}\Big|_{\mathcal{A}_t}(t, x) = u_{\mathcal{A}} \cdot \nabla a + \frac{\partial a}{\partial t}, \quad x \in \mathcal{C}_t, \tag{II.1.14}$$

where the additional term appearing in the equations is the ALE velocity $u_{\mathcal{A}}(t, X) = \frac{\partial x}{\partial t}(t, \mathcal{A}_t^{-1}(x))$, $X \in \mathcal{C}_0$, which is the velocity of the moving domain. In the Eulerian frame $u_{\mathcal{A}} = 0$, as $\forall t \in [0, T]$, $\mathcal{C}_0 = \mathcal{C}_t$. Therefore, $\mathcal{A}_t$ describes the position of the moving domain at time t, which is in accordance with the deformation of the fluid continuum at its boundary, but obeys a different evolution in its interior.

II.1.3 Fluid model with moving bodies

The motion of a body can be described using two reference frames: a local one, translating and rotating with the body, and a global one, fixed in space. A change of reference frame allows to express the motion of the body in the global reference frame, that is usually preferred for the description of fluid flows. If one denotes by $O - xyz$ the global reference frame and by $P(t) - XYZ$ the local reference frame, one can describe the motion of $P(t) - XYZ$ as seen from $O - xyz$ by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = P(t) + R(t) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{II.1.15}$$

Here $R(t)$ is the rotation matrix allowing to change the reference frame from local to global, whose columns contain the coordinates of the local basis in the global reference frame. The change of reference frame impacts all the vector and matrix quantities that are computed in the local reference frame, in our case the moment of inertia and the deformation velocity. In the following, moving bodies will be described using the global reference frame.

Let us consider Navier-Stokes equations in moving domain to describe the motion of the Newtonian fluid medium. Let $\mathcal{C}_t \subset \mathbb{R}^d$, where $d = 2, 3$ denote the region occupied by the fluid at time t, μ the constant fluid viscosity and ρ the constant fluid density. Let $\mathcal{S}_t \subset \mathbb{R}^d$ be the domain that is occupied by the body, of constant density ρ_s. At each time instant t, the fluid is in contact with the swimmer, and $\bar{\mathcal{C}}_t \cap \bar{\mathcal{S}}_t = \partial\mathcal{S}_t$.

The velocity field $u(t, x) :]0, T] \times \mathcal{C}_t \to \mathbb{R}^d$ and the pressure field $p(t, x) :]0, T] \times \mathcal{C}_t \to \mathbb{R}$ satisfy Navier-Stokes equations

$$\rho\Big(\frac{\partial u}{\partial t} + (u \cdot \nabla)u\Big) = -\nabla p + \mu\Delta u + f \qquad \text{on }]0, T] \times \mathcal{C}_t, \tag{II.1.16a}$$

$$\nabla \cdot u = 0 \qquad \text{on }]0, T] \times \mathcal{C}_t, \tag{II.1.16b}$$

$$u = \bar{u} \qquad \text{on }]0, T] \times \partial\mathcal{S}_t, \tag{II.1.16c}$$

$$u(0, x) = u_0(x) \qquad \text{on } \{0\} \times \mathcal{C}_0, \tag{II.1.16d}$$

$$u(t, x) = h(t, x) \qquad \text{on }]0, T] \times \partial\mathcal{C}_{t,D}, \tag{II.1.16e}$$

$$\sigma(t, x)\vec{n} = (-p\mathbb{I} + \mu D(u))\vec{n} = g(t, x) \qquad \text{on }]0, T] \times \partial\mathcal{C}_{t,N}, \tag{II.1.16f}$$

where $\bar{u}(t, x) : [0, T] \times \mathcal{C}_t \to \mathbb{R}^d$ results from the interaction between the fluid and the body and $f(t, x) : [0, T] \times \mathcal{C}_t \to \mathbb{R}^d$ represents external volume forces. Here $h(t, x)$ and $g(t, x)$ represent the

Dirichlet and Neumann boundary conditions, prescribed on the respective portions of the fluid boundary $]0,T]\times\partial\mathcal{C}_{t,D}$ and $]0,T]\times\partial\mathcal{C}_{t,N}$. We have that $\partial\mathcal{C}_t=\partial\mathcal{C}_{t,D}\cup\partial\mathcal{C}_{t,N}\cup\partial\mathcal{S}_t$ for all times. In the case of swimming, $\bar{u}$ can be split into two contributions. The first one is determined by the translational velocity $U(t):[0,T]\to\mathbb{R}^d$ and angular velocity $\omega(t):[0,T]\to\mathbb{R}^{d^*}$ of the swimmer around its center of mass x^{CM}, where $d^*=1$ if $d=2$ and $d^*=3$ if $d=3$, while the second one is given by the deformation velocity of the solid u_d, if present. The function $\bar{u}$ then becomes $\bar{u}(t,x)=U(t)+\omega(t)\times(x-x^{CM})+R(t)u_d(t,\mathcal{A}_t^{-1}(x))$.
The translational and angular velocities result from the global force and torque balance on the swimmer, which impose that any acceleration of the swimming body is the result of its interaction with the fluid or with external net forces and torques, when present. If we denote by m the swimmer's mass and by $J(t)$ its moment of inertia in the local reference frame, by F_{ext} and M_{ext} the external force and momentum acting on the particle, the Newton equations read as

$$m\frac{dU}{dt}=F_{ext}-\int_{\partial\mathcal{S}_t}\sigma(u,p)\vec{n}, \tag{II.1.17a}$$

$$\frac{d[R(t)J(t)R(t)^T\omega]}{dt}=M_{ext}-\int_{\partial\mathcal{S}_t}(x-x^{CM})\times\sigma(u,p)\vec{n}, \tag{II.1.17b}$$

$$\frac{d\theta(t)}{dt}=\omega, \tag{II.1.17c}$$

where θ are the Euler angles used to describe the orientation of the body and to compute the rotation matrix $R(t)$. In two dimensions, $\theta\in[0,2\pi)$ and

$$R=R(\theta)=\begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}, \tag{II.1.18}$$

while in three dimensions $\theta\in[0,2\pi]\times[0,\pi]\times[-\pi,\pi]$ and

$$R=R(\theta)=R_z(\theta_z)R_y(\theta_y)R_x(\theta_x) \tag{II.1.19}$$

and R_z,R_y,R_x are the matrices expressing a rotation around the corresponding axes of the global frame.

The fluid problem is reformulated in the reference domain $\mathcal{C}_0$ thanks to the Arbitrary-Lagrangian Eulerian maps $\mathcal{A}_t:\mathcal{C}_0\to\mathcal{C}_t$, which are continuous, invertible and relate the current fluid domain $\mathcal{C}_t$ to the reference one $\mathcal{C}_0$. After this manipulation, equations (II.1.16) become

$$\rho\Big(\frac{\partial u}{\partial t}|_{\mathcal{A}}+(u-u_{\mathcal{A}})\cdot\nabla u\Big)=-\nabla p+\mu\Delta u+f \quad \text{on } \mathcal{C}_t, \tag{II.1.20a}$$

$$\nabla\cdot u=0 \quad \text{on } \mathcal{C}_t, \tag{II.1.20b}$$

$$u=U+\omega\times(x-x^{CM})+R(t)u_d\circ\mathcal{A}_t^{-1}(t,x) \quad \text{on } \partial\mathcal{S}_t. \tag{II.1.20c}$$

where $\frac{\partial u}{\partial t}|_{\mathcal{A}_t}(x)=\frac{\partial(u\circ\mathcal{A}_t)}{\partial t}(t,\mathcal{A}_t^{-1}(x))$ and $u_{\mathcal{A}}(t,x)=\frac{\partial x}{\partial t}(t,\mathcal{A}_t^{-1}(x))$ is the ALE velocity. As before we define the ALE maps to be $\mathcal{A}_t(X)=X+\phi_t(X)$, where $\phi_t(X)$ is obtained by solving

$$\begin{cases}\nabla\cdot((1+\tau(X))\nabla\phi_t)=0 & \text{in } \mathcal{C}_0,\\ \phi_t|_{\partial\mathcal{S}_0}=\bar{\phi}_t & \text{on } \partial\mathcal{S}_0.\end{cases} \tag{II.1.21}$$

The function $\bar{\phi}_t$ encodes the boundary displacement between $\partial\mathcal{S}_0$ and $\partial\mathcal{S}_t$ and equation (II.1.21) extends such boundary displacement inside the fluid domain. The proposed variational formulation of the coupled equations (II.1.17)-(II.1.20) follows closely [85], which ties the fluid and

body velocity spaces via the boundary conditions of the fluid problem. The following functional spaces are used as the test and trial spaces for the variational formulation of the coupled problem

$$Q^t = \{p \,|p(t,\cdot) \in L^2(\mathcal{C}_t))\}, \quad T = \{U \,|U(t) \in \mathbb{R}^d\}, \quad W = \{\omega \,|\, \omega(t) \in \mathbb{R}^{d^*}\},$$
$$V^t = \{u \,|\, u(t,\cdot) \in [H^1(\mathcal{C}_t)]^d, u = U + \omega \times (x - x^{CM}) \text{ on } \partial\mathcal{S}_t \text{ for } U \in T \text{ and } \omega \in W\}.$$

In the variational formulation of the problem, we look for the solution $(u, p, U, \omega) \in V^t \times Q^t \times T \times W$ such that, for all the test functions $(\tilde{u}, \tilde{p}, \tilde{U}, \tilde{\omega}) \in V^t \times Q^t \times T \times W$ one has

$$\int_{\mathcal{C}_t} \rho\Big(\partial_t u|_{\mathcal{A}} + (u - u_{\mathcal{A}}) \cdot \nabla u\Big) \cdot \tilde{u}\, dx + 2\mu \int_{\mathcal{C}_t} D(u) : D(\tilde{u})\, dx$$
$$- \int_{\partial\mathcal{S}_t} (-p\mathbb{I} + 2\mu D(u))\vec{n} \cdot \tilde{u}\, dS - \int_{\mathcal{C}_t} p\nabla \cdot \tilde{u}\, dx = \int_{\mathcal{C}_t} f \cdot \tilde{u}\, dx, \quad \text{(II.1.22)}$$

$$\int_{\mathcal{C}_t} \tilde{p}\nabla \cdot u\, dx = 0, \quad \text{(II.1.23)}$$

$$m\frac{dU}{dt} \cdot \tilde{U} = F_{ext} \cdot \tilde{U} - \int_{\partial\mathcal{S}_t} (-p\mathbb{I} + 2\mu D(u))\vec{n} \cdot \tilde{U}\, dS, \quad \text{(II.1.24)}$$

$$\frac{d[R(t)J(t)R(t)^T\omega]}{dt} \cdot \tilde{\omega} = M_{ext} \cdot \tilde{\omega} - \int_{\partial\mathcal{S}_t} (-p\mathbb{I} + 2\mu D(u))\vec{n} \times (x - x^{CM}) \cdot \tilde{\omega}\, dS. \quad \text{(II.1.25)}$$

As $\tilde{u} = \tilde{U} + \tilde{\omega} \times (x - x^{CM})$ on $\partial\mathcal{S}_t$, one can express the boundary integral in (II.1.22) as a combination of the integrals in (II.1.24)-(II.1.25)

$$\int_{\partial\mathcal{S}_t} (-p\mathbb{I}+2\mu D(u))\vec{n}\cdot\tilde{u}\, dS = \int_{\partial\mathcal{S}_t} (-p\mathbb{I}+2\mu D(u))\vec{n}\cdot\tilde{U}\, dS + \int_{\partial\mathcal{S}_t} (-p\mathbb{I}+2\mu D(u))\vec{n}\times(x-x^{CM})\cdot\tilde{\omega}\, dS, \quad \text{(II.1.26)}$$

which implies that encoding of rigid velocity boundary conditions in the test space ensures the transmission of hydrodynamic forces at the boundary of the swimmer. The variational equation coupling (II.1.22)-(II.1.24)-(II.1.25) via (II.1.26) reads

$$\int_{\mathcal{C}_t} \rho\Big(\partial_t u|_{\mathcal{A}} + (u - u_{\mathcal{A}}) \cdot \nabla u\Big) \cdot \tilde{u}\, dx + 2\mu \int_{\mathcal{C}_t} D(u) : D(\tilde{u})\, dx - \int_{\mathcal{C}_t} p\nabla \cdot \tilde{u}\, dx$$
$$+ m\frac{dU}{dt} \cdot \tilde{U} + \frac{d[R(t)J(t)R(t)^T\omega]}{dt} \cdot \tilde{\omega} = \int_{\mathcal{C}_t} f \cdot \tilde{u}\, dx + F_{ext} \cdot \tilde{U} + M_{ext} \cdot \tilde{\omega}, \quad \text{(II.1.27)}$$

II.1.4 The swimming problem

The purpose of this section is to collect the fluid and solid models presented before and proposing the general mathematical form of the swimming problem that we will address. In all generality, we considered Navier-Stokes equations in moving domain to model the fluid flow. In the next section, we will discuss the necessary changes if the fluid model was based on Stokes equations.

Let $\mathcal{F}_0 \subseteq \mathbb{R}^d$, $d = 2, 3$, be the initial configuration of the fluid domain and $\mathcal{A}_t : \mathcal{C}_0 \to \mathcal{C}_t$ the smooth function that maps the reference fluid computational domain onto the computational domain $\mathcal{C}_t$ occupied by the fluid at time t. Let $\mathcal{S}_0 \subseteq \mathbb{R}^d$ be the domain occupied by the swimmer at time $t = 0$. Let (u, p) be the fluid velocity and pressure defined over $\mathcal{F}_t$, μ the viscosity, f the fluid volume forces. We denote by U and ω the translational and angular velocities of

the swimmer, by x^{CM} its center of mass. The coupled problem, expressing the kinematic and dynamic coupling of the fluid with the swimmer, reads: find (u, p, U, ω) such that

$$\begin{cases} \rho\Big(\dfrac{\partial u}{\partial t}|_{\mathcal{A}} + (u - u_{\mathcal{A}}) \cdot \nabla u\Big) = -\nabla p + \mu \Delta u + f, & \text{in } \mathcal{C}_t, \\ \nabla \cdot u = 0, & \text{in } \mathcal{C}_t, \\ m\dfrac{dU}{dt} = \displaystyle\int_{\partial \mathcal{S}} (-p\mathbb{I} + 2\mu D(u))\vec{n}\, dS, & \\ \dfrac{d[R(t)J(t)R(t)^T\omega]}{dt} = \displaystyle\int_{\partial \mathcal{S}} (-p\mathbb{I} + 2\mu D(u))\vec{n} \times (x - x^{CM})\, dS, & \\ u = U + \omega \times (x - x^{CM}(t)) + R(t)u_d \circ \mathcal{A}_t^{-1}(t, x), & \text{on } \partial\mathcal{C}_t \cap \partial\mathcal{S}_t \\ u(t, x) = h(t, x) & \text{on }]0, T] \times \partial\mathcal{C}_{t,D}, \\ \sigma(t, x)\vec{n} = (-p\mathbb{I} + \mu D(u))\vec{n} = g(t, x) & \text{on }]0, T] \times \partial\mathcal{C}_{t,N}. \end{cases} \tag{II.1.28}$$

The map $\mathcal{A}_t(X)$ is defined as $\mathcal{A}_t(X) = X + \phi_t(X)$, where $\phi_t(X)$ is the solution of the following elliptic problem

$$\begin{cases} \nabla \cdot ((1 + \tau(X))\nabla\phi_t(X)) = 0 & \text{in } \mathcal{C}_0, \\ \phi_t(X) = \bar{\phi}_t(X), & \text{on } \partial\mathcal{C}_0\ , \end{cases} \tag{II.1.29}$$

where $\bar{\phi}_t(X)$ is obtained by integrating $\dot{\bar{\phi}}_t(X) = U + \omega \times (X + \phi(X) - X^{CM} - \phi(X^{CM})) + R(t)u_d(t, X)$. In this general formulation of the problem, the solid motion affects the fluid medium via the deformation velocity u_d, defined on the boundary of the swimmer. The deformation velocity u_d encodes the swimming stroke, and it varies with the micro-swimmer under consideration. As before, we divide our analysis into three categories of swimmers: swimmers composed of rigid parts, swimmers with prescribed deformation and elastic swimmers, to focus on the peculiarities of each of them.

II.1.4.1 Rigid swimmers

In the case of swimmers made of one rigid body, the propulsion mechanism is due to external forces and torques F_{ext} and M_{ext}. Using a non-zero F_{ext}, one can model the object being dragged in the fluid, for example by optical tweezers or magnetic forces [61]. Using a non-zero M_{ext}, one can model an object being rotated by an external torque, for example originated by a magnetic field [107]. When considering several rigid objects moving relatively to each other, it is possible to use u_d to prescribe the relative velocities between them. For example, in the case of the three-sphere swimmer [91], it is possible to translate the relative motion of the bodies into relative velocity prescription: the relative velocity can be modelled as a square wave, where positive values correspond to bodies approaching each other and negative values correspond to bodies moving away from each other.

II.1.4.2 Swimmers with prescribed deformation

In presence of an analytical formula for bodies' deformation velocity, this could be directly substituted to u_d. If the stroke is recovered from a simplified model, it must be transformed into a function that is compatible with domain and range of u_d. For example, one could have

an ODE model describing the centerline of a sperm's tail [62]

$$\begin{cases} \dfrac{d\mathbf{r}}{dl} = -\mathbf{e}_3, \\ \dfrac{d\mathbf{e}_1}{dl} = -\kappa_f \mathbf{e}_3 + \tau_f \mathbf{e}_2, \\ \dfrac{d\mathbf{e}_2}{dl} = -\tau_f \mathbf{e}_1, \\ \dfrac{d\mathbf{e}_3}{dl} = \kappa_f \mathbf{e}_1. \end{cases} \tag{II.1.30}$$

where $\mathbf{r}$ is the position vector, $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ give the orientation of the local reference frame and $l \in [0, L]$. In this case, the solution of the reduced model is extended as is to whole section because u_d is defined over the swimmer's boundary.

II.1.4.3 Elastic swimmers

In the case of elastic swimmers, the deformation velocity u_d is obtained from the solution of two or three-dimensional elasticity equations. In this formulation, the fluid and the swimmer fully interact and are able to influence the respective dynamics. Since u_d is not imposed in advance (analytically or via reduced models), it evolves in time due to the fluid-solid interaction. The coupling conditions between the fluid and the solid models have to translate the physical principles of continuity of stress and velocity fields on the common boundaries. At the same time, the geometric continuity of the fluid and solid domains is imposed. The expression of the coupling conditions depends on the ALE frame that was chosen to rewrite the fluid problem and decouples the motion of the fluid continuum from the fluid computational domain. The continuity of the solid and fluid velocities at the common boundary is written as

$$\frac{\partial \eta}{\partial t} - u \circ \mathcal{A}_t = 0, \quad \text{on } \partial\mathcal{S}_0 \cap \partial\mathcal{C}_0, \text{ for } t > 0, \tag{II.1.31}$$

where the velocity of the fluid continuum is expressed in the ALE reference frame. Let now $F_{\mathcal{A}_t} = \nabla \mathcal{A}_t$ be the Jacobian matrix of the ALE map $\mathcal{A}_t$; the continuity of stresses at the common boundary is written as

$$F\Sigma\vec{n} - \sigma F_{\mathcal{A}_t}^{-T} \vec{n} \det(\nabla \mathcal{A}_t) = 0, \quad \text{on } \partial\mathcal{S}_0 \cap \partial\mathcal{C}_0, \tag{II.1.32}$$

where σ_f is the Newtonian fluid stress tensor in the Eulerian reference frame, $F_{\mathcal{A}_t}^{-T} \det(\nabla \mathcal{A}_t)$ performs the change of frame from Eulerian to Arbitrary-Lagrangian-Eulerian and $\vec{n}$ is the outward-pointing normal of the solid domain. Finally, the geometric continuity is guaranteed by

$$x(X, t) - \mathcal{A}_t(X) = 0, \quad \text{on } \partial\mathcal{S}_0 \cap \partial\mathcal{C}_0, \tag{II.1.33}$$

which states that interpenetration of the fluid and solid domain is not possible. In the case of elastic swimmers, the hyper-elasticity equations

$$\rho_s \frac{\partial^2 \eta}{\partial t^2} - \nabla \cdot (F\Sigma) = f \quad \text{on } \mathcal{S}_0, \tag{II.1.34}$$

are solved for $\eta(t, X)$, coupled to (II.1.28) via the coupling conditions which link $u_d(t, X) = \frac{\partial \eta}{\partial t}(t, X)$ with the solution of the elasticity equations.

II.1.5 Discussion about Stokes equations

As it was discussed in the introduction, Stokes equations are the limit of Navier-Stokes equations for $Re = 0$. In this section we specialise the fluid-rigid body formulation we discussed for Navier-Stokes equations to the Stokes case. It will become clear that all the terms related to fluid or body inertia are neglected, and the balance between external and fluid forces and torques must be ensured at each time instant. In absence of external forces and torques, the balances of fluid forces and torques are known under the name of "self-propulsion constraints".

We first recall some concepts and results linked to the functional analytic framework of Stokes equations, that are useful for the finite element treatment. In particular, we derive the variational formulation of these equations in appropriate Hilbert spaces and discuss under which conditions the variational problem of the Stokes system has a unique solution. Mathematical analysis of these equations is well understood: existence and uniqueness of boundary value problems involving Dirichlet and Neumann conditions have been proved [19]. We restrict the case of homogeneous Dirichlet boundary conditions in a fixed domain, for the sake of simplicity.

Let $\mathcal{F} \subseteq \mathbb{R}^d$, $d = 2, 3$ the region occupied by the fluid domain, and let $u : \mathcal{F} \to \mathbb{R}^d$ and $p : \mathcal{F} \to \mathbb{R}$ be the fluid velocity and pressure fields. In the low Reynolds number limit, $Re \to 0$, incompressible Navier-Stokes equations reduce to Stokes equations

$$\begin{cases} -\mu\Delta u + \nabla p = f, & \text{in } \mathcal{F}, \\ \nabla \cdot u = 0, & \text{in } \mathcal{F}, \\ u = 0, & \text{on } \partial\mathcal{F}. \end{cases}$$

Suppose $f \in [L^2(\mathcal{F})]^d$. Let $(v, m) \in [H_0^1(\mathcal{F})]^d \times L^2(\mathcal{F})$ and $D(u) = \frac{1}{2}(\nabla u + \nabla u^T)$. The variational formulation of the aforementioned boundary value problem reads: find $(u, p) \in [H_0^1(\mathcal{F})]^d \times L^2(\mathcal{F})$ such that

$$\begin{aligned} &2\mu \int_{\mathcal{F}} D(u) : D(v)\, dx - \int_{\mathcal{F}} p\nabla \cdot v\, dx = \int_{\mathcal{F}} f \cdot v\, dx, && \forall v \in [H_0^1(\mathcal{F})]^d, \\ &\int_{\mathcal{F}} m\nabla \cdot u\, dx = 0, && \forall m \in L^2(\mathcal{F}). \end{aligned} \tag{II.1.35}$$

Let us remark that in II.1.35 we used the "mechanical" formulation of Stokes equations, where ∇u could be substituted by $D(u)$ thanks to the incompressibility of the velocity field and ∇v could be substituted by $D(v)$ since

$$\nabla v = D(v) + \frac{1}{2}(\nabla v - \nabla v^T), \qquad A : \frac{1}{2}(\nabla v - \nabla v^T) = 0 \text{ for all } A \text{ symmetric.} \tag{II.1.36}$$

This formulation, based on the symmetric gradient $D(u)$ is closer to physics and allows higher accuracy when imposing Neumann boundary conditions or computing fluid forces through boundary integration. Let us denote the bilinear and linear forms appeared in (II.1.35) as

$$\begin{aligned} a(u, v) &= 2\mu \int_{\mathcal{F}} D(u) : D(v)\, dx, \qquad b(v, p) = -\int_{\mathcal{F}} p\nabla \cdot v\, dx, \\ (f, v) &= \int_{\mathcal{F}} f \cdot v\, dx, \qquad b(u, m) = -\int_{\mathcal{F}} m\nabla \cdot u\, dx. \end{aligned}$$

We mention two ways to prove that this problem has a unique solution. For the first one, one defines the Hilbert space $H^0_{div} = \{v \in [H_0^1(\mathcal{F})]^d | \nabla \cdot v = 0\}$ and remarks $H^0_{div} \subset [H_0^1(\mathcal{F})]^d$. The variational formulation (II.1.35) reduces to finding $u \in H^0_{div}$ such that

$$a(u, v) = (f, v) \qquad \forall v \in H^0_{div}. \tag{II.1.37}$$

One can prove that the bilinear form $a(u,v)$ is coercive and continuous on H^0_{div} and that (f,v) is a continuous linear form on H^0_{div}. Lax-Milgram theorem for elliptic problems then guarantees existence and uniqueness of the solution to this problem.

The second way to solve this problem is to find the pair $(u,p) \in [H^1_0(\mathcal{F})]^d \times L^2(\mathcal{F})$ that, for all $(v,m) \in [H^1_0(\mathcal{F})]^d \times L^2(\mathcal{F})$, satisfies the following saddle-point formulation

$$\begin{aligned} &a(u,v) + b(v,p) = (f,v), \\ &b(u,m) = 0, \end{aligned} \tag{II.1.38}$$

over the Hilbert space $[H^1_0(\mathcal{F})]^d \times L^2(\mathcal{F})$. One can prove that the bilinear forms $a(\cdot,\cdot)$ and $b(\cdot,\cdot)$ are continuous, $a(\cdot,\cdot)$ is coercive over $X = \{v \in [H^1_0(\mathcal{F})]^d |\, b(v,m) = 0 \, \forall m \in L^2(\mathcal{F})\}$ and that there exists $\beta > 0$ such that

$$\forall m \in L^2(\mathcal{F}) \quad \exists v \in [H^1_0(\mathcal{F})]^d, \quad v \neq 0 : b(v,m) \geq \beta \|v\|_{[H^1_0(\mathcal{F})]^d} \|m\|_{L^2(\mathcal{F})}. \tag{II.1.39}$$

These conditions ensure existence and uniqueness of the solution (see [106]) and (II.1.39) is equivalent to the continuous version of the *inf-sup* or *Ladyzhenskaya-Babuška-Brezzi* condition. This theoretical result has consequences in the discrete setting as it constrains the choice of the discretization spaces for the fluid velocity and pressure.

Let us now consider the motion of a rigid particle P in a Stokes flow. As usual, we call $u(t,x) : [0,T] \times \mathcal{F}_t \to \mathbb{R}^d$ and $p(t,x) : [0,T] \times \mathcal{F}_t \to \mathbb{R}$ the fluid velocity and pressure, μ the viscosity, $f(t,x) : [0,T] \times \mathcal{F}_t \to \mathbb{R}^d$ the fluid volume forces; we denote by $U(t) : [0,T] \to \mathbb{R}^d$ and $\omega(t) : [0,T] \to \mathbb{R}^{d^*}$ the translational and angular velocities of the particle. Let F_{ext} and M_{ext} be the external force and momentum acting on the particle, x^{CM} its center of mass, $\vec{n}$ the unit normal pointing outward of the fluid domain and $\mathrm{d}S$ the area measure. The fluid-structure problem consists in finding (u,p,U,ω) satisfying

$$\begin{cases} -\mu\Delta u + \nabla p = f, & \text{in } \mathcal{F}_t, \\ \nabla \cdot u = 0, & \text{in } \mathcal{F}_t, \\ u = U + \omega \times (x - x^{CM}), & \text{on } \partial P, \\ 0 = F_{ext} - \displaystyle\int_{\partial P} (-p\mathbb{I} + 2\mu D(u))\vec{n}\, dS, \\ 0 = M_{ext} - \displaystyle\int_{\partial P} (-p\mathbb{I} + 2\mu D(u))\vec{n} \times (x - x^{CM})\, dS. \end{cases} \tag{II.1.40}$$

For the variational formulation of this problem, let $\tilde{u}, \tilde{p}, \tilde{U}, \tilde{\omega}$ denote the test functions. The equations become

$$2\mu \int_{\mathcal{F}_t} D(u) : D(\tilde{u})\, dx - \int_{\partial P} (-p\mathbb{I} + 2\mu D(u))\vec{n} \cdot \tilde{u}\, dS - \int_{\mathcal{F}_t} p\nabla \cdot \tilde{u}\, dx = \int_{\mathcal{F}_t} f \cdot \tilde{u}\, dx, \tag{II.1.41}$$

$$\int_{\mathcal{F}} \tilde{p} \nabla \cdot u\, dx = 0, \tag{II.1.42}$$

$$0 = F_{ext} \cdot \tilde{U} - \int_{\partial P} (-p\mathbb{I} + 2\mu D(u))\vec{n} \cdot \tilde{U}\, dS, \tag{II.1.43}$$

$$0 = M_{ext} \cdot \tilde{\omega} - \int_{\partial P} (-p\mathbb{I} + 2\mu D(u))\vec{n} \times (x - x^{CM}) \cdot \tilde{\omega}\, dS. \tag{II.1.44}$$

Following [85], the space of test functions is the collection of $(\tilde{u}, \tilde{U}, \tilde{\omega}) \in [H^1(\mathcal{F})]^d \times \mathbb{R}^d \times \mathbb{R}^{d^*}$ that satisfies boundary conditions $\tilde{u} = \tilde{U} + \tilde{\omega} \times (x - x^{CM})$ on ∂P. Hence, from previous equations (II.1.41)-(II.1.44), the relationship

$$\int_{\partial P} (-pI + 2\mu D(u))\vec{n} \cdot \tilde{u}\, dS = \int_{\partial P} (-pI + 2\mu D(u))\vec{n} \cdot \tilde{U}\, dS + \int_{\partial P} (-pI + 2\mu D(u))\vec{n} \times (x - x^{CM}) \cdot \tilde{\omega}\, dS, \tag{II.1.45}$$

holds, which allows the "compact" reformulation of the equations (II.1.41)-(II.1.43)-(II.1.44) as

$$2\mu \int_{\mathcal{F}_t} D(u) : D(\tilde{u})\, dx - \int_{\mathcal{F}_t} p\nabla \cdot \tilde{u}\, dx = \int_{\mathcal{F}_t} f \cdot \tilde{u}\, dx + F_{ext} \cdot \tilde{U} + M_{ext} \cdot \tilde{\omega}. \tag{II.1.46}$$

Since the boundary conditions that impose rigid body motion are encoded in the test (and trial) functions, the coupling in the fluid-particle problem can be addressed in a monolithic manner.

Chapter II.2

Solution strategy

In this chapter the swimming problem (II.1.28) is discretized using the finite element method, and the different aspects of its solution are addressed. Section II.2.1 is devoted to the discretization of swimmers; section II.2.2 focuses on the problems of mesh adaptation, domain discretization and remeshing; sections II.2.3-II.2.4 present the general formulation of the fluid and swimming problems in their discrete versions, while II.2.5 reports the description of the algebraic solution strategies.

II.2.1 Swimmers discretization

The domain occupied by the swimmer is triangulated using the same low order geometric approximation that will be used in the fluid's triangulation. However, high order geometric approximations, based on high order discretization of displacement and ALE map, could be also used. This approach is explored in [98, 24], which considered Navier-Stokes equations in moving domain and fluid-structure interaction problems with a quadratic geometric approximation.

Even though the interaction between the swimmer and the fluid is concentrated on the common boundary, the interior of the swimmer is discretized for all swimmer categories under study. For the three categories, the triangulation is used to perform the integrals defining the swimmer's center of mass and inertia tensor. In addition to this, the triangulation is needed to discretize the elasticity equations describing the deformation of elastic swimmers.

II.2.1.1 Rigid swimmers

Rigid swimmers are defined by their center of mass and inertia tensor, which are stored using arrays of suitable dimensions (2×1 and 1×1 in 2D, 3×1 and 3×3 in 3D). If several bodies are present and they are connected to one another, it is necessary to model the whole collection of bodies as a body itself. In this case, the center of mass and the inertia tensor of the collection of bodies has to be computed and updated following the motion of the swimmer.

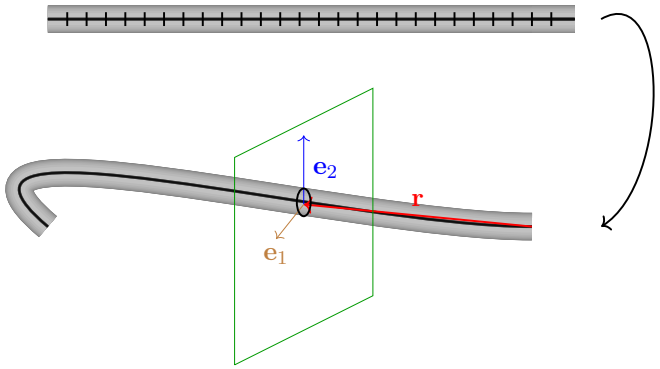


Figure II.2.1: The figure shows the procedure we follow to construct the three-dimensional tail as an extension of the centerline shape, dictated by (IV.2.3). First, the equations are solved in the reference, horizontal, configuration. Then, based on its coordinate in the horizontal direction, we assign to each node its position in the deformed configuration by using the local reference frame $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ and the linear interpolation of $\mathbf{r}$.

II.2.1.2 Swimmers with prescribed deformation

In the case of swimmers with prescribed deformation, we examined two possibilities. If the deformation velocity is analytically prescribed in the swimmer's reference frame, it can be easily evaluated for different times. In this case, the swimmer's displacement can be recovered by integrating the deformation velocity via an accurate explicit 4th order Runge-Kutta scheme and used to update the ALE map in the fluid domain.

In the case of reduced modelling based on ODEs, in which the position of the swimmer's tail is described in the local reference frame, we first solve the equations using an explicit 4th order Runge-Kutta scheme. We recall that, in our particular case, the ODE system describes the position of the centerline of the swimmer at each time instant (the variable being the arc-length l). Second, we compute the extension of the centerline displacement to the whole swimmer section based on the arc-length coordinate of the points. It is possible to recover the deformation velocity via automatic differentiation, since we solve the ODEs to obtain the tail's displacement, and use it to prescribe u_d on the boundary of the fluid domain.

In figure II.2.1, the construction of the three-dimensional tail is illustrated: first, the domain $[0, L]$ is subdivided in smaller segments and a three dimensional cylinder of radius a and length L is meshed; second, based on their x coordinate, the cylinder's nodes are displaced according to the solution of the ODEs, that is by performing a linear interpolation of the solution $\mathbf{r}$. The coordinates in the perpendicular directions to the cylinder axis become the components in the direction of $\{\mathbf{e}_1, \mathbf{e}_2\}$, giving $\mathbf{x}(l) = \mathbf{r}(l) + a\mathbf{e}_1(l) + a\mathbf{e}_2(l)$, where we suppose that the vectors $\{\mathbf{e}_1, \mathbf{e}_2\}$ remain constant in each discretization interval. Tail's coordinates are used to compute the tail's displacement, needed for the ALE maps, and the time derivative of the displacement, appears as a Dirichlet boundary condition in the fluid problem.

II.2.1.3 Elastic swimmers

The discretization of the hyper-elasticity equations (II.1.4)-(II.1.6) is performed using $\mathbb{P}^1$ continuous finite elements. Let us denote by $\mathcal{S}_0^h$ the triangulated elastic domain and let us consider the finite element spaces

$$\begin{aligned} M_h &= \{\eta \big| \eta \in [H^1(\mathcal{S}_0^h)]^d \cap [\mathbb{P}^1(\mathcal{S}_0^h)]^d\}, \\ M_{h,0} &= \{\eta \big| \eta \in [H_0^1(\mathcal{S}_0^h)]^d \cap [\mathbb{P}^1(\mathcal{S}_0^h)]^d\}. \end{aligned} \tag{II.2.1}$$

Let us denote the approximation of the second time derivative by $\partial_\tau^2 \eta$, obtained by applying a second order, unconditionally stable Newmark scheme

$$\begin{aligned} \ddot{\eta}^{n+1} &= \frac{1}{\beta \Delta t^2}(\eta^{n+1} - \eta^n) - \frac{1}{\beta \Delta t}\dot{\eta}^n - \Big(\frac{1}{2\beta} - 1\Big)\ddot{\eta}^n, \\ \dot{\eta}^{n+1} &= \frac{\gamma}{\beta \Delta t}(\eta^{n+1} - \eta^n) - \Big(\frac{\gamma}{\beta} - 1\Big)\dot{\eta}^n - \Delta t\Big(\frac{\gamma}{2\beta} - 1\Big)\ddot{\eta}^n, \end{aligned} \tag{II.2.2}$$

where $\beta = 0.25$ and $\gamma = 0.5$. The discrete problem at $t = t_{n+1}$ consists in finding $\eta^{t_{n+1}} \in M_h$ such that $\forall \phi \in M_{h,0}$

$$\int_{\mathcal{S}_0^h} \rho_s \partial_\tau^2 \eta^{t_{n+1}} \cdot \phi + \int_{\mathcal{S}_0^h} F^{t_{n+1}} \Sigma^{t_{n+1}} : \nabla \phi = \int_{\mathcal{S}_0^h} f^{t_{n+1}} \cdot \phi + \int_{\partial \mathcal{S}_0^h} F^{t_{n+1}} \Sigma^{t_{n+1}} \vec{n} \cdot \phi \, \mathrm{d}S. \tag{II.2.3}$$

The solution strategy is based on Newton non-linear iterations via the solution of the discrete linearized system presented in (II.1.7): until $\|\eta^{t_{n+1},k} - \eta^{t_{n+1},k-1}\| / \|\eta^{t_{n+1},k}\| \leq \varepsilon$ one solves

$$\begin{aligned} &\int_{\mathcal{S}_0^h} \rho_s \partial_\tau^2 \eta^{t_{n+1},k} \cdot \phi + \int_{\mathcal{S}_0^h} dF^{t_{n+1},k} \Sigma^{t_{n+1},k} : \nabla \phi + \int_{\mathcal{S}_0^h} F^{t_{n+1},k} d\Sigma^{t_{n+1},k} : \nabla \phi \\ &\qquad = \int_{\mathcal{S}_0^h} f^{t_{n+1}} \cdot \phi + \int_{\partial \mathcal{S}_0^h} F^{t_{n+1},k} \Sigma^{t_{n+1},k} \vec{n} \cdot \phi \, \mathrm{d}S + \int_{\mathcal{S}_0^h} F^{t_{n+1},k-1} \Sigma^{t_{n+1},k-1} : \nabla \phi. \end{aligned} \tag{II.2.4}$$

The discretization of the active elasticity equations is performed analogously, giving the following discrete problems to be addressed in the active stress case

$$\begin{aligned} &\int_{\mathcal{S}_0^h} \rho_s \partial_\tau^2 \eta^{t_{n+1}} \cdot \phi + \int_{\mathcal{S}_0^h} F^{t_{n+1}} (\Sigma^{t_{n+1}} - \Sigma_a^{t_{n+1}}) : \nabla \phi \\ &\qquad = \int_{\mathcal{S}_0^h} f^{t_{n+1}} \cdot \phi + \int_{\partial \mathcal{S}_0^h} F^{t_{n+1}} (\Sigma^{t_{n+1}} - \Sigma_a^{t_{n+1}}) \vec{n} \cdot \phi \, \mathrm{d}S, \end{aligned} \tag{II.2.5}$$

and active strain case, respectively

$$\begin{aligned} &\int_{\mathcal{S}_0^h} \rho_s \partial_\tau^2 \eta^{t_{n+1}} \cdot \phi + \int_{\mathcal{S}_0^h} |\det(F_a^{t_{n+1}})| F^{t_{n+1}} (F_a^{t_{n+1}})^{-1} [\Sigma(F)^{t_{n+1}} - \Sigma(F_a)^{t_{n+1}}] (F_a^{t_{n+1}})^{-T} : \nabla \phi \\ &= \int_{\mathcal{S}_0^h} f^{t_{n+1}} \cdot \phi + \int_{\partial \mathcal{S}_0^h} |\det(F_a^{t_{n+1}})| F^{t_{n+1}} (F_a^{t_{n+1}})^{-1} [\Sigma(F)^{t_{n+1}} - \Sigma(F_a)^{t_{n+1}}] (F_a^{t_{n+1}})^{-T} \vec{n} \cdot \phi \, \mathrm{d}S. \end{aligned} \tag{II.2.6}$$

As the linearised systems are similar to (II.2.4), they are not detailed here.

II.2.2 ALE map discretization

As the swimmer moves, the geometry of the fluid domain changes. In our framework we supposed the interface between the solid and fluid domains to be conforming, and the evolution of the fluid domain to be described in the ALE frame. The computation of the ALE map obeys equation (II.1.29); however, in the discrete setting, it is not guaranteed that displacing the mesh according to the solution of (II.1.29) will produce a valid triangulation when the mesh deformation is too important. Mesh quality measures assess the validity of the triangulation: if the minimum of the mesh quality field is above a predetermined threshold (depending on the quality measure, but typically close to 1 for fair measures [43]) the mesh deformation is "small" enough to evolve the domain just according to the ALE map, computed by solving (II.1.29); if the minimum of the mesh quality field falls below the threshold, the resulting mesh deformation is not viable and a remeshing procedure has to be considered before applying the ALE map. In order to handle this possibility, at every time step we work with three domains: the initial domains $\hat{\mathring{\mathcal{S}}}, \hat{\mathring{\mathcal{C}}}$, where the swimmer's equations are possibly cast, the reference domains $\hat{\mathcal{S}}_{t^n}, \hat{\mathcal{C}}_{t^n}$, domains of the discrete ALE maps after mesh reconstruction, and the current domain $\mathcal{S}_{t^{n+p}}, \mathcal{C}_{t^{n+p}}$. Let us denote by X the coordinates in the initial domains, X_{t_n} the coordinates in the n-th reference domain and by x the coordinates in the current domain. The initial domain and the reference domain are related by a bijective map $\mathcal{R}_s^{t^n} : \partial\hat{\mathring{\mathcal{S}}} \to \partial\hat{\mathcal{S}}_{t^n}$ ($\mathcal{R}_v^{t^n} : \hat{\mathring{\mathcal{S}}} \to \hat{\mathcal{S}}_{t^n}$) that encodes the correspondence between those surface (volume) geometric entities that are preserved across remeshing. In particular, this correspondence is defined on the domain occupied by the swimmer and whose connectivity remains unchanged. The reference and current domains are related by the displacement field, defined over the latest reference domain $\mathcal{S}_{t^{n+p}} = (\mathbb{I} + \eta(t^{n+p}))(\hat{\mathcal{S}}_{t^n})$. Relationships $\mathcal{R}_s^{t^n}, \mathcal{R}_v^{t^n}$ are valid until remeshing is again performed, in which case the correspondence is rebuilt in order to relate $\hat{\mathring{\mathcal{S}}}$ and the new reference domain. These steps are illustrated in figure II.2.2: the initial domain $\hat{\mathring{\mathcal{S}}}$ is connected with each reference domain via the correspondence maps $\mathcal{R}_s^t, \mathcal{R}_v^t$, and between remeshing steps swimmers evolve according to the displacement field η or the extension of the deformation velocity u_d, defined over $\hat{\mathring{\mathcal{S}}}$. The inverse correspondence $\mathcal{R}^{-1} : \hat{\mathcal{S}}_{t^n} \to \hat{\mathring{\mathcal{S}}}$ allows to evaluate the deformation velocity u_d (when available) in the reference domain at time t^n. Essentially, $\mathcal{R}^{t^n}$ corresponds to an ALE map between the initial domain $\hat{\mathring{\mathcal{S}}}$ and current reference domain $\hat{\mathcal{S}}^{t_n}$ of the swimmer, restricted to the geometric elements that are preserved across remeshing.

II.2.2.1 Small mesh deformations

In the case of small mesh deformations, equation (II.1.29) is solved using $\mathbb{P}^1$ continuous finite elements over the triangulated reference fluid domain $\hat{\mathcal{C}}_{t_0}^h$, where t_0 is the reference time, i.e. the latest time instant in which remeshing has been realised. Let us define the finite element spaces

$$\begin{aligned} X_{\bar{\phi}}^h &= \{\phi | \phi \in [H^1(\hat{\mathcal{C}}_{t_0}^h)]^d \cap [\mathbb{P}^1(\hat{\mathcal{C}}_{t_0}^h)]^d, \phi = \bar{\phi} \text{ on } \partial\hat{\mathcal{C}}_{t_0}^h\}, \\ X_0^h &= \{\phi | \phi \in [H_0^1(\hat{\mathcal{C}}_{t_0}^h)]^d \cap [\mathbb{P}^1(\hat{\mathcal{C}}_{t_0}^h)]^d\}. \end{aligned} \tag{II.2.7}$$

The computational domain $\mathcal{C}_{t_{n+1}}^h = \mathcal{A}_h^{t_{n+1}}(\hat{\mathcal{C}}_{t_0}^h)$ is obtained by computing the discrete ALE map $\mathcal{A}_h^{t_{n+1}}(X) = X + \sum_{t=t_0}^{t_{n+1}} \phi_h^t(X_t)$ via the solution of

$$\begin{cases} \nabla \cdot ((1 + \tau(X))\nabla\phi_h^{t_{n+1}}) = 0, & \text{on } \hat{\mathcal{C}}_{t_0}^h, \\ \phi_h^{t_{n+1}} = \bar{\phi}^{t_{n+1}}, & \text{on } \partial\hat{\mathcal{C}}_{t_0}^h. \end{cases} \tag{II.2.8}$$

Here $\bar{\phi}^{t_{n+1}}(X_{t_0}) = \int_{t_n}^{t_{n+1}} U + \omega \times (X_{t_0} + \phi^{t_n}(X_{t_0}) - X_{t_0}^{CM} - \phi^{t_n}(X_{t_0}^{CM})) + R(t_n) u_d(t, \mathcal{R}^{-1}(X_{t_0}))\,\mathrm{d}t$ and τ is a piecewise constant coefficient, defined on each element e of the domain's discretization as $\tau\big|_e = (1 - V_{min}/V_{max})/(V_e/V_{max})$, where V_{max}, V_{min} and V_e are the volumes of the largest, smallest and current element of the domain discretization [64]. This discontinuous coefficient takes large values for elements of smaller volume, so that the mesh deformation is mainly supported on larger elements. The time integration of $\bar{\phi}^{t_{n+1}}$ is performed numerically in two steps: first, the contributions coming from the translational and deformation velocities are integrated to compute the new center of mass $\bar{\phi}_1^{t_{n+1}}(X_{t_0}^{CM})$

$$\begin{aligned} \theta_{n+1} &= (t_{n+1} - t_n)\omega^n + \theta_n, \quad R(t_{n+1}) = R(\theta_{n+1}), \\ \bar{\phi}_1^{t_{n+1}}(X_{t_0}) &= (t_{n+1} - t_n)U^n + R(t_{n+1}) \int_{t_n}^{t_{n+1}} u_d \circ \mathcal{R}_{t_0}^{-1}(t, X_{t_0})\,\mathrm{d}X; \end{aligned} \tag{II.2.9}$$

second, the orientation of the body is computed using the new center of mass

$$\bar{\phi}^{t_{n+1}}(X_{t_0}) \;\approx\; R(t_{n+1})(X_{t_0} + \phi_1^{t_{n+1}}(X_{t_0}) - \phi_1^{t_{n+1}}(X_{t_0}^{CM})) + \phi_1^{t_{n+1}}(X_{t_0}^{CM}) - X_{t_0}. \tag{II.2.10}$$

U and ω are approximated at t_n since their values at t_{n+1} are computed during the solution of the fluid problem on $\mathcal{C}_{t_{n+1}}^h$. Few fixed point iterations are performed at each time step to ensure the convergence of the body's position. The deformation velocity $u_d \circ \mathcal{R}^{-1}$ is approximated with a higher order scheme, as a Runge-Kutta scheme if the analytical form of the velocity is known or a reduced model is available. If $u_d \circ \mathcal{R}^{-1}(t, X)$ is due to elasticity effects, it is possible to write it as $u_d \circ \mathcal{R}^{-1}(t, X) = \partial_t \eta(t, X)$, where $\eta(t, X)$ is the solution of the elasticity equations.

The variational form the problem is finding $\phi_h^{t_n} \in X_{\bar{\phi}}^h$ such that

$$\begin{aligned} &\int_{\mathcal{C}_0^h} (1 + \tau(X)) \nabla \phi_h^{t_n}(X) : \nabla v \;\mathrm{d}x = 0 \qquad && \forall v \in X_0^h, \\ &\phi_h^{t_n} = \bar{\phi}^{t_n}, && \text{on } \partial\hat{\mathcal{C}}_{t_0}^h. \end{aligned} \tag{II.2.11}$$

II.2.2.2 Mesh adaptation and remeshing

If the domain deformation is too large, the geometric evolution of the triangulated domain, following the extension of the boundary displacement $\bar{\phi}$ into the fluid domain via the ALE maps, may prevent an accurate solution of the system. This problem presents when the quality of the simplexes, a measure that depends on their geometry (e.g. edge lengths, angles, volume), is degraded. We first describe in detail a few quality measures that are computed by performing dimensionless ratios of geometric quantities [43]. These measures are *fair*, meaning that they satisfy the criteria of normalization, boundedness, non-dimensionality and they can detect all degenerate simplices [43]. Later, we address the remeshing procedures and algorithms that we considered to solve our problem.

In the case of a two dimensional simplicial mesh, the first quality measure we present is the ratio ρ of the radii of the inscribed and circumscribed circumferences to each element, r and R respectively. The quantity $2\rho = 2r/R$ is non-dimensional, takes value 1 when the triangle is equilateral and it tends to 0 when the triangle shape is needle-like or flattened (in [105] the inverse of ρ is chosen as a quality measure, so poor quality elements are characterised by $\frac{1}{\rho} \to \infty$). It is possible to express this quantity as a combination of the area A, half perimeter p and edge lengths $|e_i|$, $i = 1, 2, 3$, of the simplex as

$$2\rho = \frac{4A^2}{|e_1||e_2||e_3|p}. \tag{II.2.12}$$

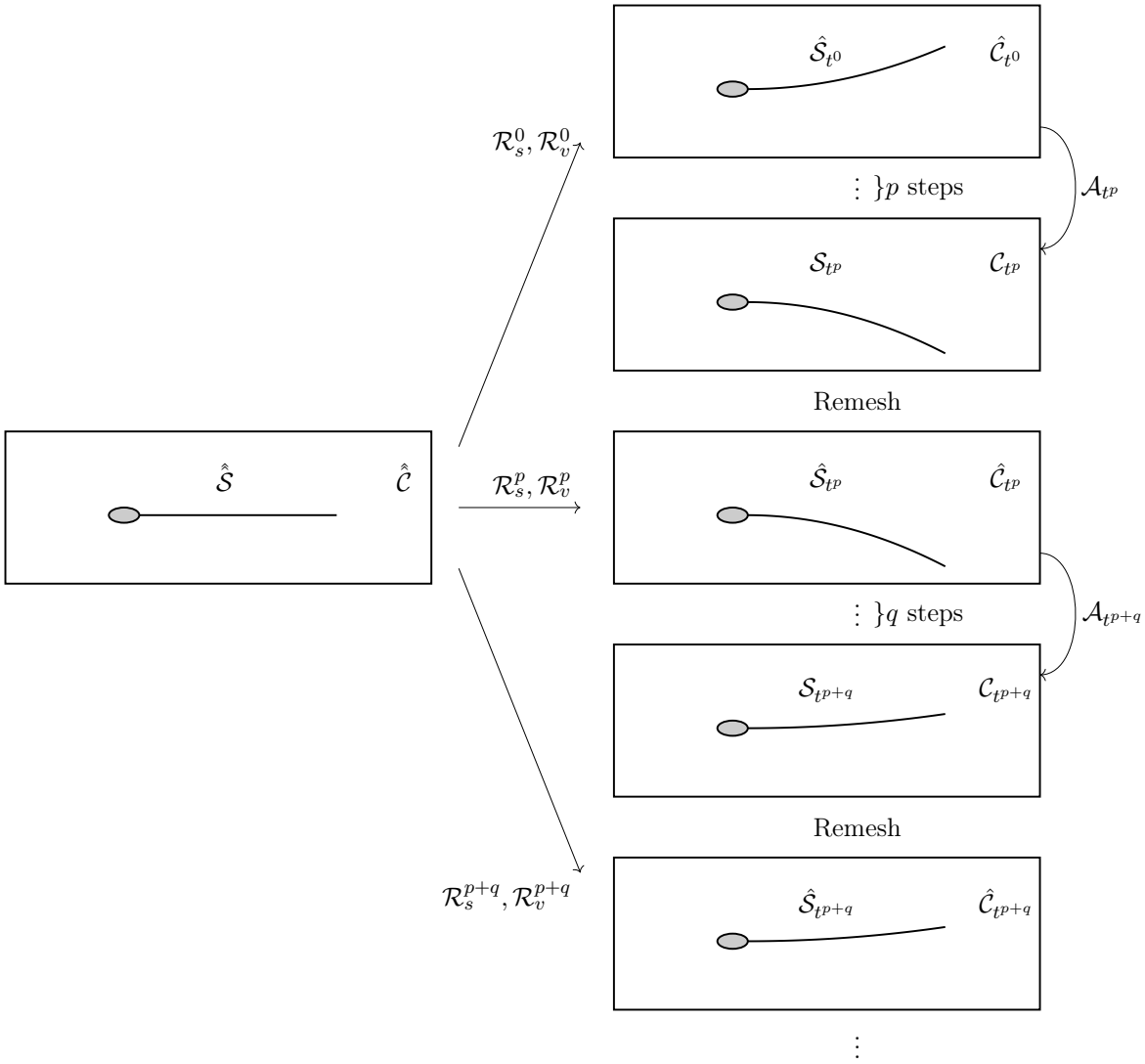


Figure II.2.2: The mesh adaptation procedure alternates remeshing steps with mesh motion. A relationship $\mathcal{R}^t$ between the initial swimmer configuration $\hat{\hat{\mathcal{S}}}$ and the current reference configuration $\hat{\mathcal{S}}_t$ allows the evaluation of displacements whose domain coincides with $\hat{\hat{\mathcal{S}}}$. Between two remeshing steps, the reference configuration of domain $\mathcal{S}_{t^{p+1}}$ is the latest remeshed domain $\hat{\mathcal{S}}_{t^p}$. Depending on the region where the connectivity is preserved, the relationship can be defined on the boundary of the swimmer ($\mathcal{R}_s^t$) or on its volume ($\mathcal{R}_v^t$).

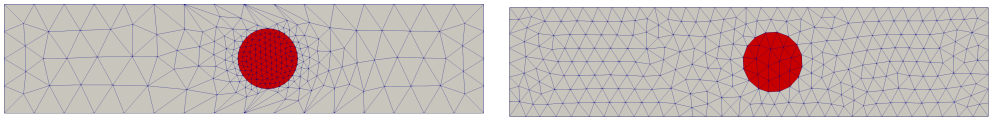


Figure II.2.3: Example of poor quality mesh around a translating circle (left) and its improvement after remeshing (right). The mesh shown here, however, is not yet satisfactory as the discretization of the boundary and surface of the disk has coarsened.

The definition of this quality measure can be extended to n-dimensional meshes as $n\rho = nr/R$, and takes the name of Normalized Shape Ratio (NSR, see [43]). Another 2D quality measure is

$$q_{2D} = \frac{4\sqrt{3}A}{|e_1|^2 + |e_2|^2 + |e_3|^2}. \tag{II.2.13}$$

It is scale invariant and $q_{2D} \leq 1$, where equality is attained only for an equilateral triangle. It is possible to show that in the interval $\frac{\sqrt{3}}{2} \leq q_{2D} \leq 1$, the element does not have any obtuse angle [12].

A 3D quality measure, function of the volume V and the facet areas $|A_i|$ of the tetrahedra, is

$$q_{3D} = \frac{2^3 3^8 \sqrt{3} V^4}{(\sum_{i=1}^{4} A_1^2)^3}. \tag{II.2.14}$$

As the previous quality measure, $q_{3D} \leq 1$ and equality is attained in the case of the equilateral tetrahedron [117]. Other measures that could be used in the case of a three dimensional problem are the NSR, that would be the ratio of the inscribed sphere and circumscribed sphere radii, or γ, defined using the length of the edges $|e_i|$ and the tetrahedron volume V following [97]:

$$\gamma = \frac{(\sqrt{\frac{1}{6}\sum_{i=1}^{6} |e_i|^2})^3}{V}. \tag{II.2.15}$$

Since this ratio takes the value $\gamma^* = 8.47967$ for an equilateral tetrahedron, the actual quality measure will be $\gamma^*/\gamma \leq 1$. Referring to the tests contained in [97], the remeshing procedure could be activated when the ratio $\gamma^*/\gamma < 0.5$, which corresponds to tetrahedra of poor quality.

The solution that was chosen to address the loss of mesh quality is the reconstruction of the computational mesh (remeshing) and the transfer of the simulation data onto the newly computed geometry. Remeshing is realised when the minimal value of the quality measure associated to simplices is lower than an empirically predetermined tolerance. For example, in figure II.2.3(left), the quality of the geometric discretization of the domain (computed via q_{2D} quality ratio) has deteriorated and its minimum has fallen below the threshold $\bar{q}_{2D} = 0.8$. Hence, a reconstruction of the triangulation is necessary to continue the calculations, and figure II.2.3(right) shows an example of this remeshing procedure. Independently from the quality measure being chosen, a common procedure of data transfer from the old to the new domain discretization is realized. In this procedure we take into account some constraints that arise from the fluid-structure problem we are solving.

Firstly, we choose to maintain the same boundary discretization across the remeshing steps, in order to keep the mass and the moment of inertia coherent throughout the simulation. This is realised by passing to the mesh adaptation algorithm the set of entities that must remain unvaried. The remeshing presented in figure II.2.3, for example, does not comply with these

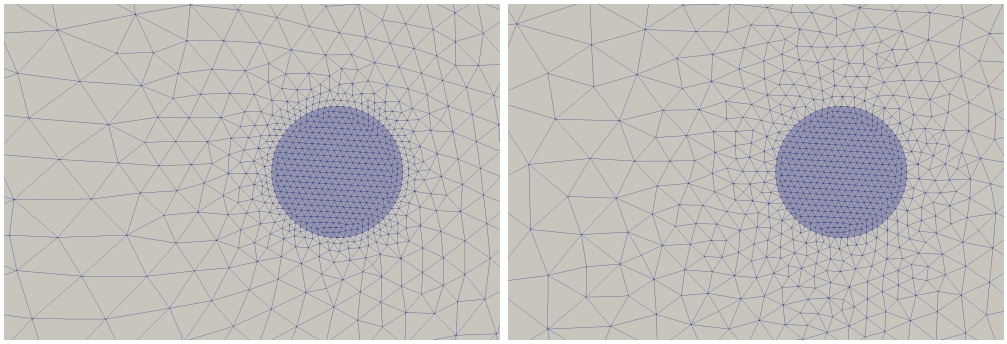


Figure II.2.4: Poor quality mesh and its improvement after remeshing. The newly computed mesh satisfies the constraint of unchanged boundary discretization of the immersed body. The mesh size is variable and finer close to the immersed body.

requirements, as the boundary discretization of the circle has changed. Secondly, depending on the distance from the swimming object, a varying mesh size is chosen, ensuring an increased accuracy close to the swimmer's boundary. Mesh size is encoded in the remeshing metric, which is defined as a scalar function $h_R \geq 0$ that prescribes the characteristic sizes of the elements. In order to capture the fluid behaviour in proximity of the swimmer, a graded remeshing strategy is chosen, with the remeshing metric being proportional to the distance d from the swimmer's boundary ($d = 0$ on $\partial\mathcal{S}_t^h$). To compute d, we use the Fast Marching Method (FMM), which is initialized on the set of degrees of freedom in contact with $\partial\mathcal{S}^h$ through an element by the exact distance computation [87]. In our case, the graded remeshing is obtained by prescribing a piecewise-defined metric depending on the characteristic size h of the previous mesh

$$h_R(x) = \begin{cases} h_{avg} & \text{if } d(x, \partial\mathcal{S}_t^h) \leq 2h, \\ h_{avg} e^{\alpha d(x,\partial\mathcal{S}_t^h)/h} & \text{if } d(x, \partial\mathcal{S}_t) > 2h, \end{cases} \tag{II.2.16}$$

where $h_R(x)$ is the mesh size at x, h_{avg} is the average mesh size of the previous mesh, $\alpha \geq 1$ is a prescribed constant. In (II.2.16) the proportionality function is continuous and exponentially increasing and a layer of fixed mesh size is defined for $d(x, \partial\mathcal{S}_t^h) \leq 2h$. It is possible to prescribe more layers of fixed mesh size by adding more cases to (II.2.16). Figure II.2.4 shows an example of remeshing where the boundary discretization remains unchanged and where the variable mesh size follows (II.2.16). The function $h_R(x)$ we presented here is discontinuous. However, since it is interpolated using a continuous finite element field and h is small, this aspect did not arise any problem. A possible choice of a continuous $h_R(x)$ could be $h_R(x) = h_{min} + d(x, \partial\mathcal{S}_t^h)(h_{max} - h_{min})/d_{max}$, where h_{min}, h_{max} are the minimal and maximal mesh size, and d_{max} is the maximal value of the distance function.

II.2.2.3 Remeshing and interpolation of fields

The geometric reconstruction of the computational domain is followed by the interpolation of the finite element fields onto the newly computed mesh. First, a localization procedure allows to pair each new node to the mesh element to which it previously belonged. Then, the interpolating polynomial is locally constructed and evaluated for each degree of freedom. For swimming simulations, the interpolation of finite element fields concerns the fluid velocity, as its values at previous time instants are needed for the discretization of the time derivative, and

the fields describing the evolution of the computational domain. In this latter case, the relevant quantities to be interpolated are the ALE maps, as they are used in the computation of the ALE velocity, and the displacement with respect to the reference domain, which is needed to compute the ALE maps. However, since at each remeshing step the newly created geometry becomes the reference configuration until the next remeshing step, it is necessary to change the reference of the displacement field η for the current and previous times. This means that, for $k = n+1, n, \ldots, n-i+1$, where i is the order of the time discretization,

$$\eta(t^k, X^R_{n+1}) = \mathcal{I}_k(\eta(t^k, \mathcal{A}_{t^k}(\mathcal{A}^{-1}_{t^{n+1}}(X_{n+1})))) \circ \mathcal{A}^{-1}_{t^k} \circ \mathcal{A}^{-1}_{t^{n+1}} - \mathcal{I}_{n+1}(\eta(t^{n+1}, X_n)), \tag{II.2.17}$$

where $\mathcal{I}_{n+1} : \mathcal{C}^h_{t_{n+1}} \to R(\mathcal{C}^h_{t_{n+1}})$ is the interpolation operator that maps functions defined over the poor quality domain $\mathcal{C}^h_{t_{n+1}}$ to its remeshed counterpart $R(\mathcal{C}^h_{t_{n+1}})$. Here X_{n+1} and X^R_{n+1} denote the coordinates in the two domains. In the first remeshing step, a relationship $\mathcal{R}$ between the reference mesh M_{ref} and the current mesh M_n is created, pairing the geometric entities that are kept fixed across the remeshing step. The relationship will be transferred at each following remeshing step, as the elements that are preserved do not change. This relationship is a bijective map between the boundary faces of the reference and current meshes, allowing to evaluate expressions that are defined in the reference configuration or in the swimmer's reference frame. The steps that are involved in the mesh adaptation process are collected and detailed in the algorithms that follow:

- Algorithm 2 shows the procedure that we follow to move the fluid mesh using elliptic smoothing. In this algorithm, the relationship $\mathcal{R}$ is used to evaluate the displacement of the boundary of the domain in its reference configuration.

Algorithm 2: Ext algorithm

Input: Mesh M_n, displacement η and relationship $\mathcal{R}$
Obtain the harmonic extension $\phi(t, X)$, with $\mathcal{R}(\eta)$ as Dirichlet data (cf. equation (II.1.29))
for i=1 **to** N_{nodes} **do**
$node(i)_{M_{n+1}} = node(i)_{M_n} + \phi(t, X)$
end for
Output: M_{n+1}

- Algorithm 3 shows the procedure that is followed to interpolate the solutions to the fluid problem and mesh displacement onto the new domain.

Algorithm 3: Interpolate algorithm

Input: Mesh M_n, M_{n+1}, fluid fields $u|_{M_n}, p|_{M_n}$, displacement field from reference configuration $\eta|_{M_n}$
$I = I(M_n, M_{n+1})$ construct the interpolation operator
for i=1 **to** N_{order} **do**
$u|_{M_{n+1}(i)} = I(u|_{M_n}(i)), p|_{M_{n+1}}(i) = I(p|_{M_n}(i))$
$\eta|_{M_{n+1}}(i) = \eta|_{M_n}(i) - \eta|_{M_n}(N_{order})$
end for
Output: $u|_{M_{n+1}}, p|_{M_{n+1}}$, $\eta|_{M_{n+1}}$

- Algorithm 4 explains the procedure that is followed to evolve the mesh and remesh, when necessary. In this case we use an incremental approach to implement the ALE transformation: the ALE map is decomposed in a fixed number of equivalent summands and steps of domain motion and remeshing are alternated until the whole deformation has been applied.

Algorithm 4: MoveRemesh algorithm

Input: Mesh M_n, relation $\mathcal{R}$, displacement η_{n+1}, and number of sub-iterations N

$\eta \leftarrow \eta_{n+1}/N$

for i=1 **to** N **do**

$M_{n+1}(i) = \text{Ext}(M_n^i, \mathcal{R}(\eta))$

if $meshQuality \leq tol$ **then**

Remesh

Interpolate

end if

end for

Output: M_{n+1}

- Algorithm 5 shows the procedure for the construction of the initial mesh from the reference one, by making use of the previously cited algorithms. This algorithm is useful, in particular, for the swimmer in section IV.2.2, as the reference and initial configurations do not coincide. In this case, $\mathcal{A}_0 \neq \mathcal{I}$ which is not usual in most of fluid-structure interaction studies.

Algorithm 5: InitialMesh algorithm

Input: Reference mesh M_{ref}, initial displacement η_0.

$[M_0, \mathcal{R}]$=Remesh(M_{ref})

MoveRemesh(M_0,$\mathcal{R}$,η_0,N)

Output: M_0

- Algorithm 6 contains the general solution strategy that combines all the previous algorithms. Starting from the reference mesh M_{ref} and initial displacement η_0, the initial mesh at time $t = 0$ is computed. Then, for $t \leq T$, the displacement η on the boundary of the fluid domain is recovered (the specific way to do this depends on the swimmer under consideration) and the displacement velocity u_d is computed by differentiating numerically η. Finally, the displacement velocity u_d is imposed on the swimmer's boundary and the solution of the fluid problem is addressed.

Algorithm 6: Solution algorithm

Input: Reference mesh M_{ref}, initial displacement η_0

M_0=InitialMesh(M_{ref},η_0)

while $t \leq T$ **do**

if $meshQuality \leq tol$ **then**

Remesh

Interpolate

end if

Evaluate η in M_{n+1}

Impose $u_d = \delta_t \eta$

Solve fluid problem

end while

Development details on the mesh adaptation procedure as it is implemented in *Feel++* are collected in chapter III.1.2.

II.2.3 The fluid-body problem

Let $\mathcal{T}_0$ be the triangulation discretizing the fluid domain $\mathcal{F}_0$ at time $t = 0$, which coincides with the computational domain $\mathcal{C}_0$. Since the computational domain $\mathcal{C}_t$ is moving according to the ALE maps $\mathcal{A}_t$, also the triangulation will be moving according to a discrete approximation of the ALE maps, that we denote $\mathcal{A}_h^t$. For each time instant t_n, $0 \leq n \leq n_T$, in which the time interval $[0, T]$ is subdivided, it is possible to define the triangulation $\mathcal{T}_{t_n} = \mathcal{A}_h^{t_n}(\mathcal{T}_0)$.

Let u_h^n and p_h^n denote the discrete approximations of the velocity and pressure fields at time t_n. The discrete approximation spaces for the fluid variables are now defined: as the domain is time dependent, the functional spaces are time dependent as well via the discrete ALE maps. In particular, they are related to the discrete approximation spaces on the reference domain via $\mathcal{A}_h^t$. The velocity and pressure spaces are, respectively,

$$\begin{aligned} V_h^t &= \{v : \mathcal{C}_t \to \mathbb{R}^d,\ v = \hat{v} \circ (\mathcal{A}_h^t)^{-1},\ \hat{v} \in H^1(\mathcal{C}_0) \cap [\mathbb{P}_N(\mathcal{C}_0)]^d\} \\ Q_h^t &= \{p : \mathcal{C}_t \to \mathbb{R},\ p = \hat{p} \circ (\mathcal{A}_h^t)^{-1},\ \hat{p} \in \mathbb{P}_{N-1}(\mathcal{C}_0)\}. \end{aligned} \tag{II.2.18}$$

We choose the Taylor-Hood finite element spaces $V_h^t - Q_h^t$ with $N = 2$ for our finite element simulations, as they satisfy the discrete version of the inf-sup condition [39]. Therefore, the velocity is discretized using continuous piecewise quadratic finite elements, and the pressure is discretized using continuous piecewise affine finite elements. The discrete variational formulation of the Navier-Stokes equations at time t_{n+1} requires finding $u_h^{n+1}, p_n^{n+1} \in V_h^{t_{n+1}} \times Q_h^{t_{n+1}}$, $(U^{n+1}, \omega^{n+1}) \in \mathbb{R}^d \times \mathbb{R}^{d^*}$ such that, for all $(v, q) \in V_h^{t_{n+1}} \times Q_h^{t_{n+1}}, (\tilde{U}, \tilde{\omega}) \in \mathbb{R}^d \times \mathbb{R}^{d^*}$

$$\begin{aligned} \rho \int_{\mathcal{C}_{t_{n+1}}} \partial_t u_h^{n+1} \cdot v - \rho \int_{\mathcal{C}_{t_{n+1}}} ((u_{\mathcal{A}_h}^{n+1} - u_h^{n+1}) \cdot \nabla_x u_h^{n+1}) \cdot v + \frac{d[R(t)J(t)R(t)^T\omega]}{dt}^{n+1} \cdot \tilde{\omega} + m\frac{dU}{dt}^{n+1} \cdot \tilde{U} \\ + 2\mu \int_{\mathcal{C}_{t_{n+1}}} D(u_h^{n+1}) : D(v) - \int_{\mathcal{C}_{t_{n+1}}} p_h^{n+1} \nabla_x \cdot v = \int_{\mathcal{C}_{t_{n+1}}} f^{n+1} \cdot v \end{aligned} \tag{II.2.19}$$

$$\int_{\mathcal{C}_{t_{n+1}}} q \nabla_x \cdot u_h^{n+1} = 0 \tag{II.2.20}$$

The degrees of freedom of u on the boundary of the body $\partial\mathcal{S}$ only depend on U and ω if we neglect u_d, as it is a known datum when solving the fluid equations. For this reason, these degrees of freedom are treated differently by considering the procedure proposed in [85]. Let us denote the degrees of freedom that belong to the boundary of the body by the subscript Γ as u_Γ, p_Γ and the others by the subscript I as u_I, p_I. Equations (II.1.41) can be split as

$$\begin{aligned} \rho \int_{\mathcal{C}_{t_{n+1}}} \partial_t u_h^{n+1} \cdot \tilde{u_I} - \rho \int_{\mathcal{C}_{t_{n+1}}} ((u_{\mathcal{A}_h}^{n+1} - u_h^{n+1}) \cdot \nabla_x u_h^{n+1}) \cdot \tilde{u_I} \\ + 2\mu \int_{\mathcal{C}_{t_{n+1}}} D(u_h^{n+1}) : D(\tilde{u_I})\, dx - \int_{\mathcal{C}_{t_{n+1}}} p_h^{n+1} \nabla \cdot \tilde{u_I}\, dx = \int_{\mathcal{C}_{t_{n+1}}} f^{n+1} \cdot \tilde{u_I}\, dx, \\ \rho \int_{\mathcal{C}_{t_{n+1}}} \partial_t u_h^{n+1} \cdot \tilde{u_\Gamma} - \rho \int_{\mathcal{C}_{t_{n+1}}} ((u_{\mathcal{A}_h}^{n+1} - u_h^{n+1}) \cdot \nabla_x u_h^{n+1}) \cdot \tilde{u_\Gamma} + 2\mu \int_{\mathcal{C}_{t_{n+1}}} D(u_h^{n+1}) : D(\tilde{u_\Gamma})\, dx \\ - \int_{\mathcal{C}_{t_{n+1}}} p_h^{n+1} \nabla \cdot \tilde{u_\Gamma}\, dx = \int_{\mathcal{C}_{t_{n+1}}} f^{n+1} \cdot \tilde{u_\Gamma}\, dx, \end{aligned} \tag{II.2.21}$$

and equation (II.1.42) can be split as

$$\begin{aligned}\int_{\mathcal{C}_{t_{n+1}}} \tilde{p}\nabla\cdot u_I\,dx = 0,\\ \int_{\mathcal{C}_{t_{n+1}}} \tilde{p}\nabla\cdot u_\Gamma\,dx = 0.\end{aligned} \tag{II.2.22}$$

Boundary terms of the form $\int_{\partial\mathcal{S}}(-p\mathbb{I}+2\mu D(u))\vec{n}\cdot\tilde{u}\,dS$ are never computed in the assembly of the system matrix. Instead of building the finite element basis spanning the constrained test space of $(\tilde{u},\tilde{U},\tilde{\omega})\in[H^1(\mathcal{C}_{t_{n+1}})]^d\times\mathbb{R}^d\times\mathbb{R}^{d^*}$ that satisfies boundary conditions $u=U+\omega\times(x-x^{CM})$ on $\partial\mathcal{S}$, we first use the standard finite element bases to discretize equation (II.1.46), getting

$$\begin{bmatrix} A_{II} & A_{I\Gamma} & 0 & 0 & B_I^T\\ A_{\Gamma I} & A_{\Gamma\Gamma} & 0 & 0 & B_\Gamma^T\\ 0 & 0 & T & 0 & 0\\ 0 & 0 & 0 & M & 0\\ B_I & B_\Gamma & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} u_I\\ u_\Gamma\\ U\\ \omega\\ p\end{bmatrix} = \begin{bmatrix} G_I\\ G_\Gamma\\ 0\\ 0\\ 0\end{bmatrix}, \tag{II.2.23}$$

where

$$A_{JK} = \rho\int_{\mathcal{C}_{t_{n+1}}}(\partial_t u_h^{n+1})_J\cdot\tilde{u_K} - \rho\int_{\mathcal{C}_{t_{n+1}}}((u_{\mathcal{A}_h}^{n+1}-u_h^{n+1})\cdot\nabla_x u_h^{n+1})_J\cdot\tilde{u_K}+ 2\mu\int_{\mathcal{C}_{t_{n+1}}} D(u_h^{n+1})_J : D(\tilde{u_K})\,dx,\quad \text{for } J,K\in I,\Gamma$$

$$B_I = -\int_{\mathcal{C}_{t_{n+1}}} p_h^{n+1}\nabla\cdot\tilde{u_I}\,dx \qquad B_\Gamma = -\int_{\mathcal{C}_{t_{n+1}}} p_h^{n+1}\nabla\cdot\tilde{u_\Gamma}\,dx,$$

$$G_I = \int_{\mathcal{C}_{t_{n+1}}} f^{n+1}\cdot\tilde{u_I}\,dx, \qquad G_\Gamma = \int_{\mathcal{C}_{t_{n+1}}} f^{n+1}\cdot\tilde{u_\Gamma}\,dx.$$

$$T = m\mathbb{I}, \qquad M = R^n J^n (R^n)^T,$$

Then, the operator

$$P = \begin{bmatrix}\mathbb{I} & 0 & 0\\ 0 & \tilde{P}_U & \tilde{P}_\omega\\ 0 & \mathbb{I} & 0\\ 0 & 0 & \mathbb{I}\end{bmatrix},$$

that satisfies the equation

$$(u_I,u_\Gamma,U,\omega)^T = P(u_I,U,\omega)^T + R(t)u_d$$

is built, as it performs the change of finite element basis from the standard Lagrange basis to the constrained one. In the previous matrix, $\tilde{P}_U$ and $\tilde{P}_\omega$ are the interpolation operators that allow the expression of u_Γ as a function of U and ω. In order to detail them, let us define the spaces

$$V_0 = \{v\in\mathcal{C}^0(\partial\mathcal{S}^h), v\big|_E\in\mathbb{P}_0(E)\ \forall E\in\partial\mathcal{S}^h\},$$
$$V_1 = \{v\in\mathcal{C}^0(\partial\mathcal{S}^h), v\big|_E\in\mathbb{P}_1(E)\ \forall E\in\partial\mathcal{S}^h\}.$$

Let us define the polynomial space $\mathbb{P}_N(\partial\mathcal{S}^h)$, where $N=2$ is the local polynomial degree of the fluid velocity approximation. The operator $\tilde{P}_U\big| : \mathbb{P}_N(\partial\mathcal{S}^h)\to V_0$ is defined by

$$\int_{\partial\mathcal{S}^h} u\varphi\,\mathrm{d}x = \int_{\partial\mathcal{S}^h} U\varphi\,\mathrm{d}x, \qquad \forall\varphi\in V_0, \tag{II.2.24}$$

and the operator $\tilde{P}_\omega\big| : \mathbb{P}_N(\partial\mathcal{S}^h) \to V_1$ is defined in an analogous manner by

$$\int_{\partial\mathcal{S}^h} u\varphi \,\mathrm{d}x = \int_{\partial\mathcal{S}^h} \omega \times (x - x^{CM})\varphi \,\mathrm{d}x, \qquad \forall \varphi \in V_1. \tag{II.2.25}$$

Consider the operator

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

i.e. the P operator extended to include pressure degrees of freedom. It is now possible to compute the matrix expressing the coupled fluid-rigid body problem by conjugation with $\mathcal{P}$

$$\mathcal{P}^T \begin{bmatrix} A_{II} & A_{I\Gamma} & 0 & 0 & B_I^T \\ A_{\Gamma I} & A_{\Gamma\Gamma} & 0 & 0 & B_\Gamma^T \\ 0 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & M & 0 \\ B_I & B_\Gamma & 0 & 0 & 0 \end{bmatrix} \mathcal{P} = \begin{bmatrix} A_{II} & A_{I\Gamma}\tilde{P}_U & A_{I\Gamma}\tilde{P}_\omega & B_I^T \\ \tilde{P}_U^T A_{\Gamma I} & \tilde{P}_U^T A_{\Gamma\Gamma}\tilde{P}_U + T & \tilde{P}_U^T A_{\Gamma\Gamma}\tilde{P}_\omega & \tilde{P}_U^T B_\Gamma^T \\ \tilde{P}_\omega^T A_{\Gamma I} & \tilde{P}_\omega^T A_{\Gamma\Gamma}\tilde{P}_U & \tilde{P}_\omega^T A_{\Gamma\Gamma}\tilde{P}_\omega + M & \tilde{P}_\omega^T B_\Gamma^T \\ B_I & B_\Gamma\tilde{P}_U & B_\Gamma\tilde{P}_\omega & 0 \end{bmatrix},$$

and the right-hand side of the coupled problem by multiplying it by $\mathcal{P}^T$

$$\mathcal{P}^T \begin{bmatrix} G_I \\ G_\Gamma \\ 0 \\ 0 \\ 0 \end{bmatrix} - \mathcal{P}^T \begin{bmatrix} A_{II} & A_{I\Gamma} & 0 & 0 & B_I^T \\ A_{\Gamma I} & A_{\Gamma\Gamma} & 0 & 0 & B_\Gamma^T \\ 0 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & M & 0 \\ B_I & B_\Gamma & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ R(t)u_d \circ \mathcal{A}_t^{-1} \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{II.2.26}$$

A similar system arises when applied to the Stokes equations, except that inertial terms disappear, including $T = M = 0$.

In the case of several independent immersed rigid bodies, the previous formulation extends naturally. We now look for a solution $(u_h^{n+1}, p_n^{n+1}, U_i^{n+1}, \omega_i^{n+1}) \in V_h^{t_{n+1}} \times Q_h^{t_{n+1}} \times [\mathbb{R}^d]^n \times [\mathbb{R}^{d^*}]^n$ to the weak formulation of the problem. Let $(\tilde{u}, \tilde{p}, \tilde{U}_i, \tilde{\omega}_i) \in V_h^{t_{n+1}} \times Q_h^{t_{n+1}} \times [\mathbb{R}^d]^n \times [\mathbb{R}^{d^*}]^n$ denote the test functions. The variational formulation reads

$$\begin{aligned} \rho \int_{\mathcal{C}_{t_{n+1}}} \partial_t u_h^{n+1} \cdot \tilde{u} - \rho \int_{\mathcal{C}_{t_{n+1}}} ((u_{\mathcal{A}_h}^{n+1} - u_h^{n+1}) \cdot \nabla_x u_h^{n+1}) \cdot \tilde{u} + \sum_{i=1}^{n} \left(\frac{d[R(t)J(t)R(t)^T\omega]}{dt}^{n+1} \cdot \tilde{\omega} + m\frac{dU}{dt}^{n+1} \cdot \tilde{U} \right) \\ + 2\mu \int_{\mathcal{C}_{t_{n+1}}} D(u_h^{n+1}) : D(\tilde{u}) - \int_{\mathcal{C}_{t_{n+1}}} p_h^{n+1} \nabla_x \cdot \tilde{u} = \int_{\mathcal{C}_{t_{n+1}}} f^{n+1} \cdot \tilde{u}, \end{aligned} \tag{II.2.27}$$

$$\int_{\mathcal{C}_{t_{n+1}}} \tilde{p} \nabla_x \cdot u_h^{n+1} = 0, \tag{II.2.28}$$

where $D(u) = \frac{1}{2}(\nabla u + \nabla u^T)$.

Following [85] once more, we choose $(\tilde{u}, \tilde{U}_i, \tilde{\omega}_i) \in V_h^{t_{n+1}} \times [\mathbb{R}^d]^n \times [\mathbb{R}^{d^*}]^n$ satisfying $\tilde{u} = \tilde{U}_i + \tilde{\omega}_i \times (x - x_i^{CM})$ on $\partial\mathcal{S}_i^h$.

If the rigid bodies are related by translational velocity constraints, as in the three-sphere swimmer example [91] we will consider later, a different formulation is considered. In the multi-body

swimmers we study, only one angular velocity and rotating frame are needed to describe the orientation of the swimmer. Since, in this case, we do not allow bodies to rotate independently of each other, the subscript i is dropped for the angular velocity to denote that it is shared among the bodies. In order to compute the contribution $\omega \times (x - x^{CM})$ to u, it is the center of mass x^{CM} of the multi-body system that is used, for which the subscript i is once more dropped. We denote by $\tilde{P}_{U_i}$ and $\tilde{P}_{\omega_i}$ the interpolation operators relative to body $\mathcal{S}_i^h$ that allow to express $u_{\partial\mathcal{S}_i^h}$ as a function of U_i and ω_i. In particular, if one denotes by D_i the number of velocity degrees of freedom that lie on $\partial\mathcal{S}_i^h$, one has that the interpolation operators relative to translational velocity is

$$\tilde{P}_U = \begin{bmatrix} P_{U_1} & \dots & 0_{D_1\times d} & \dots & 0_{D_1\times d} \\ 0_{D_i\times d} & \dots & P_{U_i} & \dots & 0_{D_i\times d} \\ 0_{D_n\times d} & \dots & 0_{D_n\times d} & \dots & P_{U_n} \end{bmatrix}. \tag{II.2.29}$$

The constraints on the relative velocities between the bodies can be imposed via Lagrange multipliers or through a modification of the operator $\mathcal{P}$.

The first method is the least invasive: additional equations and unknowns are added to a pre-existing discretized problem of n independent bodies, as described in equations (II.1.22)-(II.1.25). Lagrange multipliers $\alpha_i \in \mathbb{R}^d$, $i = 1 \dots n-1$ are introduced to impose the constraints on the translational velocities U_i onto the differential formulation. The previous constraints will appear in the equations describing the rigid motion of the solid bodies through the prescription of the relative translational velocities W_{in} between the i-th and the n-th body. Equations (II.1.24) will be substituted by

$$m_i\dot{U}_i \cdot \tilde{U}_i + \alpha_i \cdot \tilde{U}_i = -\int_{\partial\mathcal{S}_i^h} (-p\mathbb{I} + 2\mu D(u))\vec{n} \cdot \tilde{U}_i \, dS, \quad i = 1 \dots n-1, \tag{II.2.30}$$

$$m_n\dot{U}_n \cdot \tilde{U}_n - \sum_{i=1}^{n-1} \alpha_i \cdot \tilde{U}_n = -\int_{\partial\mathcal{S}_n^h} (-p\mathbb{I} + 2\mu D(u))\vec{n} \cdot \tilde{U}_n \, dS, \tag{II.2.31}$$

$$\alpha_i \cdot (U_i - U_n) = \alpha_i \cdot W_{in}, \quad i = 1 \dots n-1. \tag{II.2.32}$$

The addition of Lagrange multipliers entails the modification of $\mathcal{P}$ by providing an additional identity matrix of size $d(n-1) \times d(n-1)$ on the diagonal. Despite its easy application, this approach deteriorates the condition number of the system matrix, as we show in table IV.1.1 by comparing different solution strategies applied to the simulation of the three sphere swimmer. The data show that, depending on the solution strategy, the Lagrange multiplier approach can lead the algebraic solver to diverge, and in general is less efficient than the $\mathcal{P}$ matrix approach, that we now present.

Instead of using Lagrange multipliers to constrain the translational velocities, a modification of the operator $\mathcal{P}$ could give the same results. In terms of finite element spaces, this consists in reducing the constrained finite element space to basis functions that satisfy $u = U_n + \omega \times (x - x^{CM}(t)) + R(t)u_d(t)$ on $\partial\mathcal{S}_i^h$, with $u_d(t)$ function of W_{in}. More precisely, the fluid boundary conditions at the solid's boundary would read:

$$\begin{aligned} &u = U_n + \omega \times (x - x^{CM}(t)) + W_{in}(t), \quad \text{for } i = 1, \dots, n-1, && \text{on } \partial\mathcal{S}_i^h, \\ &u = U_n + \omega \times (x - x^{CM}(t)), && \text{on } \partial\mathcal{S}_n^h, \end{aligned} \tag{II.2.33}$$

which gives $u_d(t, x) = W_{in}(t)$ for $x \in \partial\mathcal{S}_i^h$, where we define $W_{nn}(t) = 0$. Now we can write

$$(u_I, u_{\partial B_i}, U_i, \omega, p)^T = \tilde{\mathcal{P}}(u_I, U_n, \omega, p)^T + R(t)u_d, \tag{II.2.34}$$

where $\tilde{\mathcal{P}}$ is given by

$$\tilde{\mathcal{P}} = \begin{bmatrix} \mathbb{I} & 0 & 0 & 0 \\ 0 & \tilde{\tilde{P}}_U & \tilde{P}_\omega & 0 \\ 0 & E & 0 & 0 \\ 0 & 0 & \mathbb{I} & 0 \\ 0 & 0 & 0 & \mathbb{I} \end{bmatrix}. \tag{II.2.35}$$

The $dn \times dn$ block that corresponds to the translational speeds, is presently substituted by the $dn \times d$ matrix

$$\begin{matrix} B_1 \left\{ \right. \\ \vdots \\ B_i \left\{ \right. \\ \vdots \\ B_n \left\{ \right. \end{matrix} \begin{bmatrix} \mathbb{I}_d \\ \vdots \\ \mathbb{I}_d \\ \vdots \\ \mathbb{I}_d \end{bmatrix} = E, \tag{II.2.36}$$

and matrix $\tilde{P}_U$ is substituted by

$$\tilde{\tilde{P}}_U = \begin{bmatrix} \tilde{P}_{U_1} \\ \vdots \\ \tilde{P}_{U_i} \\ \vdots \\ \tilde{P}_{U_n} \end{bmatrix}. \tag{II.2.37}$$

The relative velocities appear on the right-hand side of the problem, as they are encoded in u_d.

II.2.4 The swimming problem

In this section we discretise the swimming problems that were presented in the modelling section. As we did there, we divide this section according to the swimmers categories we identified earlier. We remark that the discretization of the swimmer's domain is necessary to compute dynamic quantities like mass, moment of inertia and center of mass, and that the coupling between the fluid and solid problems is performed by using the boundary discretization.

II.2.4.1 Rigid swimmers

The discrete version of the swimming problem for rigid swimmers, at time t_{n+1}, requires finding $u_h^{n+1}, p_n^{n+1} \in V_h^{t_{n+1}} \times Q_h^{t_{n+1}}$ and $(U^{n+1}, \omega^{n+1}) \in \mathbb{R}^d \times \mathbb{R}^{d^*}$, for all $(v, q) \in V_h^{t_{n+1}} \times Q_h^{t_{n+1}}$, as defined in (II.2.18), and $(\tilde{U}, \tilde{\omega}) \in \mathbb{R}^d \times \mathbb{R}^{d^*}$ such that

$$\rho \int_{\mathcal{C}_{t_{n+1}}} \partial_t u_h^{n+1} \cdot v - \rho \int_{\mathcal{C}_{t_{n+1}}} ((u_{\mathcal{A}_h}^{n+1} - u_h^{n+1}) \cdot \nabla_x u_h^{n+1}) \cdot v + 2\mu \int_{\mathcal{C}_{t_{n+1}}} D(u_h^{n+1}) : D(v)$$
$$+ \frac{d[R(t)J(t)R(t)^T\omega]}{dt}^{n+1} \cdot \tilde{\omega} + m \frac{dU}{dt}^{n+1} \cdot \tilde{U} - \int_{\mathcal{C}^h_{t_{n+1}}} p_h^{n+1} \nabla_x \cdot v = \int_{\mathcal{C}^h_{t_{n+1}}} f^{n+1} \cdot v + F_{ext}^{n+1} \cdot \tilde{U} + M_{ext}^{n+1} \cdot \tilde{\omega}, \tag{II.2.38}$$

$$\int_{\mathcal{C}^h_{t_{n+1}}} q \nabla_x \cdot u_h^{n+1} = 0, \tag{II.2.39}$$

These equations are valid in the case of one rigid body and of articulated swimmers, composed of several connected rigid bodies. In the first case, the mass, inertia tensor and center of mass are relative to the unique rigid body, while in the second case, they concern the whole articulated swimmer. The solution of the fluid-rigid coupled problem follows the formulation just presented in section II.2.3, based on the projection of u_h^{n+1} onto rigid-body-like velocities on the common boundaries.

The computational domain $\mathcal{C}_{t_{n+1}}^h = \mathcal{A}_h^{t_{n+1}}(\mathcal{C}_{t_0}^h)$ is obtained by computing the discrete ALE map $\mathcal{A}_h^{t_{n+1}} = x + \sum_{t=t_0}^{t_{n+1}} \phi_h^t$ via the solution of

$$\begin{cases} \displaystyle\int_{\mathcal{C}_0^h} (1+\tau(X))\nabla\phi_h^{t_{n+1}}(X) : \nabla v \, \mathrm{d}x = 0 & \forall v \in X_0^h \\ \phi_h^{t_{n+1}} = \bar{\phi}^{t_{n+1}}, & \text{on } \partial\mathcal{C}_{t_0}^h, \end{cases} \tag{II.2.40}$$

where $\bar{\phi}^{t_{n+1}}(X)$ is discretized as in (II.2.9)-(II.2.10). In this case, u_d is either zero (in the case of unique rigid body) or a function in analytical form in the case of relative motion between the rigid bodies. In this second case, an accurate integration scheme is used to compute $\int_{t_n}^{t_{n+1}} u_d \circ (\mathcal{A}_h^{t_n})^{-1} \, \mathrm{d}t$. The implementation of relative motion constraints can be realised using Lagrange multipliers or by modifying matrix P as described in section II.2.3.

II.2.4.2 Swimmers with prescribed deformation

The case of swimmers with prescribed deformation differs from the rigid swimmers' case only in the part concerning the treatment of u_d when using a reduced model for the swimmer's deformation. In this case, since the model we considered provided the shape of the swimmer at each time instant, the velocity was computed using automatic differentiation.

II.2.4.3 Elastic swimmers

The solution of discrete, coupled fluid-elastic problems can be realised using two types of schemes, i.e. monolithic or partitioned. In the first case, the fluid, structure and geometric problems are assembled in a unique system and solved simultaneously at each time instant; in the second case each of the fluid, solid and geometric sub-problems are individually solved. This second approach is preferable when the fluid and solid solvers are readily and independently available. This second technique, however, requires an iterative procedure that cycles between the solution of the different sub-problems to impose the coupling conditions [41, 73].

In partitioned algorithms, the coupling conditions can be treated implicitly, semi-implicitly or explicitly [73]. In the first case, at each step of the iterative loop that imposes the kinematic, dynamic and geometric continuity at the boundary for a fixed time, the geometry of the fluid domain is updated, which leads to a recalculation of the terms of the fluid problem. The semi-implicit approach, instead, keeps the geometry unvaried during the loop on the coupling conditions. In this case, since the geometric coupling is not exactly satisfied, more iterations and a smaller time step are required to ensure stability of the coupling [22]. The explicit treatment of coupling conditions does not guarantee the energy conservation at the interface, and using coupling schemes like Dirichlet-Neumann (Dirichlet boundary conditions on the fluid and Neumann on the solid) is not possible as they are not transparent to added-mass instabilities, appearing when the densities of the fluid and the structure are similar, as pointed out in [42] who proposed a Generalized Robin-Neumann scheme that is transparent to added-mass effects.

II.2.5 Algebraic representation and solution

The size and the sparsity of the matrices lead us to use iterative methods to solve the associated algebraic systems, and adapted preconditioning strategies are necessary to reduce the computational effort associated with the numerical solution. In this section we describe each of the systems we solve and the associated preconditioning strategies.

II.2.5.1 The algebraic systems

An algebraic system is associated to each of the following solution steps: computation of the ALE map, computation of the swimmer's dynamics using an elasticity model, solution of the problem coupling fluid flow and rigid-body motion. The algebraic system associated with the computation of the ALE maps, according to equation (II.2.8), is sparse, symmetric and positive definite. The preconditioning strategy that is employed for this system is based on algebraic multigrid preconditioners (`AMG`), that project the problem onto coarser spaces that are found by selecting the nodes providing larger contributions to the system matrix. This choice is dictated by the usage of an unstructured mesh for the domain discretization and the discontinuous nature of the coefficient $\tau(X)$ [20]. The algebraic system associated with the hyper-elasticity equations as reported in equation (II.2.3) is also solved using algebraic multigrid preconditioners coupled with an iterative algebraic solver like `GMRES`.

The system associated with the coupling of fluid and rigid body motion has a natural block structure, as shown in equation (II.2.23). This block structure, typical of indefinite problems discretised with mixed finite elements, is efficiently preconditioned with methods based on algebraic factorization, which approximate the Schur complement via a convection-diffusion-like problem in the pressure itself [38]. Let the system matrix $\mathcal{A} = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & 0 \end{bmatrix}$ be factorised in a block LDU form, with inverse of the form

$$\mathcal{A}^{-1} = \underbrace{\begin{pmatrix} I & -A_{00}^{-1}A_{01} \\ 0 & I \end{pmatrix}}_{U^{-1}} \underbrace{\begin{pmatrix} A_{00}^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix}}_{D^{-1}} \underbrace{\begin{pmatrix} I & 0 \\ -A_{10}A_{00}^{-1} & I \end{pmatrix}}_{L^{-1}}, \tag{II.2.41}$$

where $S = -A_{10}A_{00}^{-1}A_{01}$ is the Schur complement. Sub-matrix A_{00} contains the terms depending exclusively on fluid, translational and angular velocity, while A_{01} and A_{10} contain the pressure terms. In presence of a multi-body swimmer where the velocity constraints among its components are expressed in terms of Lagrange multipliers (modelled as in (II.2.32)), A_{00} will contain also the terms depending on the Lagrange multipliers. The preconditioner P_A has the same form of $\mathcal{A}^{-1}$, where A_{00}^{-1} and S^{-1} are substituted with less expensive versions $\hat{A}_{00}^{-1}$ and $\hat{S}^{-1}$

$$P_A = \underbrace{\begin{pmatrix} I & -\hat{A}_{00}^{-1}A_{01} \\ 0 & I \end{pmatrix}}_{U^{-1}} \underbrace{\begin{pmatrix} \hat{A}_{00}^{-1} & 0 \\ 0 & \hat{S}^{-1} \end{pmatrix}}_{D^{-1}} \underbrace{\begin{pmatrix} I & 0 \\ -A_{10}\hat{A}_{00}^{-1} & I \end{pmatrix}}_{L^{-1}}. \tag{II.2.42}$$

The choice of efficient $\hat{A}_{00}^{-1}$ and $\hat{S}^{-1}$ guarantees that the preconditioner is spectrally equivalent to the inverse matrix $P_A \approx \mathcal{A}^{-1}$.

In case of low Reynolds number flows, the preconditioner $\hat{S}$ of the Schur complement is proportional to the pressure mass matrix $M_p = [m_{ij}]$, $m_{ij} = \int_{\mathcal{C}_t} q_i q_j$ where q_i span the finite element space of the pressure. The scaling factor of the pressure mass matrix is the fluid viscosity, giving

$\hat{S}_{\text{pmm}} = \frac{1}{\mu} M_p$. In the case of $\mathbb{P}^1$ pressure approximation, the lumped version of M_p can be used as well. Both M_p and its lumped version are spectrally equivalent to the Schur complement S [38], and they provide a preconditioner which is independent from the mesh size h. In case of higher Reynolds number flows, a preconditioner based on a convection-diffusion problem in the pressure variable is used to approximate S. This preconditioner is given by the successive application of three operators $\hat{S}_{\text{pcd}} = M_p F_p^{-1} A_p$, where $A_p = [a_{ij}]$, $a_{ij} = \int_{\mathcal{C}_t} \nabla q_i : \nabla q_j$, and $F_p = [f_{ij}]$, where $f_{ij} = \mu \int_{\mathcal{C}_t} \nabla q_i : \nabla q_j + \int_{\mathcal{C}_t} (u_h \cdot \nabla q_j) q_i$, u_h is the most recent approximation of the fluid's velocity field and q_i span the finite element space of the pressure, as before.

Finally, $\hat{A}_{00}^{-1}$ can be chosen by exploiting the block-diagonal form of A_{00} (in case of Stokes equations or the Oseen approximation of Navier-Stokes equations), that is via multigrid approximation of the inverse of each of the blocks. Even when A_{00} is not block diagonal (e.g. $\mathcal{A}$ is the Jacobian matrix associated to Navier-Stokes equations), the previous preconditioner can be applied [38].

II.2.5.2 Preconditioning

In order to show that the preconditioning strategy for the fluid-body problem is adapted to the problem under consideration, we compare it to more generic preconditioners and verify its scaling properties. *Strong parallel scalability*, that refers to the decrease of the solution time as the problem size remains constant and the computational power increases, is used to compare the different preconditioning strategies. The speed-up, defined as the ratio of the solution time on one processor $T(1)$ and the solution time $T(N)$ on N processors,

$$S = \frac{T(1)}{T(N)},$$

is the measure of strong scalability we consider. We will look also at scalability for multigrid preconditioners, which stands for the invariance of the number of iterations needed by the iterative solver to converge at each time step, as the size of the problem grows.

There are two types of preconditioners we have explored for the solution of the coupled fluid and rigid body problem: monolithic preconditioners and block preconditioners. In the first group we considered LU preconditioning from the $MUMPS$ library. In the second group, we considered preconditioners based on algebraic factorisation with different preconditioning strategies on each of the blocks, ranging from algebraic multigrid to LU factorization.

Chapter II.3

Verification and validation

This chapter presents some of the computational tests that were devised to verify the correctness of the algorithms. The results that are presented in this section are obtained with *Feel++*. In particular, section II.3.1 collects tests on remeshing and mesh adaptation; section II.3.2 tests the swimmer gait models, especially the solution of active elasticity and the interplay of imposed deformation and deformation velocity; section II.3.3 presents a third set of tests consisting in two instances of immersed rigid bodies settling under gravity, in 2D and 3D.

II.3.1 Mesh adaptation tests

In these tests, available for sequential and parallel execution[1], the ability to effectively perform remeshing and function interpolation across a remeshing step is verified. Different cases are tested to verify if the procedure is robust in 2D and 3D: domains with one and two materials are remeshed, in order to see if mesh features are preserved in presence of a solid body immersed in a fluid; the interpolation of constants, linear functions, quadratic functions is verified to be exact; the solution of an elliptic problem is also interpolated to see the effects of interpolation on a piecewise polynomial finite element field; several remeshing steps with different metrics are also performed, in order to further check the robustness of the algorithm under repeated applications.

Let V_h be the finite element space defined on the initial discretization of $\Omega = [0,1]^2$ in 2D ($\Omega = [0,1]^3$ in 3D), and V_h^R the finite element space defined on the new discretization of Ω. Let u^* be the function to be interpolated. We denote by $u^R \in V_h^R$ the approximation of u^* on the new finite element space, by $I : V_h \to V_h^R$ the interpolation operator between the old and new finite element spaces and by $I(u) \in V_h^R$ the interpolation of the function $u \in V_h$ on the new mesh, where u is the approximation of u^* in V_h. The elliptic problem that is solved in Ω is

$$\begin{cases} -\Delta u = f, & \text{on } \Omega, \\ u = \sin(\pi x), & \text{on } \partial\Omega, \\ f = \pi^2 \sin(\pi x), \end{cases} \tag{II.3.1}$$

and its exact solution is $u^*(x) = \sin(\pi x)$. In this case, u^R will be the representation of the exact solution on V_h^R and $I(u)$ the interpolation of the numerical solution of the problem in

[1] These tests are located in the *Feel++* repository in file *feelpp/testsuite/feelmesh/test_remesh.cpp*.

V_h. The results of the interpolation are collected in table II.3.1, where the L^2 norm of the interpolation error $\|I(u)-u^R\|_{L^2}$ is reported. The results show the exactness of the interpolation on constants, linear functions and quadratic functions when using $\mathbb{P}^2$ continuous finite elements. The interpolation error on the elliptic problem is not negligible, but it is reduced when the mesh size of the initial discretization of Ω decreases. Algorithm 7 shows the sequence of operations that are performed to assess the interpolation error for each function, using different remeshing metrics g_i. As the results of the tests are similar in sequential and parallel, the table only reports

Algorithm 7: Interpolation test

Input: Mesh M_0, functions $\{u_i^*\}_{i=1}^{N_{functions}}$, projection operators Π, Π^R onto V_h, V_h^R, metrics $\{g_m\}_{m=1}^{N_{remesh}}$

```
Create V_h on M_0
for m=1 to N_remesh do
    M_m = Remesh(M_0, g_m)
    for i=1 to N_functions do
        Create V_h^R on M_m
        u = Π(u_i^*) or u = Solve(II.3.1)
        u^R = Π^R(u_i^*)
        Build I : V_h → V_h^R
        Compute ||I(u) − u^R||_{L^2}
    end for
end for
```

the result obtained when running the tests in sequential.

II.3.2 Swimming tests

II.3.2.1 Prescribed deformation model

A correct approximation of the beating tail is fundamental to recover the behaviour and interaction of the swimmer with the surrounding fluid. Hence, it is necessary to choose an appropriate time integration scheme to recover the tail's shape from its deformation velocity. Since we restrict to "prescribed deformation" models, the evaluation of the deformation velocity is inexpensive and schemes with many evaluations can be chosen.

	One material		Two materials	
Interpolation error (2D)	Constant	1.75e-15	Constant	1.75e-15
	Linear	5.52e-16	Linear	1.52e-16
	Quadratic	8.51e-16	Quadratic	8.51e-16
	Elliptic problem	8.02e-5	Elliptic problem	9.33e-6
Interpolation error (3D)	Constant	5.55e-15	Constant	5.54e-15
	Linear	2.31e-15	Linear	1.57e-15
	Quadratic	2.34e-15	Quadratic	1.55e-15
	Elliptic problem	3.32e-5	Elliptic problem	6.49e-5

Table II.3.1: Test on the interpolation error across remeshing steps.

Figure II.3.1: Comparison between the position of the two-dimensional approximation of the tail beating proposed in (IV.2.3) and its reconstruction from the deformation velocity, using different time integration schemes. On the left, the reconstruction of the tail position from the deformation velocity using a Runge-Kutta 4th order scheme (green) is compared with the one using an implicit Euler scheme (blue). On the right, for comparison, the position of the tail (red, obtained by solving (IV.2.3) via 4th order Runge-Kutta in the arclength variable, for every time instant t) is again compared to the reconstruction from the deformation velocity using implicit Euler (blue).

We focus on the tail beating proposed in section IV.2.2, where the centerline of the flagellum is obtained via the solution of the system of ODEs (IV.2.3). At each time instant the solution of (IV.2.3) describes the position of the tail, which can be used as a term of comparison for the shape resulting from time integration of the deformation velocity. In order to have a clearer picture of the effects of the time integration schemes, we restricted to the two-dimensional projection, on the plane xz, of the tail beating proposed in section IV.2.2. Automatic differentiation is used to derive the deformation velocity of the centerline from equation (IV.2.3). Two time discretization schemes were compared to assess their performance on the reconstruction of the swimmer's body from its deformation velocity u_d, namely implicit Euler and 4th order Runge-Kutta (in the time variable). Figure II.3.1 compares the tail reconstruction of the two integration schemes with the position of the tail, obtained by solving (IV.2.3) at every time instant with a 4-th order Runge-Kutta scheme (in the arclength variable). One can see that using a first order implicit Euler scheme is not sufficient to recover the exact position of the swimmer's beating tail, while a Runge-Kutta scheme of 4th order is able to recover it. This leads us to prefer the latter integration scheme to reconstruct the swimmer's body.

As the volume of the swimmers we consider is discretized, the reconstruction of the swimming gait is coupled to mesh adaptation and remeshing. Figure II.3.2 illustrates that it is possible to subdivide the gait reconstruction in sub-steps that alternate mesh smoothing and remeshing. Further tests were performed to verify the relationship between the reference and current meshes for the evaluation of u_d, and to monitor the evolution of the swimmer's volume (or area in 2D) along the simulation. We have shown previously that constant functions were preserved across remeshing, hence verifying that the volume of the swimmer is preserved across remeshing is a corollary of the previous tests. It is not guaranteed, however, that the volume of the swimmer remains constant while swimming, especially if the gait results from an extension of a 1D model.

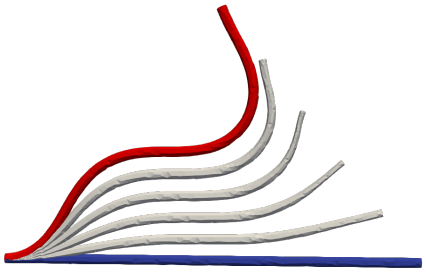


Figure II.3.2: Mesh degradation can be avoided if the displacement between two tail positions is subdivided in equal parts. Mesh smoothing and remeshing are alternately applied in order to move between the initial and final tail configurations while guaranteeing a minimal value of mesh quality.

II.3.2.2 Active elasticity models

In order to test the solution to the active elasticity problems (II.1.10)-(II.1.11), we perform a convergence study on manufactured solutions. In order to verify the correct implementation of the 2D active elasticity models, we started from the exact solution

$$\eta^* = \begin{bmatrix} 0 \\ \sin(x) \end{bmatrix},$$

to construct the test problems to be solved. In the active stress case, the active stress tensor

$$\Sigma_a = \begin{bmatrix} 0 & 0 \\ 0 & -x\sin(y) \end{bmatrix},$$

was prescribed, while in the active strain case, the active component of the deformation gradient

$$F_a = \begin{bmatrix} 1 & 0 \\ 0 & 1.02 + 0.01\sin(y) \end{bmatrix},$$

was prescribed. The forcing term f was obtained by inserting these components and the exact solution in equations (II.1.10) and (II.1.11). In the test problems we considered, equations (II.1.10) and (II.1.11) were solved with f computed as before, and Dirichlet boundary conditions are equal to η^* on the swimmer's surface. The convergence tests presented in figures II.3.3 and II.3.4 show the expected quadratic decrease of the L^2 error and linear decrease of the H^1 error as the maximum mesh size h_{max} of the swimmer's triangulation decreases.

II.3.3 Fluid tests

II.3.3.1 Sphere subject to gravity

In this section we consider a test that validates a fundamental building block for swimming simulation, that is the interaction between fluid and solid bodies, in three dimension. It consists in computing terminal velocity and fluid forces on a rigid sphere that is immersed in a viscous

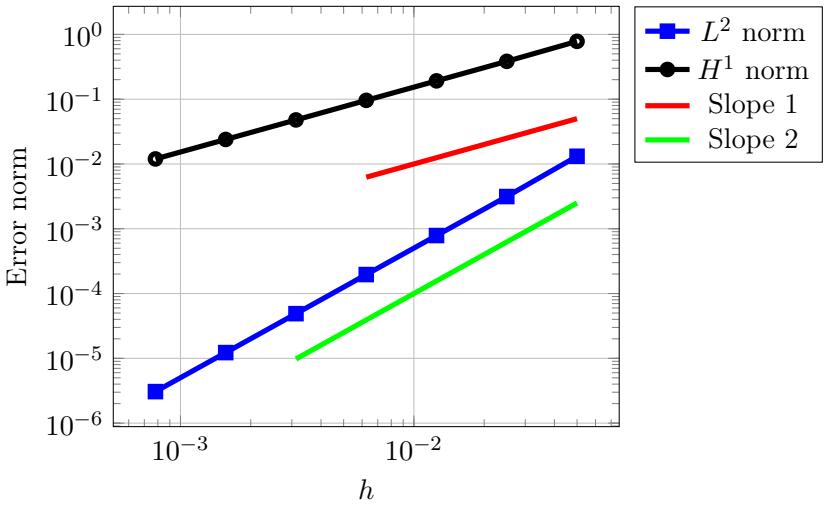


Figure II.3.3: Convergence test for active stress elastic model.

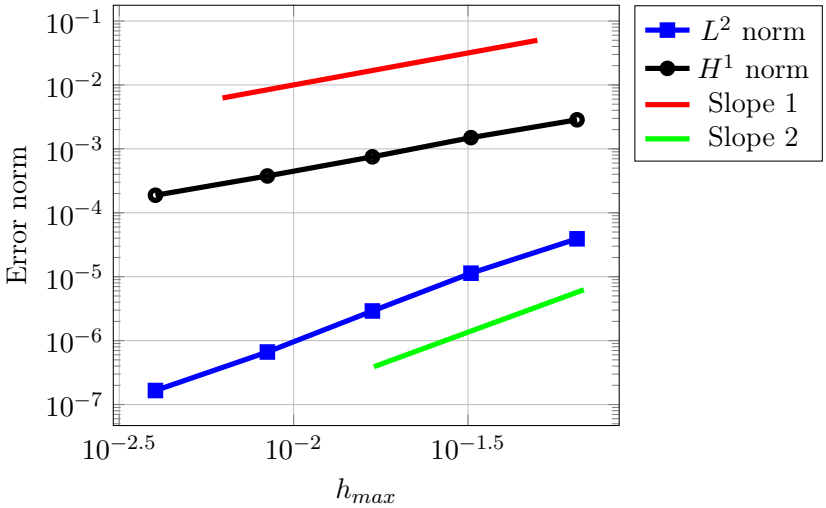


Figure II.3.4: Convergence test for active strain elastic model.

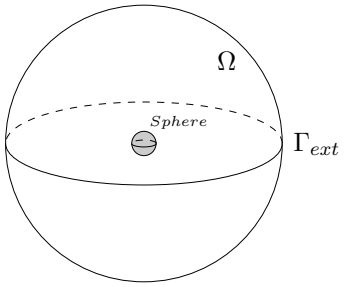


Figure II.3.5: Computational domain for the settling sphere example.

medium and is settling under the action of gravity. The following Stokes and settling velocity formulas are used as benchmarks for the fluid-rigid body coupling solver. The Stokes formula relates the drag force and the translational velocity of the sphere via the linear relationship $F = 6\pi\mu RV$, where R is the radius of the sphere, V its velocity and μ is the viscosity of the fluid. If we suppose that the sphere is settling under gravity and viscous dissipation has levelled out transitory effects, the drag force F will balance the gravity force and the settling velocity will be given by

$$V = \frac{2(\rho_s - \rho)gR^2}{9\mu}, \tag{II.3.2}$$

where ρ_s and ρ are the sphere and fluid densities, respectively, and g is the gravity acceleration.

The computational domain, represented in figure II.3.5, consists in a small solid sphere immersed in a fluid medium that is itself bounded by a spherical container. On the external boundaries of the fluid domain, homogeneous Dirichlet boundary conditions are imposed, while on the boundary of the sphere one has $u = U + \omega \times (x - x^{CM})$, i.e. the equality between the fluid velocity and the velocity of the solid. As gravity is the driving force of the motion, it is possible to evaluate both the settling speed and the drag force as products of the computations. The results are reported in Figure II.3.6, which shows the percentage error for the settling velocity (left figure) and the drag force (right figure) with respect to their theoretical values, when refining the domain discretization. In table II.3.2 we report the statistics of the mesh that were used, in terms of number of elements $N_{elements}$ and degrees of freedom for pressure N_p and velocity N_u. An analysis of the preconditioning strategies is also carried out on this toy problem by comparing LU preconditioning strategy to block preconditioning. In order to compare the two methods, we looked at the ratio between the average solution time of a single time-step using LU preconditioning, where the preconditioner is constructed at every time-step, or algebraic factorization with block preconditioning. We looked at two different block splitting strategies depending on the position of the rigid body degrees of freedom in the splitting. Table II.3.2, on the left, shows that, as the number of degrees of freedom grows, the block preconditioning approach can provide a faster solution when compared to LU preconditioning: this speed up depends on the preconditioning strategies for each individual block. For instance, block preconditioning where LU factorisation was used on the velocity blocks, was less efficient than LU factorisation on the whole matrix for mesh M_0 and M_1. Using algebraic multigrid to precondition the velocity block A_{00}, more adapted to the problem under consideration, gives faster solution on mesh M_1 and M_2 than the other two approaches. In both cases that we investigated, the degrees of freedom corresponding to rigid-body velocities are assigned to block A_{00} (cf. section II.2.5), and this splitting choice is denoted by the acronym FS_1. The $LU - Pmm - GAMG$ and $GAMG - Pmm - Jacobi$ acronyms refer to the preconditioning strategy for the two sub-blocks: LU or algebraic multigrid for the A_{00} sub-block, and Pmm with algebraic multigrid or $Jacobi$ preconditioning for

Mesh	N_u	N_p	$N_{elements}$	h_{max}
M_0	163773	7277	48340	2.22947
M_1	250350	11015	78867	1.96095
M_2	1044645	44096	298478	1.18534

Mesh	LU	$FS_1^{LU-Pmm-GAMG}$	$FS_1^{GAMG-Pmm-Jacobi}$
M_0	1	0.86	0.65
M_1	1	0.61	1.18
M_2	1	1.031	5.2

Mesh	LU	FS_1	FS_2
M_0	-	3	23
M_1	-	7	21
M_2	-	3	12

Table II.3.2: The top table reports the mesh characteristics for the falling sphere example. The bottom tables compare the preconditioning strategies when the problem is solved on $N = 24$ processors. On the left, numbers represent the ratios of average solution time t_{LU}/t_{prec}. The block preconditioning approach proves more effective than LU preconditioning with more refined meshes. We refer to the body of the text to describe the acronyms about the splitting strategies. On the right, the number of iterations of the Krylov subspace solver is reported. This table shows that the number of iterations does not depend on the mesh size, but it depends on the choice of the blocks: preconditioner FS_1 collects the fluid and rigid-body velocities in the same block, while preconditioner FS_2 assigns the rigid-body velocities to the block containing the pressure degrees of freedom.

the Schur's complement sub-block. Table II.3.2, on the right, shows that the number of Krylov iterations does not depend on the mesh size, but it depends on the type of block splitting. The number of iterations varies slightly as the mesh is refined, but it does not show an increasing trend. Strong scaling of the fluid-solid coupling is also tested in the case of multi-body swimmers (chapter IV.1), where further comparison between solution time for different problem sizes are presented.

II.3.3.2 Falling two-dimensional rigid bodies

The tests we propose here differ in two aspects from the previous one: they are two dimensional and the fluid regime has a higher Reynolds number. The first test is taken from [14, Section 3.3.3] and considers a two dimensional disk settling under the effect of gravity. In this test we compare the vertical position of the disk with the corresponding curve from [14]. Since we do not have an algorithm that handles contacts between the solid and the domain, the simulation is stopped when the cylinder reaches the bottom of the box; whence the representation of a shorter time window if compared with [14].

The computational domain is a $[0, 2] \times [0, 6]$cm box filled with fluid, which contains a disk of radius 0.125 cm and centered in $(1, 4)$. The fluid density is $\rho = 1\,\mathrm{g/cm^3}$ and the fluid viscosity is $\mu = 0.01\,\mathrm{cm^2/s}$, while the solid density is $\rho_s = 1.5\,\mathrm{g/cm^3}$. The value of gravity acceleration is $g = 980\,\mathrm{cm/s^2}$. For this simulation, the time step was $\Delta t = 10^{-3}$s and the non-linearity of Navier-Stokes equations was solved using the Newton method. Figure II.3.7, on the left, reports the velocity field at time $t = 0.1$s, and on the right the vertical coordinate of the disk's center of mass, which shows a good agreement with the results presented in [14]. In [14], the fluid-solid interaction was solved via immersed boundary method.

A second test of cylinder sedimentation is taken from [130]. The geometry of the testcase is the same as before: the geometrical parameters defining the computational domain, the cylinder and its initial position have the values we have previously reported. The fluid parameters, however,

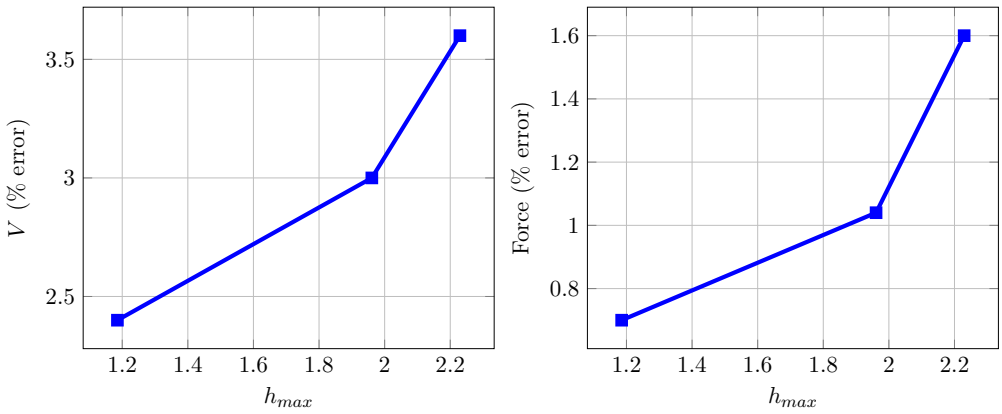


Figure II.3.6: Percentage error on the translational velocity of a sphere subject to gravity (left) and associated drag force (right). The exact values are computed via (II.3.2) where $\rho_s = 10^{-3}$, $\rho = 10^{-6}$, $g = 9.81$, $R = 0.1$, $\mu = 1$.

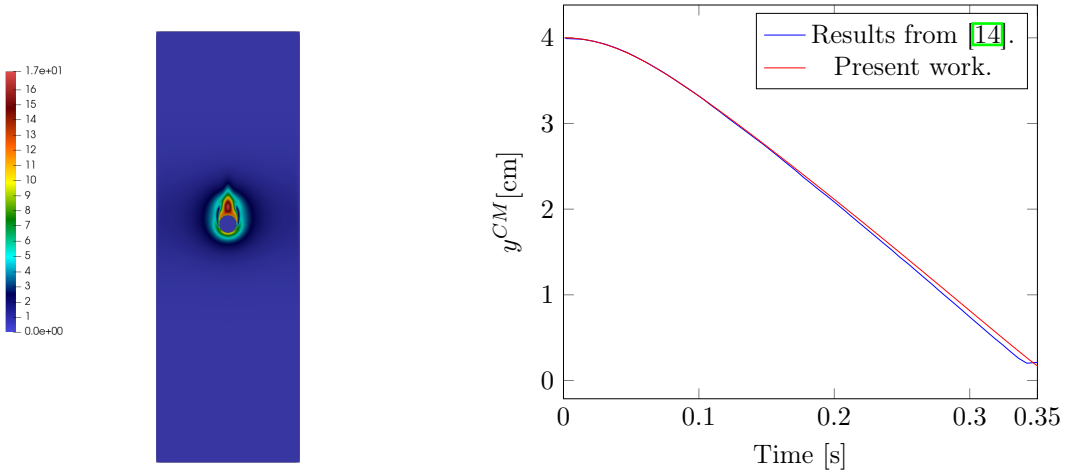


Figure II.3.7: In these figures we report the results relative to the first testcase of a falling two-dimensional cylinder. On the left, we show the velocity field at time $t = 0.1$ s. On the right, we present the comparison of our results with [14] on the vertical coordinate of the disk's center of mass. A good agreement between the two results is found.

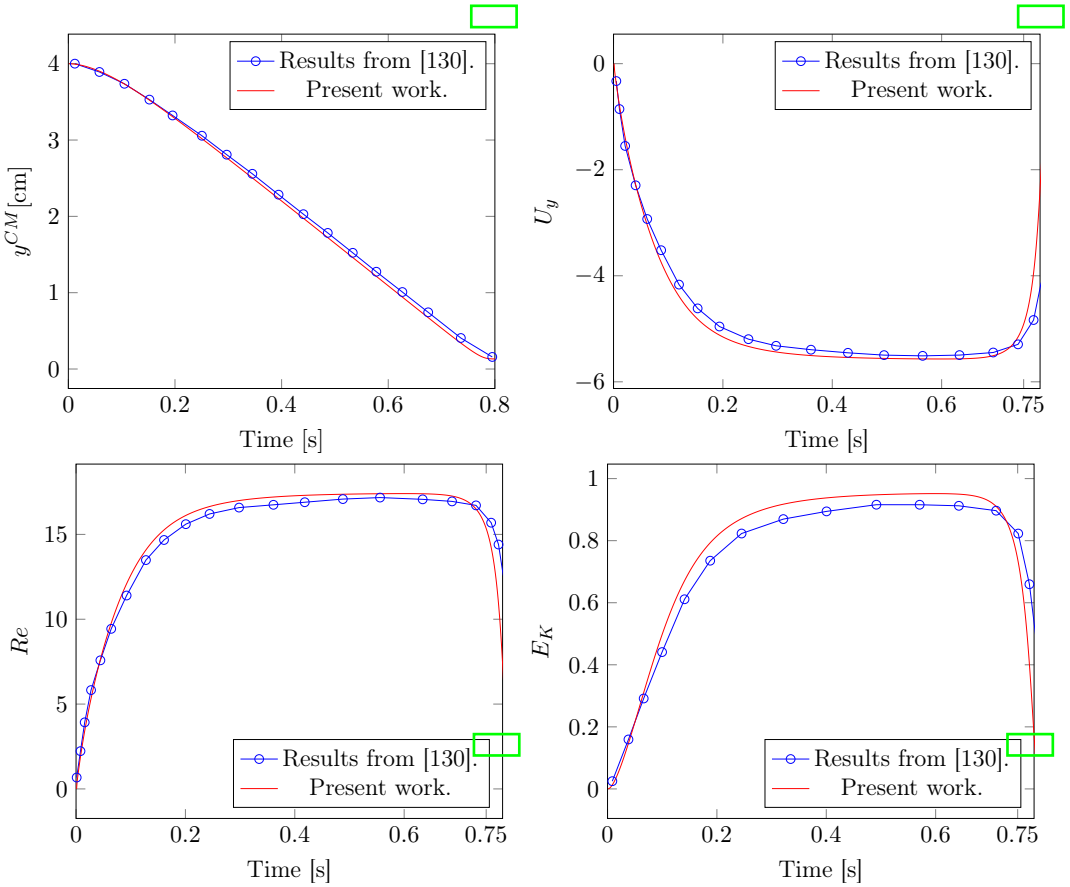


Figure II.3.8: In these figures we report the results of the second falling cylinder test. The blue lines correspond to Wan and Turek results in [130], while the red lines correspond to our results. In the top left figure we report the time evolution of the y coordinate of the center of mass of cylinder; in the top right the evolution of its vertical velocity U_y; in the bottom left figure the Reynolds number; in the bottom right the kinetic energy. The figures show a good agreement between our results and those contained in [130].

are different: the fluid density is $\rho = 1\,\mathrm{g/cm^3}$, the fluid viscosity is $\mu = 0.1\,\mathrm{cm^2/s}$ and the solid density is $\rho_s = 1.25\,\mathrm{g/cm^3}$. Also in this case, we used a time step of $\Delta t = 10^{-3}$s. In figure II.3.8 we compare our results to [130] and find a good agreement. In [130], the fictitious boundary method was employed to address the fluid-solid interaction, and the finite element method was used to solve the fluid equations. The Reynolds number is computed as $Re = \rho_s D\sqrt{U_x^2 + U_y^2}/\mu$, where $D = 2R = 0.25\,\mathrm{cm}$ is the diameter of the cylinder, and the kinetic energy is computed as $E_K = 0.5\rho_s \pi R^2 (U_x^2 + U_y^2)$.

A third test on the simulation of a falling body in a fluid consist in looking at the settling trajectory of an ellipse in a long channel. We compare our results to those in [134], which were obtained via a multi-block Lattice Boltzmann method and an ALE based finite element method. The fluid domain is a $[0, 0.4] \times [0, 2.8]$cm channel, in which an ellipse of axes 0.1cm and 0.05cm is located at $(0.2, 2.4)$. The initial configuration of the ellipse is also rotated of 45° with respect to the case where the long axis is parallel to the x axis. In this case, the viscosity of the fluid is $\mu = 0.01\,\mathrm{cm^2/s}$, the fluid density is $\rho = 1\,\mathrm{g/cm^3}$ and the solid density is $\rho_s = 1.1\,\mathrm{g/cm^3}$. Figure

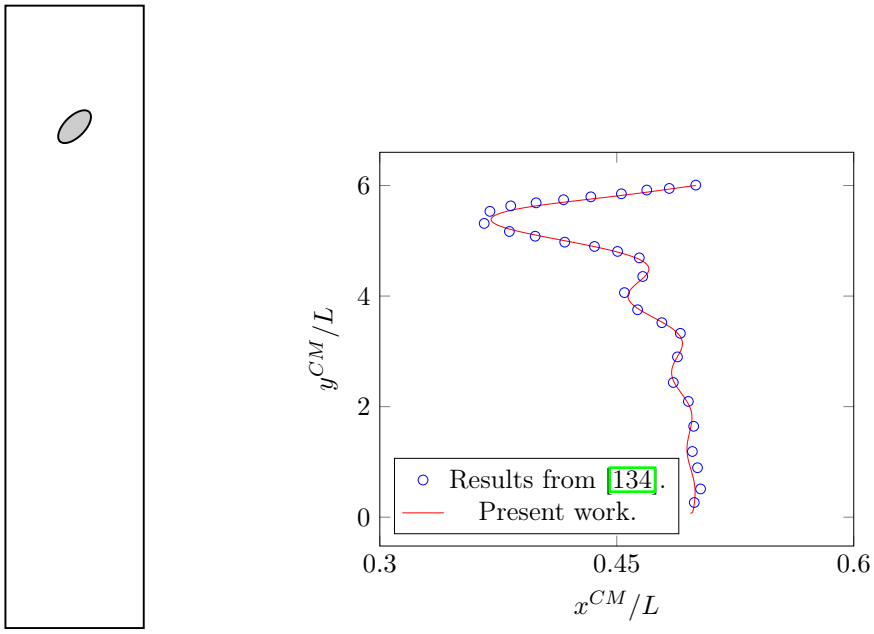


Figure II.3.9: On the left figure, the geometry of the problem is presented: at $t = 0$s the ellipse forms a 45° angle with the horizontal direction, and it is located in a long rectangular cavity. In the left figure, the x and y coordinates of the ellipse's center of mass are rescaled with respect to $L = 0.4\,\text{cm}$ and plotted one against the other. The resulting trajectory is in good agreement with the points from the benchmark case in [134].

II.3.9 shows the trajectory followed by the center of mass of the ellipse, scaled with respect to the characteristic size of the problem $L = 0.4\,\text{cm}$.

Part III

Implementation in *Feel++*

This part collects the software contributions to the *Feel++* library motivated by swimming simulation. The implementation of rigid body motion, remeshing and a generic computational model for swimmers, to be interfaced with the fluid solver, are presented.

Chapter III.1

Developments

The software contributions to the *Feel++* library are presented in this chapter: section III.1.1 describes the implementation details for the motion of immersed bodies and articulated swimmers; section III.1.2 discusses the computational aspects of mesh adaptation; section III.1.3 presents a generic application with which different swimmers can be simulated.

Within the *Feel++* framework it is possible to build and configure toolboxes that solve problems ranging from Newtonian fluid mechanics (Navier-Stokes and Stokes equations), non-linear elasticity to coupled fluid-structure interaction problems. These toolboxes are based on the *Feel++* core library, and handle initialisation of the data structures, solution via finite elements of the differential equations and post-processing of the results. Figure III.1.1 presents the inheritance diagram of the classes implementing the fluid, solid and fluid-structure interaction: first `ModelBase` handles the execution of the toolboxes at the lowest level (creating folders for data, interface for parallel computations); then, `ModelAlgebraic` and `ModelNumerical` contain the algebraic data structures that are necessary for the solution of the discrete problem, for the post-processing and time-stepping; `ModelPhysics` contains the data structures that characterise the physics of the problem (e.g. material properties, turbulence models). These classes are then extended by `FluidMechanics`, `SolidMechanics` or `FluidSolidInteraction`, that fill the general interface proposed by the previous classes with the specific discrete variational problem coming from the discretisation of the differential equations. The development originated by this thesis influenced mainly the `FluidMechanics` class, through the addition of data structures that handle moving bodies and remeshing.

Apart from the *Feel++* swimming framework, which is recent and open source, other software solutions already exist to simulate swimming. Using finite elements, but not only, the commercial COMSOL Multi-physics[1], is capable to solve fluid-structure interaction problems [109, 33]; via the open source CFD software OpenFOAM[2], which uses finite elements, finite volumes or Lagrangian particle tracking, it is possible to solve the fluid equations and interface a in-house solid solver to simulate the swimmer's motion; other options are softwares based on the immersed boundary method, like [15], or on libraries for boundary element method, like BEMLIB[3] for [57]. To our knowledge, ours is one of the few open source platforms for the simulation of (micro-)swimming.

[1] www.comsol.com
[2] www.openfoam.com
[3] http://dehesa.freeshell.org/BEMLIB/

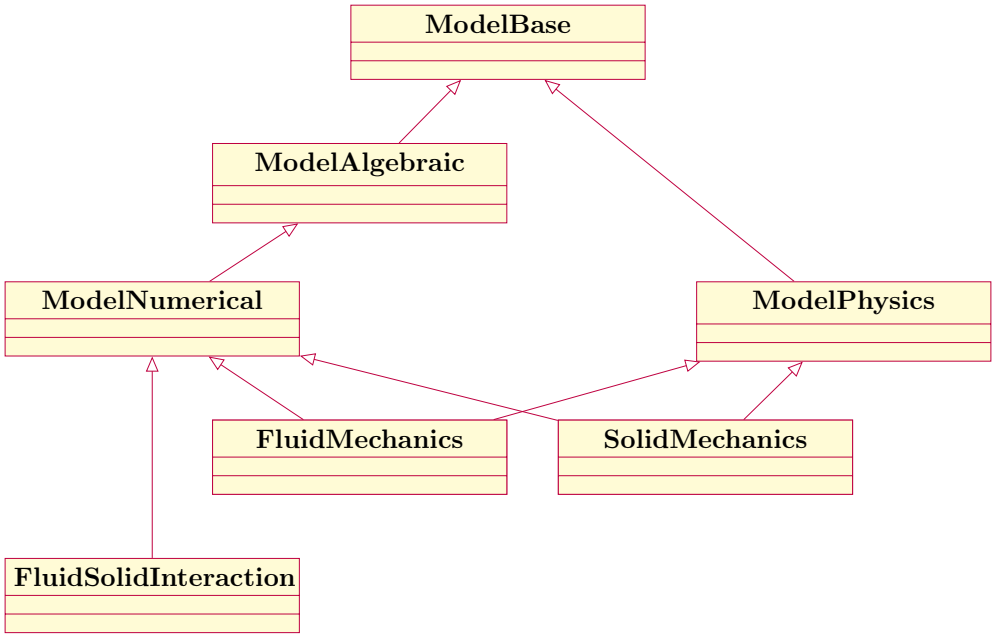


Figure III.1.1: UML diagram for the fluid, solid and fluid-solid interaction toolboxes.

III.1.1 Motion of immersed bodies

III.1.1.1 Rigid bodies

The motion of rigid bodies in a Newtonian fluid, as modelled in the context of this thesis, was added to the toolbox dedicated to fluid mechanics problems. A generic interface for an immersed rigid body is provided by the class `Body`, whose attributes are

- mass,
- center of mass,
- moment of inertia,
- Euler angles,
- rotation matrix $R(t)$ between the local and global reference frames

of the body, and other variables and labels identifying its particular physical model, i.e. "`body`". The methods of the class consist in accessors and mutators for the attributes, as well as routines that compute mass, center of mass and moment of inertia of the body. In Figure III.1.2 a synthetic view of the `Body` class is shown: apart from the attributes that were already mentioned, we highlighted the method expressing the moment of inertia J of the body in the laboratory reference frame as $R(t)JR(t)^T$.

The kinematics of immersed rigid bodies is encoded in the `BodyBoundaryCondition` class, that contains the translational and angular velocity on which the fluid velocity $u = U + \omega \times (x - x^{CM})$ at the surface of the body depends. In particular, `BodyBoundaryCondition` contains

- translational/angular velocity and the corresponding $\mathbb{P}^0$ finite element spaces,

- expressions that allow the imposition of $\omega \times (x - x^{CM})$ (since it's a space-dependent quantity that needs to be evaluated),
- routines that create the $\tilde{P}$ matrices which couple the rigid body velocity with the degrees of freedom of the fluid velocity (see section II.1.3).

In order to make this class extensible to an immersed elastic body, it is also possible to add a velocity field $u_d(X)$ such that the velocity at the interface of the fluid domain is $U + \omega \times (x - x^{CM}) + R(t)u_d \circ \mathcal{A}_t^{-1}(x)$.

Since the toolboxes are configured via `json` configuration files that prescribe material properties and boundary conditions of the specific problem, a collection of new keywords relative to the `Body` framework has been added to include rigid bodies. In particular, in the `Materials` section of the configuration file one specifies the volume markers and density value of the rigid body as

```
"Materials":{...,
  "RigidBody":{
    "markers": "VolumeMarkers",
    "physics": "body",
    "rho": "DensityValue"}
}
```

while in the `BoundaryConditions` section it is necessary to notify the presence of the rigid body via its surface markers and its material as

```
"BoundaryConditions":
{
  ...,
  "fluid":
  {
    "body":
    {
      "":
      {
        "markers":"SurfaceMarkers",
        "materials":"RigidBody"
        //  "translational-velocity":"{0.5,-1}"
        // , "angular-velocity":"0"
        // , "elastic-velocity":"{u_dX(x),u_dY(x)}"
      }
    }
  }
}
```

It is also possible to impose the translational or angular velocity of the body as a Dirichlet boundary condition, as well as use the `elastic-velocity` keyword to impose the velocity of deformation $u_d(X)$ for an elastic body. A synthetic version of the `BodyBoundaryCondition` is shown in figure III.1.2, collecting the attributes and routines that have been described.

A complete example of `json` configuration file for a rigid body, moving in a fluid, is now presented: it concerns the simulation of a sphere subject to gravity in the low Reynolds number regime, using Navier-Stokes equations. In this case, the computational domain is divided in two parts:

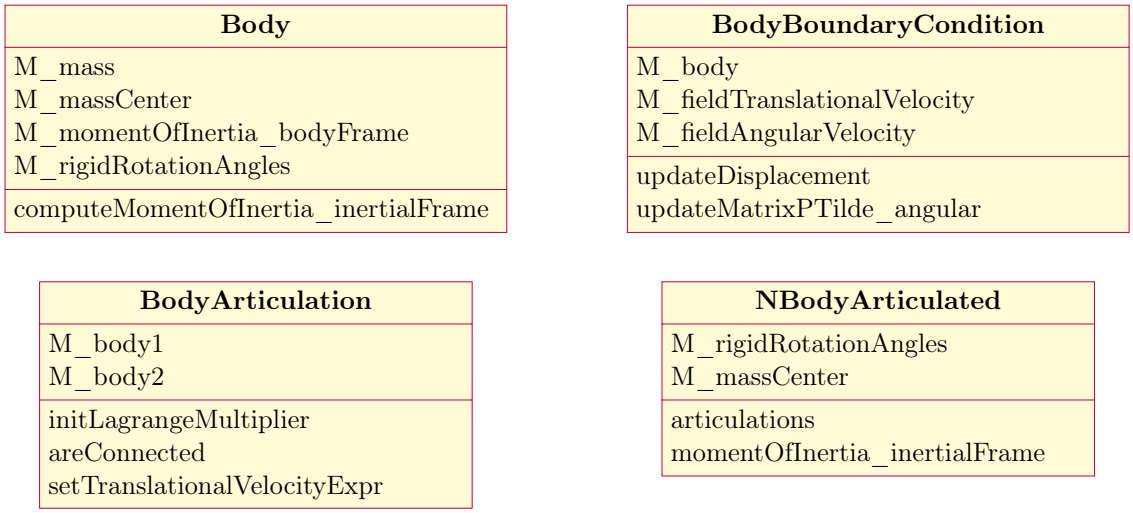


Figure III.1.2: Synthetic UML diagrams for the new classes that were added to `FluidMechanics`. The `Body` class collects the variables describing the position of the body and its dynamical properties, as well as the routines to update them. `BodyBoundaryCondition` contains the variables that describe the rigid and deformation velocity of the body, while `NBodyArticulated` allows to describe a body which is made of several `Body` components, connected and moving relatively to each other as prescribed in `BodyArticulation`.

the `Fluid` part, occupied by the fluid and the `SphereVol` part, occupied by the solid sphere. The interface between the two is named `SphereSurf`, and corresponds to the surface of the spherical object. The `Models` and `Materials` sections collect the information on physical models and the values of the physical coefficients; the section `BoundaryConditions` contains the information about the boundary conditions to be imposed on the differential problem; the `PostProcess` section specifies the fields and measurements to be exported along the simulation.

```
{
  "Name": "FallingSphere",
  "ShortName":"FallingSphere",
  "Models":
  {
    "equations":"Navier-Stokes"
  },
  "Materials":
  {
    "Fluid":{
      "markers":["Fluid"],
      "physics":"fluid",
      "rho":"1e-6",
      "mu":"1"
    },
    "Solid":{
      "markers":"SphereVol",
      "physics":"body",
      "rho":"1e-3"
```

```
      }
    },
    "BoundaryConditions":
    {
      "velocity":
      {
        "Dirichlet":
        {
          "Walls":
          {
            "expr":"{0,0,0}"
          },
          "Outlet":
          {
            "expr":"{0,0,0}''
          }
        }
      },
      "fluid":
      {
        "body":
        {
          "p1_mark":
          {
            "markers":["SphereSurf"],
            "materials":
            {
              "names":["Solid"]
            }
          }
        }
      }
    },
    "PostProcess":
    {
      "Exports":
      {
        "fields":["velocity","pressure","pid","displacement"]
      },
      "Measures":
      {
        "Forces":"Sphere"
      }
    }

  }
```

III.1.1.2 Articulated swimmers

In this thesis, an articulated swimmer is a multi-body object whose components are interconnected and can move relatively to each other. Following the formulation proposed in section II.2.3, the connection between two bodies will be defined in terms of their relative linear velocity. The value of the relative velocity as well as the pairs of bodies forming an articulation are encoded in the `BodyArticulation` class. Since the articulation is defined in terms of relative velocities, and these are prescribed on the boundary of the bodies, this class depends on the `BodyBoundaryCondition` classes of the two bodies that are connected via the articulation. Via a Lagrange multiplier field, defined over each pair of bodies, the relative velocity can be imposed for an arbitrary number of articulations; a second approach based on matrix $\tilde{P}$, as presented in II.2.3, can be chosen.

The articulation connection between two bodies is prescribed in the `json` configuration file in an additional block as

```
1  ‘‘BoundaryConditions":
2  {
3     ...,
4     ‘‘fluid":
5     {
6       ‘‘body":
7       {
8         ‘‘":
9         {
10          ‘‘markers":‘‘SurfaceMarkers1",
11          ‘‘materials":‘‘RigidBody1"
12
13        },
14      ‘‘":
15        {
16          ‘‘markers":‘‘SurfaceMarkers2",
17          ‘‘materials":‘‘RigidBody2",
18          "articulation":
19          {
20            "body":"SurfaceMarkers1",
21            "translational-velocity":"RelativeLinearVelocity"
22          }
23        }
24      }
25    }
26 }
```

The “master” body is defined as usual, by detailing its markers and materials. The “slave” body, on the other hand, is characterised by an additional `articulation` section which specifies the master body and the relative velocity between the two. In general, an articulated body can be composed of multiple articulations, and this information is stored in the class `NBodyArticulated` which is briefly presented in Figure III.1.2, together with `BodyArticulation`.

We provide an example of `json` configuration file for the simulation of an articulated three-sphere swimmer in the Stokes regime, realising the swimming gait proposed in [91], i.e. retraction and

extension of the joints between the spheres with a constant speed, in a non-reciprocal fashion.

```
{
  "Name": "three_sphere swimmer",
  "ShortName":"three_sphere swimmer",
  "Models":
  {
    "equations":"Stokes"
  },
  "Materials":
  {
    "Fluid":{
      "physics":"fluid",
      "rho":"1",
      "mu":"1"
    },
    "SphLeft":{
      "physics":"body",
      "rho":1e-1
    },
    "SphCent":{
      "physics":"body",
      "rho":1e-1
    },
    "SphRight":{
      "physics":"body",
      "rho":1e-1
    }
  },
  "Parameters":
  {
    "eps":1e-10
  },
  "BoundaryConditions":
  {
    "velocity":
    {
      "Dirichlet":
      {
        "BoxWalls":
        {
          "expr":"{0,0,0}"
        }
      }
    },
    "fluid":
    {
      "body":
      {
        "SphereCenter":
```

```
          {
            "markers":["SphereCenter"],
            "materials":
            {
              "names":["SphCent"]
            }
          },
          "SphereRight":
          {
            "markers":["SphereRight"],
            "materials":
            {
              "names":["SphRight"]
            },
            "articulation":
            {
              "body":"SphereCenter",
              "translational-velocity":"4*pulse(t,1-eps,2-eps,4)-4
*pulse(t,3-eps,4-eps,4):t:eps"
            }
          },
          "SphereLeft":
          {
            "markers":["SphereLeft"],
            "materials":
            {
              "names":["SphLeft"]
            },
            "articulation":
            {
              "body":"SphereCenter",
              "translational-velocity":"4*(pulse(t,0,1-eps,4)+
pulse(t,4-eps,4,4)) -4*pulse(t,2-eps,3-eps,4):t:eps"
            }
          }
        }
      }
    },
    "PostProcess":
    {
      "Exports":
      {
        "fields":["velocity","pressure","pid","displacement"]
      },
      "Measures":
      {
        "Quantities":
        {
          "names":"all"
```

```
96            }
97          }
98        }
99
100     }
```

This configuration file uses the `pulse(t,a,b,c)` function, which implements a square wave of period `c`, of unit value in [`a`,`b`], to prescribe the relative speed between the spheres.

Since multi-sphere articulated swimmers are often used as benchmarks or toy models, a tool for the simulation of N-spheres swimmer, with $3 \leq N \leq 10$ aligned spheres, has been implemented. Its purpose is the automatic creation of the configuration files for the fluid toolbox (`.cfg`, `.json`, `.geo`) and the configuration file for the block preconditioner (based on algebraic factorisation) required to launch the simulation of the N-spheres swimmer propagating a travelling wave. This tool is also available for the two dimensional version of these swimmers.

III.1.2 Mesh adaptation

Mesh adaptation and reconstruction were necessary to the numerical solution of swimming problems. In fact, working with conforming interfaces between the fluid and the moving solids may lead to important deformations of the discrete computational domain and mesh degradation. The *Feel++* library is interfaced with the *MMG* and *ParMMG* libraries in order to delegate sequential and parallel remeshing tasks. Interfacing the two libraries required:

- conversions between the *Feel++* and *MMG* mesh data structures (in sequential and parallel);
- creating a table that communicates the indices of mesh faces shared between two processors (in parallel);
- extensive testing of the mesh adaptation tools.

The conversion between the two data structures is ensured by the functions `mesh2Mmg` and `Mmg2mesh` of the class `Remesh`, defined in `feel/feelmesh/remesh.hpp`

mesh2Mmg converts the *Feel++* mesh structure into *MMG*'s mesh structure. The information transferred during sequential remeshing include the coordinates of mesh points, the table of elements and the markers of preserved geometrical entities. At first, if some facets were preserved and parallel remeshing was considered, the global facet index required by the *ParMMG* interface was computed using the Cantor pairing function. This function, $C = C(a, b)$, is a bijective function that, given a pair of integers a, b, outputs a unique integer of the form

$$C(a, b) = \frac{(a + b)(a + b + 1)}{2} + b.$$

In the three-dimensional case, the Cantor function was applied twice to get, from the nodes' indices a, b, c, the global face index $C(a, b, c) = C(C(a, b), c)$. However, the Cantor pairing function can lead to integer overflow, since $C(a, b, c)$ grows fast in three-dimensions. Hence, the global face index is no longer constructed and local indices are used. In the communication between two processors, instead of associating to each face on the processors' interface a global index, a table is constructed, pairing the local indices of the face on the respective processors. This table is then sorted depending on the rank of the current

processor (if the first processor has a smaller rank than the adjacent, the first column is sorted and the second accordingly in order to keep the pairings unvaried; if the rank of the first processor is larger, sorting is applied on the second column).

mmg2Mesh converts the *MMG* mesh structure into a *Feel++* mesh structure by defining the new geometrical entities.

`Remesh` contains other functions to access the remeshing functionalities in *Feel++*:

setMetric can be used to prescribe a metric function which imposes the local element size for the new mesh;

remesher creates the `Remesh` object from the initial mesh and the fixed geometrical entities (elements, edges). The syntax of this function is

Code III.1.1: Syntax of the function `remesher`

```
// r -> mesh data structure after remeshing
// mesh -> mesh data structure before remeshing
// marked_elements -> elements that are  fixed across remeshing
// marked_faces -> faces that are fixed across remeshing
auto r = remesher(mesh,marked_elements,marked_faces,{});
```

The last parameter of `remesher` is a second mesh structure related to `mesh`, and it is optional. If specified, `remesher` creates a relationship table $\mathcal{R}$ between the optional "parent" mesh and the "child" mesh, that is the result of remeshing procedure. This development was motivated by the need to evaluate expressions in the local reference frame of swimming bodies, as when an analytical expression for tail beating is available. As the mesh is reconstructed, the parent-child relationship maintains the connectivity of a subset of faces or elements previously identified, and allows the interpolation/evaluation of expressions on the parent mesh. The syntax of `remesher` with the optional parameter becomes

```
auto r=remesher(mesh,marked_elements,marked_faces,parent);
```

The relationship table is a data structure that was already employed in the creation of submesh structures, and it consists of a pointer to the parent mesh and a bidirectional map, containing the bijective correspondence between the parent and child geometric elements.

execute applies the remeshing procedure via the *MMG* remeshing routines.

As mesh quality assessment is the necessary step that initiates mesh reconstruction, three quality measures are proposed in the header `quality.hpp`: 2d Normalised Shape Ratio, q_{2D} (see (II.2.13)) and q_{3D} in three dimensions (see (II.2.14)) can be computed for each element in the mesh. Another aspect, complementary to `setMetric`, is the computation of the metric function, i.e. the local mesh size of the reconstructed mesh. In particular, it is possible to impose a mesh size that is proportional to the distance from the swimmer, this latter computed via the Fast Marching Method available in *Feel++*.

The remeshing tools are tested in the *Feel++* Testsuite, in `feelmesh/test_remesh.cpp`. Tests are realized in two-dimensional and three-dimensional geometries, both in sequential and parallel, to verify that interpolating a finite element field onto the newly constructed mesh does not introduce additional errors. This is verified across several remeshing steps and with different requirements on the mesh, like fixing the discretization of a boundary or of a volume (these tests are conducted on fixed domains). Other tests combining moving domains and mesh adaptation are performed: they are closer to the swimming problem and add the problems of time discretization and domain matching to the interpolation of fields. Tests were conducted in two

and three-dimensional geometries with and without moving rigid inclusions immersed in a fluid: fluid equations are solved at the same time and interpolation of fields across the remeshing steps was performed to continue the simulation. Interpolation of fields was performed in the case of an order 1 time scheme, for which the polynomial approximation of the time derivatives contains terms defined on the domain immediately prior to remeshing. In the case of higher order time schemes, the interpolation of previous values of the fields must be preceded by domain reversion. In presence of immersed bodies, the interpolation of the fluid quantities on the fluid subdomain is realised by an interpolation operator defined as in III.1.2

Code III.1.2: Interpolation operator on a portion of the domain

```
// op_vel -> interpolation operator for the fluid velocities
// fSpace1 -> fluid velocity function space  on the old mesh
// fSpace2 -> fluid velocity function space on the new mesh
// marked_elements2 -> marked elements in the new mesh where the fluid
  velocity is defined
// backend -> the options ensure that the operator is rebuilt from scratch at
   every remeshing step
auto op_vel = opInterpolation( _domainSpace=fSpace1,
_imageSpace=fSpace2,
_range=marked_elements2,
_backend=backend(_name="B",_rebuild=true));
```

In our tests we imposed a constant discretization of the moving boundary and volume of the moving body (when present), since it ensured constant values for the volume and inertia matrix. The case of several bodies in the fluid was also considered. Some examples of the geometries that were treated are collected in figure III.1.3. Figure III.1.4 presents an example of parallel remeshing of a cube.

III.1.3 Swimmer toolbox

Swimmers interact with the surrounding fluid and propel by inducing a velocity field on their surface and exchanging momentum with the surrounding medium. The specifics of each swimmer (gait, shape, internal dynamics) are not directly relevant to the coupling, but they are necessary to compute the surface velocity field. Following this reasoning, a general class that abstracts the "coupling" interface of a swimmer is created, which is then complemented by a system of plug-ins that allow to choose the swimmer model among the available ones.

In computational terms, this is done by using the concept of factory design pattern. Here, instead of calling a constructor to instantiate the generic `Swimmer` object, one calls a factory method which defers the instantiation to one of the subclasses which specifies the particular swimmer to be created. Simulations in chapter IV.2.1 are performed using this computational model. The interface of the `Swimmer` generic class contains the following functions:

updateState which updates the displacement and velocity fields of the swimmer, its center of mass and inertia tensor at the prescribed time instant.

stepper which specifies the time stepping scheme that is used to compute the displacement field from the velocity field of the swimmer.

Three subclasses are currently available:

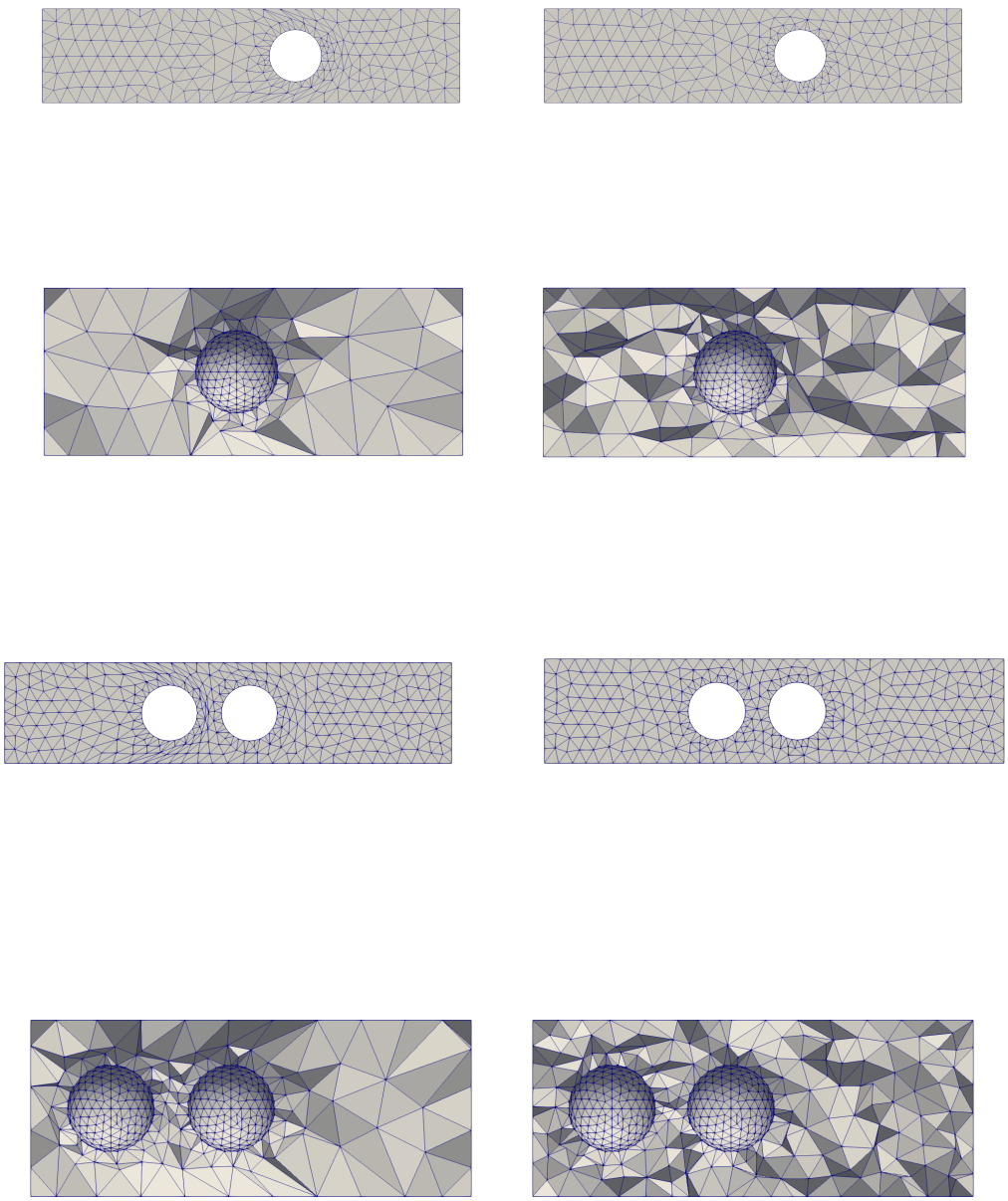


Figure III.1.3: Geometries that were considered in the remeshing testcases with moving objects. We report the two and three-dimensional rectangular domains with the corresponding hollow inclusions. On the left column, the triangulations before remeshing are presented. On the right, the results of the mesh adaptation.

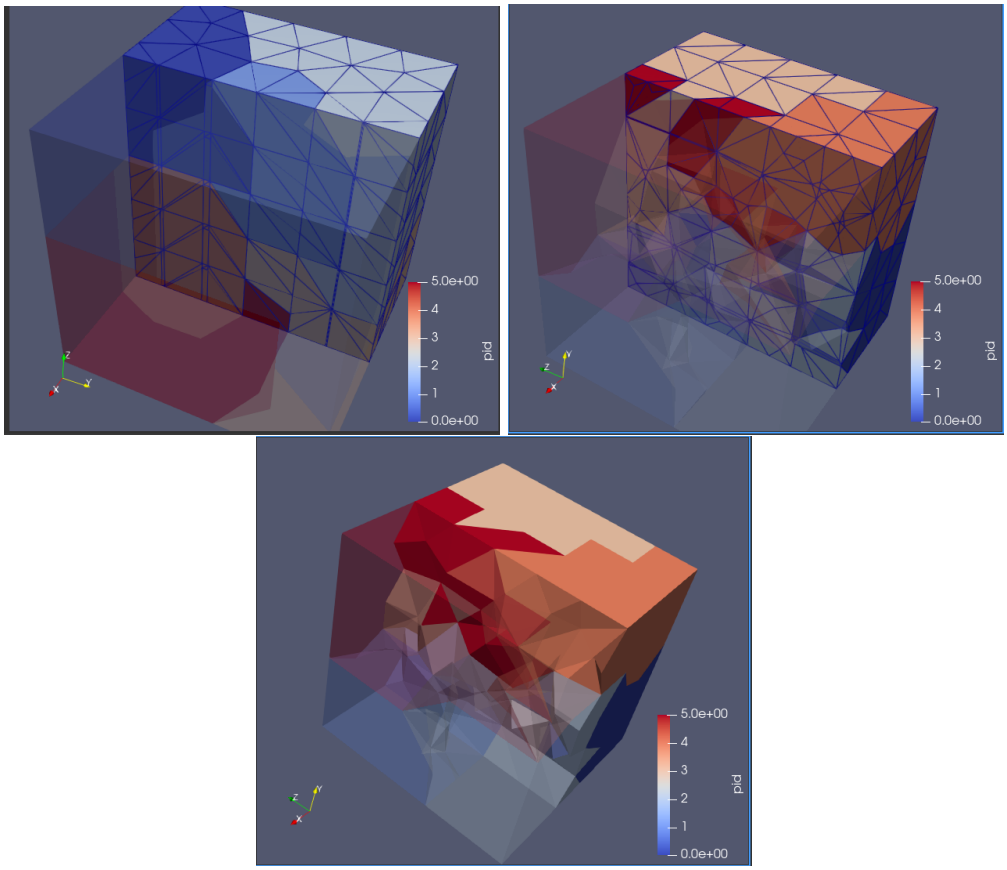


Figure III.1.4: Remeshing a cube in parallel using six processors. On the top left, a section of the initial mesh is presented, while on the top right the same section of the remeshed domain is shown. On the bottom, the partition of the mesh among the processors is displayed.

spermatozoon where the velocity of body deformation is prescribed via an analytical formula. This class computes the displacement field using a Runge-Kutta scheme. Its name derives from the two-dimensional tail beating of a spermatozoon model.

nature where the position of the swimmer's tail is described through a set of ODEs and the deformation velocity is computed via automatic differentiation. In this case, the position of the swimmer at $t = 0$ does not coincide with its reference position. Its name derives from the fact that the ODE model we consider comes from fitting biological observations.

active-elasticity where the displacement and velocity of the swimmer's body are computed using an active elasticity model.

The interfaces and methods of the swimmer subclasses are verified in a separate application, `gait`. In this application, mesh adaptation and remeshing occupy a fundamental part: swimmers' body deformation is subdivided into sub-steps that are alternated with mesh quality assessments and remeshing in order to avoid invalid elements. The number of sub-steps is fixed for the moment, but a future development might propose an adaptive strategy to establish the number of subdivisions. In fact, when returned elements or elements of extremely bad quality are detected after updating the geometry, mesh deformation should be reverted and the displacement step should be divided in two sub-steps. This procedure should be applied iteratively, until the whole displacement has been imposed. For a first order geometrical discretization, reverted elements can be detected by checking if the Jacobian of the transformation between the reference and current finite elements has negative determinant. If one is using a higher order geometric discretization, one should use Bezier polynomials as in [63].

The swimmer's domain is preserved across remeshing by fixing its elements and facets, while the relationship $\mathcal{R}$ between the reference and current domain is used to evaluate expressions that are defined in the swimmer's reference frame, as for the `spermatozoon` subclass. These tools were discussed in the previous chapter on mesh adaptation. We show in figure III.1.5 the beating tail of the `nature` subclass example, which required frequent remeshing. The colours correspond to different time instants of the deformation cycle (in chronological order, red, orange, yellow, green and blue).

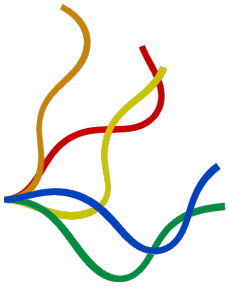


Figure III.1.5: Tail beating for the two-dimensional sperm cell described by the `nature` subclass. The colours correspond to different instants (in chronological order: red $t = 0\,\mathrm{ms}$, orange $t = 3\,\mathrm{ms}$, yellow $t = 10\,\mathrm{ms}$, green $t = 15\,\mathrm{ms}$ and blue $t = 19\,\mathrm{ms}$).

Part IV

Swimming computational experiments

This part is dedicated to numerical experiments on swimming. Different swimmers are considered: the three dimensional three-sphere swimmer; its extensions to multi-sphere swimmers and their two-dimensional analogues; an example on reinforcement learning for gait optimisation, applied to two-dimensional multi-sphere swimmers; two examples regarding sperm cells, in two and three dimensions. Below we describe in more detail each of these cases.

The first category of swimmers we considered in chapter IV.1 are collections of rigid bodies moving relatively to each other. In this family, we look at the three sphere swimmer and its extensions to four and five sphere swimmers. Here, the relative velocities between the spheres are prescribed and our results for the three-sphere swimmer are in accordance with the literature [91]. We also consider the two dimensional analogues of these swimmers and use Q-learning to recover their optimal swimming strategy.

Then, in chapter IV.2, we consider a two-dimensional model of a sperm cell. The problem is modelled as the solution of fluid equations in moving domain, where the body deformation is analytically known. In this case, since the propulsion is guaranteed by the predetermined body deformations, the internal dynamics (elastic behaviour, internal motors) of the body is completely neglected. A three-dimensional version of this same propulsion is presented. We then considered the motion produced by an asymmetric beating tail. In this case, the problem is formulated as the solution of fluid equations in moving domain, where the body deformation is numerically calculated via a system of ODEs. In this case, the ODEs model the position of a one-dimensional approximation of the spermatozoon's tail, which is then radially extended to be imposed on the two-dimensional tail of the spermatozoon. Also in this case the internal dynamics of the body is neglected, and the propulsion is guaranteed by the deformations arising from the computed displacement.

Chapter IV.1

Numerical experiments with rigid bodies

In this chapter we present several examples of articulated swimmers composed of collections of rigid bodies. In section IV.1.1, the modelling of the articulated multi-body swimmers we consider is briefly recalled; in section IV.1.2, the testcase of the three-sphere swimmer is presented, to validate our approach; in section IV.1.3, a four-sphere and five-sphere swimmers are simulated; in section IV.1.4, a scaling study is proposed to analyse the implementation of articulations with respect to different preconditioning strategies; in section IV.1.5, a Q-learning algorithm is applied to two dimensional examples of articulated swimmers to learn the optimal swimming strategy, and other two-dimensional instances of multi-circle swimmers are presented.

IV.1.1 Articulated swimmers

We define an articulated micro-swimmer as an object composed of n rigid bodies among which a reference body $\mathcal{S}_n$ is identified. The latter body is linked to all the other bodies $\mathcal{S}_i$, for $i \in \{1 \dots n-1\}$, by thin and hydro-dynamically negligible arms. The length of these links can be changed via "internal motors" that impose a relative speed between the bodies, leading to self-propulsion. In principle, each $\mathcal{S}_i$ can have a different shape, but we choose to work with spheres for benchmarking purposes. In fact, the motion of some multi-sphere swimmers can be analytically computed by using the appropriate Green kernel of Stokes equations [91].

The velocities U_i of bodies $\mathcal{S}_i$, $i = 1 \dots n-1$, are expressed as functions of U_n via constraints of the form

$$U_i = U_n + \mathbf{W}_{in}(t), \quad i = 1 \dots n-1, \tag{IV.1.1}$$

where $\mathbf{W}_{in}(t)$ represents the relative velocity between $\mathcal{S}_i$ and $\mathcal{S}_n$. The addition of these constraints to (II.1.40) completes the formulation of the swimming problem for the articulated swimmers. We notice that the resulting system is a particular instance of the general case presented in (II.1.28), where $u_d(t,x)$ is given by combining the constraints in (IV.1.1).

The formulation we just described applies directly to the three-sphere swimmer [91], an articulated swimmer composed of three aligned spheres. Here the reference body $\mathcal{S}_n = \mathcal{S}_3$ is the central sphere, which is connected by extensible arms to the other two spheres $\mathcal{S}_1$ and $\mathcal{S}_2$. The formulation can be applied as well to the planar three sphere swimmer [7] or to the four sphere

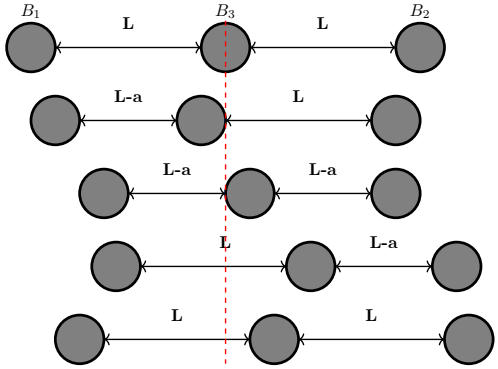


Figure IV.1.1: Representation of the three-sphere swimmer and its swimming gait. The gait is composed of four strokes in which one of the arms is alternatively retracted or elongated. The alternation guarantees the non-reciprocity of the motion.

swimmer [6], whose spherical bodies are initially placed on the vertices of an equilateral triangle and tetrahedron, respectively. In those cases, the relative velocity vectors $\mathbf{W}_{in}$ should be carefully computed, as each extensible arm connects $\mathcal{S}_i$ to the barycenter of the swimmer, and not to $\mathcal{S}_n$ directly as in the case of the three-sphere swimmer.

IV.1.2 The three-sphere micro-swimmer

The three-sphere micro-swimmer [91] is a three-dimensional swimmer composed of three aligned spheres having the same radius R. The two outer spheres are connected to the central one by extensible links, and the propulsion of the swimmer is ensured by changing the lengths of the connecting arms between two fixed values. The arm shrinkage is performed in a non-reversible fashion, in order to break the time-reversal symmetry of the Stokes equations, with a constant relative speed between the central sphere and the approaching one. For example, if the left arm is shrinking and the right arm keeps its length fixed, the translational speed U_3 and U_2 of the central and right sphere coincide, while the speed of the left sphere, moving with relative speed $\mathbf{W}_{13}$ with respect to the center sphere, will be $U_3 + \mathbf{W}_{13}$. The (non-reciprocal) stroke is composed of 4 steps, as shown in Figure IV.1.1, where the lengths of the two arms are alternately modified. A quantitative example of a swimming three-sphere swimmer is presented in [91]. Let us define R to be the radius of each sphere, $L = 10R$ the length of each link at its rest position and $a = 4R$ the maximal variation for the length of each arm (leaving a link length of $6R$ when the arms are completely retracted). The central sphere is displaced by $0.16R$ in the positive direction at the end of the 4-step stroke. In the first step, the travelled distance is $1.35R$ in the negative direction; in the second step, it is $1.44R$ in the positive direction; in the third step, it is $1.44R$ in the positive direction; in the fourth step, it is $1.35R$ in the negative direction.

Using the Lagrange multipliers formulation, we are able to recover the displacement at each step of the 4-step stroke reported in [91]. Figure IV.1.2 (left), translates the steps of the body deformation in terms of relative velocities between the central and lateral spheres. Figure IV.1.2 (right), represents the motion of B_3 during several repetitions of the 4-step stroke. The results were obtained using the library *Feel++* [22] and in particular the toolbox that solves Navier-Stokes equations in moving domain with immersed rigid bodies. The formulations based on Lagrange multipliers and $\mathcal{P}$ modification are implemented and available in *Feel++* Github

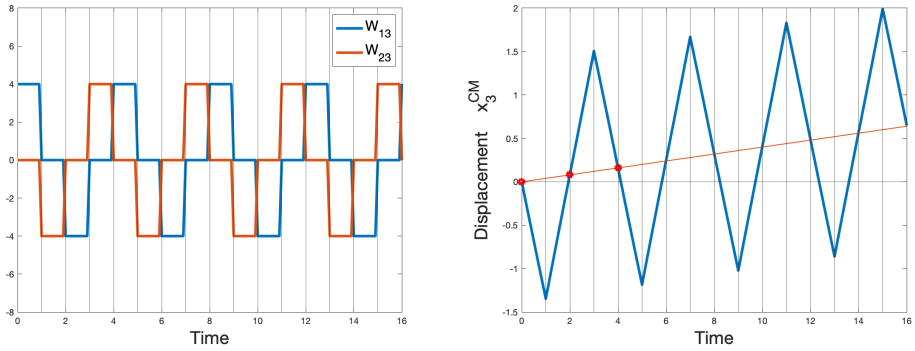


Figure IV.1.2: The left figure presents the relative speed $\mathbf{W}_{13}$, between the central and left sphere, and the relative speed $\mathbf{W}_{23}$, between the central and right sphere, as functions of time. The right figure shows, in blue, the position of the central sphere during the 4-step stroke. The red line intersects the trajectory of the central sphere at the red circles, which mark the $0.08R$ and $0.16R$ displacements predicted in [91] after 2 and 4 steps composing the swimming stroke.

repository [25], and can be used to reproduce the results in sequential and parallel.

IV.1.3 Four and five sphere swimmers

We now focus on the simulation of the $N = 4$ and $N = 5$ sphere micro-swimmers, and compare the swimming speed we obtain with that obtained by [124] using approximate analytical methods. In this case, during the swimming stroke all arms are active and their lengths follow a co-sinusoidal pattern of the form

$$L_i(t) = L + \lambda \cos(t + \phi_i) \quad \text{for } i = 1, \dots, N-1, \tag{IV.1.2}$$

where L is the average arm length and λ is the amplitude of the oscillations around L. In [124], the propulsion speed is computed as a sum of swimming contributions depending on all the possible sphere triplets one can form in a swimmer composed of N spheres. In our case, where the radii R of the spheres are of unit value and the average arm length is L, the propulsion speed depends on two dimensionless quantities, $\epsilon = \lambda/L$ and $\delta = 3R/2L$, and on the phase differences $\varphi_{ij} = \phi_i - \phi_j$. According to [124], the non-dimensional swimming speed of the four and five sphere swimmers are, respectively,

$$V_{4SS} = \epsilon^2 \delta \frac{7}{64} \left(\left(1 + \frac{41}{63}\right) \Big(\sin(\varphi_{12}) + \sin(\varphi_{23}) \Big) + 2\frac{41}{63} \sin(\varphi_{12} + \varphi_{23}) \right),$$

$$V_{5SS} = \frac{\epsilon^2 \delta}{50} \Big[(4a + b + c)(\sin\varphi_{12} + \sin\varphi_{34}) + (5a + 2b)\sin\varphi_{23} + (a + 2b + c)[\sin\varphi_{13} + \sin\varphi_{24}] + (a + 2c)\sin\varphi_{14} \Big],$$

where $a = 7/8$, $b = 41/18$ and $c = 151/72$.

Taking $L = 8$, $\phi_1 = 0, \phi_2 = \pi/3, \phi_3 = 2\pi/3$ in the four sphere case and $\phi_1 = 0, \phi_2 = \pi/4, \phi_3 = \pi/2, \phi_4 = 3\pi/4$ in the five sphere case, the expected average velocity values are

$$V_{4SS} = -1.2 \cdot 10^{-3}, \quad V_{5SS} = -2.1 \cdot 10^{-3}, \tag{IV.1.3}$$

N_{dof} $(u+p)$	Formulation	Preconditioner	N. iterations	Time
311000	Lagrange multipliers	LU	-	190s
311000	Lagrange multipliers	fieldsplit	41	33s
311000	Lagrange multipliers	gasm+LU	-	∞
311000	Matrix $\tilde{\mathcal{P}}$	LU	-	190s
311000	Matrix $\tilde{\mathcal{P}}$	fieldsplit	36	30s
311000	Matrix $\tilde{\mathcal{P}}$	gasm+LU	170	80s
888000 (2.85)	Lagrange multipliers	fieldsplit	50 (1.22)	110s (3.33)
888000 (2.85)	Lagrange multipliers	gasm+LU	-	∞
888000 (2.85)	Matrix $\tilde{\mathcal{P}}$	fieldsplit	43 (1.19)	97s (3.23)
888000 (2.85)	Matrix $\tilde{\mathcal{P}}$	gasm+LU	350(2.06)	505s (6.31)
2094000 (6.73)	Lagrange multipliers	fieldsplit	66 (1.60)	350s (10.6)
2094000 (6.73)	Lagrange multipliers	gasm+LU	-	∞
2094000 (6.73)	Matrix $\tilde{\mathcal{P}}$	fieldsplit	53 (1.47)	293s (9.76)
2094000 (6.73)	Matrix $\tilde{\mathcal{P}}$	gasm+LU	395(2.32)	2550s (31.87)

Table IV.1.1: Numerical study on the algebraic solution of the three-sphere swimmer problem. The simulation was launched on 8 cores, and "Time" is the runtime for one time iteration. For the three meshes, the average element size is 3, 2 and 1.5 respectively. For the second and third mesh, in parentheses, we reported the ratio of degrees of freedom, number of iterations and time with respect to the first mesh.

and the non-dimensional displacements after a complete stroke of period 2π are $X_{4SS} = \varepsilon^2 V_{4SS}$ and $X_{5SS} = \varepsilon^2 V_{5SS}$. Their dimensional counterparts are

$$X_{4SS} = \frac{2}{3\epsilon^2} V_{4SS} = -5.45 \cdot 10^{-2}, \quad X_{5SS} = \frac{2}{3\epsilon^2} V_{5SS} = -9.02 \cdot 10^{-2}, \tag{IV.1.4}$$

where the factor $2/3$ comes from inverting the non-dimensionalisation of δ, while $L\epsilon^2$ comes from the relationship between the non-dimensional velocity and displacement $V_{iSS} = \epsilon^2 X_{iSS}$ [124, eq. (4.1)], for $i = 4, 5$. After approximately 2π seconds, the average displacements that we computed by averaging the position of the centers of mass of the spheres are $X^{sim}_{4SS} \approx -5.51 \cdot 10^{-2}$ and $X^{sim}_{5SS} \approx -8.42 \cdot 10^{-2}$, whose values are similar to those computed in (IV.1.4).

IV.1.4 Scaling tests

We dedicate this section to study the efficiency of the different solution strategies on large problems involving multi-body swimmers. Table IV.1.1 presents the algebraic solution strategy and runtime per time iteration for the simulation of a three-sphere swimmer on 8 cores. It is shown that block preconditioning (fieldsplit) leads to faster solution and that the formulation based on $\tilde{\mathcal{P}}$ is slightly more efficient than the approach based on Lagrange multipliers as the mesh is refined. Block preconditioners are built using PETSc fieldsplit, and a comparison is performed with MUMPS LU preconditioning and PETSc $gasm + LU$ preconditioners based on Additive Schwarz domain decomposition with LU preconditioning on the sub-domains. We notice that the $gasm + LU$ preconditioner fails on the Lagrange Multiplier formulation while LU becomes very expensive as we refine the mesh. The results show that generic preconditioning strategies are not scaling well — or do not scale at all — in terms of iteration count or computing time. Our solution strategy, based on block preconditioning, is almost optimal in the sense that the number of iterations increases only slightly as the number of unknowns increases.

N_{proc}	1	2	4	8
$N_{elements}$	73682	73682	73682	73682
N_{dof}	310916	310916	310916	310916
Constructor	205s	131s	59.4s	21.8s
Speed-up constructor	1	1.57	3.45	9.38
Solution	1720s	940s	469s	277s
Speed-up solution	1	1.83	3.67	6.21
Post-process	2.14s	1.14s	0.76s	0.46s
Speed-up post-process	1	1.88	2.80	4.65
Time-step update	0.12s	0.39s	0.27s	0.20s
Speed-up time-step update	1	0.32	0.46	0.62
Time per step	1927.26s	1072.53s	529.43s	299.46s
Speed-up	1	1.799	3.64	6.43

Table IV.1.2: Scalability test on the three-sphere swimmer. The speed-up are computed as the ratio between the time required to accomplish each step on $N_{proc} = 1$ and the corresponding number of processors.

We propose also an analysis on the scaling properties of each part of the simulation, whose results are collected in table IV.1.2. *LU* preconditioning was used in this case. We notice that the time per step we report here is larger than the one shown in table IV.1.1, as the results were obtained on different machines. While the construction and solution parts scale well according to the number of processors, the post-processing part does not. This is due to the fact that disk writing in parallel does not scale perfectly. Parallel simulation of this testcase was made possible by a partition of the domain that assigned all the velocity degrees of freedom over the boundaries of the swimmer to a unique processor. This pre-processing allowed the Lagrange multipliers constraints to be easily imposed and avoided the communication of translational velocity values of the rigid bodies between processors.

IV.1.5 Two dimensional multi-sphere micro-swimmers

IV.1.5.1 Q-learning for multi-sphere micro-swimmers

We now turn our attention to two-dimensional swimmers as their behaviour is qualitatively similar to their three-dimensional counterparts and they are a less expensive alternative to investigate the behaviours of multi-body swimmers. In collaboration with Youssef Essousy, intern in Université de Strasbourg, a Q-learning algorithm was applied to two-dimensional multi-sphere micro-swimmers to "learn" the optimal swimming stroke. This study was inspired by [122], who studied the same problem in three dimensions and recovered the travelling wave motion as the optimal swimming stroke for a multi-sphere swimmer in a Newtonian fluid.

A learning problem is defined by means of its state space S, action spaces A_i (one for each state) and the reward function r. The state space S collects all the possible arm configurations that the N-sphere micro-swimmer can take: since each arm can have two length values, the state space is discrete and isomorphic to $\{0,1\}^{N-1}$. The action space A_i contains the actions that are accessible from state $s_i \in S$ and that modify the length value of one arm only: the cardinality of A_i is directly proportional to the number of spheres. The reward function at step n is the sum of the swimmer's displacements δ_i realised after the i-th action, for $i = 1, \dots, n$. In particular,

the reward function we consider is $r_n = \sum_{i=1}^{n} \delta_i \cdot e_1$, i.e. the cumulative displacement in the direction e_1, whose purpose is selecting the propulsion strategy ensuring the largest displacement in the desired direction. The last element of the Q-learning algorithm is the Q-matrix, which collects information about the reward for previous state-action pairs, and is an approximation of the optimal action-value function [131]. The Q-learning algorithm 8 updates the Q-matrix Q according to the rule

$$Q(s_n, a_n) \leftarrow Q(s_n, a_n) + \alpha[r_n + \gamma \max_{a_{n+1}} Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)], \tag{IV.1.5}$$

where $0 \leq \alpha \leq 1$ is a constant defining the learning rate and $0 \leq \gamma \leq 1$ weights the influence of the new value $Q(s_{n+1}, a_{n+1})$ on the update of the Q-matrix. A variant of the Q-learning

Algorithm 8: Q-learning algorithm

Input: s_0 initial state; set $Q(s, a)$ to 0
  for i=1 **to** N_{learn} **do**
    Choose a_i using the policy given by Q
    Take action a_i, observe the new displacement δ_i and state s_i
    Update Q via (IV.1.5)
  end for
Output: Q, r

algorithm, namely double Q-learning (see Algorithm 9), appears to learn the optimal swimming strategy faster than the classical version. This algorithm uses two Q matrices, Q_A and Q_B, that are randomly updated and used to choose the next action. The idea behind this approach is that $Q_A(s', a^*)$ can be considered as an unbiased estimate for the value of action a^* in the Q_B function, as the two learn from different experience samples [53]. We apply Q-learning to the

Algorithm 9: Double Q-learning algorithm

Input: s_0 initial state; set $Q_A(s, a), Q_B(s, a)$ to 0
  for i=1 **to** N_{learn} **do**
    Choose a_i using the policy given by Q_A, Q_B
    Take action a_i, observe the new displacement δ_i and state s_i
    Choose (randomly) to update Q_A or Q_B
    if update Q_A **then**
      Let $a^* = \arg\max_a(Q_A)(s_i, a)$
      Update $Q_A(s, a) \leftarrow Q_A(s, a) + \alpha[r_n + \gamma Q_B(s_i, a^*) - Q_A(s, a)]$
    end if
    if update Q_B **then**
      Let $a^* = \arg\max_a(Q_B)(s_i, a)$
      Update $Q_B(s, a) \leftarrow Q_B(s, a) + \alpha[r_n + \gamma Q_A(s_i, a^*) - Q_B(s, a)]$
    end if
    $s \leftarrow s_i$
  end for
Output: Q_A, Q_B, r

N-sphere micro-swimmers with $3 \leq N \leq 5$, and in the three cases the optimal swimming policy, namely the travelling wave, is found. The transitions between two states are simulated using *Feel++* and the articulated swimmer framework we have developed. Figure IV.1.3 shows the

learning process for the three swimmers under study. In the $N = 3$ case, as the travelling wave policy is the only strategy allowing a forward propulsion, the swimmer is not able to advance until this policy is learnt. In the first iterations, the swimmer oscillates around its initial position; then, once the travelling wave strategy is learnt, the cumulative reward (that is the motion in the desired direction) increases steadily. Even if the swimming policy is learnt, there are still actions which can be taken at random via the ϵ-greedy scheme, in order to ensure a thorough exploration of the state-action space. These random choices are responsible of the steps where the cumulative reward does not increase in the linear part of the graphs. In the $N = 4$ and $N = 5$ cases, the state-action space is larger and several strategies ensure forward propulsion, as one can see from the different slopes in the corresponding figures. Learning the travelling wave strategy is harder in these instances, and many more learning steps are required. In order to be sure that the policy being learnt is the optimal one, the Q-learning algorithm is run several times with a number of learning steps N_{learn} increasing with the number of spheres, and the results are compared.

Remark. The Q-learning algorithm converges when the underlying dynamics has the Markov property, meaning that state s_{n+1} only depends on the previous state and action pair (s_n, a_n) and not on the whole history of the process. Due to the Markov property, we were able to reduce the computational cost of Q-learning by storing the displacement δ_n that resulted from choosing action a_n when the system started from state s_n. This implies that every transition between two states needs to be simulated only once via finite elements. Simulating these transitions, when using 8 processors, takes few seconds with *Feel++*.

Q-learning was also applied to study a two dimensional three-sphere swimmer which could vary its arm lengths among a finite set of possibilities, meaning that it could choose to shrink its arms of $\varepsilon \in \{1, 2, 3, 4\}$. In this case, Q-learning recovered a travelling wave as optimal policy and $\varepsilon = 4$. This result was expected as it is in line with the quadratic dependence of the three-sphere swimmer's velocity on ε [91]. Figure IV.1.4 reports these results: on the left, a first simplified case considered only two values of $\varepsilon \in \{2, 4\}$. The different slopes in the figure show that different swimming strategies are available even in this simplified case, but finally the travelling wave with $\varepsilon = 4$ is recovered. The same result is obtained when considering $\varepsilon \in \{1, 2, 3, 4\}$, but, as before, the number of learning steps must be increased in order to ensure the thorough exploration of the state-action space.

IV.1.5.2 More three-sphere swimmers

Other variations of two dimensional three-sphere swimmers were studied: first, we considered three sphere swimmers whose arms are not aligned and which are able to move in circles, as [77] did in three dimensions; then, we looked at a three sphere swimmer in a shear flow and observe its long-time behaviour.

In the first set of swimmers, two configurations are chosen: one where the arms form an angle of $\beta = 53°$ and one where the arms form an angle $\beta = 140°$. These two values of β were chosen, among those considered in [77], to simulate a case of acute and obtuse β. Moreover, it was possible to visually compare the radii of the corresponding trajectories and easily estimate the size of the computational domain to perform a long time simulation of their circular trajectory. The three circles are of unit radius, and the imposed motion makes swimmers retract and extend their arms between $L = 9$ and $L = 10$ alternately. In both cases the swimmer moves in counter-clockwise direction, and the circle described by the $\beta = 140°$ has a smaller radius than the

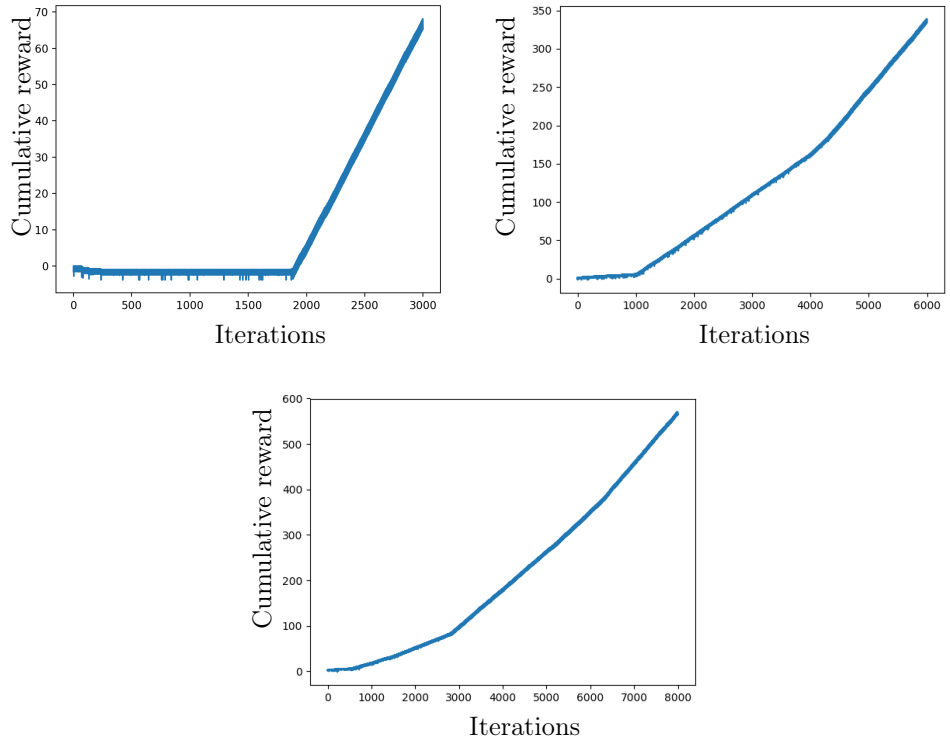


Figure IV.1.3: Learning the optimal policy via Q-learning, for different values of N. From the top left $N = 3$, $N = 4$ and $N = 5$.

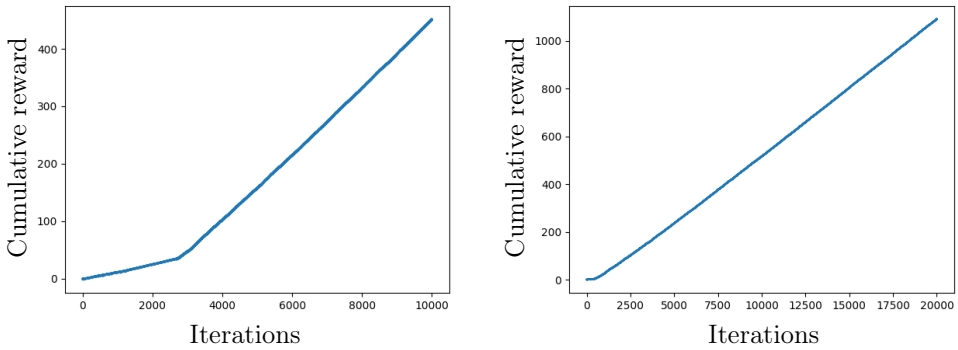


Figure IV.1.4: Learning the optimal arm extension via Q-learning. On the left, only two values of arm extensions were available (2 and 4); on the right, four values of arm extensions were available (1, 2, 3 and 4).

one described by the $\beta = 53°$ one. Figure IV.1.5 presents the trajectories of the center of mass of the swimmers. Only a short portion of the trajectory was simulated, and the different "orientation" of the arcs of circle described by the two swimmers is due to the different tangent to their circular trajectory at $t = 0$. Using these data, we computed the average radius of the trajectories via least squares and obtained that $R_{\beta=53°} \approx 0.49$ is larger than $R_{\beta=140°} \approx 0.32$, which is in qualitative agreement with the results obtained in three dimensions by [77]. In these cases, the minimal mesh quality threshold was fixed to 0.4 and it was computed with q_{2D} (see (II.2.13)).

The three sphere swimmer in shear flow is composed of three circles of unit radius separated by arms of constant length $L = 10$. The central circle is not allowed to translate and the swimmer performs a rotational motion around the fixed central sphere, caused by the interaction of the imposed shear flow with the swimmer. The shear flow is imposed by prescribing a shear velocity $u_{shear}(x, t)$ that increases linearly from $-u_{max}$ (on the bottom of the computational domain) to u_{max} (on the top of the computational domain). In correspondence of $y = 0$, the shear velocity is zero. The angular speed decreases as the swimmer reaches the horizontal configuration (as shown in figure IV.1.6, on the left), as this position is a stable equilibrium for the system when the radii of the circles are small and the swimmer can be approximated by an elongated rod [120]. However, the equilibrium is not found as the circles we consider are not of negligible radius. Along the simulation, the length of the arms does not vary and the outer spheres perform circular trajectories around the origin, as shown in figure IV.1.6 on the right, for the left sphere.

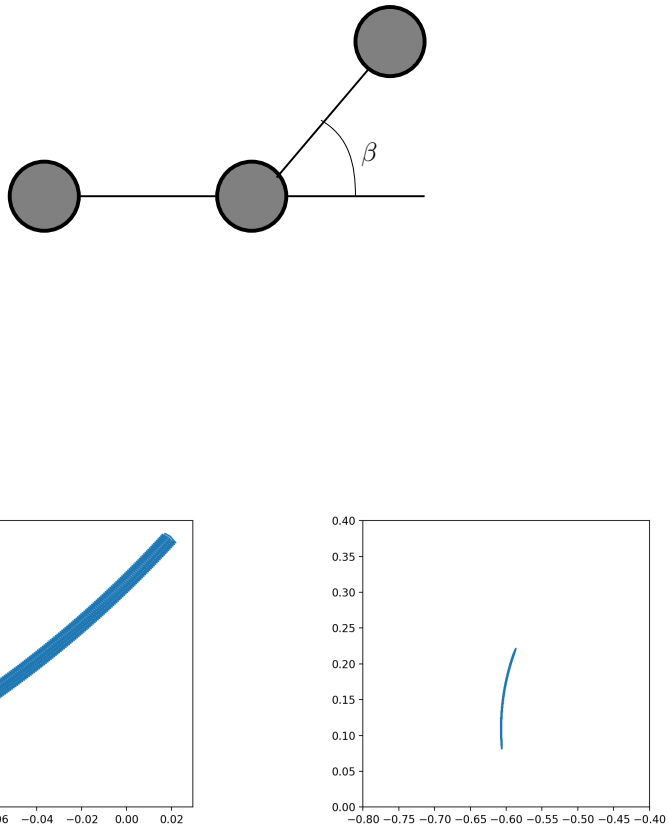


Figure IV.1.5: On the top, schematic representation of the three circle swimmer with non-aligned arms. On the bottom left, position of the center of mass of the three circle swimmers with $\beta = 53°$. On the bottom right, position of the center of mass of the three circle swimmer with $\beta = 140°$. A least square estimation of the average radii of the trajectories reveals that the circle described by the $\beta = 53°$ swimmer has a larger radius than the circle described by the $\beta = 140°$ swimmer, which is in qualitative agreement with [77].

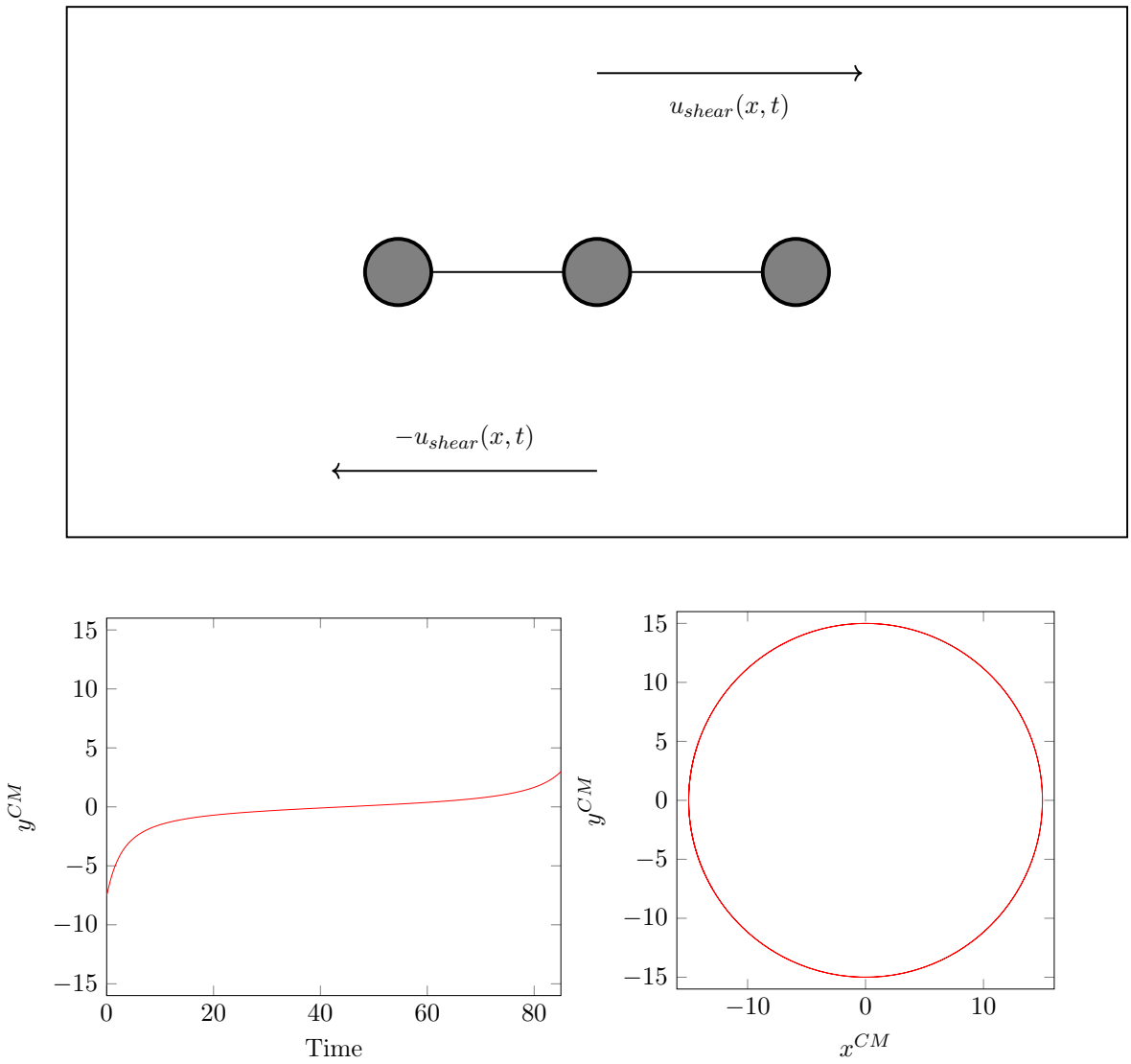


Figure IV.1.6: On the top, the representation of the three circle swimmer in shear flow. The shear velocity $u_{shear}(x,t)$ increases linearly from the bottom towards the top of the computational domain. On the bottom left, the y coordinate of the center of mass of the left sphere: close to the horizontal configuration ($y = 0$), the slope of the curve decreases as the system is attracted towards the equilibrium position. On the bottom right, trajectory of the center of mass of the left sphere: the trajectory is a circle around the center of mass of the central sphere, i.e. the origin of the coordinate system.

Chapter IV.2

Numerical experiments with prescribed deformation

In this chapter, two examples of swimming sperm cells are presented: section IV.2.1 is dedicated to the simulation of a two dimensional spermatozoon with prescribed sinusoidal beating of the flagellum; section IV.2.2 is focused on the simulation of a two-dimensional sperm cell with asymmetric prescribed beating.

IV.2.1 Sperm cell with planar beating of the flagellum

In this first section, we focus on a two-dimensional model of a sperm cell swimming in a channel, inspired by the study on sperm cell motion realised in [109]. In this case, the analytical expression of the tail's deformation is known in advance, and the fluid problem can be solved for velocity u and pressure p. Thanks to the coupling algorithm of section II.2, the translational and angular velocities of the swimmer, U and ω, can be computed at the same time.

The computational domain is represented in figure IV.2.1, and it consists in a two-dimensional rectangular channel Ω in which the sperm cell moves. The deformation of the swimmer's tail is such that the cell, on average, moves only in the horizontal direction. In order to solve the fluid

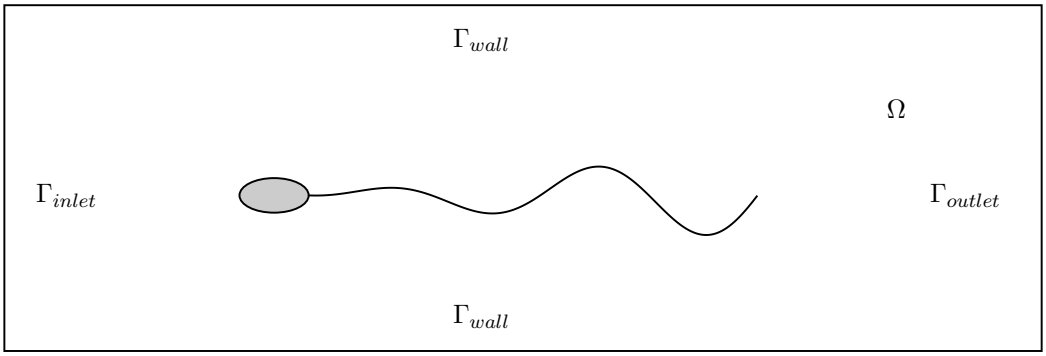


Figure IV.2.1: Schematic representation of the computational domain for the two-dimensional sperm cell example.

Fluid		Geometry		u_d		
density	viscosity	a semi-axis	b semi-axis	λ	T	L
$1000\,\mathrm{kg/m^3}$	$0.001\,\mathrm{Pa\,s}$	$2\,\mu\mathrm{m}$	$1\,\mu\mathrm{m}$	$20\,\mu\mathrm{m}$	$0.3\,\mathrm{s}$	$50\,\mu\mathrm{m}$

Table IV.2.1: In this table we report the values needed for the reproduction of the two-dimensional sperm cell example.

problem, homogeneous Dirichlet boundary conditions are imposed on the channel's walls (top and bottom boundaries Γ_{wall}) and homogeneous Neumann boundary conditions are imposed on the inlet Γ_{inlet} and outlet Γ_{outlet} of the channel. The fluid-solid coupling condition imposes that, on the head boundary, $u = U + \omega \times (x - x^{CM})$, where x^{CM} is the swimmer's center of mass in the fluid reference frame. On the tail boundary $u = U + \omega \times (x - x^{CM}) + R(t)u_d \circ \mathcal{A}_t^{-1}(t, x)$ is imposed, where $u_d(t, X)$ is the elastic velocity in the swimmer's reference frame and it has the form

$$u_d(t, X) = \begin{bmatrix} \frac{2\pi}{4T}[A_{max}(X - X_j)/L]^2 \frac{2\pi}{\lambda} \cos(4\pi(\frac{t}{T} - \frac{X}{\lambda})) \\ \frac{2\pi}{T}[A_{max}(X - X_j)/L] \cos(2\pi(\frac{t}{T} - \frac{X}{\lambda})) \end{bmatrix}. \tag{IV.2.1}$$

Equation (IV.2.1) represents the velocity of a sinusoidal wave of linearly increasing amplitude, propagating from the head of the swimmer to the tip of its tail. While the y-component of u_d comes from the time derivative of the sinusoidal wave

$$Y = \frac{A_{max}(X - X_j)}{L} \sin\left(2\pi\left(\frac{t}{T} - \frac{X}{\lambda}\right)\right), \tag{IV.2.2}$$

the x-component comes from a constraint of inextensibility on the sperm tail [121]. In the previous equations, T is the period of the wave, λ is the wavelength of the wave, L is the length of the tail, X_j is the position of the head-tail junction, A_m is the maximal amplitude measured at the flagellum's distal end. The evaluation of (IV.2.1) in the swimmer's reference frame motivated the development of the relationship $\mathcal{R}$ between meshes (see section II.2.2): thanks to the fact that the discretization of the swimmer's boundary is left unchanged by the mesh adaptation algorithm, it is possible to associate to each node of the deformed boundary its initial coordinates.

We report in Table IV.2.1 the values of the parameters in the expression of u_d, the fluid properties and the sperm geometric parameters. We consider two values for the channel half-height, namely $7.5\,\mu\mathrm{m}$ and $15\,\mu\mathrm{m}$, and compare our results to [109], who used the software COMSOL for the swimming simulations. In particular, in figure IV.2.2 we report the comparison between the propulsion velocities we computed and those found in [109]. We notice that, in the case of the smaller channel half-height and small amplitudes, our values correspond to the results from the literature; larger amplitudes, instead, do not follow the results of [109]. In the case of a larger channel height, the propulsion speed we compute is smaller than the reference value, but it still shows a linearly increasing behaviour. The discrepancy between our results and [109] is not easy to be explained: the fact that we can reproduce only partially the results at small amplitudes, and both overestimate and underestimate the speed at larger amplitudes, does not show a clear path towards the solution. A parameter that was not specified in [109] was the thickness of the flagellum, as well as the way to extend the motion of the tail's centerline (if the flagellum had a finite thickness) to the whole surface. These two aspects might have an impact on the fluid-swimmer interaction and its swimming speed. Other aspects that might justify (at least partially) the discrepancy are different mesh sizes and time steps, that were not specified in [109].

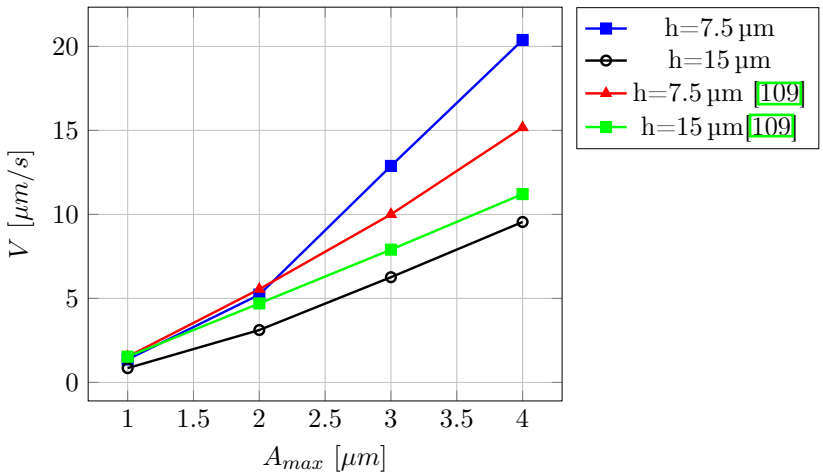


Figure IV.2.2: Comparison of swimming speeds for a 2D swimming spermatozoon in different channels. Blue and black symbols correspond to our results in the case of $h = 7.5\,\mu m$ and $h = 15\,\mu m$ channels, while red and green symbols correspond to results obtained in [109, Fig.7b].

It is possible to simulate the same oscillatory swimming in three dimensions, just by changing few parameters in the configuration of the model, and by extending u_d to 0 for the z coordinate. In figure IV.2.3 we show few time instants of the simulation of a three-dimensional sperm cell, whose beating flagellum propagates a sinusoidal wave of linearly increasing amplitude. The wave is restricted to the xy plane and its maximal amplitude is $A_{max} = 4\,\mu m$. Figure IV.2.4 shows a section of the fluid field and the velocity magnitude on two different sections.

IV.2.2 Asymmetric beating of the flagellum

In experimental studies on spermatozoa, researchers can track the tail's position during a beating period thanks to fast cameras and tail staining [45, 62]. As the filament is assumed to have negligible thickness, fitting of the experimental data provides ODE models for the filament kinematics. In [62], the tail beating was recorded and the set of ODEs

$$\begin{cases} \dfrac{d\mathbf{r}}{dl} = -\mathbf{e}_3, \\ \dfrac{d\mathbf{e}_1}{dl} = -\kappa_f \mathbf{e}_3 + \tau_f \mathbf{e}_2, \\ \dfrac{d\mathbf{e}_2}{dl} = -\tau_f \mathbf{e}_1, \\ \dfrac{d\mathbf{e}_3}{dl} = \kappa_f \mathbf{e}_1. \end{cases} \tag{IV.2.3}$$

was proposed to represent it, where $l \in [0, L]$. At all time instants, the solution to these ODEs provides the tail shape, via the position vector $\mathbf{r}$, and the orientation of the local reference frame $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. The vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ describe the local (Cosserat) reference frame of the swimmer's tail: $\mathbf{e}_3$ defines the tail's tangent direction and $\mathbf{e}_1, \mathbf{e}_2$ span the normal plane to the tail's centerline. These ODEs depend on two parameters: a constant twist parameter $\tau_f > 0$ and a mean curvature parameter K_0, contributing to the definition of the curvature

$$\kappa_f(l, t) = K_0 + B\cos(\omega_0 t - \lambda l).$$

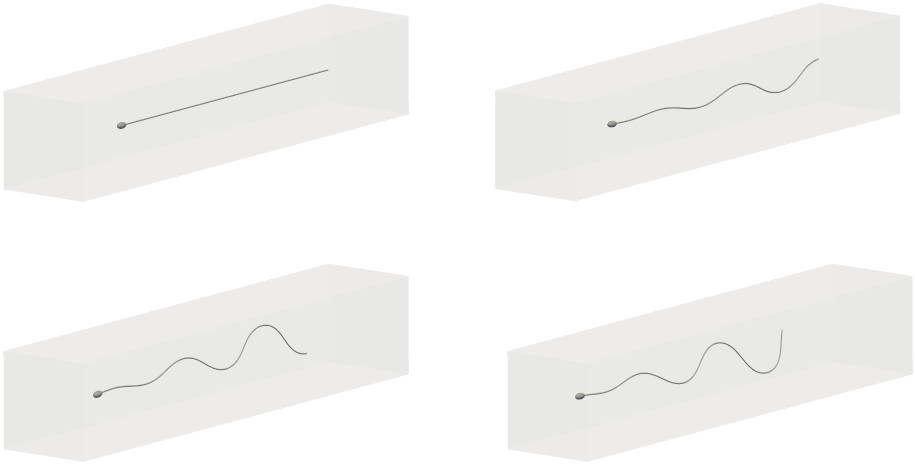


Figure IV.2.3: Three dimensional simulation of a swimming sperm cell with planar flagellar beating. The plane of the beating flagellum is the xy plane, and the maximal amplitude of the sinusoid is $A_{max} = 4\,\mu\text{m}$. The four images show the position and shape of the sperm cell at the time instants $t = 0\,\text{s}$, $t = 0.5\,\text{s}$, $t = 1.85\,\text{s}$ and $t = 2.65\,\text{s}$.

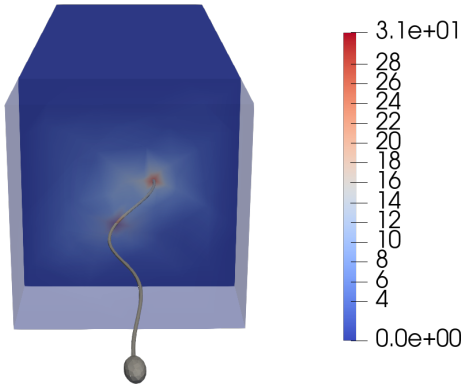


Figure IV.2.4: Fluid section for the three dimensional swimmer with planar beating. The colour represents the magnitude of the fluid velocity.

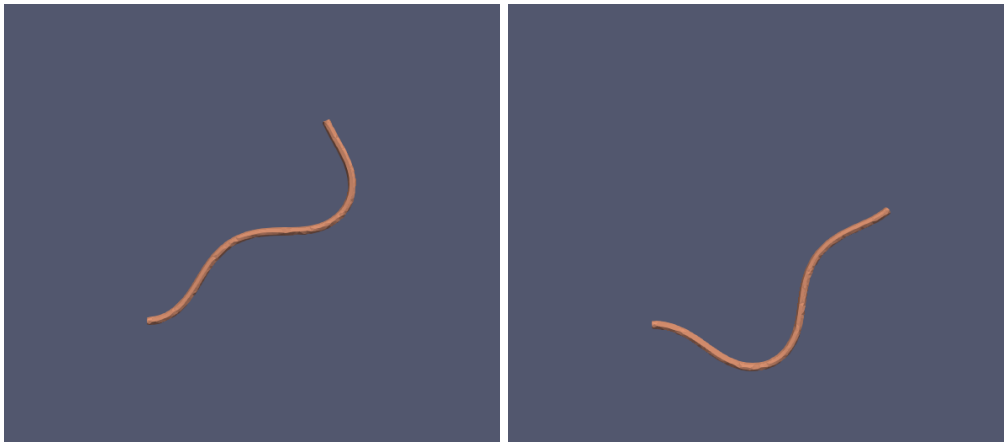


Figure IV.2.5: Two screenshots of the beating pattern governed by (IV.2.3) when the parameter values coincide with those in [62].

Tail	Twist	$\kappa_f(l,t)$			
L	τ_f	K_0	B	ω_0	λ
$41\,\mu m$	$0\,\mu m^{-1}$	$0.0351\,\mu m^{-1}$	$0.16\,\mu m^{-1}$	$0.273\,ms^{-1}$	$0.212\,\mu m^{-1}$

Table IV.2.2: The first two tables collect the parameters for IV.2.3

The two parameters have different roles: the twist parameter τ_f allows the swimming path to be three dimensional, while $K_0 > 0$ provides circular trajectories. The combination of the two effects gives rise to 3D helices. Planar circular motions of spermatozoa were already observed and modelled in [45], modelled and simulated in [6]. Since equations (IV.2.3) only model the shape of the tail's centerline, we provide an extension of $\mathbf{r}$ to the whole section of the tail because in our simulations we are considering a tail of finite thickness. The ODEs are solved using the `odeint` library, and a Runge-Kutta scheme of 4th order. The values of the parameters for this numerical experiment are collected in table IV.2.2.

Also in this case frequent remeshing is needed because the tail deformation is important, as Figure IV.2.5 shows. Moreover, reaching the initial configuration, that is the position of the tail at $t = 0$, required a pre-processing phase. In this phase, the displacement between the horizontal configuration of the tail and its position at $t = 0$ was divided in equal parts, and mesh adaptation was needed to reach the initial configuration as described in section II.2.2.

In figure IV.2.6 the mesh, the pressure and velocity fields are shown at three different time instants. We notice that large pressure differences form on the two sides of the flagellum, and that these are responsible of the swimmer's motion. The asymmetric tail beating we propose in this section allows the swimmer to follow a circular trajectory [62]; in figure IV.2.7 we show the trajectory of the center of mass of a two-dimensional sperm cell with no head, when it follows the prescribed beating of (IV.2.3). In [62], an estimate of the radius of the circular trajectory is given, and it is approximately $r_p \approx 1.3/K_0 = 37\,\mu m$. If one imagines to associate to the trajectory we obtain in IV.2.7 an equivalent radius, one can see that its magnitude is comparable to r_p. However, further investigations should be performed to improve the quality of our approximation of the problem: first, we should consider a swimmer with head, as r_p refers to the trajectory of a sperm cell with head; second, the thickness of the flagellum should be decreased, since r_p is computed using a one-dimensional approximation of the tail.

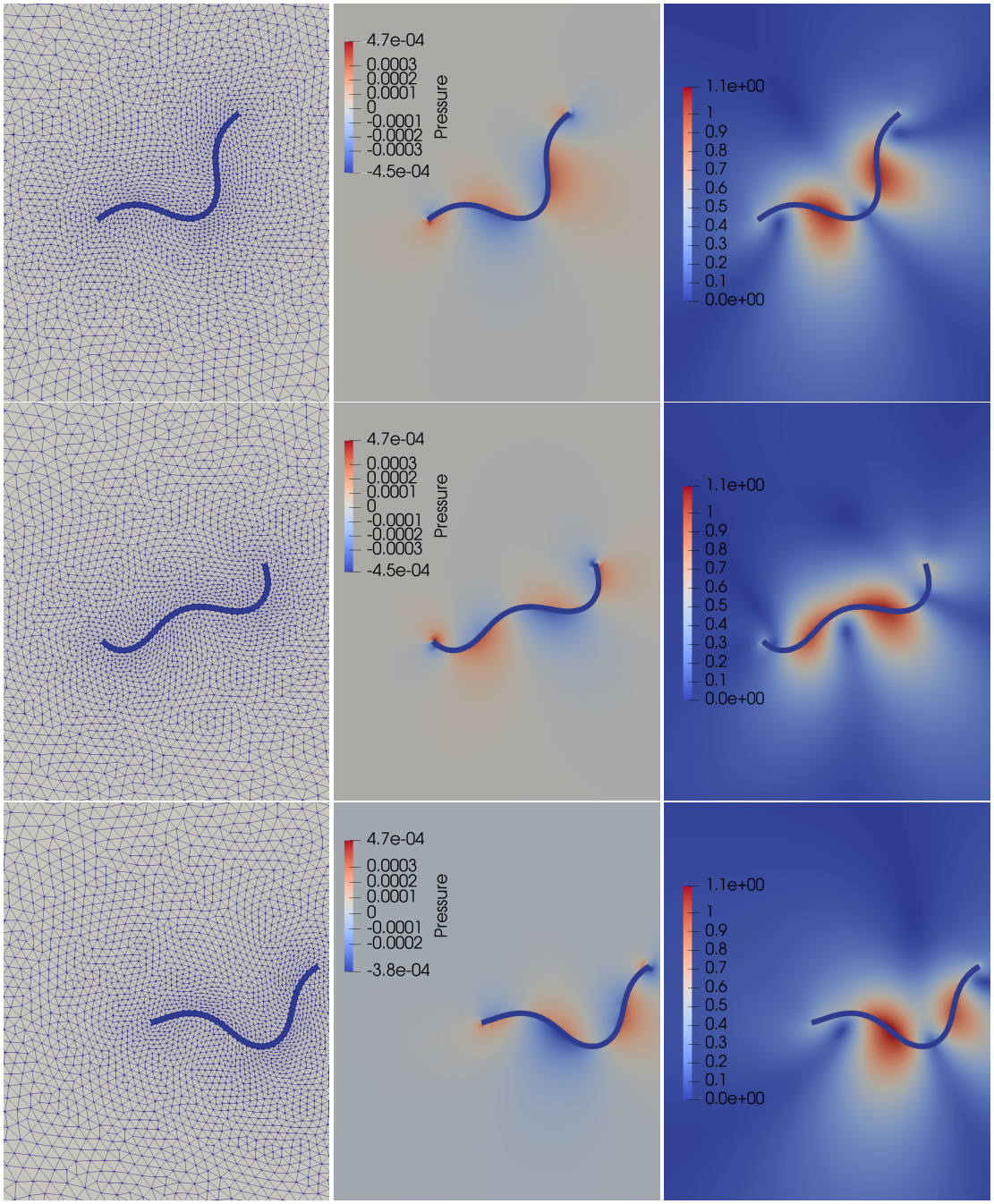


Figure IV.2.6: These pictures show the mesh, pressure and velocity fields corresponding to the asymmetrically beating flagellum. From top to bottom, they correspond to the time instants $t = 16.25\,\mathrm{s}$, $t = 48.75\,\mathrm{s}$ and $t = 64.05\,\mathrm{s}$.

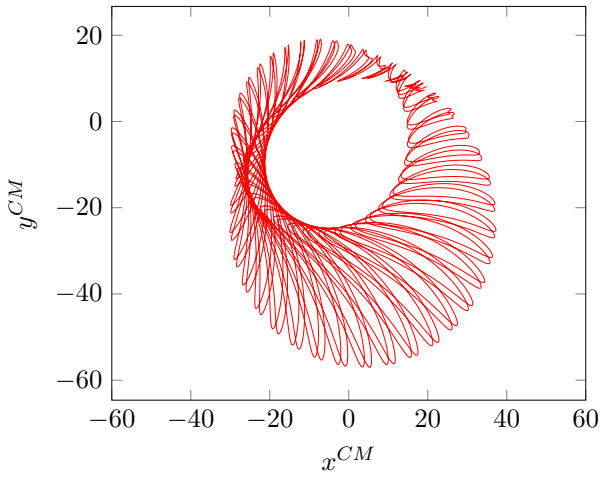


Figure IV.2.7: Oscillations around a circular path of the center of mass of the asymmetrically beating flagellum.

Conclusions and perspectives

In this thesis we have presented two mathematical and computational frameworks for the study of micro-swimming, with brief excursions in the shape optimisation of micro-robots inspired by flagellated bacteria and gait optimisation via Q-learning.

First, we modelled the swimming problem using the Boundary Integral formulation of Stokes equations and used the Boundary Element method via the *Gypsilab* library for the numerical simulations. This choice was dictated by a class of swimmers we aimed to model, i.e. flagellated micro-swimmers with rotating flagella. As their swimming strategy modifies strongly the fluid domain, any triangulation of the latter requires several reconstruction steps to carry on the simulations. Moreover, this method handles three-dimensional problems via a less expensive two-dimensional domain discretisation. A framework for the simulation of flagellated micro-swimmers was created with the objective of studying the parametric shape optimisation of flagellated micro-swimmers via black-box optimisation. Two optimisation strategies were investigated: first, the tail shape or head parameters were optimised by keeping the rest of the swimmer's shape fixed; second, the whole set of parameters was optimised at the same time. The first approach is similar to "practical" parametric optimisation, where just one feature of the swimmer is optimised [128]. The second approach, based on Bayesian optimisation, was more generic and gave better propulsion speeds. In this second case, the set of possible positions and orientations of the tails was enlarged with respect to the first one. In order to study a more realistic model, we considered a variation of the multi-flagellated swimmers to account for elasticity at the flagellar junctions. A preliminary study using Resistive Force Theory was conducted, and some preliminary tests are presented to study the effects of the fluid-elastic interaction.

In the second part of our work, we focused on the differential formulation of the micro-swimming problem and its finite element simulation. Aspects concerning the motion of rigid and elastic bodies in fluids were studied and a computational framework was proposed. This framework is complementary to the toolboxes already available in *Feel++* for the study of fluid and solid mechanics. Different test-cases were considered to validate the results, and numerical experiments were conducted to illustrate the framework, from multi-body swimmers to deformable flagellated ones. Even though we presented passive and active elasticity models and briefly discussed their application to elastic swimmers, we were not able to propose simulation results of elastic swimmers in this manuscript, essentially for lack of time. The swimming problem motivated the addition of functionalities to the *Feel++* finite element library, in order to handle the computational issues raised by mesh motion and change of reference frames. The usage of the mesh adaptation libraries *MMG* and *ParMMG*, the handling of rigid-body motion were added to the already available fluid-structure interaction formalism. Thanks to these additions, the *Feel++* swimming framework is now a computational tool that can be used to perform long time physical simulations where swimmers experience large displacements. To our knowledge, it is one of the few open source platforms for the numerical simulation of (micro-)swimming (see

chapter III.1) and, among them, perhaps the only one using a conforming Arbitrary Lagrangian Eulerian approach.

In the near future, the framework will be extended to swimmers whose motion is described by passive and active elasticity models. However, already with the current setting, many research opportunities open in the study of the long time behaviour of multi-body swimmers in bulk fluid or next to rigid boundaries. As we have mentioned in the introduction, applying reinforcement learning to these micro-swimmers, in order to find the optimal swimming strategy next to boundaries, could be investigated. However, more refined learning algorithms should be considered to efficiently handle the increasing sizes of the state and action spaces, and to search the optimal policy. In a second moment, complex fluid models (biological and viscoelastic fluids [115]) could be added to the framework, with the purpose of increasing the physical and practical interest of the *Feel++* swimming framework. In fact, in recent years, the scientific and engineering community has started to address the challenges concerning the shape and navigation of swimmers in heterogeneous and complex media [116, 80]. Finally, the framework could be used to investigate problems of shape optimisation for micro-swimmers, as it provides the fluid solver as well as a number of optimisation and numerical methods that are commonly used for shape and topology optimisation [32].

Other numerical methods could be investigated to solve the fluid-structure problem, and the immersed boundary method would be a natural choice due to an already existing implementation of the level set method in the *Feel++* library [99]. In this case, we could compare the different numerical methods and choose the more appropriate one for the swimming problem at hand.

Bibliography

[1] A. T. Chwang and T. Y. Wu, *A note on the helical movement of micro-organisms*, Proceedings of the Royal Society of London. Series B. Biological Sciences, 178 (1971), pp. 327–346.

[2] J. K. Alageshan, A. K. Verma, J. Bec, and R. Pandit, *Path-planning microswimmers can swim efficiently in turbulent flows*, arXiv:1910.01728 [physics, stat], (2019). arXiv: 1910.01728.

[3] Y. Alapan, O. Yasa, B. Yigit, I. C. Yasa, P. Erkoc, and M. Sitti, *Microrobotics and microorganisms: Biohybrid autonomous cellular robots*, Annual Review of Control, Robotics, and Autonomous Systems, 2 (2019), pp. 205–230.

[4] F. Alouges and M. Aussal, *Fem and bem simulations with the gypsilab framework*, 2018.

[5] F. Alouges, A. DeSimone, L. Giraldi, and M. Zoppello, *Self-propulsion of slender micro-swimmers by curvature control: N-link swimmers*, International Journal of Non-Linear Mechanics, 56 (2013), pp. 132 – 141. Soft Matter: a nonlinear continuum mechanics perspective.

[6] F. Alouges, A. DeSimone, L. Heltai, A. Lefebvre-Lepot, and B. Merlet, *Optimally swimming stokesian robots*, Discrete and Continuous Dynamical Systems - B, 18 (2013), p. 1189–1215.

[7] F. Alouges, A. DeSimone, and A. Lefebvre, *Optimal strokes for axisymmetric microswimmers*, The European Physical Journal E, 28 (2009), pp. 279–284.

[8] F. Alouges, A. Desimone, and A. Lefebvre-Lepot, *Optimal strokes for low reynolds number swimmers: An example*, J. Nonlinear Science, 18 (2008), pp. 277–302.

[9] F. Alouges and L. Giraldi, *Enhanced Controllability of Low Reynolds Number Swimmers in the Presence of a Wall*, Acta Applicandae Mathematicae, 128 (2013), pp. 153–179.

[10] D. Ambrosi, G. Arioli, F. Nobile, and A. Quarteroni, *Electromechanical Coupling in Cardiac Dynamics: The Active Strain Approach*, SIAM Journal on Applied Mathematics, 71 (2011), pp. 605–621.

[11] J. E. Avron, O. Kenneth, and D. H. Oaknin, *Pushmepullyou: an efficient microswimmer*, New Journal of Physics, 7 (2005), pp. 234–234.

[12] R. E. Bank and J. Xu, *An algorithm for coarsening unstructured meshes*, Numerische Mathematik, 73 (1996), pp. 1–36.

[13] K. Bente, A. Codutti, F. Bachmann, and D. Faivre, *Biohybrid and Bioinspired Magnetic Microswimmers*, Small, 14 (2018), p. 1704374.

[14] M. Bergmann, J. Hovnanian, and A. Iollo, *An Accurate Cartesian Method for Incompressible Flows with Moving Boundaries*, Communications in Computational Physics, 15 (2014), pp. 1266–1290. Edition: 2015/06/03 Publisher: Cambridge University Press.

[15] M. Bergmann and A. Iollo, *Modeling and simulation of fish-like swimming*, Journal of Computational Physics, 230 (2011), pp. 329–348.

[16] M. Bergounioux and Y. Privat, *Shape Optimization with Stokes Constraints over the Set of Axisymmetric Domains*, SIAM Journal on Control and Optimization, 51 (2013), pp. 599–628.

[17] R. K. Bock and W. Krischer, *The Data Analysis BriefBook*, Accelerator Physics, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[18] C. A. Brebbia, J. C. F. Telles, and L. C. Wrobel, *Boundary Element Techniques*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1984.

[19] F. Brezzi and M. Fortin, *Mixed and hybrid finite elements methods*, Springer series in computational mathematics, Springer-Verlag, 1991.

[20] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial, Second Edition*, Society for Industrial and Applied Mathematics, second ed., Jan. 2000.

[21] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, *Machine Learning for Fluid Mechanics*, Annual Review of Fluid Mechanics, 52 (2020), pp. 477–508.

[22] V. Chabannes, *Vers la simulation des écoulements sanguins*, PhD thesis, Université de Grenoble, July 2013.

[23] V. Chabannes, M. Ismail, C. Prud'homme, and M. Szopos, *Hemodynamic simulations in the cerebral venous network: A study on the influence of different modeling assumptions*, Journal of Coupled Systems and Multiscale Dynamics, 3 (2015), pp. 23–37.

[24] V. Chabannes, G. Pena, and C. Prud'homme, *High-order fluid–structure interaction in 2D and 3D application to blood flow in arteries*, Fifth International Conference on Advanced COmputational Methods in ENgineering (ACOMEN 2011), 246 (2013), pp. 1–9.

[25] V. Chabannes and C. Prud'homme, *Github feel++ repository.*

[26] C. Cherubini, S. Filippi, P. Nardinocchi, and L. Teresi, *An electromechanical model of cardiac tissue: Constitutive issues and electrophysiological effects*, Progress in Biophysics and Molecular Biology, 97 (2008), pp. 562 – 573. Life and Mechanosensitivity.

[27] S. Childress, *Mechanics of Swimming and Flying*, Cambridge University Press, 1 ed., July 1981.

[28] S. P. Cook, C. J. Brokaw, C. H. Muller, and D. F. Babcock, *Sperm Chemotaxis: Egg Peptides Control Cytosolic Calcium to Regulate Flagellar Responses*, Developmental Biology, 165 (1994), pp. 10–19.

[29] R. Cortez, L. Fauci, and A. Medovikov, *The method of regularized Stokeslets in three dimensions: Analysis, validation, and application to helical swimming*, Physics of Fluids, 17 (2005), p. 031504. Publisher: American Institute of Physics.

[30] M. Curatolo and L. Teresi, *The virtual aquarium: Simulations of fish swimming*, 10 2015.

[31] ——, *Modeling and simulation of fish swimming with active muscles*, Journal of Theoretical Biology, 409 (2016), pp. 18–26.

[32] C. Dapogny, P. Frey, F. Omnès, and Y. Privat, *Geometrical shape optimization in fluid mechanics using FreeFem++*, Structural and Multidisciplinary Optimization, 58 (2018), pp. 2761–2788.

[33] E. Demir and S. Yesilyurt, *Low Reynolds number swimming of helical structures in circular channels*, Journal of Fluids and Structures, 74 (2017), pp. 234–246.

[34] R. Dreyfus, J. Baudry, M. Roper, M. Fermigier, H. Stone, and J. Bibette, *Microscopic artificial swimmers*, Nature, 437 (2005), pp. 862–5.

[35] Y. El Alaoui-Faris, J.-B. Pomet, S. Régnier, and L. Giraldi, *Optimal actuation of flagellar magnetic microswimmers*, Physical Review E, 101 (2020).

[36] J. Elgeti, U. B. Kaupp, and G. Gompper, *Hydrodynamics of Sperm Cells near Surfaces*, Biophysical Journal, 99 (2010), pp. 1018–1026.

[37] J. Elgeti, R. G. Winkler, and G. Gompper, *Physics of microswimmers—single particle motion and collective behavior: a review*, Reports on Progress in Physics, 78 (2015), p. 056601.

[38] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*, Oxford University Press, June 2014.

[39] A. Ern and J.-L. Guermond, *Finite Elements II: Galerkin Approximation, Elliptic and Mixed PDEs*, vol. 73 of Texts in Applied Mathematics, Springer International Publishing, Cham, 2021.

[40] R. Featherstone, *Rigid Body Dynamics Algorithms*, Springer-Verlag, Berlin, Heidelberg, 2007.

[41] M. A. Fernández, *Coupling schemes for incompressible fluid-structure interaction: implicit, semi-implicit and explicit*, SeMA Journal, 55 (2011), pp. 59–108.

[42] M. A. Fernández, J. Mullaert, and M. Vidrascu, *Generalized Robin-Neumann explicit coupling schemes for incompressible fluid-structure interaction: Stability analysis and numerics*, International Journal for Numerical Methods in Engineering, 101 (2015), pp. 199–229.

[43] D. A. Field, *Qualitative measures for initial meshes*, International Journal for Numerical Methods in Engineering, 47 (2000), pp. 887–906.

[44] P. I. Frazier, *A Tutorial on Bayesian Optimization*, arXiv:1807.02811 [cs, math, stat], (2018). arXiv: 1807.02811.

[45] B. Friedrich, I. Riedel-Kruse, J. Howard, and F. Jülicher, *High-precision tracking of sperm swimming fine structure provides strong test of resistive force theory*, The Journal of experimental biology, 213 (2010), pp. 1226–34.

[46] C. Gadelha, B. Wickstead, and K. Gull, *Flagellar and ciliary beating in trypanosome motility*, Cell Motility and the Cytoskeleton, 64 (2007), pp. 629–643.

[47] M. Giacomini, L. Borchini, R. Sevilla, and A. Huerta, *Separated response surfaces for flows in parametrised domains: Comparison of a priori and a posteriori PGD algorithms*, Finite Elements in Analysis and Design, 196 (2021), p. 103530.

[48] L. Giraldi, P. Martinon, and M. Zoppello, *Controllability and optimal strokes for N-link microswimmer*, in 52nd IEEE Conference on Decision and Control, Firenze, Dec. 2013, IEEE, pp. 3870–3875.

[49] J. GRAY and G. J. HANCOCK, *The propulsion of sea-urchin spermatozoa*, Journal of Experimental Biology, 32 (1955), pp. 802–814.

[50] D. Gérard-Varet and L. Giraldi, *Rough wall effect on micro-swimmers*, ESAIM: Control, Optimisation and Calculus of Variations, 21 (2015), pp. 757–788.

[51] A. J. Hanson and H. Ma, *Parallel transport approach to curve framing*, tech. rep., 1995.

[52] J. Happel and H. Brenner, *Low Reynolds number hydrodynamics*, Mechanics of Fluids and Transport Processes, Springer Netherlands, 1983.

[53] H. Hasselt, *Double q-learning*, in Advances in Neural Information Processing Systems, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, eds., vol. 23, Curran Associates, Inc., 2010.

[54] J. J. L. Higdon, *The hydrodynamics of flagellar propulsion: helical waves*, Journal of Fluid Mechanics, 94 (1979), p. 331–351.

[55] Q. Huang and T. A. Cruse, *Some notes on singular integral techniques in boundary element analysis*, International Journal for Numerical Methods in Engineering, 36 (1993), pp. 2643–2659.

[56] K. Ishimoto, *Hydrodynamic evolution of sperm swimming: Optimal flagella by a genetic algorithm*, Journal of Theoretical Biology, 399 (2016), pp. 166–174.

[57] K. Ishimoto and E. A. Gaffney, *Boundary element methods for particles and microswimmers in a linear viscoelastic fluid*, Journal of Fluid Mechanics, 831 (2017), pp. 228–251.

[58] K. Ishimoto, E. A. Gaffney, and B. J. Walker, *Regularized representation of bacterial hydrodynamics*, Physical Review Fluids, 5 (2020), p. 093101.

[59] K. Ishimoto and E. Lauga, *The N-flagella problem: elastohydrodynamic motility transition of multi-flagellated bacteria*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 475 (2019), p. 20180690. Publisher: Royal Society.

[60] Jacob Gardner and Matt Kusner and Zhixiang and Kilian Weinberger and John Cunningham, *Bayesian Optimization with Inequality Constraints*, in Proceedings of the 31st International Conference on Machine Learning, vol. 32 of Proceedings of Machine Learning Research, 2014, pp. 937–945.

[61] D. Jang, J. Jeong, H. Song, and S. K. Chung, *Targeted drug delivery technology using untethered microrobots: a review*, Journal of Micromechanics and Microengineering, 29 (2019), p. 053002.

[62] J. Jikeli, L. Alvarez, B. Friedrich, L. Wilson, R. Pascal, R. Colin, M. Pichlo, A. Rennhack, C. Brenker, and U. Kaupp, *Sperm navigation along helical paths in 3d chemoattractant landscapes*, Nature Communications, 6 (2015).

[63] A. Johnen, J.-F. Remacle, and C. Geuzaine, *Geometrical validity of high-order triangular finite elements*, Engineering with Computers, 30 (2014), pp. 375–382.

[64] H. Kanchi and A. Masud, *A 3D adaptive mesh moving scheme*, International Journal for Numerical Methods in Fluids, 54 (2007), pp. 923–944.

[65] P. Kanehl and T. Ishikawa, *Fluid mechanics of swimming bacteria with multiple flagella*, Phys. Rev. E, 89 (2014), p. 042704.

[66] J. B. Keller and S. I. Rubinow, *Slender-body theory for slow viscous flow*, Journal of Fluid Mechanics, 75 (1976), p. 705–714.

[67] M. C. Kennedy and A. O'Hagan, *Bayesian calibration of computer models*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63 (2001), pp. 425–464.

[68] S. Kim and S. Karrila, *Microhydrodynamics: Principles and Selected Applications*, Butterworth - Heinemann series in chemical engineering, Dover Publications, 2005.

[69] K. Kitagawa, *Boundary element analysis of viscous flow*, 55 (1988).

[70] J. P. Kleijnen, *Kriging metamodeling in simulation: A review*, European Journal of Operational Research, 192 (2009), pp. 707–716.

[71] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, *Machine learning–accelerated computational fluid dynamics*, Proceedings of the National Academy of Sciences, 118 (2021), p. e2101784118.

[72] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, *Convergence Properties of the Nelder–Mead Simplex Method in Low Dimensions*, SIAM Journal on Optimization, 9 (1998), pp. 112–147.

[73] M. Landajuela, M. Vidrascu, D. Chapelle, and M. A. Fernández, *Coupling schemes for the FSI forward prediction challenge: Comparative study and validation*, International Journal for Numerical Methods in Biomedical Engineering, 33 (2017), p. e2813.

[74] L. Landau and E. Lifshitz, *Mechanics*, Butterworth-Heinemann, Elsevier Science, 1976.

[75] J. Larson, M. Menickelly, and S. M. Wild, *Derivative-free optimization methods*, Acta Numerica, 28 (2019), pp. 287–404.

[76] E. Lauga and T. R. Powers, *The hydrodynamics of swimming microorganisms*, Reports on Progress in Physics, 72 (2009), p. 096601.

[77] R. Ledesma-Aguilar, H. Löwen, and J. M. Yeomans, *A circle swimmer at low Reynolds number*, The European Physical Journal E, 35 (2012), p. 70.

[78] C. T. Lefèvre, A. Bernadac, K. Yu-Zhang, N. Pradel, and L.-F. Wu, *Isolation and characterization of a magnetotactic bacterial culture from the mediterranean sea*, Environmental Microbiology, 11 (2009), pp. 1646–1657.

[79] D. Li, M. Jeong, E. Oren, T. Yu, and T. Qiu, *A Helical Microrobot with an Optimized Propeller-Shape for Propulsion in Viscoelastic Biological Media*, Robotics, 8 (2019), p. 87.

[80] G. Li, E. Lauga, and A. M. Ardekani, *Microswimming in viscoelastic fluids*, Journal of Non-Newtonian Fluid Mechanics, 297 (2021), p. 104655.

[81] B. Liebchen and H. Löwen, *Optimal navigation strategies for active particles*, EPL (Europhysics Letters), 127 (2019), p. 34003.

[82] J. Lighthill, *Flagellar hydrodynamics*, SIAM Review, 18 (1976), pp. 161–230.

[83] J. Longuski and P. Tsiotras, *Analytical solutions for a spinning rigid body subject to time-varying body-fixed torques, part i: Constant axial torque.*, Journal of Applied Mechanics-transactions of The Asme, (1993).

[84] A. Lotfi, D. Marcsa, Z. Horváth, C. Prud'Homme, and V. Chabannes, *Numerical Simulation of Coupled Electromagnetic and Thermal Problems in Permanent Magnet Synchronous Machines.*, 2019, pp. 693–701.

[85] B. Maury, *Direct simulations of 2d fluid-particle flows in biperiodic domains*, Journal of Computational Physics, 156 (1999), pp. 325 – 351.

[86] K. I. McKinnon, *Convergence of the nelder–mead simplex method to a nonstationary point*, SIAM Journal on optimization, 9 (1998), pp. 148–158.

[87] T. Metivet, V. Chabannes, M. Ismail, and C. Prud'homme, *High-Order Finite-Element Framework for the Efficient Simulation of Multifluid Flows*, Mathematics, 6 (2018), p. 203.

[88] S. Mohanty, I. S. M. Khalil, and S. Misra, *Contactless acoustic micro/nano manipulation: a paradigm for next generation applications in life sciences*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 476 (2020), p. 20200621.

[89] J. L. Moran and J. D. Posner, *Phoretic Self-Propulsion*, Annual Review of Fluid Mechanics, 49 (2017), pp. 511–540.

[90] F. Mroue, *Cardiac electromechanical coupling: modeling, mathematical analysis and numerical simulation*, theses, Ecole Centrale de Nantes (ECN) ; Université Libanaise, Oct. 2019.

[91] A. Najafi and R. Golestanian, *Simple swimmer at low Reynolds number: Three linked spheres*, Physical Review E, 69 (2004), p. 062901.

[92] J. Nascimento, L. Shi, S. Meyers, P. Gagneux, N. Loskutoff, E. Botvinick, and M. Berns, *The use of optical tweezers to study sperm competition and motility in primates*, Journal of the Royal Society, Interface / the Royal Society, 5 (2008), pp. 297–302.

[93] J. A. Nelder and R. Mead, *A Simplex Method for Function Minimization*, The Computer Journal, 7 (1965), pp. 308–313.

[94] J. Ohayon, D. Ambrosi, and J.-L. Martiel, *Hyperelastic Models for Contractile Tissues*, in Biomechanics of Living Organs, Elsevier, 2017, pp. 31–58.

[95] S. D. Olson, S. S. Suarez, and L. J. Fauci, *Coupling biochemistry and hydrodynamics captures hyperactivated sperm motility in a simple flagellar model*, Journal of Theoretical Biology, 283 (2011), pp. 203–216.

[96] A. Oulmas, N. Andreff, and S. Régnier, *Closed-loop 3d path following of scaled-up helical microswimmers*, in 2016 IEEE International Conference on Robotics and Automation (ICRA), May 2016, pp. 1725–1730.

[97] V. Parthasarathy, C. Graichen, and A. Hathaway, *A comparison of tetrahedron quality measures*, Finite Elements in Analysis and Design, 15 (1994), pp. 255 – 261.

[98] G. Pena, C. Prud'homme, and A. Quarteroni, *High order methods for the approximation of the incompressible Navier–Stokes equations in a moving domain*, Computer Methods in Applied Mechanics and Engineering, 209-212 (2012), pp. 197–211.

[99] C. S. Peskin, *The immersed boundary method*, Acta Numerica, 11 (2002), pp. 479–517.

[100] N. Phan-Thien, T. Tran-Cong, and M. Ramia, *A boundary-element analysis of flagellar propulsion*, Journal of Fluid Mechanics, 184 (1987), p. 533–549.

[101] T. R. Powers, *Role of body rotation in bacterial flagellar bundling*, Physical Review E, 65 (2002), p. 040903. Publisher: American Physical Society.

[102] C. Pozrikidis, *Boundary Integral and Singularity Methods for Linearized Viscous Flow*, Cambridge Texts in Applied Mathematics, Cambridge University Press, Cambridge, 1992.

[103] Prud´homme, Christophe, Chabannes, Vincent, Doyeux, Vincent, Ismail, Mourad, Samake, Abdoulaye, and Pena, Goncalo, *Feel++ : A computational framework for galerkin methods and advanced numerical methods*, ESAIM: Proc., 38 (2012), pp. 429–455.

[104] E. M. Purcell, *Life at low reynolds number*, American Journal of Physics, 45 (1977), pp. 3–11.

[105] P. Pébay and T. Baker, *Analysis of triangle quality measures*, Math. Comput., 72 (2003), pp. 1817–1839.

[106] A. Quarteroni, *Numerical Models for Differential Problems*, MS&A, Springer Milan, 2010.

[107] Quispe, J and Régnier, S., *Magnetic miniature swimmers with multiple rigid flagella*, in 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 9237–9243.

[108] J. Quispe Navarrete, A. Oulmas, and S. Régnier, *Geometry optimization of helical swimming at low reynolds number*, 07 2019, pp. 1–6.

[109] S. E. Razavi and A. S. Ahmadi, *An ale-based finite element model of flagellar motion driven by beating waves: A parametric study*, Computers in Biology and Medicine, 66 (2015), pp. 179 – 189.

[110] L. M. Rios and N. V. Sahinidis, *Derivative-free optimization: a review of algorithms and comparison of software implementations*, Journal of Global Optimization, 56 (2013), pp. 1247–1293.

[111] B. Rodenborn, C.-H. Chen, H. L. Swinney, B. Liu, and H. P. Zhang, *Propulsion of microorganisms by a helical flagellum*, Proceedings of the National Academy of Sciences, 110 (2013), pp. E338–E347.

[112] C. Rorai, M. Zaitsev, and S. Karabasov, *On the limitations of some popular numerical models of flagellated microswimmers: importance of long-range forces and flagellum waveform*, Royal Society Open Science, 6 (2019), p. 180745.

[113] M. Sahin and K. Mohseni, *An arbitrary Lagrangian–Eulerian formulation for the numerical simulation of flow patterns generated by the hydromedusa Aequorea victoria*, Journal of Computational Physics, 228 (2009), pp. 4588–4605.

[114] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*, Springer Series in Statistics, Springer New York, New York, NY, 2018.

[115] P. Saramito, *Complex fluids: Modeling and Algorithms*, vol. 79 of Mathématiques et Applications, Springer International Publishing, Cham, 2016.

[116] D. Schamel, A. G. Mark, J. G. Gibbs, C. Miksch, K. I. Morozov, A. M. Leshansky, and P. Fischer, *Nanopropellers and Their Actuation in Complex Viscoelastic Media*, ACS Nano, 8 (2014), pp. 8794–8801.

[117] M. S. Shephard and M. K. Georges, *Automatic three-dimensional mesh generation by the finite octree technique*, International Journal for Numerical Methods in Engineering, 32 (1991), pp. 709–749.

[118] H. Shum, *Microswimmer propulsion by two steadily rotating helical flagella*, Micromachines, 10 (2019).

[119] J. Simons, L. Fauci, and R. Cortez, *A fully three-dimensional model of the interaction of driven elastic filaments in a Stokes flow with applications to sperm motility*, Reproductive Biomechanics, 48 (2015), pp. 1639–1651.

[120] M. Taghiloo and M. Miri, *Three-sphere magnetic swimmer in a shear flow*, Physical Review E, 88 (2013), p. 023008.

[121] G. I. Taylor, *Analysis of the swimming of microscopic organisms*, Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 209 (1951), pp. 447–461.

[122] A. C. H. Tsang, P. W. Tong, S. Nallan, and O. S. Pak, *Self-learning how to swim at low Reynolds number*, Physical Review Fluids, 5 (2020), p. 074101.

[123] F. Vergnet, *Structures actives dans un fluide visqueux : modélisation, analyse mathématique et simulations numériques*, PhD thesis, Université Paris-Saclay, 2019.

[124] V. A. Vladimirov, *On the self-propulsion of an n-sphere micro-robot*, Journal of Fluid Mechanics, 716 (2013), p. R1.

[125] A. VV., *Surgical Robotics. Systems Applications and Visions*, Springer.

[126] B. J. Walker, K. Ishimoto, and E. A. Gaffney, *Pairwise hydrodynamic interactions of synchronized spermatozoa*, Physical Review Fluids, 4 (2019), p. 093101.

[127] B. J. Walker, R. J. Wheeler, K. Ishimoto, and E. A. Gaffney, *Boundary behaviours of Leishmania mexicana: A hydrodynamic simulation study*, Journal of Theoretical Biology, 462 (2019), pp. 311–320.

[128] D. Walker, M. Kübler, K. I. Morozov, P. Fischer, and A. M. Leshansky, *Optimal length of low Reynolds number nanopropellers*, Nano Letters, 15 (2015), pp. 4412–4416.

[129] S. W. Walker and E. E. Keaveny, *Analysis of shape optimization for magnetic microswimmers*, SIAM Journal on Control and Optimization, 51 (2013), pp. 3093–3126.

[130] D. Wan and S. Turek, *Direct numerical simulation of particulate flow via multigrid FEM techniques and the fictitious boundary method*, International Journal for Numerical Methods in Fluids, 51 (2006), pp. 531–566.

[131] C. J. C. H. Watkins and P. Dayan, *Q-learning*, Machine Learning, 8 (1992), pp. 279–292.

[132] D. M. Woolley and G. G. Vernon, *A study of helical and planar waves on sea urchin sperm flagella, with a theory of how they are generated*, The Journal of Experimental Biology, 204 (2001), pp. 1333–1345.

[133] Z. Wu, J. Troll, H.-H. Jeong, Q. Wei, M. Stang, F. Ziemssen, Z. Wang, M. Dong, S. Schnichels, T. Qiu, and P. Fischer, *A swarm of slippery micropropellers penetrates the vitreous body of the eye*, Science Advances, 4 (2018), p. eaat4388.

[134] Z. Xia, K. W. Connington, S. Rapaka, P. Yue, J. J. Feng, and S. Chen, *Flow patterns in the sedimentation of an elliptical particle*, Journal of Fluid Mechanics, 625 (2009), pp. 249–272.

[135] X. Yan, Q. Zhou, M. Vincent, Y. Deng, J. Yu, J. Xu, T. Xu, T. Tang, L. Bian, Y.-X. J. Wang, K. Kostarelos, and L. Zhang, *Multifunctional biohybrid magnetite microrobots for imaging-guided therapy*, Science Robotics, 2 (2017), p. eaaq1155.

[136] S.-D. Zhang, N. Petersen, W.-J. Zhang, S. Cargou, J. Ruan, D. Murat, C.-L. Santini, T. Song, T. Kato, P. Notareschi, Y. Li, K. Namba, A.-M. Gué, and L.-F. Wu, *Swimming behaviour and magnetotaxis function of the marine bacterium strain mo-1*, Environmental Microbiology Reports, 6 (2014), pp. 14–20.

Presentations and publications during the doctoral studies

Oral presentations

- **Developments in Feel++ for micro-swimming simulations**

  Séminaire d'équipe Inria CALISTO - 10/12/2020

- **Numerical methods for micro-swimming**

  Journée du LabEx IRMIA - 07/10/2020

- **Mathematical and computational framework for low Reynolds number swimmers using ALE approach**

  Talk in Parallel Sessions at "9e Biennale Française des Mathématiques Appliquées et Industrielles" (Congrès SMAI 2019) - 16/05/2019

- **Optimal transport and cut locus of a surface**

  Talk at the Séminaire Doctorants de l'IRMA - 31/01/2019

Poster presentations

- **Optimal shapes for biflagellated microswimmers**

  Poster presentation at the workshop "Micro-swimmers and Soft Robotics" (Haifa)- 04/02/2020

- **Méthodes numériques pour la micro-natation**

  Poster presentation for the doctoral school MSII 269 - 08/10/2019

- **Swimming at low Reynolds number: a PDE approach**

  Poster at the Journée de Restitution des projets INFINITI du CNRS (Paris)- 31/08/2018

Articles

- **Microswimming framework in *Feel++***

  Luca Berti, Vincent Chabannes, Laetitia Giraldi and Christophe Prud'homme

  (in preparation)

- **Modeling and finite element simulation of multi-sphere swimmers**

  Luca Berti, Vincent Chabannes, Laetitia Giraldi and Christophe Prud'homme

  Comptes Rendus. Mathématique, Tome 359 (2021) no. 9, pp. 1119-1127.

- **Shapes enhancing the propulsion of multiflagellated helical microswimmers**

  Luca Berti, Mickaël Binois, François Alouges, Matthieu Aussal, Christophe Prud'homme and Laetitia Giraldi

  (preprint)

- **Swimming at low Reynolds number**

  Luca Berti, Laetitia Giraldi and Christophe Prud'homme

  ESAIM Proceedings and Surveys (2020). Vol. 60, p. 46-60.

Luca BERTI

Numerical methods and optimisation for micro-swimming

Résumé

Cette thèse a pour but la modélisation et la simulation de nageurs à faible nombre de Reynolds. Deux cadres computationnels ont été étudiés pour cela, et ils ont été utilisés en synergie avec des algorithmes d'optimisation.

En premier lieu, nous avons étudié un problème d'optimisation de forme paramétrique pour des nageurs multi-flagelles. Les simulations sont basées sur la formulation intégrale des équations de Stokes et sur la méthode des éléments de frontière, tandis que l'optimisation est réalisée avec des algorithmes du type boîte-noire.

En deuxième lieu, le problème de la micro-natation est modélisé en utilisant le formalisme Arbitrary-Lagrangian-Eulerian et simulé avec la méthode des éléments finis. Un cadre pour la simulation de différents nageurs est proposé : nageurs multi-corps, flagellés ou élastiques peuvent être étudiés avec cette plateforme modulaire. Nous en montrons l'interaction avec des algorithmes de machine learning en étudiant un nageur multi-sphères bidimensionnel.

Mots clés : micro-natation, éléments finis, éléments de frontière, optimisation

Résumé en anglais

This thesis is devoted to modelling and simulating swimmers at low Reynolds number. Two computational frameworks are devised for these purposes, and they are used in synergy with optimisation algorithms.
In the first part we study a problem of parametric shape optimisation for multi-flagellated micro-swimmers. The simulations are based on the Boundary Integral formulation of Stokes equations and the Boundary Element method, while the optimisation is carried out with black-box algorithms.
In the second part, the problem of swimming micro-organisms is modelled using the Arbitrary-Lagrangian-Eulerian formalism and simulated via the Finite Element method. A coherent framework for the simulation of different swimmers is proposed: multi-body, flagellated or elastic micro-swimmers can be investigated via this modular platform. We show that interactions with optimisation or machine-learning algorithms are possible by studying the gait optimisation of a bidimensional multi-sphere micro-swimmer.

Keywords : micro-swimming, finite element method, boundary element method, optimization