INTERNSHIP REPORT

Shape optimisation for rigid objects in a Stokes flow

Author:
Lucas Palazzolo

Supervisors:
Luca Berti
Michaël Binois
Laetitia Giraldi
Christophe Prud'homme

August 18, 2023

Contents

Introduction

Shape optimization is a field of study that focuses on finding the best possible shape for a given object or system in order to optimize certain performance criteria or objectives which are typically a function of differential equations defined on the shape itself. The shape of an object can have a significant impact on its behavior, efficiency, and functionality. By altering the shape, engineers and researchers aim to improve various aspects such as aerodynamics, structural strength, heat transfer, and fluid flow characteristics [1,7,12]. Shape optimization is employed in a wide range of disciplines, including engineering, physics, biology, and design. The optimisation problem can be written as

$$\inf_{\Omega \in \Omega_{ad}} J(\Omega, u(\Omega))$$

with u solution of a PDE defined on the domain Ω. Several researchers have made significant contributions to the field of shape optimization. One notable figure is Jean Céa, a French mathematician who made substantial advancements in shape optimization theory [4]. Céa developed the concept of shape derivatives and established the theoretical foundations for solving shape optimization problems using variational methods. His work laid the groundwork for many subsequent developments in the field.

There are various approaches and methods employed in shape optimization. These methods can be broadly categorized into three main families: parametric shape optimization, geometric shape optimization and topological shape optimization. In this report we focus on geometric and parametric optimisation.

Geometric shape optimisation

Geometric shape optimization focuses on directly modifying the shape of an object to achieve the desired objectives via a non-parametric deformation field θ. It involves manipulating the geometry of the object to improve its performance. We consider a reference domain Ω_0, which we assume to be a regular bounded open subset of $\mathbb{R}^N$. Let $\partial\Omega_0$ the boundary of this domain. We denote with θ the deformation applied to the initial domain to get the new domain Ω, i.e.

$$\Omega = (\mathrm{Id} + \theta)(\Omega_0).$$

This deformation is illustrated in Figure 2 where a deformation field is applied to a reference domain. In the pursuit of finding shape derivative of the cost function, a well-known method called Cea's method, developed by Cea in [4], proves to be both simple and effective. However, it should be noted that Cea's method is considered a *formal* approach as it relies on certain assumptions regarding the regularity of the problem's data. In section 1, we look at the numerical solution of these two problems using two numerical methods : the gradient descent [1] and the null space gradient flow [7].

The gradient descent method is a classic method for solving geometric shape optimisation problems. In this method, an initial shape is iteratively modified by moving in

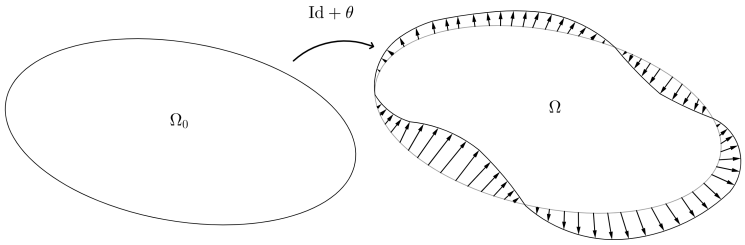

Figure 2: Geometric shape optimisation principle : the Ω_0 domain is deformed according to a deformation field θ such that the new Ω domain is given by $\Omega = (\mathrm{Id} + \theta)(\Omega_0)$.

the direction of the negative gradient of the objective function. The process continues until convergence to an optimal shape is achieved. The null space gradient flow method involves finding the shape that minimizes the objective function while satisfying a set of constraints. It utilizes the notion of null space and range space directions to guide the shape optimization process.

To conduct our simulations, we employ the `Feel++` open-source library [13]. It allows solution arising of all PDE from the formulation of the shape optimisation problems.

Parametric shape optimisation

In contrast to geometric shape optimization, parametric shape optimization involves defining a parameterized representation of the shape and then optimizing the values of these parameters to achieve specific objectives. This approach is advantageous as it allows us to work in a finite-dimensional space, which can simplify the problem. However, it's important to note that this finite-dimensional representation limits the set of achievable shapes. An example of a parameterized shape is illustrated in Figure 3, where an ellipsoid is defined by three parameters: a, b, and c. To achieve these parameterized de-

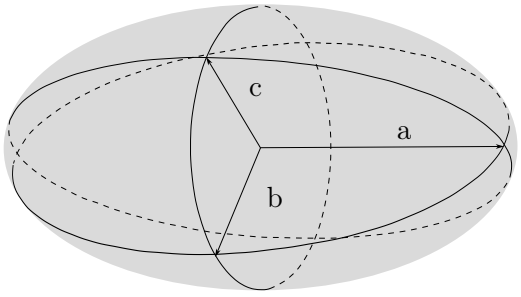

Figure 3: Ellipsoid parametrized by three different parameters: a, b and c.

formations, the Free-Form Deformation (FFD) technique is commonly employed. FFD

allows for the deformation of a parameterized shape representation by manipulating control points. By adjusting the positions of these control points, the region containing the shape is deformed, resulting in a corresponding deformation of the shape itself. FFD provides relatively smooth and regular deformations, which helps avoid issues related to irregularity. An effective approach to solving finite optimization problems involving parameterized shapes is through Bayesian optimization [9].

Bayesian optimization is a robust and efficient technique widely used in optimization tasks. It addresses the challenge of finding the optimal point within a given objective function, which might be computationally expensive or time-consuming to evaluate. At its core, Bayesian optimization employs a probabilistic model, often a Gaussian process, to represent the unknown objective function. This model captures a prior belief about the behavior of the function and gets updated as new observations are made. By iteratively selecting the next set of parameters to evaluate based on an acquisition function that balances exploration and exploitation, Bayesian optimization effectively explores the parameter space to identify promising regions for optimization. One of its notable strengths is its ability to handle noisy and black-box objective functions. It can optimize functions even in scenarios where evaluations are noisy or when the function is not directly accessible but can only be obtained through a slow and costly process. This makes Bayesian optimization particularly valuable in applications such as hyperparameter tuning of machine learning models.

However, despite the transition to a finite-dimensional problem, the optimization process may still involve a high-dimensional space with constraints. In such cases, the Scalable Constraint Bayesian Optimization (SCBO) algorithm proves to be highly effective [6]. This algorithm builds upon Bayesian optimization principles while incorporating constraints into the optimization process. As a result, it offers an efficient way to optimize shapes while adhering to specific constraints.

For our simulations, we make use of the open-source libraries `Gypsilab` [11] and `BoTorch` [2]. The former provides Boundary Element Method (BEM) tools to solve the Stokes problem within an infinite domain. The latter library, `BoTorch`, offers the SCBO algorithm implementation based on the principles outlined in [6].

1 Geometric Shape Optimisation

This section focuses on geometric shape optimisation. In Subsection 1.1, the various tools required to solve problems of this type are presented [1, 4]. This is followed by a description of the two resolution methods, gradient descent [1] and null space gradient flows [7], in Subsection 1.2. Two examples of use are then studied: in Subsection 1.3 a linear elasticity problem, the cantilever [1], and in Subsection 1.4 a Stokes fluid problem, a rigid solid in a Stokes flow [12]. Finally, a brief conclusion is proposed in Subsection 1.5.

1.1 Introduction

1.1.1 Definitions

As presented in [1], in order to describe the set of admissible forms, we need to introduce the following set of diffeomorphisms.

Definition 1.1.1. *We define a space of diffeomorphisms on* $\mathbb{R}^N$ *(which can be seen as a deformation of the identity) by*

$$\mathcal{T} = \left\{ T \mid (T - Id) \in W^{1,\infty}(\Omega, \mathbb{R}^N) \text{ and } (T^{-1} - Id) \in W^{1,\infty}(\Omega, \mathbb{R}^N) \right\}.$$

Now we can clarify the admissible shapes obtained by deformation of Ω_0, that are defined by the following set.

Definition 1.1.2. *The set of admissible shapes obtained by deformation of* Ω_0 *is given by*

$$C(\Omega_0) = \left\{ \Omega \subset \mathbb{R}^N \mid \exists T \in \mathcal{T}, \quad \Omega = T(\Omega_0) \right\},$$

where $\mathcal{T}$ *is defined by Definition 1.1.1.*

In view of the above definitions, it is natural to consider the vector field θ such as

$$T = \mathrm{Id} + \theta \qquad \text{with} \qquad \theta \in W^{1,\infty}(\Omega, \mathbb{R}^N) \text{ and } T \in \mathcal{T}.$$

In order to be able to define a differentiability notion in Ω_0 with respect to θ, we need the following lemma which guarantees that if θ is small enough then $T = \mathrm{Id} + \theta$ is indeed a diffeomorphism which belongs to $\mathcal{T}$.

Lemma 1.1.1. *For all* $\theta \in W^{1,\infty}(\Omega, \mathbb{R}^N)$ *that verify* $\|\theta\|_{W^{1,\infty}(\Omega,\mathbb{R}^N)} < 1$*, the map* $T = Id + \theta$ *is a bijection of* $\mathbb{R}^N$ *that belongs to* $\mathcal{T}$ *defined by Definition 1.1.1.*

Proof. See [1, Lemma 6.13] □

All that remains is to define the notion of differentiability in relation to the domain that we call shape derivation or differentiability with respect to the domain. The following definition applies to differentiability in the Fréchet sense as well as in the Gâteaux sense.

Definition 1.1.3. *Let $J(\Omega)$ a map such that $J : C(\Omega_0) \to \mathbb{R}$. J s said to be differentiable with respect to the domain on Ω_0 if*

$$\theta \mapsto J\left((Id + \theta)(\Omega_0)\right)$$

is differentiable in 0 in the Banach space $W^{1,\infty}(\Omega, \mathbb{R}^N)$.

Let us introduce some notation to define the matrix scalar product.

Definition 1.1.4. *Let A and B be two real matrices of dimension N. We define the scalar product of two matrices as*

$$A : B = tr(A^T B).$$

and we will note thereafter $\|\cdot\|$ the associated norm.

1.1.2 Derivation of integrals

In this subsection, Definition 1.1.3 of differentiability with respect to the domain will be applied to volume or surface integrals. The paragraph will be brief, for more details we point the reader to [1]. For the rest of the paper, we denote n as the unit normal pointing outwards to the domain.

Proposition 1.1.1. *Let Ω_0 be a regular bounded open subset of $\mathbb{R}^N$. Let $f \in W^{1,1}(\mathbb{R}^N)$ and J be the map of $C(\Omega_0)$ in $\mathbb{R}$ defined by*

$$J(\Omega) = \int_\Omega f.$$

Then J is differentiable in Ω and for all $\theta \in W^{1,\infty}(\Omega, \mathbb{R}^N)$ we have

$$DJ(\Omega)(\theta) = \int_\Omega \nabla \cdot (\theta f) = \int_{\partial\Omega} \theta \cdot n f.$$

Proof. See [1, Proposition 6.22] □

Proposition 1.1.2. *Let Ω_0 be a regular bounded open subset of $\mathbb{R}^N$. Let $f \in W^{2,1}(\mathbb{R}^N)$ and J be the map of $C(\Omega_0)$ in $\mathbb{R}$ defined by*

$$J(\Omega) = \int_{\partial\Omega} f.$$

Then J is differentiable in Ω and for all $\theta \in C^1(\mathbb{R}^N, \mathbb{R}^N)$ we have

$$DJ(\Omega)(\theta) = \int_{\partial\Omega} (\nabla f \cdot \theta + f(\nabla \cdot \theta) - f(\nabla\theta n \cdot n)) = \int_{\partial\Omega} \theta \cdot n \left(\frac{\partial f}{\partial n} + (\nabla \cdot n) f\right).$$

Proof. See [1, Proposition 6.24 & Lemma 6.25] □

Remark 1.1.1. *Note that we need $\theta \in C^1(\mathbb{R}^N, \mathbb{R}^N)$. Indeed, we need to define the trace of $\nabla\theta$ on $\partial\Omega$, and therefore the hypothesis $\theta \in W^{1,\infty}(\Omega, \mathbb{R}^N)$ is not sufficient.*

1.1.3 Derivation of domain-dependent functions and equations: Cea's method

In the following, we consider a cost function J (depending on a domain Ω) which we wish to minimize (or maximize) by finding the shape of the optimal domain. Additionally, we assume that J relies on a variable u, which is a solution to a specific differential equation defined on the domain Ω. Consequently, it becomes necessary to differentiate the cost function and the differential equation with respect to the domain.

A simple and efficient method for calculating values is the Cea's method developed in [4]. This method also allows us to "guess" the definitions of the adjoint state p. However, it is important to note that this method is *formal*, as it relies on certain assumptions about the regularities of the problem's data, particularly concerning the solution u. For more rigorous calculations, the use of Eulerian and Lagrangian derivatives, as described in [1], becomes necessary. This method incorporates the concepts of the primal problem or state problem, the dual problem or adjoint problem, and the Lagrangian. The problem is presented as follows.

Cost Function : Let J be a (domain-dependent) cost function that we seek to minimize (or maximize) such that

$$\begin{array}{rcl} J : C(\Omega_0) & \to & \mathbb{R} \\ \Omega & \mapsto & J(\Omega, u(\Omega)), \end{array}$$

where J *depends* of the solution u of a differential equation. For ease of reading, we omit the u in the notation of the cost function. We seek to solve the following problem

$$\inf_{\Omega \in \Omega_{ad}} J(\Omega),$$

where Ω_{ad} corresponds to the admissible domains.

Primal Equation : Let u be the solution of a problem, called a state problem or primal problem. Let E be the map governing the variational formulation associated with the primal problem such that

$$\begin{array}{rcl} E : V \times V & \to & \mathbb{R} \\ (v, q) & \mapsto & E(v, q), \end{array}$$

where V is, in our case, a Sobolev space. Then, by definition of the weak formulation, for all $q \in V$ we have

$$E(u, q) = 0,$$

with u the solution of the primal problem.

The Lagrangian : The Cea's method is based on duality. For this purpose, we consider the primal equation as a constraint and introduce the following Lagrangian

$$\begin{array}{rcl} \mathcal{L} : C(\Omega_0) \times V \times V & \to & \mathbb{R} \\ (\Omega, v, q) & \mapsto & J(\Omega) + E(v, q), \end{array}$$

which is the sum of the objective function and the variational formulation of the primal equation. We must not forget that J also depends on v ! It is absolutely important for the following to consider that the three variables are independent of each other. Thus, the space V must be independent of the domain Ω. When the state problem has a Dirichlet condition, we must add another Lagrange multiplier, as explained in [1], $\psi \in V$ (independent of the other variables), such that

$$\begin{array}{rcl} \mathcal{L} : C(\Omega_0) \times V \times V \times V & \to & \mathbb{R} \\ (\Omega, v, q, \psi) & \mapsto & J(\Omega) + E(v, q) + F(v, \psi), \end{array}$$

where F represents the constraint associated with the Dirichlet condition and $\tilde{E}$ the integration of the strong formulation. To simplify the following, we will assume that E, $\tilde{E}$ and F are bilinear. We then quickly obtain the different equations and the gradient of J (assuming all the necessary regularities).

Primal problem or state problem : Indeed, we have for all $(q, \phi) \in V^2$

$$\left\langle \frac{\partial \mathcal{L}}{\partial q}(\Omega, u, q), \phi \right\rangle = 0, \tag{1.1}$$

with u solution of the primal problem, i.e. for all $\phi \in V$

$$E(u, \phi) = 0, \tag{1.2}$$

by bilinearity of E. This results in u being the solution to the variational formulation (1.2) of the primal problem.

Dual problem or adjoint problem : For all $(v, \phi) \in V^2$, we have

$$\left\langle \frac{\partial \mathcal{L}}{\partial v}(\Omega, v, p), \phi \right\rangle = 0, \tag{1.3}$$

with p solution of the dual problem, i.e. for all $\phi \in V$

$$E(\phi, p) + \frac{\partial J}{\partial v}(\Omega)(\phi) = 0, \tag{1.4}$$

by bilinearity of E and by the fact that J also depends on v. This results in p being solution to the variational formulation (1.4) of the dual problem.

Gradient of the cost function : Since $u(\Omega)$ is a solution of the primal problem and thus vanishes the variational formulation (1.2), we have

$$\mathcal{L}(\Omega, u(\Omega), q) = J(\Omega) \qquad \forall q \in V.$$

The key point is that this is valid for all q in V. Thus, we can take q in particular in the following as the solution of the dual problem (similarly for Lagrange multipliers in

the case of Dirichlet conditions). To emphasize the dependence of the solution u on the domain, we write $u(\Omega)$. By independence of the variable q with respect to Ω and deriving this relation using chain rule (always assuming that we have the necessary regularities to do so), we get

$$DJ(\Omega)(\theta) = \frac{\partial \mathcal{L}}{\partial \Omega}\left(\Omega, u(\Omega), q\right)(\theta) + \left\langle \frac{\partial \mathcal{L}}{\partial v}\left(\Omega, u(\Omega), q\right), u'(\Omega)(\theta) \right\rangle.$$

By taking $q = p(\Omega)$ solution of the dual problem (1.4), the last term vanishes. Finally, we come to

$$DJ(\Omega)(\theta) = \frac{\partial \mathcal{L}}{\partial \Omega}(\Omega, u(\Omega), p(\Omega))(\theta). \tag{1.5}$$

1.2 Solution methods

To minimize the cost function $J(\Omega)$, we differentiate according to the variable θ which parametrizes the form $\Omega = (\mathrm{Id} + \theta)(\Omega_0)$. In the various problems we study, certain boundaries will be assumed to be fixed, i.e. non-deformable, and noted Γ_{fixed}. We denote Γ the deformable part of $\partial\Omega$, thus

$$\partial\Omega = \Gamma \cup \Gamma_{fixed}.$$

In [1] a method is quickly defined for all cost functions whose derivative can be written as follows

$$DJ(\Omega)(\theta) = \int_\Gamma \theta \cdot n G(\Omega), \tag{1.6}$$

where $G(\Omega)$ is a function (which depends of the state and of the adjoint) which is called the shape gradient. In the following, we will assume that we are studying an optimisation problem with an equality constraint in order to preserve the volume of the domain. Thus, we have

$$\Omega_{ad} = \{\Omega \in C(\Omega_0) \mid \Gamma_{fixed} \subset \partial\Omega,\ g(\Omega) = 0\} \quad \text{with} \quad g(\Omega) = |\Omega| - |\Omega_0|. \tag{1.7}$$

The non-deformation constraints the boundaries Γ_{fixed} are taken into account by simply imposing

$$\theta = 0 \qquad \text{on} \qquad \Gamma_{fixed},$$

i.e. we impose a null displacement field at these boundaries. On top of that, by using the Proposition 1.1.1 the form derivative of the volume constraint, $g(\Omega)$, is given by

$$Dg(\Omega)(\theta) = \int_\Gamma \theta \cdot n. \tag{1.8}$$

Two methods will be presented to solve this type of problems: gradient descent and Null Space Gradient Flow.

1.2.1 Solution by a gradient descent method

The gradient descent (GD) method is an optimization technique used to minimize a function iteratively. By leveraging the gradient, which indicates the direction of steepest ascent, the algorithm takes steps in the opposite direction to reach the optimal solution. The process continues until convergence is achieved, resulting in the parameters or variables that minimize the function.

By taking a new form, Ω_t, such as

$$\Omega_t = (Id + \theta_t)(\Omega_0) \qquad \text{with} \qquad \theta_t = -tG(\Omega_0)n,$$

where $t > 0$, we have

$$DJ(\Omega_0)(\theta_t) = -t \int_{\Gamma_0} G(\Omega_0)^2 < 0.$$

Thus, for a sufficiently small step size t, we have

$$J(\Omega_t) < J(\Omega_0) \qquad \text{if} \qquad G(\Omega_0) \neq 0.$$

It can be noted that to have $DJ(\Omega_0)(\theta) < 0$, we need to choose $\theta \cdot n > 0$. Let $l \in \mathbb{R}$ a Lagrange multiplier for the volume constraint. The domain optimality condition, by using the Lagrange multiplier theorem, can be seen as

$$DJ(\Omega)(\theta) + lDg(\Omega)(\theta) = \int_{\Gamma} \theta \cdot n \left[l + G(\Omega)\right] = 0. \tag{1.9}$$

Numerical implementation : For the numerical implementation, a domain sequence, Ω_k, is computed which verifies the following constraints

$$\partial\Omega_k = \Gamma_k \cup \Gamma_{fixed}.$$

Let $t > 0$ a fixed descent step, as explained in [1] the following iterations are performed until convergence, for $k \geq 0$:

$$\Omega_{k+1} = (\mathrm{Id} + \theta_k)(\Omega_k)$$

with

$$\theta_k = \begin{cases} -t \left[l_k + G(\Omega_k)\right] n_k & \text{on } \Gamma_k \\ 0 & \text{on } \Gamma_{fixed} \end{cases}$$

where n_k is the normal vector of $\partial\Omega_k$ and $l_k \in \mathbb{R}$ a Lagrange multiplier. This choice makes (1.9) negative. It should be noted that one needs to extend the definition of θ_k over all Ω_k and not only over $\partial\Omega_k$. To extend the trace of θ_k on $\partial\Omega_k$ inside Ω_k, we can solve the following system

$$\begin{cases} -\Delta\theta_k = 0 & \text{on } \Omega_k \\ \theta_k = 0 & \text{on } \Gamma_{fixed} \\ \theta_k = -t \left[l_k + G(\Omega_k)\right] n_k & \text{on } \Gamma_k. \end{cases}$$

Once we know θ_k about Ω_k, we can deform the whole mesh to obtain a new one of the domain Ω_{k+1}. It will be advisable to remesh the domain from time to time when the mesh becomes of poor quality measure in `Feel++` (which leads to numerical errors). The quality measures implemented are taken from [8]. A 2D quality measure, function of the area A and the edge lengths $|e_i|$ of the simplex, is

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

We have the inequality $q_{2D} \leq 1$ and the equality is attained only for an equilateral triangle. A 3D quality measure function of the volume V and the facet area $|A_i|$ of the tetrahedra, is

$$q_{3D} = \frac{2^3 3^8 \sqrt{3} V^4}{\left(\sum_{i=1}^4 A_i^2\right)^3}.$$

By solving the above field extension equation, we can end up with a regularity problem. For the cantilever framework, one needs a higher regularity. To do this, in [1], it is explained that it is sufficient to solve the following system

$$\begin{cases} -\Delta\theta_k = 0 & \text{on } \Omega_k \\ \theta_k = 0 & \text{on } \Gamma_{fixed} \\ \frac{\partial\theta_k}{\partial n} = -t\left[l_k + G(\Omega_k)\right] n_k & \text{on } \Gamma_k. \end{cases}$$

which still provides a descent direction. Indeed, we obtain

$$\begin{aligned} DJ(\Omega_k)(\theta_k) + l_k Dg(\Omega_k)(\theta_k) &= \int_{\Gamma_k} \theta_k \cdot n_k \left[l_k + G(\Omega_k)\right] \\ &= t^{-1} \int_{\Omega_k} \Delta\theta_k \cdot \theta_k - t^{-1} \int_{\Gamma_k} \theta_k \cdot \nabla\theta_k n_k, \end{aligned}$$

because of the boundary conditions and the fact that $\Delta\theta_k = 0$. By integrating, we finally obtain that

$$DJ(\Omega_k)(\theta_k) + l_k Dg(\Omega_k)(\theta_k) = -t^{-1} \int_{\Omega_k} \|\nabla\theta_k\|^2 \leq 0.$$

Concerning the choice of the implementation of the Lagrange multiplier, the same model as *G.Allaire* (`http://www.cmap.polytechnique.fr/~allaire/map562/cantilever.edp`) will be taken into account (a kind of augmented Lagrangian). Thus, the Lagrange multiplier l_k will be a convex combination of the Lagrange multiplier at the previous iteration and the integration of the normalized shape gradient, and we add a normalized constraint to preserve the volume. We thus have

$$l_k = a l_{k-1} + b \frac{\int_{\Gamma_k} G(\Omega_k)}{|\Gamma_k|} + c \frac{|\Omega_k| - |\Omega_0|}{|\Omega_0|},$$

with a, b and c parameters to be chosen according to the problem studied.

1.2.2 Null Space Gradient Flow

Null Space Gradient Flow (NSGF), presented in [7], is a new approach to solving constrained optimization problems. Unlike traditional methods, this technique does not require necessarily the adjustment of non-physical parameters, making it highly practical and reliable. Its applicability to shape optimization applications further enhances its usefulness.

The key concept behind NSGF is to modify the gradient flow algorithm to accommodate the presence of constraints. This is achieved by solving an Ordinary Differential Equation (ODE) that governs the optimization trajectories, denoted as $x(t)$. The ODE takes the form:

$$\dot{x} = -\alpha_J \xi_J(x(t)) - \alpha_C \xi_C(x(t))$$

Here, α_J and α_C positive parameters, control the influence of the objective function and constraint violation, respectively. $\xi_J(x)$ and $\xi_C(x)$ represent the null space and range space directions, respectively. The null space direction, $\xi_J(x)$, is obtained by projecting the gradient $\nabla J(x)$ onto the cone of feasible directions, ensuring descent while respecting the constraints. The range space direction, $\xi_C(x)$, guides the optimization path smoothly towards the feasible region.

We consider the case where the optimization takes place on a Hilbert space V with inner product $\langle ., . \rangle_V$. The transpose of the differential of a function in infinite dimension is defined as follows.

Definition 1.2.1. *Let $g : V \to \mathbb{R}^p$ a differentiable function and x a point of V. Then, for any $\mu \in \mathbb{R}^p$ it exists a unique vector $Dg^T(x)(\mu) \in V$ such as*

$$\forall \mu \in \mathbb{R}^p \quad \forall \phi \in V \qquad \left\langle Dg^T(x)(\mu), \phi \right\rangle_V = \mu^T Dg(x)(\phi).$$

The linear operator $Dg^T(x) : \mathbb{R}^p \to V$ is called the transpose of $Dg(x)$.

For equality constrained problems (see Definition 3.3 in [7]) , null space step ξ_J and range space step ξ_C are defined as the following.

Definition 1.2.2. *For any domain Ω, the null space and range space directions $\xi_C(\Omega) : \mathbb{R}^N \to \mathbb{R}^N$ and $\xi_J(\Omega) : \mathbb{R}^N \to \mathbb{R}^N$ are defined by*

$$\begin{aligned}
\xi_C(\Omega)(X) =& Dg^T(\Omega) \left[(Dg(\Omega) Dg^T(\Omega))^{-1} g(\Omega) \right](X), \\
\xi_J(\Omega)(X) =& \nabla J(\Omega)(X) - Dg^T(\Omega) \left[(Dg(\Omega) Dg^T(\Omega))^{-1} Dg(\Omega) \nabla J(\Omega) \right](X).
\end{aligned}$$

In order to control the step size $\|\theta_n\|_{L^\infty(\mathbb{R}^N, \mathbb{R}^N)}$ and keep all values of the displacement of the order of the mesh size, the parameters $\alpha_{J,n}$ and $\alpha_{C,n}$ are updated dynamically. We consider $A_J > 0$ and $A_C > 0$ two parameters and $n_0 > 0$ the n_0-th iteration. The

coefficients are updated at every iteration according to the following rules :

$$\alpha_{J,n} = \begin{cases} \frac{A_J \texttt{hmin}}{\|\xi_J(\Omega_n)\|_{L^\infty}} & n < n_0, \\ \frac{A_J \texttt{hmin}}{\max\{\|\xi_J(\Omega_n)\|_{L^\infty}, \|\xi_J(\Omega_{n_0})\|_{L^\infty}\}} & n \geq n_0, \end{cases}$$

$$\alpha_{C,n} = \min\left\{0.9, \frac{A_C \texttt{hmin}}{\max\{\texttt{1e-9}, \|\xi_C(\Omega_n)\|_{L^\infty}\}}\right\}.$$

As explained in [7], A_J and A_C don't need fine tuning. Thus, to obtain a more general algorithm with the least possible parameters, we take $A_J = A_C = 1$. These normalizations ensure that the infinite norm of the various terms is less than the smallest mesh size, i.e.

$$\forall n \geq 0 \quad \|\alpha_{J,n}\xi_J(\Omega_n)\|_{L^\infty} \leq A_J \texttt{hmin} \quad \text{and} \quad \|\alpha_{C,n}\xi_C(\Omega_n)\|_{L^\infty} \leq \min\{\texttt{0.9}, A_C \texttt{hmin}\}.$$

The key point of this method is to be able to determine the transpose of the differential of the equality constraint. To do this, consider a Hilbert space $V = H^1(\mathbb{R}^N)$ such that $V \subset W^{1,\infty}(\Omega, \mathbb{R}^N)$ with the following scalar product

$$\forall (\theta, \theta') \in V^2, \quad \langle \theta, \theta' \rangle_V = \int_\Omega \gamma^2 \nabla\theta : \nabla\theta' + \theta \cdot \theta'$$

where the parameter γ is set proportional to the minimum mesh element size. Remember that the g equality constraint, (1.7), is written as

$$\begin{array}{rcl} g : V & \to & \mathbb{R} \\ \theta & \mapsto & |(\mathrm{Id} + \theta)(\Omega)| - |\Omega_0|. \end{array}$$

and its differential, (1.8), is defined by

$$\begin{array}{rcl} Dg(\Omega) : V & \to & \mathbb{R} \\ \theta & \mapsto & \int_\Gamma \theta \cdot n. \end{array}$$

By abuse of notation, the most of the time we write $g(\Omega)$ instead of $g(\theta)$. The differential of the cost function is assumed to be of the form (1.6) with $G(\Omega)$ the shape gradient. Thus, as described in [7] Chapter 1 on page 54, the gradient of the cost function, ∇J, is a regularised extension of $G(\Omega)n$, i.e

$$\nabla J(\Omega) = G(\Omega)n \quad \text{on } \Gamma, \tag{1.10}$$

with n the external unit normal. Let's determine Dg^T. First at all, let $e = e_1$ be the canonical basis of $\mathbb{R}$. Then, by linearity of differential, we have

$$\forall \mu \in \mathbb{R} \quad Dg^T(\Omega)(\mu) = \mu_1 Dg^T(\Omega)(e_1).$$

with $\mu = \mu_1 e_1$. Then, by taking $\mu = e_1$, for all $\phi \in V$ and by using Definition 1.2.1, we obtain

$$\langle Dg^T(\Omega)(e_1), \phi \rangle_V = e_1 Dg(\Omega)(\phi),$$

in other words,

$$\int_\Omega \gamma^2 \nabla \left(Dg^T(\Omega)(e_1) \right) : \nabla \phi + Dg^T(\Omega)(e_1) \cdot \phi = \int_\Gamma \phi \cdot n.$$

We want to solve

$$\int_\Omega \gamma^2 \nabla U : \nabla \phi + U \cdot \phi = \int_\Gamma \phi \cdot n,$$

for all ϕ in V with $U = Dg^T(\Omega)(e_1)$. By using a integration of parts formula, it follows that

$$\int_\Omega \left(-\gamma^2 \Delta U + U \right) \cdot \phi = \int_\Gamma \left(I - \gamma^2 \nabla U \right) n \cdot \phi - \int_{\Gamma_{fixed}} \gamma^2 \nabla U n \cdot \phi.$$

By taking $\nabla U n = I \frac{n}{\gamma^2}$ on Γ and $\nabla U n = 0$ on Γ_{fixed}, we therefore need to solve the following problem

$$\begin{cases} -\gamma^2 \Delta (Dg^T(\Omega)(e_1)) + Dg^T(\Omega)(e_1) = 0 & \text{in } \Omega, \\ \frac{\partial Dg^T(\Omega)(e_1)}{\partial n} = \frac{1}{\gamma^2} n & \text{on } \Gamma, \\ \frac{\partial Dg^T(\Omega)(e_1)}{\partial n} = 0 & \text{on } \Gamma_{fixed}. \end{cases} \tag{1.11}$$

Then, we have

$$(Dg(\Omega) Dg^T(\Omega))^{-1} g(\Omega) = \frac{g(\Omega)}{\int_{\partial\Omega} Dg^T(\Omega)(e_1) \cdot n},$$

and

$$(Dg(\Omega) Dg^T(\Omega))^{-1} Dg(\Omega) \nabla J(\Omega) = \frac{\left(\int_{\partial\Omega} \nabla J(\Omega) \cdot n \right)}{\int_\Omega Dg^T(\Omega)\,(e_1) \cdot n}.$$

Finally, the null space and range space directions can be expressed as

$$\xi_C(\Omega)(X) = \left(\frac{g(\Omega)}{\int_{\partial\Omega} Dg^T(\Omega)(e_1) \cdot n} \right) Dg^T(\Omega)\,(e_1)\,(X), \tag{1.12}$$

and

$$\xi_J(\Omega)(X) = \nabla J(\Omega)(X) - \frac{\left(\int_{\partial\Omega} \nabla J(\Omega) \cdot n \right) Dg^T(e_1)(X)}{\int_\Omega Dg^T(\Omega)\,(e_1) \cdot n}. \tag{1.13}$$

All the steps above are valid when there are several equality constraints, i.e. $g : V \to \mathbb{R}^p$ with $p \geq 1$. Furthermore, it is possible to extend this method to inequality constraints as explained in [7].

Numerical Implementation : For the numerical implementation, a domain sequence, Ω_k, is computed which verifies the following constraints

$$\partial\Omega_k = \Gamma_k \cup \Gamma_{fixed},$$

and

$$\Omega_{k+1} = (\mathrm{Id} + \theta_k)(\Omega_k).$$

Using the NSGF method, we define the displacement field θ_k over all Ω_k such that

$$\begin{cases} -\Delta\theta_k = 0 & \text{in } \Omega_k \\ \theta_k = 0 & \text{on } \Gamma_{fixed} \\ \frac{\partial\theta_k}{\partial n} = -\alpha_C \xi_C(\Omega_k) - \alpha_J \xi_J(\Omega_k) & \text{on } \Gamma_k. \end{cases}$$

1.3 Linear-elasticity : the cantilever

We consider a homogeneous isotropic elastic solid which occupies a bounded domain Ω in $\mathbb{R}^2$. We suppose that the boundary is composed of three parts

$$\partial\Omega = \Gamma \cup \Gamma_N \cup \Gamma_D$$

where Γ is a variable part traction-free (homogeneous Neumann), Γ_D is a fixed part on which the solid is fixed (homogeneous Dirichlet) and Γ_N is a fixed part on which f forces are applied (inhomogeneous Neumann). The cantilever problem is illustrated in the Figure 4. In the following, we will study the example of the cantilever. The displacement $u : \mathbb{R}^N \to \mathbb{R}^N$ is the solution of the system

$$\begin{cases} -\nabla \cdot \sigma = 0 & \text{in } \Omega, \\ \sigma = 2\mu e(u) + \lambda tr(e(u))I & \text{in } \Omega, \\ u = 0 & \text{on } \Gamma_D, \\ \sigma n = f & \text{on } \Gamma_N, \\ \sigma n = 0 & \text{on } \Gamma, \end{cases} \tag{1.14}$$

with $\sigma(u) \in M_N(\mathbb{R})$ the stress tensor, $\lambda, \mu > 0$ the Lamé coefficients of the material and $e(u) = \frac{1}{2}\left(\nabla u + \nabla u^T\right)$ the deformation tensor. We wish to solve the following minimisation problem

$$\inf_{\Omega \in \Omega_{ad}} \left\{ J(\Omega) = \int_{\Gamma_N} f \cdot u \right\} \tag{1.15}$$

where the set of admissible forms is defined as

$$\Omega_{ad} = \{\Omega \in C(\Omega_0) \mid \Gamma_D \cup \Gamma_N \subset \partial\Omega,\ g(\Omega) = 0\} \tag{1.16}$$

with g defined by (1.7). It should be noted that the domain of integration of the cost function does not depend on Ω because Γ_N is fixed. Thus, the dependence with respect to the domain is ensured only through the solution of the previous model.

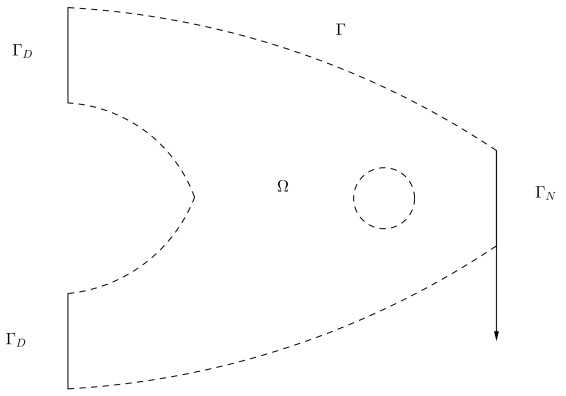

Figure 4: Illustration of the 2D cantilever with a hole. The Γ_D and Γ_N boundaries (Dirichlet condition and Neumann condition) are fixed. The Γ boundary (of which the hole is a part) is movable in order to optimize the shape.

1.3.1 Theoretical formulation of the problem

In the following, we will denote

$$H^k_\Gamma(\Omega) = \left\{ v \in H^k(\Omega) \mid v|_\Gamma = 0 \right\},$$

the set of functions belonging to $H^k(\Omega)$ and which vanish on Γ. Let us start by defining the weak formulation of (1.14) in order to find the dual problem and the gradient of the cost function. The divergence of σ is given by

$$\nabla \cdot \sigma = \left(\sum_{j=1}^{N} \frac{\partial \sigma_{ij}}{\partial x_j} \right)_{1 \leq i \leq N}.$$

Let $u \in H^1_{\Gamma_N}(\Omega)$ solution of (1.14). By using the fact that $tr(e(u)) = \nabla \cdot u$, the equation can be written for $1 \leq i \leq N$ as

$$-\sum_{j=1}^{N} \frac{\partial}{\partial x_j} \left[\mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \lambda (\nabla \cdot u) \delta_{ij} \right] = 0.$$

By multiplying by a test function $\phi \in H^1_{\Gamma_D}(\Omega)$ and integrating over the domain Ω, we have

$$\int_\Omega \sum_{j=1}^{N} \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_j} \right) \frac{\partial \phi_i}{\partial x_j} + \int_\Omega \lambda (\nabla \cdot u) \frac{\partial \phi_i}{\partial x_i} = \int_{\Gamma_N} f_i \phi_i.$$

In addition, we have the following result

$$\sum_{i,j=1}^{N} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \frac{\partial \phi_i}{\partial x_j} = \frac{1}{2} \sum_{i,j=1}^{N} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \left(\frac{\partial \phi_i}{\partial x_j} + \frac{\partial \phi_j}{\partial x_j} \right) = 2e(u) : e(\phi) \quad (1.17)$$

Finally, summing over the index i in the previous formulation and using this result, we obtain

$$\int_\Omega 2\mu e(u) : e(\phi) + \int_\Omega \lambda(\nabla \cdot u)(\nabla \cdot \phi) = \int_{\Gamma_N} f \cdot \phi. \tag{1.18}$$

Cea's method for determining the dual problem and the gradient of the cost function : We denote the Lagrangian of this problem by

$$\begin{array}{rcl} \mathcal{L} : \Omega_{ad} \times \left(H^1_{\Gamma_D}(\mathbb{R}^N)\right)^2 & \to & \mathbb{R} \\ (\Omega, v, q) & \mapsto & \int_\Omega 2\mu e(v) : e(q) + \int_\Omega \lambda(\nabla \cdot v)(\nabla \cdot q) - \int_{\Gamma_N} f \cdot q + \int_{\Gamma_N} f \cdot v. \end{array}$$

It should be noted that the different variables of $\mathcal{L}$ are independent because we consider Lagrange multipliers on $H^1_{\Gamma_D}(\mathbb{R}^N)$ and not $H^1_{\Gamma_D}(\Omega)$ and especially that Γ_N is independent of Ω because it is fixed.

Dual problem : Let us calculate the various partial derivatives of $\mathcal{L}$. Let's start with the partial derivative with respect to q in the $\phi \in H^1_{\Gamma_D}(\mathbb{R}^N)$ direction

$$\left\langle \frac{\partial \mathcal{L}}{\partial q}(\Omega, v, q), \phi \right\rangle = \int_\Omega 2\mu e(v) : e(\phi) + \int_\Omega \lambda(\nabla \cdot v)(\nabla \cdot \phi) - \int_{\Gamma_N} f \cdot \phi,$$

which, when it vanishes, corresponds to the variational formulation of the primal problem (1.18). Now, the partial derivative w.r.t v : let $\phi \in H^1_{\Gamma_D}(\mathbb{R}^N)$

$$\left\langle \frac{\partial \mathcal{L}}{\partial v}(\Omega, v, q), \phi \right\rangle = \int_\Omega 2\mu e(\phi) : e(q) + \int_\Omega \lambda(\nabla \cdot \phi)(\nabla \cdot q) + \int_{\Gamma_N} f \cdot \phi,$$

which, when it vanishes, corresponds to the weak formulation of the dual problem :

$$\begin{cases} -\nabla \cdot \sigma = 0 & \text{on } \Omega \\ \sigma = 2\mu e(p) + \lambda tr(e(p))I & \text{on } \Omega \\ p = 0 & \text{on } \Gamma_D \\ \sigma n = -f & \text{on } \Gamma_N \\ \sigma n = 0 & \text{on } \Gamma \end{cases}$$

Note that this corresponds exactly to the primal problem with a minus sign. This is due in particular to the choice of J and the boundary conditions of the primal problem. Indeed, we have J which depends on f on Γ_N and the boundary conditions are all null except on Γ_N which is equal to f. Thus, we can avoid solving the dual problem in this case and simply use the solution of the primal problem.

Shape gradient : Finally, let's calculate the partial derivative of $\mathcal{L}$ with respect to the Ω domain in the θ direction

$$\frac{\partial \mathcal{L}}{\partial \Omega}(\Omega, v, q)(\theta) = \int_{\partial\Omega} \theta \cdot n \left[2\mu e(v) : e(q) + \lambda(\nabla \cdot v)(\nabla \cdot q)\right]$$

by using Proposition 1.1.1. When we evaluate this derivative with the state $u(\Omega)$ and the adjoint state $p(\Omega) = u(\Omega)$, we find exactly the value of the derivative of the cost function

$$\frac{\partial \mathcal{L}}{\partial \Omega}(\Omega, u(\Omega), p(\Omega))(\theta) = -\int_\Gamma \theta \cdot n \left[2\mu \|e(u)\|^2 + \lambda tr(e(u))^2\right] = DJ(\Omega)(\theta), \tag{1.19}$$

as explained here (1.5). Thus by definition, in (1.19) we obtain the following shape gradient

$$G(\Omega) = -2\mu \|e(u)\|^2 - \lambda (tr(e(u)))^2. \tag{1.20}$$

1.3.2 Experimental Evaluation

In this section, we will present various results on the optimization problem concerning the shape of cantilevers. The initial domain considered is a trapezoid with two feet in the 2D case (four feet in the 3D case). However, it is important to note that any other shape could have been selected as long as it shared the same fixed boundaries. The choice of a trapezoid as the initial shape is advantageous due to its simplicity and its ability to approximate the solution of the problem. Throughout this section, unless specified otherwise, all results have been obtained using $\mathbb{P}_1$ continuous finite elements. All simulations are carried out using the classic gradient descent method. The NSGF method is used in the following example, a rigid body in a Stokes fluid.

2D simulations for various types of cantilever : In this section, we present the application of shape optimization to a 2D cantilever with two pillars, as illustrated in Figure 4. The specific parameter values used in the analysis are detailed in Table 1.

Symbol	Value (dimensionless)
λ	$50/9$
μ	$350/27$
H	9
L_1	8
L_2	2
f	$(0, -1)$

Table 1: Geometric parameters and Lamé coefficient of the initial domain for the 2D case of the cantilever. The Lamé coefficients are λ and μ. H represents the height of the trapezoid, L_1 the large base and L_2 the small base. The force applied to the curve Γ_N is defined by f.

No hole : We begin by considering the simplest case, which is the cantilever without any holes, as depicted in Figures 6 and 5. To ensure clear visibility of the mesh in print, a discretization parameter of $h = 0.4$ is set. However, it is important to note that employing a finer discretization would result in more accurate outcomes. For the

initialization of the Lagrange multiplier, we choose $l = 0.5$. Additionally, the values of $a = b = 0.5$ and $c = 10$ are set, along with a descent step of $t = 0.2$. These specific values have been determined through empirical testing to achieve optimal performance.

$J(\Omega_0)$	$J(\Omega_{n_{final}})$	$\lvert\lvert\Omega_0\rvert - \lvert\Omega_{n_{final}}\rvert\rvert$	$\lVert\theta_{n_{final}}\rVert_{H^1}$	n_{final}
4.6834	3.1346	$1.3630e-2$	$1.696e-2$	125

Table 2: Comparison of the cost function between the initial domain and the final domain, the volume error between the two domains, the H^1-norm of the displacement field at the end and the number of iterations for the 2D cantilever without hole.

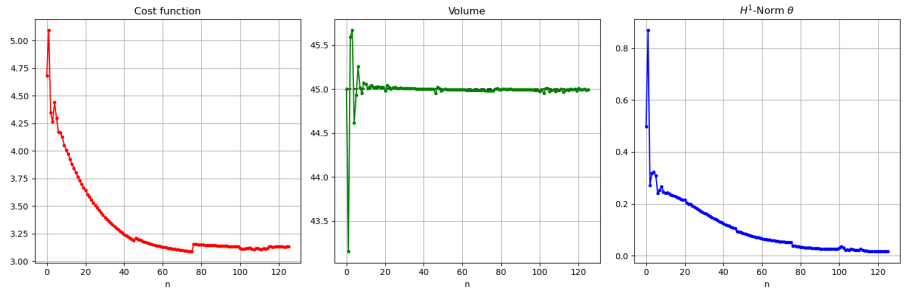

Figure 5: Evolution of the cost function, the volume of the domain Ω_k and the norm of θ_k of the 2D cantilever without hole with the following parameters: $h = 0.4$, $t = 0.2$, $l = 0.5$, $a = b = 0.5$ and $c = 10$. The iterations range from 0 to 125.

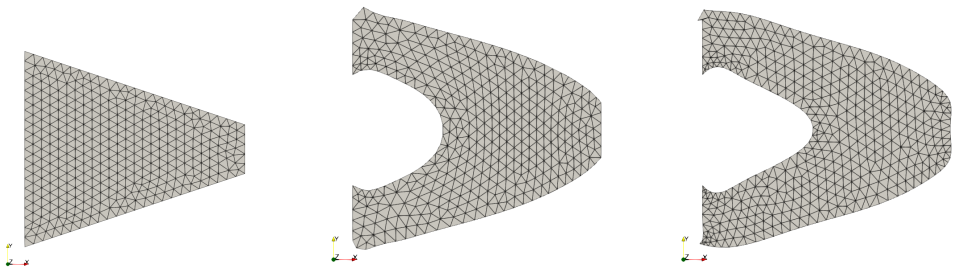

Figure 6: Visualisation of the results obtained during different iterations of the algorithm for the shape optimisation of the 2D cantilever without hole with the following parameters: $h = 0.4$, $t = 0.2$, $l = 0.5$, $a = b = 0.5$ and $c = 10$. The initial domain ($k = 0$) is shown on the left. The intermediate domain ($k = 60$) is displayed in the middle. The final domain ($k = 125$) is displayed on the right.

One hole : The second test consists of studying the cantilever with a single hole. The

results obtained are presented in Figures 8 and 7. We use a discretization parameter of $h = 0.4$. For the Lagrange multiplier, we initialize with $l = 0.5$, and set $a = b = 0.5$ and $c = 10$ with a descent step of $t = 0.2$.

$J(\Omega_0)$	$J(\Omega_{n_{final}})$	$\lvert\lvert\Omega_0\rvert - \lvert\Omega_{n_{final}}\rvert\rvert$	$\lVert\theta_{n_{final}}\rVert_{H^1}$	n_{final}
4.7035	3.2162	$2.9980e-3$	$1.8906e-2$	125

Table 3: Comparison of the cost function between the initial domain and the final domain, the volume error between the two domains, the H^1-norm of the displacement field at the end and the number of iterations for the one hole 2D cantilever.

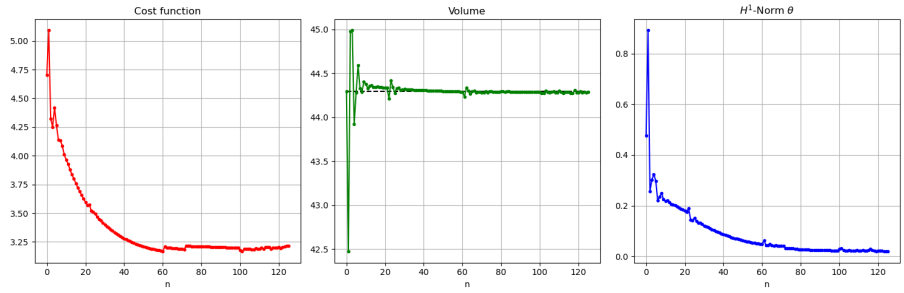

Figure 7: Evolution of the cost function, the volume of the domain Ω_k and the norm of θ_k of the 2D cantilever with one hole with the following parameters: $h = 0.4$, $t = 0.2$, $l = 0.5$, $a = b = 0.5$ and $c = 10$. The iterations range from 0 to 125.

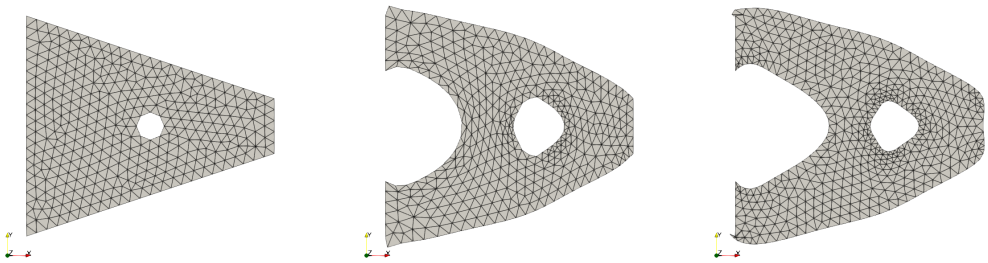

Figure 8: Visualisation of the results obtained during different iterations of the algorithm for the shape optimisation of the 2D cantilever with one hole with the following parameters: $h = 0.4$, $t = 0.2$, $l = 0.5$, $a = b = 0.5$ and $c = 10$. The initial domain ($k = 0$) is shown on the left. The intermediate domain ($k = 60$) is displayed in the middle. The final domain ($k = 125$) is displayed on the right.

Four holes : The last type of cantilever studied is one with four holes in its domain. The

results are displayed in Figures 10 and 9. We use a discretization parameter of $h = 0.4$. For the Lagrange multiplier, we initialize with $l = 0.5$, and set $a = b = 0.5$ and $c = 10$ with a descent step of $t = 0.2$.

$J(\Omega_0)$	$J(\Omega_{n_{final}})$	$\lvert\lvert\Omega_0\rvert - \lvert\Omega_{n_{final}}\rvert\rvert$	$\lVert\theta_{n_{final}}\rVert_{H^1}$	n_{final}
5.1589	3.3770	$2.5317e-3$	$1.5615e-2$	125

Table 4: Comparison of the cost function between the initial domain and the final domain, the volume error between the two domains, the H^1-norm of the displacement field at the end and the number of iterations for the four holes 2D cantilever.

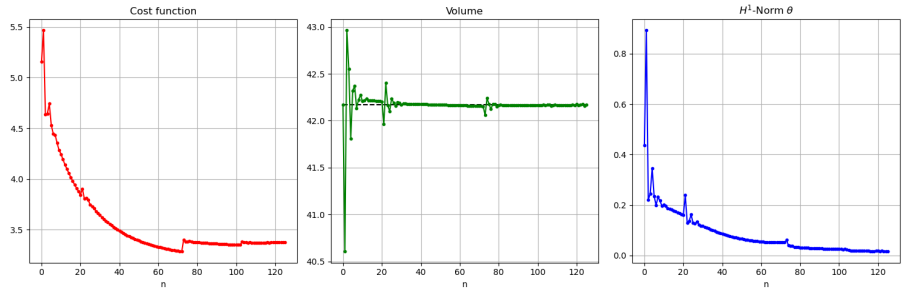

Figure 9: Evolution of the cost function, the volume of the domain Ω_k and the norm of θ_k of the 2D cantilever with 4 holes with the following parameters: $h = 0.4$, $t = 0.2$, $l = 0.5$, $a = b = 0.5$ and $c = 10$. The iterations range from 0 to 125.

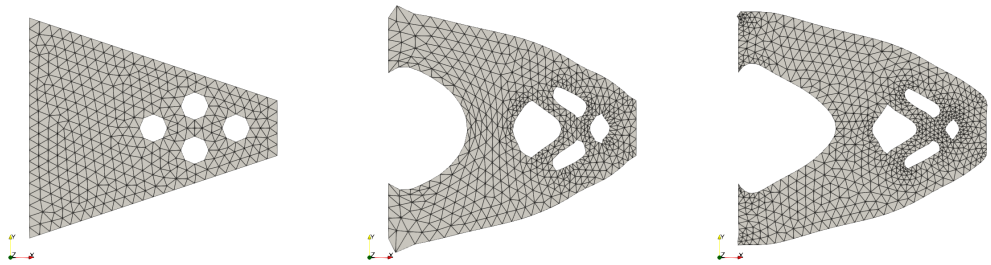

Figure 10: Visualisation of the results obtained during different iterations of the algorithm for the shape optimisation of the 2D cantilever with 4 holes with the following parameters: $h = 0.4$, $t = 0.2$, $l = 0.5$, $a = b = 0.5$ and $c = 10$. The initial domain ($k = 0$) is shown on the left. The intermediate domain ($k = 60$) is displayed in the middle. The final domain ($k = 125$) is displayed on the right.

The results presented above demonstrate that satisfactory outcomes can be achieved us-

ing the proposed method. Notably, the obtained shapes closely resemble those reported in various papers that focus on optimizing the geometrical shape of cantilevers, such as [1]. Furthermore, we successfully decrease the cost function while approaching the initial volume of the domain, which aligns precisely with the desired behavior. It is worth noting that potential geometric instabilities, in the form of spikes, may emerge on Γ_D after a certain number of iterations. These spikes are also observed in the 3D case, as illustrated in the subsequent figures.

3D simulations for various types of cantilever : In this section, we address the case of the 3D cantilever, which introduces additional complexities compared to the 2D case and results in longer computation times. Incorporating holes into the structure also increases the risk of encountering mesh superposition issues. The overall geometry of the 3D cantilever remains similar to the 2D case, with the addition of four pillars, and boundary conditions are now applied to surfaces instead of curves. The specific parameter values used in the analysis are provided in detail in Table 5.

Symbol	Value (dimensionless)
λ	$50/9$
μ	$350/27$
H	9
C_1	8
C_2	2
f	$(0, -1, 0)$

Table 5: Geometric parameters and Lamé coefficient of the initial domain for the 2D case of the cantilever. The Lamé coefficients are λ and μ. H represents the height of the truncated pyramid, C_1 the side of the largest square base and C_2 the side of the smallest square base. The force applied to the surface Γ_N is defined by f.

No hole : Let's first consider the case without a hole. We use a discretization parameter of $h = 0.8$. For the coefficients affecting the optimization problem, we set $l = 0.5$, $a = b = 0.5$, $c = 3$, and choose a descent step t of 0.1. The results of this test are presented in Figures 12 and 11.

$J(\Omega_0)$	$J(\Omega_{n_{final}})$	$\lvert\lvert\Omega_0\rvert - \lvert\Omega_{n_{final}}\rvert\rvert$	$\lVert\theta_{n_{final}}\rVert_{H^1}$	n_{final}
4.8060	3.0484	0.1439	$2.1647e-2$	400

Table 6: Comparison of the cost function between the initial domain and the final domain, the volume error between the two domains, the H^1-norm of the displacement field at the end and the number of iterations for the no hole 3D cantilever.

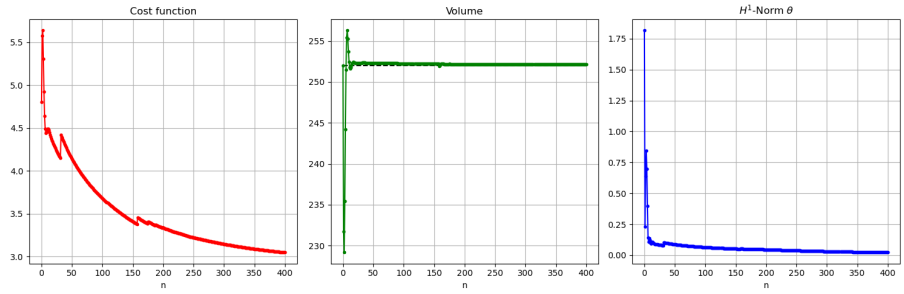

Figure 11: Evolution of the cost function, the volume of the domain Ω_k and the norm of θ_k of the 3D cantilever without hole with the following parameters: $h = 0.8$, $t = 0.1$, $l = 0.5$, $a = b = 0.5$ and $c = 2$. The iterations range from 0 to 400.

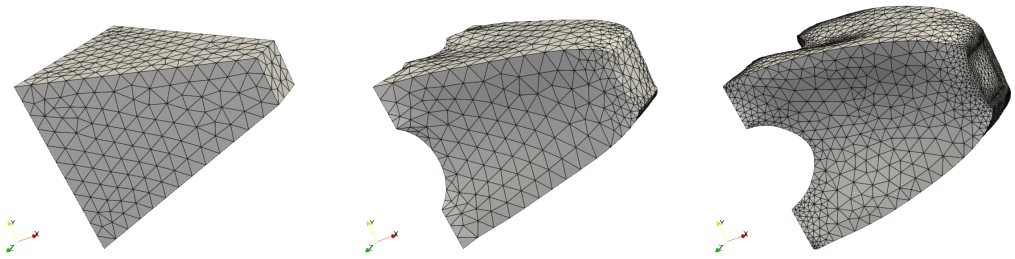

Figure 12: Visualisation of the results obtained during different iterations of the algorithm for the shape optimisation of the 3D cantilever without hole with the following parameters: $h = 0.8$, $t = 0.1$, $l = 0.5$, $a = b = 0.5$ and $c = 2$. The initial domain ($k = 0$) is shown on the left. The intermediate domain ($k = 150$) is displayed in the middle. The final domain ($k = 400$) is displayed on the right.

One hole : In the next test, we add a hole inside the material. The results are presented in Figures 14 and 13. We use a discretization parameter of $h = 0.8$. The other coefficients used are $l = 0.5$, $a = b = 0.5$, $c = 2$, and $t = 0.08$.

$J(\Omega_0)$	$J(\Omega_{n_{final}})$	$\lvert\lvert\Omega_0\rvert - \lvert\Omega_{n_{final}}\rvert\rvert$	$\lVert\theta_{n_{final}}\rVert_{H^1}$	n_{final}
6.4813	3.7726	0.2582	$4.7253e-2$	250

Table 7: Comparison of the cost function between the initial domain and the final domain, the volume error between the two domains, the H^1-norm of the displacement field at the end and the number of iterations for the one hole 3D cantilever.

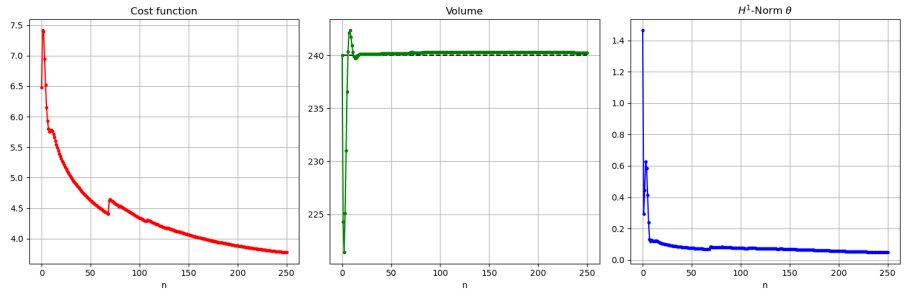

Figure 13: Evolution of the cost function, the volume of the domain Ω_k and the norm of θ_k of the 3D cantilever with holes with the following parameters: $h = 0.8$, $t = 0.08$, $l = 0.5$, $a = b = 0.5$ and $c = 2$. The iterations range from 0 to 250.

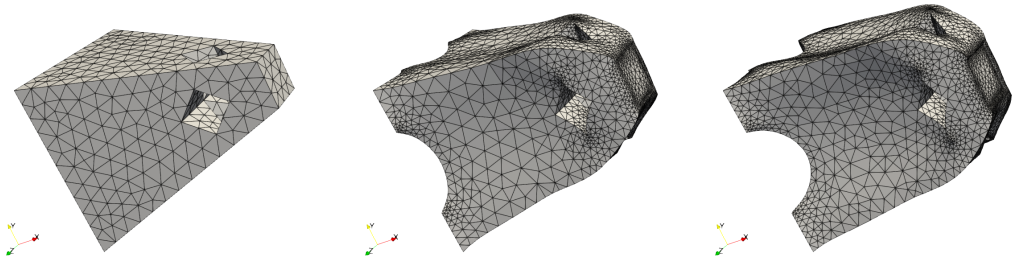

Figure 14: Visualisation of the results obtained during different iterations of the algorithm for the shape optimisation of the 3D cantilever with holes with the following parameters: $h = 0.8$, $t = 0.08$, $l = 0.5$, $a = b = 0.5$ and $c = 2$. The initial domain ($k = 0$) is shown on the left. The intermediate domain ($k = 150$) is displayed in the middle. The final domain ($k = 250$) is displayed on the right.

It is important to note that the chosen optimal shape in the simulations effectively reduces the cost function while maintaining the volume. However, due to limitations, we had to prematurely halt the simulations. If allowed to continue, the cost function would have been further minimized, resulting in a more optimal shape. Nonetheless, the obtained shape presents several issues. Specifically, certain areas such as the feet and top part exhibit more prominent outgrowths, which can be attributed to volume conservation. The shape optimization process tends to hollow out the cantilever below the desired volume (between the four legs). As a result, to compensate for this volume loss, protrusions seem to emerge after a certain number of iterations. Another significant issue encountered is mesh collision and overlap. As illustrated in Figure 15, the mesh nodes intertwine at the hole, deviating from the desired behavior. Additionally, the discontinuities observed in the curves of the different simulations correspond to the remeshing that occurs during the iterations.

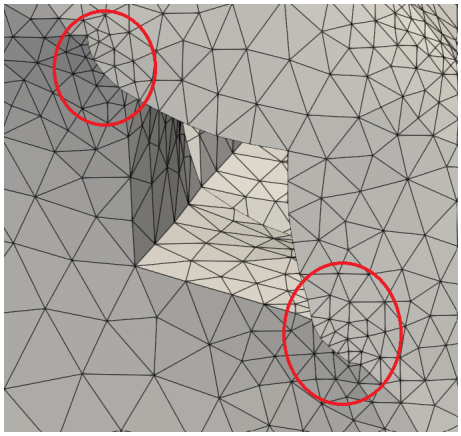

Figure 15: Example of mesh collision for the 3D cantilever with holes with the following parameters: $h = 0.8$, $t = 0.08$, $l = 0.5$, $a = b = 0.5$ and $c = 2$ at the 250th iteration.

1.4 A rigid body in a Stokes flow

We consider a rigid object S set in motion into an incompressible fluid with viscosity μ at low Reynolds number. The fluid occupies a bounded domain Ω in $\mathbb{R}^N$. We suppose that the boundary is composed of two parts

$$\partial\Omega = \Gamma_S \cup \Gamma_B,$$

where Γ_S is a variable part associated with the boundary of the rigid object and Γ_B is a fixed part corresponding to the boundary of the domain containing the fluid. Normally, we would consider an edge at an infinite distance from the object. However, in finite elements we are limited for this kind of hypothesis. It is impossible to consider an infinite wall, and the further away the edge of the domain containing the fluid is, the more elements are needed to mesh it, which has a huge impact on calculation time, particularly in 3D. We will therefore study shape optimisation in a context that is slightly different from [12]. The Stokes problem is illustrated in the Figure 16. Taking the same conditions as in [12], we consider a linear background flow U^∞ and U the translational velocities of the object. The fluid velocity field $u : \mathbb{R}^N \to \mathbb{R}^N$ and the pressure field $p : \mathbb{R}^N \to \mathbb{R}$ is the solution of the Stokes equations

$$\begin{cases} -\mu\Delta u + \nabla p = 0 & \text{in } \Omega, \\ \nabla \cdot u = 0 & \text{in } \Omega, \\ u = U & \text{on } \Gamma_S, \\ u = U^\infty & \text{on } \Gamma_B. \end{cases} \tag{1.21}$$

We wish to solve the following minimisation problem

$$\inf_{\Omega \in \Omega_{ad}} \left\{ J_\alpha(\Omega) = \int_{\Gamma_S} \sigma(u,p)n \cdot \alpha \right\} \tag{1.22}$$

where n is the normal to Γ_S pointing *outwards with respect to the fluid domain* (therefore inward w.r.t the object) and σ is the stress tensor defined as

$$\sigma(u, p) = -pI + 2\mu e(u)$$

where e is the deformation tensor, given by

$$e(u) = \frac{1}{2}\left(\nabla u + \nabla u^T\right).$$

In paper [12], the definition of the normal follows the opposite convention. Thus, one must consider $-n$ to pass from our formulation to the one in [12]. Depending on the choice of α, we are able to minimise certain coefficients associated with the large resistance tensor, which depends only on the shape of the object (see [12] for more details). We define the set of admissible forms as

$$\Omega_{ad} = \{\Omega \in C(\Omega_0) \mid \Gamma_B \subset \partial\Omega,\ g(\Omega) = 0\} \tag{1.23}$$

where g is defined as in (1.7). It should be noted, that unlike the cantilever case, now the cost function depends on the domain.

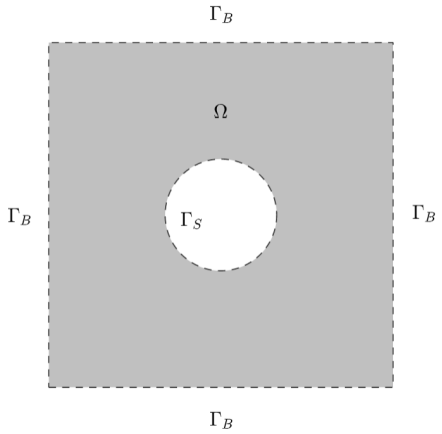

Figure 16: Illustration of the stokes problem. The Γ_B boundaries is fixed. The Γ_S boundary is movable in order to optimize the shape.

By taking precise values for U, α and U^∞, one can express the cost function as an input to the large strength tensor presented in Table 8 and explained in [12]. This brings us back to a problem of resistance: what is the optimal shape to have the least possible resistance on the fluid. They correspond with the projection of the hydrodynamic drag force and torque - exerted by the moving particle to the fluid - along one of the basis vectors.

J_α	U	α	U^∞
K_{ij}	e_j	e_i	0
Q_{ij}	$e_j \wedge (x,y,z)^T$	$e_i \wedge (x,y,z)^T$	0
C_{ij}	$e_j \wedge (x,y,z)^T$	e_i	0

Table 8: Cost function J_α associated with the entries of the grand resistance tensor (see [12]) with respect to the choice of U, α and U^∞.

The problem setup is well presented in Figure 17 taken from the article [12]. Several examples of resistance problem are illustrated on the right side of the figure for different entries of the grand resistance tensor.

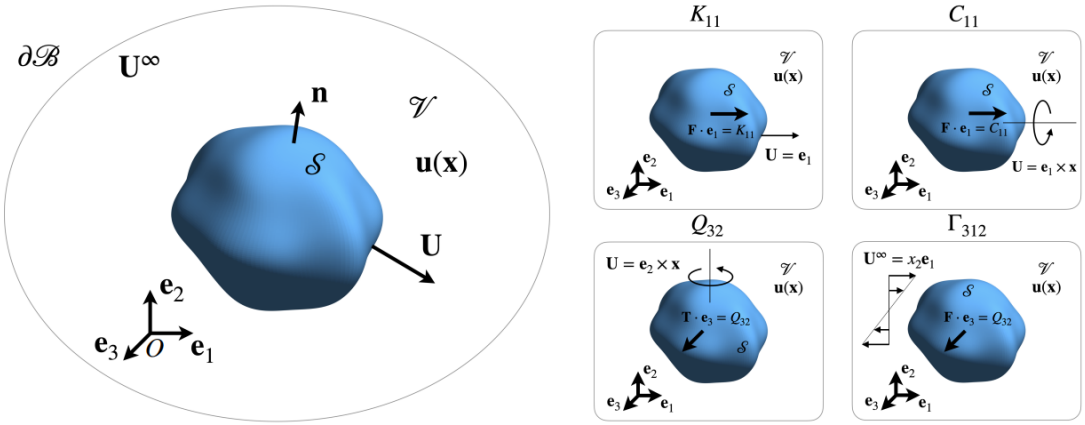

Figure 17: Problem setup taken from [12] : according the previous notation, we have $S = \mathcal{S}$, $B = \mathcal{B}$, $\Omega = \mathcal{V}$ and the opposite direction of the normal, i.e $-n$. The left figure illustrate the Stokes problem in three dimension. In the right side, several examples of resistance problem for different entries of the grand resistance tensor are presented.

1.4.1 Theoretical formulation of the problem

Let us start by defining the following property which is an important result by integration of parts in the field of fluid mechanics and which will be of great help to us in the future.

Proposition 1.4.1. *Let v and ϕ in $H^1(\mathbb{R}^N)$, we have*

$$-\int_\Omega (\Delta v + \nabla(\nabla \cdot v)) \cdot \phi = 2\int_\Omega e(v) : e(\phi) - 2\int_{\partial\Omega} e(v) n \cdot \phi.$$

Proof. Let v and ϕ in $H^1(\mathbb{R}^N)$. First of all, by integration by parts we obtain the following relation

$$-\int_\Omega \sum_{j=1}^N \frac{\partial}{\partial x_j}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\phi_i = \int_\Omega \sum_{j=1}^N \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\partial \phi_i}{\partial x_j} - \int_{\partial\Omega} \sum_{j=1}^N \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\phi_i n_j$$

By summing over the index i on the left and right, and using the relation (1.17), we finally have that

$$-\int_\Omega (\Delta v + \nabla(\nabla \cdot v)) \cdot \phi = 2\int_\Omega e(v) : e(\phi) - \int_{\partial\Omega} 2e(v)n \cdot \phi.$$

□

Cea's method for determining the dual problem and the gradient of the cost function : Since we are working with Dirichlet conditions, we must use the form of the equation (1.21) directly for the variational formulation and add to the Lagrangian, Lagrange multipliers associated with Dirichlet constraints. We denote the Lagrangian of this problem by

$$\begin{array}{rcl} \mathcal{L}_\alpha : \Omega_{ad} \times \left(H^1(\mathbb{R}^N)\right)^4 \times \left(L^2(\mathbb{R}^N)\right)^2 & \to & \mathbb{R} \\ (\Omega, v, h1, \lambda, \beta, q, h2) & \mapsto & \int_{\Gamma_S} \sigma(v,q)n \cdot \alpha + \int_\Omega [-\mu\Delta v + \nabla q] \cdot h_1 \\ & & + \int_\Omega (\nabla \cdot v)h_2 + \int_{\Gamma_S} \lambda \cdot (v - U) \\ & & + \int_{\Gamma_B} \beta \cdot (v - U^\infty) \end{array}$$

It is easy to see that the variables are independent of each other.

Dual problem : Let's start by determining the dual problem. In all the following, for the sake of readability, we will omit the different variables of the Lagrangian. Let's derive $\mathcal{L}_\alpha$ w.r.t q. For all $\phi \in H^1(\mathbb{R}^N)$, we have

$$\left\langle \frac{\partial \mathcal{L}_\alpha}{\partial q}, \phi \right\rangle = \int_{\Gamma_S} \sigma(0,\phi)n \cdot \alpha + \int_\Omega \nabla\phi \cdot h_1.$$

Using a integration by parts formula, then

$$\left\langle \frac{\partial \mathcal{L}_\alpha}{\partial q}, \phi \right\rangle = -\int_{\Gamma_S} \phi(n \cdot \alpha) - \int_\Omega \phi\nabla \cdot h_1 + \int_{\partial\Omega} \phi(n \cdot h_1).$$

Taking ϕ with compact support in Ω, the variables $h_1 = u^*$ and $h_2 = p^*$ and using (1.3), we finally have

$$\nabla \cdot u^* = 0 \qquad \text{in } \Omega. \tag{1.24}$$

Let's derive $\mathcal{L}_\alpha$ w.r.t v. For all $\phi \in H^1(\mathbb{R}^N)$, then

$$\left\langle \frac{\partial \mathcal{L}_\alpha}{\partial v}, \phi \right\rangle = \int_{\Gamma_S} \sigma(\phi,0)n \cdot \alpha + \int_\Omega -\mu\Delta\phi \cdot h_1 + h_2\nabla \cdot \phi + \int_{\Gamma_S} \lambda \cdot v + \int_{\Gamma_B} \beta \cdot v. \tag{1.25}$$

Using twice integration by parts, we obtain the following formulation

$$\left\langle \frac{\partial \mathcal{L}_\alpha}{\partial v}, \phi \right\rangle = -\int_\Omega [\mu\Delta h_1 + \nabla h_2] \cdot \phi + \int_{\partial\Omega} \mu\nabla h_1 n \cdot \phi - \mu\nabla\phi n \cdot h_1 + h_2(n \cdot \phi) + \int_{\Gamma_S} \lambda \cdot v + 2\mu e(\phi)n \cdot \alpha + \int_{\Gamma_B} \beta \cdot \phi.$$

Taking ϕ with compact support in Ω, the variables $h_1 = u^*$ and $h_2 = -p^*$ and using (1.3), even if it means changing the sign of p^*, we finally have that

$$-\mu\Delta u^* + \nabla p^* = 0 \qquad \text{in } \Omega. \tag{1.26}$$

Let us start from the equation (1.25) and apply the Proposition 1.4.1 twice in order to obtain the following result

$$\begin{aligned}\left\langle \frac{\partial \mathcal{L}_\alpha}{\partial v}, \phi \right\rangle = &- \int_\Omega \mu \left[\Delta h_1 + \nabla(\nabla \cdot h_1) + \nabla h_2\right] \cdot \phi + \int_\Omega \mu \nabla(\nabla \cdot \phi) \cdot h_1 + \int_{\partial\Omega} h_2 (n \cdot \phi) \\ &+ 2\mu \int_{\partial\Omega} e(h_1) n \cdot \phi - e(\phi) n \cdot h_1 + \int_{\Gamma_B} \beta \cdot \phi + \int_{\Gamma_S} \lambda \cdot \phi + 2\mu e(\phi) n \cdot \alpha.\end{aligned}$$

Let us take the test functions ϕ with zero divergence such that $e(\phi)n$ vanishes on $\partial\Omega$ and ϕ describes $L^2(\Gamma_S)$ (respectively $L^2(\Gamma_B)$). By taking the variables $h_1 = u^*$ and $h_2 = -p^*$ solution of (1.26) and (1.24), and using (1.3), we obtain

$$2\mu e(u^*)n + p^* n + \lambda = 0 \qquad \text{on } \Gamma_S, \tag{1.27}$$

$$2\mu e(u^*)n + p^* n + \beta = 0 \qquad \text{on } \Gamma_B. \tag{1.28}$$

Now, we take the test functions ϕ with zero divergence such that ϕ vanishes on $\partial\Omega$ and $e(\phi)n$ describes $L^2(\Gamma_S)$ (respectively $L^2(\Gamma_B)$). By taking the variables $h_1 = u^*$ and $h_2 = -p^*$ solution of (1.26) and (1.24), and using (1.3), we obtain

$$u^* = \alpha \qquad \text{on } \Gamma_S, \tag{1.29}$$

$$u^* = 0 \qquad \text{on } \Gamma_B. \tag{1.30}$$

Using the equations (1.24), (1.26), (1.27), (1.28), (1.29) and (1.30), we find the following dual problem

$$\begin{cases} -\mu\Delta u^* + \nabla p^* = 0 & \text{in } \Omega, \\ \nabla \cdot u^* = 0 & \text{in } \Omega, \\ u^* = \alpha & \text{on } \Gamma_S, \\ u^* = 0 & \text{on } \Gamma_B, \end{cases} \tag{1.31}$$

with

$$\begin{cases} \lambda^* = -2\mu e(u^*)n - p^* n & \text{on } \Gamma_S, \\ \beta^* = -2\mu e(u^*)n - p^* n & \text{on } \Gamma_B. \end{cases} \tag{1.32}$$

Unlike the cantilever, the dual problem is not the same as the primal problem. Thus, one cannot restrict oneself to solving only the primal problem. Both problems must be solved before the shape gradient can be computed.

Shape gradient : The calculation of the shape gradient requires the use of several tricks which we will summarise in three properties. Let's start with two tricks used in [12].

Proposition 1.4.2. *Let u be the solution to problem* (1.21)*, then*

$$n \cdot \nabla(u - U)n = \nabla \cdot (u - U) = 0.$$

The relation is also valid for $u = u^*$ solution of the problem (1.31)*.*

Proof. See [12, proof of Proposition 1] by taking the other convention for the normal. □

Proposition 1.4.3. *Let u be the solution to problem* (1.21) *and u^* solution to problem* (1.31)*, then*

$$e(u^*)n \cdot \nabla(u - U)n = e(u^*) : e(u - U).$$

The relation is also valid by exchanging the role of u and u^*, and by taking $U = \alpha$.

Proof. See [12, proof of Proposition 1] by taking the other convention for the normal. □

The following proposition will be very useful to simplify the expression of the shape gradient.

Proposition 1.4.4. *Let (u^*, p^*) solution to problem* (1.31)*, u solution to problem* (1.21) *and f such as*

$$f(\Omega) = -\int_{\partial\Omega} \sigma(u^*, \pm p^*)n \cdot (u - U), \tag{1.33}$$

then we have

$$Df(\Omega)(\theta) = -2\mu \int_{\Gamma_S} \theta \cdot n \left[e(u^*) : e(u - U)\right].$$

The relation is also valid by exchanging the role of (u, p) and (u^*, p^*) , and by taking $U = \alpha$.

Proof. To simplify, we will note $a_\pm = \sigma(u^*, \pm p^*)n$. Using the fact that Γ_S is fixed and Proposition 1.1.2, we obtain

$$Df(\Omega)(\theta) = -\int_{\Gamma_S} \theta \cdot n \left[\nabla \left(a_\pm \cdot (u - U)\right) \cdot n + (\nabla \cdot n)a_\pm \cdot (u - U)\right].$$

Thanks to the boundary conditions, the last term vanishes and we have

$$Df(\Omega)(\theta) = -\int_{\Gamma_S} \theta \cdot n \left[\nabla \left(a_\pm \cdot (u - U)\right) \cdot n\right].$$

However, we have that $\nabla \left(a_\pm \cdot (u - U)\right) \cdot n = (\nabla a_\pm)(u - U) \cdot n + \nabla(u - U)a_\pm \cdot n$. Thus, injecting this equation into the above integral while neglecting the terms due to the boundary conditions, we obtain

$$\begin{aligned} Df(\Omega)(\theta) &= -\int_{\Gamma_S} \theta \cdot n \left[\nabla(u - U)a_\pm \cdot n\right] \\ &= -\int_{\Gamma_S} \left[2\mu e(u^*)\nabla(u - U)n \cdot n \pm p\nabla(u - U)n \cdot n\right]. \end{aligned}$$

Proposition 1.4.2 allows us to vanish the last term when the first term can be rewritten using Proposition 1.4.3, which gives us

$$Df(\Omega)(\theta) = -2\mu \int_{\Gamma_S} \theta \cdot n \left[e(u^*) : e(u - U) \right].$$

The reasoning is exactly the same when we exchange the roles of (u, p) and (u^*, p^*), and by taking $U = \alpha$. □

Now we can move on to the calculation of the shape gradient. For this, we apply the Cea's method assuming all the necessary regularities. We thus obtain

$$\mathcal{L}_\alpha(\Omega, v, h_1, \lambda, \beta, q, h_2) = \int_{\Gamma_S} \sigma(v, q) n \cdot \alpha + \int_\Omega \left[-\mu \Delta v + \nabla q \right] \cdot h_1 + \int_\Omega (\nabla \cdot v) h_2 + \int_{\Gamma_S} \lambda \cdot (v - U) + \int_{\Gamma_B} \beta \cdot (v - U^\infty). \quad (1.34)$$

By taking as variables in the derivative of the Lagrangian the solutions of the primal problem (u, p) and of the dual problem (u^*, p^*), the terms $-\Delta u + \nabla p$ as well as the divergence of u and u^* vanish. Moreover, since the boundary Γ_B is fixed, by deriving all the integrals Γ_B cancel out. Thus, we obtain the following relationship :

$$\frac{\partial \mathcal{L}_\alpha}{\partial \Omega}(\Omega, u, u^*, \lambda^*, \beta^*, p, -p^*)(\theta) = Df(\Omega)(\theta) + Dg(\Omega)(\theta),$$

with

$$f(\Omega) = \int_{\Gamma_S} \sigma(v, q) n \cdot \alpha \quad \text{and} \quad g(\Omega) = \int_{\Gamma_S} \lambda \cdot (v - U).$$

To derive the function f and g, we use Proposition 1.4.4. Thus, we have

$$\frac{\partial \mathcal{L}_\alpha}{\partial \Omega}(\Omega, u, u^*, \lambda^*, \beta^*, p, -p^*)(\theta) = 2\mu \int_{\Gamma_S} \theta \cdot n \left[e(u) : e(\alpha) - e(u^*) : e(u - U) \right].$$

Finally, we obtain that

$$DJ(\Omega)(\theta) = -2\mu \int_{\Gamma_S} \theta \cdot n \left[e(u^*) : e(u) - e(u) : e(\alpha) - e(u^*) : e(U) \right]. \quad (1.35)$$

The choices of U and α that will be selected are presented in Table 8. Thus, we will always have

$$e(U) = e(\alpha) = 0,$$

which allows to simplify the shape gradient. We finally obtain that

$$DJ(\Omega)(\theta) = -2\mu \int_{\Gamma_S} \theta \cdot n \left[e(u^*) : e(u) \right]. \quad (1.36)$$

Thus by definition, in (1.35), we have the following shape gradient

$$G(\Omega) = -2\mu \left[e(u^*) : e(u) - e(u) : e(\alpha) - e(u^*) : e(U) \right]. \quad (1.37)$$

Of particular note, if we assume the that U and α are taking in Table 8, then

$$G(\Omega) = -2\mu e(u^*) : e(u). \quad (1.38)$$

1.4.2 Experimental Evaluation

In this section, we present various results on the Stokes optimization problem. To align with the findings in [12], we initially consider a sphere as the solid at the center of the fluid domain. It is worth noting that choosing a different shape at the beginning could lead to different local minima, as discussed in the cantilever problem. The primal and dual problems are solved using $\mathbb{P}_2$ continuous finite elements for velocity and $\mathbb{P}_1$ continuous finite elements for pressure. The expansion problem solution is also expressed with a $\mathbb{P}_1$ continuous finite elements. It is important to keep in mind that our assumptions differ from those in the paper [12]. In particular, we assume that the Γ_B edge is infinitely far from the Γ_S edge, which is not achievable using finite elements. Consequently, the obtained results may differ. The results obtained for the GD method and the NSGF method will be presented.

2D simulation for K_{11} case : After introducing the context of the resistance problem in the Stokes case, we consider the 2D K_{11} resistance problem. The initial fluid domain is a square with a spherical hole at its center. The parameter values chosen are described in Table 9, and the initial domain is illustrated in Figure 18.

Symbol	Value (dimensionless)
μ	1
C	$(0, 0)$
R	1
L	10

Table 9: Geometric and physics parameters of the initial domain for the 2D case of Stokes. The viscosity of the fluid is designed by μ. C represents the center of the sphere and the box, R the radius of the sphere and L the side length of the box.

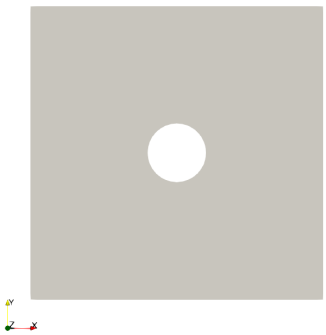

Figure 18: Initial geometry for the 2D case of Stokes.

For this specific test case, we employed an initial discretization of $h = 0.2$ around the spherical surface and $h = 1$ around the square. The optimization parameters we selected were $l = 20$, $a = b = 0.5$, and $c = 1e3$. To prevent the spherical solid from growing exponentially and filling the fluid domain, we increased the value of the parameter c. This emphasis on volume conservation yielded satisfactory results, which are presented in Table 9 and illustrated in Figure 20. In the NSGF method, remeshing plays a crucial role, particularly in the 2D case. To avoid convergence issues, we fixed the mesh at the edge of the solid, as the deformation was not significant. Consequently, the initial mesh was much more refined, with $h = 0.05$ at the solid boundary.

Method	$J(\Omega_0)$	$J(\Omega_{n_{final}})$	$\big\|\|\Omega_0\| - \|\Omega_{n_{final}}\|\big\|$	$\|\theta_{n_{final}}\|_{H^1}$	n_{final}
GD	16.7729	14.3058	$5.2101e-05$	$1.0154e-4$	500
NSGF	16.9087	14.5917	$1.1554e-3$	$3.3883e-4$	500

Table 10: Comparison of the cost function between the initial domain and the final domain, the volume error between the two domains, the H^1-norm of the displacement field at the end and the number of iterations for the resistance problem K_{11} in 2D.

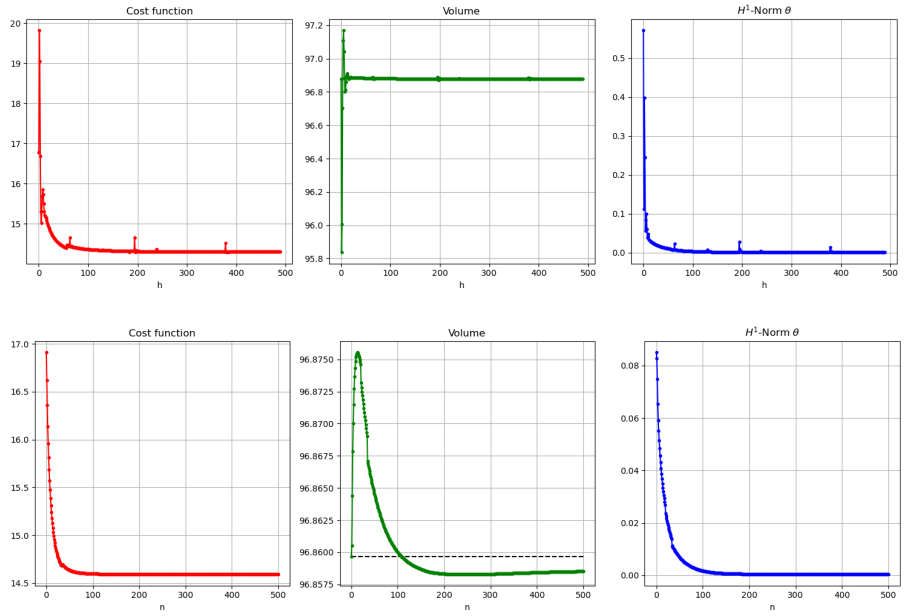

Figure 19: Evolution of the cost function, the volume of the domain Ω_n and the H^1-norm of θ_n for the 2D K_{11} resistance problem with the GD method (top) and the NSGF method (bottom).

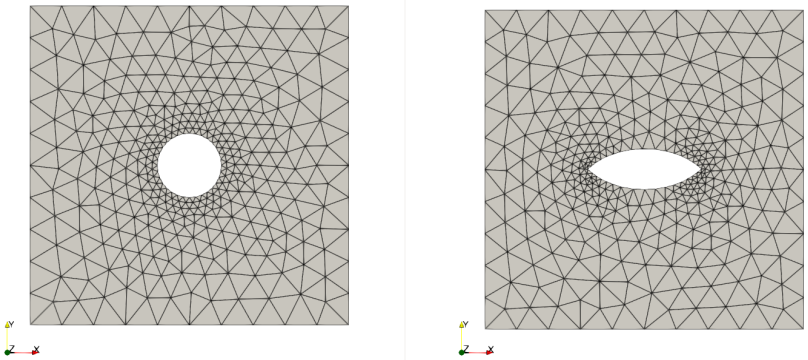

Figure 20: Initial domain (left) and final domain (right) for the GD method.

The shape obtained in 2D closely resembles a rugby ball, as described in [12]. It effectively preserves the initial volume and achieves a lower cost function compared to the initial domain. The results obtained with the two methods exhibit striking similarity. The only noticeable difference is that the NSGF method converges to a shape with extremities that resemble a lemon, as depicted in Figure 21.

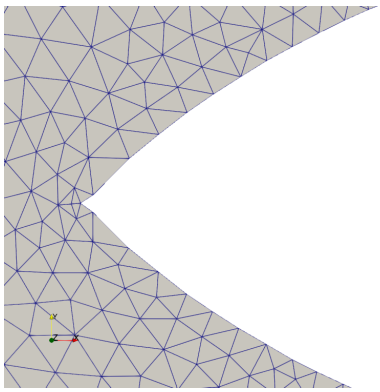

Figure 21: Left extremity of the final shape of the solid with the NSGF method.

3D simulations for various resistance problem : We now move on to the 3D case. The three-dimensional study requires more effort and time than the 2D case, particularly due to remeshing, which introduces additional nodes within the fluid. The initial fluid representation is a cube with a spherical hole at its center. The parameter values chosen are described in Table 11, and the initial domain is illustrated in Figure 22.

Symbol	Value (dimensionless)
μ	1
C	$(0, 0, 0)$
R	1
L	10

Table 11: Geometric and physics parameters of the initial domain for the 3D case of Stokes. The viscosity of the fluid is designed by μ. C represents the center of the sphere and the box, R the radius of the sphere and L the side length of the box.

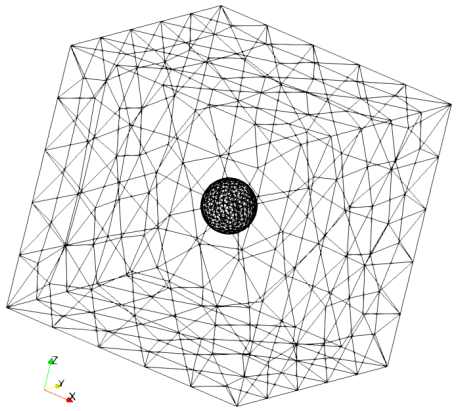

Figure 22: Illustration of the stokes problem. The Γ_B boundary is fixed. The Γ_S boundary is movable in order to optimize the shape.

In the following sections, we investigate the cases K_{11}, K_{12}, Q_{11}, Q_{12}, and C_{11} to compare our results with those presented in [12]. For all simulations except the Q_{12} case, we use the optimization parameters $t = 0.03$, $l = 20$, $a = b = 0.5$, and $c = 1e3$. In the Q_{12} case, we employ $t = 5e-4$, $l = 20$, $a = b = 0.5$, and $c = 1e5$.

K_{11} resistance problem :

Method	$J(\Omega_0)$	$J(\Omega_{n_{final}})$	$\lvert\lvert\Omega_0\rvert - \lvert\Omega_{n_{final}}\rvert\rvert$	$\lVert\theta_{n_{final}}\rVert_{H^1}$	n_{final}
GD	29.6246	27.2228	$2.9460e-4$	$5.0356e-4$	125
NSGF	29.6246	27.6108	$1.0395e-2$	$6.6623e-2$	33

Table 12: Comparison of the cost function between the initial domain and the final domain, the volume error between the two domains, the H^1-norm of the displacement field at the end and the number of iterations for the resistance problem K_{11} in 3D.

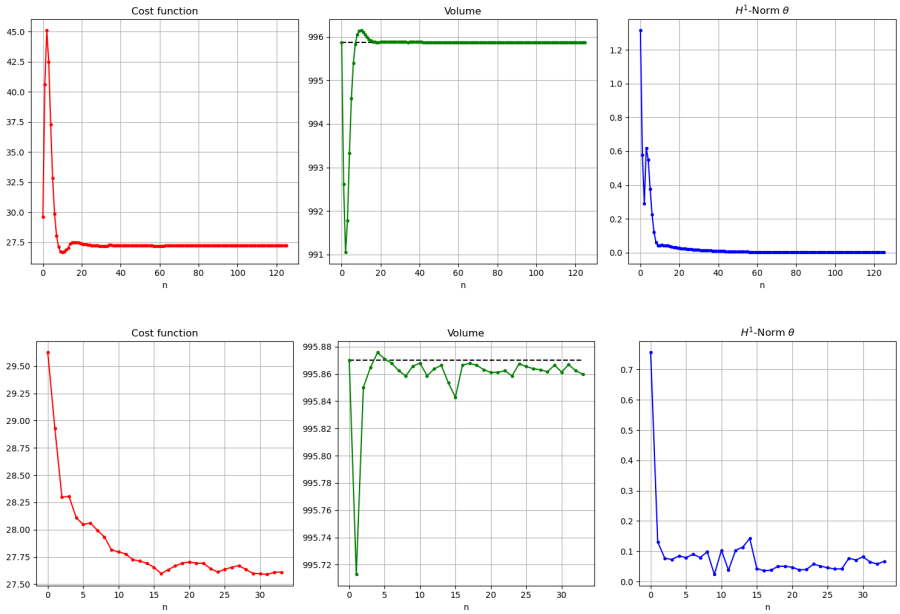

Figure 23: Evolution of the cost function, the volume of the domain Ω_n and the H^1-norm of θ_n for the 3D K_{11} resistance problem with the GD method (top) and the NSGF method (bottom).

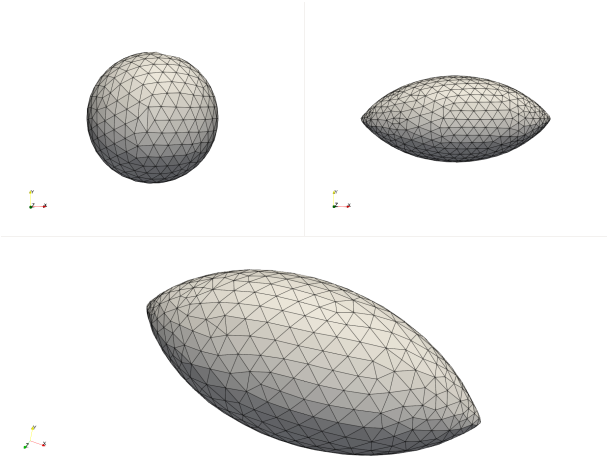

Figure 24: K_{11} setup : Initial domain (upper left) and final domain (upper right and bottom) for the GD method.

In the three-dimensional case, similar to the two-dimensional case, we successfully achieve the optimal rugby ball shape while preserving volume and reducing the cost function. However, we observe a behavior in the NSGF method that is comparable to the 2D case.

It appears to be highly sensitive to remeshing, particularly at the ends, resulting in a shape reminiscent of a lemon. Maintaining a very fine mesh specifically around the object, without altering it during remeshing, is not feasible. To obtain a satisfactory shape, an enormous number of points and consequently an very large number of elements would be required, especially when remeshing inside the fluid. As a result, the simulation needs to be terminated before the mesh collapses at the ends, as illustrated in Figure 25. However, it seems that the other configurations are less sensitive to remeshing in the NSGF method.

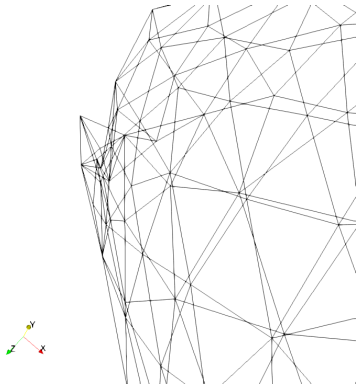

Figure 25: Example of mesh collapse at one end for K_{11} 3D with the NSGF method.

K_{12} resistance problem :

Method	$J(\Omega_0)$	$J(\Omega_{n_{final}})$	$\lvert\lvert\Omega_0\rvert - \lvert\Omega_{n_{final}}\rvert\rvert$	$\lVert\theta_{n_{final}}\rVert_{H^1}$	n_{final}
GD	-0.0188	-18.3755	0.2041	$9.1405e-2$	45
NSGF	-0.0188	-17.9411	$8.5382e-3$	$2.5186e-3$	850

Table 13: Comparison of the cost function between the initial domain and the final domain, the volume error between the two domains, the H^1-norm of the displacement field at the end and the number of iterations for the resistance problem K_{12} in 3D.

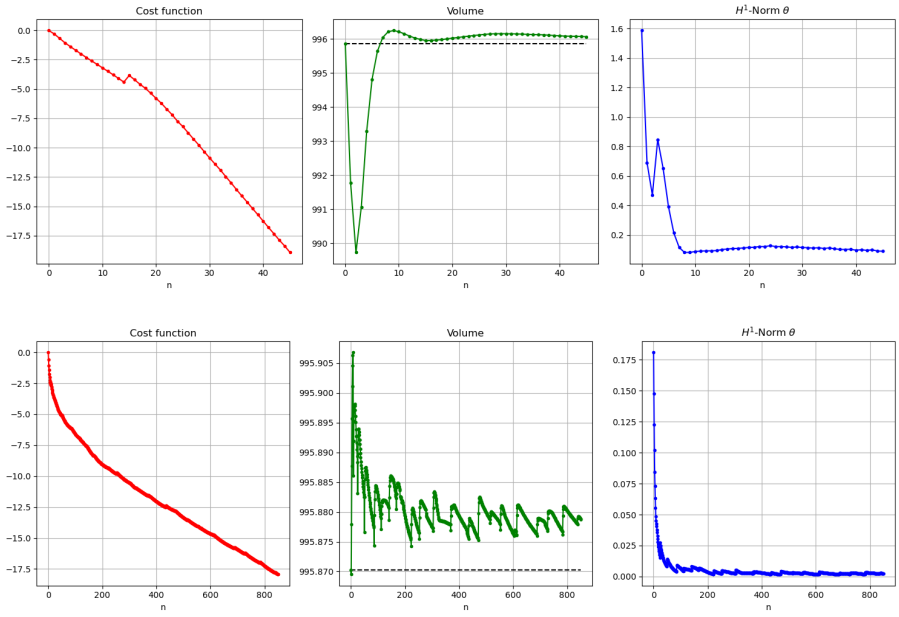

Figure 26: Evolution of the cost function, the volume of the domain Ω_n and the H^1-norm of θ_n for the 3D K_{12} resistance problem with the GD method (top) and the NSGF method (bottom).

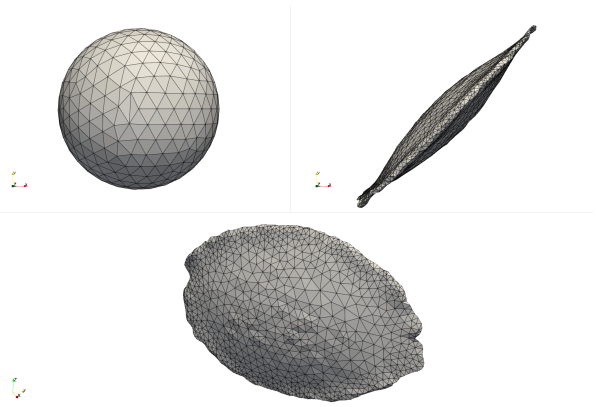

Figure 27: K_{12} setup : Initial domain (upper left) and final domain (upper right and bottom) for the GD method.

As depicted in Figure 26, the simulations are terminated before the cost function reaches a local minimum due to mesh collapse. Despite efforts to adjust parameters such as the descent step t, the issue of mesh collapse persists. There are minimal differences in the final shape obtained between the two methods. Nevertheless, we manage to achieve a shape that closely resembles the one presented in the reference article, although it should

be noted that the problem we are studying differs slightly as the boundary is at a finite distance from the rigid object.

Q_{11} resistance problem :

Method	$J(\Omega_0)$	$J(\Omega_{n_{final}})$	$\lvert\lvert\Omega_0\rvert - \lvert\Omega_{n_{final}}\rvert\rvert$	$\lVert\theta_{n_{final}}\rVert_{H^1}$	n_{final}
GD	24.1304	16.3814	$5.4293e-3$	$5.0132e-3$	500
NSGF	24.1304	16.9092	$5.7716e-3$	$1.7381e-3$	873

Table 14: Comparison of the cost function between the initial domain and the final domain, the volume error between the two domains, the H^1-norm of the displacement field at the end and the number of iterations for the resistance problem Q_{11} in 3D.

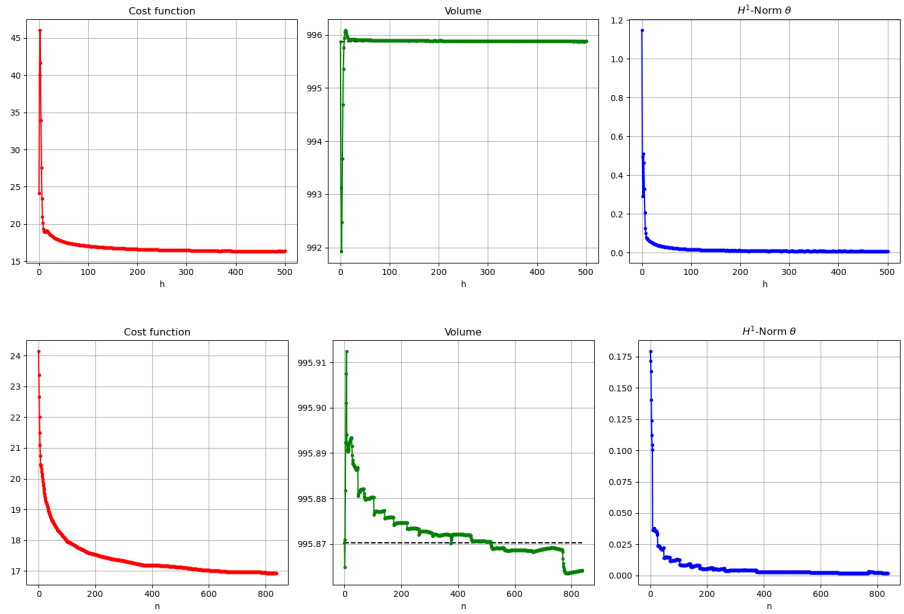

Figure 28: Evolution of the cost function, the volume of the domain Ω_n and the H^1-norm of θ_n for the 3D Q_{11} resistance problem with the GD method (top) and the NSGF method (bottom).

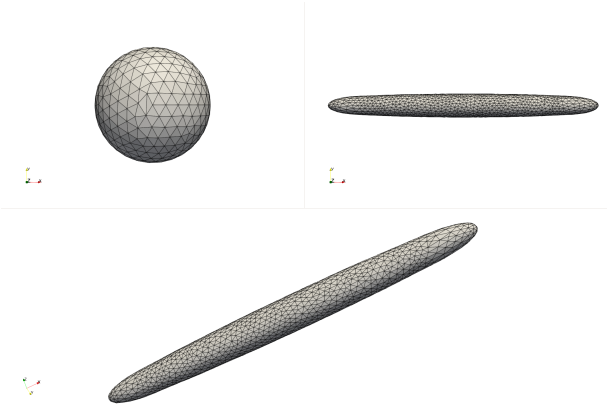

Figure 29: Q_{11} setup : Initial domain (upper left) and final domain (upper right and bottom) for the GD method.

In this case, premature termination of the simulation is less evident, and convergence is observed with good volume retention. The shapes obtained are virtually identical between the two resolution methods. However, a notable observation is that the ends of the object approach the edges of the cube until they collide with them. This phenomenon is not observed when the cube boundaries are assumed to be at infinity, as depicted in [12].

Q_{12} resistance problem :

Method	$J(\Omega_0)$	$J(\Omega_{n_{final}})$	$\lvert\lvert\Omega_0\rvert - \lvert\Omega_{n_{final}}\rvert\rvert$	$\lVert\theta_{n_{final}}\rVert_{H^1}$	n_{final}
GD	$-9.0952e-3$	-323.3069	$3.9189e-2$	$3.900e-2$	980
NSGF	$-9.0952e-3$	-234.4491	$1.5241e-2$	$6.3734e-3$	1000

Table 15: Comparison of the cost function between the initial domain and the final domain, the volume error between the two domains, the H^1-norm of the displacement field at the end and the number of iterations for the resistance problem Q_{12} in 3D.

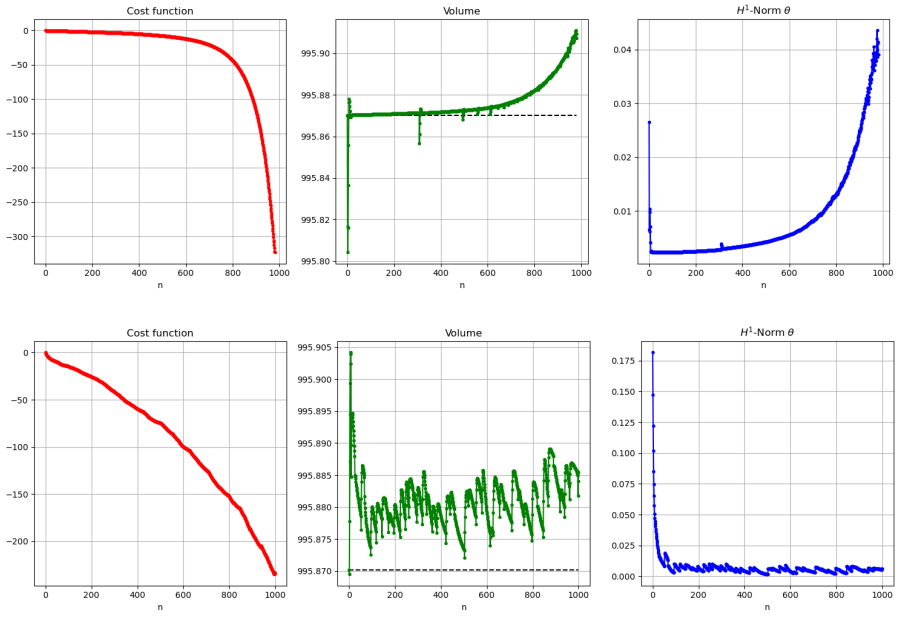

Figure 30: Evolution of the cost function, the volume of the domain Ω_n and the H^1-norm of θ_n for the 3D Q_{12} resistance problem with the GD method (top) and the NSGF method (bottom).

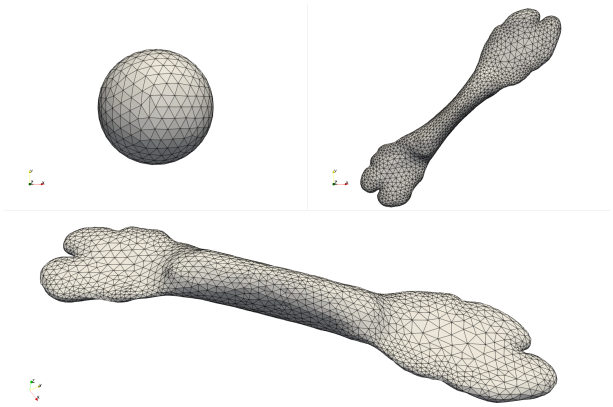

Figure 31: Q_{12} setup : Initial domain (upper left) and final domain (upper right and bottom) for the GD method.

The Q_{12} case proves to be the most challenging among the studied configurations. The boundary of the cube plays a significant role in the obtained results. Contrary to the expected dumbbell shape, we observe the ends of the solid flattening, leading to mesh collapse and an abrupt termination of the simulation. Additionally, these ends are positioned too closely to the cube boundary, resembling the observations made in the Q_{11}

case. Despite multiple attempts to optimize the parameters, we encounter a divergence $\|\theta_n\|_{H^1}$ during the iterations when using the Gradient Descent (GD) method. The shapes obtained using the two different methods exhibit a notable similarity. They possess the same overall profile, with slight variations in the prominence of the ends.

C_{11} resistance problem :

Method	$J(\Omega_0)$	$J(\Omega_{n_{final}})$	$\lvert\lvert\Omega_0\rvert - \lvert\Omega_{n_{final}}\rvert\rvert$	$\|\theta_{n_{final}}\|_{H^1}$	n_{final}
GD	$2.7004e-3$	-1.2593	$4.2977e-2$	$4.4303e-2$	74
NSGF	$2.7004e-3$	-1.7085	$9.7921e-4$	$1.2600e-3$	450

Table 16: Comparison of the cost function between the initial domain and the final domain, the volume error between the two domains, the H^1-norm of the displacement field at the end and the number of iterations for the resistance problem C_{11} in 3D.

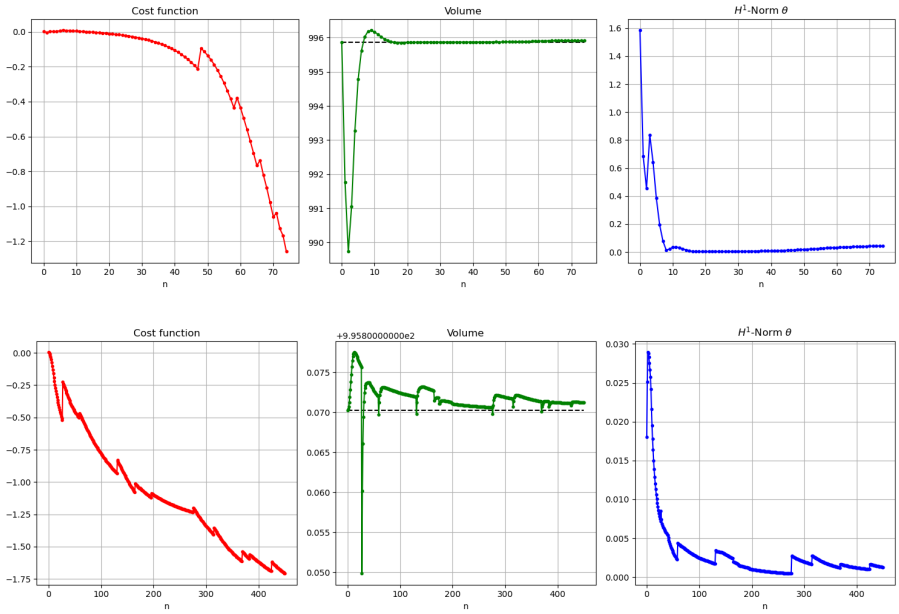

Figure 32: Evolution of the cost function, the volume of the domain Ω_n and the H^1-norm of θ_n for the 3D C_{11} resistance problem with the GD method (top) and the NSGF method (bottom).

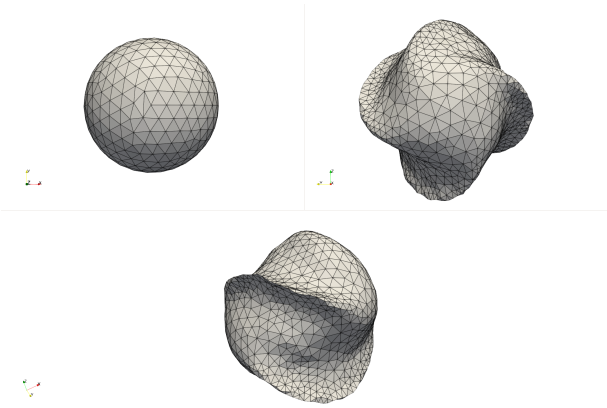

Figure 33: C_{11} setup : Initial domain (upper left) and final domain (upper right and bottom) for the GD method.

Mesh collapse continues to be evident in this case, mainly due to sharp edges forming in specific regions. However, the dynamics align with the description provided in [12], and volume conservation is satisfactory. Interestingly, we observe the emergence of four blades on the solid, each with varying prominence. It is worth noting that the difference between the two methods is apparent in Figure 34, where the blades are not necessarily in the same positions.

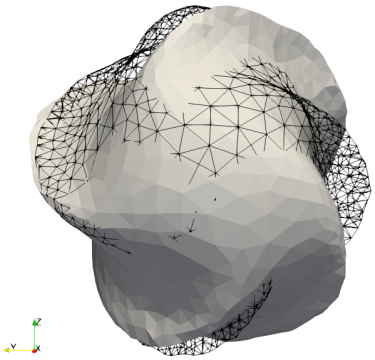

Figure 34: The black wireform corresponds to the result obtained using the GD method. The grey surface is the shape obtained using the NSGF method.

Despite the presence of walls, the shapes obtained in our study generally align closely with the expected results. However, some differences can be observed, which may be attributed to the specific assumptions made in our approach, such as the presence of walls, which differs from the use of a Boundary Element Method (BEM) in [12]. Additionally, the lack of regularity in the displacement field can lead to mesh collisions, contributing

to variations in the final shapes.

Comparing the two methods employed in our study, it is evident that the NSGF method tends to converge at a slower rate. This slower convergence is attributed to its dependency on the size of the smallest mesh element. However, one advantage of the NSGF method is the reduced occurrence of oscillations during iterations, particularly regarding volume conservation.

1.5 Discussion and perspectives

Geometric shape optimization is a powerful tool for exploring complex shape spaces. However, it is important to note that these types of problems often exhibit numerous local minima, making the search for global minima challenging. In this regard, the gradient-based methods employed in our study allow us to find local minima efficiently.

By leveraging the descent gradient, users have the ability to navigate the landscape of local minima and optimize the shape towards desirable objectives. This flexibility in convergence speed and exploration of the local minima space is a valuable aspect of gradient-based methods. One advantage of using the null space gradient flows method is the improved volume conservation compared to the gradient descent method, despite the increased number of iterations required for convergence. Maintaining volume during optimization is crucial for ensuring the physical accuracy of the resulting shapes.

Nevertheless, challenges related to mesh collapse and sensitivity persist. Certain problem configurations can lead to collisions between mesh elements, necessitating careful consideration and potential modifications to the problematic formulation. Approaches such as introducing constraints or regularization terms to promote mesh quality, as explored in [5], can help mitigate these challenges and enhance the overall robustness of the optimization process.

To ensure reproducibility and facilitate further investigation, we have made all the examples and results available in a public GitHub repository [13] and the documentation can be found here.

2 Bayesian Optimisation for Parametric Shape Optimisation

This section focuses on parametric shape optimisation by Bayesian methods. In Subsection 2.1, we describe how Gaussian processes work and why they're useful [9]. A Bayesian optimization algorithm for problems constrained by high-dimensional inequalities (SCBO) [6] is presented in Subsection 2.2. This is followed by a description of the Free-Form-Deformation [10] which is used to parameterize the shape, in Subsection 2.3. In Subsection 2.4, a Stokes fluid problem, a rigid solid in a Stokes flow [12] is studied. Finally, a brief conclusion is proposed in Subsection 2.5.

2.1 Gaussian Process

Gaussian Process (GP) is a probabilistic modeling technique used to approximate functions when direct evaluation is computationally expensive. Instead of evaluating the expensive function f at every point, GPs allow us to model f based on limited data and make predictions about its behavior with uncertainty estimates. It finds valuable applications in various fields such as weather forecasting, simulations, and optimization tasks. GPs provide an efficient way to handle complex functions and is particularly useful when computational resources are limited.

2.1.1 Prior Distribution

In Gaussian Process, we start with a prior distribution over functions. This prior represents our belief about the underlying function's behavior before observing any data. Conventionally, we assume a zero mean function for simplicity, but the prior can also be defined with a non-zero mean function if we have some prior knowledge about the function. We can write the prior as:

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

where $\mathcal{GP}(0, K(x, x'))$ denotes a Gaussian Process with a mean of 0 and a covariance (kernel) function $K(x, x')$ that specifies how the function values at different points are related. In the following, we will denote $Y(x)$ as the random variable associated with the Gaussian process prior.

2.1.2 Observations

Next, we obtain a set of n observations, usually in the form of data points with associated function values as $\{(x_i, y_i)\}_{i=1}^{n}$. These observations will help us update our prior belief to get a more informed estimate of the underlying function. The observed function values are related to the true underlying function values by noise, which is usually assumed to be Gaussian with zero mean and a known variance σ_n^2. If we know the true function, there is no need to add noise. We have:

$$y = f(x) + \varepsilon$$

where ε is a noise vector such that $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$. In the following, we will denote respectively X_n and Y_n as the matrix and the random variable for $\{x_i\}_{i=1}^n$ and $\{y_i\}_{i=1}^n$. The concatenation of all observations is defined as $D_n = (X_n, Y_n)$.

2.1.3 Posterior Distribution

Given the prior and observations, we want to infer the underlying function that generated the data. The objective is to update the prior to a posterior distribution, which represents our belief about the function after observing the data. The posterior distribution is obtained by conditioning the prior on the observed data. We suppose that $Y(x)$ and Y_n have a joint multivariate normal (MVN) distribution with mean zero and covariance function $\Sigma(x, x')$. From [9], the posterior distribution is given by:

$$Y(x) \mid D_n \sim \mathcal{N}\left(\mu(x), \sigma^2(x)\right)$$

with

$$\mu(x) = \Sigma\left(x, X_n\right) \Sigma_n^{-1} Y_n \; \sigma^2(x) \quad \text{and} \quad \sigma^2(x) = \Sigma(x, x) - \Sigma\left(x, X_n\right) \Sigma_n^{-1} \Sigma\left(X_n, x\right),$$

where $\Sigma_n = \Sigma(X_n, X_n)$. If we know the true function, then there is no noise and:

$$\Sigma(x, x') = K(x, x').$$

Otherwise, if noise is present (a hyperparameter), we have to consider it, and the covariance function becomes:

$$\Sigma(x, x') = K(x, x') + \sigma_n^2 \mathrm{I}.$$

Additionally, we can consider other hyperparameters to fine-tune our model. The goal of hyperparameters is to control and fine-tune the behavior of the learning algorithm. Unlike model parameters, which are learned from the data, hyperparameters are set manually before the learning process begins. As presented in [9], we can introduce scale, lengthscale, and other parameters. One way to define some hyperparameters as scale or lengthscale is to maximize the likelihood.

2.2 Scalable Constrained Bayesian Optimization

Let's now focus on optimization. Let $c : \mathbb{R}^d \to \mathbb{R}^m$ be m inequality constraints, which can be written as $c(x) = (c^1(x), ..., c^m(x))$. We seek to solve the following problem:

$$\inf_{x \in C} f(x)$$

where $C = \left\{x \in \Omega \subset \mathbb{R}^d \mid c(x) \leq 0\right\}$. One way to achieve this is through the Scalable Constrained Bayesian Optimization (SCBO) algorithm, proposed in [6]. This method allows solving optimization problems for expensive functions under costly inequality constraints using GPs, significantly reducing the number of calls to different functions. Moreover, it handles high-dimensional problems with a large number of constraints. For

example, it was tested in a 124-dimensional problem with 68 constraints in [6]. The various evaluations conducted in the article demonstrate that it outperforms previous methods for constrained high-dimensional problems. The SCBO method aligns perfectly with our intended approach.

Suppose we have a set of n observations for the cost function and constraint functions, given as $\{(x_i, y_i, c_i)\}_{i=1}^n$. Then, we consider X_n, Y_n, and C_n as the matrix and the random variables associated with the observations, and $D_n = (X_n, Y_n, C_n)$. We denote $\hat{f}(x_i)$ and $\hat{c}(x_i)$ as realizations of $Y(x_i) \mid D_n$ and $C(x_i) \mid D_n$ respectively. In the following, we consider that we know the true functions for the cost function and constraint functions. However, the reasoning is the same with the addition of noise, in which case we need to consider the averages of the posterior distributions instead of the true functions.

2.2.1 Transformations of the Cost and Constraint Functions

We are interested in finding the feasible zones, i.e., when the constraints are satisfied. Depending on the problem studied, it may be necessary to use a transformation of the constrained functions. By enlarging the range around the vicinity of zero, the change of sign that is decisive for admissibility is emphasized. In the following, we will use the bilog transformation: $bilog(y) = \text{sign}(y) \ln(1 + |y|)$.

2.2.2 Trust Region

The SCBO algorithm uses the trust region approach, allowing the selection of samples locally around the best current point to solve high-dimensional problems where the acquisition functions spread the samples due to large uncertainty, struggling to focus on promising solutions. The trust region is represented by a hypercube defined by its center and length. Let's consider n points $\{x_i\}_{i=1}^n$ within Ω. We calculate $m + 1$ realizations from the posterior distributions, as follows:

$$\left\{ \begin{pmatrix} \hat{f}(x_i) \\ \hat{c}(x_i) \end{pmatrix} \right\}_{i=1}^{n}.$$

Then, we define the set of feasible constraints for these realizations as:

$$\hat{F} = \{x_i \mid \hat{c}(x_i) \leq 0\}.$$

To choose the center x^* of the hypercube, two cases are considered depending on the cardinality of $\hat{F}$:

$$x^* = \begin{cases} \text{argmin}_{x \in \hat{F}} \hat{f}(x) & \text{if } \hat{F} \neq \emptyset, \\ \text{argmin}_{x \in \{x_i\}_{i=1}^n} \sum_{j=1}^m \max\left\{\hat{c}^j(x), 0\right\} & \text{if } \hat{F} = \emptyset, \end{cases}$$

i.e., either $\hat{F}$ is non-empty, and we take the point minimizing the GP representing the cost function, or $\hat{F}$ is empty, and we take the point minimizing the maximum constraint

violation. During the various iterations of the algorithm, the center of the trust region will move to explore areas of great interest. This method is analogous to Thomson sampling for constrained optimization. Thompson Sampling is an algorithm for decision-making under uncertainty. It assigns probability distributions to choices based on prior beliefs, samples from them, selects the choice with the highest sample, updates the distribution based on observed outcomes, and repeats to maximize cumulative rewards. Additionally, the size of the trust region, i.e., the length of the hypercube, will be adapted. Let τ_s and τ_f be the success rate and failure rate respectively, and n_s and n_f the number of successes and failures. Success is achieved when a better point than the center of the hypercube is obtained. We say that $\tilde{x}$ is a better point than x when x does not satisfy the constraints, and $\tilde{x}$ has a smaller constraint violation, or when x satisfies the constraints, and $\tilde{x}$ also satisfies them while minimizing the objective more. The length of the hypercube is then adapted as follows:

$$\text{if } n_s = \tau_s \text{ then } \begin{cases} L = \min\{2L, L_{max}\}, \\ n_s = 0, \end{cases} \quad \text{and if } n_f = \tau_f \text{ then } \begin{cases} L = L/2, \\ n_s = 0, \end{cases}$$

with L_{max} and L_{min} denoting the maximum and minimum length of the hypercube. If $L < L_{min}$, then a new trust region is defined.

2.2.3 Algorithm

Now that certain points have been clarified, we can describe how SCBO works. To illustrate this, Figure 35 depicts the different steps of the algorithm.

1. We start by taking a number of initial points within the Ω domain and defining the trust region using maximum utility as explained above.
2. Until the number of candidates/budget falls to zero:
   (a) Gaussian models associated with the cost function and the constraint functions are constructed from the observations.
   (b) Randomly choose r points inside the hypercube.
   (c) We consider a number of q batches. For each batch, we calculate a realization of the models at the level of discretization obtained by the previous r points. The best of the r points is then selected for each batch.
   (d) For the q best points obtained in the previous step, the objective and constraints are evaluated.
   (e) These new points are added to the previous observations.
   (f) The new center (if there is one) of the trust region is calculated, and its length is modified if necessary.
3. The algorithm returns the best admissible point.

A disturbance probability, as introduced in [2], has been incorporated. This probability, denoted as $p_{perturb}$, is determined by the following formula:

$$p_{perturb} = \min\left\{1, \frac{20}{d}\right\},$$

where d signifies the problem's dimension and q denotes the batch size. The introduction of this perturbation probability serves the purpose of mitigating the edge effect that can arise when working in high dimensions. This effect can lead the algorithm to become trapped at the boundaries of the hypercube. By selectively modifying only certain coordinates of a given point in accordance with this probability, the algorithm can evade this phenomenon and continue to explore the solution space more effectively.

The code from [2] required a few adjustments to rectify specific errors. For instance, the determination of the hypercube center was revised to consider the minimum of the maximum violated constraints, as opposed to merely selecting the argmax of the obtained values.

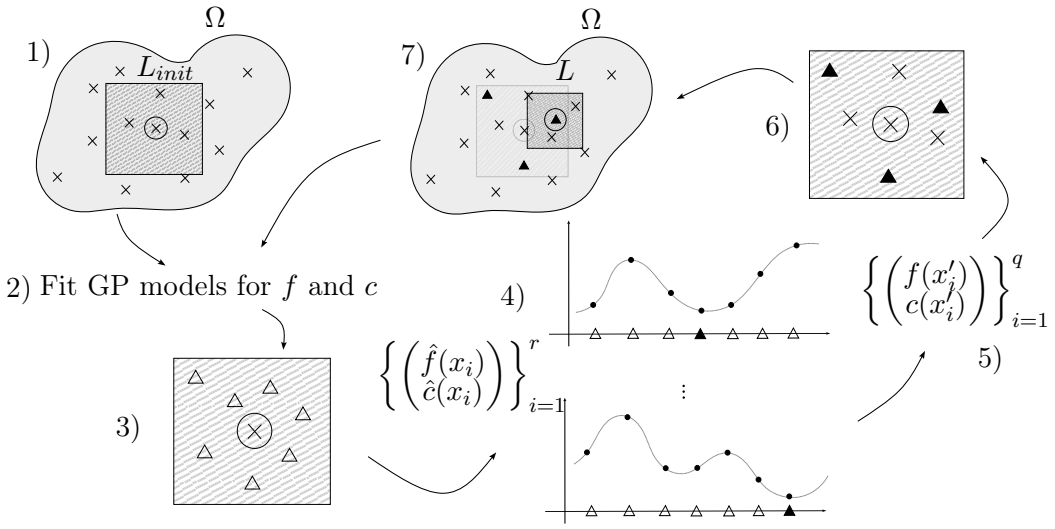

Figure 35: SCBO representation: 1) Initial points are taken from Ω (represented by crosses), and the trust zone is defined (a grid with the central point surrounded and the edge of the zone defined by a square). 2) The Gaussian models associated with the objective f and the constraints c are constructed. 3) r random points (represented by triangles) are taken from the trust zone. 4) A batch of q realizations $\{\begin{pmatrix}\hat{f}(x_i) & \hat{c}(x_i)\end{pmatrix}^T\}_{i=1}^{r}$ is calculated. For each realization, the x-axis corresponds to the x points in Ω and is discretized by the $\{x_i\}_{i=1}^{r}$ (the triangles) chosen previously. The y-axis shows the realizations. For simplicity, a single graph is drawn for the objective and the constraints. For each realization, we keep the best point (completely filled triangle). 5) For the q best points $\{x_i'\}_{i=1}^{q}$, we calculate the real value of the objective and the constraints. 6) These q new points are added to the previous observations. 7) The trust zone is readjusted by modifying its center and length.

2.3 Shape Parameterisation

To use the methods mentioned, we need to work in finite dimensions. Therefore, we must be able to parameterize the shape of our domain to transform the infinite-dimensional problem into a finite-dimensional one. This is where Free-Form Deformation (FFD) comes into play [10].

To illustrate how FFD works, let's consider the case in three dimensions. Suppose we have a sufficiently regular domain $D_0 \subset \mathbb{R}^3$ in which lies Ω_0, the shape we wish to deform. The FFD consists of deforming the entire D_0 domain using the control points and not just Ω_0. Depending on the chosen D_0 domain, we can opt for either global or local deformation of the shape, i.e., whether or not to completely enclose the shape. We define ψ as a diffeomorphism between the domain D_0 and the unit cube, i.e.:

$$\begin{array}{rcl} \psi : D_0 & \to & [0,1]^3 \\ (x,y,z) & \mapsto & (s,t,p). \end{array}$$

In this mapping, (s,t,p) are the coordinates of a point in the unit cube, and (x,y,z) are the coordinates of the corresponding point in the original domain D_0. The FFD approach allows us to smoothly and flexibly deform the shape of Ω_0 by manipulating the control points located in the unit cube. By adjusting the positions of these control points, we can deform the entire domain D_0 and study how the shape Ω_0 changes accordingly.

The FFD method provides a powerful tool to parameterize the shape of the domain and perform shape optimization in a finite-dimensional space, making it feasible to apply constrained optimization techniques such as SCBO.

On the D_0 domain, we define a regular grid using control points $P^0_{i,j,k}$ with $i \in [\![0, I]\!]$, $j \in [\![0, J]\!]$ and $k \in [\![0, K]\!]$. In the following, we will consider D_0 to be a hypercube and the control points to be equispaced, i.e.

$$P^0_{i,j,k} = \begin{pmatrix} i/I \\ j/J \\ k/K \end{pmatrix}.$$

Each control point can be moved in 3 directions. Thus, we define $\nu_{i,j,k} \in \mathbb{R}^3$ the vector applied to $P^0_{i,j,k}$. We then have

$$P_{i,j,k} = P^0_{i,j,k} + \nu_{i,j,k}.$$

Figure 36 illustrates a layout of control points with possible movements.

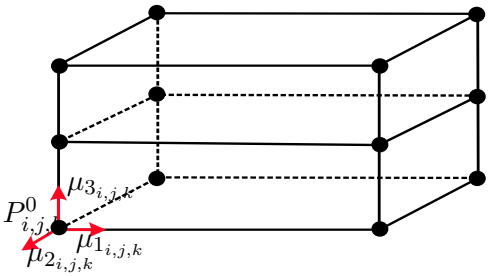

Figure 36: Uniformly distributed control points with $I = 2$, $J = 2$ and $K = 3$.

The FFD described here is based on Bernstein polynomials (2.1). Other alternatives can be considered, such as the use of splines, NURBS, etc. Bernstein polynomials are defined as

$$b_l^M(\gamma) = \binom{M}{l}(1-\gamma)^{M-l}\gamma^l \tag{2.1}$$

with $M \in \mathbb{N}^*$, $l \in [\![0, M]\!]$ and $\gamma \in [0, 1]$. Note that $b_l^M(0) = b_l^M(1) = 0$. We can therefore define the application T applying the deformation obtained by moving the control points. We have

$$\begin{array}{rcl} T : D_0 \times \mathbb{R}^{3\times(I+1)\times(J+1)\times(K+1)} & \to & D(\nu) \\ (x, y, z, \nu) & \mapsto & \psi^{-1}\left(\sum_{i=0}^{J}\sum_{j=0}^{J}\sum_{k=0}^{K} b_{i,j,k}^{I,J,K}\left(\psi(x, y, z)\right) P_{i,j,k}(\nu)\right), \end{array}$$

such that we note $P_{i,j,k}(\nu) = P_{i,j,k}$ so as not to forget the ν dependency and with

$$b_{i,j,k}^{I,J,K}(s, t, p) = b_i^J(s)b_j^J(t)b_k^K(p),$$

tensor products of Bernstein polynomials.

By the definition of FFD, deformations are exclusively confined within the domain D_0 due to the cancellation of Bernstein polynomials at the boundary. Notably, FFD is a global deformation technique, encompassing the entire specified domain D_0. In contrast, methods like NURBS are local, where the movement of a single control point affects only a limited region. The global nature of FFD implies that a shift in a single control point induces deformation across the entire D_0 domain, leading to continuous and regular transformations.

This regularity ensures a smooth and consistent deformation process, promoting stability in the optimization process. Considering D_0 and our Ω_0 shape discretized into a

set of nodes, we can effectively apply FFD to them. By using FFD as the parameterization technique, we achieve the flexibility to efficiently explore and optimize shapes within a finite-dimensional space while maintaining the desirable properties of regularity and smoothness in the deformations. These qualities contribute to stable and reliable optimization results.

2.4 A rigid body in a Stokes flow

Let's revisit the prior scenario, that of a rigid object immersed in a Stokes flow. In contrast to Subsection 1.4, we will now tackle the solving of PDEs using boundary element methods (BEM), an approach detailed below and referenced in [3]. Distinct from finite element techniques, BEM allows for the consideration of walls at infinite distances. We introduce G_{ij} for $(i,j) \in \{1,2,3\}^2$ as the Green's tensor kernel, facilitating the expression of fluid velocity in cases involving an unbounded domain.

In the given context, let S denote a rigid object that moves within an incompressible fluid of viscosity μ, operating under low Reynolds numbers. The fluid occupies an infinite domain Ω within $\mathbb{R}^N$. Additionally, we consider U to represent the velocities of the object. The fluid velocity field $u : \mathbb{R}^N \to \mathbb{R}^N$ and the pressure field $p : \mathbb{R}^N \to \mathbb{R}$ are solutions to the Stokes equations:

$$\begin{cases} -\mu\Delta u + \nabla p = 0 & \text{in } \Omega, \\ \nabla \cdot u = 0 & \text{in } \Omega, \\ u = U & \text{on } \Gamma_S, \\ \|u\|, p \to 0 & \text{as } \|x\| \to +\infty. \end{cases} \tag{2.2}$$

To address such a problem, we can employ the code provided by the library `Gypsilab` [11], which necessitates only the Γ_S boundary mesh. To parameterize Γ_S, representing the solid's surface, we will employ M control points utilizing the FFD method outlined in Subsection 2.2. The set of admissible shapes is defined as follows:

$$\Omega_{ad} = \left\{ P \in \mathbb{R}^{N\times M} \mid g(P) = 0 \right\} \quad \text{with} \quad g(P) = |S(P)| - |S_0|, \tag{2.3}$$

where $S(P)$ corresponds to the deformed solid resulting from the control point-induced deformation of S_0 using FFD. For S_0, we will consider the unit sphere. So if we take the control points at their initial position, i.e. if we take $\nu = 0$, we have no deformation and so we obtain the unit sphere.

Our objective is to solve the same minimization problem as (1.22), namely:

$$\inf_{\Omega\in\Omega_{ad}} \left\{ J_\alpha(\Omega) = \int_{\Gamma_S} \sigma(u,p)n \cdot \alpha \right\},$$

where n denotes the outward-pointing normal vector to Γ_S with respect to the fluid domain, and $\alpha \in \mathbb{R}^N$.

2.4.1 Boundary Element Method

The Boundary Element Method (BEM) is a numerical technique used for solving various types of partial differential equations. Unlike finite element methods that discretize the entire domain, BEM focuses on discretizing only the boundary of the problem domain. This makes BEM particularly well-suited for problems involving domains with complex geometries and/or infinite domains, where traditional finite element methods may become computationally expensive.

The fundamental concept behind the BEM is to represent the solution of a partial differential equation (PDE) in terms of its boundary values. By doing so, the problem is reduced from solving the PDE in the entire domain to solving integral equations on the boundary.

Let's start by expressing the fluid velocity through a boundary integral involving σn. Drawing from [3], for the situation defined in (2.2), the fundamental solution reads:

$$G_{ij}(x, x_0) = \frac{1}{8\pi\mu}\left(\frac{\delta_{ij}}{\|x - x_0\|} + \frac{(x_i - x_{0_i})(x_j - x_{0_j})}{\|x - x_0\|}\right), \tag{2.4}$$

where x represents the location of the fluid and x_0 corresponds to the site of an impulsive force. This enables the expression of velocity through an boundary integral:

$$u_j(x_0) = -\frac{1}{8\pi\mu}\int_{\Gamma_S} \sigma_{ik}(x) n_k(x) G_{ij}(x, x_0) dx, \tag{2.5}$$

valid for $x_0 \in \Gamma_S$. The fundamental pressure solution is relatively straightforward:

$$p(x, x_0) = \frac{(x - x_0)}{4\pi\|x - x_0\|^3},$$

providing the pressure value at x stemming from an impulsive force at x_0.

In what follows, we denote f as the surface tension σn. Suppose $(\phi^k)_{k=1}^n$ are piecewise-linear continuous functions spanning Γ_S, and we express $f_j(x)$ as:

$$f_j(x) = \sum_{k=1}^{n} f_j^k \phi^k(x).$$

By multiplying the basic functions and integrating over the boundary, we arrive at:

$$\int_{\Gamma_S} U_i \phi^l(x) dx + \int_{\Gamma_S}\left(\int_{\Gamma_S} G_{ij}(x, y) \sum_{k=1}^{n} f_j^k \phi^k(y) dy\right) \phi^l(x) dx = 0 \quad \text{for } l \in [\![1, n]\!]. \tag{2.6}$$

Through the exclusion of rotational motion, self-propulsion, and external forces, our task simplifies to solving a linear system:

$$Gf + J^T U = 0. \tag{2.7}$$

For an in-depth exploration of the G and J matrices, as well as their implementation, we refer the interested reader to [3]. Once σn, is determined, we can recover the velocity expression using equation (2.5). However, in our present scenario, we solely require σn for the computation of the cost function, rendering this step unnecessary. For alternative objective functions, such a step might indeed be crucial.

2.4.2 Experimental Evaluation

Let us proceed by revisiting the analysis of the various scenarios presented in the previous section for varying values of U and α. It is imperative to bear in mind that the outcomes may significantly diverge from those previously derived, given the inclusion of walls at infinite distance. In this context, the BEM code outlined in [11] for tackling the Stokes equation (2.2) enables examination solely in 3D. Accordingly, we will omit the 2D problem. Certain adaptations were essential to facilitate the examination of all feasible cases, not just restricted to a constant U vector.

Furthermore, the computation time for this algorithm, particularly concerning the function `stokesletEQ`, escalates rapidly based on the number of elements in the mesh. By averaging over approximately twenty trials, the ensuing computation times are obtained:

Mesh Size (h)	Time (seconds)
0.35	1.028
0.3	1.934
0.2	4.521
0.1	38.617

Table 17: Computation time of the `stokesletEQ` function in relation to the mesh size of the unit sphere for the K_{11} case.

For the subsequent simulations, a mesh size of $h = 0.35$ is generally employed unless otherwise specified. In fact, it is not mandatory to operate directly with a refined mesh because the optimal control points acquired using a "coarse" mesh can be extended to the much more detailed geometry to induce deformation and consequently acquire a more intricate shape. Nevertheless, for certain intricate shapes, this strategy might significantly influence the results. If the sought-after optimal shape exhibits substantial complexity, then utilizing a coarse mesh can potentially lead to outcomes entirely divergent from the expected ones.

Since the shapes are parameterized by control points, we adopt $J(P)$ instead of $J(\Omega)$, where the control points P correspond to the shape Ω. Thus, in the SCBO algorithm, we equate the notation x within Ω with P in a bounded $\mathbb{R}^{3\times M}$. Consequently, we will refer to $J(P)$ to denote the control points that parameterize the form. Furthermore, for the ensuing simulations, we will consider three control points in each direction, summing

up to 27 control points, yielding a problem of dimension 27×3. If we assume central symmetry in the shape and fix the central point, this is then reduced to a 39-dimensional problem.

The SCBO algorithm necessitates inequality constraints as opposed to equality constraints. Thus, we translate the volume constraint into two inequality constraints as follows:

$$\tilde{g}(P) = \begin{pmatrix} |S(P)| - |S_0| - \varepsilon \\ |S_0| - |S(P)| - \varepsilon \end{pmatrix},$$

where $\varepsilon = 1e-3$. This encapsulates the relationship $|||S(P)| - |S_0|| \leq \varepsilon$. The various parameters used for SCBO and taken from [6] are listed in Table 18

Parameter	Value
L_{init}	0.8
L_{min}	0.5^9
L_{max}	1.6
τ_s	$\max\{3, \lceil d/10 \rceil\}$
τ_f	$\lceil d/q \rceil$
Batch size	25

Table 18: The different parameters chosen for the SCBO algorithm.

In our investigation of each resistance problem, we will explore five distinct scenarios. These scenarios involve altering the problem's dimensionality by considering symmetrical or non-symmetrical shapes, incorporating a constraint-compliant solution (such as the unit sphere) into the initial data, and modifying the search boundaries for the control points.

K_{11} **resistance problem :** The outcomes of our investigations are summarized in Table 19. After determining the optimal control points using the SCBO algorithm on a coarse mesh, we extend the analysis by applying these control points to a much finer mesh. Remarkably, across all the examined problems, the derived control points consistently lead to objective values below those achievable with stationary control points (unit sphere), for both the coarse and fine meshes. This observation holds true even when directly incorporating the unit sphere into the initial data. Additionally, it is noteworthy that the algorithm successfully discovers shapes adhering to the imposed constraints, even when initialized with random data. The SCBO algorithm consistently yields improved shapes compared to the unit sphere and successfully respects the constraints even when starting with random initial data. We can also see that the results can be quite different from what was expected when applying the control points to the refined mesh.

n°	Sym.	Sphere	Bounds	$J(P)$	$J_{fine}(P)$	$\big\lvert\lvert S_0\rvert - \lvert S_{n_{final}}\rvert\big\rvert$	n_{final}
	X	X	X	18.5028	18.8277	0	0
1	Yes	No	$[-2,2]^{39}$	17.8103	18.0682	$1.6063e-05$	1958
2	Yes	Yes	$[-2,2]^{39}$	17.7504	17.9898	$8.3610e-04$	1243
3	No	Yes	$[-2,2]^{81}$	17.8480	18.2005	$8.7425e-04$	2875
4	Yes	No	$[-4,4]^{39}$	18.0185	18.3499	$1.4644e-04$	2657
5	Yes	Yes	$[-4,4]^{39}$	17.7546	17.9930	$8.3937e-04$	1393

Table 19: K_{11}: Comparison of the cost function for $h = 0.4$, the cost function for $h = 0.1$, the volume error and the number of iterations for the optimal points obtained as a function of different assumptions: central symmetry, addition of the unit sphere to the initial data, search limits for the control points. The first line corresponds to the unit sphere, i.e. when $P = 0$.

Pour le cas n°2 et le cas n°5, les points de contrôles et donc la forme obtenus sont quasiment identiques. Ainsi, une seule des deux formes sera présenté dans la Figure
K_{12} **resistance problem :**

n°	Sym.	Sphere	Bounds	$J(P)$	$J_{fine}(P)$	$\big\lvert\lvert S_0\rvert - \lvert S_{n_{final}}\rvert\big\rvert$	n_{final}
	X	X	X	$2.1864e-3$	$8.0577e-6$	0	0
1	Yes	No	$[-2,2]^{39}$	-4.3513	-3.8661	$2.3605e-4$	1893
2	Yes	Yes	$[-2,2]^{39}$	-6.9982	-7.0749	$7.5805e-04$	5012

Table 20: K_{12}: Comparison of the cost function for $h = 0.4$, the cost function for $h = 0.1$, the volume error and the number of iterations for the optimal points obtained as a function of different assumptions: central symmetry, addition of the unit sphere to the initial data, search limits for the control points. The first line corresponds to the unit sphere, i.e. when $P = 0$.

A cause du maillage grossier, il peut arriver que l'on tombe dans des minimums locaux et donc reste bloquer dedans. Une solution serait de rajouter un restart dans l'algo
Q_{11} **resistance problem :**

n°	Sym.	Sphere	Bounds	$J(P)$	$J_{fine}(P)$	$\lvert\lvert S_0\rvert - \lvert S_{n_{final}}\rvert\rvert$	n_{final}
	X	X	X	24.0779	25.0453	0	0
1	Yes	No	$[-2,2]^{39}$	18.2367	19.1702	$8.7867e-04$	3709
2	Yes	Yes	$[-2,2]^{39}$	17.7504	17.9898	$8.3610e-4$	1243

Table 21: Q_{11}: Comparison of the cost function for $h = 0.4$, the cost function for $h = 0.1$, the volume error and the number of iterations for the optimal points obtained as a function of different assumptions: central symmetry, addition of the unit sphere to the initial data, search limits for the control points. The first line corresponds to the unit sphere, i.e. when $P = 0$.

Q_{12} resistance problem :

n°	Sym.	Sphere	Bounds	$J(P)$	$J_{fine}(P)$	$\lvert\lvert S_0\rvert - \lvert S_{n_{final}}\rvert\rvert$	n_{final}
	X	X	X	$2.1864e-3$	$8.0577e-6$	0	0
1	Yes	No	$[-2,2]^{39}$	-4.3513	-3.8661	$2.3605e-4$	1893
2	Yes	Yes	$[-2,2]^{39}$	17.7504	17.9898	$8.3610e-4$	1243

Table 22: Q_{12}: Comparison of the cost function for $h = 0.4$, the cost function for $h = 0.1$, the volume error and the number of iterations for the optimal points obtained as a function of different assumptions: central symmetry, addition of the unit sphere to the initial data, search limits for the control points. The first line corresponds to the unit sphere, i.e. when $P = 0$.

C_{11} resistance problem :

n°	Sym.	Sphere	Bounds	$J(P)$	$J_{fine}(P)$	$\lvert\lvert S_0\rvert - \lvert S_{n_{final}}\rvert\rvert$	n_{final}
	X	X	X	$2.1864e-3$	$8.0577e-6$	0	0
1	Yes	No	$[-2,2]^{39}$	-4.3513	-3.8661	$2.3605e-4$	1893
2	Yes	Yes	$[-2,2]^{39}$	17.7504	17.9898	$8.3610e-4$	1243

Table 23: C_{11}: Comparison of the cost function for $h = 0.4$, the cost function for $h = 0.1$, the volume error and the number of iterations for the optimal points obtained as a function of different assumptions: central symmetry, addition of the unit sphere to the initial data, search limits for the control points. The first line corresponds to the unit sphere, i.e. when $P = 0$.

2.5 Discussion and perspectives

Parametric shape optimization provides an intuitive approach for studying complex shape spaces in a computationally manageable way. The transition from infinite to finite dimensions is a fundamental step that justifies the application of Bayesian optimization

techniques.

Gaussian processes, employed in Bayesian optimization, serve to significantly reduce the number of function evaluations required for the optimization process. This becomes particularly critical when the objective and constraint functions entail substantial computational costs.

The obtained results are quite promising, showcasing the algorithm's ability to effectively minimize the objective function and outperform the benchmark solution : the unit sphere. Nonetheless, the number of iterations necessary to reach a potentially optimal point can vary significantly depending on the specific problem. By pre-analyzing the problem and exploiting certain shape characteristics (e.g., symmetry), it's possible to improve the optimization process and achieve improved convergence rates. However, it's worth noting that some attained shapes may be less satisfactory compared to others, potentially deviating from those documented in previous studies [12].

Introducing restarts into the algorithm could enhance its exploration of the shape space. Although restarts are integrated in [6], this aspect is not currently incorporated in the `BoTorch` library [2], and it might be a beneficial modification to consider.

The selection of the number of control points holds significance as it shapes both the scope of the shape space and the complexity of the problem. A balance must be find between the space's size and the associated computational demands. Depending on the expected shape characteristics, alternative parameterization strategies, like using splines for less regular shapes, could be explored.

3 Conclusion

Throughout this internship, we explored geometric and parametric shape optimization, with a specific focus on a notable problem in fluid mechanics: a rigid object immersed in a Stokes flow.

In our exploration of geometric shape optimization, we employed the Céa method. Within this framework, we introduced two distinct resolution techniques: the gradient descent approach and the null space gradient flow method. Illustrating the efficacy of these methods, we applied them to two problems—the classical cantilever problem and a rigid object navigating through a Stokes flow. The practical implementation was realized using the `Feel++` library, and a documentation repository is readily accessible here.

Shifting our attention to the domain of parametric shape optimization, we studied Bayesian methods. Employing the Scalable Constrained Bayesian Optimization algorithm—an effective solution for high-dimensional problems—we linked it with shape parameterization using the Free-Form-Deformation technique. This combined approach yielded encouraging outcomes. The simulations are made by using the `Gypsilab` and `BoTorch` libraries.

After the fact, possible improvements to the optimisation of geometric shapes can come from obtaining greater regularity in the displacement field and refining the mesh. In addition, parallelization of the code is an important issue, given the often prolonged computation times associated with certain simulations. As far as the optimisation of parametric shapes is concerned, it would be interesting to take a closer look at the various case studies. Increasing the number of control points or adopting various shape parameterisation schemes, such as the use of splines, will provide a better understanding of the influence of parametric choices on the shapes obtained. Once again, parallel computing appears to be an essential strategy for mitigating the computational intensity inherent in simulations, especially given the exponential increase in computation time as mesh size increases.

Looking ahead, parametric shape optimization techniques, with Bayesian optimization strategies, could by use in the field of swimming robots with flagella. The challenge resides in optimizing both the robot's head and the positioning of its flagella, with a view to comparing the attained results with the findings of [3].

In conclusion, this internship journeyed through the intricacies of geometric and parametric shape optimization, unraveling versatile methodologies and their practical implementations across fluid mechanics.

References

[1] G. Allaire. *Conception optimale de structures*, volume 58 of *Mathématiques & applications*. Springer Berlin Heidelberg, 2006.

[2] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020.

[3] L. Berti. *Numerical methods and optimization for micro-swimming*. These de doctorat, Strasbourg, Dec. 2021.

[4] J. Cea. Conception optimale ou identification de formes, calcul rapide de la dérivée directionnelle de la fonction coût. *M2AN - Modélisation mathématique et analyse numérique*, 20(3):371–402, 1986.

[5] C. Dapogny, P. Frey, F. Omnès, and Y. Privat. Geometrical shape optimization in fluid mechanics using FreeFem++. *Structural and Multidisciplinary Optimization*, 58(6):2761–2788, Dec. 2018.

[6] D. Eriksson and M. Poloczek. Scalable Constrained Bayesian Optimization, Feb. 2021. arXiv:2002.08526 [cs, stat].

[7] F. Feppon. *Shape and topology optimization of multiphysics systems*. These de doctorat, Université Paris-Saclay (ComUE), Dec. 2019.

[8] D. Field. Qualitative measures for initial meshes. *International Journal for Numerical Methods in Engineering*, 47:887–906, Feb. 2000.

[9] R. B. Gramacy. *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Chapman Hall/CRC, Boca Raton, Florida, 2020. `http://bobby.gramacy.com/surrogates/`.

[10] A. Koshakji, A. Quarteroni, and G. Rozza. Free form deformation techniques applied to 3d shape optimization problems. 01 2013.

[11] matthieuaussal. matthieuaussal/gypsilab, Mar. 2023. original-date: 2017-07-10T12:22:09Z.

[12] C. Moreau, K. Ishimoto, and Y. Privat. *Shapes optimising grand resistance tensor entries for a rigid body in a Stokes flow*. July 2022.

[13] C. Prud'homme, V. Chabannes, T. Metivet, T. Saigre, L. Sala, C. Trophime, A. SAMAKE, L. Berti, and C. V. Landeghem. feelpp/feelpp: Feel++ v111 alpha.1, Mar. 2023.