UNIVERSITÀ DEGLI STUDI DI PADOVA

UNIVERSITY OF PADOVA

Civil, Environmental and Architectural department DICEA

Master's degree in Mathematical Engineering

Topology Optimization inside a fluid region

Andrea Gorgi, Badge 2010658
Gianmarco Boscolo, Badge 2029060

Academic year 2021/2022

Contents

1 Introduction

Abstract The target of this paper is to analyze and develop a layout topology optimization algorithm for laminar flows at steady state and a general objective function to minimize or maximize, giving also some insights to the used mathematical models for the numerical solution of the Navier-Stokes equations.

1.1 Topology optimization, an overview

Layout optimization has applied with growing interest in the most diverse production environments in the last decades, with the desire to find the best layout of a domain for a specific problem maximizing or minimizing a chosen functional. Usually, layout optimization can be categorized into three types, i.e. size optimization, shape optimization and topology optimization. Those three types can be easily distinguished as demonstrated in (1), where the compliance of the cantilever is minimized respectively using the size optimization, shape optimization and topology optimization approaches ([5]).
In size optimization, the structure is parameterized using sizes and positions of geometrical features, and the corresponding parameters are chosen to be the optimization variables; the shape optimization, instead, parameterizes the geometrical characteristics of structures using the spline interpolation etc.,where the sample points are the optimization variables. After parameterization, the optimization variables are iteratively evolved in the predefined feasible regions, using global optimization or gradient information-based optimization algorithm. Size optimization and shape optimization are subjected to the inflexibility on changing the topology of the initial guesses of the structure, and thus are characterized by a strong dependence on the initial guesses of the optimization variables.

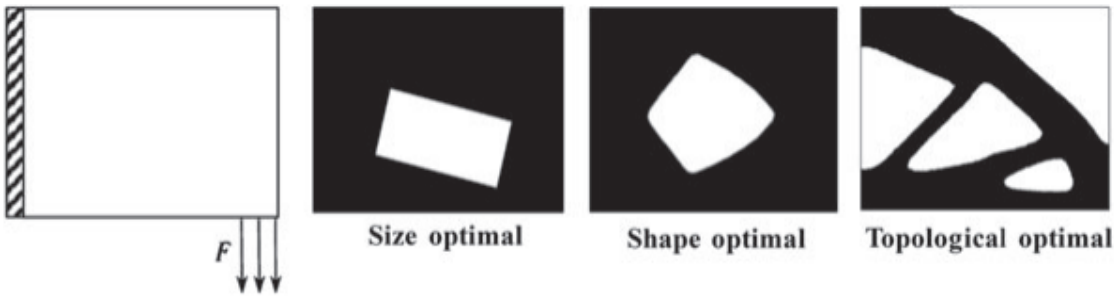


Fig.(1) Size optimization, shape optimization and topology optimization of a cantilever with minimizing compliance

The topology optimization on the other side is far less constrained by the guess on the initial configuration, and therefore should be a more powerful, flexible and robust tool for the structure optimization. It uses material distribution represented by the optimization variables. Since the material distribution is determined, topology optimization can fix the topology, shape and size of structures simultaneously.
Topology optimization arises from the structural optimization problem in elasticity and compliance mechanisms, but it has been extended to multiple physical problems, such as acoustics, electromagnetics, fluidics, optics and thermal dynamics.

Nowadays is widely used especially in engineering fields, but mostly at the concept level of a design process. In fact, due to the free forms that naturally occur, the result is often difficult to manufacture and needs to be fine-tuned for manufacturability. Adding constraints to the formulation in order to increase the manufacturability is an active field of research.

Several approaches, such as the evolutionary techniques, the homogenization method , the density method and the level set method, have been developed for the implementation of topology optimization (density and level set methods, [5]). For the aim of this project we only used the density method approach.

The idea of the density method is to introduce an artificial friction force, proportional to the fluid velocity, to obtain a smooth transition between the solid and the fluid phase. The analogous physical point of view is that the domain is intended as completely filled with a porous media, where porosity in each point is the optimization parameter. Clearly, on a practical point of view the original task was a discrete optimization problem, discriminating only between completely solid or completely void areas of the domain, without any intermediate configuration of porous media. The problem is instead treated with continuous variables to allow the usage of fast optimization techniques like the Method of Moving Asymptotes (MMA, [17]).Solid wall and open channels then correspond to the limits of very low and very high permeability, respectively.

Usually topology optimization problems are solved with the finite element method (FEM). Then, the design is optimized using either gradient-based mathematical programming techniques such as the optimality criteria ([11]) algorithm and MMA or non gradient-based algorithms such as genetic algorithms.

2 Problem definition

As written in the previous section, the main idea is to introduce an artificial friction force $\mathbf{f}$ to smoothly approximate the behaviour of the porous media. Since the method aims to work for laminar cases and low Reynolds numbers, the friction force is assumed to be proportional to the fluid velocity $\mathbf{u}$, according to Stokes' drag, $\mathbf{f} = -\alpha\mathbf{u}$. Even without computing the perfect value for α, it's reasonable that the drag force increases with viscosity and decreases with higher permeability, thus we'd expect something in the form $\alpha(\mathbf{r}) = \frac{\mu}{k}$, $k(\mathbf{r})$ local permeability of the medium at position $\mathbf{r} = (x, y)$ (in 2 dimensions). This approach is thus in principle correct only for $Re < 1$, but since this is an approximation and has just the scope of preventing the fluid from entering the domain parts with low permeability, it practically works even for higher Reynolds numbers.
The stationary flow in a certain computational domain Ω is obtained as solution of the steady state Navier-Stokes equations fo an incompressible fluid

$$\begin{cases} (\mathbf{u}\cdot\nabla)\mathbf{u} + \dfrac{1}{\rho}\nabla p - \nu\Delta\mathbf{u} = \dfrac{1}{\rho}\mathbf{f} & in\ \Omega \\ \nabla\cdot\mathbf{u} = 0 & in\ \Omega \\ \mathbf{u}(\mathbf{r}) = \mathbf{u}_D(\mathbf{r}), & on\ \Sigma_D \\ [\dfrac{p}{\rho}\mathbf{I} - \nu\nabla\mathbf{u}]\cdot\mathbf{n} = \mathbf{g}(\mathbf{r}), & on\ \Sigma_N, \end{cases} \tag{2.1}$$

where $\mathbf{f}$ is in general the force field, p is the pressure, ρ and ν are the density and the cinematic viscosity of the fluid and $\mathbf{n}$ is the external normal. The last two equations are respectively the Dirichlet and the Neumann boundary conditions, prescribing either the velocity at the boundary, or the external forces, $\mathbf{n}\cdot\sigma$, $\sigma = -p\mathbf{I} + \mu\nabla\mathbf{u}$ Cauchy stress tensor.
We now introduce a design variable $\gamma(\mathbf{r}) \in [0, 1]$ controlling the local permeability of the introduced medium, $\gamma = 0$ correspond to the solid material and $\gamma = 1$ to no material. Calling q a real and positive parameter for the $\alpha(\gamma)$'s shape tuning, and using a convex interpolation to relate $\alpha(\mathbf{r})$ and $\gamma(\mathbf{r})$, we have

$$\alpha(\gamma) \equiv \alpha_{min} + (\alpha_{max} - \alpha_{min})\frac{q(1-\gamma)}{q+\gamma}, \tag{2.2}$$

where α_{min} and α_{max} are the boundary values for α. A perfectly impermeable wall would correspond to a value $\alpha_{max} = +\infty$, but for numerical issues is preferable to have a big but finite value. Clearly the best

value for α_{min} is, indicating an area without solid material.
We recall the meaning of two important dimensionless numbers in fluid mechanics, the Reynolds and the Darcy numbers.
The Reynolds number represent the ratio between inertia and viscous forces,

$$Re = \frac{LU_\infty}{\nu}, \tag{2.3}$$

L and U_∞ characteristic values for length and velocity of the considered flow. When $Re >> 1$ the flow is mostly defined by the action of the convective term and viscous forces can be neglected, whereas if $Re << 1$ the opposite happens.
The Darcy number,

$$Da = \frac{k}{L^2} = \frac{\mu}{\alpha_{max} L^2}, \tag{2.4}$$

describes instead the ratio between viscous and porous friction forces. If $Da >> 1$ porous forces can be neglected, wheread for $Da << 1$ are the viscous forces that have negligible effects on the flow.
Almost impermeable solid materials are obtained by very low Darcy numbers, $Da \leq 10^{-5}$

2.1 Optimization problem

We can now state explicitly our optimal design task as continuous constrained non-linear optimization problem

$$\begin{cases} \min\limits_{\gamma} J(\mathbf{u}, p, \gamma) & \\ \text{subject to} & : \int_\Omega \gamma(\mathbf{r}) d\mathbf{r} - \beta|\Omega| \leq 0, \qquad \text{Volume constraint,} \\ & : 0 \leq \gamma(\mathbf{r}) \leq 1, \qquad \text{Design variable bounds,} \\ & : \text{Navier Stokes system(2.1)}, \qquad \text{Governing equations} \end{cases} \tag{2.5}$$

Using the standard notation for optimization problems, J will also be referred as the objective function, $\gamma, \mathbf{u}$ and p as the the decision variables and the equalities or inequalities that the system has to satisfy as the constraints. We recall that the original problem would have been in a discrete formulation, but a continuous one allow the use of fast programming algorithms like MMA (6). In this case J is the chosen functional to minimize (for maximization is sufficient to substitute the original J with $-J$), and some interesting choices for this functional will be discussed in following chapters. The volume constraint is an additional constraint that bind the fluid to occupy no more than a certain percentage of the total domain. In particular, in some cases, this extra bound prevents the algorithm from finding only trivial solution.
Most of the tools for solving continuous optimization problem are based on the gradient of the functional to minimize with respect to the optimization variables, and so is the chosen Method of Moving Asymptotes. From eq.(2.5) is easy to see that once a value for γ has been fixed, it's possible to solve the governing equations for $\mathbf{u}$ and p. Hence, $\mathbf{u}[\gamma]$ and $p[\gamma]$ defines implicit functions for γ.
If we try to compute the gradient of J we have

$$\frac{d}{d\gamma}[J(\mathbf{u}[\gamma], p[\gamma], \gamma)] = \frac{\partial J}{\partial \gamma} + \int_\Omega [\frac{\partial J}{\partial \mathbf{u}} \cdot \frac{\partial \mathbf{u}}{\partial \gamma} + \frac{\partial J}{\partial p} \cdot \frac{\partial p}{\partial \gamma}] d\mathbf{r}. \tag{2.6}$$

Since $\mathbf{u}[\gamma]$ and $p[\gamma]$ are implicit, it's not so simple to evaluate the derivative $\frac{\partial \mathbf{u}}{\partial \gamma}$ and $\frac{\partial p}{\partial \gamma}$ directly. The idea is then to make use of the adjoint method to eliminate them from the gradient formulation (2.6). We will therefore compute a set of adjoint multipliers for the governing equations (2.1). The optimization algorithm is then:

1. Choose an initial value for γ ,
2. Solve Governing equations (2.1) for $\mathbf{u}$ and p with a FEM algorithm (3.3),

3. Compute derivative of objective function and constraints with respect to gamma:

   (a) Solve the adjoint problem of (2.1) to eliminate partial derivative of implicit functions from the formulation

   (b) Compute the gradient of the objective function

4. Use MMA to update the value of γ to the value that minimizes J based on past iteration history and gradient informations,

5. Check convergence, if no, go back to step (2), if yes end the process.

We remark that the most expensive step of the process in terms of computational time is point 2 given that we would have to solve each time a system of non linear partial differential equations.

3 FEM analysis

The first interesting step is the computation of a solution of the Navier-Stokes equation (2.1) given a value of the design variable γ.
The method of choice in this paper for treating the system of partial differential equations is the FEM, "Finite Element Method". This is a special Galerkin method based on a variational formulation of the original problem in appropriate function spaces that is in practice discretized in finite dimensional subspaces ("trial spaces") consisting in polynomial functions. This approach allows the discretization to inherit most of the continuous structure of the starting problem that gives high computational flexibility and leads to a possible rigorous mathematical error analysis. We briefly recall the main features of two other widely used formulations (see [14]):

1. **Finite difference methods (FDM)**: Approximation of the Navier-Stokes equations in their "strong" form by finite differences:

   (a) + easy implementation

   (b) - problems along curved boundaries,

   (c) - difficult stability and convergence analysis,

   (d) - difficult mesh adaptation

2. **Finite volume methods(FVM)**: Approximation of the Navier-Stokes equations as a system of (cell-wise) conservation equations:

   (a) + based on "physical" conservation properties,

   (b) - problems on unstructured meshes,

   (c) - difficult stability and convergence analysis,

   (d) - only heuristic mesh adaptation.

This is a brief and superficial classification, and several aspects are still subject of controversial discussions of pros and cons of the various approaches. Anyway in many cases the differences between the methods are smoothed down, in particular between the FEM and FVM. In fact some of the FVMs can be interpreted as variants of certain "mixed" FEMs.

3.1 Stokes: Weak and Galerkin formulation

We now start by deriving the weak formulation of a simplified version of the Navier-Stokes equation, the Stokes equation:

$$\begin{cases} \nabla p - \mu \Delta \mathbf{u} = \mathbf{f} & in\ \Omega \\ \nabla \cdot \mathbf{u} = 0 & in\ \Omega \\ \mathbf{u}(\mathbf{r}) = \mathbf{u}_D(\mathbf{r}), & on\ \Sigma_D \\ [p\mathbf{I} - \mu \nabla \mathbf{u}] \cdot \mathbf{n} = \mathbf{g}(\mathbf{r}), & on\ \Sigma_N. \end{cases} \tag{3.1}$$

The solution to this system of PDEs describes a flow with a negligible convective term with respect to the viscous forces, thus a slow viscous flow $Re << 1$. This simplification usually pops out for small-scale biological applications, like movement in capillaries. To derive the weak formulation of eq.(3.1) we look for solution $(\mathbf{u}, p) \in \mathcal{V} \times \mathcal{Q}$, where (chap. 5 [12]):

$$\begin{aligned} \mathcal{V} &= [H^1_\Gamma(\Omega)]^d \\ \mathcal{Q} &= L^2(\Omega), \end{aligned} \tag{3.2}$$

where d is the space dimension of the problem (2 in our case), $H^1_\Gamma(\Omega) = \{v \in H^1(\Omega) \mid v = 0\ in\ \Gamma \subset \partial\Omega\}$, H^1 Hilbert space and Γ portion of the boundary where we want to impose our Dirichlet boundary conditions. If the boundary condition are only of Dirichlet type for the velocity, the pressure appears only under a gradient sign in the equations and it would be defined only up to a constant. In this case is thus necessary an additional condition to prevent oscillations,

$$\int_\Omega p d\mathbf{r} = 0. \tag{3.3}$$

As first thing we compute the weak formulation of these equations using the standard variational approach, hence we multiply each term of the first equation by a test function $\mathbf{v} \in [H^1_\Gamma(\Omega)]^d$, and each term of second equation by a $q \in L^2(\Omega)$ and integrate over Ω.

$$\begin{cases} \int_\Omega [-\mu \Delta \mathbf{u} \cdot \mathbf{v} + \nabla p \cdot \mathbf{v}] d\mathbf{r} = \int_\Omega \mathbf{f} \cdot \mathbf{v} d\mathbf{r} \\ \int_\Omega q \nabla \cdot \mathbf{u} d\mathbf{r} = 0 \end{cases} \tag{3.4}$$

Then we apply Gauss divergence theorem, obtaining

$$\begin{cases} \int_\Omega \mu \nabla \mathbf{u} : \nabla \mathbf{v} d\mathbf{r} - \int_\Omega p \nabla \cdot \mathbf{v} d\mathbf{r} - \int_{\partial\Omega} \mu \nabla \mathbf{u} \cdot \mathbf{v} \cdot \mathbf{n} d\sigma + \int_{\partial\Omega} p \mathbf{v} \cdot \mathbf{n} d\sigma = \int_\Omega \mathbf{f} \cdot \mathbf{v} d\mathbf{r} \\ \int_\Omega q \nabla \cdot \mathbf{u} d\mathbf{r} = 0. \end{cases} \tag{3.5}$$

We need the test functions $\mathbf{v}$ to be zero in the Dirichlet boundaries, thus we choose $\mathbf{v} \in [H^1_{\Sigma_D}(\Omega)]^d$. Simplifying the notation we end up with

$$\begin{cases} a(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) + \int_{\Sigma_N} [p\mathbf{I} - \mu \nabla \mathbf{u}] \cdot \mathbf{v} \cdot \mathbf{n} d\sigma = F(\mathbf{v}) \\ b(\mathbf{u}, q) = 0. \end{cases} \tag{3.6}$$

The term integrated in the Neumann boundary Σ_N is automatically computed using the Neumann boundary conditions in (3.1), where we made use of the bilinear and linear forms (assuming μ constant in the domain)

$$\begin{aligned} a(\mathbf{v}, \mathbf{w}) &= \mu \int_\Omega \nabla \mathbf{v} : \nabla \mathbf{w} d\mathbf{r} \\ b(\mathbf{v}, q) &= -\int_\Omega q \nabla \cdot \mathbf{v} d\mathbf{r} \\ F(\mathbf{v}) &= \int_\Omega \mathbf{f} \cdot \mathbf{v} d\mathbf{r}. \end{aligned} \tag{3.7}$$

Since the Neumann b.c. are imposed automatically in the weak formulation of the problem, they are also referred as "natural" condition, whereas the Dirichlet ones, that have to be forced in the strong form, are called "essential".
We can now proceed and obtain the Galerkin formulation.

To this scope, let us define a regular triangulation $\mathcal{T}_h(\Omega)$ of the domain Ω, and a finite subsets $\mathcal{V}_h \subset \mathcal{V}$, $\mathcal{Q}_h \subset \mathcal{Q}$, dependent on the triangulation. calling $\{\mathbf{w}_i\}$ the basis of $\mathcal{V}_h$ and $\{J_i\}$ the basis of $\mathcal{Q}_h$, the solution $(\mathbf{u}, p)$ will be expressed as

$$\begin{cases} \mathbf{u}(\mathbf{r}) = \sum_{j=1}^{N} \mathbf{u}_j \mathbf{w}_j(\mathbf{r}), \\ p(\mathbf{r}) = \sum_{k=1}^{M} p_k J_k(\mathbf{r}). \end{cases} \tag{3.8}$$

The FEM discretization of the Stokes equation is then the finite dimensional counterpart of problem (3.1) excluding the boundary conditions:

$$\begin{aligned} a(\mathbf{u_h}, \mathbf{v}) + b(\mathbf{v}, p_h) = F(\mathbf{v}) \quad & \forall \mathbf{v} \in \mathcal{V}_h(\mathcal{T}_h) \subset [H^1_\Gamma(\Omega)]^d, \\ b(\mathbf{u_h}, 1) = 0 \quad & \forall q \in \mathcal{Q}_h(\mathcal{T}_h) \subset L^2(\Omega), \end{aligned} \tag{3.9}$$

from which, equalizing the equations for each $\mathbf{v}$ in the basis for $\mathcal{V}_h$ and each q in the basis for $\mathcal{Q}_h$, we can build the linear system:

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}. \tag{3.10}$$

The matrix elements are

$$A = \{a_{ij}\} \qquad a_{ij} = a(\mathbf{w}_i, \mathbf{w}_j) = \mu \int_\Omega \nabla \mathbf{w}_i : \nabla \mathbf{w}_j d\mathbf{r} \qquad i, j = 1, ..., N; \tag{3.11}$$

$$B = \{b_{ki}\} \qquad b_{ki} = b(\mathbf{w}_i, J_k) = -\int_\Omega J_k \nabla \cdot \mathbf{w}_i d\mathbf{r} \qquad k = 1, ..., M; i = 1, ..., N; \tag{3.12}$$

$$\mathbf{f} = \{f_i\} \qquad f_i = \int_\Omega \mathbf{f} \cdot \mathbf{w}_i d\mathbf{r} \qquad i = 1, ..., N. \tag{3.13}$$

It can be proved ([13]-[12]) that the above problem is well posed (i.e. the solution exists, is unique and is a continuous function of the data), if the following properties are satisfied:

1. the bilinear form $a(\cdot, \cdot)$ is continuous and coercive:

   $|a(\mathbf{v}, \mathbf{w})| \leq \gamma ||v||_{\mathcal{V}_h} ||\mathbf{w}||_{\mathcal{V}_h}, \qquad a(\mathbf{v}, \mathbf{v}) \geq \alpha ||\mathbf{v}||_{\mathcal{V}_\mathbf{h}}{}^2$ for all $\mathbf{v}, \mathbf{w} \in \mathcal{V}_h$, for some $\gamma, \alpha > 0$;

2. the bilinear form $b(\cdot, \cdot)$ is continuous and satisfies the inf-sup condition, i.e.:

   $b(\mathbf{v}, q)| \leq \delta ||\mathbf{v}||_{\mathcal{V}_h} ||q||_{\mathcal{Q}_h}, \qquad b(\mathbf{v}, q) \geq \beta ||\mathbf{v}||_{\mathcal{V}_h} ||q||_{\mathcal{Q}_h}$ for all $(\mathbf{v}, q) \in \mathcal{V}_h \times \mathcal{Q}_h$, for some $\delta, \beta > 0$.

If such properties holds trues, the discretized model has some very useful bounds for the errors:

$$\begin{aligned} ||\mathbf{u} - \mathbf{u}_h||_{H^1(\Omega)} \leq C_1 ||\mathbf{u} - \mathbf{v}||_{H^1(\Omega)} + C_2 ||p - q||_{L^2(\Omega)} \quad & \forall (\mathbf{v}, q) \in \mathcal{V}_h \times \mathcal{Q}_h \\ ||p - p_h||_{H^1(\Omega)} \leq C_3 ||\mathbf{u} - \mathbf{v}||_{H^1(\Omega)} + C_4 ||p - q||_{L^2(\Omega)} \quad & \forall (\mathbf{v}, q) \in \mathcal{V}_h \times \mathcal{Q}_h. \end{aligned} \tag{3.14}$$

3.2 Elements choice

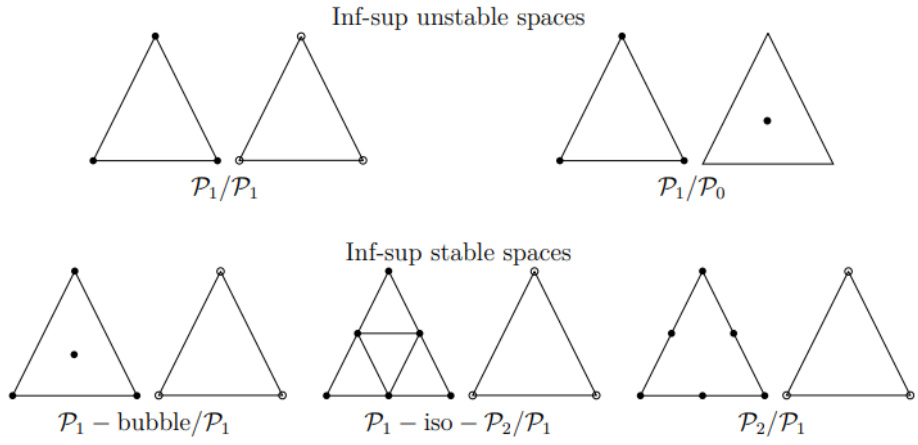
The above conditions for well-posedness of the problem are satisfied if the *discrete inf-sup condition* (also known as Babuska-Brezzi condition) holds: **For every** p **there must be a v such that**

$$p^T B\mathbf{v} \geq \beta ||\mathbf{v}||||p|| \tag{3.15}$$

for a fixed $\beta > 0$. This condition guarantees that the system matrix $\mathcal{M} = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}$ is invertible. Such condition would immediately fail if a nonzero presure p for which $Bp = 0$ exists. In the discrete case this is equivalent to impose that B has maximum rank (full column rank). In particular from (3.15) every nonzero pressure p must not be orthogonal to every $B^T\mathbf{v}$. Hence the space of all $A^T\mathbf{v}'s$ must have at least the dimension of the pressure space. In our case B represent the divergence operator, and this is the reason why often the velocity elements are defined by polynomial with one degree higher than the ones for the pressure elements. Unfortunately, dimension of spaces are not sufficient to prove the discrete inf-sup condition (3.15), and it's necessary to check each particular couple of spaces $(\mathcal{V}_h, \mathcal{Q}_h)$. The general idea is then that the two spaces cannot be chosen independently and that, loosely speaking, we need to allow enough degrees of freedom in $\mathcal{V}_h$ with respect to $\mathcal{Q}_\langle$ (see the references [12] for a more complete explanation).
Here some typical results, writing P_0, P_1, P_2 for constant, linear and quadratic polynomials on triangles, and Q_0, Q_1, Q_2 for constant, bilinear, and biquadratic elements on quadrilaterals:

1. Velocities in P_1, pressures in P_0: failure
2. Velocities in P_1, pressures in P_1: failure
3. Velocities in P_2, pressures in P_1: **well posed**
4. Velocities in Q_1, pressures in Q_1: failure
5. Velocities in Q_2, pressures in Q_1: **well posed**.

The simplest choice for a triangular discretization would then be the P_2/P_1 approach, but since working with linear functions is way simpler than working with quadratic ones, we exploit the possibility of enrich the velocity space but starting from a P_1/P_1 approach, without moving to the next order.
The chosen space is the P_1 - iso - P_2/P_1, which enriches the velocity space by uniformly refining the triangulation (connects the midpoints of each triangle). Details of the matrix entries definition and of other possible inf-sup stable spaces are given in ([12], chap.3).



3.3 Navier Stokes: Weak and Galerkin formulation

With the same procedure used for the Stokes problem, we can derive the weak formulation for the system (2.1). We are interested in the steady state of the flow, but from a numerical point of view the best approach is to solve the time dependent Navier - Stokes system up to convergence of the flow, to check the effective arrival toa stationary state. For this reason we build the Weak and Galerkin formulation for the complete Navier-Stokes equations,

$$\begin{cases} \partial_t \mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \frac{1}{\rho}\nabla p - \nu\Delta\mathbf{u} = \mathbf{f} & in\ \Omega \\ \nabla\cdot\mathbf{u} = 0 & in\ \Omega \\ \mathbf{u}(\mathbf{r}) = \mathbf{u}_D(\mathbf{r}), & on\ \Sigma_D \\ [\frac{p}{\rho}\mathbf{I} - \nu\nabla\mathbf{u}]\cdot\mathbf{n} = \mathbf{g}(\sigma), & on\ \Sigma_N. \end{cases} \tag{3.16}$$

Now, multiplying all members by appropriate test functions, integrating over the domain Ω and apply Green's theorem, we end up with

$$\int_\Omega (\partial_t\mathbf{u})\cdot\mathbf{v}d\mathbf{r} + \int_\Omega [\mathbf{u}\cdot\nabla\mathbf{u}]\cdot\mathbf{v}d\mathbf{r} + \int_\Omega \nu\nabla\mathbf{u}:\nabla\mathbf{v}d\mathbf{r} - \frac{1}{\rho}\int_\Omega \nabla\cdot\mathbf{v}pd\mathbf{r} = \int_\Omega \mathbf{f}\cdot\mathbf{v}d\mathbf{r} \qquad \forall\mathbf{v}\in\mathcal{V}, \tag{3.17}$$

$$\int_\Omega q\nabla\cdot\mathbf{u} = 0 \qquad \forall q\in\mathcal{Q}, \tag{3.18}$$

where $\mathcal{V}$ and $\mathcal{Q}$ are the same used above and we implicitly assumed only Dirichlet boundary conditions. Using a short notation

$$\begin{aligned} (\partial_t\mathbf{u},\mathbf{v}) + n(\mathbf{u},\mathbf{u},\mathbf{v}) + a(\mathbf{u},\mathbf{v}) + b(\mathbf{v},p) = F(\mathbf{v}) \quad \forall\mathbf{v}\in\mathcal{V}, \\ b(\mathbf{u},q) = 0 \quad \forall q\in\mathcal{Q}, \end{aligned} \tag{3.19}$$

with

$$(\mathbf{v},\mathbf{w}) = \int_\Omega \mathbf{v}\mathbf{w}d\mathbf{r} \qquad n(\mathbf{v},\mathbf{w},\mathbf{z}) = \int_\Omega (\mathbf{v}\cdot\nabla\mathbf{w})\cdot\mathbf{z}d\mathbf{r}$$

$$a(\mathbf{v},\mathbf{w}) = \mu\int_\Omega \nabla\mathbf{v}:\nabla\mathbf{w}d\mathbf{r} \qquad b(\mathbf{v},q) = -\int_\Omega q\nabla\cdot\mathbf{v}d\mathbf{r}$$

$$F(\mathbf{v}) = \int_\Omega \mathbf{f}\cdot\mathbf{v}d\mathbf{r}.$$

Note that the "tri-linear" form $n(\cdot,\cdot,\cdot)$ contains the nonlinear convective term, but here is considered for simplicity as a tri-linear form. All these linear, bi-linear and tri-linear forms are bounded and thus continuous, but in this case, even if it can be proved that a solution exists, uniqueness is completely proved only for two dimensional cases.
The Galerkin discretized formulation, using backward Euler time-stepping, is

$$(\mathbf{u}_h^{n+1},\mathbf{v}) + \Delta t_n[n(\mathbf{u}_h^{n+1},\mathbf{u}_h^{n+1},\mathbf{v}) + a(\mathbf{u}_h^{n+1},\mathbf{v}) + b(\mathbf{v},p_h^{n+1})] = (\mathbf{u}_h^n,\mathbf{v}) + \Delta t_n(\mathbf{f}(t_{n+1},\mathbf{v}) \qquad \forall v\in\mathcal{V}_h, \tag{3.20}$$

$$b(\mathbf{u}_h^{n+1},q) = 0 \qquad \forall q\in\mathcal{Q}_h. \tag{3.21}$$

As for the Stokes problem, we have

$$\begin{cases} \mathbf{u}_h(\mathbf{r},t) = \sum_{j=1}^{N}\mathbf{u}_j(t)\mathbf{w}_j(\mathbf{r}), \\ p_h(\mathbf{r},t) = \sum_{k=1}^{M} p_k(t)J_k(\mathbf{r}). \end{cases}$$

In matrix form, we get

$$\begin{bmatrix} (\frac{M}{\Delta t} + N(\mathbf{u}_h^{n+1}) + K), & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t} M \mathbf{u}^n + \mathbf{f} \\ 0 \end{bmatrix}, \tag{3.22}$$

where

$$\begin{aligned}
M = \{m_{ij}\} \qquad & m_{ij} = (\mathbf{w}_i, \mathbf{w}_j) = \int_\Omega \mathbf{w}_i \cdot \mathbf{w}_j d\mathbf{r} && i,j = 1,...,N; \\
N(\mathbf{u}_h) = \{n_{ij}(\mathbf{u}_h)\} \quad & n_{ij}(\mathbf{u}_h) = n(\mathbf{u}_h, \mathbf{w}_i, \mathbf{w}_j) = \int_\Omega [(\mathbf{u}_h \cdot \nabla)\mathbf{w}_i] \cdot \mathbf{w}_j d\mathbf{r} && i,j = 1,...,N; \\
K = \{k_{ij}\} \qquad & k_{ij} = a(\mathbf{w}_i, \mathbf{w}_j) = \nu \int_\Omega \nabla \mathbf{w}_i : \nabla \mathbf{w}_j d\mathbf{r} && i,j = 1,...,N; \\
B = \{b_{ki}\} \qquad & b_{ki} = b(\mathbf{w}_i, J_k) = -\frac{1}{\rho} \int_\Omega \phi_k \nabla \cdot \mathbf{w}_i d\mathbf{r} && k = 1,...,M; i = 1,...,N; \\
\mathbf{f} = \{f_i\} \qquad & f_i = \int_\Omega \mathbf{f} \cdot \mathbf{w}_i d\mathbf{r} && i = 1,...,N.
\end{aligned}$$

Since this formulation include the same problem of the Stokes equations for the coupling of the pressure and velocity discretization spaces, we will keep the previous choice of $P_1 - iso - P_1/P_2$ elements even for this formulation.
The main concern in this case is the treatment of the Convection term $n(\mathbf{u}_h^{n+1}, \mathbf{u}_h^{n+1}, \mathbf{v})$. This is a standard convection term with $\mathbf{u}_h^{n+1}$ as velocity field that carries the momentum in and out of the system. In order to solve the flow with a linear system, we need to linearize such term, and the idea is to fix the value of the velocity field to something like $\beta = \mathbf{u}_h^{n+1}$ in each step, hence

$$N(\beta) = \{n_{ij}(\beta)\} \qquad n_{ij}(\beta) = n(\beta, \mathbf{w}_i, \mathbf{w}_j) = \int_\Omega [(\beta \cdot \nabla)\mathbf{w}_i] \cdot \mathbf{w}_j d\mathbf{r} \qquad i,j = 1,...,N. \tag{3.23}$$

For large Reynolds number (large convection term), is necessary to introduce numerical diffusion in the system to stabilize the algorithm. Briefly, the problem is that the shape functions "weight" in the same way all the nodes of the system for the solution computation, but if there is a strong convective field the upstream nodes will clearly have a higher impact on the downstream ones than the opposite. The standard is thus the SUPG ("Streamline Upwind Petrov Galerkin), which is a residual based stabilization, which practically adds diffusion in the same direction of the convection term.
We then clearly have to stabilize the method in the choice for β. Two possible approaches are the following.

1. **Oseen apprach.** In this case at each time step we use the velocity field of the last computed time, $\beta = u_h^n$. In this case the alogithm is simple, but it can be proved that some restrictions on the time step Δt have to be applied for stability, and since our goal is to achieve steady state as fast as possible in the computations, we choose the following method,
2. **Picard iteration.** This apprach is based on a standard nonlinear solver (root finding), and it practically works through an inner loop for the β stabilization. At each time step we firstly fix $\beta = u_h^{n+1,0} = u_h^n$ as the first tentative term, and then we keep solving the system, without increasing the time, and update it's value until convergence. Convergence in this case could be approximated by a test on the difference between the β of two following iterations.

For a complete analysis of the just stated terms look at [12] or [13].

Forcing term

As explained above, the optimization algorithm works by introducing a forcing term of the type $f = -\alpha \mathbf{u}$, which in our formulation becomes $f = -\frac{\alpha}{\rho}\mathbf{u}$ due to the ρ division of the original equations. Differently

from what is showed in last chapters, since this term depends on the unknown $\mathbf{u}$, it cannot be left in the right hand side of the equations. We remark that if another external force field acts on the system and it doesn't depend on the unknowns it should be treated with the above procedure, but some calibration should be developed on the α_{min}, α_{max} and q coefficient to stabilize the interaction between the fictitious force $-\alpha\mathbf{u}$ (generally one increases the artificial force to prevent the entering of the fluid in the porous material even in the case of strong external fields that pushe in that direction [5]).
Thus in out case the real formulation is

$$\begin{aligned}(\partial_t\mathbf{u},\mathbf{v})+n(\mathbf{u},\mathbf{u},\mathbf{v})+a(\mathbf{u},\mathbf{v})+b(\mathbf{v},p)+\frac{1}{\rho}M_\alpha(\mathbf{u},\mathbf{v}) \quad &\forall\mathbf{v}\in\mathcal{V},\\ b(\mathbf{u},q)=0 \quad &\forall q\in\mathcal{Q},\end{aligned} \tag{3.24}$$

with

$$M_\alpha(\mathbf{u},v)=\int_\Omega \alpha(\mathbf{r})\mathbf{u}\mathbf{v}d\mathbf{r}. \tag{3.25}$$

In the discretized formulation

$$\begin{bmatrix}(\frac{M}{\Delta t}+\frac{1}{\rho}M_\alpha+N(\mathbf{u}_h^{n+1})+K), & B^T\\ B & 0\end{bmatrix}\begin{bmatrix}\mathbf{u}^{n+1}\\ p^{n+1}\end{bmatrix}=\begin{bmatrix}\frac{1}{\Delta t}M\mathbf{u}^n+\mathbf{f}\\ 0\end{bmatrix}, \tag{3.26}$$

$M_\alpha=\{m_{\alpha i,j}\}$, $m_{\alpha i,j}=\int_\Omega \alpha_i\mathbf{w}_i\cdot\mathbf{w}_j d\mathbf{r}$.

3.4 Boundary conditions

The aim of this project is to analyze the best shape of the solid material inside the domain to minimize or maximize the chosen functional, but clearly on different external conditions. For this reason we will spend some time discussing on the most important conditions that are usually applied to the system. It is well known that for a boundary value problem, there are three possible types of boundary conditions, Dirichlet, Neumann and Robin, which briefly speaking, prescribe at the boundary of the domain respectively, the value, the flux or a mixture of the two for the unknowns of the problem. In our case this corresponds to set the velocity value, its flux, or a combination of the two at the contour. The contour can be subdivided in parts with different boundary conditions, in values or type, with the only constraint to have just one condition for each node at the boundary, in order to avoid an overconstrained problem.

$$\begin{aligned}\mathbf{u}(\mathbf{r})&=\mathbf{g}_D(\mathbf{r}), && \mathbf{r}\in\Sigma_D, && \text{(Dirichlet BCs)}\\ [p\mathbf{I}-\mu\nabla\mathbf{u}(\mathbf{r})]\cdot\mathbf{n}&=\mathbf{q}_N(\mathbf{r}) && \mathbf{r}\in\Sigma_N && \text{(Neumann BCs)}\\ \beta\mathbf{u}+[p\mathbf{I}-\mu\nabla\mathbf{u}(\mathbf{r})]\cdot\mathbf{n}&=\mathbf{q}_C(\mathbf{r}) && \mathbf{r}\in\Sigma_C && \text{(Robin BCs)}\end{aligned}$$

For brevity, we will here report the features of just the Dirichlet and Neumann types, since the Robin's one is far less employed in simple problems like the ones presented here. An important point to stress is that since the pressure never appears in the governing equations outside from a derivative, it's necessary to impose at least one Neumann B.C., otherwise it would only be defined up to a constant and would generate oscillations in the solution. Another possibility to avoid oscillations (briefly discussed above), without this Neumann condition is to impose a further equation setting the mean pressure to a chosen value, like zero,

$$\int_{\partial\Omega} p d\mathbf{r}=0. \tag{3.27}$$

For the same reason, it's compulsory to impose at least one Dirichlet B.C. for the velocity at least in the Stokes problem.

Dirichlet B.C.

The Dirichlet boundary conditions, as told before, aims to fix the values of the unknowns at the chosen boundary $\Sigma_D \subset \partial\Omega$. In our discretized formulation, this corresponds to fix the nodal values of the velocity

$$\mathbf{u}_i = \mathbf{g}_i, \qquad \forall i \text{ Dirichlet}, \tag{3.28}$$

where $\mathbf{g}_i$ is the value that has to be imposed at node i of the Dirichlet boundary (using above notations, $\mathbf{g}_i = \mathbf{g}_D(\mathbf{r}_i)$, $\mathbf{r}_i$ position of node i). Other possibilities, like fixing just one of the velocity components are just trivial extensions of this one. The pressure value is instead related to the flux of the velocity, hence the case of fixed pressure in the boundary will be treated with the Neumann boundary conditions).
Such conditions is usually called "essential", since to enforce them, equation i of the Galerkin formulation, with i index of a Dirichlet node, has to be completely substituted with something like (3.28). More specifically such constraints have to be imposed in the strong formulation, changing the trial space of the solution candidates.
From a computational point of view one simply substitute the FEM equations at the Dirichlet nodes with the correct ones, usually with a "lifting function" or a "penalty method" approach ([12]), and symmetrizing the final system with some basic steps. From the mathematical side, instead, it is more correct to consider the Dirichlet conditions application as further equations applied to the system, with the introduction of $2 * N_D$ extra unknowns λ, N_D number of Dirichlet nodes. Recalling that the steady state formulation of Navier Stokes system for N nodes in the velocity space, and M in the pressure space, is

$$\begin{cases} n(\mathbf{u},\mathbf{u},\mathbf{w}_{1,i}) + a(\mathbf{u},\mathbf{w}_{1,i}) + b(\mathbf{w}_{1,i},p) + M_\alpha(\mathbf{u},\mathbf{w}_{1,i}) = 0, & i = 1,...,N \\ n(\mathbf{u},\mathbf{u},\mathbf{w}_{2,i}) + a(\mathbf{u},\mathbf{w}_{2,i}) + b(\mathbf{w}_{2,i},p) + M_\alpha(\mathbf{u},\mathbf{w}_{2,i}), & i = 1,...,N \\ b(\mathbf{u},J_k) = 0, & k = i-2N,\ i = 2N+1,...,2N+M, \end{cases} \tag{3.29}$$

will then become, defining $\mathcal{I}_D = \{i | \mathbf{r} \in \Sigma_D)\}$ set of all the indices of the nodes at the Dirichlet boundary,

$$\begin{cases} n(\mathbf{u},\mathbf{u},\mathbf{w}_{1,i}) + a(\mathbf{u},\mathbf{w}_{1,i}) + b(\mathbf{w}_{1,i},p) + M_\alpha\mathbf{u},\mathbf{w}_{1,i}) - \lambda_{1,i}\dfrac{\partial D_i}{\partial u_{1,i}} = 0, & i \in \{1,...,N\} \\ n(\mathbf{u},\mathbf{u},\mathbf{w}_{2,i}) + a(\mathbf{u},\mathbf{w}_{2,i}) + b(\mathbf{w}_{2,i},p) + M_\alpha(\mathbf{u},\mathbf{w}_{2,i}) - \lambda_{2,i}\dfrac{\partial D_i}{\partial u_{2,i}} = 0, & i \in \{1,...,N\} \\ b(\mathbf{u},J_k) = 0, & k = i-2N,\ i = 2N+1,...,2N+M, \\ D_{i,1}(u_{1,i}) = 0; \quad D_{i,2}(u_{2,i}) = 0 & i \in \{1,...,N\}. \end{cases} \tag{3.30}$$

The D_i are simply the pointwise enforcement of the Dirichlet conditions, basically $D_{1,i} = u_{1,i} - g_{1,i}$ of (3.28) if $i \in \mathcal{I}_D$ and $D_{1,i} = 0$ otherwise, and the same for the second component. Clearly, the last equations in (3.30) are not trivially satisfied only if $i \in \mathcal{I}_D$, and in the same way, $\frac{\partial D_i}{\partial u_{1,i}}$ and $\frac{\partial D_i}{\partial u_{2,i}}$ are not zero, and the λs have a role in the equations, only if i is a Dirichlet node, therefore only N_D equations and unknowns are really added to the system. The idea is that if the considered node is of Dirichlet, the corresponding equations of the original FEM is useless, since the correct one is the one defined by D_i, and its residual is not zero but equal to something that we have called λ. (3.30) is just the correction of the Galerkin formulation due to the application of the Dirichlet conditions.
With a shorter notation, $\frac{\partial D_i}{\partial u_{1,i}} = R_{1,i}$ and $\frac{\partial D_i}{\partial u_{2,i}} = R_{2,i}$.

Neumann B.C.

The Neumann bounday conditions are used to impose the velocity flux at the boundary. This condition is usually called "Natural" because it appears naturally in the FEM formulation. Practically we set the value of $[\frac{p}{\rho}\mathbf{I} - \nu\nabla\mathbf{u}] \cdot \mathbf{n} = \mathbf{q}_N$ in Σ_N. In the general case the steady state formulation of the Navier Stokes

formulation, with just Neumann BCs is

$$\begin{cases} n(\mathbf{u},\mathbf{u},\mathbf{w}_{1,i}) + a(\mathbf{u},\mathbf{w}_{1,i}) + b(\mathbf{w}_{1,i},p) + \displaystyle\int_{\Sigma_N} [\frac{p}{\rho}\mathbf{I} - \nu\nabla Bu]\cdot\mathbf{w}_{1,i}\cdot\mathbf{n}d\sigma + M_\alpha(\mathbf{u},\mathbf{w}_{1,i}), & i = 1,...,N \\ n(\mathbf{u},\mathbf{u},\mathbf{w}_{2,i}) + a(\mathbf{u},\mathbf{w}_{2,i}) + b(\mathbf{w}_{2,i},p) + \displaystyle\int_{\Sigma_N} [\frac{p}{\rho}\mathbf{I} - \nu\nabla Bu]\cdot\mathbf{w}_{2,i}\cdot\mathbf{n}d\sigma + M_\alpha(\mathbf{u},\mathbf{w}_{2,i}), & i = 1,...,N \\ b(\mathbf{u},J_k) = 0, & k = i-2N,\ i = 2N+1,...,2N \end{cases} \tag{3.31}$$

that is

$$\begin{cases} n(\mathbf{u},\mathbf{u},\mathbf{w}_{1,i}) + a(\mathbf{u},\mathbf{w}_{1,i}) + b(\mathbf{w}_{1,i},p) + \displaystyle\int_{\Sigma_N} \mathbf{q}_N(\mathbf{r})\cdot\mathbf{w}_{1,i}d\sigma + M_\alpha(\mathbf{u},\mathbf{w}_{1,i}), & i = 1,...,N \\ n(\mathbf{u},\mathbf{u},\mathbf{w}_{2,i}) + a(\mathbf{u},\mathbf{w}_{2,i}) + b(\mathbf{w}_{2,i},p) + \displaystyle\int_{\Sigma_N} \mathbf{q}_N(\mathbf{r})\cdot\mathbf{w}_{2,i}d\sigma + M_\alpha(\mathbf{u},\mathbf{w}_{2,i}), & i = 1,...,N \\ b(\mathbf{u},J_k) = 0, & k = i-2N,\ i = 2N+1,...,2N+M, \end{cases} \tag{3.32}$$

General external conditions

In a general flow problem there are at least three typical boundary conditions employed in the majority of the cases ([14])

1. $\mathbf{u}_{|\Gamma_{in}} = \mathbf{u}^{in}$, Inflow condition;
2. $\mathbf{u}_{|\Gamma_{rigid}} = 0$, Solid wall/"no slip" condition;
3. $([p\mathbf{I} + \mu\nabla\mathbf{u}]\cdot\mathbf{n})_{|\Gamma_{out}} = (p\mathbf{n} + \mu\partial_n\mathbf{u})_{|\Gamma_{out}} = 0$, Outflow condition.

Clearly, $\Gamma_{in}, \Gamma_{solid}$ and Γ_{out} are all non intersecting portion of $\partial\Omega$. Usually the inflow and solid wall condition is specified with a Dirichlet type BC prescribing the velocity of the ingoing flow or setting all the velocity components to zero. The treatment of the outflow boundary condition is instead a little more tricky. Numerical simulation of flow problems usually requires the truncation of a conceptionally unbounded flow region to a bounded computational domain, with conditions that aims to numerically replace the behaviour of the field left out, which are exactly the inflow and outflow condition.
The outflow condition stated above, $(p\mathbf{n}+\mu\partial_n\mathbf{u})_{|\Gamma_{out}} = 0$ has proven to be well suited for laminar parallel flows, and since it's naturally imposed without prescribing any boundary conditions, it's called the "do nothing" approach or "free outflow" condition. This approach usually yields very satisfactory results (3), but it's important to analyze what lies underneath it's formulation. With some trivial computations, calling $S \subset \partial\Omega$ the portion of the boundary where we want to apply such condition,

$$p\mathbf{n} + \mu\partial_n\mathbf{u} = 0 \implies \int_S p\mathbf{n}d\sigma = \mu\int_S \partial_n\mathbf{u}d\sigma = -\nu\int_S \partial_t\mathbf{u}d\sigma = \mathbf{u}(s_2) - \mathbf{u}(s_1) = 0, \tag{3.33}$$

where we have used the incompressibility of the flow

$$\nabla\cdot\mathbf{u} = 0 \implies \partial_n\mathbf{u} + \partial_t\mathbf{u} = 0, \tag{3.34}$$

and t is the direction perpendicular to the normal. $\mathbf{u}(s_1) = \mathbf{u}(s_2) = 0$ since we are supposing to apply the outflow boundary apart from the inflow condition, and thus near to the solid wall boundary where the velocity is identically zero due to the "no slip" condition. Therefore, the mean pressure over S must be zero:

$$\int_S pd\sigma = 0, \tag{3.35}$$

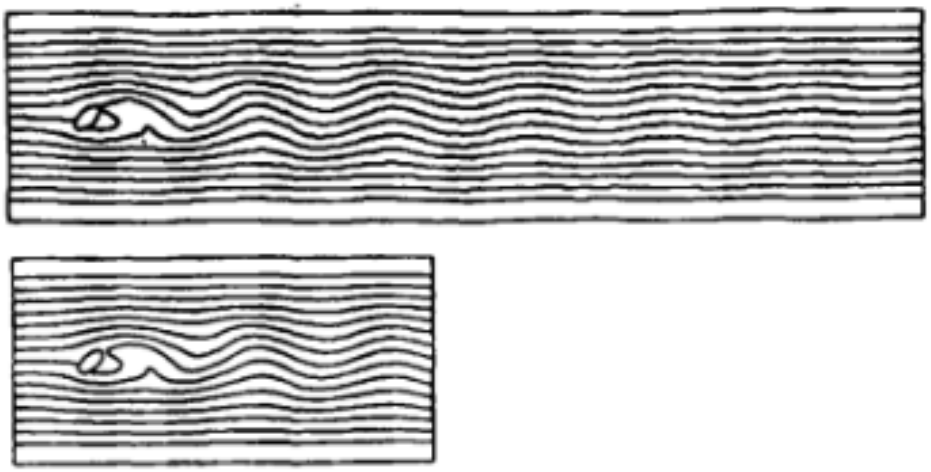


Fig.(2) The effect of the "do nothing" outflow condition shown by streamline plots for flow around an inclined ellipse at $Re = 500$ after 100 time steps. From [7]

hence, from the physical point of view, the "do nothing" outflow it's equivalent to apply a mean pressure of zero on such boundary.
From this some problem arise when multiple outflow region have to be imposed to the system, and an example is shown in figure (3). Here we see the see the results in terms of steady streamlines for a Poiseuille flow with same inflow condition and same variational formulation, but domain with one leg of the pipe longer than the other in the second formulation. It's clear that from the first simulation to the second, the flux across the upper boundary has increased, due to the fact that the path on the lower one has become longer, and since the inflow for the two channels is the same, and so is the mean pressure downstream, the pressure gradient it's evidently greater in the upper path.
An idea is to apply different mean pressures to each outflow boundary, given that

$$\begin{aligned} p\mathbf{n} + \mu\partial_n\mathbf{u} = \mathbf{q}_N \implies \int_S p\mathbf{n} + \mu\partial_n\mathbf{u}d\sigma = \int_S p\mathbf{n}d\sigma - \int_S \mu\partial_t\mathbf{u}d\sigma = \int_S p\mathbf{n}d\sigma \\ \bar{p} = \frac{1}{|S|}\int_S p\mathbf{n}d\sigma \implies \int_S p\mathbf{n} + \mu\partial_n\mathbf{u}d\sigma = |S|\bar{p}. \end{aligned} \tag{3.36}$$

More insights on this and some other outflow conditions approach are given in [14] or [7].

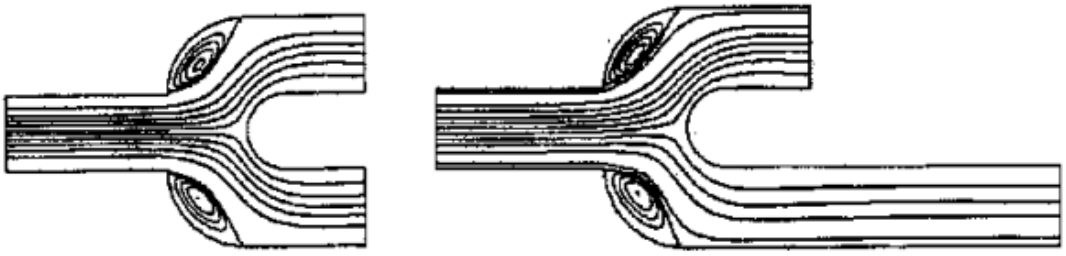


Fig.(3) The effect of the "do nothing" outflow condition shown by streamline plots for flow through a bifurcating channel at $Re = 20$. From [14]

4 Discretization and sensitivity analysis

Each finite element analysis starts from the idea of approximate the solution as a finite sum of shape functions, that in practice are weight functions of the nodal approximated values. Calling $\{J_{i,n}(\mathbf{r})\}$ the set of finite element basis functions on the velocity space ($i = 1$ for the x-direction, 2 for the y-direction), and $\{\psi_n(\mathbf{r})\}$ the ones for the pressure space,

$$\begin{cases} u_i(\mathbf{r}) = \sum_{j=1}^{N} u_{i,j}\mathbf{w}_{i,j}(\mathbf{r}), \\ p(\mathbf{r}) = \sum_{k=1}^{M} p_k J_k(\mathbf{r}). \end{cases} \tag{4.1}$$

Here we made explicit $\mathbf{w}_{1,j}(\mathbf{r}) = \begin{bmatrix} w_{1,j}(\mathbf{r}) \\ 0 \end{bmatrix}$ and $\mathbf{w}_{2,j}(\mathbf{r}) = \begin{bmatrix} 0 \\ w_{2,j}(\mathbf{r}) \end{bmatrix}$ $\forall j = 1, ..., N$. As said above, we've chosen a P1-iso-P2 element pairs, with linear approximation in all the elements, but with a grid refinement for the velocity consent the stabilization of the algorithm. There is no reason to use different basis function for the two components of the velocity, therefore $w_{1,j} = w_{2,j}$, $\forall j = 1, ..., N$.
The idea is that even the design variable $\gamma(\mathbf{r})$ can be approximated through a combination of shape functions, in the form

$$\gamma(\mathbf{r}) = \sum_{j=1}^{N} \gamma_j \psi_j(\mathbf{r}). \tag{4.2}$$

Actually, we have chosen to use a linear approximation in the velocity mesh, hence $\psi_j = w_{1,j} = w_{2,j}$ for every node j.

Consider the weak formulation associated to the Steady Navier Stokes equations (3.24 without the temporal term). Since we are interested in the optimization for the steady state, it makes sense compute the adjoint system for those equation, without the further difficulty of the temporal term. A more complete and rigorous derivation will be developed in chapter (7). Let's define $\{L_i(\mathbf{u}, p, \gamma) - \lambda_i(\gamma)R_i\}$, $i = 1, ..., 2N + M$ as the set of differential operators associated to the first $2N + M$ equations of the linear system for the steady state Navier-Stokes problem and $D_i(\mathbf{u})$ as the ones for the N_D equations of Dirichlet BCs. With respect to above chapter (3.4), we here define $R_{3,i} = 0$, $\forall i = 1, ...M$ to have a more uniform notation. Explicitely, the $2N + M + N_D$ equations, $N + N$ associated to the velocity space, M to the pressure, and the last N (of which only N_D are really effective) to the Dirichlet BCs are

$$\begin{cases} L_i - \lambda_i R_i = n(\mathbf{u}, \mathbf{u}, \mathbf{w}_{1,i}) + a(\mathbf{u}, \mathbf{w}_{1,i}) + b(\mathbf{w}_{1,i}, p) + M_\alpha(\mathbf{u}, \mathbf{w}_{1,i}) - \lambda_{1,i}\dfrac{\partial D_{1,i}}{\partial u_{1,i}}, \quad i = 1, ..., N \\ L_{N+i} - \lambda_{N+i}R_{N+i} = n(\mathbf{u}, \mathbf{u}, \mathbf{w}_{2,i}) + a(\mathbf{u}, \mathbf{w}_{2,i}) + b(\mathbf{w}_{2,i}, p) + M_\alpha(\mathbf{u}, \mathbf{w}_{2,i}) - \lambda_{2,i}\dfrac{\partial D_{2,i}}{\partial u_{2,i}}, \quad i = 1, ..., N \\ L_i = b(\mathbf{u}, J_k), \quad k = i - 2N;\ i = 2N + 1, ..., 2N + M, \\ D_{1,i} = 0; D_{2,i} = 0, \quad i = 1, ..., N. \end{cases} \tag{4.3}$$

For more clearance in the notation, it's better redefine $\{L_{1,i} - R_{1,i}\}$, $i = 1, ..., N$, the first N equations, associated to u_1, $\{L_{2,i} - R_{2,i}\}$, $i = 1, ..., N$, the second N equations, associated to u_2, and $\{L_{3,i} - R_{3,i}\}$, $i = 1, ..., M$ the M rows associated to p.
The Navier Stokes system is solved at each step of the optimization loop with with an appropriate value of γ. Hence, $\cong$ and p are each time built in order to satisfy $L_i(\mathbf{u}, p, \gamma) - \lambda_i(\gamma)R_i = 0$ and $D_i(\mathbf{u}) = 0$ for each value of γ, for all i's. Note that the R_is for the Dirichlet coupling in the FEM equations are independent from the design variable γ, since they are ones if the corresponding node is of Dirichlet type and zero otherwise, and this does not clearly change with γ, but the λ_i depend on it, since they are the residual of the practically useless FEM equations associated to the Dirichlet nodes, and the residual, as product of the variational equation that does depend on γ, still depends on it.
In this sense, we have the operators $L_i(\mathbf{u}(\gamma), p(\gamma), \gamma) - \lambda_i(\gamma)R_i$, and $D_i(\mathbf{u}(\gamma)) = 0$, and all of them are

equal to zero for all possible γ, that is they are constants in γ, and

$$\begin{aligned} \frac{d}{d\gamma}[L_i(\mathbf{u}(\gamma), p(\gamma), \gamma) - \lambda_i(\gamma) R_i] = 0, \quad \forall i = 1, ..., 2N + M, \\ \frac{d}{d\gamma} D_i(\mathbf{u}(\gamma)) = 0, \quad \forall i = 1, ..., 2N, \end{aligned} \tag{4.4}$$

that is equivalent to

$$\begin{aligned} \frac{d}{d\gamma}[L_{1,i}(\mathbf{u}(\gamma), p(\gamma), \gamma) - \lambda_{1,i}(\gamma) R_{1,i}] = 0, \quad \forall i = 1, ..., N; \\ \frac{d}{d\gamma}[L_{2,i}(\mathbf{u}(\gamma), p(\gamma), \gamma) - \lambda_{2,i}(\gamma) R_{2,i}] = 0, \quad \forall i = 1, ..., N; \\ \frac{d}{d\gamma} L_{3,i}(\mathbf{u}(\gamma), p(\gamma), \gamma) = 0, \quad \forall i = 1, ..., M \\ \frac{d}{d\gamma} D_{1,i}(\mathbf{u}(\gamma)) = 0, \quad \forall i = 1, ..., N, \\ \frac{d}{d\gamma} D_{2,i}(\mathbf{u}(\gamma)) = 0, \quad \forall i = 1, ..., N, \end{aligned} \tag{4.5}$$

We remark now that all this procedure has the only goal to simplify the computation of the gradient of the functional with respect to the design parameter, $\frac{dJ}{d\gamma}(\mathbf{u}, p, \gamma)$, that is $\frac{dJ}{d\gamma}(u_1, u_2, p, \gamma)$,and remove the partial derivation over γ for the implicit functions $u_1(\gamma)$ and $p(\gamma)$ in

$$\frac{dJ}{d\gamma} = \frac{\partial J}{\partial \gamma} + \frac{\partial J}{\partial u_1}\frac{\partial u_1}{\partial \gamma} + \frac{\partial J}{\partial u_2}\frac{\partial u_2}{\partial \gamma} + \frac{\partial J}{\partial p}\frac{\partial p}{\partial \gamma}. \tag{4.6}$$

We know however that in our discretized problem we compute the solution functions $\mathbf{u}$ and p only through known shape functions and unknown weights, which are the nodal values of $\mathbf{u}$ and p, that in particular are the outputs of the solution of the linear system. Therefore it makes sense to make explicit the dependence on such nodal values, instead of the unknown functions u_1, u_2 and p. With $J = J(\gamma, u_{1,1}, ..., u_{1,N}, u_{2,1}, ..., u_{2,N}, p_1, ..., p_M)$, (4.6) becomes

$$\frac{dJ}{d\gamma} = \frac{\partial J}{\partial \gamma} + \frac{\partial J}{\partial u_{1,1}}\frac{\partial u_{1,1}}{\partial \gamma} + ... + \frac{\partial J}{\partial u_{1,N}}\frac{\partial u_{1,N}}{\partial \gamma} + \frac{\partial J}{\partial u_{2,1}}\frac{\partial u_{2,1}}{\partial \gamma} + ... + \frac{\partial J}{\partial u_{2,N}}\frac{\partial u_{2,N}}{\partial \gamma} + \frac{\partial J}{\partial p_1}\frac{\partial p_1}{\partial \gamma} + ... + \frac{\partial J}{\partial p_M}\frac{\partial p_M}{\partial \gamma}. \tag{4.7}$$

The idea now is to add the $\{\frac{d(L_1 - \lambda_1 R_1)}{d\gamma}\}$, $\{\frac{d(L_2 - \lambda_2 R_2)}{d\gamma}\}$ and $\{\frac{dL_3}{d\gamma}\}$ to this equation, each multiplied by an appropriate factor $\{u_{a1,i}\}$, $\{u_{a2,i}\}$ and $\{p_{ai}\}$ and the $\{\frac{dD_i}{d\gamma}\}$ multiplied by $\lambda_{a,i}$. Note that all these terms are zero for definition, and thus we are not introducing any error in the original equation and that

$$\begin{aligned} \frac{d}{d\gamma} L_{l,i}(u_1, u_2, p, \gamma) &= \frac{\partial L_{l,i}}{\partial \gamma} + \frac{\partial L_{l,i}}{\partial u_{1,1}}\frac{\partial u_{1,1}}{\partial \gamma} + ... + \frac{\partial L_{l,i}}{\partial u_{1,N}}\frac{\partial u_{1,N}}{\partial \gamma} + \frac{\partial L_{l,i}}{\partial u_{2,1}}\frac{\partial u_{2,1}}{\partial \gamma} + ... + \frac{\partial L_{l,i}}{\partial u_{2,N}}\frac{\partial u_{2,N}}{\partial \gamma} + \\ &+ \frac{\partial L_{l,i}}{\partial p_1}\frac{\partial p_1}{\partial \gamma} + ... + \frac{\partial L_{l,i}}{\partial p_M}\frac{\partial p_M}{\partial \gamma}; \\ \frac{d}{d\gamma}(\lambda_{l,i}\gamma R_{l,i}) &= \frac{\partial}{\partial \gamma}(\lambda_{l,i}) R_{l,i}; \\ \frac{dD_{l,i}(u_1, u_2)}{d\gamma} &= \frac{\partial D_{l,i}}{\partial u_{1,1}}\frac{\partial u_{1,1}}{\partial \gamma} + ... + \frac{\partial D_{l,i}}{\partial u_{1,N}}\frac{\partial u_{1,N}}{\partial \gamma} + \frac{\partial D_{l,i}}{\partial u_{2,1}}\frac{\partial u_{2,1}}{\partial \gamma} \\ &= R_{l,i}\frac{\partial u_{l,i}}{\partial \gamma}. \end{aligned} \tag{4.8}$$

In this way we obtain

$$\frac{dJ}{d\gamma} = \frac{\partial J}{\partial \gamma} + \sum_{j=1}^{N}\left[\frac{\partial J}{\partial u_{1,j}}\frac{\partial u_{1,j}}{\partial \gamma} + \frac{\partial J}{\partial u_{2,j}}\frac{\partial u_{2,j}}{\partial \gamma}\right] + \sum_{j=1}^{M}\frac{\partial J}{\partial p_j}\frac{\partial p_j}{\partial \gamma} + \\ + \sum_{i=1}^{N}\left[u_{a1,i}\frac{d}{d\gamma}(L_{1,i} - \lambda_{i,1}R_{i,1}) + u_{a2,i}\frac{d}{d\gamma}(L_{2,i} - \lambda_{i,2}R_{i,2}) + \lambda_{a1,i}\frac{d}{d\gamma}D_{1,i} + \lambda_{a2,i}\frac{d}{d\gamma}D_{2,i}\right] + \sum_{i=1}^{M} p_{ai}\frac{d}{d\gamma}L_{3,i} \tag{4.9}$$

which becomes

$$\begin{aligned}\frac{dJ}{d\gamma} = &\frac{\partial}{\partial \gamma}\left[J + \sum_{i=1}^{N}(u_{a1,i}L_{1,i} + u_{a2,i}L_{2,i}) + \sum_{i=1}^{M} p_{ai}L_{3,i}\right] + \\ &+ \sum_{k=1}^{N}\frac{\partial u_{1,k}}{\partial \gamma}\left[\frac{\partial J}{\partial u_{1,k}} + \sum_{i=1}^{N}(u_{a1,i}\frac{\partial L_{1,i}}{\partial u_{1,k}} + u_{a2,i}\frac{\partial L_{2,i}}{\partial u_{1,k}}) + \sum_{i=1}^{M} p_{ai}\frac{\partial L_{3,i}}{\partial u_{1,k}} + \lambda_{1a,k}R_{1,k}\right] + \\ &+ \sum_{k=1}^{N}\frac{\partial u_{2,k}}{\partial \gamma}\left[\frac{\partial J}{\partial u_{2,k}} + \sum_{i=1}^{N}(u_{a1,i}\frac{\partial L_{1,i}}{\partial u_{2,k}} + u_{a2,i}\frac{\partial L_{2,i}}{\partial u_{2,k}}) + \sum_{i=1}^{M} p_{ai}\frac{\partial L_{3,i}}{\partial u_{2,k}} + \lambda_{2a,k}R_{2,k}\right] + \\ &+ \sum_{k=1}^{M}\frac{\partial p_k}{\partial \gamma}\left[\frac{\partial J}{\partial p_k} + \sum_{i=1}^{N}(u_{a1,i}\frac{\partial L_{1,i}}{\partial p_k} + u_{a2,i}\frac{\partial L_{2,i}}{\partial p_k}) + \sum_{i=1}^{M} p_{ai}\frac{\partial L_{3,i}}{\partial p_k}\right] + \\ &- \sum_{i=1}^{N}\frac{\partial \lambda_{1,i}}{\partial \gamma}R_{1,i}u_{a1,i} - \sum_{i=1}^{N}\frac{\partial \lambda_{2,i}}{\partial \gamma}R_{2,i}u_{a2,i}.\end{aligned} \tag{4.10}$$

From the above derivation we see that all the inconvenient derivatives with respect to $u_{i,k}$ or p_k over γ disappear from the equation if each one of the sums inside the square brackets is zero. Moreover, also the derivatives $\frac{\partial \lambda_{l,i}}{\partial \gamma}$ disappears from the equation if the corresponding $R_{l,i}u_{al,i} = 0$, $l = 1, 2$, and removing the trivial cases when $R_{l,i} = 0$ of the non Dirichlet nodes, we have others $2N_D$ operators to set equal to zero.
We note that there are in total $2N + M$ square brackets that we wanna equalize to zero plus the $2N_D$ of the Dirichlet conditions, and $2N + M + 2N_D$ free coefficients, $u_{a1,i}$, $u_{a2,i}$, pa_i and $\lambda_{a1,i}, \lambda_{a2,i}$ for the $i \in \mathcal{I}_D$. It's then defined a new system of equation, that we will call the "Adjoint" of the original,

$$\begin{cases}\dfrac{\partial J}{\partial u_{1,k}} + \displaystyle\sum_{i=1}^{N}(u_{a1,i}\frac{\partial L_{1,i}}{\partial u_{1,k}} + u_{a2,i}\frac{\partial L_{2,i}}{\partial u_{1,k}}) + \sum_{i=1}^{M} p_{ai}\frac{\partial L_{3,i}}{\partial u_{1,k}} + \lambda_{1a,k}R_{1,k} = 0, & k = 1, ..., N \\ \dfrac{\partial J}{\partial u_{2,k}} + \displaystyle\sum_{i=1}^{N}(u_{a1,i}\frac{\partial L_{1,i}}{\partial u_{2,k}} + u_{a2,i}\frac{\partial L_{2,i}}{\partial u_{2,k}}) + \sum_{i=1}^{M} p_{ai}\frac{\partial L_{3,i}}{\partial u_{2,k}} + \lambda_{2a,k}R_{2,k} = 0, & k = 1, ..., N \\ \dfrac{\partial J}{\partial p_k} + \displaystyle\sum_{i=1}^{N}(u_{a1,i}\frac{\partial L_{1,i}}{\partial p_k} + u_{a2,i}\frac{\partial L_{2,i}}{\partial p_k}) + \sum_{i=1}^{M} p_{ai}\frac{\partial L_{3,i}}{\partial p_k} = 0, & k = 1, ..., M \\ R_{1,i}u_{a1,i} = 0; \;\; R_{2,i}u_{a2,i} = 0, & i = 1, ..., N.\end{cases} \tag{4.11}$$

Recalling that practically the R_i are one or zero respectively if i
is of Dirichlet or not, it's easy to see that the last equations are substantially defining a zero Dirichlet BC for the adjoint on the same regions where Dirichlet conditions where defined for the governing equations. If we are able to compute such values for $u_{a1,i}$, $u_{a2,i}$, pa_i, the gradient of the functional reduces to

$$\frac{dJ}{d\gamma} = \frac{\partial}{\partial \gamma}\left[J + \sum_{i=1}^{N}(u_{a1,i}L_{1,i} + u_{a2,i}L_{2,i}) + \sum_{i=1}^{M} p_{ai}L_{3,i}\right], \tag{4.12}$$

which in our case is equal to

$$\frac{dJ}{d\gamma} = \frac{\partial J}{\partial \gamma} + \frac{\partial \alpha}{\gamma}\sum_{i=1}^{N}(u_{a1,i}\mathbf{u}\cdot\mathbf{w}_{1,i} + u_{a2,i}\mathbf{u}\cdot\mathbf{w}_{2,i}) = \frac{\partial J}{\partial \gamma} + \frac{\partial \alpha}{\partial \gamma}\mathbf{u}\cdot\mathbf{u}_a. \tag{4.13}$$

Adjoint system derivation

We will now show the main passages to obtain the adjoint system with this lax but direct approach.

$$\sum_{i=1}^{N} u_{a1,i} \frac{\partial L_{1,i}}{\partial u_{1,k}} = \sum_{i=1}^{N} u_{a1,i} \frac{\partial}{\partial u_{1,k}} [n(\mathbf{u}, \mathbf{u}, \mathbf{w}_{1,i}) + a(\mathbf{u}, \mathbf{w}_{1,i}) + b(\mathbf{w}_{1,i}, p) + M_\alpha(\mathbf{u}, \mathbf{w}_{1,i}) + G(\mathbf{w}_{i,1}). \quad (4.14)$$

Here $G(\mathbf{w}_{i,1})$ is the contribution for the Neumann BCs, but since it's assumed independent from all the unknowns it won't have any effect on the following passages, and thus we won't report it anymore. Let's first analyze the linear terms. Given that $\mathbf{u} = \sum_{i=1}^{N} [u_{1,i}\mathbf{w}_{1,i} + u_{2,i}\mathbf{w}_{2,i}]$, and taking the derivatives inside the integral defined by the a, b and M_α operators,

$$\begin{aligned}
\sum_{i=1}^{N} u_{a1,i} \frac{\partial}{\partial u_{1,k}} a(\mathbf{u}, \mathbf{w}_{1,i}) &= \sum_{i=1}^{N} u_{a1,i} a(\frac{\partial}{\partial u_{1,k}} \mathbf{u}, \mathbf{w}_{1,i}) = \sum_{i=1}^{N} u_{a1,i} a(\mathbf{w}_{1,k}, \mathbf{w}_{1,i}) = a(\mathbf{w}_{1,k}, \sum_{i=1}^{N} u_{a1,i}\mathbf{w}_{1,i}) \\
&= a(\sum_{i=1}^{N} u_{a1,i}\mathbf{w}_{1,i}, \mathbf{w}_{1,k}), \\
\sum_{i=1}^{N} u_{a1,i} \frac{\partial}{\partial u_{1,k}} b(\mathbf{w}_{1,i}, p) &= 0, \\
\sum_{i=1}^{N} u_{a1,i} \frac{\partial}{\partial u_{1,k}} M_\alpha(\mathbf{u}, \mathbf{w}_{1,i}) &= M_\alpha(\sum_{i=1}^{N} u_{a1,i}\mathbf{w}_{1,i}, \mathbf{w}_{1,k}),
\end{aligned}$$

where we have used the symmetry of the $a(\cdot)$ and $M_\alpha(\cdot)$ operators. The convective term, of the form

$$n(\mathbf{v}, \mathbf{w}, \mathbf{z}) = \int_\Omega (\mathbf{v} \cdot \nabla \mathbf{w}) \cdot \mathbf{z} d\mathbf{r}, \quad (4.15)$$

gives instead the following (exploiting the trilinearity of the term)

$$\begin{aligned}
\sum_{i=1}^{N} u_{a1,i} \frac{\partial}{\partial u_{1,k}} n(\mathbf{u}, \mathbf{u}, \mathbf{w}_{1,i}) &= \sum_{i=1}^{N} u_{a1,i} [n(\frac{\partial}{\partial u_{1,k}} \mathbf{u}, \mathbf{u}, \mathbf{w}_{1,i}) + n(\mathbf{u}, \frac{\partial}{\partial u_{1,k}} \mathbf{u}, \mathbf{w}_{1,i})] = \\
&= \sum_{i=1}^{N} u_{a1,i} [n(\mathbf{w}_{1,k}, \mathbf{u}, \mathbf{w}_{1,i}) + n(\mathbf{u}, \mathbf{w}_{1,k}, \mathbf{w}_{1,i})] \quad (4.16) \\
&= n(\mathbf{w}_{1,k}, \mathbf{u}, \sum_{i=1}^{N} u_{a1,i}\mathbf{w}_{1,i}) + n(\mathbf{u}, \mathbf{w}_{1,k}, \sum_{i=1}^{N} u_{a1,i}\mathbf{w}_{1,i}).
\end{aligned}$$

Of course, due to the symmetry of the equations, the same results holds for the $L_{2,i}$ equations with the corresponding indices "2" that replaces the "1"s. We compute now the terms for L_3

$$\begin{aligned}
\sum_{i=1}^{M} p_{a,i} \frac{\partial L_{3,i}}{\partial u_{1,k}} &= \sum_{i=1}^{M} p_{a,i} \frac{\partial}{\partial u_{1,k}} b(\mathbf{u}, J_i) = \sum_{i=1}^{M} p_{a,i} b(\frac{\partial}{\partial u_{1,k}} \mathbf{u}, J_i) \\
&= \sum_{i=1}^{M} p_{a,i} b(\mathbf{w}_{1,k}, J_i) = b(\mathbf{w}_{1,k}, \sum_{i=1}^{M} p_{a,i} J_i) = b(\mathbf{w}_{1,k}, p_a).
\end{aligned} \quad (4.17)$$

The first N equations of the adjoint system are then, from (4.11),

$$\begin{aligned}
&a(\sum_{i=1}^{N} u_{a1,i}\mathbf{w}_{1,i}, \mathbf{w}_{1,k}) + M_\alpha(\sum_{i=1}^{N} u_{a1,i}\mathbf{w}_{1,i}, \mathbf{w}_{1,k}) + n(\mathbf{w}_{1,k}, \mathbf{u}, \sum_{i=1}^{N} u_{a1,i}\mathbf{w}_{1,i}) + n(\mathbf{u}, \mathbf{w}_{1,k}, \sum_{i=1}^{N} u_{a1,i}\mathbf{w}_{1,i}) + \\
&+a(\sum_{i=1}^{N} u_{a2,i}\mathbf{w}_{2,i}, \mathbf{w}_{1,k}) + M_\alpha(\sum_{i=1}^{N} u_{a2,i}\mathbf{w}_{2,i}, \mathbf{w}_{1,k}) + n(\mathbf{w}_{1,k}, \mathbf{u}, \sum_{i=1}^{N} u_{a2,i}\mathbf{w}_{2,i}) + n(\mathbf{u}, \mathbf{w}_{1,k}, \sum_{i=1}^{N} u_{a2,i}\mathbf{w}_{2,i}) + \\
&+b(\mathbf{w}_{1,k}, p_a) + \lambda_{1a,k} R_{1,k} = -\frac{\partial J}{\partial u_{1,k}}.
\end{aligned} \quad (4.18)$$

Using linearity of the terms,

$$a(\mathbf{w}_{1,k}, \mathbf{u}_a) + M_\alpha(\mathbf{u}_a, \mathbf{w}_{1,k}) + n(\mathbf{w}_{1,k}, \mathbf{u}, \mathbf{u}_a) + n(\mathbf{u}, \mathbf{w}_{1,k}, \mathbf{u}_a) + b(\mathbf{w}_{1,k}, p_a) + \lambda_{1a,k} R_{1,k} = -\frac{\partial J}{\partial u_{1,k}}. \tag{4.19}$$

In the left hand side we have basically for the adjoint system the same formulation that we had for the master Navier Stokes problem, apart from the n terms that we will now analyze deeper.

$$n(\mathbf{w}_{1,k}, \mathbf{u}, \mathbf{u}_a) = \int_\Omega (\mathbf{w}_{1,k} \cdot \nabla \mathbf{u}) \cdot \mathbf{u}_a d\mathbf{r} = \int_\Omega (\mathbf{u}_a \cdot \nabla \mathbf{u}) \cdot \mathbf{w}_{1,k} d\mathbf{r} = n(\mathbf{u}_a, \mathbf{u}, \mathbf{w}_{1,k}), \tag{4.20}$$

that is the variational formulation for a term of type $u_a \cdot \nabla u$. The other term instead need to be treated with the integration by parts rule,

$$\begin{aligned} n(\mathbf{u}, \mathbf{w}_{1,k}, \mathbf{u}_a) &= \int_\Omega (\mathbf{u} \cdot \nabla \mathbf{w}_{1,k}) \cdot \mathbf{u}_a d\mathbf{r} = \int_\Omega (\mathbf{u} \otimes \mathbf{u}_a) : \nabla \mathbf{w}_{1,k} d\mathbf{r} \\ &= -\int_\Omega \nabla \cdot (\mathbf{u}_a \otimes \mathbf{u}) \cdot \mathbf{w}_{1,k} d\mathbf{r} + \int_{\partial\Omega} (\mathbf{u}_a \otimes \mathbf{u}) \cdot \mathbf{w}_{1,k} \cdot \mathbf{n} d\sigma \\ &= -\int_\Omega [\nabla \mathbf{u}_a \cdot \mathbf{u} + \mathbf{u}_a \nabla \cdot \mathbf{u}] \cdot \mathbf{w}_{1,k} d\mathbf{r} + \int_{\partial\Omega} (\mathbf{u}_a \cdot \mathbf{w}_{1,k}) \mathbf{u} \cdot \mathbf{n} d\sigma \\ &= -\int_\Omega (\nabla \mathbf{u}_a \cdot \mathbf{u}) \cdot \mathbf{w}_{1,k} d\mathbf{r} + \int_{\Sigma_N} [(\mathbf{u} \cdot \mathbf{n}) \mathbf{u}_a] \cdot \mathbf{w}_{1,k} d\sigma. \end{aligned} \tag{4.21}$$

In this way we can build the whole system for the adjoint problem, and the remarkable thing is that most of the matrices of the linear system will be the same of the ones used for the master problem. Moreover, in this case all the terms appearing in the formulation are linear, thus the resolution will surely require less computational time.
The term integrated over Σ_N corresponds to the Neumann BCs, since it can be thought as resulting from an integral of type $\int_{\Sigma_N} [\frac{p}{\rho}\mathbf{I} - \mu \nabla \mathbf{u}] \cdot \mathbf{w}_i \cdot \mathbf{n}$, after setting $[\frac{p}{\rho}\mathbf{I} - \mu \nabla \mathbf{u}] \cdot \mathbf{n} = (\mathbf{u} \cdot \mathbf{n}) \mathbf{u}_a$ in Σ_N. Of course if others integrals over the Neumann boundary will appear in the formulation due to the $\frac{\partial J}{\partial u_{1,k}}$ derivatives, the resulting Neumann BC will be the sum of the two terms (look at the complete derivation of the adjoint system below).
Clearly the adjoint equations associated to $u_{a2,k}$ will be exactly these replacing $\mathbf{w}_{1,k}$ with $\mathbf{w}_{2,k}$. Let's now compute the M equations for p_{ak}.

$$\begin{aligned} \sum_{i=1}^N u_{a1,i} \frac{\partial L_{1,i}}{\partial p_k} &= \sum_{i=1}^N u_{a1,i} \frac{\partial}{\partial p_k} [n(\mathbf{u}, \mathbf{u}, \mathbf{w}_{1,i}) + a(\mathbf{u}, \mathbf{w}_{1,i}) + b(\mathbf{w}_{1,i}, p) + M_\alpha(\mathbf{u}, \mathbf{w}_{1,i})] = \sum_{i=1}^N u_{a1,i} \frac{\partial}{\partial p_k} b(\mathbf{w}_{1,i}, p) \\ &= \sum_{i=1}^N u_{a1,i} b(\mathbf{w}_{1,i}, \frac{\partial}{\partial p_k} p) = \sum_{i=1}^N u_{a1,i} b(\mathbf{w}_{1,i}, J_k) = b(\sum_{i=1}^N u_{a1,i} \mathbf{w}_{1,i}, J_k) \\ \sum_{i=1}^N u_{a2,i} \frac{\partial L_{2,i}}{\partial p_k} &= b(\sum_{i=1}^N u_{a2,i} \mathbf{w}_{2,i}, J_k) \\ \sum_{i=1}^M p_{a,i} \frac{\partial L_{3,i}}{\partial p_k} &= 0 \end{aligned} \tag{4.22}$$

The resulting equations for the p_a are

$$b(\sum_{i=1}^N u_{a1,i} \mathbf{w}_{1,i}, J_k) + b(\sum_{i=1}^N u_{a2,i} \mathbf{w}_{2,i}, J_k) = -\frac{\partial J}{\partial p_k} \implies b(\mathbf{u}, J_k) = -\frac{\partial J}{\partial p_k}, \tag{4.23}$$

where the left hand side is the variational form of $-\nabla \cdot \mathbf{u}$ (the minus comes from the definition of the b operator).

Summarizing, the resulting adjoint system is exactly identical to the one that can be derived from the following continuous formulation (with the standard notation, without the Lagrange coefficients λ),

$$\begin{cases} -\nu\Delta u_a - (\mathbf{u}\cdot\nabla)\cdot\mathbf{u}_a + \nabla\mathbf{u}\cdot\mathbf{u}_a + \nabla p_a + \alpha\mathbf{u} = -\dfrac{\partial J}{\partial \mathbf{u}}, & in\ \Omega \\ \nabla\cdot\mathbf{u}_a = \dfrac{\partial J}{\partial p} & in\ \Omega \\ \mathbf{u}_a = 0, & on\ \Sigma_D \\ [\dfrac{p_a}{\rho}\mathbf{I} - \nu\nabla\mathbf{u}_a]\cdot\mathbf{n} = (\mathbf{u}\cdot\mathbf{n})\mathbf{u}_a, & on\ \Sigma_N. \end{cases} \tag{4.24}$$

Note that the boundary conditions could have some extra terms depending on the chosen functional.

5 The objective functional

The standard optimization problems are usually formulated as minimisation problems unless described differently, and we'll proceed in the same way. Hence, a reduction of the objective functional leads to an improved design. The objective functionals are classified as boundary- or volume-based

$$J(\mathbf{u}, p; \gamma) = \int_\Omega \beta_1 A(\mathbf{u}, \nabla\mathbf{u}, p; \gamma))d\mathbf{r} + \int_{\partial\Omega} \beta_2 B(\mathbf{u}, p; \gamma)d\sigma, \tag{5.1}$$

where β_1 and β_2 are space independent parameters that can be used to weight the importance of the two possible scopes of the functional (in the following we will always assume $\beta_1 = \beta_2 = 0$.

As stated above, the type of functional can slightly change the boundary conditions of the adjoint system (due to the partial derivatives in u_k or p_k), since, as discussed in previous chapter, all the terms integrated over the Neumann boundary can be summed and considered as Neumann BCs. Moreover, we said that the λs parameters depends on γ, but since this is true only through the values of $\mathbf{u}$ and p, it's actually $\lambda(\mathbf{u}(\gamma), p(\gamma))$, which leads to $\frac{\partial\lambda_k}{\partial\gamma} = \sum_{i=1}^{N} \frac{\partial\lambda_k}{\partial u_{1,i}}\frac{u_{1,i}}{\gamma} + \frac{\partial\lambda_k}{\partial u_{2,i}}\frac{u_{2,i}}{\gamma} + \sum_{i=1}^{M}\frac{\partial\lambda_k}{\partial p_i}\frac{p_i}{\gamma}$. This formulation is actually useless if no terms integrated over Σ_D appears due to the functional, but if one or more does, theirs effect can be collected in the Dirichlet BCs by gathering the corresponding derivative ($\frac{p_i}{\gamma}$, since such term must be taken from the $p_{a,i}$ equations of the adjoint, just because the others are weighted on the w_i test functions, which are by definition zero in Σ_D, so any term would disappear) and setting to zero $R_{l,k}u_{al,k}\frac{\partial\lambda_k}{\partial u_{l,k}} + term = 0$, $l = 1, 2$. The final adjoint system for all functionals of the just presented type is

$$\begin{cases} -\nu\Delta u_a - (\mathbf{u}\cdot\nabla)\cdot\mathbf{u}_a + \nabla\mathbf{u}\cdot\mathbf{u}_a + \dfrac{1}{\rho}\nabla p_a + \alpha\mathbf{u}_a = -\dfrac{1}{\rho}\dfrac{\partial A}{\partial\mathbf{u}} + \dfrac{1}{\rho}\nabla\cdot\dfrac{\partial A}{\partial\nabla\mathbf{u}}, & in\ \Omega \\ \nabla\cdot\mathbf{u}_a = \dfrac{\partial A}{\partial p} & in\ \Omega \\ \mathbf{u}_a = -\dfrac{\partial B}{\partial p}\mathbf{n}, & on\ \Sigma_D \\ [\dfrac{p_a}{\rho}\mathbf{I} - \nu\nabla\mathbf{u}_a]\cdot\mathbf{n} = (\mathbf{u}\cdot\mathbf{n})\mathbf{u}_a + \dfrac{1}{\rho}\dfrac{\partial A}{\partial\nabla\mathbf{u}}\mathbf{n} + \dfrac{1}{\rho}\dfrac{\partial B}{\partial\mathbf{u}}, & on\ \Sigma_N. \end{cases} \tag{5.2}$$

We are now reporting the most interesting possibilities for the functional ([16]):

Power dissipation

Let's first study the case of minimization of the power dissipation inside the domain, only case really simulated in this paper. In such case

$$J = \int_{\Omega} \left[\frac{1}{2}\mu ||\nabla \mathbf{u} + (\nabla \mathbf{u})^T||^2 + \alpha ||\mathbf{u}||^2 \right] d\mathbf{r}, \tag{5.3}$$

hence $A = \frac{1}{2}\mu ||\nabla \mathbf{u} + \nabla^T \mathbf{u}||^2 + \alpha ||\mathbf{u}||^2$. This formulation, valid in absence of heat forces and adiabatic BCs, can be derived as in [16], by scalar multiplying the velocity field to the momentum equation and using again the integration by parts rule, but in a simpler way, it's just the sum of the power dissipation due to the viscosity (proportional to the stress tensor $\frac{\nabla \mathbf{u} + \nabla^T \mathbf{u}}{2}$), and the power of the external force field, which in our case is $\alpha \mathbf{u}$ times the velocity. Applying the divergence theorem to the functional definition, one can easily check that for a steady incompressible fluid above formulation is equivalent to

$$J = \int_{\partial\Omega} [\sigma \cdot \mathbf{u} - \frac{1}{2}\rho ||\mathbf{u}||^2 \mathbf{u}] \cdot \mathbf{n} d\sigma, \tag{5.4}$$

where $\sigma = \mu \frac{\nabla \mathbf{u} + \nabla^T \mathbf{u}}{2} - p\mathbf{I}$ is the Cauchy stress tensor for an incompressible Newtonian fluid. $\sigma \cdot \mathbf{n}$ is thus the external force acting on the system boundary, and $\sigma \cdot \mathbf{u} \cdot \mathbf{n}$ it's the work done on the system by this force, that I want as close as possible to the the work done by the dynamic pressure $\frac{1}{2}\rho ||\mathbf{u}||^2 \cdot \mathbf{u} \cdot n$.

In addition, in the case of solid wall BCs on the wall boundary $\mathbf{u} = 0$ everywhere in $\partial\Omega$, above formulation reduces to

$$J = -\int_{\partial\Omega} [p + \frac{1}{2}\rho ||\mathbf{u}||^2] \mathbf{u} \cdot \mathbf{n} d\sigma. \tag{5.5}$$

Borrvall and Petersson showed [2] that for Stokes flow with Dirichlet boundary conditions everywhere on the boundary $\partial\Omega$, the problem of minimizing the total power dissipation inside the domain subject to a volume constraint on the material distribution is mathematically well-posed. Moreover it was proven that in the case where $\alpha(\gamma)$ is a linear function, the optimal material distribution is fully discrete-valued. Convexity instead implies that the slope (negative) value of α at $\gamma = 0$ is larger than at $\gamma = 1$; therefore there will be a neighbourhood around the discrete interface where it pays to move material from the solid side to the void.
In our case the interpolation equation is (2.2 and we have $\alpha'(0) = (\alpha_{min} - \alpha_{max})\frac{(1+q)}{q}$ and $\alpha'(1) = (\alpha_{min} - \alpha_{max})\frac{q}{1+q}$. Hence, for large values of q the interpolation is almost linear and we expect almost discrete interfaces, whereas for small q we expect smeared out interfaces in the optimized solution. Many hydraulic optimization problems could be formulated as the minimization of the dissipation power in order to use such theoretical results. For example, when (5.5) holds, if the system is driven with a prescribed flow rate then minimizing the total power dissipation is clearly equivalent to minimizing the pressure drop across the system, and if the system is driven at a prescribed pressure drop, then the natural design objective will be to maximize the flow rate which is equivalent to maximizing the dissipated power.

Inverse design

Here, the quadratic deviation from a target velocity distribution $\bar{\mathbf{u}}_{trg}$ is considered as a sample for a volume-declared objective functional

$$J = \int_{\Omega} ||\mathbf{u} - \bar{\mathbf{u}}_{trg}||^2 d\mathbf{r} \tag{5.6}$$

The search for a layout that optimally reproduces a predefined flow is often called "inverse design"

Velocity component in a point

An interesting case of study is the choice of the functional as a term multiplied by a Dirac delta in a fixed point, hence optimizing the corresponding value of the term in the fixed chosen point. For example, we could choose to minimize a velocity component in a point $\mathbf{r}^*$ just by choosing $A = \mathbf{u}(\mathbf{r})\delta(\mathbf{r} - \mathbf{r}^*)$.

Weighted criteria

Provided that some fixed weights β, we can redefine a multi-objective optimization problem as a single scalar one

$$J = \sum_{i=1}^{N_o} \beta_i J_i, \quad \sum_{i=1}^{N_o} \beta_i = 1. \tag{5.7}$$

The individual contributions J_i may either share the same objective surface (volume), or may be declared on different surfaces (volumes).
As said above, for brevity, in this paper we will only analyze the results for the minimization of the power dissipation.

6 Method of moving asymptotes (MMA)

Th method of moving asymptotes, MMA, is an extremely efficient algorithm for constrained non linear optimization, developed by Krister Svamberg ([17]-[18]) based on a special type of convex approximation. Here we won't develop all the computations needed to completely define the method. The code used in our schemes has been adapted from the MATLAB version in 99 lines reported in [11].

6.1 General description of the method

Consider an optimization problem (in a standard form) :
P: minimize

$$f_0(\mathbf{x}) \quad (\mathbf{x} \in \mathscr{R}^N),$$

subject to

$$f_i(\mathbf{x}) \leq \hat{f}_i, \quad for\ i = 1, ..., M$$
$$\underline{x}_j \leq x_j \leq \bar{x}_j, \quad j = 1, ..., N,$$

where $\mathbf{x} = (x_1, ..., x_N)^T$ is the vector of design variable ($x_j = \gamma_j$ in our case), $f_0(\mathbf{x})$ is the objective function (J in our case), and $f_(\mathbf{x}) \leq \hat{f}_i$ are the constraints, with the additional lower and upper bounds for each variables $\underline{x}_j$ and $\bar{x}_j$. In short, the MMA scheme follow this iterative scheme:

1. Choose a starting point $\mathbf{x}^0$, and let the iteration index k = 0.
2. Given an iteration point $\mathbf{x}^{(k)}$, compute $f_i^{(x^{(k)})}$ and the gradients $\nabla f_i(\mathbf{x}^{(k)})$ for $i = 1, ..., M$.
3. Generate a subproblem $P^{(k)}$ by replacing, in P, the (usually implicit) functions f_i by approximating explicit functions $f_i^{(k)}$, based on the calculations from step 2.

4. Solve $P^{(k)}$ and let the optimal solution of this subproblem be the next iteration point $x^{(k+1)}$. Let $k = k + 1$ and go to step 2.

The algorithm stops when the chosen convergence criteria is fulfilled (in our case is a test on the solutions difference between consecutive iterations). Substantially, each $f_i^{(k)}$ is obtained by a linearization of f_i in variables of the type $1/(x_j - L_j)$ or $(U_j - x_j)$ dependent on the sings of the derivatives of f_i at $\mathbf{x}^{(k)}$. L_j and U_j are usually called "moving asymptotes" and are free to change between iterations.

6.2 Some insights of the method

Looking at above schematization, is clear that to define the method we must describe how the functions $f_i^{(k)}$ are defined and how each subproblem $p^{(k)}$ should be solved. We will only analyze the first question, just giving the brief idea for the second.
Assuming to have chosen $L_k^{(k)}$ and $U_j^{(k)}$ fot the current iteration k, such that $L_j^{(k)} < x_j^{(k)} < U_j^{(k)}$, for each $i = 0, ..., M$, we define $f_i^{(k)}$ as

$$f_i^{(k)}(\mathbf{x}) = r_i^{(k)} + \sum_{j=1}^{N} \left(\frac{p_{ij}^{(k)}}{U_j^{(k)} - x_j} + \frac{q_{ij}^{(k)}}{x_j - L_j^{(k)}} \right), \tag{6.1}$$

where

$$p_{ij}^{(k)} = \begin{cases} (U_j^{(k)} - x_j^{(k)})^2 \partial f_i/\partial x_j, & if\ \partial f_i/\partial x_j > 0 \\ 0, & if\ \partial f_i/\partial x_j \leq 0 \end{cases} \tag{6.2}$$

$$q_{ij}^{(k)} = \begin{cases} 0, & if\ \partial f_i/\partial x_j \geq 0 \\ -(x_j^{(k)} - L_j^{(k)})^2 \partial f_i/\partial x_j, & \partial f_i/\partial x_j < 0 \end{cases} \tag{6.3}$$

$$\mathbf{r}_i^{(k)} = f_i(\mathbf{x}^{(k)}) - \sum_{j=1}^{N} \left(\frac{p_{ij}^{(k)}}{U_j^{(k)} - x_j(k)} + \frac{q_{ij}^{(k)}}{x_j(k) - L_j^{(k)}} \right) \tag{6.4}$$

and all derivatives $\partial f_i/\partial x_j$ are evaluated at $\mathbf{x} = \mathbf{x}^{(k)}$. With this formulation, $f_i^{(k)}$ is a first order approximation of f_i at $\mathbf{x}^{(k)}$,

$$f_i^{(k)}(\mathbf{x}^{(k)}) = f_i(\mathbf{x}^{(k)}) \quad \text{and} \quad \partial f_i^{(k)}/\partial x_j = \partial f_i/\partial x_j \ at\ \mathbf{x} = \mathbf{x}^{(k)}, \quad for\ i = 0, ..., M,\ j = 1, ..., N.$$

It can be easily checked that since $p_{ij}^{(k)},\ q_{ij}^{(k)} \geq 0$, $f_i^{(k)}$ is a convex function, with second derivatives at $\mathbf{x} = \mathbf{x}^{(k)}$

$$\frac{\partial^2 f_i^{(k)}}{\partial x_j^2} = \begin{cases} \dfrac{2\partial f_i/\partial x_j}{U_j^{(k)} - x_j^{(k)}}, & if\ \partial f_i/\partial x_j > 0 \\ -\dfrac{2\partial f_i/\partial x_j}{x_j^{(k)} - L_j^{(k)}}, & if\ \partial f_i/\partial x : j < 0. \end{cases} \tag{6.5}$$

In particular, this means that the closer $L_j^{(k)}$ and $U_j^{(k)}$ are chosen to $x_j^{(k)}$, the larger will become the second derivatives, and therefore the more "curved" will be those functions and the more conservative will become the approximation of the original problem (since the approximation will be valid only close to the points $\mathbf{x}^{(k)}$). On the opposite, when $L_j^{(k)}$ and $U_j^{(k)}$ are chosen far from $\mathbf{x}^k$ then the approximating functions become more linear.
Assuming the lower and upper bounds $\underline{x}_j$ and $\bar{x}_j$ are phisically reasonable, as simple choice for $L_j^{(k)}$ and $U_j^{(k)}$ is

$$L_j^{(k)} = x_j - s_0(\bar{x}_j - \underline{x}_j), \quad and \quad U_j^{(k)} = \bar{x}_j + s_0(\bar{x}_j - \underline{x}_j), \tag{6.6}$$

7with a fixed real number $s_0 = 0 - 1$. The problem of this formulation is that the aymptotes would be fixed for each iteration, when it would be more clever to allow them to vary according to some reasonable rules.
An heuristical idea would be to, accordingly to the feature explained above,

- move the asymptotes closer to the current iteration point when the process tends to oscillate and needs to be stabilized.
- move the asymptotes away from the current iteration point if the process is monotone and slow, and needs to be relaxed.

With such approximating functions, comes the $P^{(k)}$ subproblem,
$P^{(k)}$: minimize

$$\sum_{j=1}^{N} \left(\frac{p_{0j}^{(k)}}{U_j^{(k)} - x_j} + \frac{q_{0j}^{(k)}}{x_j - L_j^{(k)}} \right) + r_0^{(k)}$$

subject to

$$\sum_{j=1}^{N} \left(\frac{p_{ij}^{(k)}}{U_j^{(k)} - x_j} + \frac{q_{ij}^{(k)}}{x_j - L_j^{(k)}} \right) + r_i^{(k)} \leq \hat{f}_i, \quad for\ i = 1, ..., M$$
$$\underline{x}_j < x_j < \bar{x}_j, \quad for\ j = 1, ..., N.$$

From this formulation the solution is usually obtained by a primal-dual algorithm, applying the Karush Kuhn Tacker conditions to the dual and solving them for example with a Fletcher Reeves gradient method. More details are given in [17] and [18], where is explained the final algorithm used in our simulations, which is based on the same approach, but with some additional artificial variables used to prevent the code from stopping at the first infeasibility due to a bad chosen starting point $\mathbf{x}^{(0)}$.

7 Adjoint system derivation

7.1 Introduction to PDE constrained optimization

Let us now introduce some the most relevant concepts in order to formulate our topology optimization problem as a control problem with PDE constraints.
The discussion will follow the arguments as they're presented in [9].

Formulation of control-constrained problems

Consider a general non-linear problem of the form:

$$\min_{(y,u) \in Y \times U} J(y, u) \quad \text{subject to} \quad e(y, u) = 0, \quad u \in U_{ad} \subseteq U. \tag{7.1}$$

Assumption 7.1. Let's make the following assumptions:

1. $U_{ad} \subset U$ is nonempty, convex and closed.

2. $J : Y \times U \to \mathbb{R}$ and $e : Y \times U \to Z$ are continuously differentiable, with U,Y,Z Banach Spaces.
3. For all $u \in V, V \subset U$, neighborhood of U_{ad}, the state equation $e(y,u) = 0$ has unique solution $y = y(u) \in Y$.
4. $e_y(y(u),u) \in \mathcal{L}(Y,Z)$ has bounded inverse $\forall u \in V \supset U_{ad}$.

Definition. ***Fréchet Differentiability***
Let V, W be normed vector spaces and $U \subseteq V$, open set. A function $f : U \to W$ is said Fréchet differentiable at $x \in U$ if there exists a bounded linear operator $D : V \to W$ such that:

$$\lim_{||h||_V \to 0} \frac{||f(x+h) - f(x) - Dh||_W}{||h||_V} = 0.$$

Definition. ***Gateaux Derivative***
Let V, W be normed vector spaces and $U \subseteq V$, open set.
The Gateaux derivative of a function $f : U \to W$ at $x \in U$ is defined as:

$$g(x) = \lim_{t \to 0} \frac{f(x+th) - f(x)}{t}$$

Observation. *If a function is Fréchet differentiable at a point then it's also Gateaux differentiable and we have: $g(x) = D = D_f(x)$, the differential of f.*

Under the previous assumptions the mapping $u \in V \mapsto Y$ is continuously Fréchet differentiable by the implicit function theorem (Dini's theorem).

General first order optimality condition

Consider the general problem (7.1) and let Assumption 7.1 hold.
The reduced problem can now be formulated.

$$\min_{u \in U} \hat{J}(u) \quad \text{subject to} \quad u \in U_{ad} \subseteq U. \tag{7.2}$$

With the reduced functional

$$\hat{J}(u) \coloneqq J(y(u), u), \tag{7.3}$$

where $V \ni u \mapsto y(u) \in Y$ is the solution operator of the state equation.

Now some classical results are stated in order to justify the use of the adjoint formulation method.

Theorem 7.2. *Let assumption 7.1 hold. If $\bar{u}$ is a local solution of the reduced problem (7.2) then $\bar{u}$ satisfies the following variational inequality:*

$$\bar{u} \in U_{ad} \quad \textit{and} \quad \left\langle \hat{J}(\bar{u}), u - \bar{u} \right\rangle_{U^*,U} \geq 0 \quad \forall u \in U_{ad}. \tag{7.4}$$

The previous variational inequality directly yields the following necessary first order condition.

$$\frac{d}{du}\hat{J}(\bar{u}) = \frac{\partial J}{\partial y}\frac{dy}{du}\Big|_{(y(\bar{u}),\bar{u})} + \frac{\partial J}{\partial u}\Big|_{(y(\bar{u}),\bar{u})} = 0. \tag{7.5}$$

Observation 7.3. *Condition (7.5)'s direct computation can result a complex task cause, specially in the PDE constrained case (e.g. Navier Stokes equations), the dependence of y with respect to u is not directly computable. Moreover since the scope of this paper is to get a domain topology optimization using numerical methods to solve both constraints and variational conditions the analytical form of y isn't known and so this solution strategy couldn't be applied.*

Finally we will use the adjoint representation of the functional derivative

$$\hat{J}'(u) = e_u(y(u), u)^* \lambda(u) + J_u(y(u), u) \tag{7.6}$$

where $\lambda(u) \in Z^*$ is the adjoint state, solving the adjoint equation

$$e_y(y(u), (u))^* \lambda(u) = -J_y(y(u), u). \tag{7.7}$$

leading to the formulation of an adjoint differential system to be solved in order to proceed with the numerical solution of the variational problem.

Let us now state an optimality condition for the adjoint formulation.

Theorem 7.4. *Let $(\bar{y}, \bar{u})$ an optimal solution of the problem* (7.1) *and let assumption 7.1 hold. Then there exists an adjoint state $\bar{\lambda} \in Z^*$ such that the following optimality conditions hold*

$$e(\bar{y}, \bar{u}) = 0; \tag{7.8}$$

$$e_y(\bar{y}, \bar{u})^* \bar{\lambda} = -J_y(\bar{y}, \bar{u}); \tag{7.9}$$

$$\bar{u} \in U_{ad}, \quad \left\langle J_u(\bar{y}, \bar{u}) + e_u(\bar{y}, \bar{u})^* \bar{\lambda}, u - \bar{u} \right\rangle_{U^*, U} \geq 0. \tag{7.10}$$

Let us now consider a more general formulation of the problem in (7.1):

$$\min_{w \in W} J(w) \quad \text{subject to} \quad G(w) \in \mathcal{K}_G, \quad w \in \mathcal{C}. \tag{7.11}$$

Thanks to the results obtained in [15](*Robinson's regularity condition*) we can now state an important theorem, playing the role of the Karush-Kuhn-Tucker conditions, introduced in [6] about the optimal solution for problem (7.11).

Theorem 7.5. *(Zowe and Kurcyusz) Let $J : W \to \mathbb{R}, G : W \to V$ be continuously Fréchet differentiable functions with Banach-spaces W, V. Further let $\mathcal{C} \subset W$ be non-empty, closed and convex, and let $\mathcal{K}_G \subset V$ be a closed convex cone. Then for any local solution $\bar{w}$ of* (7.11) *at which Robinson's regularity condition is satisfied, the following optimality condition holds:*
There exist a Lagrange multiplier $\bar{q} \in V^*$ with

$$G(\bar{w}) \in \mathcal{K}_G, \tag{7.12}$$

$$\bar{q} \in \mathcal{K}_G^\circ := \left\{ q \in V^* \middle| \langle q, v \rangle_{V^*, V} \leq 0 \, \forall v \in \mathcal{K}_G \right\}, \tag{7.13}$$

$$\langle \bar{q}, G(\bar{w}) \rangle_{V^*, V} = 0, \tag{7.14}$$

$$\bar{w} \in \mathcal{C}, \quad \langle J'(\bar{w}) + G'(\bar{w})^* \bar{q}, w - \bar{w} \rangle_{W^*, W} \geq 0 \ \ \forall w \in \mathcal{C}. \tag{7.15}$$

Observation 7.6. *Using the Lagrangian function*

$$L(w, q) := J(w) + \langle q, G(w) \rangle_{V^*, V} \tag{7.16}$$

we can write (7.15) *in the following more compact form*

$$\bar{w} \in \mathcal{C}, \quad \langle L_w(\bar{w}, \bar{q}), w - \bar{w} \rangle_{W^*, W} \geq 0 \ \ \forall w \in \mathcal{C}. \tag{7.17}$$

Application to PDE-constrained optimization

Consider now an optimal control problem like (7.1), for which the state equation, $e(y, u) = 0$, is a PDE (*e.g. 2-dim Navier-Stokes*), requiring also $c(y) \in \mathcal{K}$ similarly to (7.11).

$$\min_{(y,u) \in Y \times U} J(y, u) \quad \text{subject to} \quad e(y, u) = 0, \ c(y) \in \mathcal{K}, \ u \in U_{ad} \subseteq U. \tag{7.18}$$

where $e : Y \times U \to Z$ and $c : Y \to R$ are continuously Fréchet differentiable, $\mathcal{K} \subset R$ is a closed convex cone and $U_{ad} \subset U$ is a closed convex set.
To keep a coherent notation we define

$$G : Y \times U \ni \begin{pmatrix} y \\ u \end{pmatrix} \mapsto \begin{pmatrix} e(y,u) \\ c(y) \end{pmatrix} \in Z \times R$$
$$\mathcal{K}_G = \{0\} \times \mathcal{K}, \qquad \mathcal{C} = Y \times U_{ad}$$

Let us now formulate the Lagrangian function, with the multiplier in the form $(\lambda, \eta) \in Z^* \times R^*$

$$\begin{aligned} \mathcal{L}(y,u,\lambda,\eta) &= J(y,u) + \langle \lambda, e(y,u) \rangle_{Z^*,Z} + \langle \eta, c(y) \rangle_{R^*,R} = \\ &= L(y,u,\lambda) + \langle \eta, c(y) \rangle_{R^*,R} \end{aligned}$$

with the Lagrangian restricted at the equality constraints

$$L(y,u,\lambda) = J(y,u) + \langle \lambda, e(y,u) \rangle_{Z^*,Z}.$$

Since $\mathcal{K}_G = \{0\} \times \mathcal{K}$ we obtain $\mathcal{K}_G^\circ = Z^* \times \mathcal{K}^\circ$, then, assuming the Robinson's regularity condition holds (*proved in* [9]) the optimality necessary conditions of theorem 7.5 become:

$$\begin{aligned} &e(\bar{y},\bar{u}) = 0, \qquad c(\bar{y}) \in \mathcal{K}, \\ &\bar{\eta} \in \mathcal{K}^\circ, \quad \langle \bar{\eta}, c(\bar{y}) \rangle_{R^*,R} = 0, \\ &\langle L_y(\bar{y},\bar{u},\bar{\lambda}) + c'(\bar{y})^* \bar{\eta}, y - \bar{y} \rangle_{Y^*,Y} \geq 0 \quad \forall y \in Y, \\ &\bar{u} \in U_{ad}, \quad \langle L_u(\bar{y},\bar{u},\bar{\lambda}), u - \bar{u} \rangle_{U^*,U} \geq 0 \quad \forall u \in U_{ad}. \end{aligned}$$

That finally yields to the corresponding of the Karush-Kuhn-Tacker conditions for PDE constrained control problems:

$$e(\bar{y},\bar{u}) = 0, \qquad c(\bar{y}) \in \mathcal{K}, \tag{7.19}$$
$$\bar{\eta} \in \mathcal{K}^\circ, \quad \langle \bar{\eta}, c(\bar{y}) \rangle_{R^*,R} = 0, \tag{7.20}$$
$$L_y(\bar{y},\bar{u},\bar{\lambda}) + c'(\bar{y})^* \bar{\eta} = 0, \tag{7.21}$$
$$\bar{u} \in U_{ad}, \quad \langle L_u(\bar{y},\bar{u},\bar{\lambda}), u - \bar{u} \rangle_{U^*,U} \geq 0 \quad \forall u \in U_{ad}. \tag{7.22}$$

7.2 Continuous adjoint formulation

The following arguments will be developed according to [19] discussion.
Consider now a state variable of the form $\begin{pmatrix} \mathbf{u} \\ p \end{pmatrix}$ and a control γ.
We can now formulate a general topology optimization problem with PDE constraints as:

$$\min_{(y,u) \in Y \times U} J(\mathbf{u}, p; \gamma) \quad \text{s. t.} \quad e(\mathbf{u}, p; \gamma) = 0, \ \gamma \in \mathcal{K} \tag{7.23}$$

where $\mathcal{K}$ is the feasible space of the design optimization variable γ and $e(\cdot)$ is the weak operator of the PDE.
Since problem (7.23) satisfies the hypothesis of theorem 7.5, according to the Karush-Kuhn-Tacker conditions for PDE constrained optimization problems [9] the solution of the optimization problem (7.23) must satisfy

$$e(\mathbf{u}, p; \gamma) = 0 \tag{7.24}$$
$$\begin{pmatrix} \big(e_{\mathbf{u}}(\mathbf{u}, p; \gamma)\big)^* & 0 \\ 0 & \big(e_p(\mathbf{u}, p; \gamma)\big)^* \end{pmatrix} \begin{pmatrix} \mathbf{u}_a \\ p_a \end{pmatrix} = \begin{pmatrix} -J_{\mathbf{u}}(\mathbf{u}, p; \gamma) \\ -J_p(\mathbf{u}, p; \gamma) \end{pmatrix} \tag{7.25}$$
$$\big(e_\gamma(\mathbf{u}, p; \gamma)\big)^* (\mathbf{u}_a, p_a) + J_\gamma(\mathbf{u}, p; \gamma) = 0. \tag{7.26}$$

where $\mathbf{u}_a$ and p_a are the adjoint variable for $\mathbf{u}$ and p respectively and $e(\cdot)^*$ is the adjoint(dual) operator of $e(\cdot)$.
As developed in [9] the dual operator of a linear one is defined as

$$\left\langle D^* u, v \right\rangle_{\mathscr{H}^*,\mathscr{H}} = \left\langle u, Dv \right\rangle_{\mathscr{Y}^*,\mathscr{Y}}, \quad \forall u \in \mathscr{Y}^*, v \in \mathscr{H} \tag{7.27}$$

where $\mathscr{H}$ and $\mathscr{Y}$ are Banach spaces, $D \in \mathscr{L}(\mathscr{H}, \mathscr{Y})$ and $D^* \in \mathscr{L}(\mathscr{Y}^*, \mathscr{H}^*)$.

7.3 (NS)-constrained topology optimization

Let's consider problem 7.23 where $e(\cdot)$ is the weak operator of time-dependent Navier-Stokes equations and $\mathcal{K} = [0, 1]$.

$$\begin{aligned} &\min_{(y,u)\in Y\times U} J(\mathbf{u}, p; \gamma) \quad \text{s. t.} \\ &\begin{cases} e(\mathbf{u}, p; \gamma) = 0, \\ \gamma \in [0, 1]. \end{cases} \end{aligned} \tag{7.28}$$

where

$$J(\mathbf{u}, p; \gamma) = \int_0^T \int_\Omega \beta_1 A(\mathbf{u}, \nabla\mathbf{u}, p; \gamma) dt + \int_0^T \int_{\partial\Omega} \beta_2 B(\mathbf{u}, p; \gamma) d\sigma dt, \tag{7.29}$$

with β_1, β_2 space-independent model parameters.

$$\begin{aligned} e(\mathbf{u}, p; \gamma) = &\int_0^T \int_\Omega \Big[\rho\frac{\partial}{\partial t} - \eta\Delta\mathbf{u} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u}) + \nabla p - \mathbf{f}\Big] \cdot \mathbf{v}\, d\Omega dt + \\ &- \int_0^T \int_\Omega q\nabla\cdot\mathbf{u}\, d\sigma dt + \int_0^T \int_{\Sigma_D} \mathbf{u}\cdot\mathbf{v}\, d\Omega dt + \\ &+ \int_0^T \int_{\Sigma_N} \Big[-p\mathbf{I} + \eta\nabla\mathbf{u}\Big]\mathbf{n}\cdot\mathbf{v}\, d\sigma dt + \int_\Omega (\mathbf{u}\cdot\mathbf{v})\big|_{t=0} d\Omega. \end{aligned} \tag{7.30}$$

The functional spaces for the variables are chosen to be

$$\begin{aligned} &\mathbf{u} \in \mathscr{U}_\Omega^d \coloneqq \Big(\mathscr{L}^2\big((0,T); \mathscr{H}(\Omega)\big)\Big)^d; \quad p \in \mathscr{P}_\Omega \coloneqq \mathscr{L}^2\big((0,T); \mathscr{L}^2(\Omega)\big) \\ &\mathbf{f} \in \mathscr{F}_\Omega^{d*} \coloneqq \Big(\mathscr{L}^2\big((0,T); \mathscr{H}^*(\Omega)\big)\Big)^d; \quad \mathbf{u}_0 \in \big(\mathscr{H}_0^1(\Omega)\big)^d; \quad \gamma \in \mathscr{L}^1(\Omega) \end{aligned}$$

where d is the spatial dimension; $\mathscr{L}^1(\Omega)$ and $\mathscr{L}^2(\Omega)$ are the first-order and second-order integrable Lebesgue spaces respectively; $\mathscr{H}^*(\Omega)$ is the dual space of the first order Hilbert space $\mathscr{H}(\Omega)$; and $\big(\mathscr{H}_0^1(\Omega)\big)^d = \big\{\mathbf{u} \in \big(\mathscr{H}(\Omega)\big)^d \big| \nabla\cdot\mathbf{u} = 0\big\}$. According to Rietz representation theorem and the Hölder inequality [20], we obtain the Bochner space $\mathscr{V}_\Omega$ is reflexive, i.e. $\mathscr{V}_\Omega^* = \mathscr{V}_\Omega$.
Finally, using the adjoint analysis method, the sensitivity analysis of problem 7.28 can be derived [21] as:

$$\left\langle u^*, \int_0^T f(t) dt \right\rangle_{\mathscr{H}^*,\mathscr{H}} = \int_0^T \left\langle u^*, f(t) \right\rangle_{\mathscr{H}^*,\mathscr{H}} dt \tag{7.31}$$

where $\mathscr{H}$ is a Bochner space, and $\mathscr{H}$ its dual, $u^* \in \mathscr{H}^*$, $f : (0, T) \to \mathscr{H}$ is Bochner-integrable, hence the integration on time and space can be switched.

Hence we get

$$\begin{aligned}
e(\mathbf{u},p;\gamma) =&\rho\int_\Omega\int_0^T\Big[\frac{\partial(\mathbf{u}\cdot\mathbf{v})}{\partial t}-\mathbf{u}\cdot\frac{\partial\mathbf{v}}{\partial t}\Big]dtd\Omega+\\
&+\eta\int_0^T\int_\Omega\Big[\nabla\mathbf{u}:\nabla\mathbf{v}-\nabla\cdot\big(\nabla\mathbf{u}\cdot\mathbf{v}\big)\Big]d\Omega dt+\\
&+\int_0^T\int_\Omega\rho\big(\mathbf{u}\cdot\nabla\big)\mathbf{u}\cdot\mathbf{v}d\Omega dt+\int_0^T\int_\Omega\big[\nabla\cdot(p\mathbf{v})-p\nabla\cdot\mathbf{v}\big]d\Omega dt+\\
&-\int_0^T\int_\Omega\mathbf{f}\cdot\mathbf{v}d\Omega dt-\int_0^T\int_\Omega q\nabla\cdot\mathbf{u}d\Omega dt+\int_0^T\int_{\Sigma_D}\mathbf{u}\cdot\mathbf{v}d\sigma dt\\
&+\int_0^T\int_{\Sigma_N}\Big[-p\mathbf{I}+\eta\nabla\mathbf{u}\Big]\mathbf{n}\cdot\mathbf{v}\,d\sigma dt+\int_\Omega(\mathbf{u}\cdot\mathbf{v})\big|_{t=0}d\Omega.
\end{aligned}$$

That, according to Gauss theory in [9], become

$$\begin{aligned}
e(\mathbf{u},p;\gamma) =&\rho\int_\Omega\big[(\mathbf{u}\cdot\mathbf{v})|_{t=T}-(\mathbf{u}\cdot\mathbf{v})|_{t=0}\big]d\Omega-\rho\int_0^T\int_\Omega\mathbf{u}\cdot\frac{\partial\mathbf{v}}{\partial t}d\Omega dt+\\
&+\eta\int_0^T\int_\Omega\nabla\mathbf{u}:\nabla\mathbf{v}d\Omega dt+\int_0^T\int_\Omega\rho\big(\mathbf{u}\cdot\nabla\big)\mathbf{u}\cdot\mathbf{v}d\Omega dt+\\
&-\int_0^T\int_\Omega p\nabla\cdot\mathbf{v}d\Omega dt-\int_0^T\int_\Omega\mathbf{f}\cdot\mathbf{v}d\Omega dt+\\
&+\int_0^T\int_{\partial\Omega}\Big[p\mathbf{I}-\eta\nabla\mathbf{u}\Big]\mathbf{n}\cdot\mathbf{v}d\sigma dt-\int_0^T\int_\Omega q\nabla\cdot\mathbf{u}d\Omega dt+\\
&+\int_0^T\int_{\Sigma_D}\mathbf{u}\cdot\mathbf{v}d\Omega dt-\int_0^T\int_{\Sigma_N}\Big[p\mathbf{I}-\eta\nabla\mathbf{u}\Big]\mathbf{n}\cdot\mathbf{v}\,d\sigma dt+\int_\Omega(\mathbf{u}\cdot\mathbf{v})\big|_{t=0}d\Omega.
\end{aligned}$$

Applying now the boundary conditions we obtain the following reduced weak operator of Navier-Stokes equations

$$\begin{aligned}
e(\mathbf{u},p;\gamma) =&\rho\int_\Omega\big[(\mathbf{u}\cdot\mathbf{v})|_{t=T}d\Omega-\rho\int_0^T\int_\Omega\mathbf{u}\cdot\frac{\partial\mathbf{v}}{\partial t}d\Omega dt+\\
&+\eta\int_0^T\int_\Omega\nabla\mathbf{u}:\nabla\mathbf{v}d\Omega dt+\int_0^T\int_\Omega\rho\big(\mathbf{u}\cdot\nabla\big)\mathbf{u}\cdot\mathbf{v}d\Omega dt+\\
&-\int_0^T\int_\Omega p\nabla\cdot\mathbf{v}d\Omega dt-\int_0^T\int_\Omega\mathbf{f}\cdot\mathbf{v}d\Omega dt-\int_0^T\int_\Omega q\nabla\cdot\mathbf{u}d\Omega dt+\\
&+\int_0^T\int_{\Sigma_D}\Big[p\mathbf{I}-\eta\nabla\mathbf{u}\Big]_{\mathbf{u}=\mathbf{u}_D}\mathbf{n}\cdot\mathbf{v}d\sigma dt+\int_0^T\int_{\Sigma_D}\mathbf{u}_D\cdot\mathbf{v}d\Omega dt.
\end{aligned}\tag{7.32}$$

In order to compute the adjoint formulation we'll now compute the Gateaux derivatives in the direction $(\mathbf{w},r)\in\mathscr{U}_\Omega^d\times\mathscr{P}_\Omega$:

$$\begin{aligned}
\big\langle e_{\mathbf{u}}(\mathbf{u},p;\gamma),\mathbf{w}\big\rangle_{\mathscr{U}_\Omega^{d*},\mathscr{U}_\Omega^d}&=\lim_{l\to0^+}\frac{e(\mathbf{u}+l\mathbf{w},p)-e(\mathbf{u},p)}{l}=...=\\
&=\rho\int_\Omega\big[(\mathbf{w}\cdot\mathbf{v})|_{t=T}d\Omega-\rho\int_0^T\int_\Omega\mathbf{w}\cdot\frac{\partial\mathbf{v}}{\partial t}d\Omega dt+\\
&+\eta\int_0^T\int_\Omega\nabla\mathbf{w}:\nabla\mathbf{v}d\Omega dt+\int_0^T\int_\Omega\rho\big(\mathbf{w}\cdot\nabla\big)\mathbf{u}\cdot\mathbf{v}d\Omega dt+\\
&+\int_0^T\int_\Omega\rho\big(\mathbf{u}\cdot\nabla\big)\mathbf{w}\cdot\mathbf{v}d\Omega dt-\int_0^T\int_\Omega\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\mathbf{w}\cdot\mathbf{v}d\Omega dt-\int_0^T\int_\Omega q\nabla\cdot\mathbf{w}d\Omega dt.
\end{aligned}$$

$$\begin{aligned}
\big\langle e_p(\mathbf{u},p;\gamma),r\big\rangle_{\mathscr{P}_\Omega{}^*,\mathscr{P}_\Omega}&=\lim_{l\to0^+}\frac{e(\mathbf{u},p+lr)-e(\mathbf{u},p)}{l}=...=\\
&-\int_0^T\int_\Omega r\nabla\cdot\mathbf{v}d\Omega dt-\int_0^T\int_\Omega\frac{\partial\mathbf{f}}{\partial p}r\cdot\mathbf{v}d\Omega dt+\int_0^T\int_{\Sigma_D}r\mathbf{n}\cdot\mathbf{v}d\sigma dt
\end{aligned}$$

Now according to (7.27) we can now compute, renaming the adjoint variables $\mathbf{u}_a$ and p_a (i.e. $\mathbf{v} = \mathbf{u}_a, q = p_a$) the dual operators of $e_{\mathbf{u}}(\cdot)$ and $e_p(\cdot)$:

$$\begin{aligned}\left\langle e^*_{\mathbf{u}}(\mathbf{u}_a, p_a; \gamma), \mathbf{w}\right\rangle_{\mathscr{U}_\Omega^{d*}, \mathscr{U}_\Omega^d} &= \rho \int_\Omega \left[(\mathbf{w}\cdot\mathbf{u}_a)|_{t=T} d\Omega - \rho \int_0^T \int_\Omega \mathbf{w}\cdot\frac{\partial \mathbf{u}_a}{\partial t} d\Omega dt + \right.\\ &+ \eta \int_0^T \int_\Omega \nabla\mathbf{w} : \nabla\mathbf{u}_a d\Omega dt + \int_0^T \int_\Omega \rho\big(\mathbf{w}\cdot\nabla\big)\mathbf{u}\cdot\mathbf{u}_a d\Omega dt + \\ &+ \int_0^T \int_\Omega \rho\big(\mathbf{u}\cdot\nabla\big)\mathbf{w}\cdot\mathbf{u}_a d\Omega dt - \int_0^T \int_\Omega \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{w}\cdot\mathbf{u}_a d\Omega dt - \int_0^T \int_\Omega p_a \nabla\cdot\mathbf{w} d\Omega dt.\end{aligned}$$

$$\left\langle e^*_p(\mathbf{u}_a, p_a; \gamma), r\right\rangle_{\mathscr{P}_\Omega{}^*, \mathscr{P}_\Omega} = -\int_0^T \int_\Omega r\nabla\cdot\mathbf{u}_a d\Omega dt - \int_0^T \int_\Omega \frac{\partial \mathbf{f}}{\partial p} r\cdot\mathbf{u}_a d\Omega dt + \int_0^T \int_{\Sigma_D} r\mathbf{n}\cdot\mathbf{u}_a d\sigma dt$$

Observation 7.7. *Applying inverse Gauss theory and integrating by parts we get:*

$$\int_0^T \int_\Omega \nabla\mathbf{w} : \nabla\mathbf{u}_a d\Omega dt = -\int_0^T \int_\Omega \Delta\mathbf{u}_a\cdot\mathbf{w} d\Omega dt + \int_0^T \int_{\Sigma_N} \nabla\mathbf{u}_a\mathbf{n}\cdot\mathbf{w} d\sigma dt$$

$$\int_0^T \int_\Omega p_a \nabla\mathbf{w} d\Omega dt = -\int_0^T \int_\Omega \nabla p_a\cdot\mathbf{w} d\Omega dt + \int_0^T \int_{\Sigma_N} p_a\mathbf{n}\cdot\mathbf{w} d\sigma dt$$

$$\begin{aligned}\int_0^T \int_\Omega \big(\mathbf{u}\cdot\nabla\big)\mathbf{w}\cdot\mathbf{u}_a d\Omega dt &= -\int_0^T \int_\Omega \big(\mathbf{u}\cdot\nabla\big)\mathbf{u}_a\cdot\mathbf{w} d\Omega dt + \int_0^T \int_\Omega \mathbf{u}\cdot\nabla(\mathbf{u}_a\cdot\mathbf{w}) d\Omega dt = \\ &= -\int_0^T \int_\Omega \big(\mathbf{u}\cdot\nabla\big)\mathbf{u}_a\cdot\mathbf{w} d\Omega dt + \\ &\quad + \int_0^T \int_\Omega \nabla\cdot\big(\mathbf{u}(\mathbf{u}_a\cdot\mathbf{w})\big) d\Omega dt - \int_0^T \int_\Omega \cancelto{0}{\nabla\cdot\mathbf{u}}(\mathbf{u}_a\cdot\mathbf{w}) d\Omega dt = \\ &= -\int_0^T \int_\Omega \big(\mathbf{u}\cdot\nabla\big)\mathbf{u}_a\cdot\mathbf{w} d\Omega dt + \int_0^T \int_{\Sigma_N} (\mathbf{u}\cdot\mathbf{n})\mathbf{u}_a\cdot\mathbf{w} d\sigma dt.\end{aligned}$$

The idea now is to apply (7.25), in order to do that we must compute the weak(Gateaux) derivatives of the objective functional

$$\begin{aligned}\left\langle J_{\mathbf{u}}(\mathbf{u}, p; \gamma), \mathbf{w}\right\rangle_{\mathscr{U}_\Omega^{d*}, \mathscr{U}_\Omega^d} &= \lim_{l\to 0} \frac{J(\mathbf{u}+l\mathbf{w}, p; \gamma) - J(\mathbf{u}, p; \gamma)}{l} = \\ &= \lim_{l\to 0} \frac{\int_0^T \int_\Omega \beta_1 A(\mathbf{u}+l\mathbf{w}, \nabla(\mathbf{u}+l\mathbf{w}), p; \gamma) d\Omega dt - \int_0^T \int_\Omega \beta_1 A(\mathbf{u}, \nabla\mathbf{u}, p; \gamma) d\Omega dt}{l} + \\ &+ \lim_{l\to 0} \frac{\int_0^T \int_{\partial\Omega} \beta_2 B(\mathbf{u}+l\mathbf{w}, p; \gamma) d\sigma dt - \int_0^T \int_{\partial\Omega} \beta_2 B(\mathbf{u}, p; \gamma) d\sigma dt}{l},\end{aligned}$$

since the integrals are independent of the limit variable we can interchange them and get

$$\begin{aligned}\left\langle J_{\mathbf{u}}(\mathbf{u}, p; \gamma), \mathbf{w}\right\rangle_{\mathscr{U}_\Omega^{d*}, \mathscr{U}_\Omega^d} &= \int_0^T \int_\Omega \beta_1 \lim_{l\to 0} \frac{A(\mathbf{u}+l\mathbf{w}, \nabla(\mathbf{u}+l\mathbf{w}), p; \gamma) - A(\mathbf{u}, \nabla\mathbf{u}, p; \gamma)}{l} d\Omega dt + \\ &+ \int_0^T \int_{\partial\Omega} \beta_2 \lim_{l\to 0} \frac{B(\mathbf{u}+l\mathbf{w}, p; \gamma) - B(\mathbf{u}, p; \gamma)}{l} d\sigma dt.\end{aligned}$$

Now, omitting the dependencies for the sake of simplicity, observe

$$\begin{aligned}&\lim_{l\to 0} \frac{A(\mathbf{u}+l\mathbf{w}, \nabla(\mathbf{u}+l\mathbf{w}), p; \gamma) - A(\mathbf{u}, \nabla\mathbf{u}, p; \gamma)}{l} = \\ &= \lim_{l\to 0} \frac{A(\mathbf{u}+l\mathbf{w}, \nabla(\mathbf{u}+l\mathbf{w}), p; \gamma) - A(\mathbf{u}+l\mathbf{w}, \nabla\mathbf{u}, p; \gamma) + A(\mathbf{u}+l\mathbf{w}, \nabla\mathbf{u}, p; \gamma) - A(\mathbf{u}, \nabla\mathbf{u}, p; \gamma)}{l} = \\ &= \frac{\partial A}{\partial \mathbf{u}}\cdot\mathbf{w} + \frac{\partial A}{\partial \nabla\mathbf{u}} : \nabla\mathbf{w}.\end{aligned}$$

Therefore we get:

$$\big\langle J_{\mathbf{u}}(\mathbf{u},p;\gamma),\mathbf{w}\big\rangle_{\mathscr{U}_\Omega^{d*},\mathscr{U}_\Omega^d}=\int_0^T\int_\Omega\Big[\frac{\partial A}{\partial\mathbf{u}}\cdot\mathbf{w}+\frac{\partial A}{\partial\nabla\mathbf{u}}:\nabla\mathbf{w}\Big]d\Omega dt+\int_0^T\int_{\partial\Omega}\beta_2\frac{\partial B}{\partial\mathbf{u}}\cdot\mathbf{w}d\sigma dt.$$

But integrating by parts we observe

$$\begin{aligned}\int_0^T\int_\Omega\Big(\frac{\partial A}{\partial\nabla\mathbf{u}}:\nabla\mathbf{w}\Big)d\Omega dt&=\int_0^T\int_\Omega\Big[\nabla\cdot\Big(\frac{\partial A}{\partial\nabla\mathbf{u}}\mathbf{w}\Big)-\Big(\nabla\cdot\frac{\partial A}{\partial\nabla\mathbf{u}}\Big)\cdot\mathbf{w}\Big]d\Omega dt=\\&=\int_0^T\int_{\partial\Omega}\frac{\partial A}{\partial\nabla\mathbf{u}}\mathbf{n}\cdot\mathbf{w}d\sigma dt-\int_0^T\int_\Omega\Big(\nabla\cdot\frac{\partial A}{\partial\nabla\mathbf{u}}\Big)\cdot\mathbf{w}d\Omega dt.\end{aligned}$$

Then we get:

$$\begin{aligned}\big\langle J_{\mathbf{u}}(\mathbf{u},p;\gamma),\mathbf{w}\big\rangle_{\mathscr{U}_\Omega^{d*},\mathscr{U}_\Omega^d}&=\int_0^T\int_\Omega\beta_1\Big[\frac{\partial A}{\partial\mathbf{u}}\cdot\mathbf{w}-\Big(\nabla\cdot\frac{\partial A}{\partial\nabla\mathbf{u}}\Big)\cdot\mathbf{w}\Big]d\Omega dt+\\&+\int_0^T\int_{\partial\Omega}\beta_1\frac{\partial A}{\partial\nabla\mathbf{u}}\mathbf{n}\cdot\mathbf{w}d\sigma dt+\int_0^T\int_{\partial\Omega}\beta_2\frac{\partial B}{\partial\mathbf{u}}\cdot\mathbf{w}d\sigma dt\end{aligned}$$

Finally, since in finite element analysis is usual to consider test-functions spaces for the velocity whose elements' value at the Dirichlet boundary Σ_D is zero, we obtain the following:

$$\begin{aligned}\big\langle J_{\mathbf{u}}(\mathbf{u},p;\gamma),\mathbf{w}\big\rangle_{\mathscr{U}_\Omega^{d*},\mathscr{U}_\Omega^d}&=\int_0^T\int_\Omega\beta_1\Big[\frac{\partial A}{\partial\mathbf{u}}\cdot\mathbf{w}-\Big(\nabla\cdot\frac{\partial A}{\partial\nabla\mathbf{u}}\Big)\cdot\mathbf{w}\Big]d\Omega dt+\\&+\int_0^T\int_{\Sigma_N}\beta_1\frac{\partial A}{\partial\nabla\mathbf{u}}\mathbf{n}\cdot\mathbf{w}d\sigma dt+\int_0^T\int_{\Sigma_N}\beta_2\frac{\partial B}{\partial\mathbf{u}}\cdot\mathbf{w}d\sigma dt\end{aligned}$$

Let's now compute the derivative w.r.t. p:

$$\begin{aligned}\big\langle J_p(\mathbf{u},p;\gamma),r\big\rangle_{\mathscr{P}_{\Omega^*},\mathscr{P}_\Omega}&=\lim_{l\to0}\frac{J(\mathbf{u},p+lr;\gamma)-J(\mathbf{u},p;\gamma)}{l}=\\&=\lim_{l\to0}\frac{\int_0^T\int_\Omega\beta_1A(\mathbf{u},\nabla\mathbf{u},p+lr;\gamma)d\Omega dt-\int_0^T\int_\Omega\beta_1A(\mathbf{u},\nabla\mathbf{u},p;\gamma)d\Omega dt}{l}+\\&+\lim_{l\to0}\frac{\int_0^T\int_{\partial\Omega}\beta_2B(\mathbf{u},p+lr;\gamma)d\sigma dt-\int_0^T\int_{\partial\Omega}\beta_2B(\mathbf{u},p;\gamma)d\sigma dt}{l},\end{aligned}$$

since the integrals are independent of the limit variable we can interchange them and get

$$\begin{aligned}\big\langle J_p(\mathbf{u},p;\gamma),r\big\rangle_{\mathscr{P}_{\Omega^*},\mathscr{P}_\Omega}&=\lim_{l\to0}\frac{J(\mathbf{u},p+lr;\gamma)-J(\mathbf{u},p;\gamma)}{l}=\\&=\int_0^T\int_\Omega\beta_1\lim_{l\to0}\frac{A(\mathbf{u},\nabla\mathbf{u},p+lr;\gamma)-A(\mathbf{u},\nabla\mathbf{u},p;\gamma)}{l}d\Omega dt+\\&+\int_0^T\int_{\partial\Omega}\beta_2\lim_{l\to0}\frac{B(\mathbf{u},p+lr;\gamma)-B(\mathbf{u},p;\gamma)}{l}d\sigma dt.\end{aligned}$$

That yields to:

$$\big\langle J_p(\mathbf{u},p;\gamma),r\big\rangle_{\mathscr{P}_{\Omega^*},\mathscr{P}_\Omega}=\int_0^T\int_\Omega\beta_1\frac{\partial A}{\partial p}rd\Omega dt+\int_0^T\int_{\partial\Omega}\beta_2\frac{\partial B}{\partial p}rd\sigma dt.$$

But since, thanks to Neumann B.C., p can be expressed by $\nabla\mathbf{u}$ in Σ_N, we also have $B(\mathbf{u},p;\gamma)=B(\mathbf{u},\nabla\mathbf{u};\gamma)$ in Σ_N, that means B is independent on p in Σ_N. Therefore we finally get:

$$\big\langle J_p(\mathbf{u},p;\gamma),r\big\rangle_{\mathscr{P}_{\Omega^*},\mathscr{P}_\Omega}=\int_0^T\int_\Omega\beta_1\frac{\partial A}{\partial p}rd\Omega dt+\int_0^T\int_{\Sigma_D}\beta_2\frac{\partial B}{\partial p}rd\sigma dt.$$

We can now formulate the adjoint PDE of the Navier-Stokes equations for out optimization problem imposing (7.25) in the weak form, that becomes:

$$\begin{aligned}\big\langle e^*_{\mathbf{u}}(\mathbf{u}_a,p_a;\gamma),\mathbf{w}\big\rangle_{\mathscr{U}_\Omega^{d*},\mathscr{U}_\Omega^d}&=-\big\langle J_{\mathbf{u}}(\mathbf{u},p;\gamma),\mathbf{w}\big\rangle_{\mathscr{U}_\Omega^{d*},\mathscr{U}_\Omega^d}\\\big\langle e^*_p(\mathbf{u}_a,p_a;\gamma),r\big\rangle_{\mathscr{P}_{\Omega^*},\mathscr{P}_\Omega}&=-\big\langle J_p(\mathbf{u},p;\gamma),r\big\rangle_{\mathscr{P}_{\Omega^*},\mathscr{P}_\Omega}.\end{aligned}\tag{7.33}$$

Now equalizing the terms under the same integral signs we obtain the adjoint PDE system:

$$\begin{cases} -\rho\dfrac{\partial \mathbf{u}_a}{\partial t} - \eta\Delta\mathbf{u}_a - \rho(\mathbf{u}\cdot\nabla)\mathbf{u}_a + \rho(\nabla\mathbf{u})\mathbf{u}_a + \nabla p_a = -\beta_1\Big(\dfrac{\partial A}{\partial \mathbf{u}} - \nabla\cdot\dfrac{\partial A}{\partial \nabla\mathbf{u}}\Big) + \dfrac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{u}_a, \text{ in } [0,T]\times\Omega \\ -\nabla\cdot\mathbf{u}_a = -\beta_1\dfrac{\partial A}{\partial p} + \dfrac{\partial \mathbf{f}}{\partial p}\cdot\mathbf{u}_a, \text{ in } [0,T]\times\Omega \\ \mathbf{u}_a(T,\mathbf{x}) = 0, \text{ in } \Omega \\ \mathbf{u}_a = -\dfrac{\partial B}{\partial p}\mathbf{n}, \text{ on } \Sigma_D \\ \big[-p_a\mathbf{I} + \eta\nabla\mathbf{u}_a\big]\mathbf{n} = -\rho(\mathbf{u}\cdot\mathbf{n})\mathbf{u}_a - \beta_1\dfrac{\partial A}{\partial\nabla\mathbf{u}}\mathbf{n} - \beta_2\dfrac{\partial B}{\partial \mathbf{u}}, \text{ on } \Sigma_N. \end{cases}$$

Since the implementation in performed using the kinematic viscosity, dividing by ρ finally yields to:

$$\begin{cases} -\dfrac{\partial \mathbf{u}_a}{\partial t} - \nu\Delta\mathbf{u}_a - (\mathbf{u}\cdot\nabla)\mathbf{u}_a + (\nabla\mathbf{u})\mathbf{u}_a + \dfrac{1}{\rho}\nabla p_a = -\beta_1\dfrac{1}{\rho}\Big(\dfrac{\partial A}{\partial \mathbf{u}} - \nabla\cdot\dfrac{\partial A}{\partial \nabla\mathbf{u}}\Big) + \dfrac{1}{\rho}\dfrac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{u}_a, \text{ in } [0,T]\times\Omega \\ -\nabla\cdot\mathbf{u}_a = -\beta_1\dfrac{\partial A}{\partial p} + \dfrac{\partial \mathbf{f}}{\partial p}\cdot\mathbf{u}_a, \text{ in } [0,T]\times\Omega \\ \mathbf{u}_a(T,\mathbf{x}) = 0, \text{ in } \Omega \\ \mathbf{u}_a = -\dfrac{\partial B}{\partial p}\mathbf{n}, \text{ on } \Sigma_D \\ \big[-\dfrac{1}{\rho}p_a\mathbf{I} + \nu\nabla\mathbf{u}_a\big]\mathbf{n} = -(\mathbf{u}\cdot\mathbf{n})\mathbf{u}_a - \beta_1\dfrac{1}{\rho}\dfrac{\partial A}{\partial\nabla\mathbf{u}}\mathbf{n} - \beta_2\dfrac{1}{\rho}\dfrac{\partial B}{\partial \mathbf{u}}, \text{ on } \Sigma_N. \end{cases} \tag{7.34}$$

We'll now proceed formulating the adjoint system for the stationary Navier-Stokes and Stokes PDEs.

Stationary Navier-Stokes adjoint system

Since in the stationary case both velocity $\mathbf{u}$ and pressure p are cosidered time-independent the integration over time doesn't influence the computation as it just produce a T factor in both sides of $e(\cdot)$ as it is defined in (7.32). Hence, neglecting the derivatives over time we get:

$$\begin{cases} -\nu\Delta\mathbf{u}_a - (\mathbf{u}\cdot\nabla)\mathbf{u}_a + (\nabla\mathbf{u})\mathbf{u}_a + \dfrac{1}{\rho}\nabla p_a = -\beta_1\dfrac{1}{\rho}\Big(\dfrac{\partial A}{\partial \mathbf{u}} - \nabla\cdot\dfrac{\partial A}{\partial \nabla\mathbf{u}}\Big) + \dfrac{1}{\rho}\dfrac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{u}_a, \text{ in } \Omega \\ -\nabla\cdot\mathbf{u}_a = -\beta_1\dfrac{\partial A}{\partial p} + \dfrac{\partial \mathbf{f}}{\partial p}\cdot\mathbf{u}_a, \text{ in } \Omega \\ \mathbf{u}_a = -\dfrac{\partial B}{\partial p}\mathbf{n}, \text{ on } \Sigma_D \\ \big[-\dfrac{1}{\rho}p_a\mathbf{I} + \nu\nabla\mathbf{u}_a\big]\mathbf{n} = -(\mathbf{u}\cdot\mathbf{n})\mathbf{u}_a - \beta_1\dfrac{1}{\rho}\dfrac{\partial A}{\partial\nabla\mathbf{u}}\mathbf{n} - \beta_2\dfrac{1}{\rho}\dfrac{\partial B}{\partial \mathbf{u}}, \text{ on } \Sigma_N. \end{cases} \tag{7.35}$$

Stationary Stokes adjoint system

Let's now consider a Stokes PDE constrained control problem. So the state equation we're considering is:

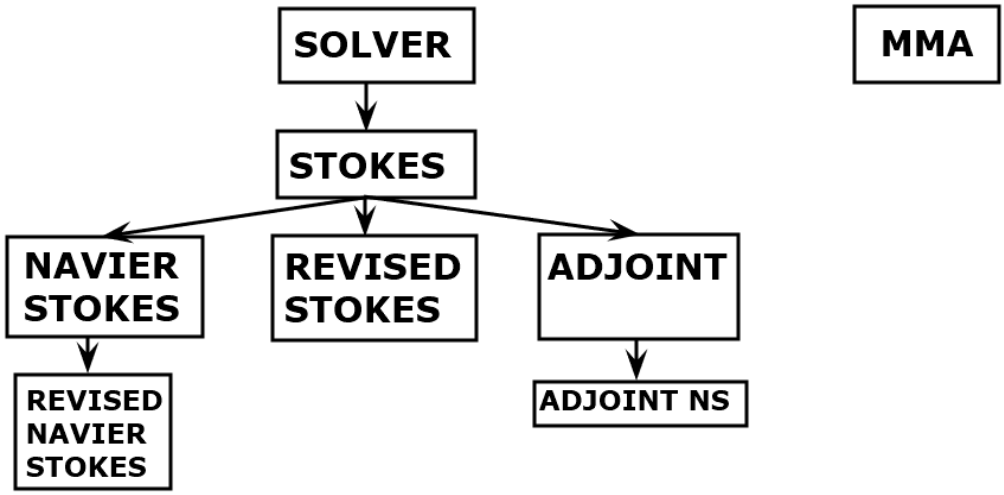
$$\begin{cases} -\nu\Delta\mathbf{u} + \dfrac{1}{\rho}\nabla p = \dfrac{1}{\rho}\mathbf{f} \\ \nabla\cdot\mathbf{u} = 0 \end{cases} \tag{7.36}$$

Therefore, following the previous general-case computation we get the adjoint system:

$$\begin{cases} -\nu\Delta\mathbf{u}_a + \dfrac{1}{\rho}\nabla p_a = -\beta_1\dfrac{1}{\rho}\Big(\dfrac{\partial A}{\partial \mathbf{u}} - \nabla\cdot\dfrac{\partial A}{\partial\nabla\mathbf{u}}\Big) + \dfrac{1}{\rho}\dfrac{\partial \mathbf{f}}{\partial\mathbf{u}}\mathbf{u}_a, \text{ in } \Omega \\ -\nabla\cdot\mathbf{u}_a = -\beta_1\dfrac{\partial A}{\partial p} + \dfrac{\partial \mathbf{f}}{\partial p}\cdot\mathbf{u}_a, \text{ in } \Omega \\ \mathbf{u}_a = -\dfrac{\partial B}{\partial p}\mathbf{n}, \text{ on } \Sigma_D \\ \big[-\dfrac{1}{\rho}p_a\mathbf{I} + \nu\nabla\mathbf{u}_a\big]\mathbf{n} = -\beta_1\dfrac{1}{\rho}\dfrac{\partial A}{\partial\nabla\mathbf{u}}\mathbf{n} - \beta_2\dfrac{1}{\rho}\dfrac{\partial B}{\partial\mathbf{u}}, \text{ on } \Sigma_N. \end{cases}$$

8 Implementation notes

Here some comments on the codes attached to this paper.
The whole program has been implemented with an object oriented programming approach, following the class hierarchy showed in figure (4). In brief,



Class hierarchy of the code.

- **SOLVER:** Has the main methods valid for all possible solvers, that is even for approaches different from the P1-iso-P2 chosen in our case. It basically contains all the functions for the setting of boundary conditions, plot the result and evaluate the errors of the solution with respect to a known function.
- **STOKES:** Has the main methods for the P1-iso-P2 approach to solve the Stokes equations. The main functions load the mesh saved in the specified directory, build the refined mesh for the velocity space, compute the linear system matrix and define the possible solvers (static, time-dependent,...).
- **REVISED STOKES:** Has the modification from the Stokes class to include a forcing term velocity-dependent, as in our case ($\alpha\mathbf{u}$).
- **NAVIER STOKES:** Modifies some of the Stokes methods in order to compute the Navier Stokes linear system. Here the solvers methods are slightly changed to include the Picard iteration scheme on the convective term.

- **REVISED NAVIER STOKES:** Has the modification from the Navier Stokes class to include a forcing term velocity-dependent, as in our case ($\alpha\mathbf{u}$).
- **ADJOINT:** Has the main method to define and solve the linear system associated to the adjoint formulation of the Stokes problem with its appropriate boundary conditions and forcing term.
- **ADJOINT NS:** Modifies the Adjoint class and defines the methods to build the adjoint system of the Navier Stokes problem.
- **MMA:** Class that defines the functional, constraints and relative derivatives computations, and the MMA algorithm.

8.1 Main file

We will now briefly analyze the main script, that is fully reported in the Appendix (12).

```
1  %%% Main script for topology optimization of Stokes problem in a channel
2  clear; close all; clc;
3
4  % Define the 2D mesh and equation BC's and coefficients
5  meshdir='mesh_COMSOL\small_holed_channel\small_holed_channel.txt';
6  nfig   = 1;
7
8  % flag_functional = 1 for min (\alpha ||u||^2)
9  %                   2 for min (\alpha ||u||^2 + 1/2* \mu ||grad(u)||^2)
10 %--------------------
11 flag_functional = 2;
12 inlet_id        = [3,4];
13 outlet_id       = [7,8];
14 %--------------------------------------------------------------------------
15 % SET PARAMETERS
16 %--------------------------------------------------------------------------
17 UseNStokes = true;
18
19 flag      = 0; % flag=1 to see the mesh
20 flag_BC   = 0; % BC: 0 for FAST penalty method
21                %   : 1 for ACCURATE lifting function method
22
23 seeEdgeID = true;
24
25 V_r       = 1;  % max portion of the total domain that can be filled by water (max ...
       volume = V_r*V_0)
26 V_0       = 1;  % total volume !!!!!!!!!!!! TO BE SET WITH GEOMETRY !!!!!!!!
27 alpha_min = 0;
28 alpha_max = 1e4;
29 q         = 10;
30 mu        = 1;
31 rho       = 1;
32 f_1       = @(x,y) 0;
33 f_2       = @(x,y) 0;
34 g         = @(x,y) 0;
35 tau       = 0.00001;  % factor of the SUPG stabilization, to keep as low as possible
36
37 V_in = 0.5;     % inlet velocity
```

This is the incipit of the code, where the user has to set all the main parameters. We recall that in particular, α_{max}, and q have to be numerically calibrated for each different problem. Then (not reported here) an object for RevisedStokes/RevisedNavierStokes, for the Adjoint/Adjoint_NS and MMA are instantiated before the optimization loop, with the chosen physical parameters.
We remark that the user should firstly create a mesh, maybe with a commercial software like COMSOL

[3], and then correct the mesh address in the meshdir variable. Then the code solves with a static solver (StatSolver) the Stokes/Navier Stokes problem and plot the result in term of streamlines.

```
%--------------------------------------------------------------------------
loop = 0;
toll = 1e-3;
change = toll + 1;

%%  LAUNCH SOLVER
while (change > toll)
%--------------------------------------------------------------------------
% UPDATE PARAMETER GAMMA
%--------------------------------------------------------------------------
alpha = alpha_min + (alpha_max - alpha_min).*q.*(1-gamma)./(q+gamma);
NStokes.alpha = alpha;
%--------------------------------------------------------------------------
% SOLVE
%--------------------------------------------------------------------------
tic;
disp('-----------------------------------')
disp('SOLVE MASTER PROBLEM');
disp('-----------------------------------');
disp('-----------------------------------');
[uh1, uh2, ph] = NStokes.StatSolver(flag_BC);
toc;

%---------------------------------------
% UPDATE STREAMLINE AND DESIGN PLOTS
%---------------------------------------
figure(3)
for iedge = 1:size(NStokes.Bound.Edges,1)
    p1 = NStokes.V.coord(NStokes.Bound.Edges(iedge,1),:);
    p2 = NStokes.V.coord(NStokes.Bound.Edges(iedge,2),:);
    plot([p1(1), p2(1)], [p1(2), p2(2)], 'k-');
    hold on;
end

[x,y] = meshgrid(NStokes.X_limits(1):0.1:NStokes.X_limits(2), ...
                 NStokes.Y_limits(1):0.1:NStokes.Y_limits(2));
u = zeros(size(x)); v = zeros(size(x));
for i = 1:numel(x)
    [u(i), v(i)]= NStokes.getApproxValue_v([x(i);y(i)],uh1,uh2);
    if norm([u(i), v(i)]) < 1e-2
        u(i) = 0;
        v(i) = 0;
    end
end
quiver(x,y,u,v, 'Color', 'b');
streamslice(x,y,u,v);
hold off;
```

Once solved the master problem for the governing equations, we update the solution paramters in the Adjoint object, and redifine it's BCs (which may depend on the solution), and finally solve even the adjoint system.

```
%--------------------------------------------------------------------------
% BUILD ADJOINT SYSTEM
%--------------------------------------------------------------------------
tic;
disp('-----------------------------------')
disp('Build Adjoint system');
disp('-----------------------------------');
disp('-----------------------------------');
Adjoint = Adjoint.Update(uh1,uh2,ph,alpha);

switch flag_functional
    case 1
```

```
        Adjoint.f_1    = -2.*alpha.*uh1/rho;
        Adjoint.f_2    = -2.*alpha.*uh2/rho;
        Adjoint.g      = zeros(Adjoint.P.Nodes);
    case 2
        Adjoint.f_1    = -2*alpha.*uh1/rho;
        Adjoint.f_2    = -2*alpha.*uh2/rho;
        Adjoint.g      = zeros(Adjoint.P.Nodes);
end

Adjoint = Adjoint.clearBC();
toc;

%--------------------------------------------------------------------------
% SET ADJOINT BC
%--------------------------------------------------------------------------
tic;
disp('-----------------------------------')
disp('SET ADJOINT BC');
disp('-----------------------------------');
disp('-----------------------------------');

for i = 1:NStokes.Bound.NBound
    Adjoint = Adjoint.setEdgeDirBC(i,@(x,y)[0,0]);
end
%--------------------------------
for i = 1:length(outlet_id)
    Adjoint = Adjoint.setEdgeNeuBC(outlet_id(i));
end
toc;

toc;
%--------------------------------------------------------------------------
% SOLVE
%--------------------------------------------------------------------------
tic;
disp('-----------------------------------')
disp('ADJOINT SOLVER');
disp('-----------------------------------');
disp('-----------------------------------')
Adjoint = Adjoint.Prepro;
[ua1, ua2, pa] = Adjoint.DirectStatSolver(flag_BC);
toc;
```

Finally we update the γ parameters solving the MMA algorithm with the new values and plot the design grph.

```
%--------------------------------------------------------------------------
% CALL MMA METHOD
%--------------------------------------------------------------------------
tic;
disp('-----------------------------------')
disp('UPDATE AND SOLVE MMA');
disp('-----------------------------------');
disp('-----------------------------------')
Optimizer = Optimizer.Update(uh1, uh2, ua1, ua2);
[gamma_new, f0val, Vol] = Optimizer.solve(gamma,loop);
toc;

% PRINT RESULTS
change = Solver.evalErr(gamma, gamma_new, NStokes.V);
loop  = loop + 1;
disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',f0val) ...
      ' Vol.: ' sprintf('%6.3f',Vol) ...
      ' ch.: ' sprintf('%6.3f',change )]);

% % PLOT DENSITIES
figure(4)
x=NStokes.V.coord(:,1); y=NStokes.V.coord(:,2);
```

```
23  M=delaunay(x,y);
24  cd triplot
25  tricontf(x,y,M,gamma_new,10);
26  colormap(gray)
27  gamma = gamma_new;
28  axis(limits);
29  cd ..
30
31  end
```

We have also prepared a version of the code with a User interface, which gives the possibility to the user to dynamically modify all the interesting parameters without the need of a manual modification of the source code (5).

```
Press ENTER to launch the last project, or insert 1 to enter the MODEL BUILDER: 1
-----------------------------------
MODEL BUILDER
-----------------------------------
||Legend||
(1) CHANGE PROJECT
(2) SET BOUNDARY EDGES
(3) CHANGE MODEL PARAMETERS
(4) CHANGE TOPOLOGY OPTIMIZATION PARAMETERS
(5) CHANGE TERMINATION METHOD
Selection: 4
Insert the new "flag_functional": 0
Insert the new "alpha_min": 0
Insert the new "alpha_max": 1e4
Insert the new "q": 10
```

Example of the user interface in the revised code

9 Model and Solver validation. Comsol comparisons

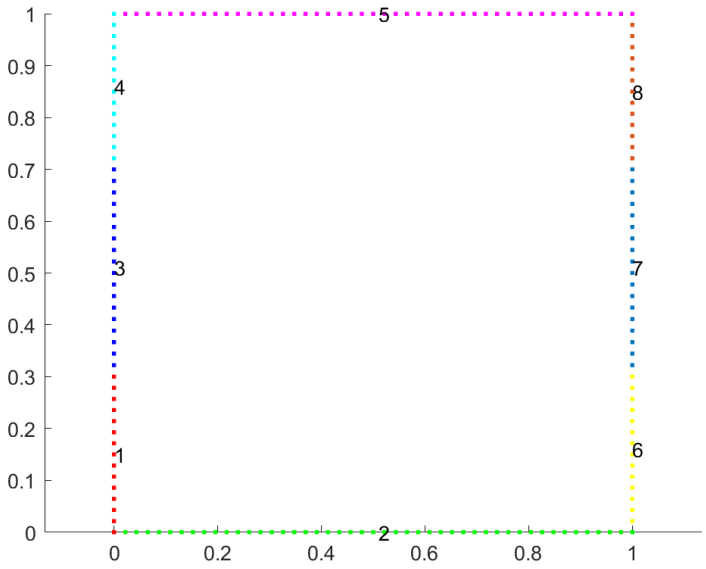
In this section we'll discuss the results obtained using the above described model and solver, comparing their NS solutions to the ones provided by the commercial software [3]. We note here the the code works without any specified dimension, given that they are coherent, for this reason in all the following results we won't specify any unit. The comparison will be performed over two different projects using different meshes for the NS equations.
The errors are computed using the following formula:

$$error^{abs} = ||sol_{COMSOL} - sol_{MATLAB}||_{L^2}, \tag{9.1}$$

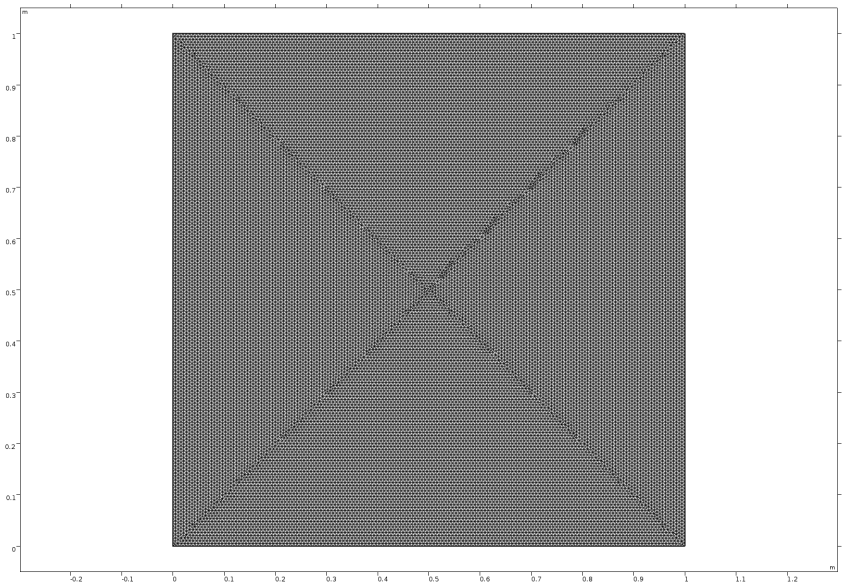
$$error^{rel} = \frac{||sol_{COMSOL} - sol_{MATLAB}||_{L^2}}{||sol_{COMSOL}||_{L^2}}. \tag{9.2}$$

Channel in a Square with $Re \approx 0.5$

Let's consider a $(1m \times 1m)$ square where, for the sake of the description's simplicity, the boundaries are denoted as follows:



Channel in a Square boundary edges



"Extremely fine" mesh for channel in a square problem

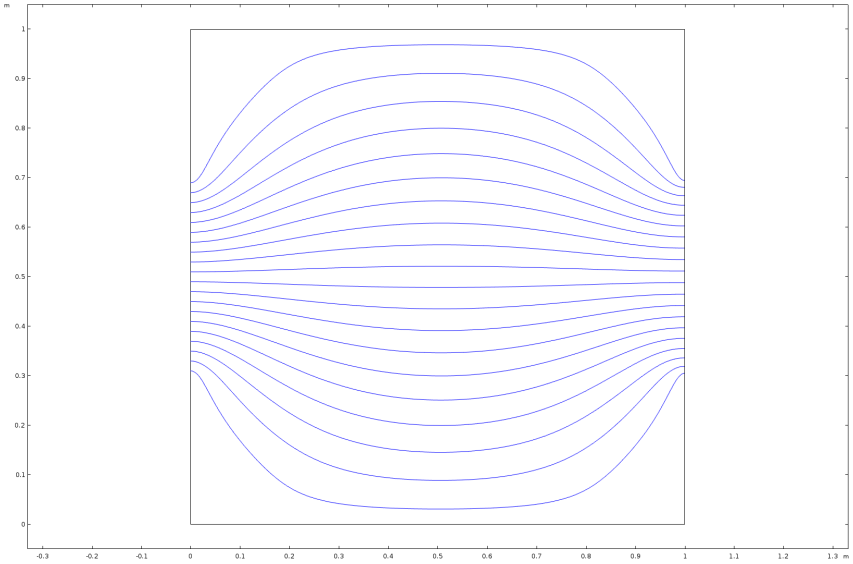
The characteristic length (*i.e. max element size*) for this mesh is 0.0067m.
Now let's consider the boundary edge (3) as the inlet, with Dirichlet boundary condition:

$$u_{in} = \big(12.5(y - 0.3)(0.7 - y),\ 0\big),$$

and the boundary edge (7) as the outlet, with Open Neumann boundary condition. In all the other edges are imposed the no-slip boundary conditions. Finally for the sake of simplicity we consider custom viscosity and density, $\mu = 1, \rho = 1$, in order to remain in laminar flow regime ($Re \approx 0.5$).

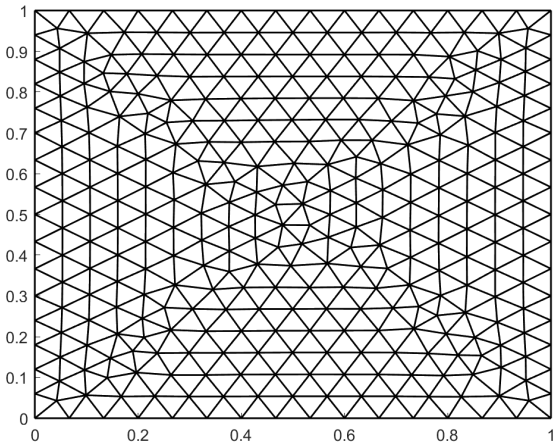
The COMSOL solution, evaluated in an "extremely fine"(*as it's denoted in the software*) triangulation, is taken as a yardstick to evaluate the correctness of the MATLAB solutions evaluated in the "coarse", "normal", "fine" COMSOL meshes.

The COMSOL streamlines for this problem are:



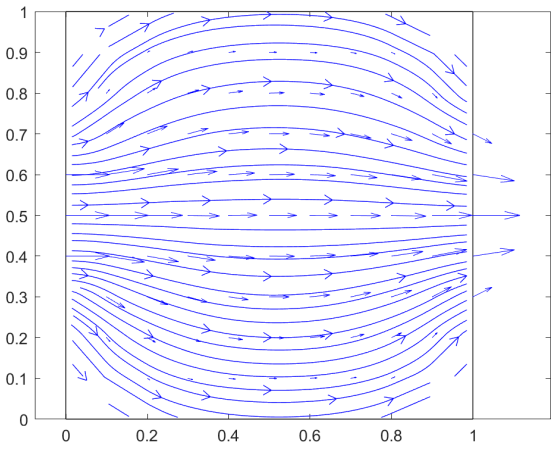
COMSOL streamilines for channel in a square problem

COARSE MESH:



"Coarse" mesh for channel in a square problem

The characteristic length for this mesh is 0.067m.
The MATLAB solution's streamlines are:

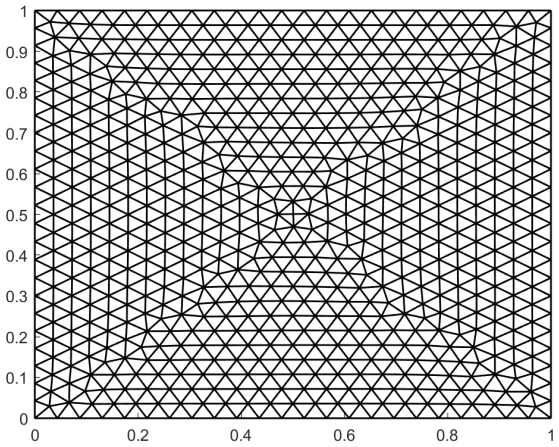


MATLAB streamlines using coarse mesh

Finally the absolute and relative errors as are:

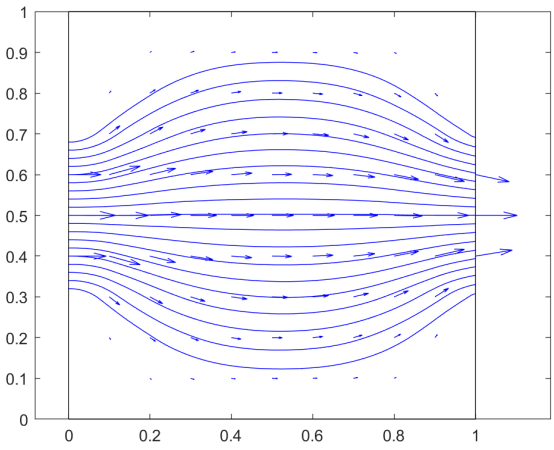
$$e_{u1}^{abs} = 4.633 \times 10^{-3}, \qquad e_{u1}^{rel} = 2.610 \times 10^{-2},$$
$$e_{u2}^{abs} = 7.855 \times 10^{-3}, \qquad e_{u2}^{rel} = 2.220 \times 10^{-1},$$
$$e_{p}^{abs} = 9.412 \times 10^{-1}, \qquad e_{p}^{rel} = 2.169 \times 10^{-1}.$$

NORMAL MESH:



"Normal" mesh for channel in a square problem

The characteristic length for this mesh is 0.045m.
The MATLAB solution's streamlines are:

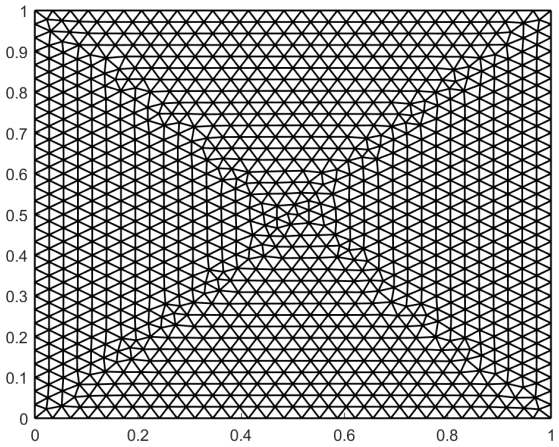


MATLAB streamlines using normal mesh

Finally the absolute and relative errors are:

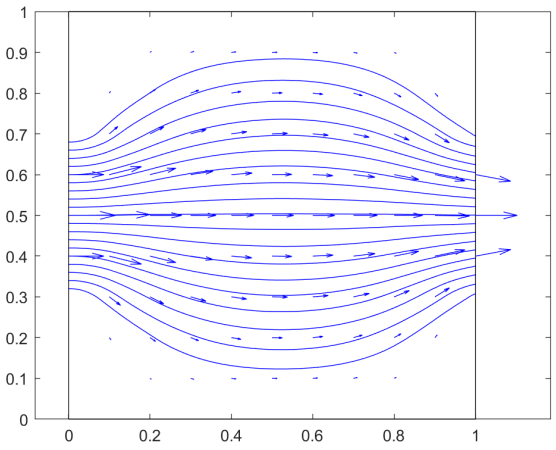
$$e_{u1}^{abs} = 4.676 \times 10^{-3}, \qquad e_{u1}^{rel} = 2.633 \times 10^{-2},$$
$$e_{u2}^{abs} = 7.261 \times 10^{-3}, \qquad e_{u2}^{rel} = 2.047 \times 10^{-1},$$
$$e_{p}^{abs} = 6.963 \times 10^{-1}, \qquad e_{p}^{rel} = 1.637 \times 10^{-1}.$$

FINE MESH:



"Fine" mesh for channel in a square problem

The characteristic length for this mesh is 0.035m.
The MATLAB solution's streamlines are:

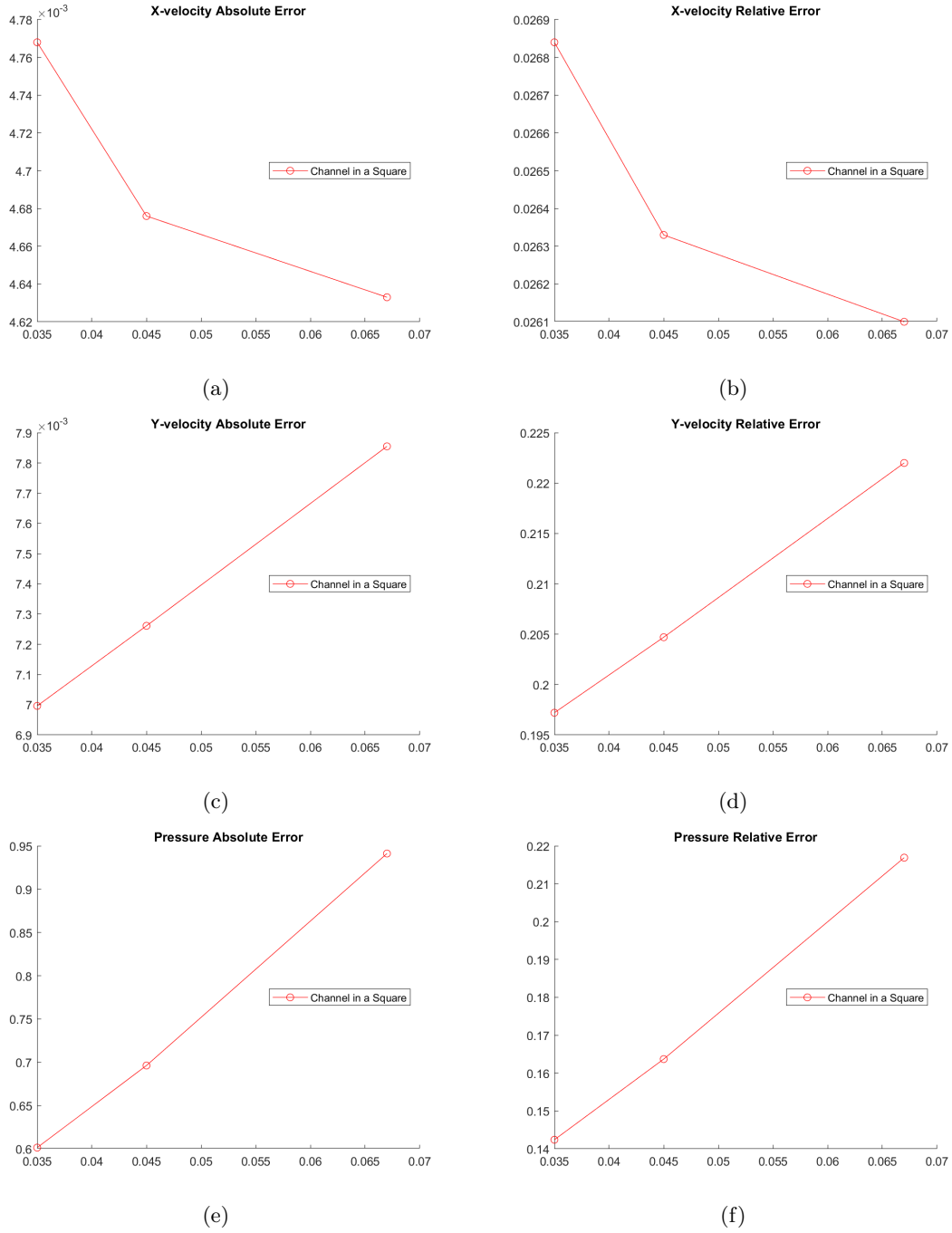


MATLAB streamlines using fine mesh

Finally the absolute and relative errors are:

$$e_{u1}^{abs} = 4.768 \times 10^{-3}, \qquad e_{u1}^{rel} = 2.684 \times 10^{-2},$$
$$e_{u2}^{abs} = 6.996 \times 10^{-3}, \qquad e_{u2}^{rel} = 1.972 \times 10^{-1},$$
$$e_{p}^{abs} = 6.014 \times 10^{-1}, \qquad e_{p}^{rel} = 1.424 \times 10^{-1}.$$

ERROR GRAPHS:



Error graphs for Channel in a Square problem

Observing the errors it emerges a reasonable behaviour of the MATLAB solutions that become more accurately as the mesh parameter decrease. It must be said that, for this problem, decreasing the mesh parameter let the X-velocity error increase, and this shouldn't happen, but the little growth (≈ 0.01 *for every refinement*) of this error is well compensated by a bigger accuracy of the other variables.
An explanation of this behaviour could be the presence of some recirculation near the lower and upper bounds that may not be appropriately described by the MATLAB software due to the presence of stabilization techniques that might make the solution too smoother, canceling the recirculation phenomena.

Plate and Cylinder in a Rectangle with $Re \approx 0.5$

Let's consider a $(5m \times 1m)$ rectangle where, for the sake of the description's simplicity, the boundaries are denoted as follows:

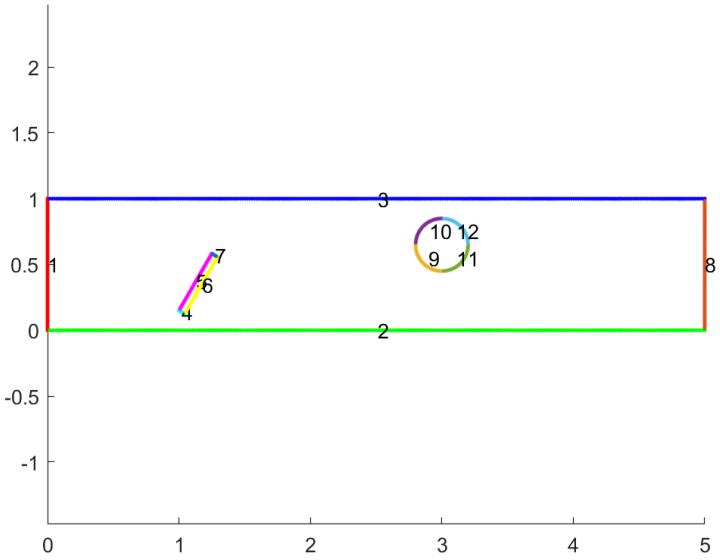
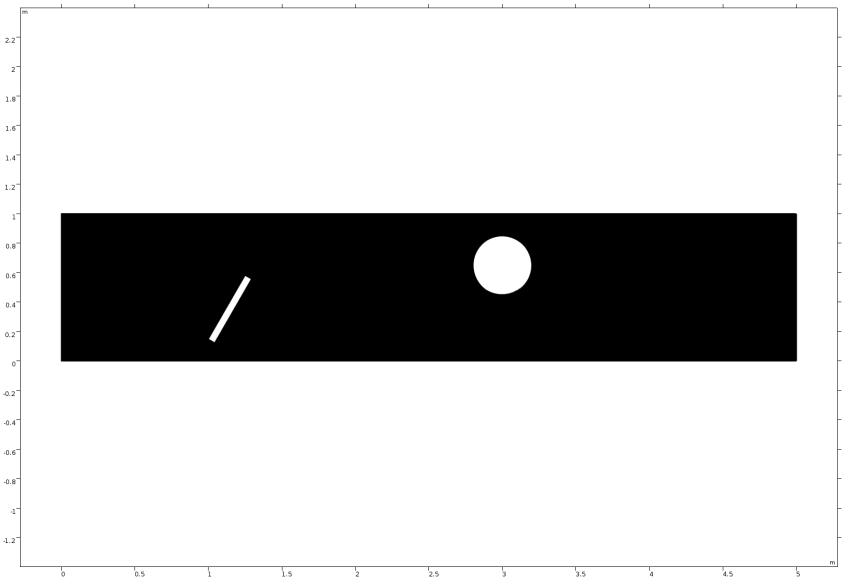


Plate and Cylinder in a Rectangle boundary edges



"Extremely fine" mesh for plate and cylinder in a rectangle problem

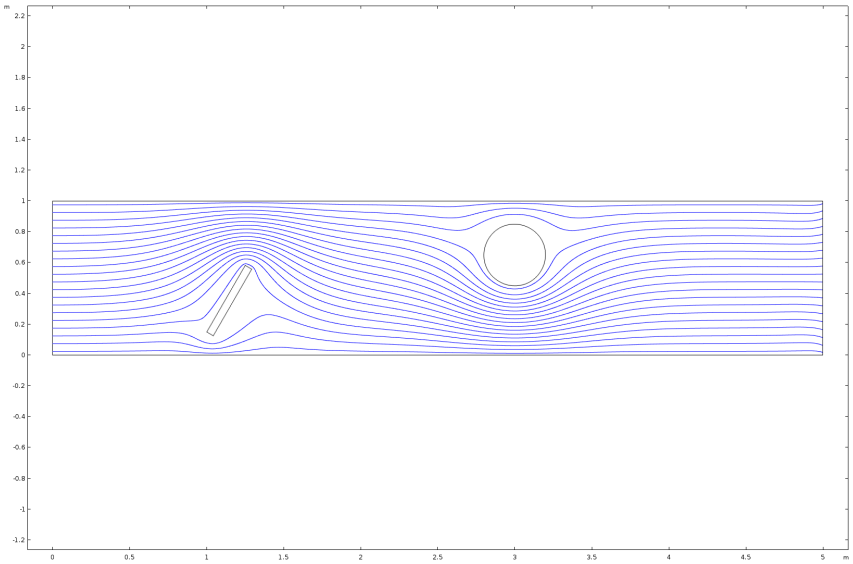
The characteristic length (*i.e. max element size*) for this mesh is 0.0067m.
Now let's consider the boundary edge (1) as the inlet, with Dirichlet boundary condition:

$$u_{in} = \big(2y(1-y),\ 0\big)$$

and the boundary edge (8) as the outlet, with Open Neumann boundary condition. In all the other edges are imposed the no-slip boundary conditions. Finally for the sake of simplicity we consider custom viscosity and density, $\mu = 1, \rho = 1$, in order to remain in laminar flow regime ($Re \approx 0.5$).

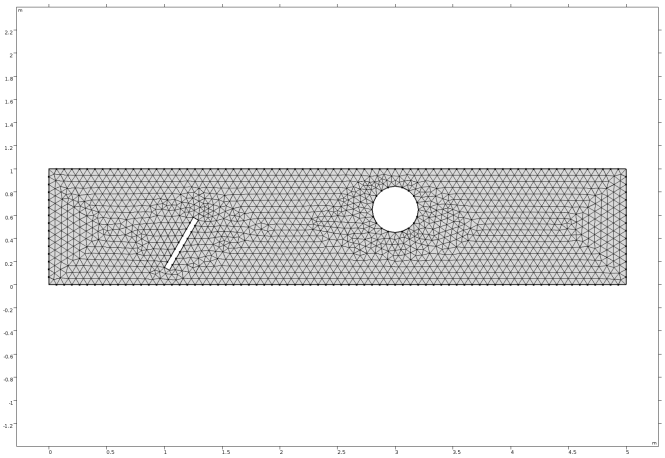
The COMSOL solution, evaluated in an "extremely fine"(*as it's denoted in the software*) triangulation, is taken as a yardstick to evaluate the correctness of the MATLAB solutions evaluated in the "coarse", "normal", "fine" COMSOL meshes.

The COMSOL streamlines for this problem are:



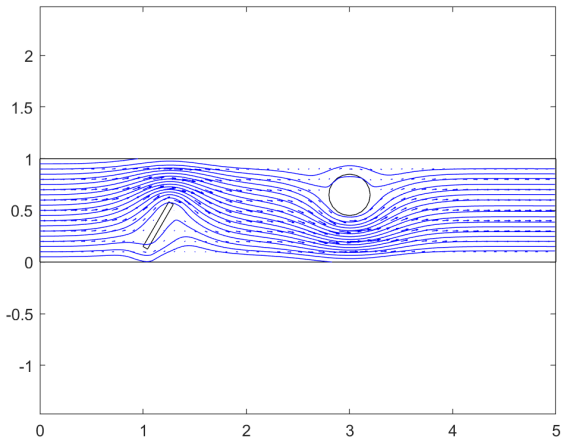
COMSOL streamilines for plate and cylinder in a rectangle problem

COARSE MESH:



"Coarse" mesh for plate and cylinder in a rectangle problem

The characteristic length for this mesh is 0.067m.
The MATLAB solution's streamlines are:

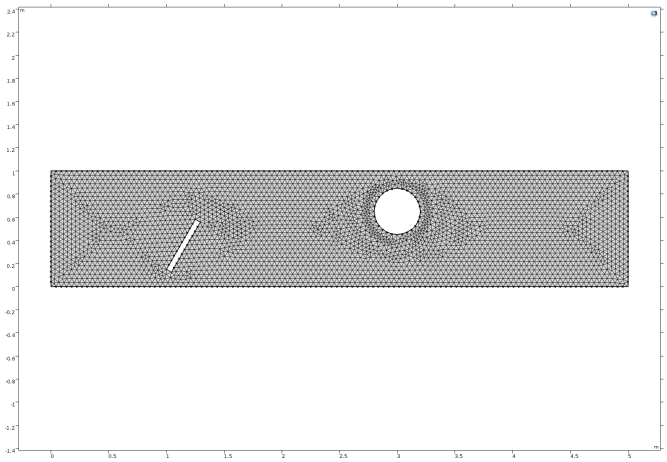


MATLAB streamlines using coarse mesh

Finally the absolute and relative errors are:

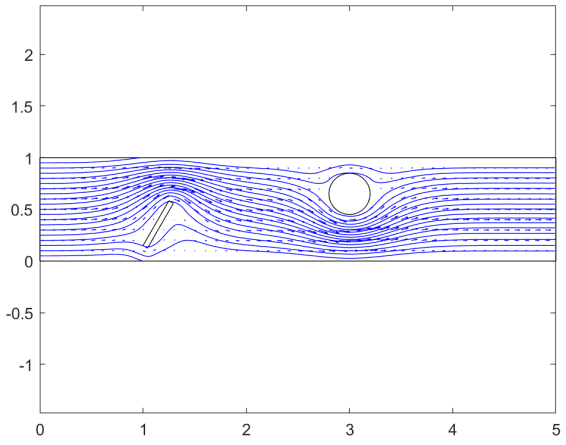
$$e_{u1}^{abs} = 6.140 \times 10^{-3}, \qquad e_{u1}^{rel} = 6.914 \times 10^{-3},$$
$$e_{u2}^{abs} = 1.304 \times 10^{-2}, \qquad e_{u2}^{rel} = 5.623 \times 10^{-2},$$
$$e_{p}^{abs} = 6.470 \times 10^{-1}, \qquad e_{p}^{rel} = 7.976 \times 10^{-3}.$$

NORMAL MESH:



"Normal" mesh for plate and cylinder in a rectangle problem

The characteristic length for this mesh is 0.045m.
The MATLAB solution's streamlines are:

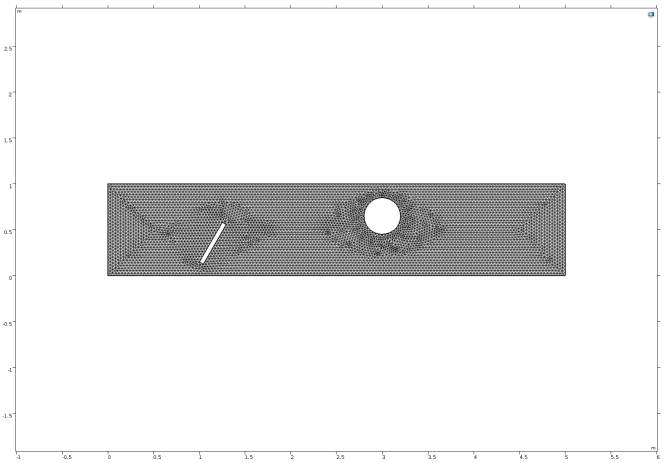


MATLAB streamlines using normal mesh

Finally the absolute and relative errors are:

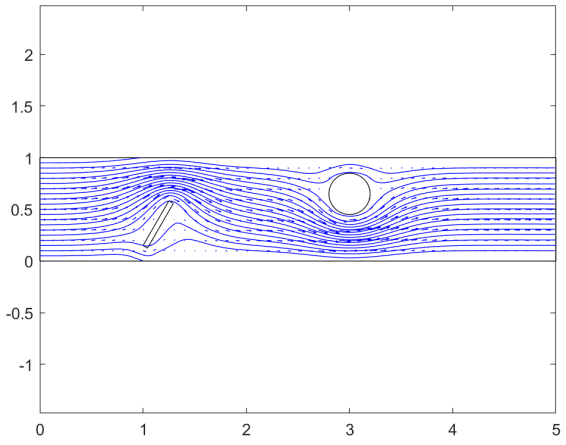
$$e_{u1}^{abs} = 5.651 \times 10^{-3}, \qquad e_{u1}^{rel} = 6.362 \times 10^{-3},$$
$$e_{u2}^{abs} = 1.243 \times 10^{-2}, \qquad e_{u2}^{rel} = 5.361 \times 10^{-2},$$
$$e_{p}^{abs} = 5.747 \times 10^{-1}, \qquad e_{p}^{rel} = 7.086 \times 10^{-3}.$$

FINE MESH:



"Fine" mesh for plate and cylinder in a rectangle problem

The characteristic length for this mesh is 0.035m.
The MATLAB solution's streamlines are:

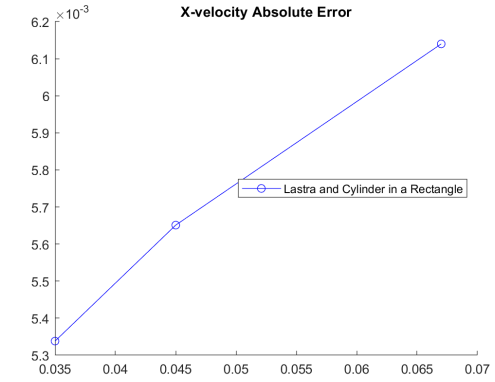


MATLAB streamlines using fine mesh

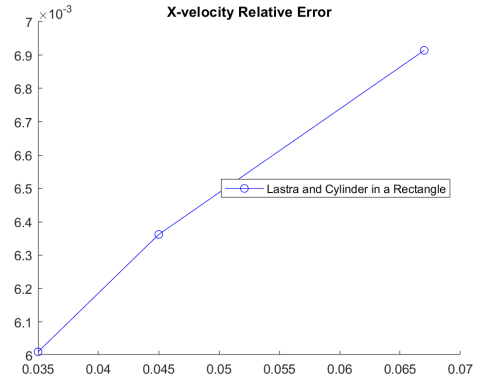
Finally the absolute and relative errors are:

$$e_{u1}^{abs} = 5.338 \times 10^{-3}, \qquad e_{u1}^{rel} = 6.010 \times 10^{-3},$$
$$e_{u2}^{abs} = 1.222 \times 10^{-2}, \qquad e_{u2}^{rel} = 5.272 \times 10^{-2},$$
$$e_{p}^{abs} = 5.744 \times 10^{-1}, \qquad e_{p}^{rel} = 7.082 \times 10^{-3}.$$

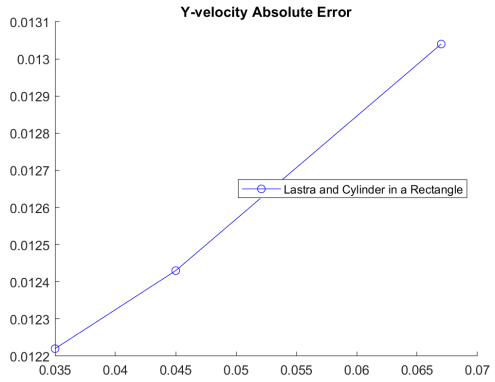
ERROR GRAPHS:



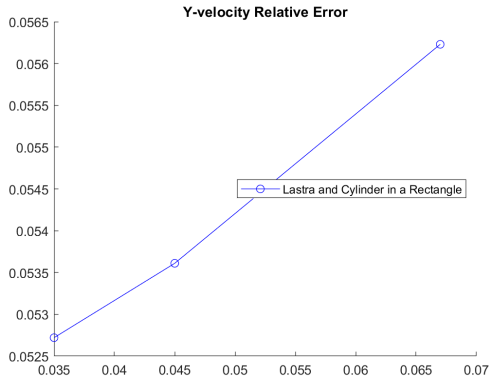
(a)



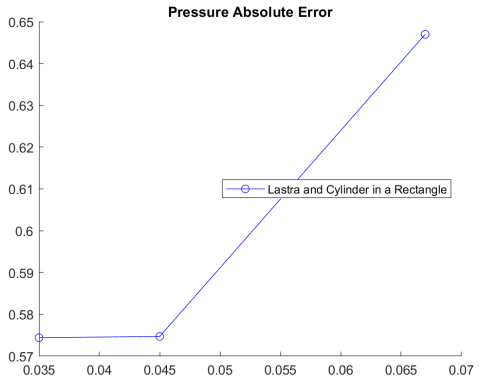
(b)



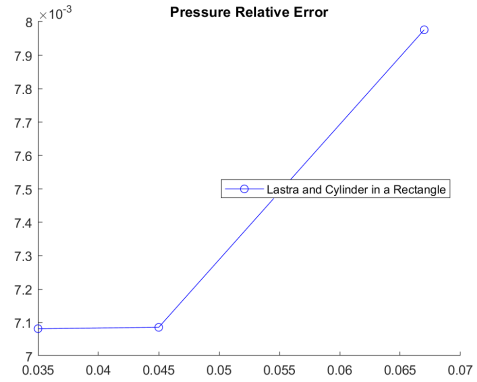
(c)



(d)



(e)

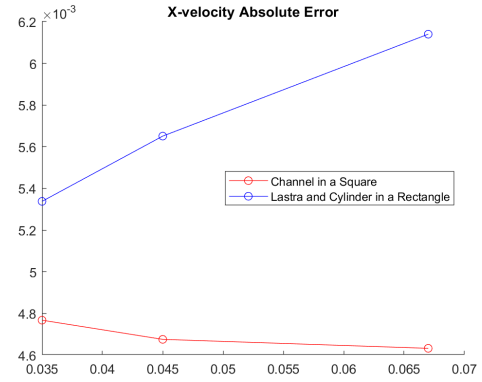


(f)

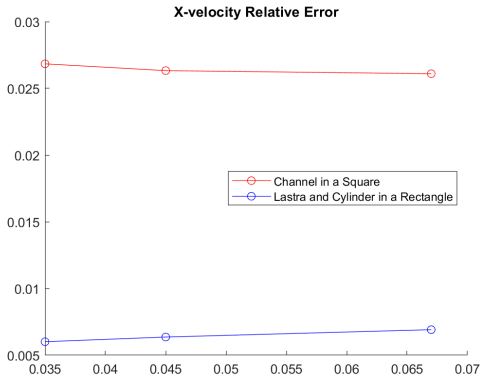
Error graphs for Plate and Cylinder in a Rectangle problem

The behaviour of the MATLAB solution's errors is coherent as it decreases as the mesh is refined.
In particular, considering the "fine" mesh the absolute error (*expressed in* L^2 *norm*) of the horizontal velocity u_1 and in the pressure p are less than the 1% of the L^2 norm of the COMSOL solution. The absolute error in u_2 is bigger then the others but still acceptable as it results to be approximately the 5.3% of the COMSOL solution.

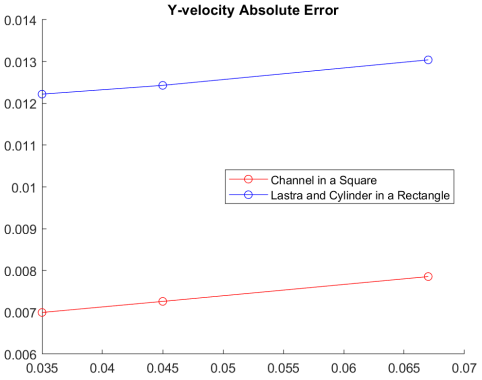
COMPARED ERROR GRAPHS:



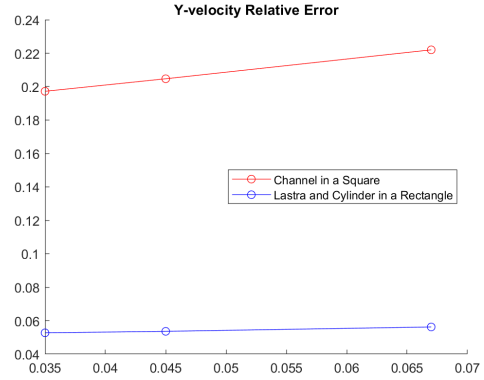
(a)



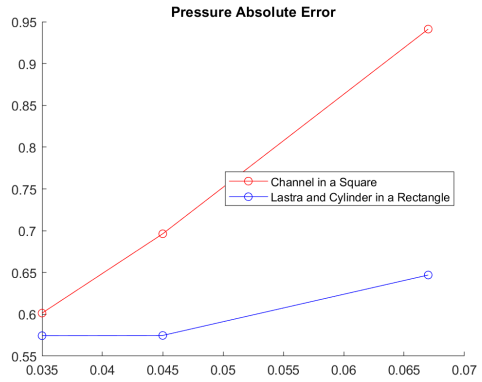
(b)



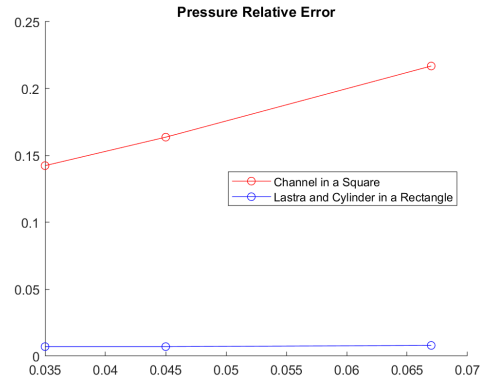
(c)



(d)



(e)



(f)

Compared errors for Channel in a Square and Plate and Cylinder in a Rectangle problems

Observing (26) emerges a general reasonable behaviour of the MATLAB solutions with respect to the COMSOL ones: decreasing the mesh parameter produces a comparable improvement in almost all the solutions.
In virtue of this the model and solver validation in the case of $Re \approx 0.5$ is completed and therefore we'll assume we can perform the optimization problem using the MATLAB solutions computed with the "fine" mesh.

Plate in a Rectangle

The following discussion will focus on the correctness of the solutions when the Reynold's number increase.

Let's consider a $(3m \times 1m)$ rectangle where, for the sake of the description's simplicity, the boundaries are denoted as follows:

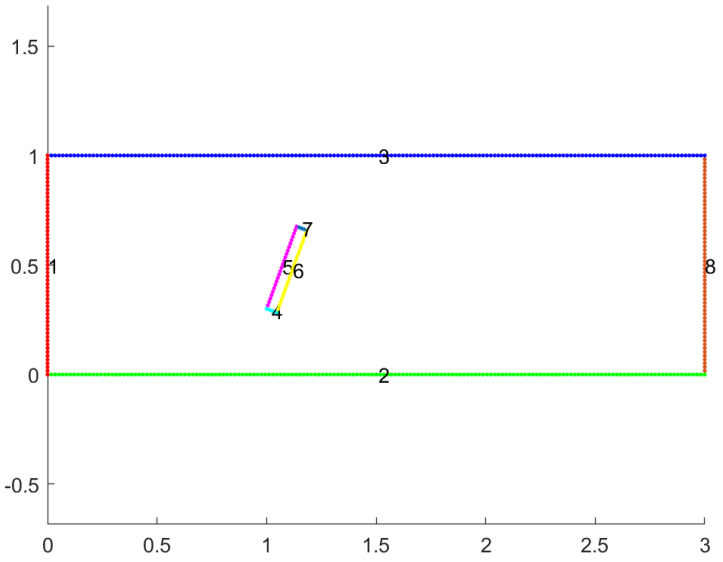


Plate in a Rectangle boundary edges

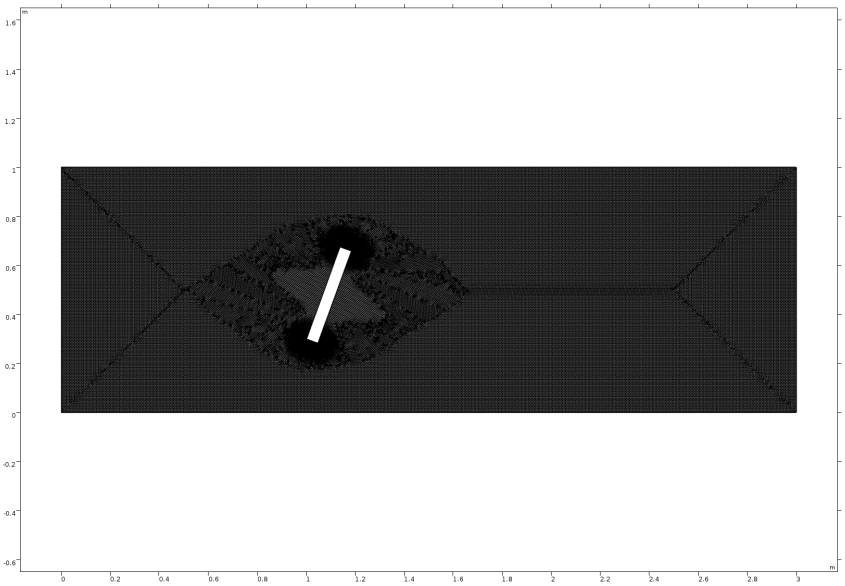
Now let's consider the boundary edge (1) as the inlet, with Dirichlet boundary condition:

$$u_{in} = \big(2y(1-y),\ 0\big)$$

and the boundary edge (8) as the outlet, with Open Neumann boundary condition. In all the other edges are imposed the no-slip boundary conditions.

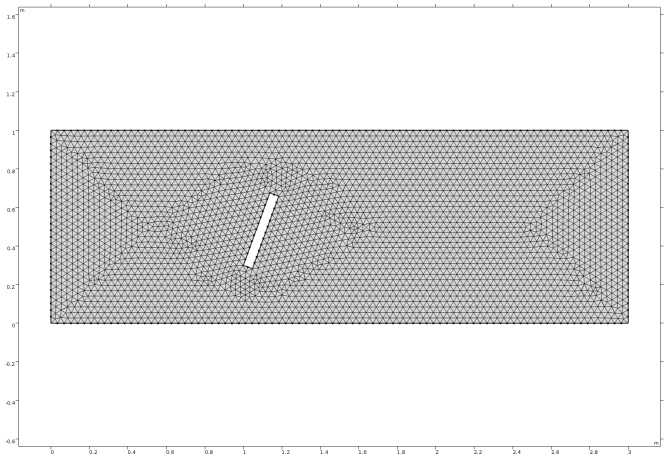
The COMSOL solution, evaluated in an "extremely fine"(*as it's denoted in the software*) triangulation, is taken as a yardstick to evaluate the correctness of the MATLAB solution evaluated in the "fine" COMSOL mesh.

MESHES:



"Extremely fine" mesh for plate in a rectangle problem

The characteristic length for this mesh is 0.0067m.

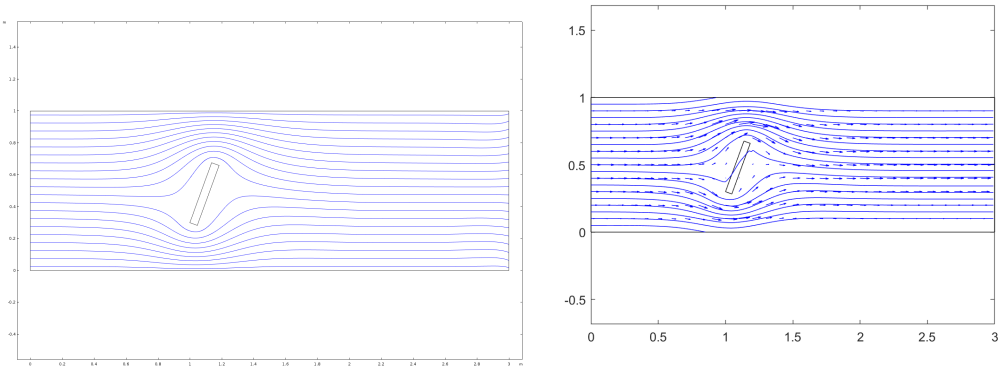


"Fine" mesh for plate in a rectangle problem

The characteristic length for this mesh is 0.035m.

CASE $Re \approx 0.5$:

The COMSOL and MATLAB solutions' streamlines are:



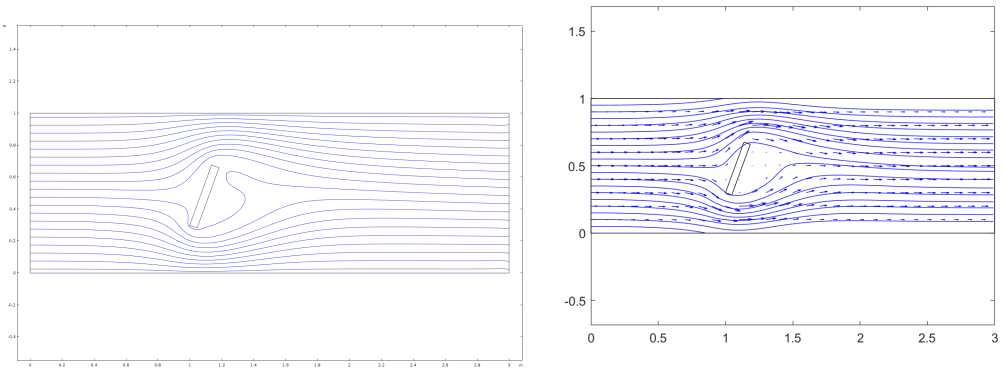
(a) *COMSOL streamlines* (b) *MATLAB streamlines.*

We observe that in the MATLAB case some streamlines seem to pass through the plate, but in reality is just a graph issue, since the MATLAB function just connect two points, one immediately before the plate, and one right after, which have not exactly zero velocity, and plots a continuous line, even if there is no flow in the plate. Finally the absolute and relative errors are:

$$e_{u1}^{abs} = 5.289 \times 10^{-3}, \qquad e_{u1}^{rel} = 8.258 \times 10^{-3},$$
$$e_{u2}^{abs} = 1.224 \times 10^{-2}, \qquad e_{u2}^{rel} = 9.835 \times 10^{-2},$$
$$e_{p}^{abs} = 6.296 \times 10^{-1}, \qquad e_{p}^{rel} = 1.804 \times 10^{-2}.$$

CASE $Re \approx 50$:

The COMSOL and MATLAB solutions' streamlines are:



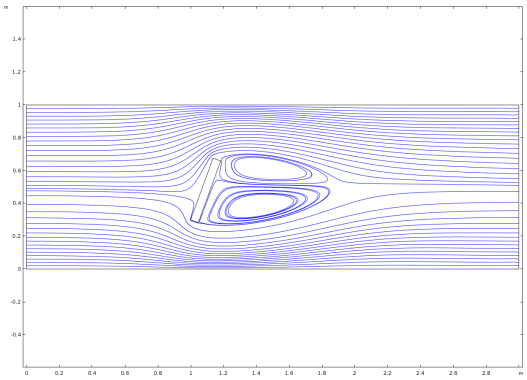
(a) *COMSOL streamlines* (b) *MATLAB streamlines*

Finally the absolute and relative errors are:

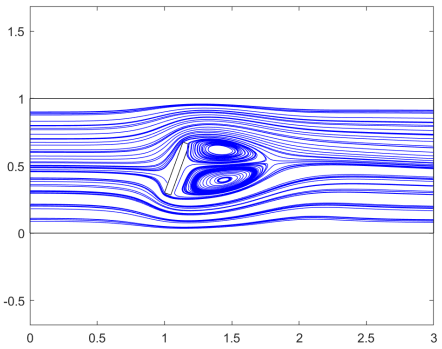
$$e_{u1}^{abs} = 3.236 \times 10^{-2}, \qquad e_{u1}^{rel} = 5.014 \times 10^{-2},$$
$$e_{u2}^{abs} = 1.565 \times 10^{-2}, \qquad e_{u2}^{rel} = 1.341 \times 10^{-1},$$
$$e_{p}^{abs} = 1.113 \qquad , \qquad e_{p}^{rel} = 2.474 \times 10^{-2}.$$

CASE $Re \approx 150$:

The COMSOL and MATLAB solutions' streamlines are:



(a) *COMSOL streamlines*

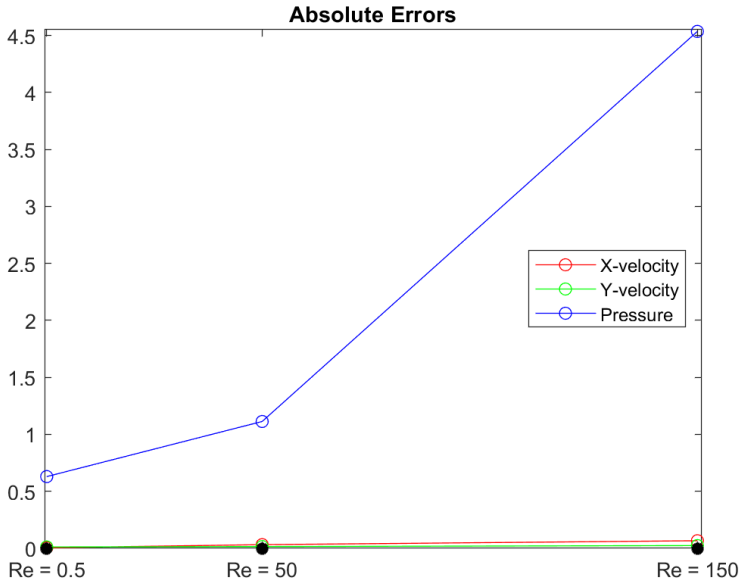


(b) *MATLAB streamlines*

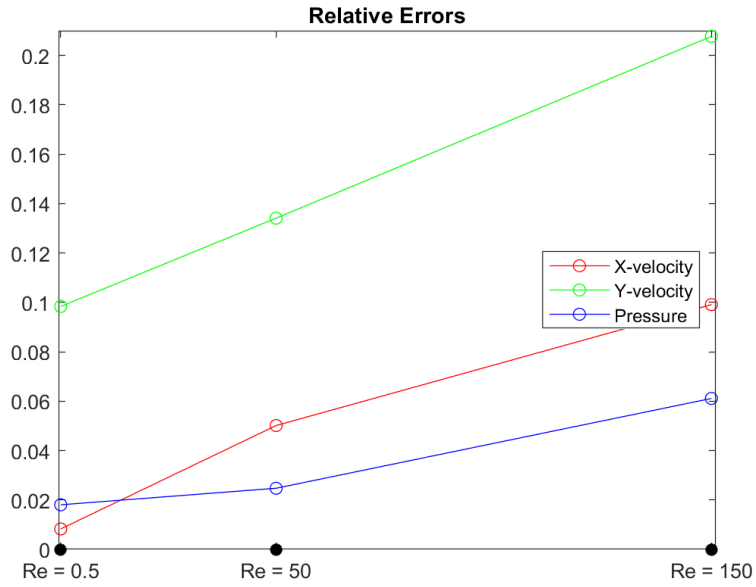
Finally the absolute and relative errors are:

$$e_{u1}^{abs} = 6.686 \times 10^{-2}, \qquad e_{u1}^{rel} = 9.909 \times 10^{-2},$$
$$e_{u2}^{abs} = 2.422 \times 10^{-2}, \qquad e_{u2}^{rel} = 2.077 \times 10^{-1},$$
$$e_{p}^{abs} = 4.535 \qquad , \qquad e_{p}^{rel} = 6.110 \times 10^{-2}.$$

ERROR GRAPHS:



(a)



(b)

Errors for Plate in a Rectangle problem with different Reynolds numbers

Looking at the errors we can observe the MATLAB solution correctly describes the fluid behaviour also for higher Reynolds numbers, obviously as the Reynolds number increase the reliability of the MATLAB solution decreases. In the above example, with $Re \approx 50$, the worst approximation occur in u_2 and it's of the order of 13%, that's acceptable from a qualitative analysis but maybe not suitable for a quantitative one. By the way the scope of this project is to work with low Reynolds numbers therefore, once more, the model and solver validation in complete.

10 Some Results

We will now briefly analyze the result obtained applying the developed topology optimization algorithm for the minimization of the total power dissipation to some simple problems.

10.1 Simple channel

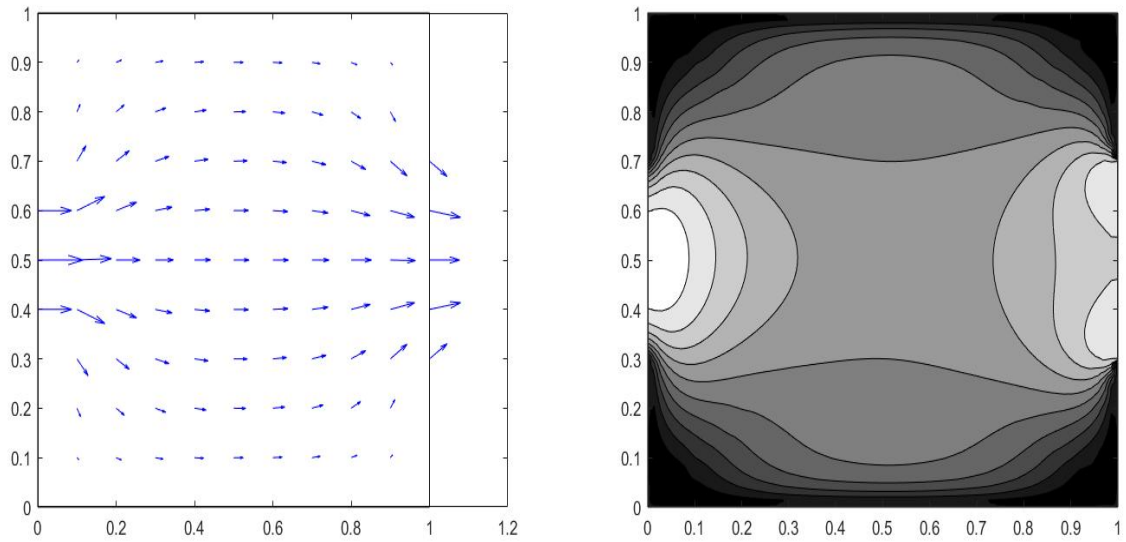
We here analyze the best form of a $[0,1]\times[0,1]$ domain presented above or the "simple channel" problem. The parameters used here are

$$\begin{aligned} V_r &= 0.6; \\ V_0 &= 1; \\ \alpha_{min} &= 0; \\ \alpha_{max} &= 1e3; \\ q &= 10; \\ \mu &= 1; \\ \rho &= 1; \end{aligned}$$

with an parabolic velocity entering as input in the left wall, id 3 in figure (**??**)

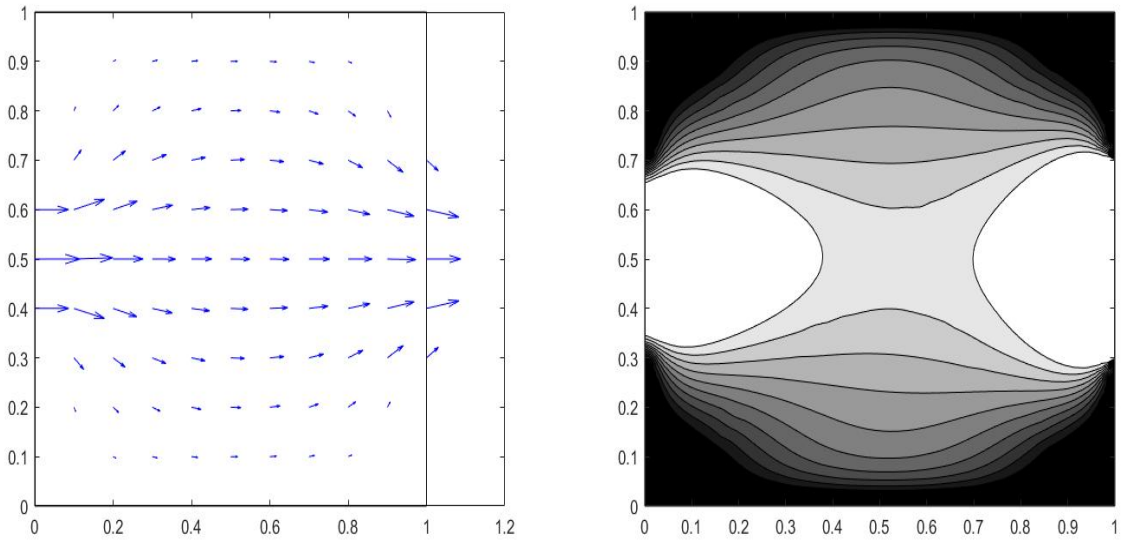
$$V_{In}(x,y) = 6.25 \cdot \begin{cases} (y-0.3)(0.7-y) \\ 0 \end{cases}$$

where the 6.25 is just a factor to have the maximum entering velocity of 0.25 and the formulation is such that the velocity is identically zero at the walls. Clearly in this case the result for the best layout of the

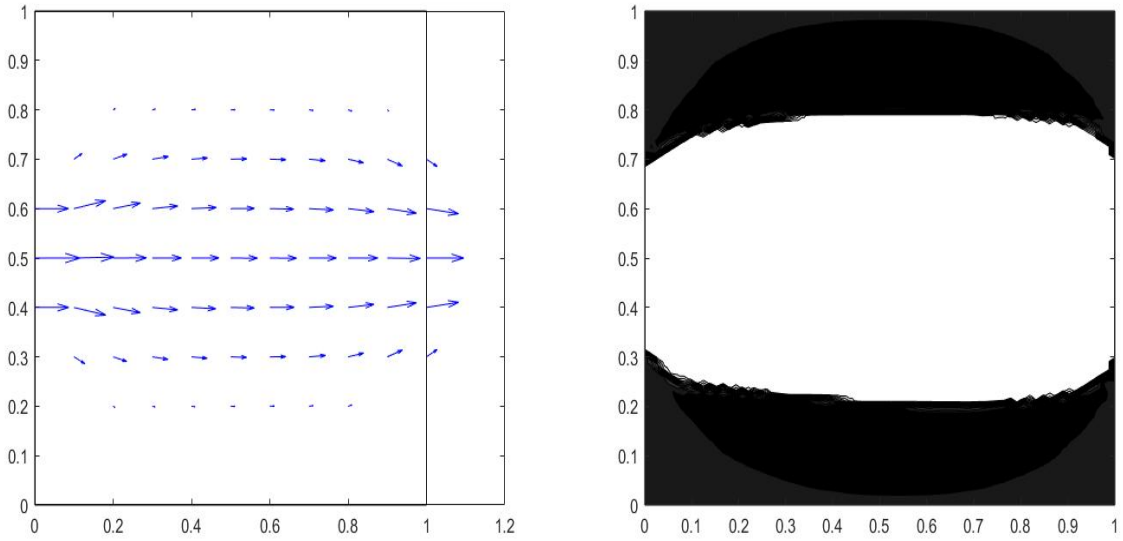


First iteration of the top opt algorithm for the simple channel problem. Velocity field (left) and design domain (right). Objective functional value: 2.44.

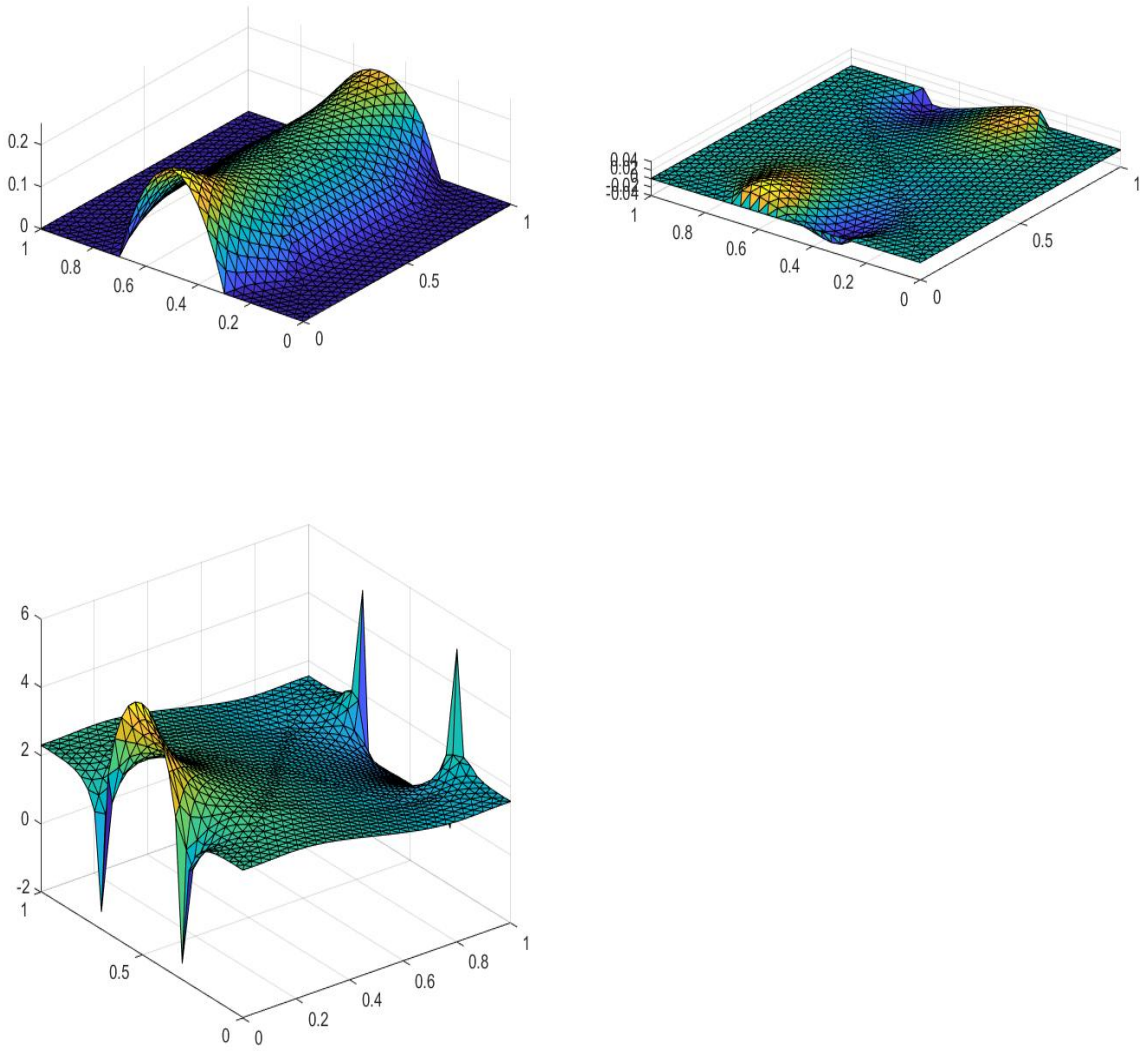
domain respect our physical interpretation of what an ideal channel would be with a maximum portion of 0.6 of the total volume available for the fluid. In this case we have $Re = 0.25$ thus the results are almost the same using the Navier Stokes or simply the Stokes solver. Below we report some final optimization results varying the maximum volume ratio available for the fluid, V_r.



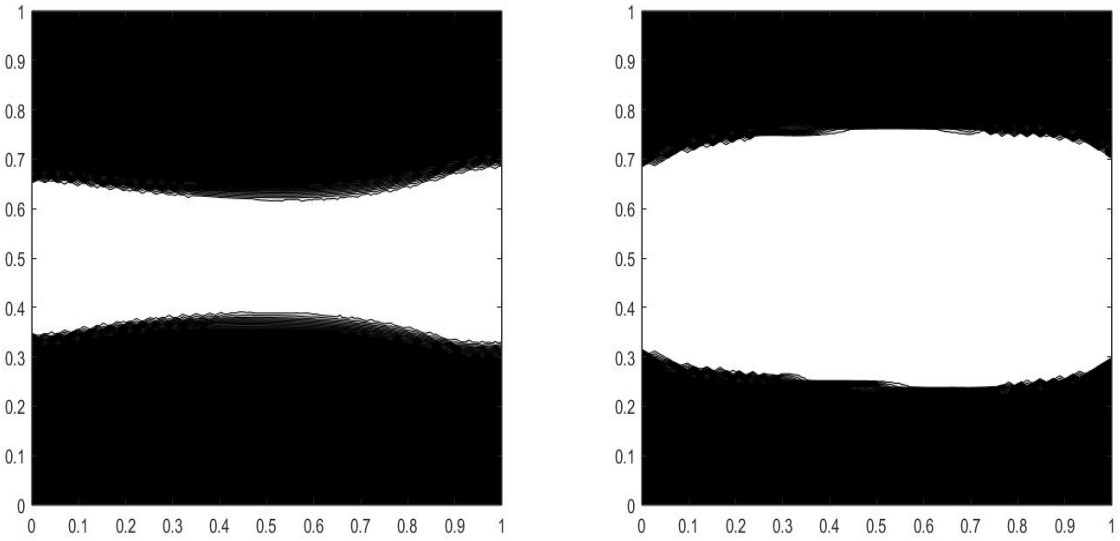
Third iteration of the top opt algorithm for the simple channel problem. Velocity field (left) and design domain (right). Objective functional value: 1.01.



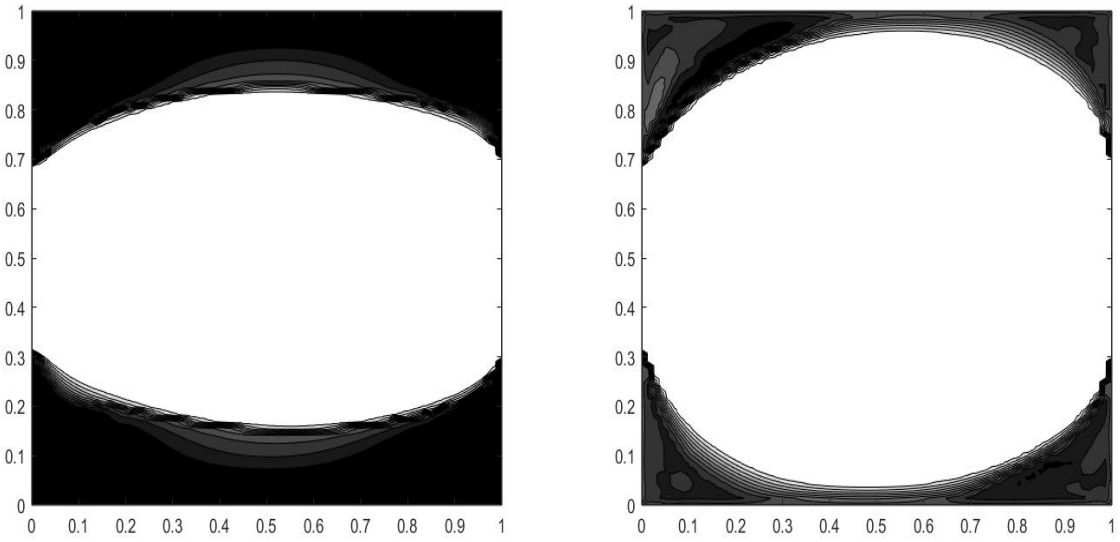
Tenth iteration of the top opt algorithm for the simple channel problem. Velocity field (left) and design domain (right). Change from previous configuration is less then 0.1 in norm, so we consider this as the final result. Objective functional value: 0.34.



Final results in terms of horizontal velocity (left), vertical velocity (middle) and pressure (right).



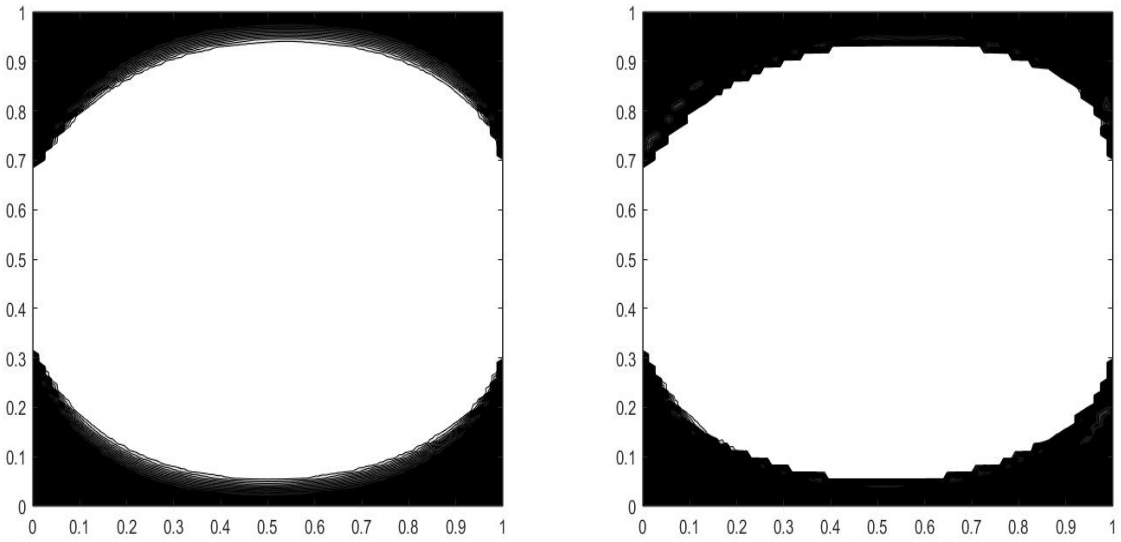
Optimal layouts for $V_r = 0.3, 0.5$ *from left to right.*



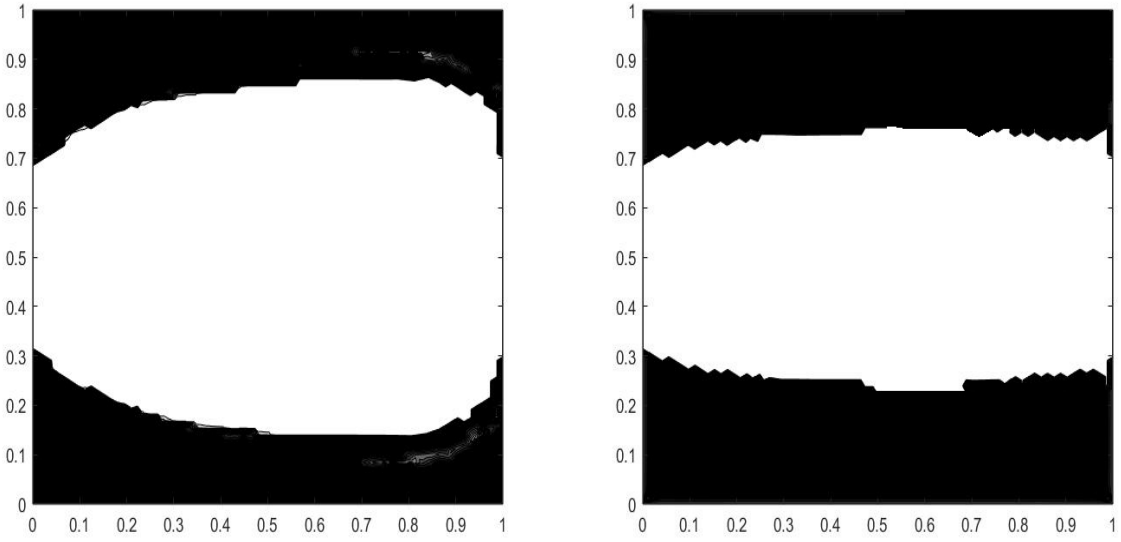
Optimal layouts for $V_r = 0.7, 0.9$ *from left to right.*

What is remarkable is that for small velocities and low Reynold numbers, the algorithm tries to make as uniform as possible the flow in the middle of the domain. Since it always has to satisfy the condition to be zero at the wall the velocity flow will still be parabolic in the vertical direction, but with a lower slope, thus diminishing the gradient. Increasing the Reynolds number on the opposite even for large V_r the optimal layout is almost like a continuous channel, to prevent the formation of eddies and pressure loss in the entering and leaving of the fluid.

See how increasing the Reynolds numbers the algorithm finds a narrow channel more convenient than



Optimal layouts for $Re = 1, 10$ *from left to right for* $V_r = 0.8$.



Optimal layouts for $Re = 100, 1000$ *from left to right for* $V_r = 0.8$.

a wider, because deviating the trajectory of the fluid requires more and more energy. In particular the restriction of the channel starts near the inlet to prevent the formation of energy consuming eddies.

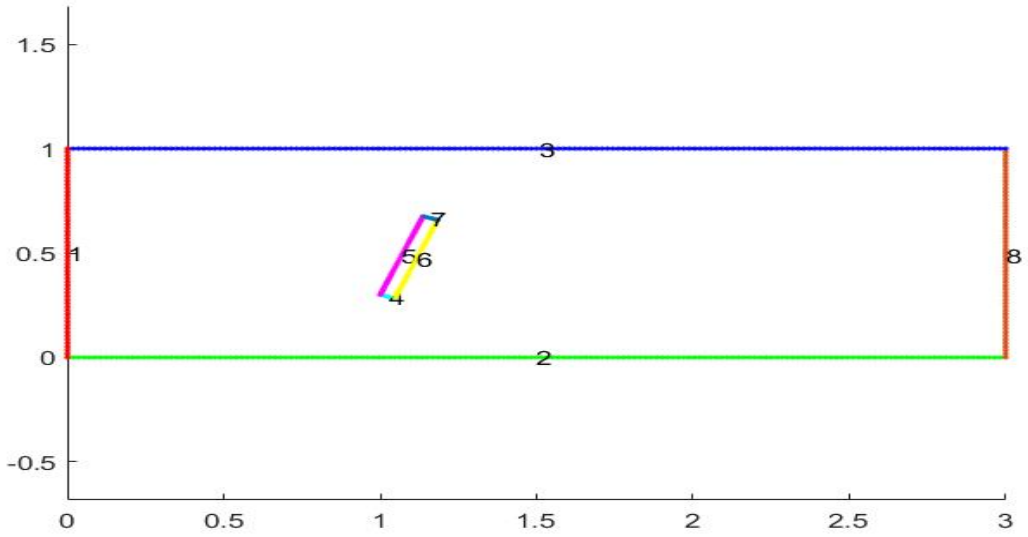
10.2 Plate Rect

The problem is to search for the best layout in a $[0,3] \times [0,1]$ domain with a rectangular obstacle. The parameters used here are

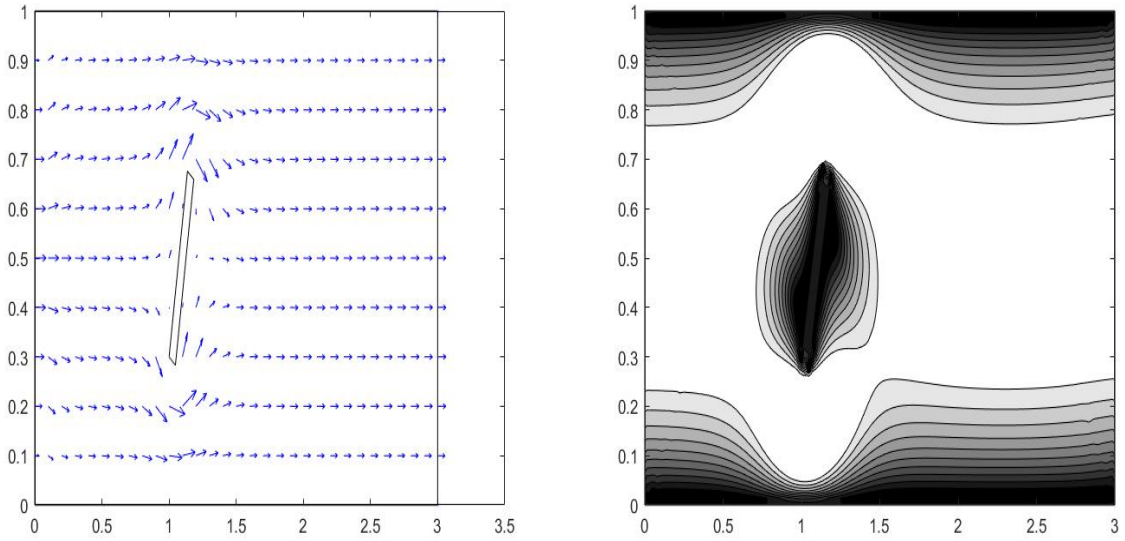
$$\begin{aligned} V_r &= 0.8; \\ V_0 &= 2.9; \\ \alpha_{min} &= 0; \\ \alpha_{max} &= 1e3; \\ q &= 1; \\ \mu &= 1; \\ \rho &= 1; \end{aligned}$$

with a parabolic velocity entering as input in the left wall, id 1 in figure (48)

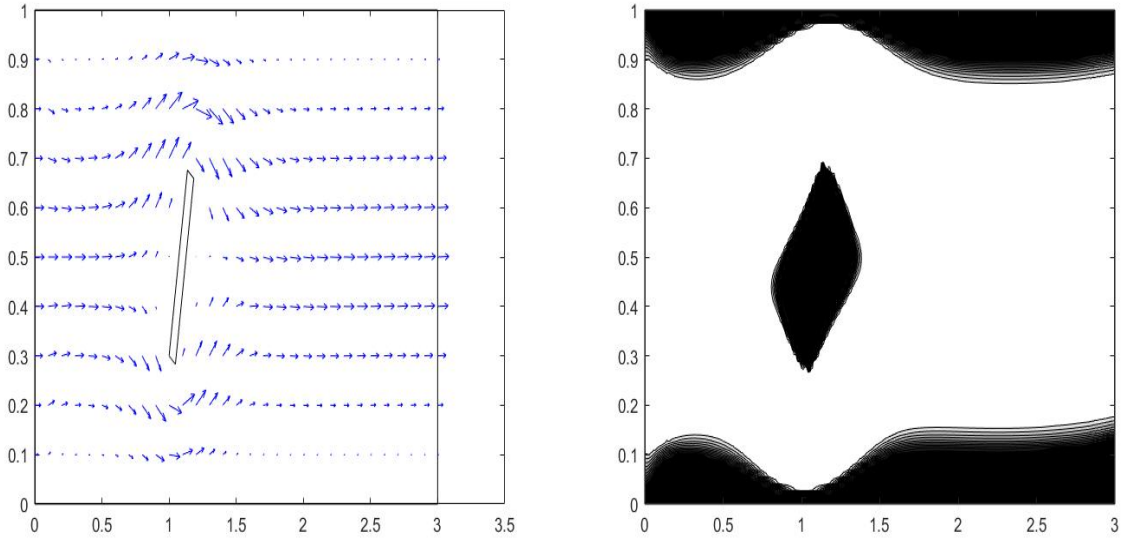
$$V_{In}(x,y) = 4 \cdot \begin{cases} y(1-y) \\ 0 \end{cases}$$



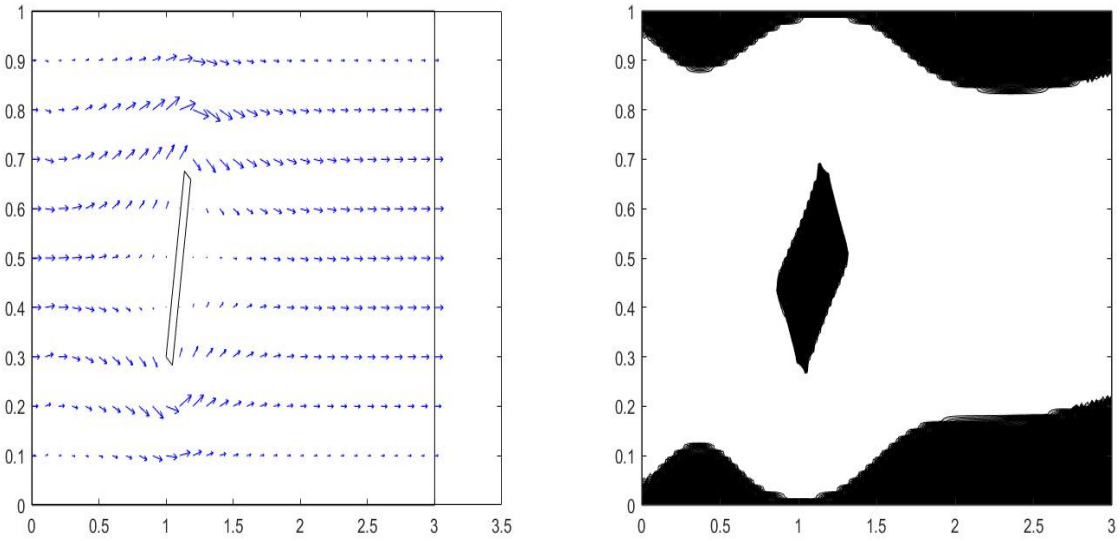
"Plate Rect" problem configuration.



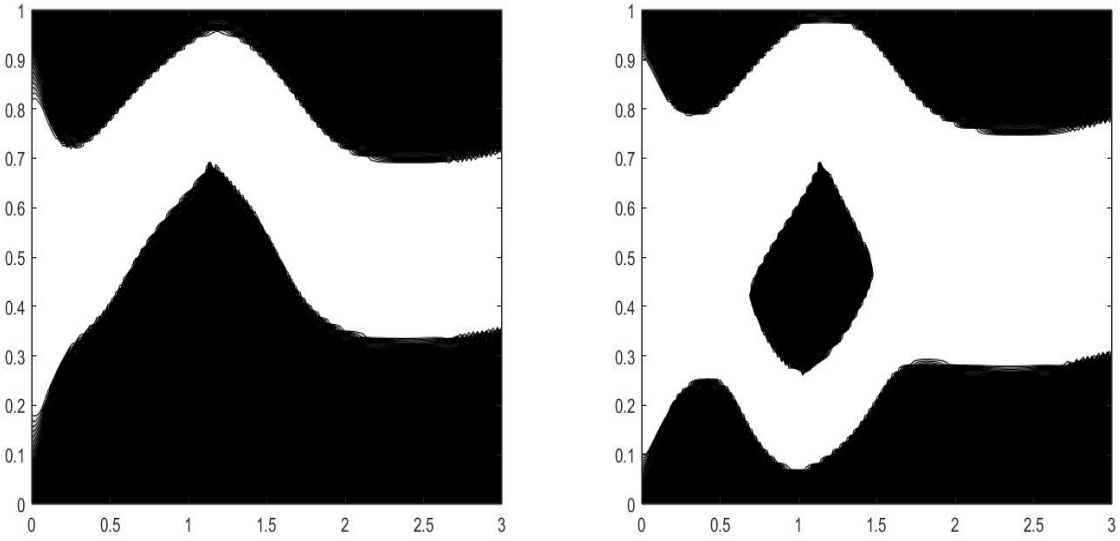
First iteration of the top opt algorithm for the "plate rect" problem. Velocity field (left) and design domain (right). Objective functional value: 116.88.



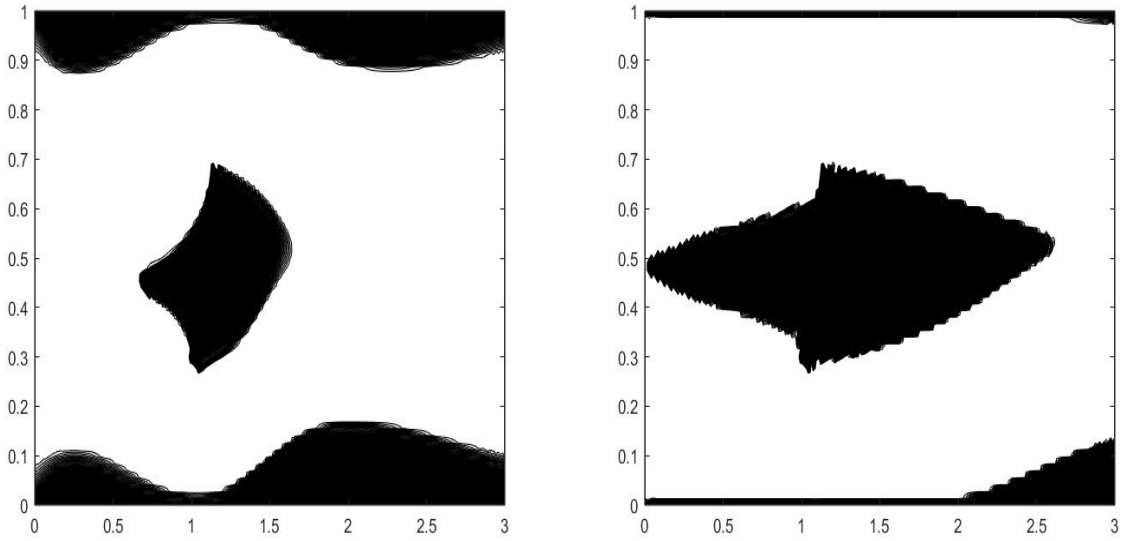
Second iteration of the top opt algorithm for the "plate rect" problem. Velocity field (left) and design domain (right). Objective functional value: 67.48.



Fifth iteration of the top opt algorithm for the "plate rect" problem. Velocity field (left) and design domain (right). Change from previous configuration is less then 0.1 in norm, so we consider this as the final result. Objective functional value: 61.32.



Optimal layouts for $V_r = 0.4, 0.6$ *from left to right.*



Re

Optimal layout t $V_r = 0.8$ for $Re = 30$ (left), and $Re = 100$ (right).

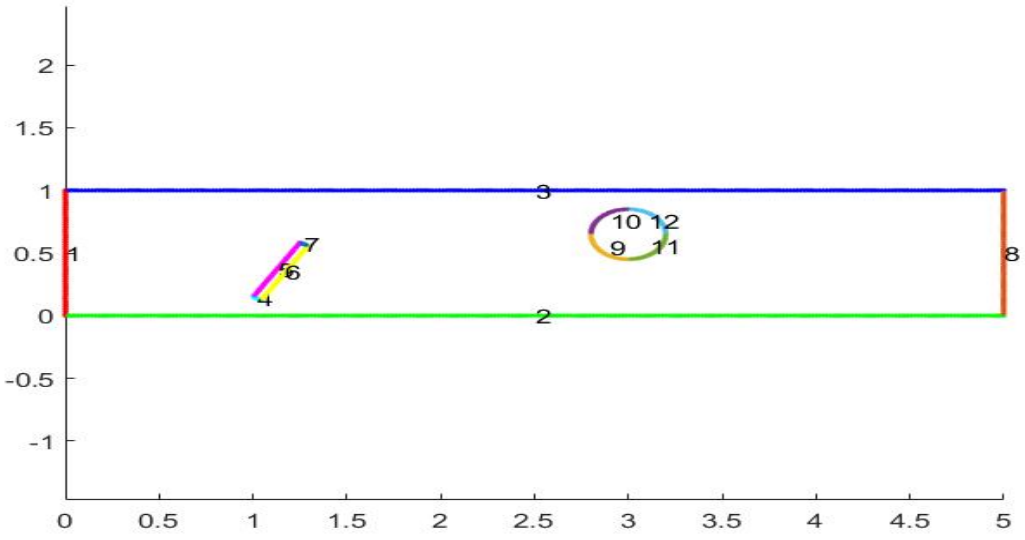
10.3 Plate and cylinder

As final example we consider now the problem showed in (16), with parameters

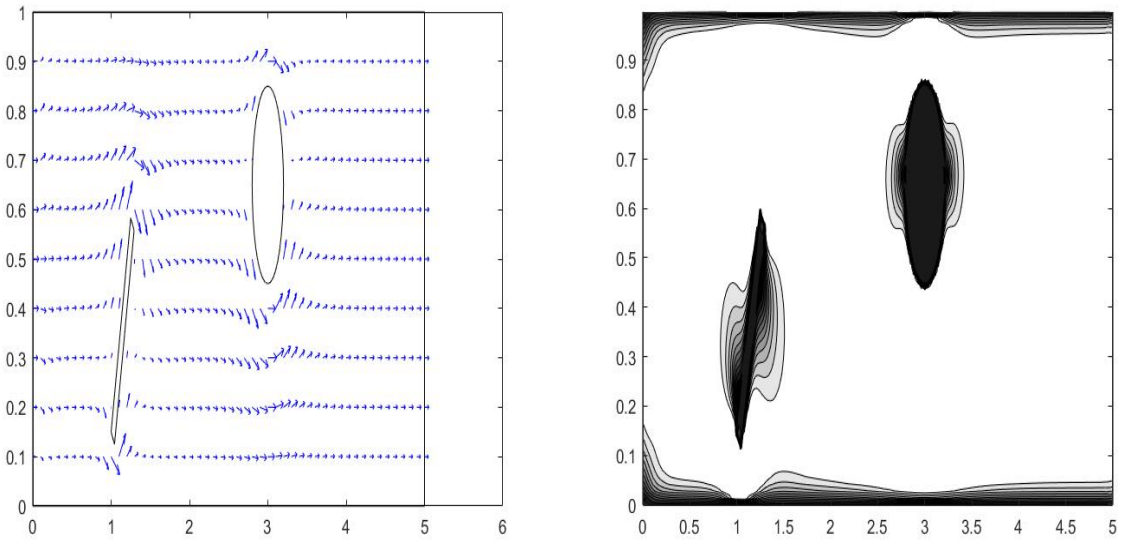
$$\begin{aligned} V_r &= 0.8; \\ V_0 &= 4.8; \\ \alpha_{min} &= 0; \\ \alpha_{max} &= 1e3; \\ q &= 1; \\ \mu &= 1; \\ \rho &= 1; \end{aligned}$$

and a parabolic velocity entering as input in the left wall, id 1 in figure (48)

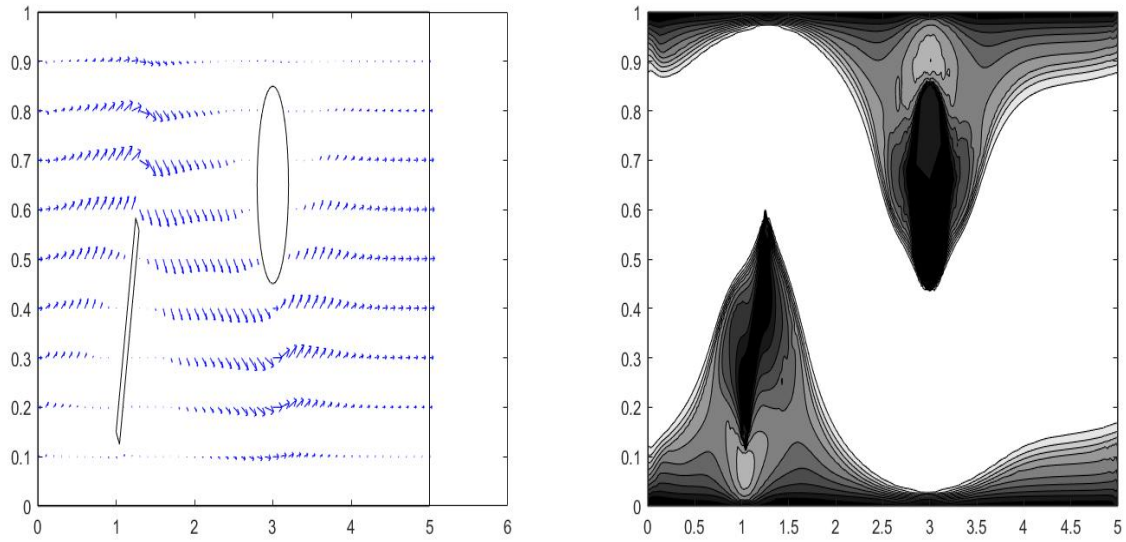
$$V_{In}(x, y) = 4 \cdot \begin{cases} y(1-y) \\ 0 \end{cases}$$



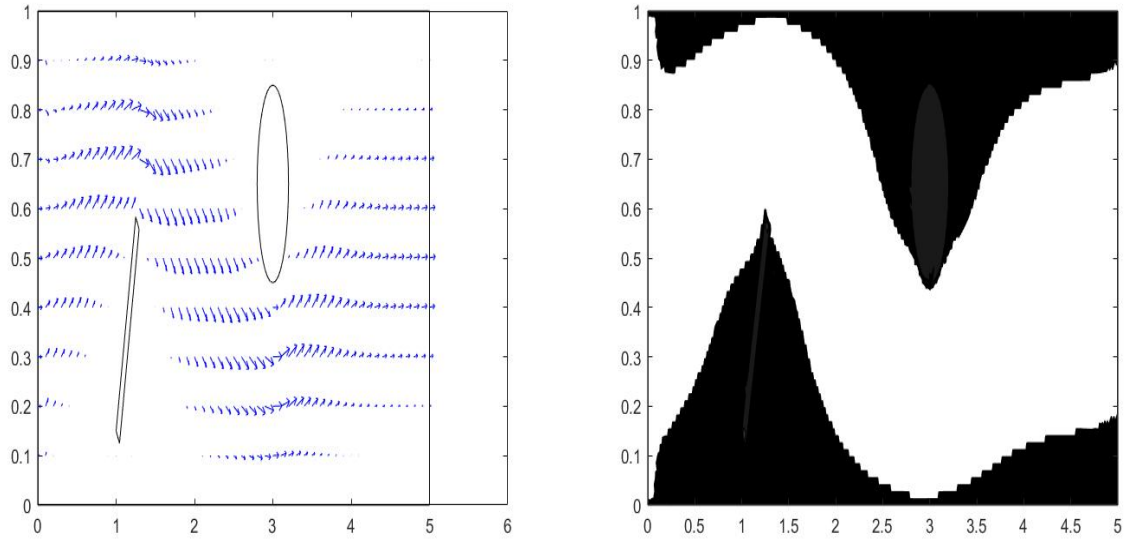
"Plate and cylinder" problem configuration.



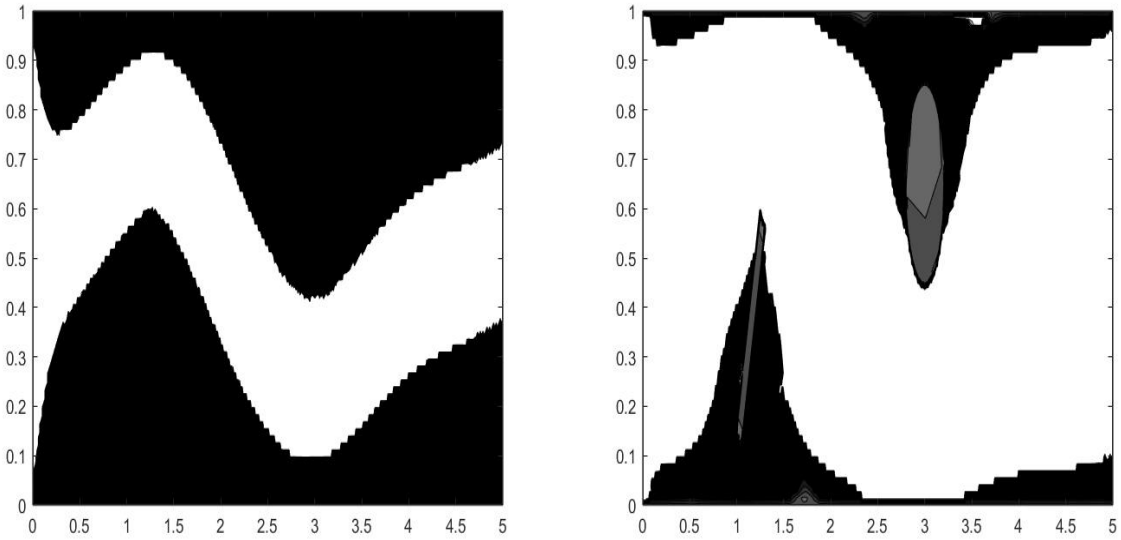
First iteration of the top opt algorithm for the "plate and cylinder" problem. Velocity field (left) and design domain (right). Objective functional value: 401.62.



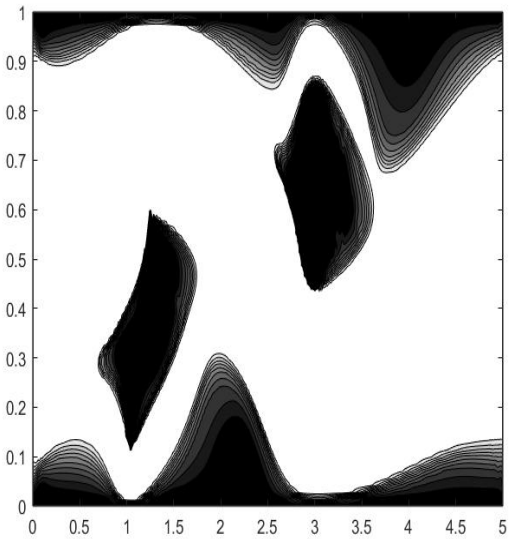
Third iteration of the top opt algorithm for the "plate and cylinde problem. Velocity field (left) and design domain (right). Objective functional value: 177.65.



Sixth iteration of the top opt algorithm for the"plate rect" problem. Velocity field (left) and design domain (right). Change from previous configuration is less then 0.1 in norm, so we consider this as the final result. Objective functional value: 137.31.



Optimal layouts for $V_r = 0.4, 1.0$ *from left to right.*



(a) *Optimal layouts for* $Re = 100$ *and* $V_r = 0.8$.

11 Conclusions

In this project we have analyzed the theory and implementation of a layout optimization in a topology approach, basing on the work conducted in [5] and [20], with the so called "density method" for the full incompressible steady Navier - Stokes equations in 2 dimensions.
We developed a FEM code to solve such governing equations and their relative adjoint formulation needed for the sensitivity analysis of the chosen functional to minimize, using a "p1-iso-p2" approach for the treatment of the velocity/pressure functional spaces, as shown in [12]. This solver seemed to work fine with respect to some comparisons on the solution conducted against the commercial code COMSOL [3].
The resulting optimal layouts for the chosen experiments seemed reasonable and this is encouraging for future developments of the algorithm in practical applications. Some problems have been encountered in the case of $Re > 100$, but the code is developed to work fine in laminar cases, and different approaches, like the level set method [5] may be applied for higher Reynold numbers.
An interesting research field is that in many cases however the algorithm can oscillate between two strong competing minima for the problem, and even an accurate calibration of the optimization parameters may not be enough to discriminate between the two or to find the best one.
In conclusion we have shown how an implementation of such theory could be a useful tool for designing fluidic devices at least for low Reynolds numbers and with different amounts of material available for the channels.

12 Appendix: main.m file

```
%%% Main script for topology optimization of Stokes problem in a channel
clear; close all; clc;

% Define the 2D mesh and equation BC's and coefficients
meshdir='mesh_COMSOL\lastra_rect_fine\lastra_rect_fine.txt';
nfig   = 1;

% flag_functional = 1 for min (\alpha ||u||^2)
%                   2 for min (\alpha ||u||^2 + 1/2* \mu ||grad(u)||^2)
%-------------------
flag_functional = 2;
inlet_id        = [1];
outlet_id       = [8];
%-------------------------------------------------------------------------
% SET PARAMETERS
%-------------------------------------------------------------------------
UseNStokes = true;

flag      = 0; % flag=1 to see the mesh
flag_BC   = 0; % BC: 0 for FAST penalty method
               %   : 1 for ACCURATE lifting function method

seeEdgeID = true;

V_r       = 0.2;
V_0       = 2.9;  %%% !!!!!!!!!!!! TO BE SET WITH GEOMETRY !!!!!!!!
alpha_min = 0;
alpha_max = 1e3;
q         = 1;
mu        = 1;
rho       = 1;
f_1       = @(x,y) 0;
f_2       = @(x,y) 0;
g         = @(x,y) 0;
tau       = 0.00001;

V_in = @(x,y) 4*[(y)*(1-y); 0];
%-------------------------------------------------------------------------
% INITIALIZE SOLVER
%-------------------------------------------------------------------------
tic;
disp('INITIALIZE SOLVER FOR MASTER PROBLEM');
disp('-----------------------------------');
disp('-----------------------------------');

if UseNStokes
    NStokes = RevisedNavierStokes(meshdir, nfig, mu, rho, f_1, f_2, g, tau, flag);
else
    NStokes = RevisedStokes(meshdir,nfig, mu, f_1, f_2, g, flag);
end

if seeEdgeID
    figure(2)
    hold on;
    % TESTING PLOT SIDES
    PlotSideNodes(NStokes.X_limits, NStokes.Y_limits, NStokes.V.coord, ...
        NStokes.Bound.Nodes, NStokes.Bound.NodesId, 1:max(NStokes.Bound.EdgesId));
    drawnow;
    hold off;
end

limits = [NStokes.X_limits, NStokes.Y_limits];
nfig = nfig+1;

toc;
%-------------------------------------------------------------------------
% INITIALIZE PARAMETER GAMMA
```

```
68  %-------------------------------------------------------------------------
69  gamma = 1.0 * ones(NStokes.V.Nodes,1);
70  %-------------------------------------------------------------------------
71  % SET BC
72  %-------------------------------------------------------------------------
73  tic;
74  disp('-----------------------------------')
75  disp('SET BC');
76  disp('-----------------------------------'); disp('-----------------------------------');
77
78
79
80
81  % INLET BC
82  %-------------------------------
83  % Edge Index imported from COMSOL
84  for i = 1:length(inlet_id)
85      NStokes = NStokes.setEdgeDirBC(inlet_id(i), V_in);
86  end
87
88
89  % WALL BC
90  %-------------------------------
91  for i = 1 : NStokes.Bound.NBound
92      if ismember(i,inlet_id) || ismember(i,outlet_id)
93          continue
94      end
95      NStokes = NStokes.setEdgeDirBC(i, @(x,y) [0,0]);
96  end
97
98
99  % DO NOTHING OUTFLOW BC
100 %-------------------------------
101 for i = 1:length(outlet_id)
102     NStokes = NStokes.setEdgeNeuBC(outlet_id(i), [0,0]);
103 end
104 toc;
105 %-------------------------------------------------------------------------
106 % PREPRO
107 %-------------------------------------------------------------------------
108 tic;
109 disp('-----------------------------------')
110 disp('PREPRO');
111 disp('-----------------------------------'); disp('-----------------------------------');
112 NStokes = NStokes.prepro;
113 toc;
114 %-------------------------------------------------------------------------
115 % INITIALIZE ADJOINT AND MMA SOLVER
116 %-------------------------------------------------------------------------
117 tic;
118 disp('-----------------------------------')
119 disp('ADJOINT SYSTEM AND MMA INITIALIZATION');
120 disp('-----------------------------------'); disp('-----------------------------------');
121 % initialize Adjoint
122 if UseNStokes
123     Adjoint = ADJOINT_NS(NStokes, mu, rho, flag_functional);
124 else
125     Adjoint = ADJOINT(NStokes, mu, rho, flag_functional);
126 end
127 % initialize MMA
128 Optimizer = MMA(q, V_r, V_0, alpha_min, alpha_max, NStokes.V, mu, flag_functional);
129 toc;
130 %-------------------------------------------------------------------------
131 loop = 0;
132 toll = 1e-3;
133 change = toll + 1;
134 [u1_0, u2_0] = NStokes.zeroInitCond();
135
136 %%  LAUNCH SOLVER
137 while (change > toll)
138 %-------------------------------------------------------------------------
```

```
% UPDATE PARAMETER GAMMA
%-------------------------------------------------------------------------
alpha = alpha_min + (alpha_max - alpha_min).*q.*(1-gamma)./(q+gamma);
NStokes.alpha = alpha;
%-------------------------------------------------------------------------
% SOLVE
%-------------------------------------------------------------------------
tic;
disp('-----------------------------------')
disp('SOLVE MASTER PROBLEM');
disp('-----------------------------------');
disp('-----------------------------------');
[uh1, uh2, ph] = NStokes.StatSolver(flag_BC, u1_0, u2_0);
% u1_0 = uh1; u2_0 = uh2;
toc;

%---------------------------------------
% UPDATE STREAMLINE AND DESIGN PLOTS
%---------------------------------------
figure(3)
for iedge = 1:size(NStokes.Bound.Edges,1)
    p1 = NStokes.V.coord(NStokes.Bound.Edges(iedge,1),:);
    p2 = NStokes.V.coord(NStokes.Bound.Edges(iedge,2),:);
    plot([p1(1), p2(1)], [p1(2), p2(2)], 'k-');
    hold on;
end

[x,y] = meshgrid(NStokes.X_limits(1):0.1:NStokes.X_limits(2), ...
                 NStokes.Y_limits(1):0.1:NStokes.Y_limits(2));
u = zeros(size(x)); v = zeros(size(x));
for i = 1:numel(x)
    [u(i), v(i)]= NStokes.getApproxValue_v([x(i);y(i)],uh1,uh2);
    if norm([u(i), v(i)]) < 1e-2
        u(i) = 0;
        v(i) = 0;
    end
end
quiver(x,y,u,v, 'Color', 'b');
% streamslice(x,y,u,v);
hold off;

%-------------------------------------------------------------------------
% BUILD ADJOINT SYSTEM
%-------------------------------------------------------------------------
tic;
disp('-----------------------------------')
disp('Build Adjoint system');
disp('-----------------------------------');
disp('-----------------------------------');
Adjoint = Adjoint.Update(uh1,uh2,ph,alpha);

switch flag_functional
    case 1
        Adjoint.f_1    = -2.*alpha.*uh1/rho;
        Adjoint.f_2    = -2.*alpha.*uh2/rho;
        Adjoint.g      = zeros(Adjoint.P.Nodes);
    case 2
        Adjoint.f_1    = -2*alpha.*uh1/rho;
        Adjoint.f_2    = -2*alpha.*uh2/rho;
        Adjoint.g      = zeros(Adjoint.P.Nodes);
    case 3
        Adjoint.f_1    = zeros(Adjoint.V.Nodes);
        Adjoint.f_2    = zeros(Adjoint.V.Nodes);
        Adjoint.g      = zeros(Adjoint.P.Nodes);
end

Adjoint = Adjoint.clearBC();
toc;

%-------------------------------------------------------------------------
% SET ADJOINT BC
```

```
%-------------------------------------------------------------------------
tic;
disp('-----------------------------------')
disp('SET ADJOINT BC');
disp('-----------------------------------');
disp('-----------------------------------');



for i = 1:NStokes.Bound.NBound
    if ismember(i,outlet_id)
        continue
    end
    Adjoint = Adjoint.setEdgeDirBC(i,@(x,y)[0,0]);
end
%------------------------------
for i = 1:length(outlet_id)
    Adjoint = Adjoint.setEdgeNeuBC(outlet_id(i));
end

toc;
%-------------------------------------------------------------------------
% SOLVE
%-------------------------------------------------------------------------
tic;
disp('-----------------------------------')
disp('ADJOINT SOLVER');
disp('-----------------------------------');
disp('-----------------------------------')
Adjoint = Adjoint.Prepro;
[ua1, ua2, pa] = Adjoint.DirectStatSolver(flag_BC);
toc;

%-------------------------------------------------------------------------
% CALL MMA METHOD
%-------------------------------------------------------------------------
tic;
disp('-----------------------------------')
disp('UPDATE AND SOLVE MMA');
disp('-----------------------------------');
disp('-----------------------------------')
Optimizer = Optimizer.Update(uh1, uh2, ua1, ua2);
[gamma_new, f0val, Vol] = Optimizer.solve(gamma,loop);
toc;

% PRINT RESULTS
change = Solver.evalErr(gamma, gamma_new, NStokes.V);
loop  = loop + 1;
disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',f0val) ...
      ' Vol.: ' sprintf('%6.3f',Vol) ...
      ' ch.: ' sprintf('%6.3f',change )]);

% % PLOT DENSITIES
figure(4)
x=NStokes.V.coord(:,1); y=NStokes.V.coord(:,2);
M=delaunay(x,y);
cd triplot
tricontf(x,y,M,gamma_new,10);
colormap(gray)
gamma = gamma_new;
axis(limits);
cd ..

end

%% PRINT FINAL RESULTS
NStokes.PrintRes(uh1,uh2,ph,5);
```

Bibliography

[1] R.B. Haber A. Gersborg-Hansen O. Sigmund. "Topology optimization of channel flow problems". In: (2005).

[2] Petersson J. Borrvall T. *Topology optimization of fluids in Stokes flow.* International Journal for Numerical Methods in Fluids 2003; 41:77–107, 2003.

[3] *COMSOL Multiphysics®, v. 6.0. www.comsol.com. COMSOL AB, Stockholm, Sweden.*

[4] *Fluid Deformation and Energy Dissipation Derivations.* `https://aguaclara.github.io/Textbook/Fluid_Deformation_and_Energy_Dissipation/FDED_Derivations.html`.

[5] Graves S. Gavish B. *The travelling salesman problem and related problems.* Operations Research Center, Massachusetts Institute of Technology, Cambridge, 2018.

[6] S. Kurcyusz J. Zowe. "Regularity and Stability for the Mathematical Programming Problem in Banach Spaces". In: (1979).

[7] Rolf Rannacher John G. Heywood and Stefan Turek. "Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations". In: (1996).

[8] Fridolin Okkels Laurits Højgaard Olesen and Henrik Bruus. "A high-level programming-language implementation of topology optimization applied to steady-state Navier–Stokes flow". In: (2005).

[9] M. Ulbrich M. Hinze R. Pinnau and S. Ulbrich. *Optimization with PDE Constraints.* Springer, 2009.

[10] MATLAB. *version R2021b.* Natick, Massachusetts: The MathWorks Inc., 2021.

[11] O.Sigmund. "A 99 line topology optimization code written in Matlab". In: (2001).

[12] Mario Putti. *Notes on Numerical Methods for Differential Equations.* Department of Mathematics "Tullio Levi-Civita", University of Padua, 2021.

[13] Alfio Quarteroni. *Numerical Models for Differential Problems.* Springer, 2014.

[14] Rolf Rannacher. "Finite Element Methods for the Incompressible Navier-Stokes Equations". In: (1999).

[15] Stephen M. Robinson. "Stability Theory for Systems of Inequalities, Part II: Differentiable Nonlinear Systems". In: (1976).

[16] Arthur Stück. *Adjoint Navier–Stokes Methods for Hydrodynamic Shape Optimisation.* 2011.

[17] Krister Svanberg. "The Method of Moving Asymptotes - A new Method for Structural Optimization". In: (1987).

[18] Krister Svanberg. "The Method of Moving Asymptotes - Modelling aspects and solution schemes". In: (1998).

[19] Z. Liu Y. Deng Y. Wu. *Topology Optimization theory for Laminar Flow.* Springer, 2018.

[20] K. Yosida. *Functional Analysis.* Springer, 1980.

[21] E. Zeidler. *Nonlinear Functional Analysis and its Applications. I, Fixed-Point Theorems.* Springer, 1986.