Dimensional model reduction of fluid-structure interaction models.

Master of Science in Biomedical Engineering - Biomechanics and Biomaterials

Valeria Secchi

896127

Supervisor

Paolo Zunino

Co-Supervisor

Miguel Fernández

Politecnico di Milano

18 December 2019

Acknowledgements

I would like to thank Professor Paolo Zunino, my supervisor, for assisting me in writing this thesis and for his availability. I would also like to thank Professor Miguel A. Fernández, my co-supervisor, for the support he gave me during my time at the Inria in Paris. I also thank the entire Inria team for their disponibility and help during my stay in France.

Abstract

Fluid-Structure Interaction (FSI) is among the most challenging problems in engineering. This work aims to study a 2D-0D dimensional model reduction of FSI. We consider an imcompressible flow, governed by Navier-Stokes equations, around a rigid obstacle, which may move with a linear elastic constraint. In particural, we discuss a classic benchmark problem of computational fluid dynamics (CFD) and FSI, that is the 2D flow around a disc. The main challenge in this development of the 2D-0D reduction model is how to represent the rigid disc as a point. We formulate a mathematical model that addresses this problem through the use of Lagrange multiplier method to enforce the kinematic and dynamic constraints between the solid and the fluid. Specifically, to deal with the pointwise evaluation of the fluid velocity at the center of the disc, we replace it with the average of the fluid velocity on the fluid-solid interface. Once the mathematical model has been formulated, we proceed with the numerical discretization using the finite element method and FreeFem++ as solver. At the computational level, our FSI model requires to use a computational mesh for the fluid domain that does not conform with the profile of the disc. We have overcome this difficulty by creating an algorithm that calculates the intersection between a parameterized circumference, representing the disc, and each edge of the mesh for the fluid domain that crosses it, using the interpolation of the Gauss point. Then we write the algebraic formulation in a matrix form in order to compute the numerical solution, obtaining a block matrix that represents the fluid structure interaction. To conclude, we conduct some tests to verify the validity of the work and we compare the results obtained using the developed algorithm with results available in literature.

Sommario

I problemi di Interazione Fluido-Struttura (FSI) sono tra i più delicati nel campo dell'ingegneria. Questa tesi si propone l'obiettivo di studiare un modello ridotto 2D-0D di un FSI. Abbiamo considerato un fluido incomprimibile, governato dalle equazioni di Navier-Stokes, che scorre attorno ad un ostacolo rigido che può muoversi secondo un modello elastico lineare. In particolare, abbiamo discusso un classico problema di riferimento nel campo della fluidodinamica computazionale (CFD) ed in quello di FSI, ovvero un flusso 2D attorno ad un disco. La sfida principale nello sviluppo di questo modello ridotto 2D-0D è stata la ricerca di una rappresentazione efficace del disco rigido come un solo punto. Abbiamo quindi formulato un modello matematico che affronta questo problema attraverso l'uso del metodo dei moltiplicatori di Lagrange per imporre i vincoli cinematici e dinamici tra il solido ed il fluido. In particolare, per risolvere la difficoltà della valutazione puntuale della velocità del fluido al centro del disco, abbiamo sostituito quest'ultima con la media della velocità del fluido all'interfaccia fluido-struttura. Una volta formulato il modello matematico, abbiamo proceduto con la discretizzazione numerica utilizzando il metodo degli elementi finiti e FreeFem++ come solutore. A livello computazionale, il nostro modello FSI richiede l'utilizzo di una mesh computazionale per il dominio fluido non conforme al profilo del disco. Questa difficoltà è stata superata creando un algoritmo che calcola l'intersezione tra una circonferenza parametrizzata, che rappresenta il disco, ed ogni elemento della mesh del dominio fluido che interseca, utilizzando le formule di quadratura di Gauss. In seguito abbiamo scritto la formulazione algebrica in forma matriciale per calcolare la soluzione numerica, ottenendo una matrice a blocchi che rappresenta l'interazione fluido-struttura. In conclusione, abbiamo effettuato alcuni test per verificare la validità del lavoro e abbiamo confrontato i risultati ottenuti utilizzando l'algoritmo sviluppato con i risultati disponibili in letteratura.

Contents

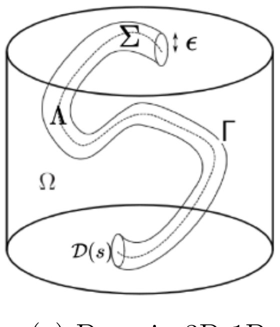
List of Figures

Chapter 1

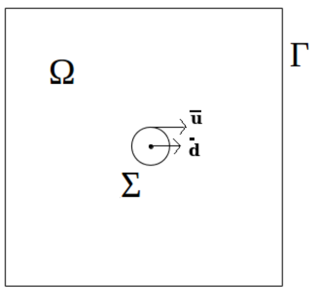
Introduction

1.1 Motivations

The aim of this project is to develop a computational model to describe the mechanical behavior of thin and slender structures (such as wires) when embedded into fluid in motion. This result would be important to enable simulations of phenomena that are still computational too expensive at the moment, such as the direct simulation of vegetation with winds or with water flows.



(a) Domain 3D-1D



(b) Domain 2D-0D

Figure 1.1: Model's domains

In order to develop a model of 3D-1D interactions, we start from a simplified case, namely 2D-0D model.

1.2 Fluid structure interaction problem

Fluid-structure interaction (FSI) is a multiphysics coupling between the laws that describe fluid dynamics and structural mechanics. This phenomenon is characterized by interactions (which can be stable or oscillatory) between a deformable or moving structure and a external or internal fluid flow.
When a fluid flow encounters a structure, stresses and strains are exerted on the solid object - forces that can lead to deformations. These deformations can be quite large or very small, depending on the pressure and velocity of the flow and the material properties of the actual structure.

If the deformations of the structure are small enough and the variations in time are also relatively slow, the fluid's behavior will not be greatly affected by the deformation, and we can concentrate ourselves only with the resultant stresses in the solid parts. However, if the variations in time are fast, greater than a few cycles per second, then even small structural deformations will lead to pressure waves in the fluid. These pressure waves lead to the radiation of sound from vibrating structures. Such problems can be treated as an acoustic-structure interaction, rather than a fluid-structure interaction.
If the deformations of the structure are large, the velocity and pressure fields of the fluid will change as a result, and we need to treat the problem as a bidirectionally coupled multiphysics analysis: the fluid flow and pressure fields affect the structural deformations, and the structural deformations affect the flow and pressure.

Fluid-structure interaction problems and multiphysics problems in general are often too complex to be solved analytically. They have to be studied by means of experiments or numerical simulation. Research in the fields of computational fluid dynamics and computational structural dynamics is still ongoing but the maturity of these fields enables numerical simulation of fluid-structure interaction. Some works that deal with fluid-interaction model are for example [2], [8] and [1].

1.2.1 Simulation's approach

Two main approaches exist for the simulation of fluid-structure interaction problems:

- Monolithic approach: the equations governing the flow and the structure are solved simultaneously
- Partitioned approach: the equations governing the flow and the displacement of the structure are solved separately, with two distinct solvers, and coupling is achieved through sub-iterations

The monolithic approach requires a code developed for this particular combination of physical problems whereas the partitioned approach preserves software modularity because an existing flow solver and structural solver may be coupled. Moreover, the partitioned approach facilitates solution of the flow equations and the structural equations with different, possibly more efficient techniques which have been developed specifically for either flow equations or structural equations. However, stability of the coupling algorithm is not always garanteed in partitioned simulations.

In conclusion, the partitioned approach allows reusing existing software which is an attractive advantage, but stability of the coupling method needs to be taken into consideration.

1.2.2 Problem formulation

The interaction of a flexible structure with a flowing fluid in which it is submersed or by which it is surrounded gives rise to a rich variety of physical phenomena with applications in many fields of engineering, for example, the stability and response of aircraft wings, the flow of blood through arteries, the response of bridges and tall buildings to winds, the vibration of turbine and compressor blades, and the oscillation of heat exchangers.

A fluid-structure problem is defined by the fluid equations, the structure equations and by the transmission conditions at the fluid-structure interface:

$$u_{\mathrm{f}} = u_{\mathrm{s}} \tag{1.1}$$

$$\sigma_{\mathrm{f}} \cdot n_{\mathrm{f}} + \sigma_{\mathrm{s}} \cdot n_{\mathrm{s}} = 0 \tag{1.2}$$

where u is the velocity and σ is the stress, respectively of the fluid (f) and the structure (s).

These conditions are the key ingredients when deriving the energy equality of the continuous fluid-structure system. When relations (1.1) and (1.2) are satisfied after time discretization, one says that the method is strongly coupled. A monolithic method is typically strongly coupled and, hopefully, is stable in the energy norm.
A straightforward way to satisfy the discrete counterpart of (1.1) and (1.2) is to simultaneously solve the fluid and the structure problems in a unique solver with a monolithic approach, as already explained in the previous section. However, this approach needs ad hoc software development and results in a global solver

which is less modular than two distinct fluid and structure solvers. In particular it is difficult to devise efficient global preconditioners, and to maintain state-of-the-art schemes in each solver. With the partitioned method, the fluid and the structure are solved with their own software and this increases the capabilities of evolution and optimization of each code. Among the partitioned schemes, we can distinguish the weakly coupled ones from the strongly coupled. A scheme is said to be weakly (or loosely) coupled when (1.1) and (1.2) are not exactly satisfied at each time step. When sub-iterations are performed at each time step, the transmission conditions (1.1) and (1.2) can be enforced with a high accuracy even though two different solvers are used and so the method is not weakly coupled. Nevertheless, partitioned procedures are often used to implement weakly coupled schemes. Indeed, many fluid-structure interaction problems, in particular in aeroelasticity reference, can be solved in practice without enforcing exactly (1.1) and (1.2), [7].

Chapter 2

Mathematical Model

With computational models playing an ever increasing role in the advancement of science, it is important to understand what does it mean to model something; recognize the implications of the conceptual, mathematical and algorithmic steps of model construction; and comprehend what models can and cannot do. Models cannot replace experiments nor can they prove that particular mechanisms are at work in a given situation. But they can demonstrate whether or not a proposed mechanism is sufficient to produce an observed phenomenon.
A mathematical model is a quantitative representation of a natural phenomenon. Like all other models used in science, its aim is to represent as incisively as possible a given object, a real phenomenon or a set of phenomena (mathematical model of a physical system, chemical system or biological system). Often the model is a representation of the reality not perfect, but however faithful.

Mathematical modeling is indispensable in many applications, is successful in many further applications, it gives precision and direction for problem solution and it enables a thorough understanding of the system modeled. Moreover it prepares the way for better design or control of a system and allows the efficient

use of modern computing capabilities.
Computers can be used to perform numerical calculations. There is a large element of compromise in mathematical modelling. However the majority of systems in the real world are far too complicated to be model in their full complexity.

2.1 The fluid model: Navier-Stokes equations

In fluid dynamics, the equations of Navier-Stokes constitute a system of non-linear partial derivative differential equations (PDEs) that describe the behaviour of a viscous fluid at a macroscopic level. The fluid is considered as a tightly deformable continuous [3].

The Navier-Stokes equations can be derived from the basic conservation of momentum and continuity equations applied to properties of fluids.
They arise from the application of Newton's second law to fluid motion, together with the assumption that the stress in the fluid is the sum of a diffusing viscous term (proportional to the gradient of velocity) and a pressure term. They result in a system of nonlinear partial differential equations; however, with certain simplifications (such as 1-dimensional motion) they can sometimes be reduced to linear differential equations. Nonlinearity in particular makes them difficult or impossible to solve; this is what causes the turbulence and unpredictability in their solutions.

We make the assumption that mass is not added or removed from the system.
As result, continuity equation turn out to be:

$$\frac{d\rho}{dt} + \nabla \cdot (\rho \mathbf{u}) = 0$$

For an incompressible fluid, the density is constant. Setting the derivative of density equal to zero and dividing through by a constant ρ, we obtain the simplest form of the **continuity equation**:

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

Let us now consider Newton's second law with the **momentum conservation** expressed for an open system with several inlets and outlets like the one illustrated in the picture below:

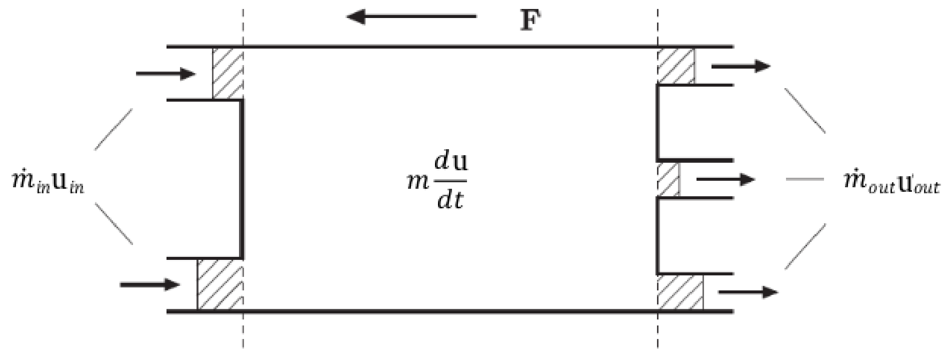

F
$\dot{m}_{in}\mathbf{u}_{in}$
$m\frac{d\mathbf{u}}{dt}$
$\dot{m}_{out}\mathbf{u}_{out}$

Figure 2.1: Navier-Stokes domain: inlet and outlet

$$\mathbf{F} = m\frac{d\mathbf{u}}{dt} + \sum m_{out}\mathbf{u}_{out} - \sum m_{in}\mathbf{u}_{in}$$

$$m\frac{d\mathbf{u}}{dt} = \sum m_{in}\mathbf{u}_{in} - \sum m_{out}\mathbf{u}_{out} + \mathbf{F}$$

where:

$$m\frac{d\mathbf{u}}{dt} = m\left[\frac{1}{m}\int_V \frac{\partial}{\partial t}(\rho\mathbf{u})dV\right] = \int_V \frac{\partial}{\partial t}(\rho\mathbf{u})dV$$

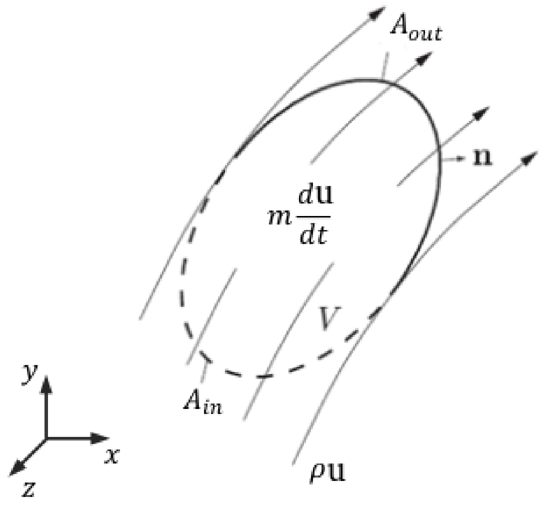


Figure 2.2: Navier-Stokes domain: forces inlet and outlet

$$\sum m_{in}\mathbf{u}_{in} - \sum m\mathbf{u}_{out} = -\int_A \mathbf{u}(\rho\mathbf{u}\cdot\mathbf{n})dA = -\int_V \nabla\cdot(\rho\mathbf{u}\mathbf{u})dV$$

$$\mathbf{F} = \int_V \rho\mathbf{g}dV - \int_A \mathbf{n}dA + \int_A \mathbf{n}\cdot\mathbf{T}dA$$

with:

- $\mathbf{g}$ = gravity force
- p = pressure
- $\mathbf{n}$ unit vector normal to the surface
- $\mathbf{T}$ viscous stress tensor

Since

$$\int_A p\mathbf{n}dA = \int_V \nabla p dV$$

and

$$\int_A \mathbf{n} \cdot \mathbf{T} dA = \int_V \nabla\ \mathbf{T} dV$$

$$\mathbf{F} = \int_V \rho\mathbf{g}dV - \int_V \nabla p dV + \int_V \nabla \cdot \mathbf{T} dV = \int_V (\rho\mathbf{g} - \nabla p + \nabla \cdot \mathbf{T}) dV$$

$$\int_V \left[\frac{\partial}{\partial t}(\rho\mathbf{u}) + \nabla \cdot (\rho\mathbf{u}\mathbf{u}) + \rho\mathbf{g} - \nabla p + \nabla \cdot \mathbf{T}\right] dV = 0$$

This derivation gives the equation for conservation of momentum, that is:

$$\frac{\partial}{\partial t}(\rho u) + \nabla \cdot (\rho u u) + \rho g - \nabla p + \nabla \cdot T = 0$$

In case of incompressible and Newtonian fluid, the constitutive stress/strain law is:

$$\mathbf{T} = \mu[\nabla\mathbf{u} + (\nabla\mathbf{u})^T]$$

$$\text{with } \nabla\ \mathbf{u} = \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_y}{\partial x} & \frac{\partial u_z}{\partial x} \\ \frac{\partial u_x}{\partial y} & \frac{\partial u_y}{\partial y} & \frac{\partial u_z}{\partial y} \\ \frac{\partial u_x}{\partial z} & \frac{\partial u_y}{\partial z} & \frac{\partial u_z}{\partial z} \end{bmatrix}$$

and μ is the dynamic viscosity

We consider an incompressible and Newtonian fluid so we can rewrite the equations in a more compact way:

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

$$\mu = constant$$

$$\nabla \cdot (\nabla \mathbf{u})^T = \nabla(\nabla \cdot \mathbf{u}) = 0$$

$$\nabla \cdot \mu[\nabla \mathbf{u} + (\nabla \mathbf{u})^T] = \nabla \cdot (\mu \nabla \mathbf{u}) = \mu \nabla^2 \mathbf{u}$$

$$\rho\left[\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\right] = -\nabla p + \mu \nabla^2 \mathbf{u} + \rho \mathbf{g}$$

- Acceleration term: change of velocity with time
- Convective term: force exerted by a particle of fluid by the other particles of fluid surrounding it
- Pressure term: fluid flow in the direction of the largest change in pressure
- Diffusion term: this term describes how fluid motion is damped
- Body force term: external forces that act on fluids, such as the gravitational force

Navier-Stokes equations can be used to determine the velocity vector field that applies to a fluid, given some initial conditions. The solution is a vector field to every point in a fluid, at any moment in a time interval. It is usually studied in three spatial dimensions, although the two (spatial) dimensional case is often useful as a model, and higher-dimensional analogues are of both pure and applied mathematical interest. Once the velocity field is calculated, other quantities of interest such as pressure or temperature may be found using dynamical equations and relations.

Navier–Stokes equations are useful because they describe the physics of many phenomena of scientific and engineering interest. They may be used to model the weather, ocean currents, water flow in a pipe and air flow around a wing. The Navier Stokes equations, in their full and simplified forms, help with the design of aircraft and cars, the study of blood flow, the design of power stations, the analysis of pollution, and many other things.

2.1.1 Stokes

The equation of motion for Stokes flow can be obtained by linearizing the steady state Navier-Stokes equations. The inertial forces are assumed to be negligible in comparison to the viscous forces, and eliminating the inertial terms of the momentum balance in the Navier–Stokes equations reduces it to the momentum balance in the Stokes equations:

$$\begin{cases} \mu\nabla^2\mathbf{u} - \nabla p + \mathbf{f} = \mathbf{0} \\ \nabla\mathbf{u} = 0 \end{cases}$$

where ρ is the fluid density and u the fluid velocity. To obtain the equations of motion for incompressible flow, it is assumed that the density, ρ , is a constant. Furthermore, we can consider the unsteady Stokes equations:

$$\rho\frac{\partial\mathbf{u}}{\partial t} + \mu\nabla^2 u - \nabla p + f = 0$$

The Stokes equations represent a considerable simplification of the full Navier-Stokes equations, especially in the incompressible Newtonian case. They are the leading-order simplification of the full Navier-Stokes equations, valid in the distinguished limit $\mathrm{Re} \to 0$.

A Stokes flow has no dependence on time other than through time-dependent boundary conditions. This means that, given the boundary conditions of a Stokes flow, the flow can be found without knowledge of the flow at any other time.

The resulting equations are linear in velocity and pressure, and therefore can take advantage of a variety of linear differential equation solvers.

2.2 Lagrange Multiplier

Another important mathematical tool that we will use is the Lagrange Multiplier method, because it allows us to enforce constraints, such as boundary conditions, at the level of the weak formulation.

In general, Lagrange multiplier method allows us to maximize or minimize functions with the constraint that we only consider points on a certain surface. To find critical points of a function $f(x, y, z)$ on a level surface $g(x, y, z) = C$ (or subject to the constraint $g(x, y, z) = C$), we must solve the following system of simultaneous equations:

$$\nabla f(x, y, z) = \lambda \nabla g(x, y, z) = 0$$

we can write this as a collection of four equations in the four unknowns x, y, z, and λ:

$$\begin{cases} f_x(x,y,z) = \lambda g_x(x,y,z) \\ f_y(x,y,z) = \lambda g_y(x,y,z) \\ f_z(x,y,z) = \lambda g_z(x,y,z) \\ g(x,y,z) = 0 \end{cases}$$

The variable λ is called a Lagrange multiplier.

In finite element methods, the Dirichlet boundary conditions can be enforced by using Lagrange multipliers and by solving related saddle point problems. By combining both discretization methods it is possible to profit from the advantages of both methods. Therefore, the Lagrange multiplier method provides a necessary but not sufficient condition for optimization in constrained problems.

Now let's formulate the problem more rigorously from a mathematical point of view:

Let A be an open set in $\mathbb{R}^{p+q}$ and a $\in$ A. Let f:A $\to \mathbb{R}$ and $g_1, ..., g_p$:A $\to \mathbb{R}$ with $g_1, ..., g_p \in \mathcal{C}^1$ (A) (functions with continuous partial derivative functions) such that the range of the Jacobian matrix

$$\left(\frac{\partial g_i}{\partial x_j}(a)\right) \quad \text{for} \quad i = 1, \ldots, p \quad \textit{and} \quad j = 1, \ldots, p+q$$

is p. Let S = x $\in$ A / $g_i(x)$ = 0, i=1,...,p and let's suppose that a $\in$ S. It is said that the function f has a relative extremum conditioned by the equations $g_i(x_1, \ldots, x_p, x_{p+1}, \ldots, x_{p+q}) = 0$ for i = $1, \ldots, p$ when there exists a neighborhood U of a in $\mathbb{R}^{p+q}$ such that $f(x) \leq f(a)$ or $f(x) \geq f(a)$ $\forall x \in S \cap U$. In the first case a is a maximum relative conditioned, and in the second case a is a minimum relative conditioned.
Notice that saying relative is similar to local and that saying conditioned is because of ligature conditions (constraints of g_i).

Theorem 1 *Lagrange Multipliers Under the conditions of the establishment of the problem above, if in addition $f \in \mathcal{C}^1(A)$, then for the function f to have an extremum relative conditioned at the point a, it's necessary to exist p real numbers $\lambda_1, \ldots, \lambda_p$ such that the function $L = f + \lambda_1 g_1 + \cdots + \lambda_p g_p$ fulfills dL(a)=0. ($\lambda_1, ..., \lambda_p$ are called Lagrange multipliers.)*
Let's suppose furthermore that $g_i, f \in \mathcal{C}^2(A)$ $\;i = 1, ..., p$ and at the point $a \in A$, is verified that dL(a)=0. Then for that f to have at a a minimum (respectively maximum) relative conditioned is sufficient $d^2L(a)(h,h) > 0 \forall h \in \mathbb{R}^{p+q}$ with $h \neq 0$ resp. $<$ 0) and $dg_i(a) = 0, i = 1, ..., p$.

There is another important apllication of the Lagrange Multiplier, deepened in several studies ad exemples in [11], [12], [5], : when it is used to impose the boundary conditions for PDEs and now we will go deeper in this explanation.

There are many situations where the boundary conditions are in fact constraint relations, for example where a point on the boundary is required to follow a prescribed path or where there is some form of cyclic continuity in the problem. These types of condition are difficult to impose on a problem when direct solution methods are used.
The method of Lagrange mulipliers can help in many cases. It has been shown that it is not computationally easy to handle the Dirichlet (essential) boundary condition if the variational principle requires fulfillment of these conditions. All Dirichlet type boundary conditions can be imposed through the use of Lagrange multipliers and it will be used to illustrate the method.

To explain the Lagrange Multiplier method for imposing boundary conditions for PDEs we consider a saddle point variational formulation which is equivalent to the variational problem: given the bilinear form $a(\cdot,\cdot) \quad V \times V \to \mathcal{R}$, given $f \in V'$ find $u \in V$ such that

$$a(u,v) = \langle f, v \rangle_\Omega \qquad \forall v \in V \tag{2.1}$$

This problem is chosen only for notational simplicity; the following statements are also valid for other second-order elliptic problems such as, e.g., the equations of linear elasticity and Stokes problem.

First of all we present here some recalls.

Let $\Omega \in \mathcal{R}^d$ (d = 2, 3) be a bounded and simply connected domain with sufficiently smooth boundary $\Gamma = \partial\Omega$, and n(x) is the exterior unit normal vector which is defined almost everywhere for $x \in \Gamma$. The coefficient functions a_{ji} (x) are assumed to be sufficient smooth satisfying $a_{ij}(x) = a_{ji}(x) \ \ for \ \ all \ \ i, j = 1, ..., d, x \in \Omega$. The interior boundary trace of a given function f(x), x ∈ Ω is:

$$\gamma_0^{int} f(x) := \lim_{\Omega \ni \tilde{x} \to x \in \Gamma} f(\tilde{x}) \qquad for \quad x \in \Gamma = \partial\Omega$$

The associated conormal derivative is:

$$\gamma_1^{int} u(x) = \lim_{\Omega \ni \tilde{x} \to x \in \Gamma} \sum_{i,j=1}^{d} n_j(x) a_{ji}(\tilde{x}) \frac{\partial}{\partial \tilde{x}_i} u(\tilde{x}) \quad for \quad x \in \Gamma = \partial\Omega \tag{2.2}$$

The scalar partial differential operator is:

$$(Lu)(x) = -\sum_{i,j=1}^{d} \frac{\partial}{\partial x_j} [a_{ji}(x) \frac{\partial}{\partial x_i} u(x)] \quad for \quad x \in \Omega \subset \mathcal{R}^d \tag{2.3}$$

We aim to solve the following problem:

$$\begin{aligned} Lu = f \qquad on \qquad \Omega \\ u = g \qquad on \qquad \Gamma \end{aligned} \tag{2.4}$$

And the Green's fist formula:

$$a(u, v) = \int_\Omega (Lu)(x) v(x) dx + \int_\Gamma \gamma_1^{int} u(x) \gamma_0^{int} v(x) ds_x \tag{2.5}$$

The Dirichlet boundary conditions are now formulated as side conditions, and the associated conormal derivative corresponds to the Lagrange multiplier. If we start from Green's first formula we obtain by introducing the Lagrange multiplier $\lambda := \gamma^{int} u \in H^{-\frac{1}{2}}(\Gamma)$ the following saddle point problem:

Find $(u, \lambda) \in$H$^1(\Omega)$x$H^{-\frac{1}{2}}(\Gamma)$ such that

$$\begin{aligned} a(u,v) - b(v,\lambda) &= \langle f, v\rangle_\Omega \\ b(u,\mu) &= \langle g, \mu\rangle_\Gamma \end{aligned} \tag{2.6}$$

is satisfied for all $(v, \mu) \in H^1(\Omega) \times H^{\frac{1}{2}}(\Gamma)$. Here we have used the bilinear form

$$b(v,\mu) := \langle \gamma_0^{int} v, \mu\rangle_\Gamma \quad for \quad (v,\mu) \in H^1(\Omega) \times H^{\frac{1}{2}}(\Gamma).$$

Let's now investigate the unique solvability of the saddle point problem (2.6), in order to do that we have to report and apply the following theorem.

Theorem 2 *Let X and Π be Banach spaces and let $A : X \to X$' and $B : X \to \Pi'$ be bounded operators. Further, we assume that A is $V_0\breve{}$elliptic,*

$$\langle Av, v\rangle \geq c_1^A ||v||_X^2 \quad for \quad all \quad v \in V_0 = kerB \tag{2.7}$$

and that the stability condition

$$c_s ||q||_\Pi \leq \sup_{0 \neq v \in X} \frac{\langle Bv, q\rangle}{||v||}_X \quad for \quad all \quad q \in \Pi \tag{2.8}$$

is satisfied.
For $g \in Im_X B$ and $f \in Im_{V_g} A$ there exists a unique solution (u, p) $\in X \times \Pi$ of the variational problem

$$\begin{aligned} \langle au, v\rangle + \langle Bv, p\rangle &= \langle f, v\rangle \\ \langle Bu, q\rangle &= \langle g, q\rangle \end{aligned} \tag{2.9}$$

where p is a Lagrange Multipler $p \in \Pi$ and (u,v) $\in X \times \Pi$ satisfying

$$||u||_X \leq \frac{1}{c_1^A}||f||_{X'} + \left(1 + \frac{c_2^A}{c_1^A}\right) c_B ||g||_{\Pi'}, \tag{2.10}$$

with c_i^A is positive constant and

$$||p||_\Pi \leq \frac{1}{c_s}\left(1 + \frac{c_2^A}{c_1^A}\right)\left\{||f||_{X'} + c_B c_2^A ||g||_{\Pi'}\right\}. \tag{2.11}$$

Since,

$$kerB := v \in H^1(\Omega) : \left\langle \gamma_0^{int} v, \mu \right\rangle_\Gamma = 0 \quad for\ \ all \quad \mu \in H^{-\frac{1}{2}}(\Gamma) = H_0^1(\Gamma). \tag{2.12}$$

we have the ker B-ellipticity of the bilinear form a(·,·). So we need to establish the stability condition:

$$c_s ||\mu||_{H^{-\frac{1}{2}}(\Gamma)} \leq \sup_{0 \neq v \in H^1(\Omega)} \frac{\langle \gamma_0^{int} v, \mu \rangle_\Gamma}{||v||_{H^1(\Omega)}} \quad for \quad all \quad \mu \in H^{-\frac{1}{2}}(\Gamma). \tag{2.13}$$

Lemma 1: The stability condition (2.13) is satisfied for all $\mu \in \mathrm{H}^{-\frac{1}{2}}(\Gamma)$.
Finally we can conclude the unique solvability of the saddle point problem (2.6) due to Theorem 2.
Recall that the bilinear form a(·,·) in the saddle point formulation (2.6) is only H_0^1-elliptic. However, it can be reformulated to obtain a formulation where the modified bilinear form $\tilde{a}(\cdot,\cdot)$ is now H^1-elliptic.
Since the Lagrange multiplier $\lambda := \gamma_1^{int} u \in H^{-\frac{1}{2}}(\Gamma)$ describes the conormal derivative of the solution u, using the orthogonality relation we have

$$\int_\Omega f(x)dx + \int_\Gamma \lambda(x) ds_x = 0 \tag{2.14}$$

And with the Dirichlet boundary condition $\gamma_0^{int} u = g$ we also have

$$\int_\Gamma \gamma_0^{int} u(x) ds_x = \int_\Gamma g(x) ds_x. \tag{2.15}$$

Hence we can reformulate the saddle point problem (2.6) to find (u, λ) $\in H^1(\Omega)$ $\times$ $H^{-\frac{1}{2}}$ such that

$$\begin{aligned} \int_\Gamma \gamma_0^{int} u(x) ds_x \int_\Gamma \gamma_0^{int} v(x) ds_x + a(u,v) - b(v,\lambda) = \\ \langle f, v\rangle_\Omega + \int_\Gamma g(x) ds_x \int_\Gamma \gamma_0^{int} v(x) ds_x \end{aligned} \tag{2.16}$$

$$b(u,\mu) + \int_\Gamma \lambda(x) ds_x \int_\Gamma \mu(x) ds_x = \langle g, \mu\rangle_\Gamma - \int_\Omega f(x) dx \int_\Gamma \mu(x) ds_x \tag{2.17}$$

is satisfied for all (v, μ) *in* $H^1(\Omega)$x $H^{-\frac{1}{2}}(\Gamma)$.

The modified saddle point problem (2.16) and (2.17) is uniquely solvable, and the solution is also the unique solution of the original saddle point problem (2.6), i.e. the saddle point formulations (2.16)–(2.17) and (2.6) are equivalent.

2.3 Mathematical formulation of the reduced FSI Problem

Since we are dealing with a fluid structure problem, we look at it as one system of equations for the fluid, one for the solid and one for the coupling conditions that link the other two.
We consider the incompressible flow around a rigid obstacle. The obstacle may be force to move according to a linear elastic constraint.

To bluid our mathematical model we considered first the system of equations for the fluid part:

$$\begin{cases} -\partial_{\mathrm{t}} u - \Delta u + \nabla p = 0 & \text{in } \Omega \\ divu = 0 & \text{in } \Omega \\ \sigma(u,p)\cdot \mathbf{n} = \hat{p} & \text{on } \Gamma_{\mathrm{p}} \\ u = 0 & \text{on } \Gamma_{\mathrm{u}} \end{cases}$$

In particular, we consider here a simplified version of the classical FSI problem involving a rigid body immersed into a flow.
Precisely, we assume that the rigid body is a disc with a small radius with respect to the characteristic dimension of the domain Ω. For this reason, calling F the total force induced by the fluid on the solid the equations of motion of the body are the following:

$$\begin{cases} m\ddot{d} + kd = F & \text{on } \Sigma \\ \ddot{d} = \partial^2 d \end{cases}$$

Finally, we have to enforce the coupling conditions between the fluid and the solid models. Since the object is small, the velocity of the points on Γ can be identified with the velocity of its center of mass, namely the point $\underline{x}$. Then, the kinematic condition may be written as $\underline{u}(\underline{x}) = \dot{\underline{d}}$. However, this condition is only partially satisfactory because it does not take into account of the distribution of the velocity around the object. Then, we decide to replace the pointwise evaluation of the fluid velocity at the center of mass with the average of the fluid velocity on the fluid-solid interface. In this way we get:

$$\frac{1}{|\Sigma|}\int_{\Sigma} u = \dot{d}$$

We proceed similarly for the equilibrium of forces. Precisely the force induced by the fluid on the object is:

$$F = -\frac{1}{|\Sigma|}\int \sigma(u,p)\cdot\mathbf{n}$$

Finally this is the system of <u>coupling conditions</u>:

$$\begin{cases} \frac{1}{|\Sigma|}\int_{\Sigma} u = \dot{d} & \text{on } \Sigma \\ F = -\frac{1}{|\Sigma|}\int \sigma(u,p)\cdot\mathbf{n} = L \end{cases}$$

Considering the **average technique** we equalize the velocity of the center of the disk, $\dot{d}$, and the average of the speed of the flow along the disk, $\frac{1}{|\Sigma|}\int_{\Sigma} u$. In this way we have a reduced problem.

Using the Lagrange Multiplier method to enforce the kinematic and dynamic constraints between the solid and the fluid, we obtain the following weak formulation of the reduced FSI Problem:

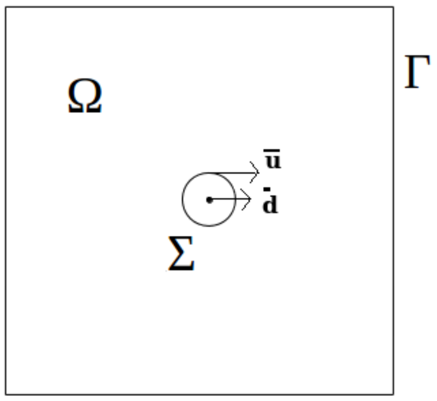


Figure 2.3: Domain 2D-0D

$$(\partial_t u, v) + (\nabla u, \nabla v) - (p, divv) + (q, divu) + L(\frac{1}{|\Sigma|}\int_\Gamma v \ - \xi) +$$

$$+ \ (\mathrm{m} + \mathrm{kd})\xi + M(\tfrac{1}{|\Sigma|}\textstyle\int_\Gamma u - \dot{d}) = 0 \qquad (2.18)$$

$$\forall v \in H^1(\Omega)$$
$$\forall q \in L^2(\Omega)$$
$$\forall L, M, \dot{d} \in \mathbb{R}^2$$

The term $\frac{1}{|\Sigma|}\int_\Gamma u \in H^{\frac{1}{2}}$ so it can be integrated and the integral is a finite number.

Chapter 3

Numerical discretization

Mathematical modeling is the discipline of representing a physical phenomenon in mathematical terms. Most of these models (equations of linear elasticity, Navier-Stokes equations of fluid mechanics, Maxwell equations of electromagnetism, etc...) are written as a partial differential equation (PDE) or a system of PDE defined on a suitable domain of Rn, n= 1,2,3. Boundary (and possibly initial) conditions complete the model. For the numerical discretization of such problem we use the finite element method, thoroughly explained ad exemple in [10].

The finite element method approssimation of the problem results in a system of algebraic equations. The method approximates the unknown function over the domain. To solve the problem, it subdivides a large system into smaller, simpler parts that are called finite elements. The simple equations that model these finite elements are then assembled into a larger system of equations that models the entire problem. FEM then uses variational methods from the calculus of variations to approximate a solution by minimizing an associated error function.

We use a finite element discretization with a non uniform partition $\mathrm{T^h}_\Omega$ of Ω of triangular element, where h is the characteristic size.

Given an admissible triangulation T_h^Ω the finite element space is defined as follows:

$$X_h^k(\Omega) = \left\{ v_h \in \mathcal{C}(\Omega) : v|_k \in \mathcal{P}^k(k) \forall k \in T_h^\Omega \right\}$$

We use a P1/P1 for the fluid discretization. More precisely, we use equal order $X_h^1(\Omega)$ approximation for velocity and pressure
In certain circumstances, most often for implementation reasons, it would be extremely useful to be able to work with equal order finite-elements, which unfortunately are not stable. Increasing the polynomial degree would result in additional computational costs and may be not the optimal strategy if the solution is not regular enough. Stabilization strategies have been proposed in the literature to make it possible to work with equal order finite-elements.
We consider a stabilization term (to be added to the standard Galerkin formulation) defined, in general, as a symmetric positive bilinear form on Q_h, denoted as

$$(s, p) : Q_\mathrm{h} \times Q_\mathrm{h} \to R$$

$$s(p_\mathrm{h}, q_\mathrm{h}) = \gamma \sum_{K \in \mathcal{T}_\mathrm{h}} h_K^2 \int_K \nabla p_h \cdot \nabla q_h d\Omega$$

- γ = stabilization coefficient
- $T_\mathrm{h}{}^\Omega$ = computational grid
- h_K^2 = area of element K
- uniformly stable for any equal order space pair
- guarantees only a linear convergence rate

Below it is rewritten the equation (2.18) in the discretization form with the Brezzi-Pitkaranta stabilization term.

$$\begin{aligned}(\partial_{\mathrm{t}} u_{\mathrm{h}}, v_{\mathrm{h}}) + (\nabla u_{\mathrm{h}}, \nabla v_{\mathrm{h}}) - (p_{\mathrm{h}}, div v_{\mathrm{h}}) + (q_{\mathrm{h}}, div u_{\mathrm{h}}) - \gamma(h^2 \nabla p_{\mathrm{h}}, \nabla q_{\mathrm{h}}) + \\ L(\frac{1}{|\Sigma|}\int_{\Gamma} v_{\mathrm{h}} - \xi_{\mathrm{h}}) + (m\ddot{d}_{\mathrm{h}} + kd_{\mathrm{h}})\xi_h + M(\frac{1}{|\Sigma|}\int_{\Gamma} u_{\mathrm{h}} - \dot{d}_{\mathrm{h}}) = 0\end{aligned} \tag{3.1}$$

We write the algebraic formulation in matrix form in order to compute the numerical solution.

We choose to use FreeFEM++ as software, since it is as a popular 2D and 3D partial differential equations (PDE) solver.
FreeFEM is a partial differential equation solver for non-linear multi-physics systems in 2D and 3D. Problems involving partial differential equations from several branches of physics, such as fluid-structure interactions, require interpolations of data on several meshes and their manipulation within one program.
FreeFEM++ includes a fast interpolation algorithm and a language for the manipulation of data on multiple meshes.
FreeFEM is written in C++ and its language is a C++ idiom.

We build the fluido-structure matrix considering the structure as an obstacle and we obtain the following result:

$$FSI = \begin{bmatrix} F & I \\ I' & Z \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ p \\ L_1 \\ L_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Z is a matrix (2x2) filled with zeros.
- To compute the matrix F we used a transient Stokes solver in matrix form, already present in the Documentation of FreeFem++, [6].
- To compute the matrix I we define a partition of the curve in segments that conform to the mesh of Ω and integrate the basis functions of each element that contains a segment of the parameterized curve on that segment.

After that we add three blocks to the matrix in order to consider also the movement of the structure.

$$FSI = \begin{bmatrix} F & 0 & I \\ 0 & 0 & M \\ I' & L & S \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ p \\ L_1 \\ L_2 \\ d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} \frac{u_1{}^{\text{n-1}}}{\tau} \\ \frac{u_2{}^{\text{n-1}}}{\tau} \\ 0 \\ -\frac{d_1{}^{\text{n-1}}}{\tau} \\ -\frac{d_2{}^{\text{n-1}}}{\tau} \\ m(\frac{2d_1{}^{\text{n-1}} - d_1{}^{\text{n-2}}}{\tau^2}) \\ m(\frac{2d_2{}^{\text{n-1}} - d_2{}^{\text{n-2}}}{\tau^2}) \end{bmatrix}$$

- F = Stokes matrix - for the first group of test cases

  $$(\partial_{\text{t}} u, v) + \nu(\nabla u, \nabla v) - (p, divv) + (q, divu) - \gamma(h^2 \nabla p, \nabla q)$$

  We used a Back Euler descretization for $\partial_t u$.

- F = Navier Stokes matrix - for the second group of test cases

  $$(\partial_{\text{t}} u, v) + \nu(\nabla u, \nabla v) - (p, divv) + (q, divu) - ((u\nabla)u, v) + \gamma\tau_{SUPG/PSPG}(\nabla p + u\nabla u, \nabla q + v\nabla v)$$

where the convective non linear term is descritized iteratively:

$$((u\nabla)u, v) = (u_n\nabla)u_{n-1}, v_n)$$

We update the stabilization term for the new equation, we replace Brezzi-Pitquaranta term with the SUPG/PSPG stabilization, moltipling it with a factor $\gamma = 0.1$.

$$\tau = \frac{1}{(\sqrt{\frac{4}{dt^2} + 4\frac{u_{n-1}^2}{h^2} + \frac{16\mu^2}{h^4}})}$$

- I = Fluid-structure interface matrix

$$(L \cdot \frac{1}{|\Sigma|}\int_\Gamma v)$$

$$(M \cdot \frac{1}{|\Sigma|}\int_\Gamma u)$$

- S = Structure matrix

$$(m\ddot{d} + kd)$$

- L = Lagrange multiplier matrix

$$(L \cdot \xi)$$

- M =

$$(M\dot{d})$$

$$FSI = \begin{bmatrix} F & 0 & I \\ 0 & 0 & M \\ I' & L & S \end{bmatrix} \qquad M = (-\tfrac{1}{\tau})\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad L = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

To compute the block I we have to develop two new algorithms in FreeFem++:

1. Intersection algorithm
2. Gauss interpolation

3.1 Intersection algorithm

The intersection algorithm aims at calculating the intersection between the profile of the obstable and the computational mesh.
As a first step we considered the 2D-0D coupling problem.

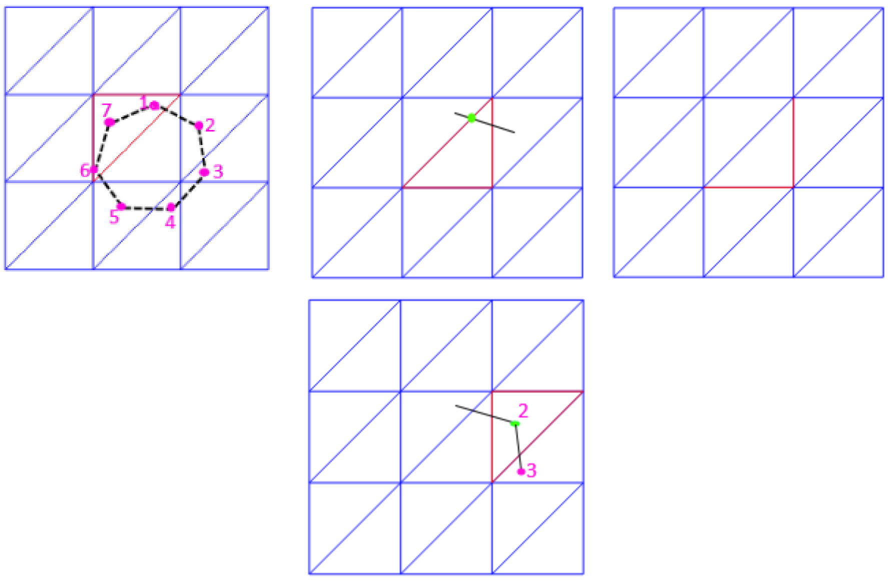
The intersection algorithm is devided in some steps:

1. Given a set of ordered point, take the first and the second to form a segment (ex. point n. 1 and n. 2)

1. Select the triangular element in which is located the first point of the segment (ex. point n. 1 belongs to the red triangle in the first picture)
2. Loop on the three edges of the selected triangle and look for intersection between the segment considered and the edge (ex. green point in the second picture)
3. Move to the adjacent triangle to the edge to which the point of intersection found belongs (ex. red triangle in the second picture)
4. Loop on the two edges of the triangle that have not been considered yet (ex. red edges in the third picture) and find the new intersection point
5. Repete the procedure until the triangle considered is the same as the one to which the second point of the segment belongs (ex. point n. 2 belongs to the red triangle in the second picture). In this way I have identified all the elements that intersect the curve between the points considered

6. Change segment: the first point become the previous second (ex. n. 2) and the second point become the follow one in the ordered set of points (ex. n. 3)

7. Continue until every point has been considered



3.2 Gauss Point interpolation

In numerical analysis, a quadrature rule is an approximation of the definite integral of a function, usually stated as a weighted sum of function values at specified points within the domain of integration. An n-point Gaussian quadrature rule, is a quadrature rule constructed to yield an exact result for polynomials of degree 2n-1 or less by a suitable choice of the nodes x_i and weights ω_i for i = 1, ..., n. The most common domain of integration for such a rule is taken as [1,1], so the rule is stated as

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} \omega_i f(x_i)$$

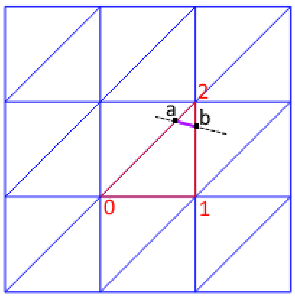
which is exact for polynomials of degree 2n-1 or less. This rule is known as the Gauss-Legendre quadrature rule.

3.2.1 Gauss Point method - Application

Once that all the intersection points are located we can proceed with the integration of the basis function only of the triangular elements involved with the intersection.

$$\int_{-1}^{1} \phi_{\mathrm{k}}(F^{\text{-}1}(\alpha(t)))|\alpha'(t)|dt \simeq \sum_{i=1}^{2} \phi_{\mathrm{k}}(F^{\text{-}1}(\alpha(t_{\mathrm{i}})))|\alpha'(t_{\mathrm{i}})|\omega_{\mathrm{i}}$$

$$F(x) = \begin{bmatrix} (x_2 - x_0) & (x_1 - x_0) \\ (y_2 - y_0) & (y_1 - y_0) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$



- ϕ_k is the basis function of the element k
- $\alpha(t_{\mathrm{i}})$ = traslation from interval [-1,1] to [a,b] $\qquad \alpha(t_{\mathrm{i}}) = \frac{a+b}{2} + \frac{b-a}{2}t_{\mathrm{i}}$
- t_i = Gauss Point
  $t_1 = \sqrt{\frac{1}{3}}$ and $t_2 = -\sqrt{\frac{1}{3}}$
- F(x) = allows to pass from the global to the local reference system , where $x_{0,1,2}$ and $y_{0,1,2}$ represent the coordinates of the vertexes of the triangle that we are considering
- weight $\omega_{\mathrm{i}} = 1$

I has a size of (nv x 2), where nv = number of vertexes of the triangular mesh.

The first column corresponds to the first component of u, u_1, in the equation (3.1), the second column to the second component, u_2, and the third component to p.

The k-row is filled with the integrated basis function of the corresponding element. The rows corresponding to element that do not intersect the parameterised curve are filled with zeros.

$$I : \begin{bmatrix} u_{10} & u_{10} \\ u_{20} & u_{20} \\ p_0 & p_0 \\ \vdots & \vdots \\ u_{1k} & u_{1k} \\ u_{2k} & u_{2k} \\ p_k & p_k \\ \vdots & \vdots \\ u_{1n} & u_{1n} \\ u_{2n} & u_{2n} \\ p_n & p_n \end{bmatrix} \qquad \mathrm{I} = \begin{bmatrix} \vdots & \vdots \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ \int_{-1}^{1} \phi_\mathrm{k} & 0 \\ 0 & \int_{-1}^{1} \phi_\mathrm{k} \\ 0 & 0 \\ \vdots & \vdots \\ \int_{-1}^{1} \phi_\mathrm{i} & 0 \\ 0 & \int_{-1}^{1} \phi_\mathrm{i} \\ 0 & 0 \\ \vdots & \vdots \end{bmatrix}$$

The last step of the algorithm consists in solve the complete matrix of the fluid-structure with the desired boundary conditions.

For the right hand side vector:

$$rhs = \begin{bmatrix} \frac{{u_1}^{\text{n-1}}}{\tau} \\ \frac{{u_2}^{\text{n-1}}}{\tau} \\ 0 \\ \frac{{d_1}^{\text{n-1}}}{\tau} \\ \frac{{d_2}^{\text{n-1}}}{\tau} \\ m(\frac{2{d_1}^{\text{n-1}}-{d_1}^{\text{n-2}}}{\tau^2}) \\ m(\frac{2{d_2}^{\text{n-1}}-{d_2}^{\text{n-2}}}{\tau^2}) \end{bmatrix} \qquad vou = \begin{bmatrix} u_1 \\ u_2 \\ p \\ L_1 \\ L_2 \\ d_1 \\ d_2 \end{bmatrix}$$

In the right hand side vector we have terms coming from the discretization of first and second degree derivatives:

$$\partial_t u = \frac{u^{\text{n}} - u^{\text{n-1}}}{\tau}$$
$$\dot{d} = \frac{d^{\text{n}} - d^{\text{n-1}}}{\tau}$$
$$\ddot{d} = \frac{d^{\text{n}} - 2d^{\text{n-1}} + d^{\text{n-2}}}{\tau}$$

$$FSI = \left(\begin{array}{c|cc|cc}
 & & & \vdots & \vdots \\
 & & & 0 & 0 \\
 & & & 0 & 0 \\
 & & & 0 & 0 \\
 & & & \vdots & \vdots \\
 & & & \int_{-1}^{1} \phi_{\mathrm{k}} & 0 \\
 & & & 0 & \int_{-1}^{1} \phi_{\mathrm{k}} \\
 & & & 0 & 0 \\
 & & & \vdots & \vdots \\
 & & & \int_{-1}^{1} \phi_{\mathrm{i}} & 0 \\
 & & & 0 & \int_{-1}^{1} \phi_{\mathrm{i}} \\
 & & & 0 & 0 \\
 & & & \vdots & \vdots \\
\hline
 & 0 & 0 & -\frac{1}{\tau} & 0 \\
 & 0 & 0 & 0 & -\frac{1}{\tau} \\
\hline
\dots\ 0\ 0\ 0\ \dots\ \int_{-1}^{1} \phi_{\mathrm{k}}\ 0\ 0\ \dots\ \int_{-1}^{1} \phi_{\mathrm{i}}\ 0\ 0\ \dots & -1 & 0 & \frac{m}{\tau^2} + k_1 & 0 \\
\dots\ 0\ 0\ 0\ \dots\ 0\ \int_{-1}^{1} \phi_{\mathrm{k}}\ 0\ \dots\ 0\ \int_{-1}^{1} \phi_{\mathrm{i}}\ 0\ \dots & 0 & -1 & 0 & \frac{m}{\tau^2} + k_2
\end{array}\right)$$

Chapter 4

Test cases for the assessment of the reduced model

To validate our model we proceed now making some test cases through which we analyze quantitatively the reduced model and the numerical discretization.

4.1 Obstacle problem with Stokes flow

Flow around a fixed or oscillating cylinder has received continued attention in the past few decades. In addition to being a building block in the understanding of bluff body dynamics, it has a large number of applications in many engineering situations. This study is the first step to investigate the feasibility of three dimensional coupled fluid structure computations using a one-dimensional model for the structure.

As first example we consider the unit square $\Omega(0,1)^2$ containing a disc with radius R.
Let $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$ be the sides of the square ordered counterclockwise starting from

the base.

The boundary conditions imposed are:

- **Neumann boundary condition** $\frac{\partial u}{\partial n} - p \cdot n = \hat{p} \cdot n$, with $\hat{p} = 1$ **on** Γ_4
- Neumann boundary condition $\frac{\partial u}{\partial n} - p \cdot n = 0$ on Γ_2 (As a result the flow is driven by pressure jump from Γ_4 to Γ_2, which is a unit pressure drop).
- **Dirichlet boundary condition** $u = 0$ **on** Γ_1 **and** Γ_3
- A set of points that parametrize a circonference

To validate the results obtained with the reduced model we compare them with those obtained from a full model, namely a fitted mesh Stokes' algorithm with an empty circle inside. Then we evaluate the model error of the reduced model. The error is calculated as the L^2 norm.

$$L^2 Error = \sqrt{\chi * \frac{(u_{s1} - u_1)^2 + (u_{s2} - u_2)^2}{(u_1^2 + u_2^2)}}$$

$$\chi = \begin{cases} 1 & if \quad (x - x_c)^2 + (y - y_c)^2 > \delta \\ 0 & otherwise \end{cases}$$

- x_c, y_c = coordinates of the center of the circle
- δ = radius of the circle outside which we want to calculate our error
- u_s, u = velocity vectors of the mesh of Stokes algorithm and the fitted mesh, respectively

where x_c and y_c are the coordinates of the center of the circle, δ is the radius of the circle outside which we want to calculate our error (it is necessary compare the error considering a circle bigger than one of the mesh) and ud and u are the velocity vectors of the mesh of Stokes algorithm and the fitted mesh, respectively.

4.1.1 Results of the obstacle problem with Stokes flow

We consider the disc in the center of the domain which is a 1x1 square, so: $x_c = 0.5$, $y_c = 0.5$
To evaluate the error more accurately we compare the two algorithms using a conforming mesh with respect to the fluid-structure interface.

The number of points on each edge and on the inner circle are constant for each mesh: namely 30 and 30 points respectively. Each mesh is illustrated below with the respective results of the velocity field.

ϵ	δ	L2Error
0.1	0.2	0.0755484
0.05	0.2	0.131741
0.025	0.2	0.162833

"Th" is the mesh for our algorithm and "Thd" is the one for FreeFem++ implemented Stokes algorithm. "np" is the number of points for the edge, "ic" is the number of points for the internal disc and "r" is the radius of the disc.

```
int np = 120;
int ic = 240;
real r = 0.025;
border C01(t=0,1){x = t; y = 0;  label = 1; }
border C02(t=0,1){x = 1; y = t; label = 2; }
border C03(t=0,1){x = 1-t; y = 1;   label = 3; }
border C04(t=0,1){x = 0; y = 1-t;   label = 4; }
border bo(t=0, 2*pi){x=r*cos(t) + 0.5; y=r*sin(t) + 0.5 ; label =
    5; }
```

```
mesh Th = buildmesh(C01(np)+C02(np)+C03(np)+C04(np)+bo(+ic));
mesh Thd = buildmesh(C01(np)+C02(np)+C03(np)+C04(np)+bo(-ic));
```

Since the intersection algorithm do not compile an intersection point when it coincides with a node of the mesh, we move the parametrized circle of 0.00000525, which is the smallest amount that the mesh takes into account, in an arbitrary direction (we chose the positive vertical direction).
FreeFem++ code for the error is:

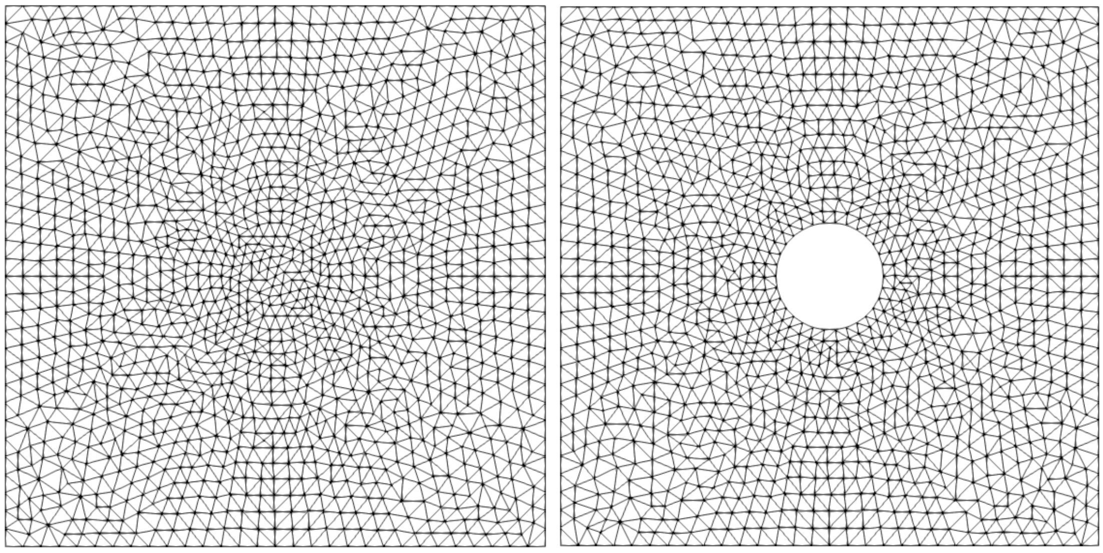
```
real L2error = sqrt(int2d(Thd)((((x-0.5)^2 + (y-0.5)^2) > 0.2)
    *(((u-ud)^2+(v-vd)^2)/(ud^2+vd^2))));
```

As you can see from the following example, expecially the last one, even if the meshes are constructed in the same way and the size of the number of points of each edge is the same, as is the number of points on the disc, the mesh is not the same because of the difference between the full disc and the empty one.

The code for the stokes comparison agorhithm is:

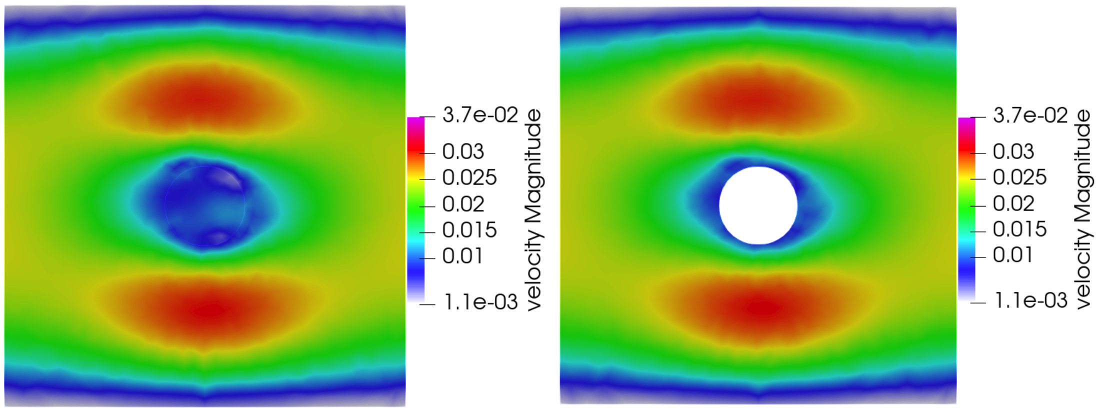
```
fespace Vhd(Thd, P1);
Vhd ud, vd;
Vhd uoldd, voldd;
fespace Qhd(Thd, P1);
Qhd pd;
Qhd ppd;
fespace Xh(Thd, [P1, P1, P1]);
varf aad ([ud, vd, pd], [uud, vvd, ppd])
    = int2d(Thd)(
           (ud*uud+vd*vvd)/dt
          +(dx(ud)*dx(uud) + dy(ud)*dy(uud) + dx(vd)*dx(vvd) + dy(
    vd)*dy(vvd))
```

```
        + gamma*hTriangle^2*(dx(pd)*dx(ppd) + dy(pd)*dy(ppd))
        - pd*(dx(uud) + dy(vvd))
        + ppd*(dx(ud) + dy(vd))
    )
    + int1d(Thd, 1, 3, 5)(10e+20*ud*uud + 10e+20*vd*vvd)
    ;
varf bb ([uoldd, voldd], [uud, vvd])
    = int2d(Thd)(
          ((uoldd*uud+voldd*vvd)/dt)
    )
    ;
varf bcl (unused, uud)
    = int1d(Thd, 4)(1*uud)
    ;
matrix Ad = aad(Xh, Xh, solver = UMFPACK);
matrix Bd = bb(Xh, Xh);
real[int] bd = bcl(0, Xh);
real[int] sold(Xh.ndof), aux(Xh.ndof);
int m, Md = T/dt;
Xh [w1, w2, wp] = [uoldd, voldd, ppd];
sold = w1[];
for (m = 0; m < Md; m++){
    aux = Bd*sold; aux += bd;
    sold = Ad^-1 * aux;
}
w1[]=sold; ud=w1; vd= w2; pd=wp;
cout << " u, v, pw = " << sold << endl;
plot(pd, [ud, vd], value=true, wait=true, cmm="t=" +m*dt, fill=1)
   ;
```



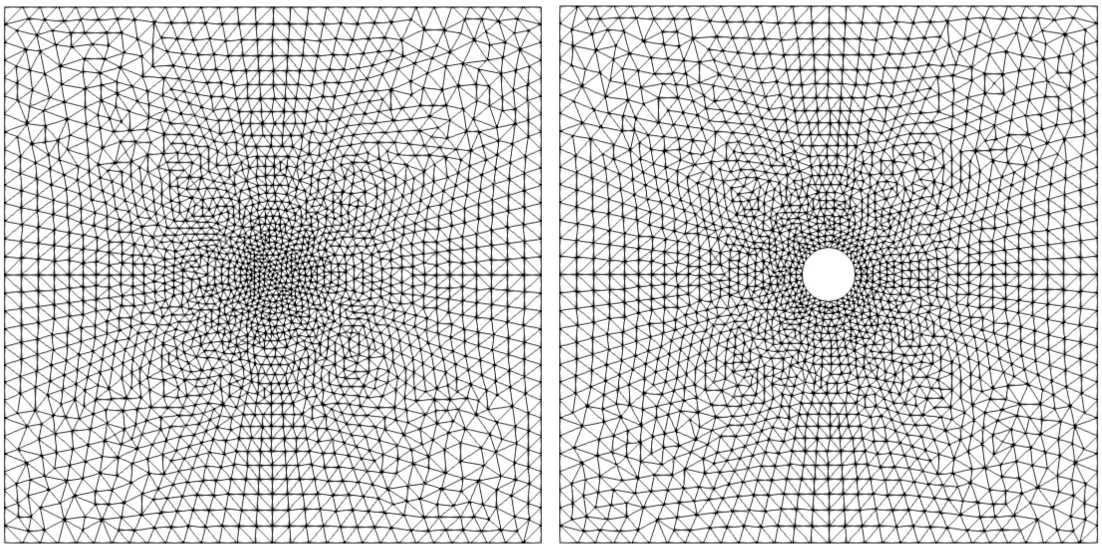
(a) Unfitted Algorithm (b) Fitted Algorithm

Figure 4.1: Mesh: Number of points of the border = 30, Number of points of the inner circle = 30; $\epsilon = 0.1$



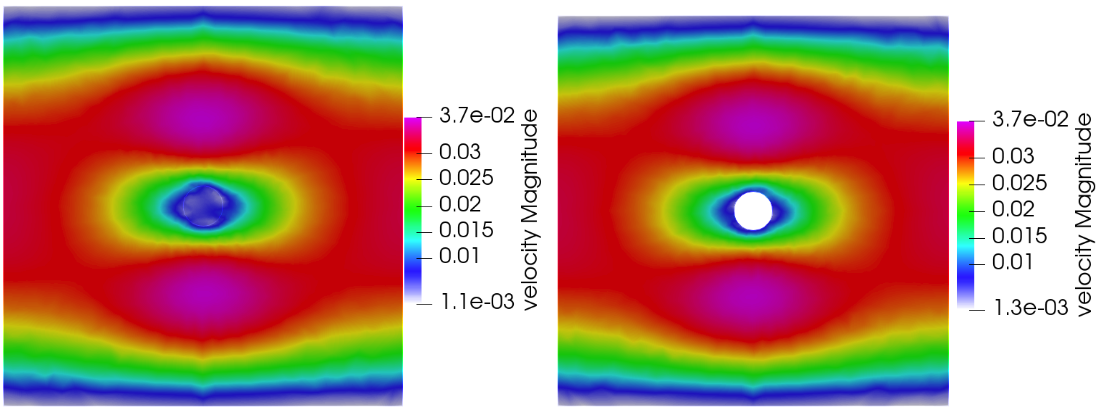
(a) Unfitted Algorithm. (b) Fitted Algorithm.

Figure 4.2: Plot of velocity; Stokes model; $P|_{\Gamma_4} - P|_{\Gamma_2} = 1$; $\epsilon = 0.1$



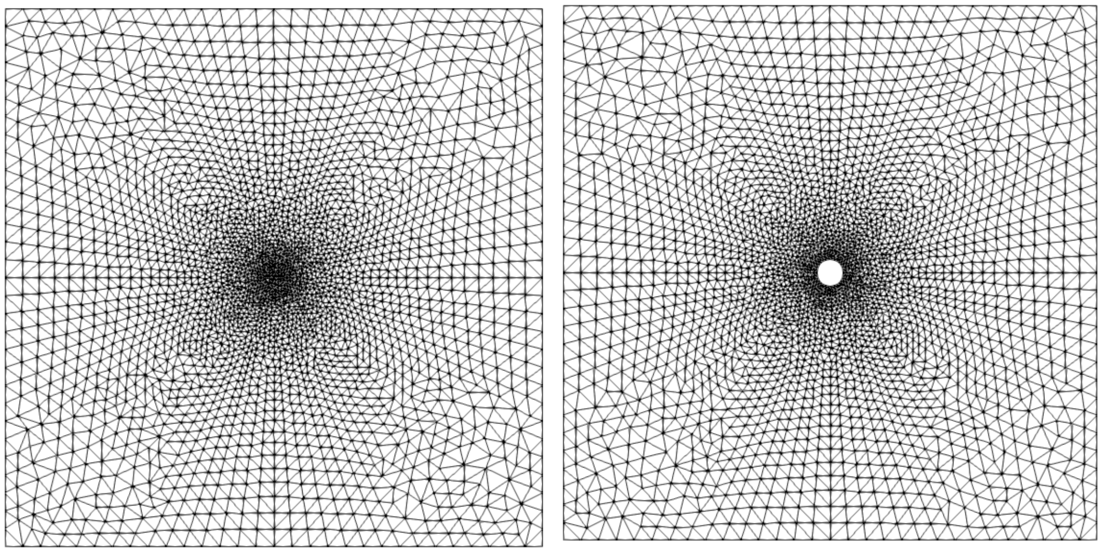
(a) Unfitted Algorithm (b) Fitted Algorithm

Figure 4.3: Mesh: Number of points of the border = 30, Number of points of the inner circle = 30; $\epsilon = 0.05$



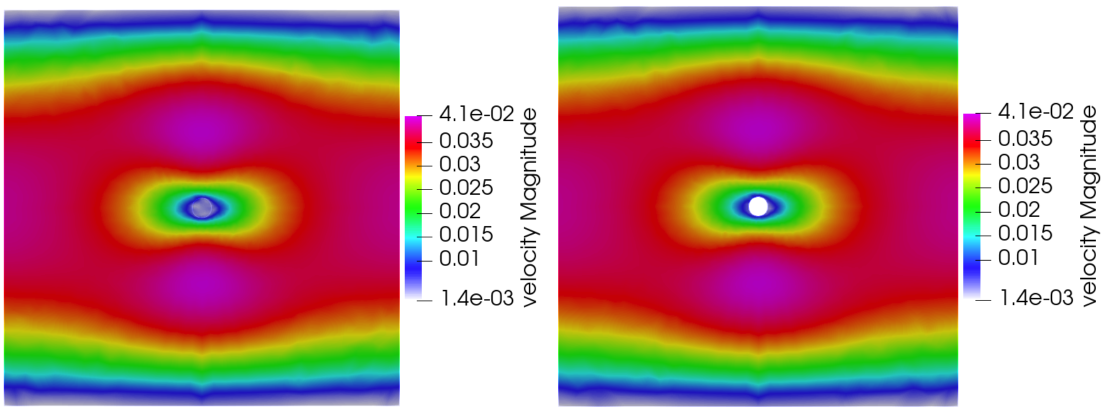
(a) Unfitted Algorithm. (b) Fitted Algorithm.

Figure 4.4: Plot of velocity; Stokes model; $P|_{\Gamma_4} - P|_{\Gamma_2} = 1$; $\epsilon = 0.05$



(a) Unfitted Algorithm (b) Fitted Algorithm

Figure 4.5: Mesh: Number of points of the border = 30, Number of points of the inner circle = 30; $\epsilon = 0.025$



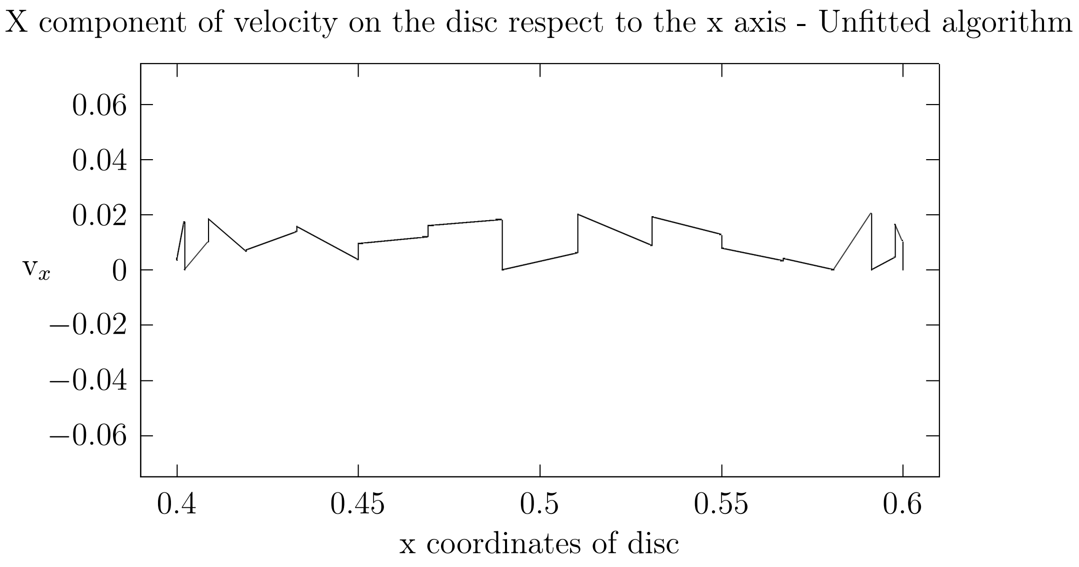
(a) Unfitted Algorithm. (b) Fitted Algorithm.

Figure 4.6: Plot of velocity; Stokes model; $P|_{\Gamma_4} - P|_{\Gamma_2} = 1$; $\epsilon = 0.025$

It is useful report the values of the velocity components on the interface of the disc with the fluid. In this way we can check if values on the disc are small as it appears from the images above.

To this purpose let's create two graphs with the horizontal component of velocity as a function of the x itself, both for the fitted algorithm and for the unfitted one. We chose to represent only the horizontal component since it is more relevant in this case where the only input is an unitary horizontal pressure on the left edge of the square.

The results obtained are satisfactory as we note that they are always very close to zero, for both models, fitted and unfitted, with a maximum value of deviation of the null value of the order of hundredths. These results are a good validation of the unfitted model since it results correctly in a null velocity along the disk, seen as an obstacle.



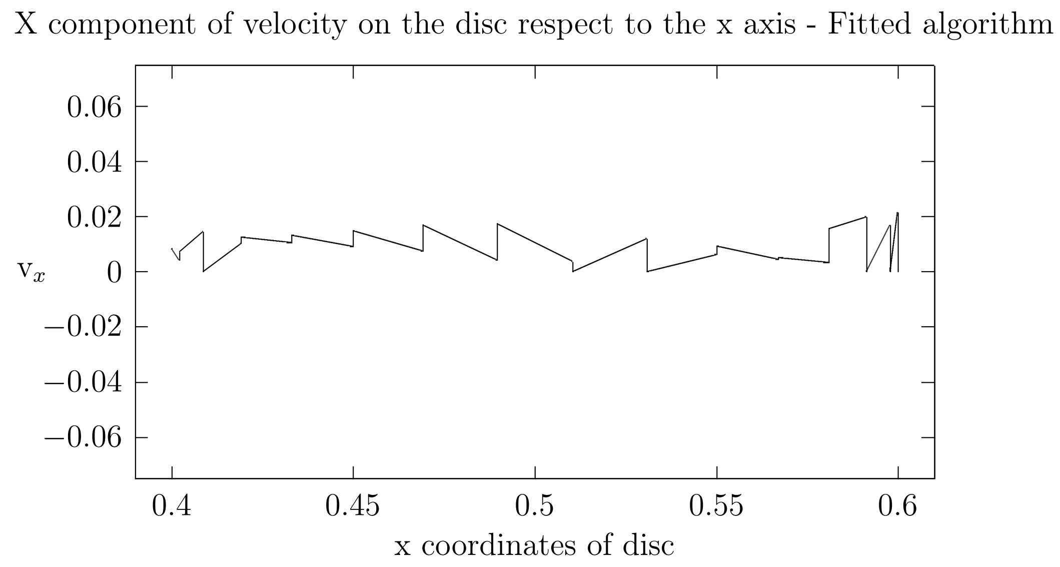
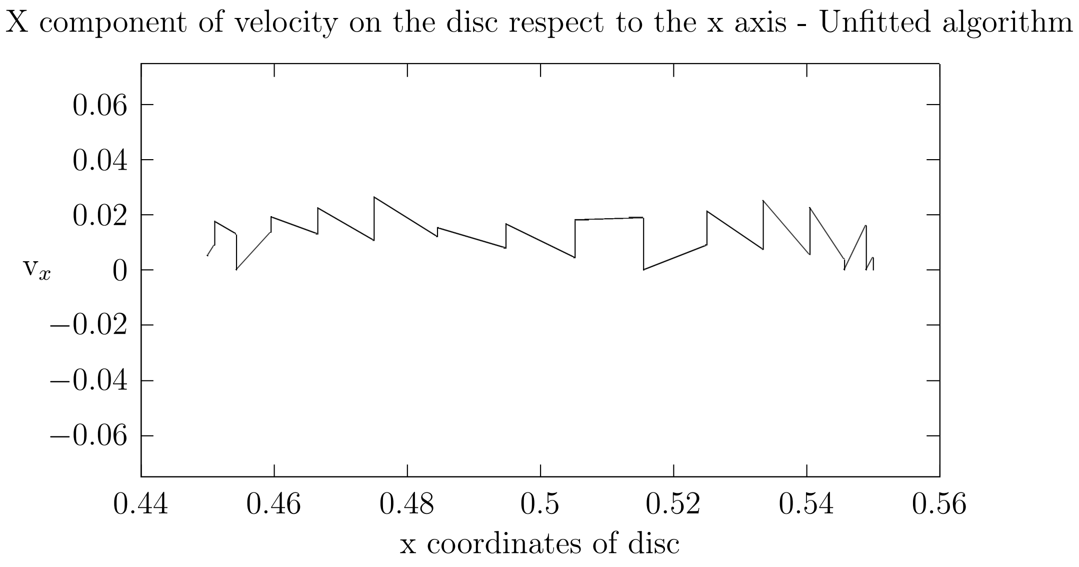


Figure 4.7: Plot of the horizontal component of the velocity along the disc with radius = 0.1; Unfitted (top) and Fitted (bottom) algorithm



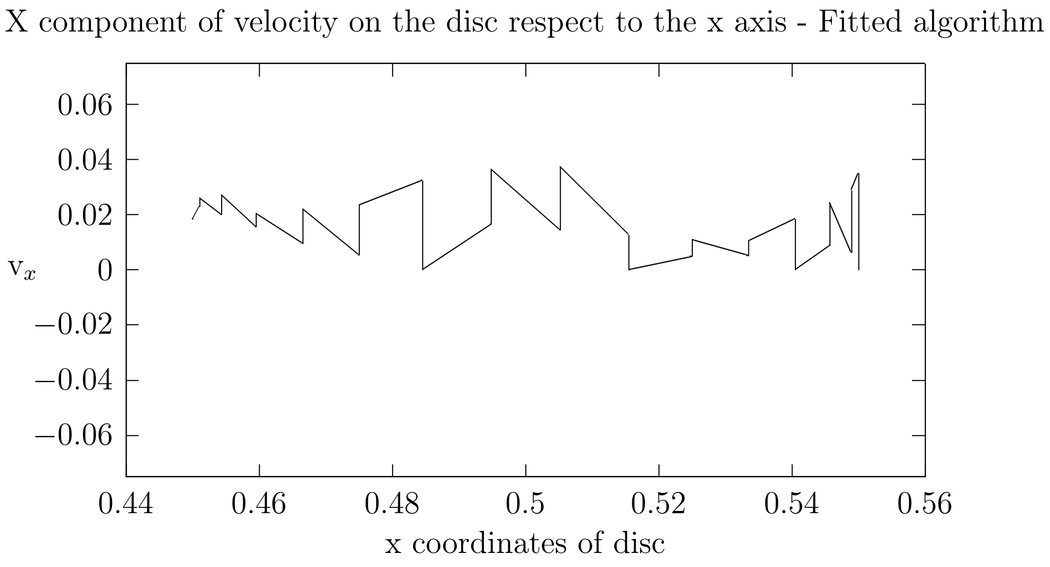
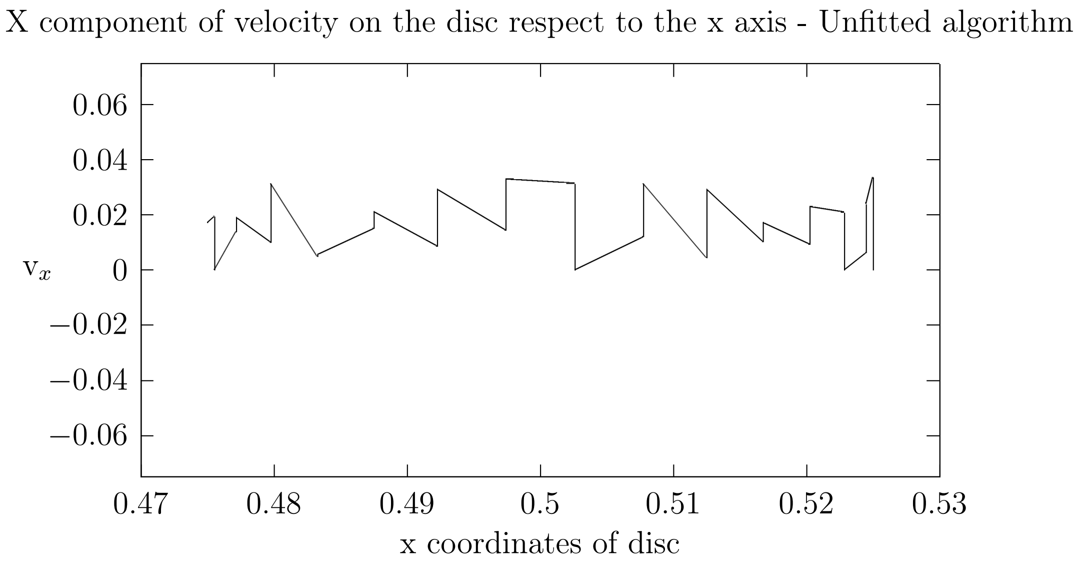


Figure 4.8: Plot of the horizontal component of the velocity along the disc with radius = 0.05; Unfitted (top) and Fitted algorithm (bottom)



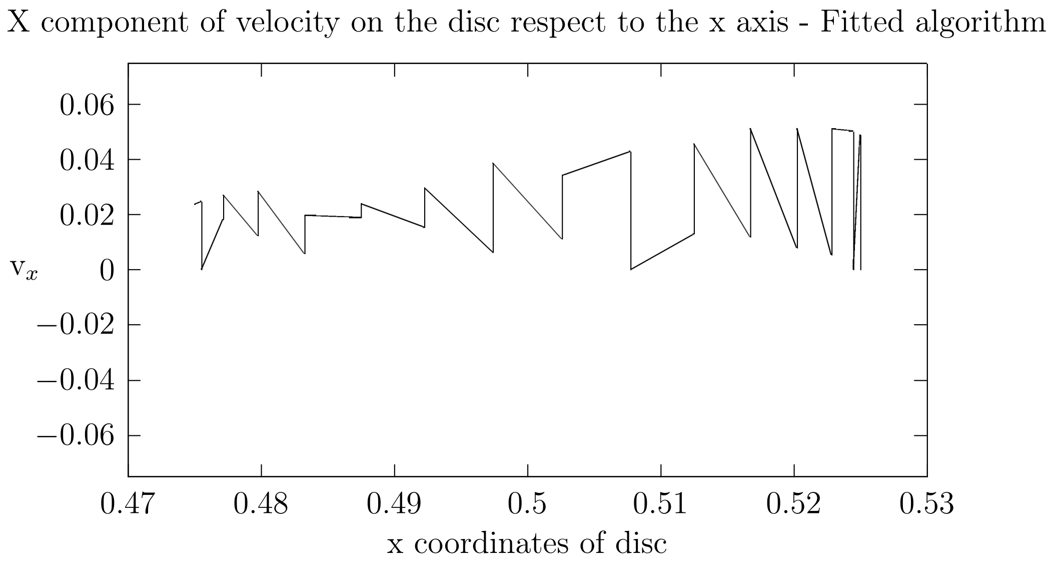
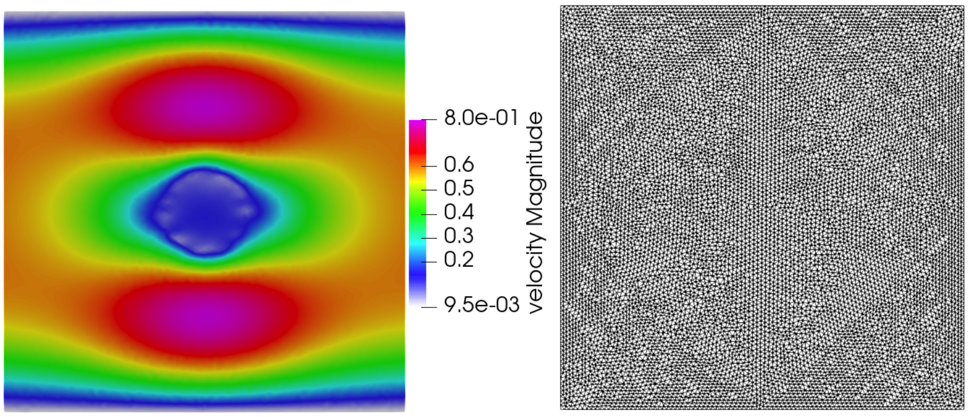


Figure 4.9: Plot of the horizontal component of the velocity along the disc with radius = 0.025; Unfitted (top) and Fitted (bottom) algorithm

4.2 Fluid-structure interaction problem with Stokes flow

In this case we consider fluid-structure interaction problem described in section 3. This is a time dependent problem and we use it to validate the behaviour of the structure. We consider the same condition of the first test case:

- **Neumann boundary condition** $\frac{\partial u}{\partial n} - p \cdot n = \hat{p} \cdot n$, with $\hat{p} = 1$ **on** Γ_4
- **Dirichlet boundary condition** $u = 0$ **on** Γ_1 **and** Γ_3
- A set of points that describe a circonference with a radius, ϵ, equal to 0.1
- A time interval from 0 to 1 with a time-step of 0.05



(a) Unfitted Algorithm. (b) Mesh, border size = 90.

Figure 4.10: Plot of velocity using a mesh not conforming with the disc; Stokes algorithm; $P|_{\Gamma_4} - P|_{\Gamma_2} = 1$

We can see that the plot is slightly different from the previous ones, even though the boundary conditions imposed are the same, because the mesh does not conform with the disc.

4.2.1 Flow driven by a unit pressure drop

We impose an unitary pressure on the left edge and look at how the structure moves in the fluid.

We plot the displacements $[d_x, d_y]$ in function of the time t of the two directions, x and y:

m = 1, k = 1, T = 5, dt = 0.005, number of points of each edge of the mesh = 120

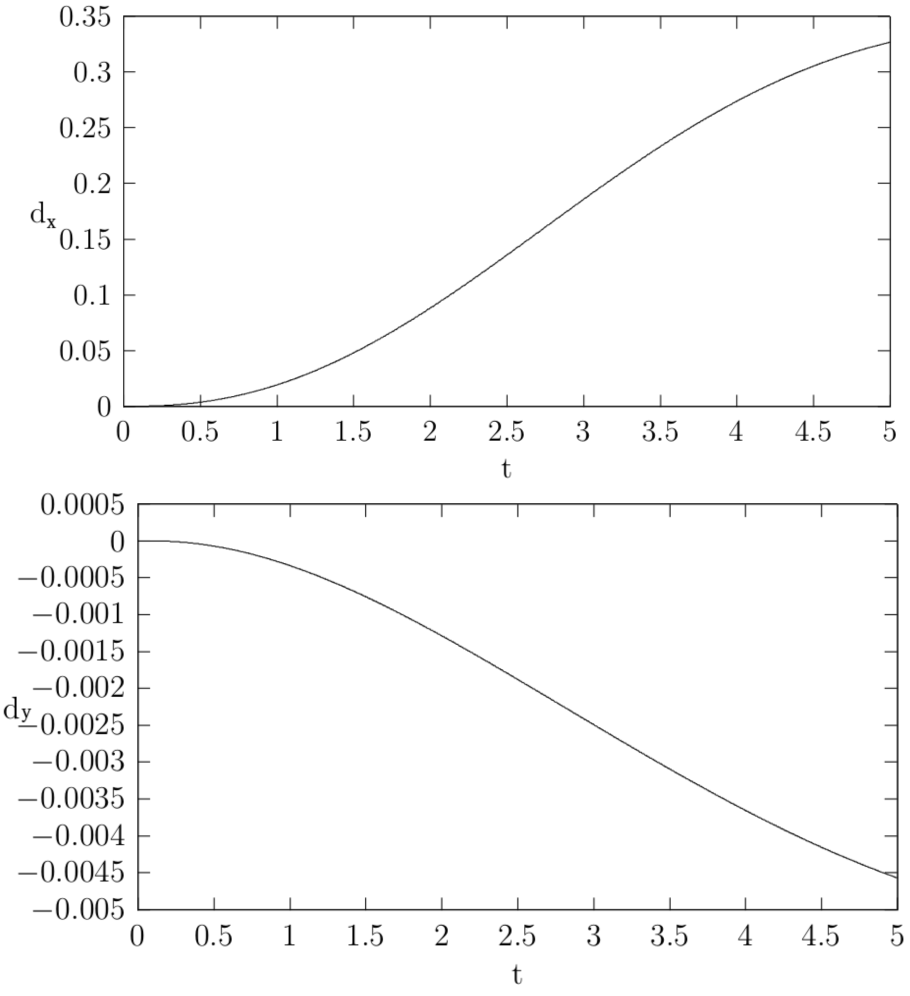


Figure 4.11: Plot of horizontal (top) and vertical (bottom) component of the displacement in function of time; Stokes algorithm; k = 1

We can note that in the horizontal direction (first picture) the structure goes on from it's starting point, moving from left to right, and it is what we expected since the only force is the pressure that pushes the disc forward. In the vertical direction (second picture) there is a slight oscillation of negligible magnitude and this is in accordance with the imposed boundaries conditions, since there are no forces applied in the vertical direction.

The displacement of the disc depends also from the value of "k" which is the stiffness of a body. It is a measure of the resistance opposed by an elastic body to deformation. As we can see in chapter 3, "k" is a value included in the "matrix of the structure", "S", the last block added to the block matrix, which modifies the characteristics of the structure itself and therefore allows to modify the interaction with the fluid. For this reason the disk will move more with a small value of the rigidity of the body as this will oppose a lower resistance to displacement, but, with a higher value of k you will notice the oscillations due to the will of the structure to resist the movement.

Considering what has just been explained we repeat the same test with a different value of k and we see how the displacement change in the two directions.

m = 1, k = 10, T = 5, dt = 0.005, number of points of each edge of the mesh = 120

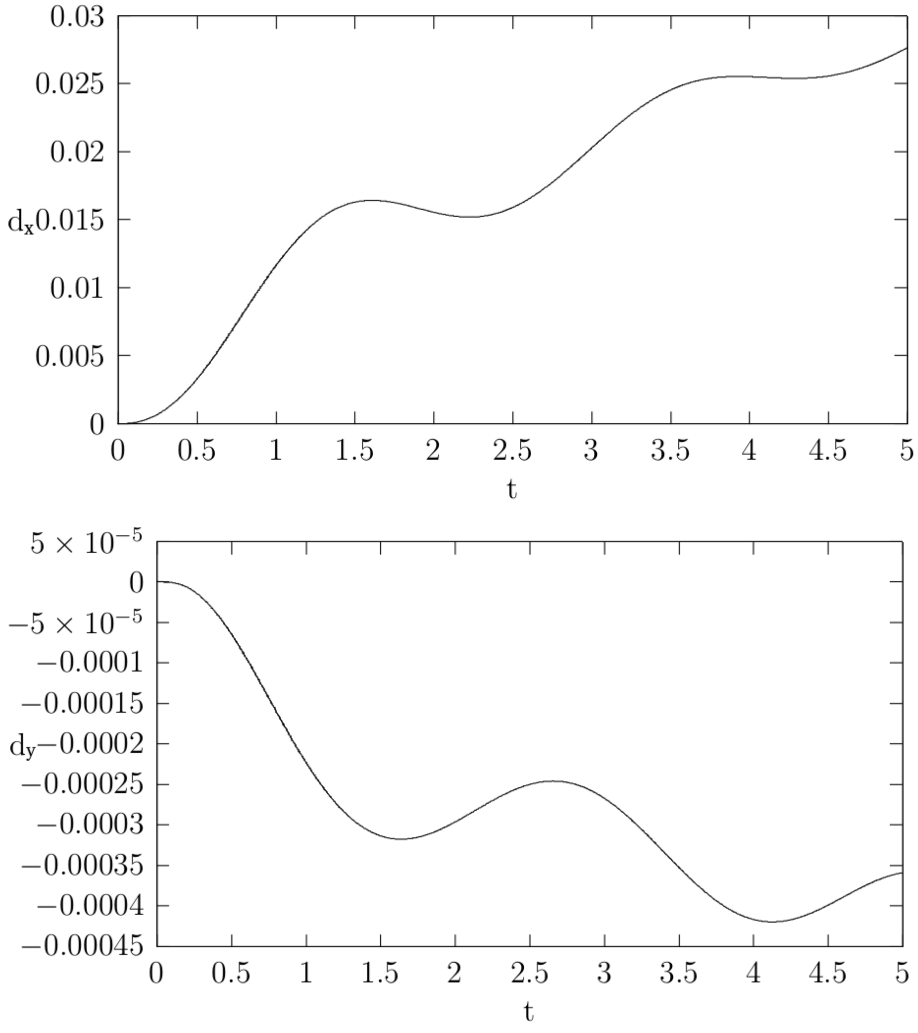


Figure 4.12: Plot of horizontal (top) and vertical (bottom) component of the displacement in function of time; Stokes algorithm; k = 10

With k = 10, the disc moves along the x-axis with an oscillation motion that seems to progressively stabilize towards equilibrium.

The following images represents the case with a null resistance offered by the body, k = 0.

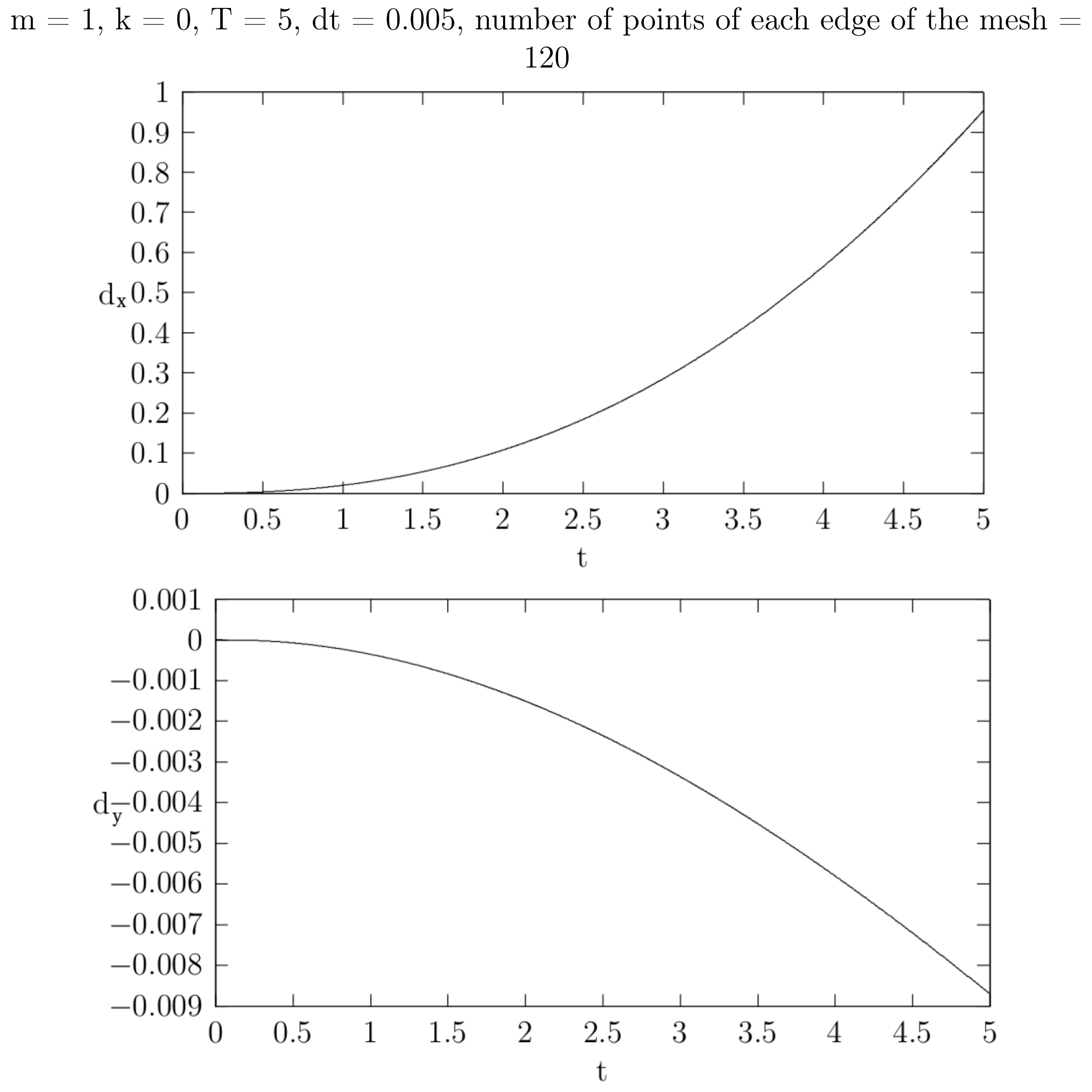


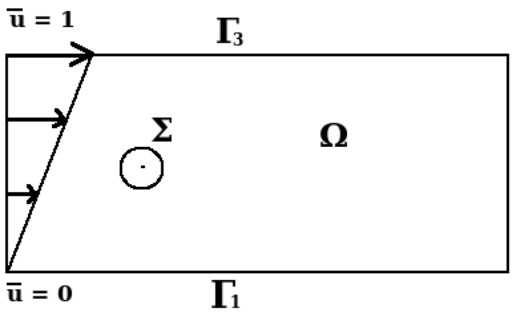
Figure 4.13: Plot of horizontal (top) and vertical (bottom) component of the displacement in function of time; Stokes algorithm; k = 0

From the graphs above we note that the structure moves more, than with k = 1 and k = 10 and without oppose any resistance to the movement.

4.2.2 Couette flow and shear layers

As second test for the obstacle test-case we impose a Couette flow.

In fluid dynamics, Couette flow is the flow of a viscous fluid in the space between two surfaces, one of which is moving tangentially relative to the other.
The Couette flow is characterized by a constant shear stress distribution.
In laminar flow regime, the velocity profile is linear. The configuration often takes the form of two parallel plates or the gap between two concentric cylinders. The flow is driven by virtue of viscous drag force acting on the fluid. The Couette configuration models certain practical problems, like flow in lightly loaded journal bearings, and is often employed in viscometry and to demonstrate approximations of reversibility.



To impose the Couette flow in our simulations we use the following boundary conditions that allow us to obtain a linear velocity profile in the domain:

- u = 1 on Γ_3
- u = -1 on Γ_1

Indeed, if we represent, as it is showed in the following picture, the flow without any disc, we can see the linear flow on the domain. This thpe of flow is also called a shear layer.

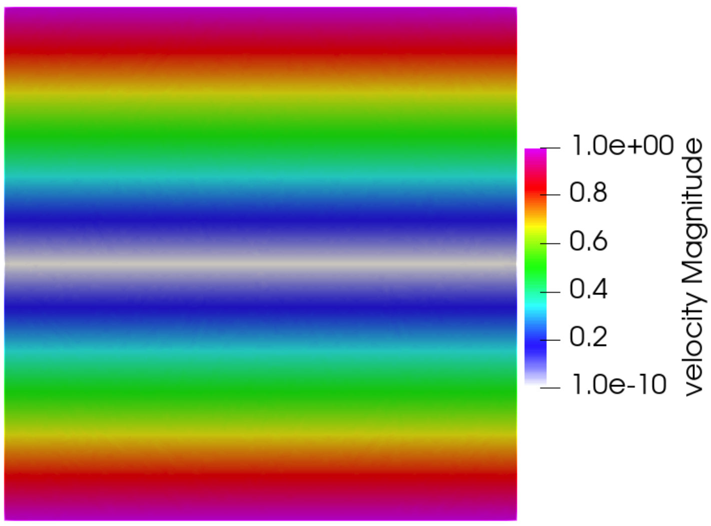


Figure 4.14: Plot of velocity: Couette flow; mesh without disc

As soon as we disrupt the flow with an obstacle, the disc in our case, we can notice that the flow is not linear anymore, as shown in the previous case where there are no obstacles, and we can study the movement of the disc in both the directions x and y. This instability is usually called the Taylor vortex instability of a Couette flow.

To study more accurately the movement of the disc, we shift the disc from the center to the upper part of the domain, in order to break the symmetry of the model.

$$x_c = 0.5 \text{ and } y_c = 0.75$$

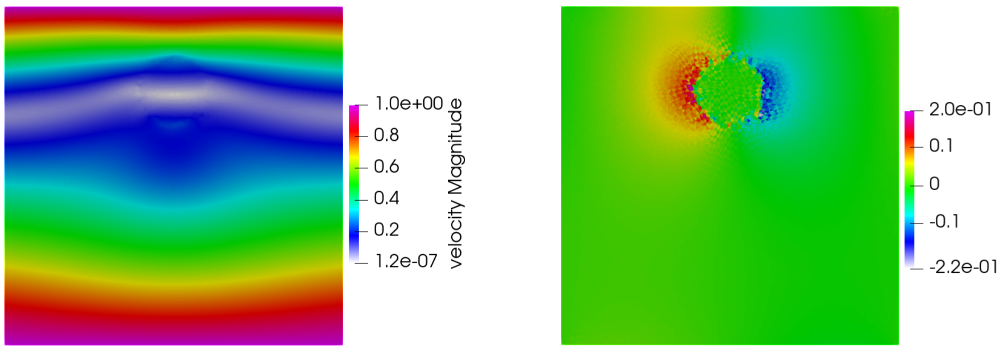


Figure 4.15: Plot of velocity and pressure profiles: Couette; mesh with one disc

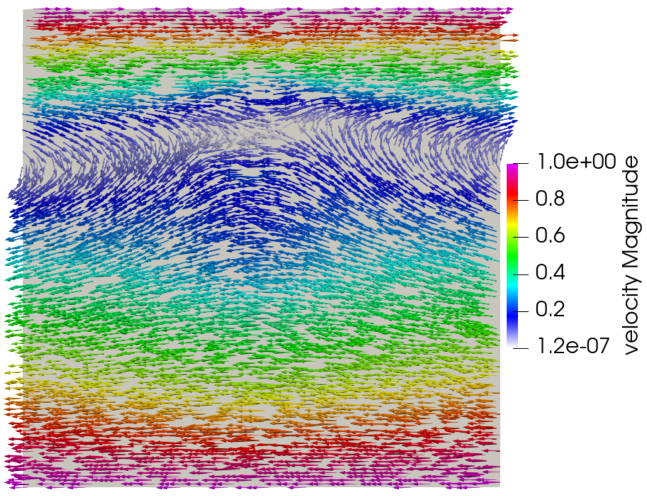


Figure 4.16: Plot of streamlines velocity field: Couette flow; mesh with one disc

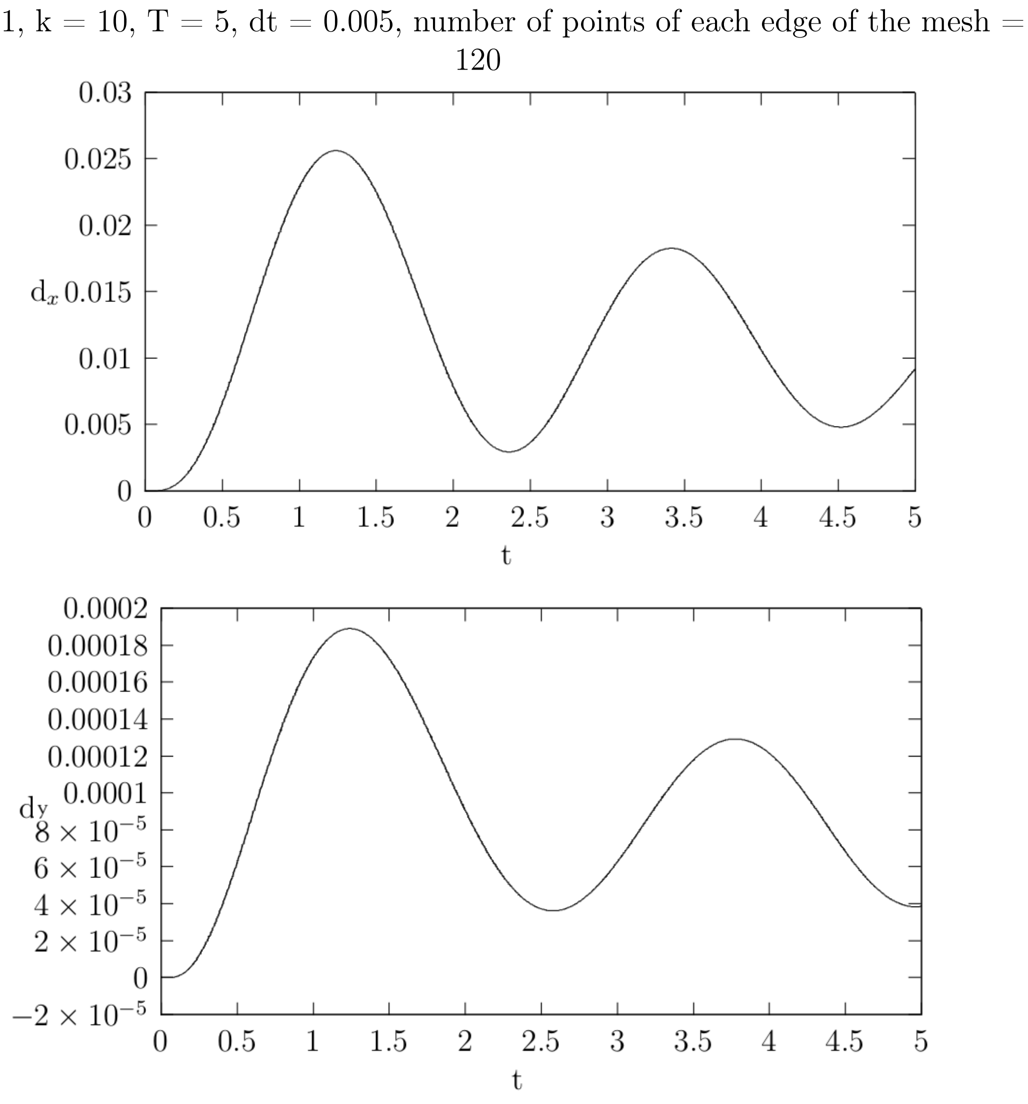


Figure 4.17: Plot of horizontal (top) and vertical (bottom) component of the displacement in function of time: Couette flow; mesh with one disc; k = 10

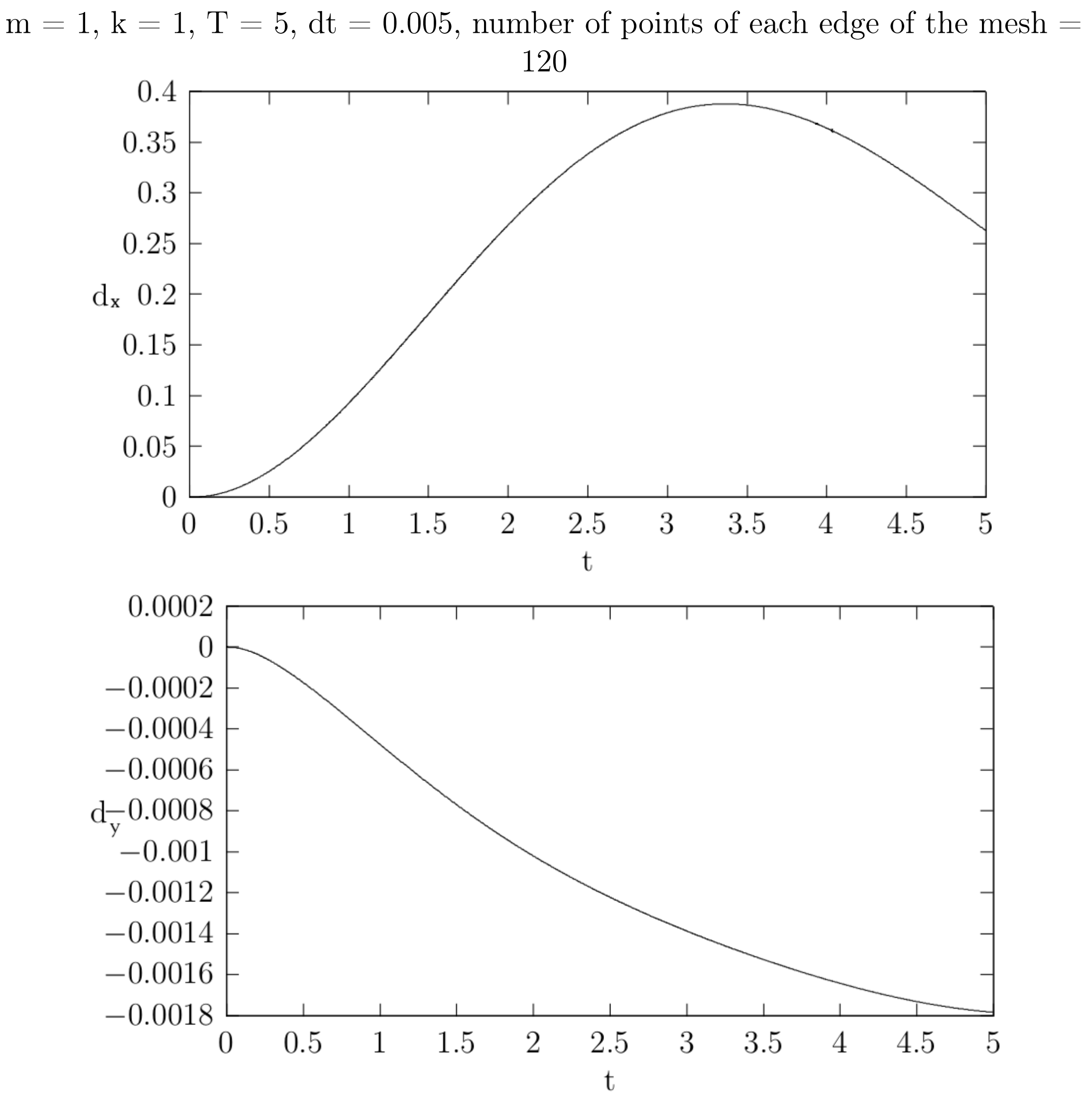


Figure 4.18: Plot of horizontal (top) and vertical (bottom) component of the displacement in function of time: Couette flow; mesh with one disc; k = 1

Now we repeat the same test with a null value of k.

m = 1, k = 0, T = 5, dt = 0.005, number of points of each edge of the mesh = 120

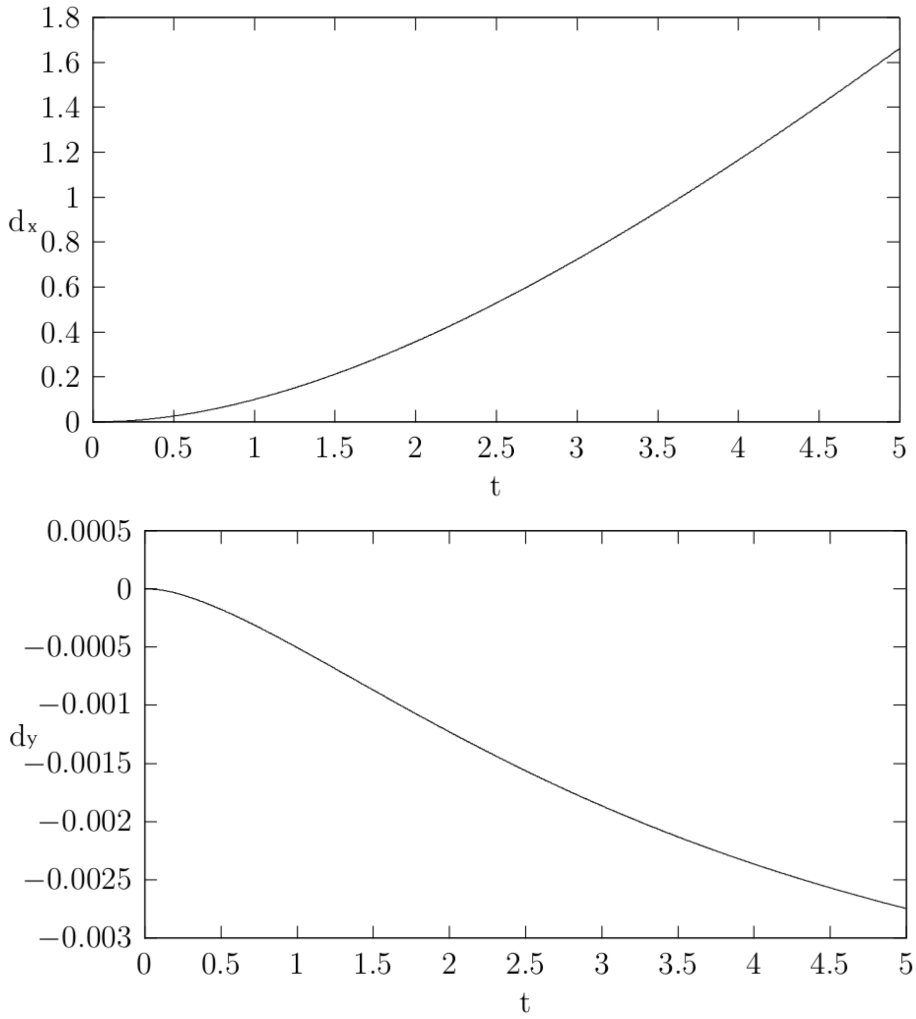


Figure 4.19: Plot of horizontal (top) and vertical (bottom) component of the displacement in function of time: Couette flow; mesh with one disc; k = 0

As in the previous case, we note that the structure moves more, even if not a lot, than with k = 1 and k = 10.

Now we add another disk to the domain in order to have a symmetrical output of the velocity field.
We use the following data:

$$x_c = 0.5 \text{ and } y_c = 0.75 \text{ is the position of the first disc.}$$
$$x_c = 0.5 \text{ and } y_c = 0.25 \text{ is the position of the second disc.}$$
$$\epsilon = \text{radius} = 0.1.$$

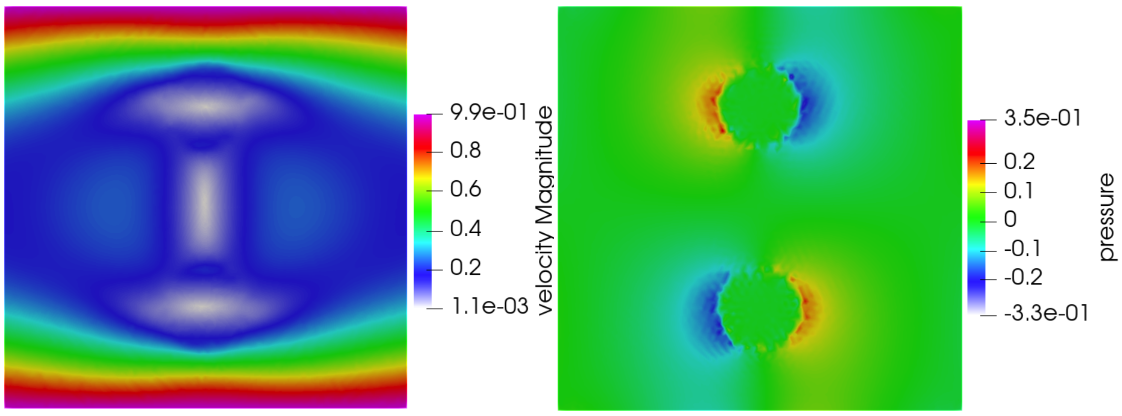


Figure 4.20: Plot of velocity and pressure profiles: Couette flow; mesh with two discs; $\epsilon = 0.1$

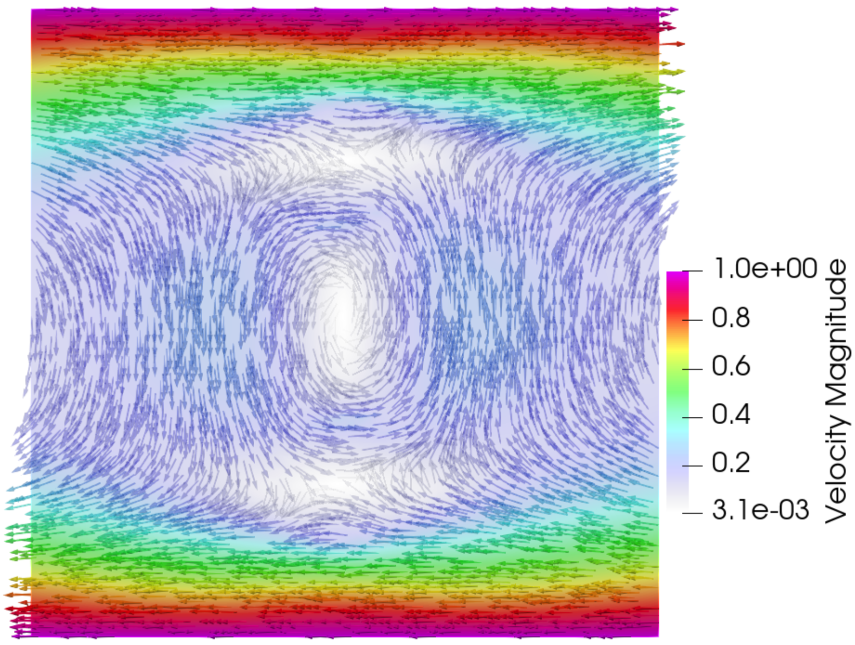


Figure 4.21: Plot of streamlines of velocity field: Couette flow; mesh with two discs; $\epsilon = 0.1$

Figure 4.21 shows the formation of a Taylor vortex at the center of the domain.

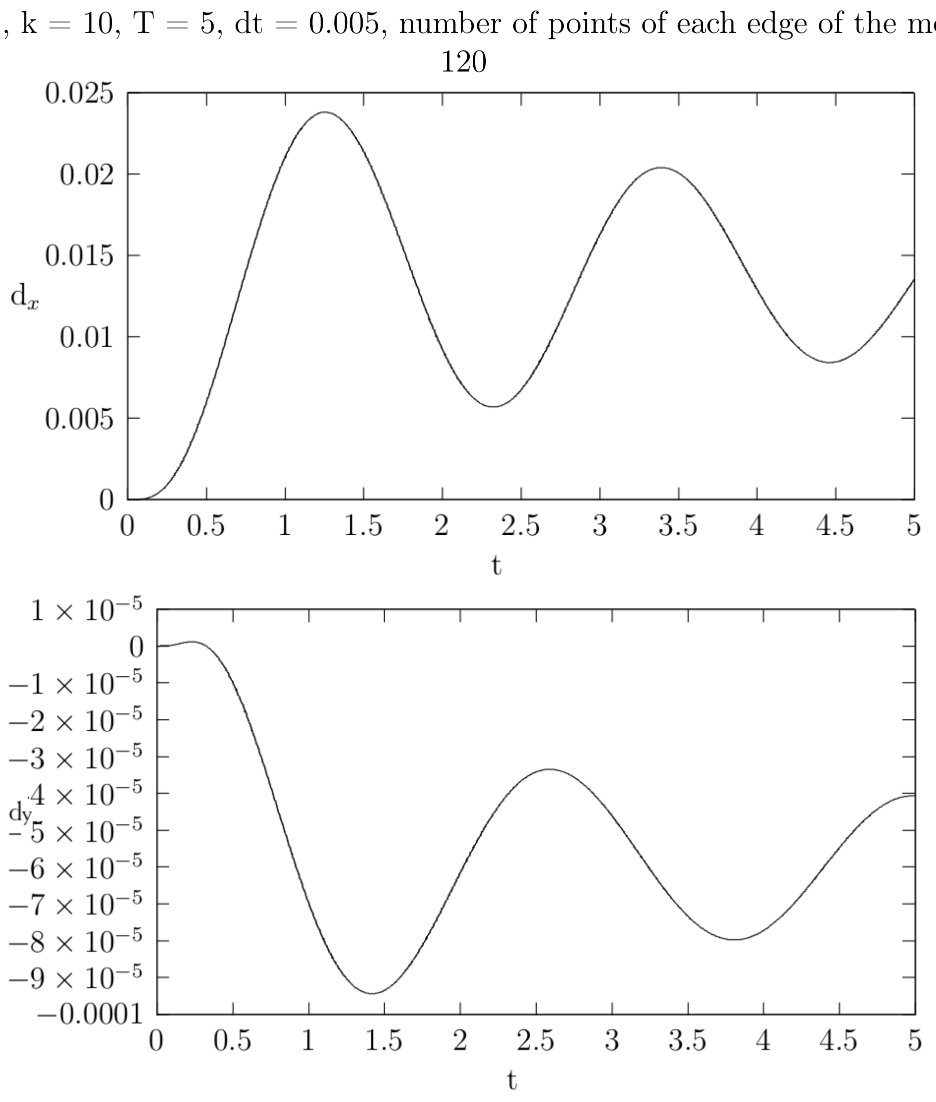


Figure 4.22: Plot of horizontal (top) and vertical (bottom) component of the displacement of the upper disc in function of time: Couette flow; mesh with two discs; $\epsilon = 0.1$

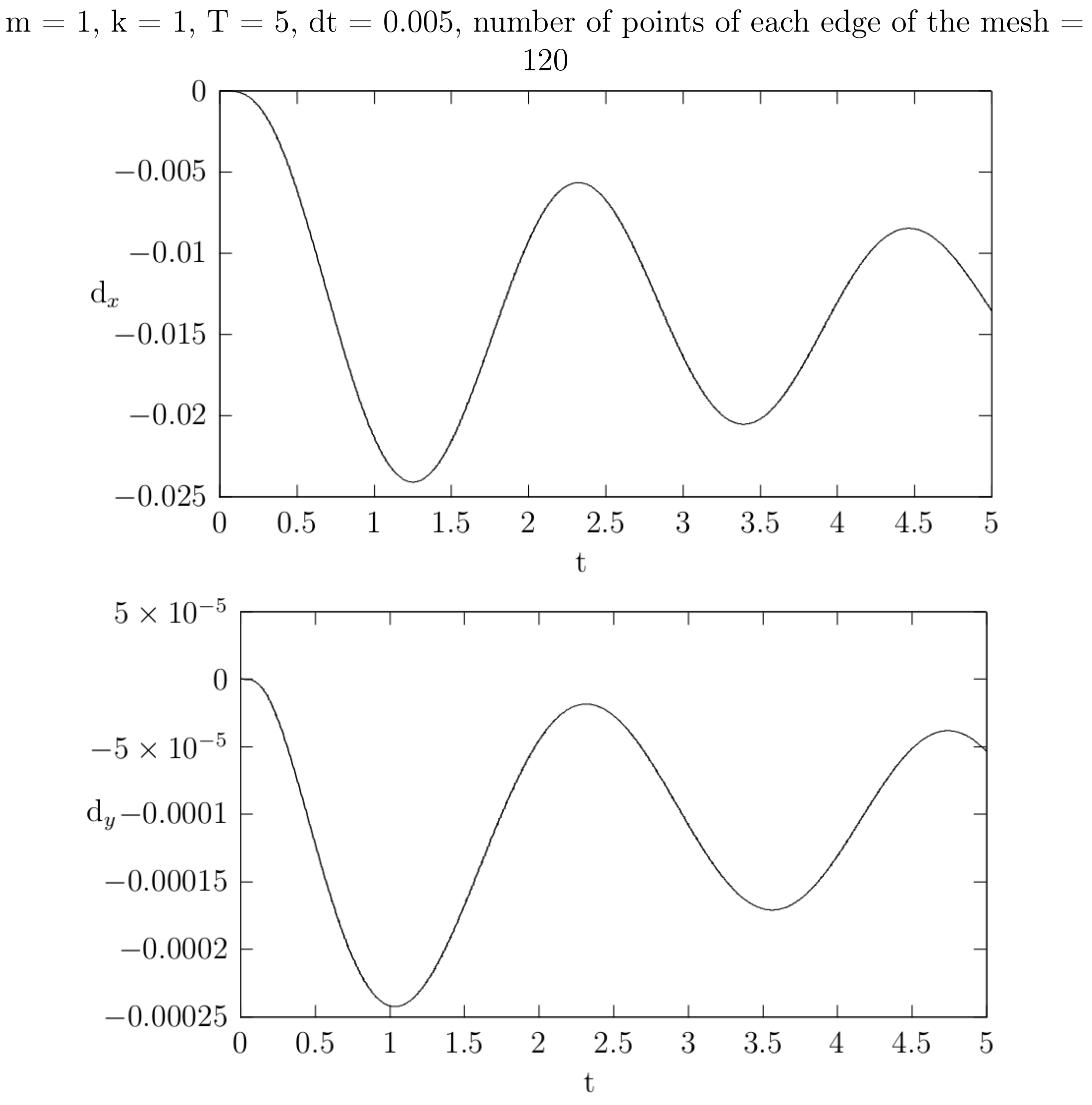


Figure 4.23: Plot of horizontal (top) and vertical (bottom) component of the displacement of the lower disc in function of time: Couette flow; mesh with two discs; $\epsilon = 0.1$

Now we repeat the same test, but halving the size of the diameter of the discs.

$x_c = 0.5$ and $y_c = 0.75$ is the position of the first disc.
$x_c = 0.5$ and $y_c = 0.25$ is the position of the second disc.
ϵ = radius = 0.05.

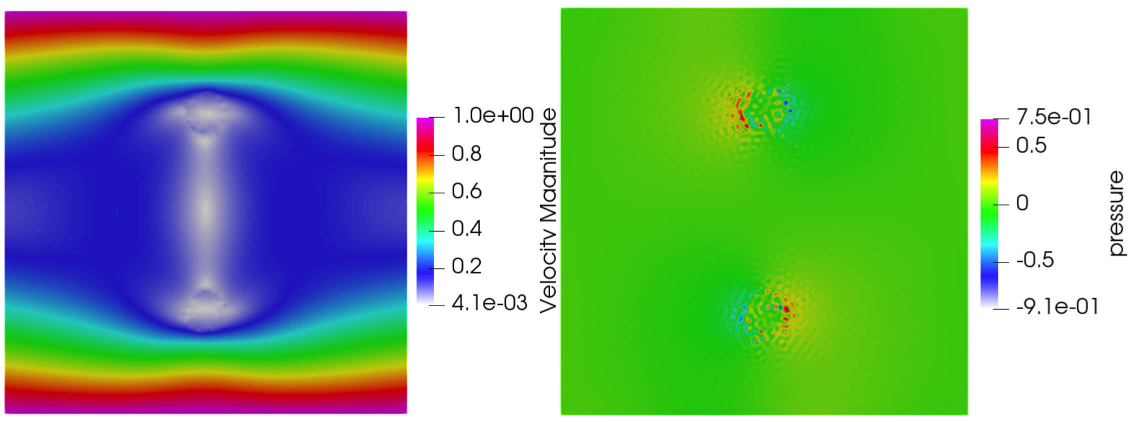


Figure 4.24: Velocity and pressure profiles: Couette flow; mesh with two discs; $\epsilon = 0.05$

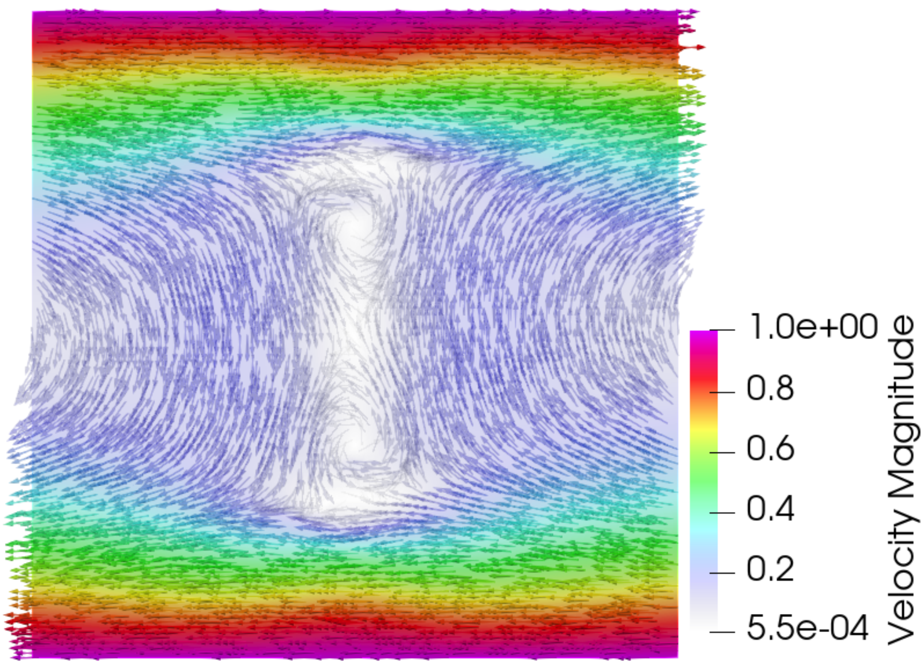


Figure 4.25: Streamlines: Couette flow; mesh with two discs; $\epsilon = 0.05$

m = 1, k = 10, T = 5, dt = 0.005, number of points of each edge of the mesh = 120

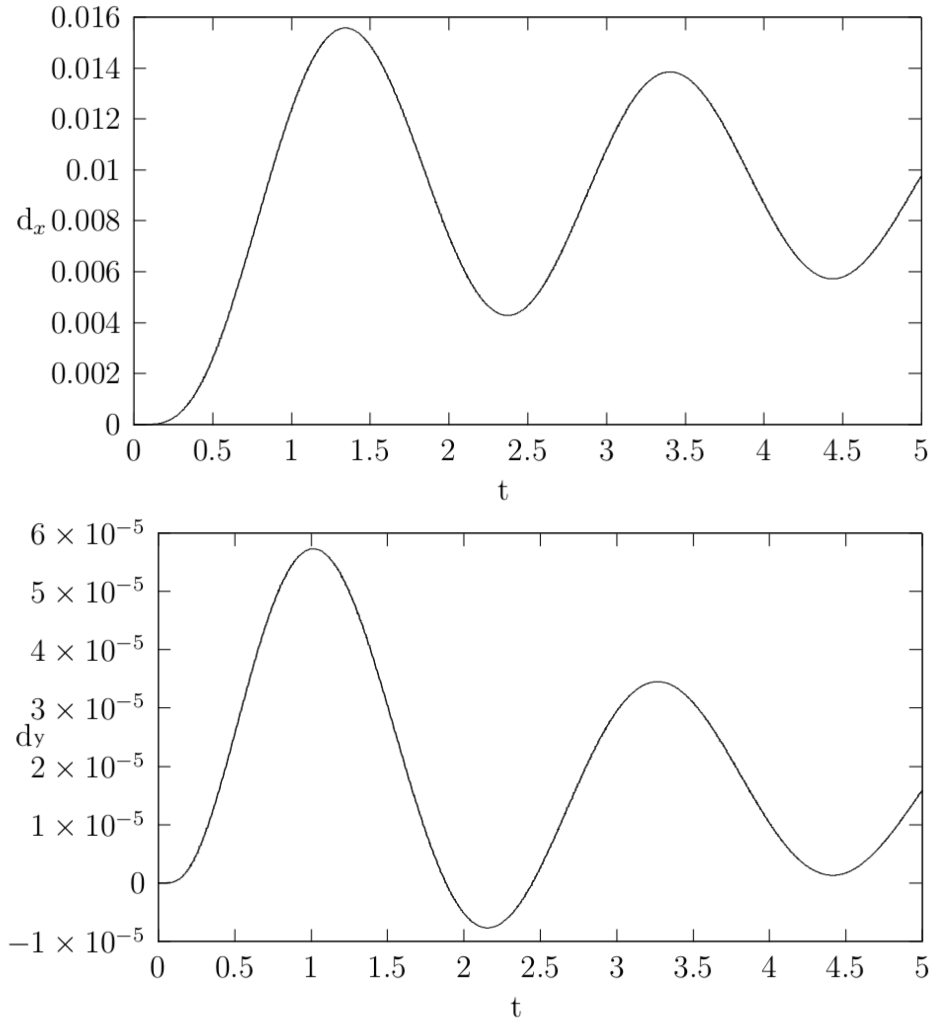


Figure 4.26: Plot of horizontal (top) and vertical (bottom) component of the displacement of the upper disc in function of time: Couette flow; mesh with two discs; $\epsilon = 0.05$

m = 1, k = 10, T = 5, dt = 0.005, number of points of each edge of the mesh = 120

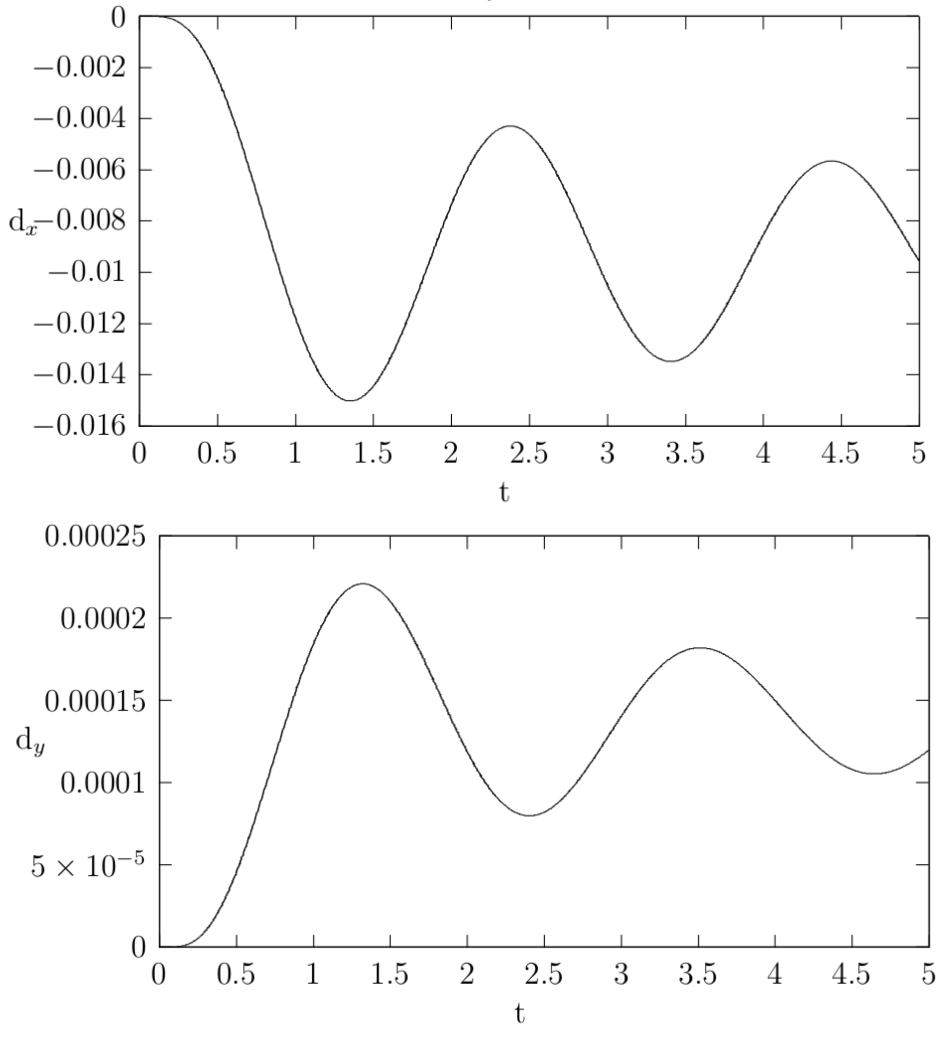


Figure 4.27: Plot of horizontal (top) and vertical (bottom) component of the displacement of the lower disc in function of time: Couette flow; mesh with two discs; $\epsilon = 0.05$

This test case is qualitatively similar to the one with $\epsilon = 0.1$.

4.3 Navier-Stokes

The last step is to complete Navier-Stokes' equation with the convective term. Only the matrix S is modified, as explained in the third chapter.

The complete equations becomes:

$$
\begin{aligned}
(\partial_{\mathrm{t}} u, v) + \nu(\nabla u, \nabla v) - (p, divv) + (q, divu) + ((u\nabla)u, v) + \\
-\gamma \tau_{\mathrm{SUPG/PSPG}}(\nabla p + u\nabla u, \nabla q + v\nabla v) + (L, \frac{1}{|\Sigma|}\int_{\Gamma} v \; - \xi) + \\
+(m\ddot{d} + kd)\xi + (M, \frac{1}{|\Sigma|}\int_{\Gamma} u - \dot{d}) = 0
\end{aligned}
\tag{4.1}
$$

4.3.1 Flow driven by a unit pressure drop

As the first test for Navier Stokes' equation, we impose an unitary pressure on the left edge and look at how the structure moves in the fluid, as we did for Stokes' algorithm. The comparison with Stokes flow test case will serve for validation of the code.

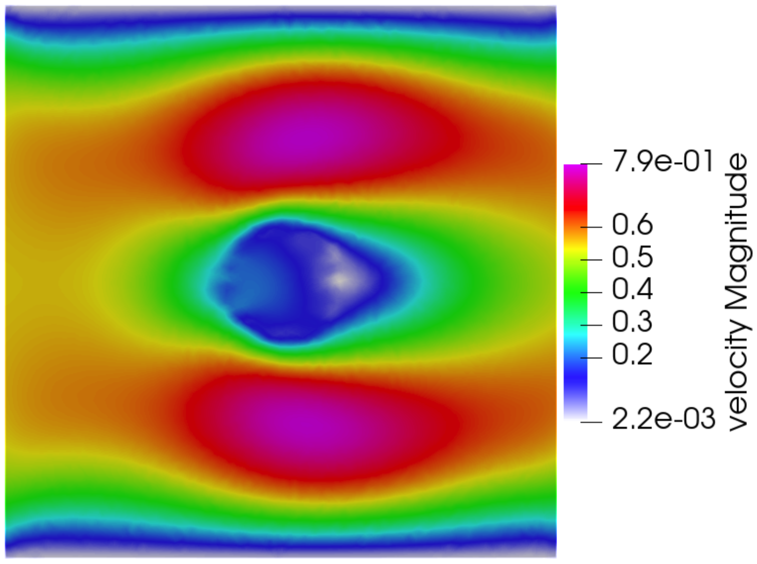


Figure 4.28: Plot of velocity; Navier-Stokes model; $P|_{\Gamma_4} - P|_{\Gamma_2} = 1$; $\epsilon = 0.1$

We plot the displacements $[d_x, d_y]$ in function of the time t of the two directions, x and y:

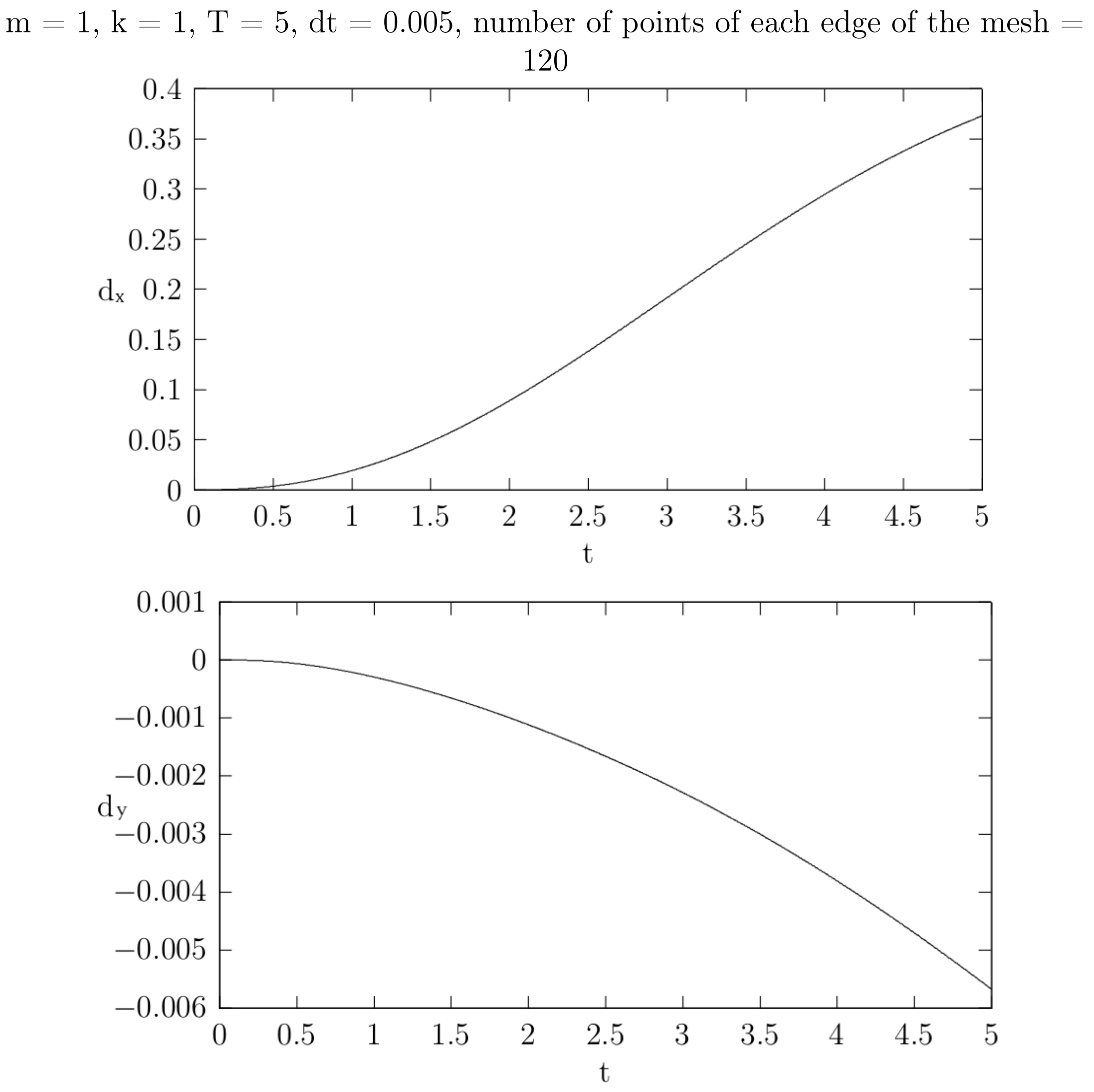


Figure 4.29: Plots of horizontal (upper) and vertical (lower) component of the displacement of the disc in function of time: Navier-Stokes model

m = 1, k = 10, T = 5, dt = 0.005, number of points of each edge of the mesh = 120

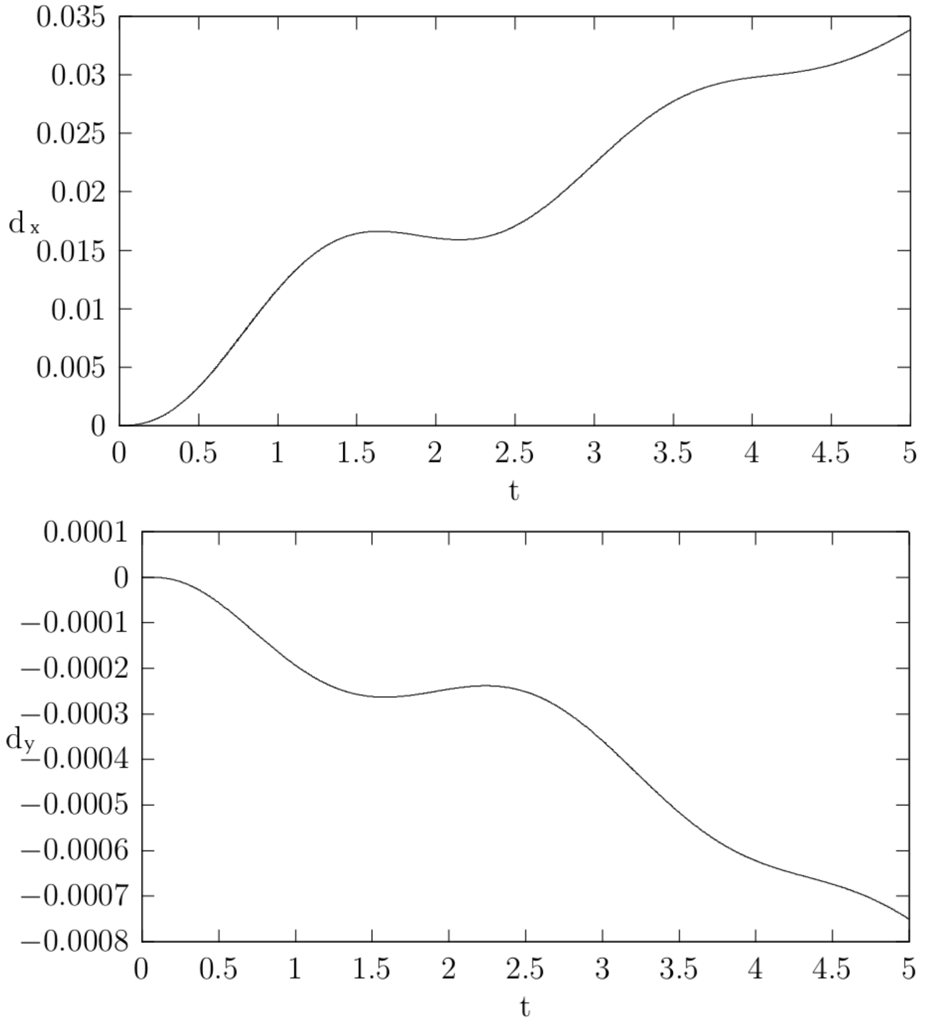


Figure 4.30: Plots of horizontal (upper) and vertical (lower) component of the displacement: Navier-Stokes model, stiffer structure of the previous case

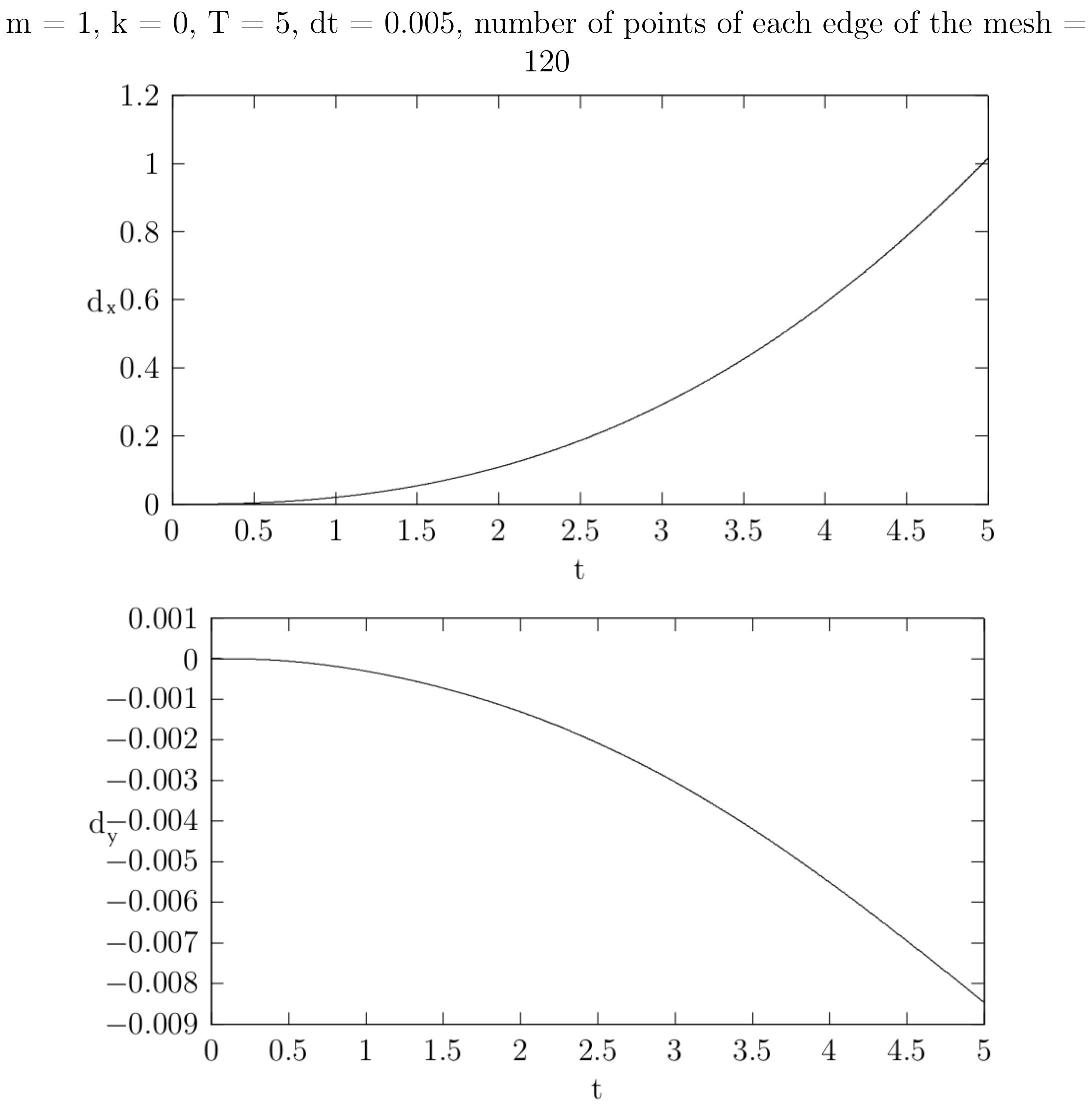


Figure 4.31: Plots of horizontal (upper) and vertical (lower) component of the displacement: Navier-Stokes model, null stiffness of the structure

We can see that the trend of the fluid-structure interaction is the same as for the model of Stokes. The values of the disc movement are also similar. So we notice a consistency in the two algorithms.

4.3.2 Von Kármán flow

In fluid dynamics, a Von Kármán vortex street is a repeating pattern of swirling vortices, caused by a process known as vortex shedding, which is responsible for the unsteady separation of flow of a fluid around blunt bodies. It is named after the engineer and fluid dynamicist Theodore von Kármán and is responsible for such phenomena as the "singing" of suspended telephone or power lines and the vibration of a car antenna at certain speeds. This problem is deeply studied ad exemple in [4] and [9].

A vortex street will only form at a certain range of flow velocities, specified by a range of Reynolds numbers (Re), typically above a limiting Re value of about 90. The (global) Reynolds number for a flow is a measure of the ratio of inertial to viscous forces in the flow of a fluid around a body or in a channel, and may be defined as a dimensionless parameter of the global speed of the whole fluid flow:

$$Re_L = \frac{UL}{\nu_0}$$

where:

- U = the free stream flow speed
- L = a characteristic length parameter of the body or channel
- ν_0 = the kinematic viscosity parameter of the fluid, which in turn is the ratio:

$$\nu_0 = \frac{\mu_0}{\rho_0}$$

between:

- ρ_0 = the fluid density.

- μ_0 = the fluid dynamic viscosity

For common flows (the ones which can usually be considered as incompressible or isothermal), the kinematic viscosity is everywhere uniform over all the flow field and constant in time, so there is no choice on the viscosity parameter, which becomes naturally the kinematic viscosity of the fluid being considered at the temperature being considered. On the other hand, the reference length is always an arbitrary parameter, so particular attention should be put when comparing flows around different obstacles or in channels of different shapes: the global Reynolds numbers should be referred to the same reference length. The reference length can vary depending on the analysis to be performed: for a body with circle sections such as circular cylinders or spheres, one usually chooses the diameter; for an airfoil, a generic non-circular cylinder or a bluff body or a revolution body like a fuselage or a submarine, it is usually the profile chord or the profile thickness, or some other given widths that are in fact stable design inputs; for flow channels usually the hydraulic diameter about which the fluid is flowing. The range of Re values will vary with the size and shape of the body from which the flow glides, as well as with the kinematic viscosity of the fluid. Over a large Re range ($40 < \mathrm{Re} < 105$ for circular cylinders; reference length is d: diameter of the circular cylinder) eddies are shed continuously from each side of the circle boundary, forming rows of vortices in its wake. The alternation leads to the core of a vortex in one row being opposite the point midway between two vortex cores in the other row. Ultimately, the energy of the vortices is consumed by viscosity as they move further down stream, and the regular pattern disappears.
When a single vortex is shed, an asymmetrical flow pattern forms around the body and changes the pressure distribution. This means that the alternate shedding of vortices can create periodic lateral forces on the body, causing it to vibrate. If the vortex shedding frequency is similar to the natural frequency of a body

or structure, it causes resonance. It is this forced vibration that, at the correct frequency, causes suspended telephone or power lines to "sing" and the antenna on a car to vibrate more strongly at certain speeds.

In this section we aims to test the reduced fluid-structure interaction model using the Von Karman vortex street case. We considered as a reference the Placzek's paper [9] which study the fluid-structure interaction between a Newtonian incompressible fluid and a cylinder. We will use the same computational domain of the paper and it is represented in Fig. 4.32 with the cylinder of diameter d.

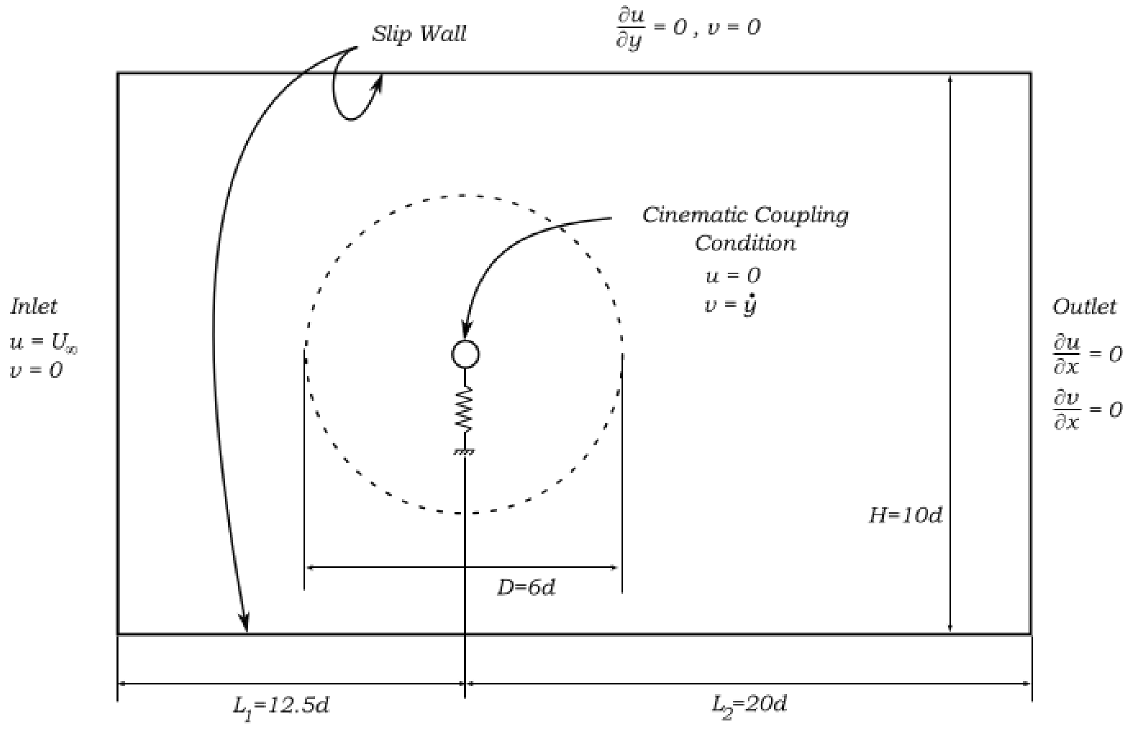

Slip Wall
$\frac{\partial u}{\partial y} = 0$, $v = 0$
Cinematic Coupling Condition
$u = 0$
$v = \dot{y}$
Inlet
$u = U_\infty$
$v = 0$
Outlet
$\frac{\partial u}{\partial x} = 0$
$\frac{\partial v}{\partial x} = 0$
H=10d
D=6d
L_1=12.5d
L_2=20d

Figure 4.32: Size and geometric disposition of the computational domain

As values used to compute Navier-Stokes test we used $\mu = 0.0035$, $\rho = 1$ and accordingly $\nu = 0.0035$.

The mesh is refined to achieve a more precise result in terms of speed and pressure. We add an inner rectangle and two circle. The outer circle is in accordance with the computational domain explained above and the inner one has the same radius of the hole in the fitted mesh, as we already have done for Stokes case in the section 4.1.1.

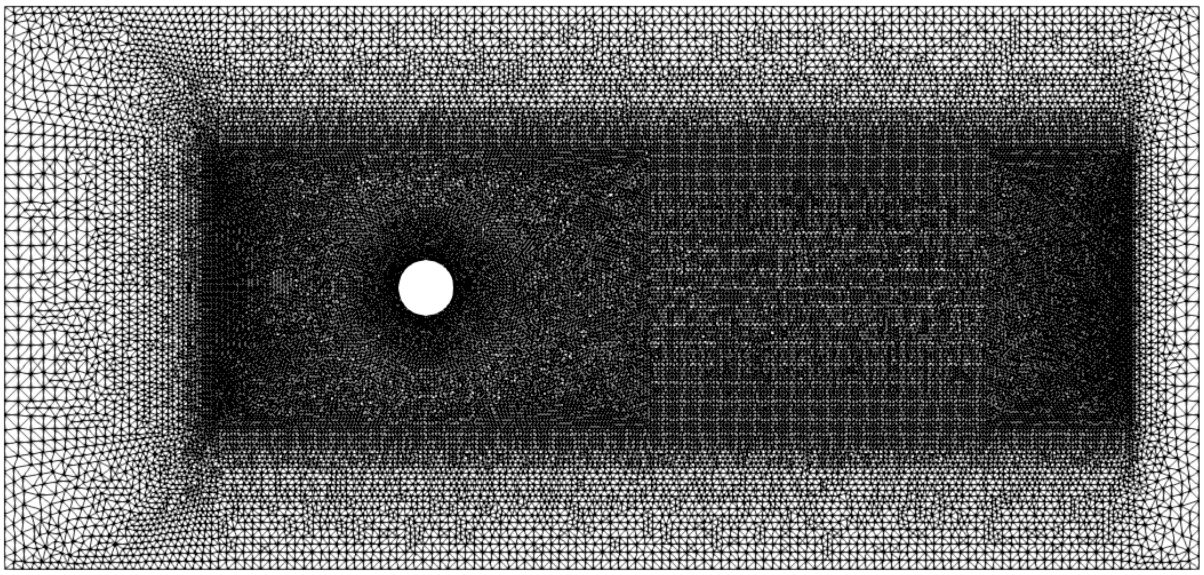
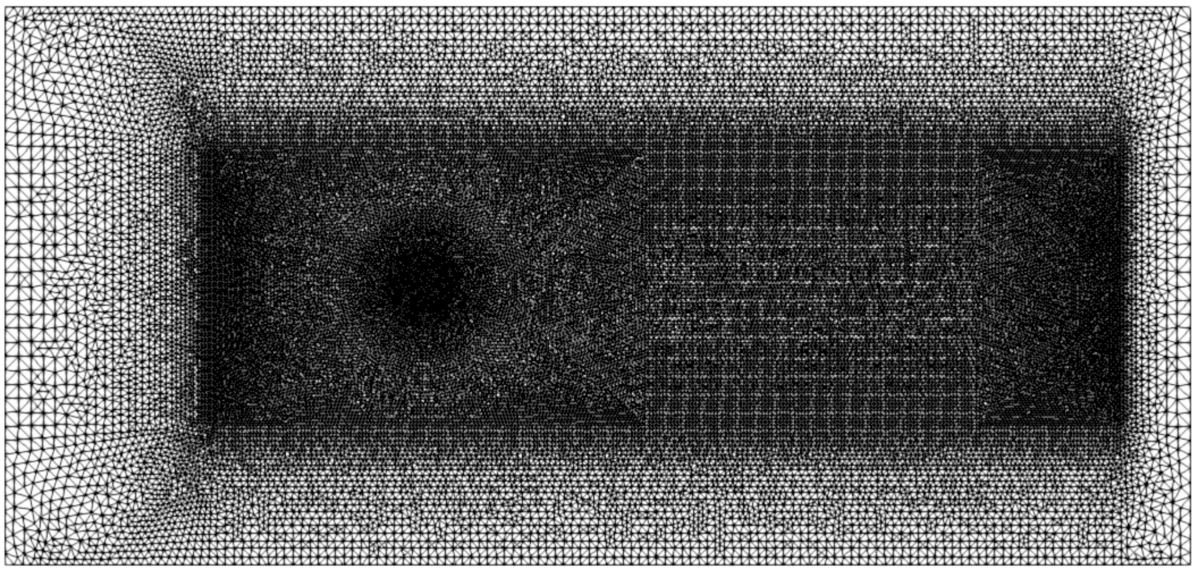


Figure 4.33: Model Mesh Von Kármán street; fitted algorithm

The parameters of the mesh are the following:

- Outer rectangle: upper and lower border have 140 points, right and left have 40 points.
- Inner rectangle: upper and lower border have 250 points, right and left have 100 points.
- Outer disc: 140 points.
- Inner disc: 90 points.



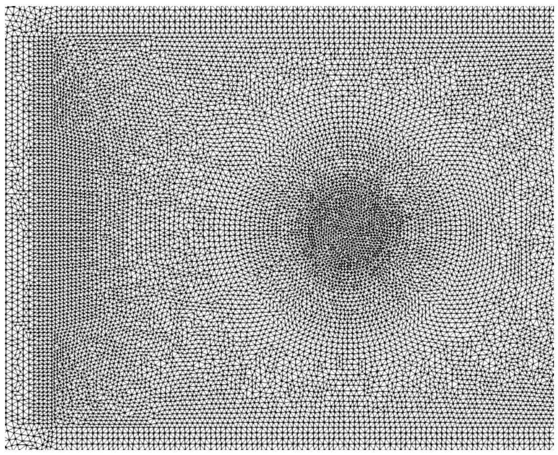


Figure 4.34: Model Mesh Von Kármán street; unfitted algorithm

Once the domain was fixed and reproduced with an appropriate mesh, different tests were made to validate the model.

The first test is with a Reynolds number equal to 40. Below are reported the images of the velocity and the pressure fields. We can notice that, due to the not enough high Re number, there is no evidence of vortex.

In order to calculate the correct minimum time to use, we have used the relation between Strouhal number and Reynolds number data taken from the mentioned paper.

In particular we define the Strouhal number as:

$$St = f_s \frac{d}{u_{in}}$$

Precisely, given St(Re), d, u_{in} we calculate f_s that is the characteristic frequency of vortex shedding. Then, we know that to observe the detachment of a vortex the numerical simulation time must be $T > \frac{1}{f_s}$.

For our tests we have, in the first case, Re = 40 and a correspondent St = 0.11 and consequently the time = 3.85 and in the second case, Re = 100 so St = 0.16 hence t = 5.6.

Below are reported the images of velocity contour and pressure, using Re = 40, taken from the paper already mentioned before. We notice that the results are comparable with the one we obtained with our namely fitted model, which results are reported in the netx pages.

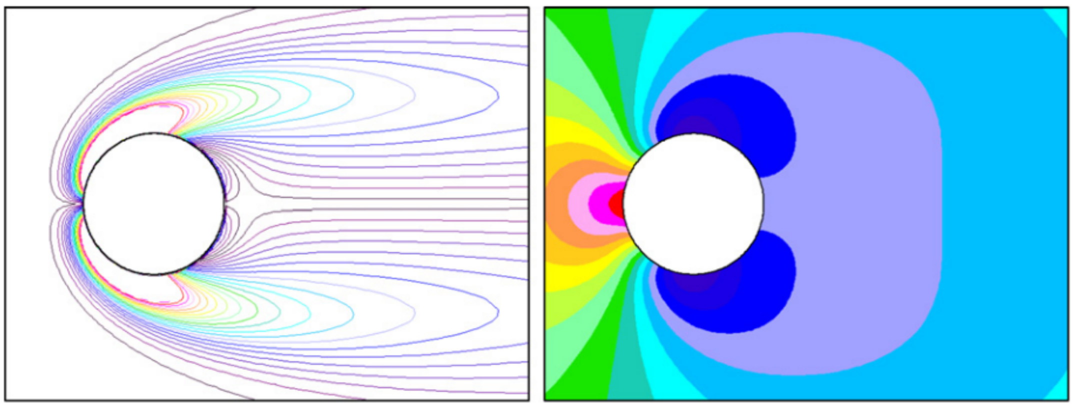


Figure 4.35: Velocity contour and Pressure; Literature and Navier-Stokes fitted algorithm; Re = 40

Now we consider the disc cut in half by the x-axis and we plot separately the velocity and the pressure on the upper half of the disc and on the lower one respect to the abscissa.

Re $= 40$, $v_x = 1.4$ on Γ_4 and $v_y = 0$ on Γ_1 and Γ_3, T $= 10$, dt $= 0.05$, $\epsilon = 0.1$

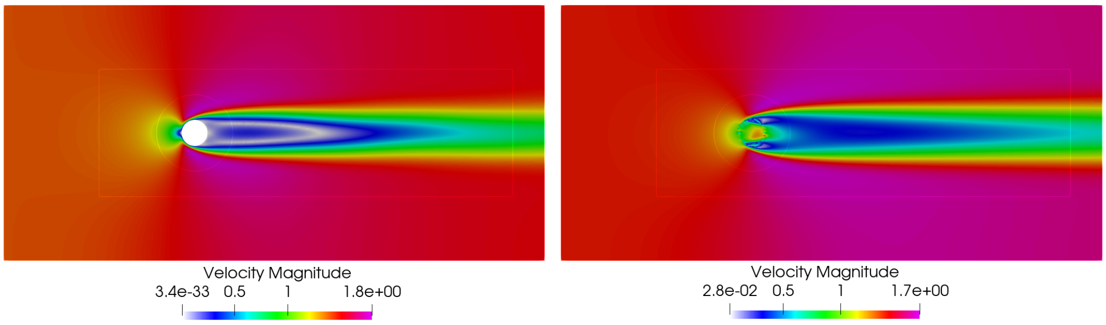


Figure 4.36: Plot of velocity; Navier-Stokes fitted and unfitted algorithms; Von Kármán street; Re $= 40$

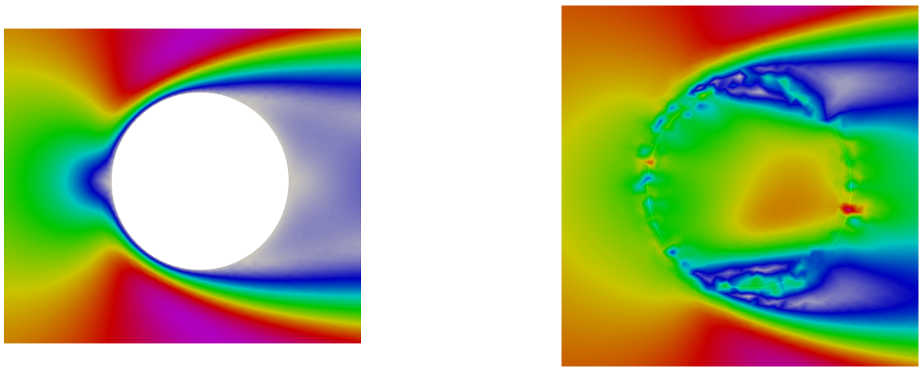


Figure 4.37: Plot of zoom of the velocity; Navier-Stokes fitted and unfitted algorithms; Von Kármán street; Re $= 40$

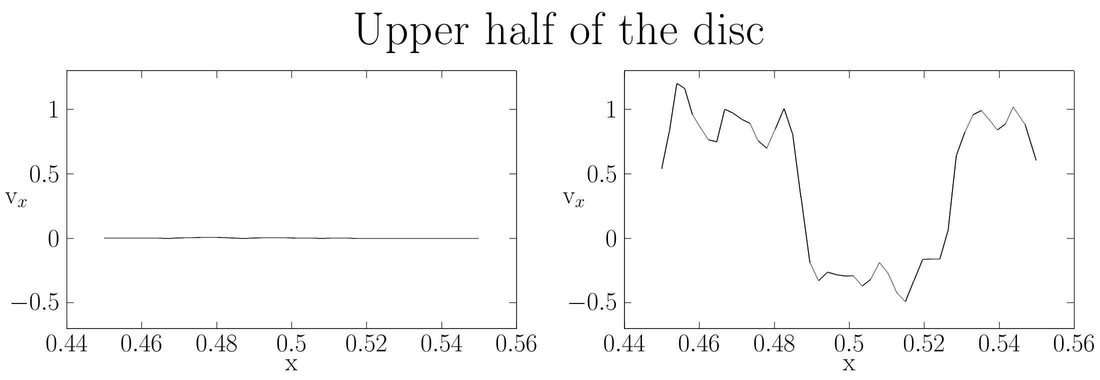


Figure 4.38: Plot of the horizontal component of the velocity along the disc; fitted and unfitted algorithm; Re $= 40$, upper half of the disc

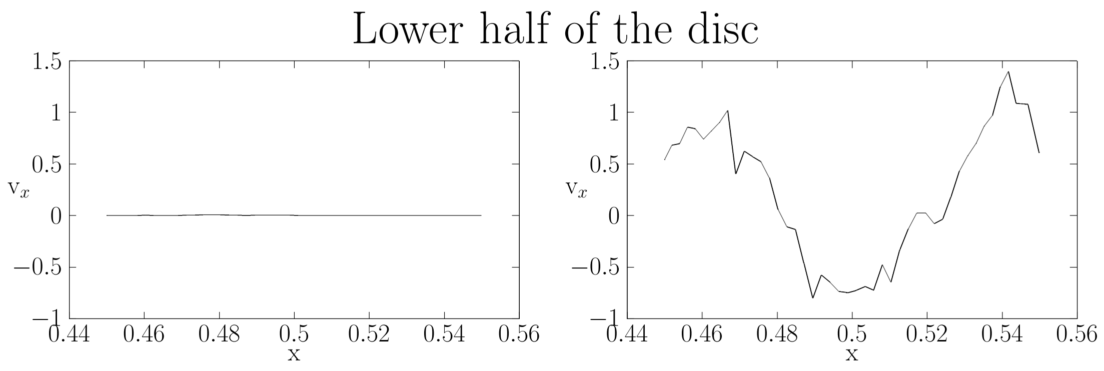


Figure 4.39: Plot of the horizontal component of the velocity along the disc - fitted and unfitted algorithm - Re 40, lower half of the disc

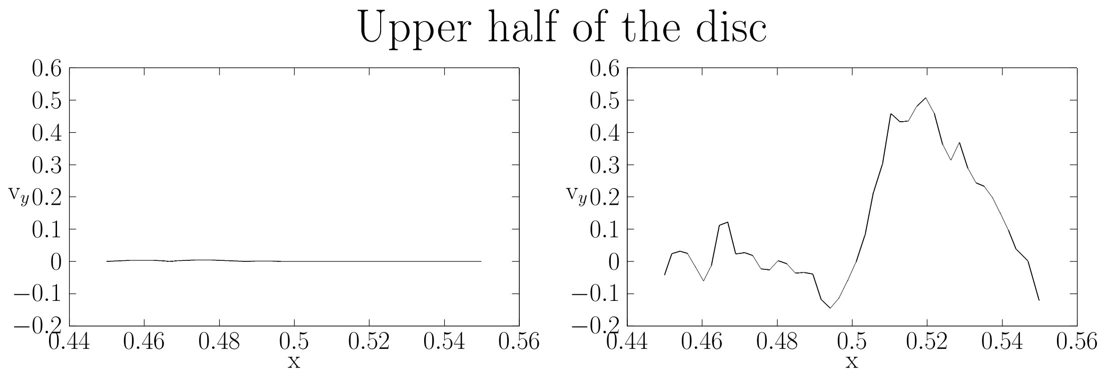


Figure 4.40: Plot of the vertical component of the velocity along the disc; fitted and unfitted algorithm; Re 40, upper half of the disc

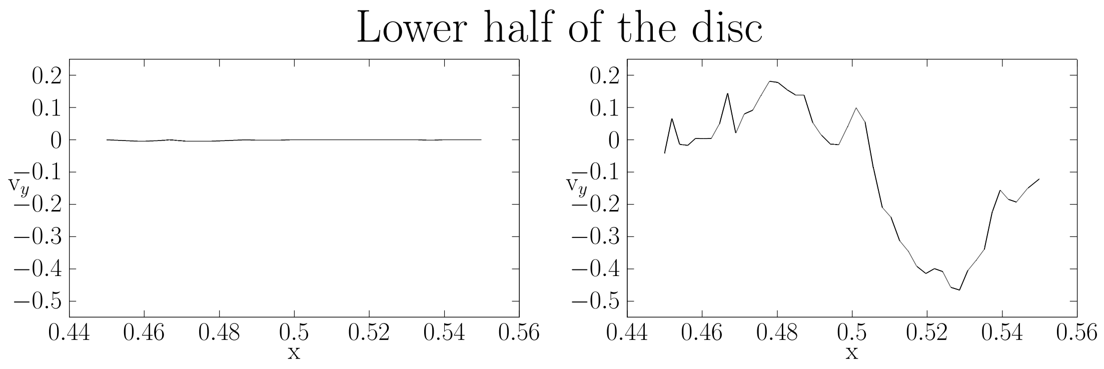


Figure 4.41: Plot of the vertical component of the velocity along the disc; fitted and unfitted algorithm; Re 40, lower half of the disc

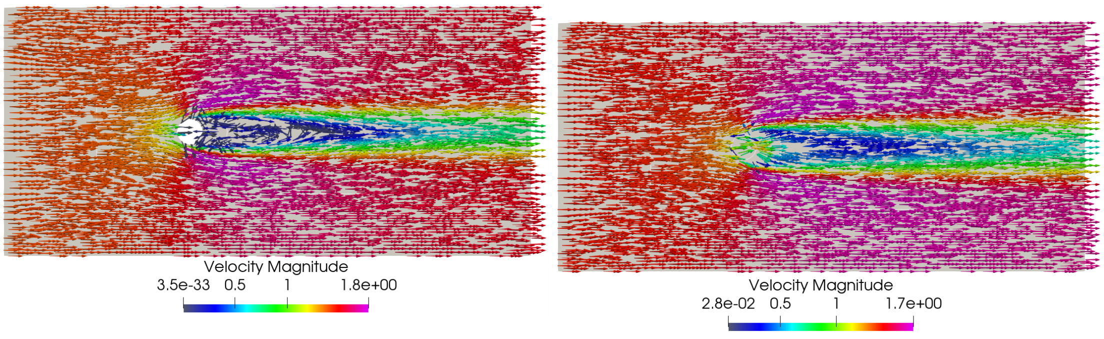


Figure 4.42: Plot of streamlines of velocity field; Navier-Stokes fitted and unfitted algorithms; Von Kármán street - Re = 40

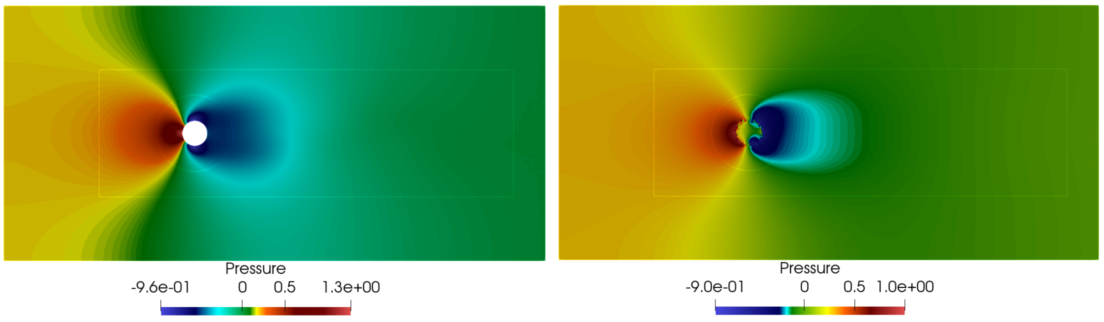


Figure 4.43: Plot of pressure; Navier-Stokes fitted and unfitted algorithms; Von Kármán street; Re = 40

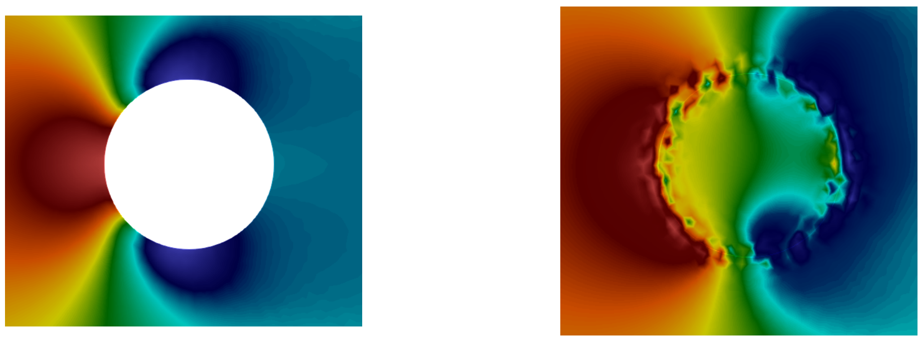


Figure 4.44: Plot of zoom of the pressure; Navier-Stokes fitted and unfitted algorithms; Von Kármán street; Re = 40

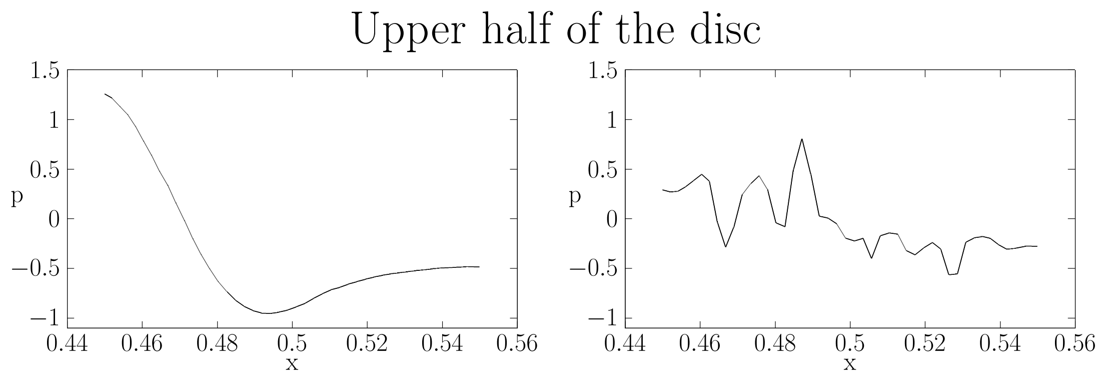


Figure 4.45: Plot of the pressure along the disc; fitted and unfitted algorithm; Re 40, upper half of the disc

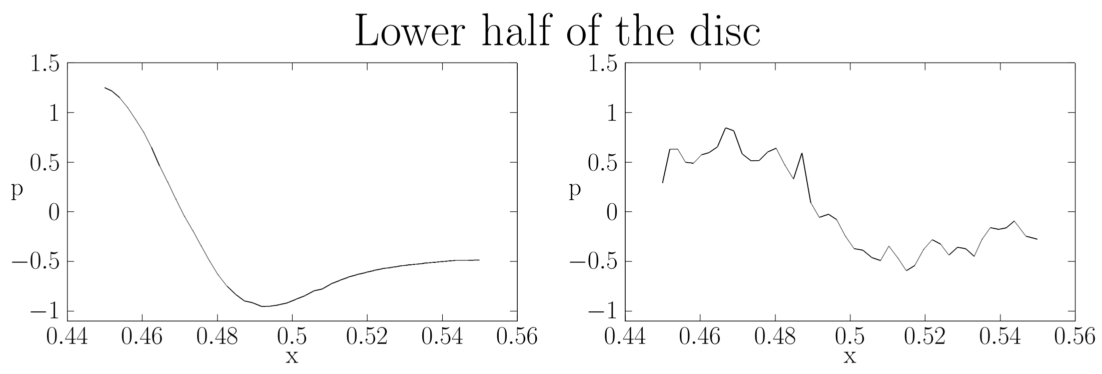


Figure 4.46: Plot of the pressure along the disc; fitted and unfitted algorithm; Re 40, lower half of the disc

The second test is performed using the same mesh as the previous case. The Reynolds number chosen is 100.

Re $=$ 100, $v_x = 3.5$ on Γ_4 and $v_y = 0$ on Γ_1 and Γ_3, T $=$ 15, dt $=$ 0.05, $\epsilon = 0.1$

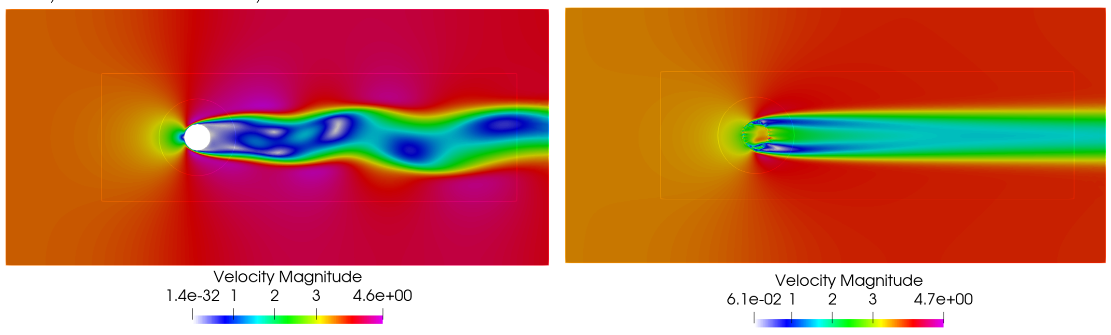


Figure 4.47: Plot of velocity; Navier-Stokes fitted and unfitted algorithms; Von Kármán street; Re $=$ 100

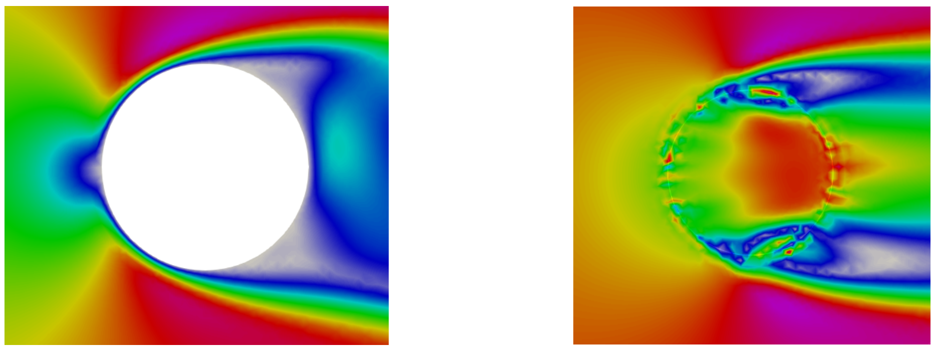


Figure 4.48: Plot of zoom of the velocity; Navier-Stokes fitted and unfitted algorithms; Von Kármán street; Re = 100

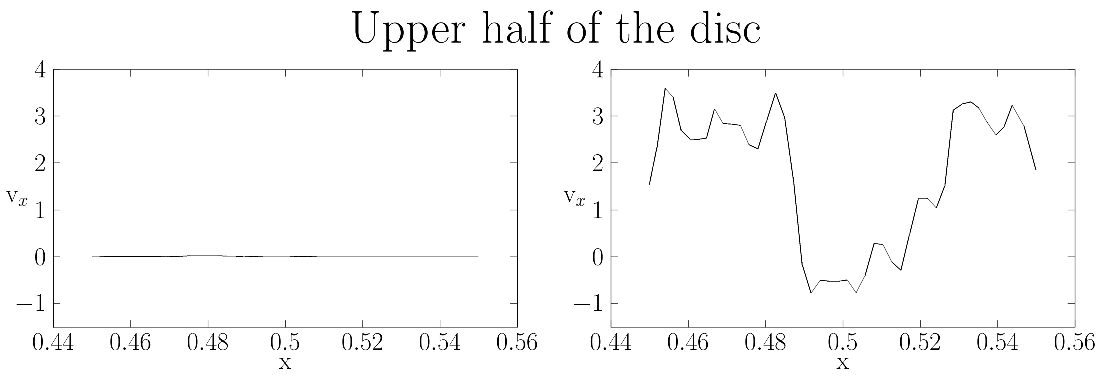


Figure 4.49: Plot of the horizontal component of the velocity along the disc; fitted and unfitted algorithm; Re 100, upper half of the disc

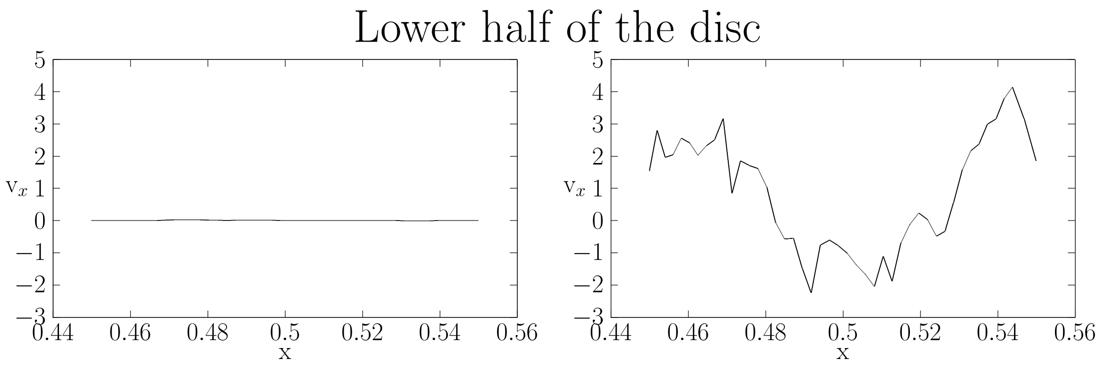


Figure 4.50: Plot of the horizontal component of the velocity along the disc; fitted and unfitted algorithm; Re 100, lower half of the disc

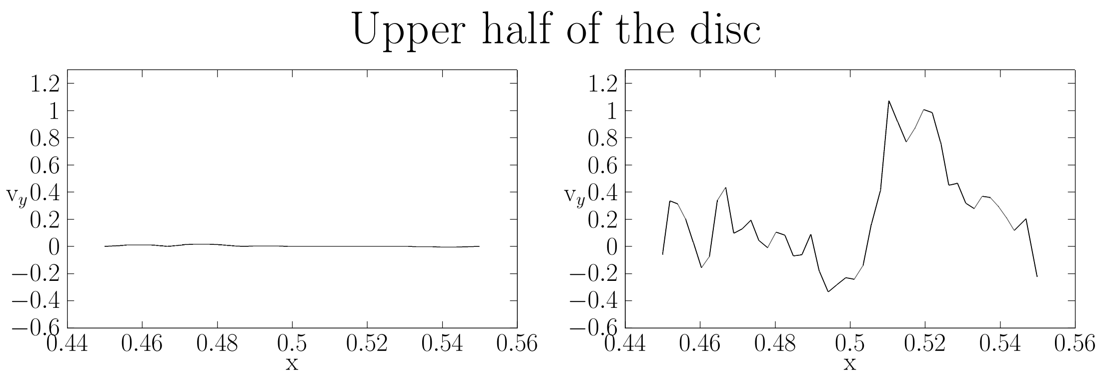


Figure 4.51: Plot of the vertical component of the velocity along the disc; fitted and unfitted algorithm; Re 100, upper half of the disc

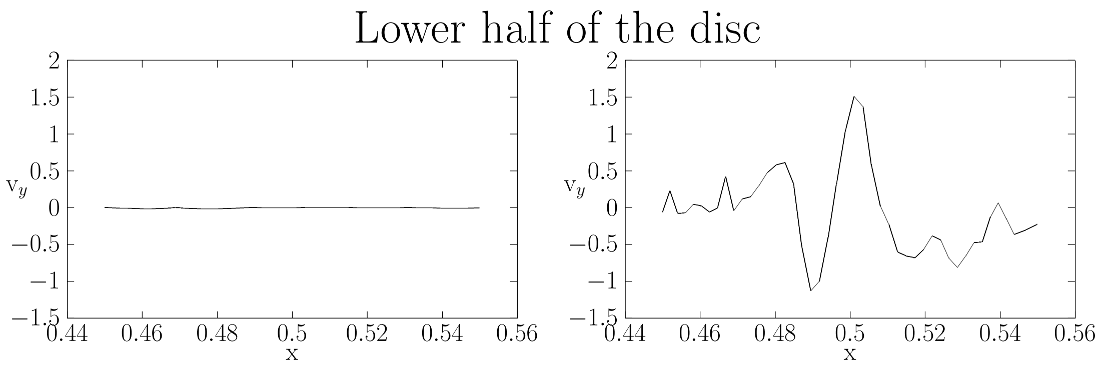


Figure 4.52: Plot of the vertical component of the velocity along the disc; fitted and unfitted algorithm; Re 100, lower half of the disc

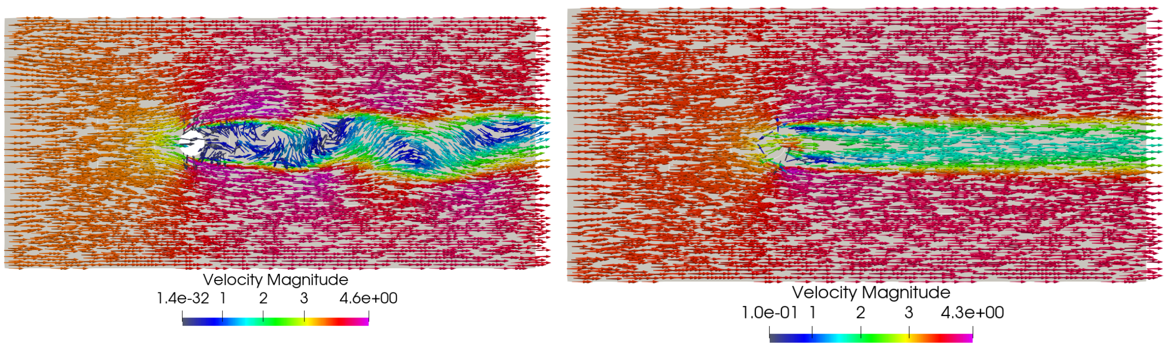


Figure 4.53: Plot of streamlines of velocity field; Navier-Stokes fitted and unfitted algorithms; Von Kármán street; Re = 100

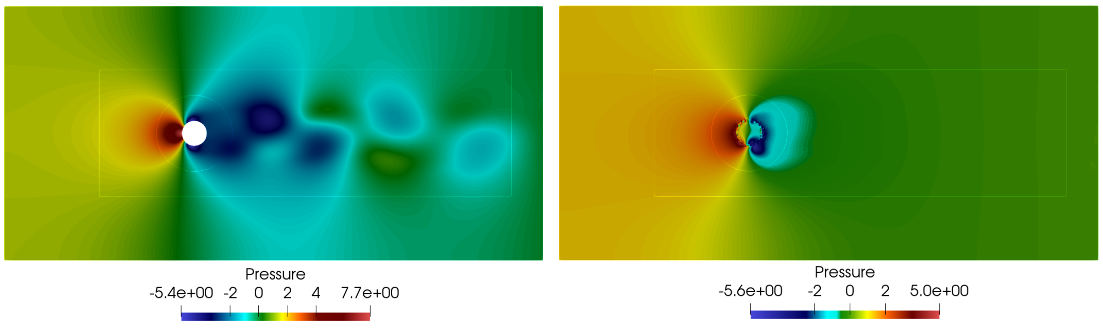


Figure 4.54: Plot of pressure; Navier-Stokes fitted and unfitted algorithms; Von Kármán street; Re = 100

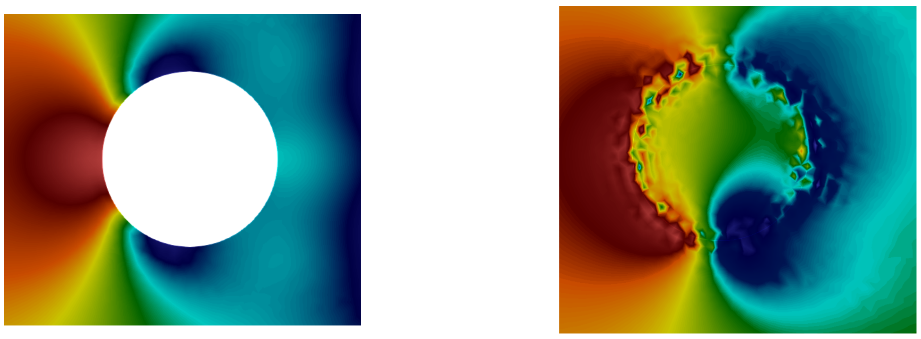


Figure 4.55: Plot of zoom of the pressure; Navier-Stokes fitted and unfitted algorithms; Von Kármán street; Re = 100

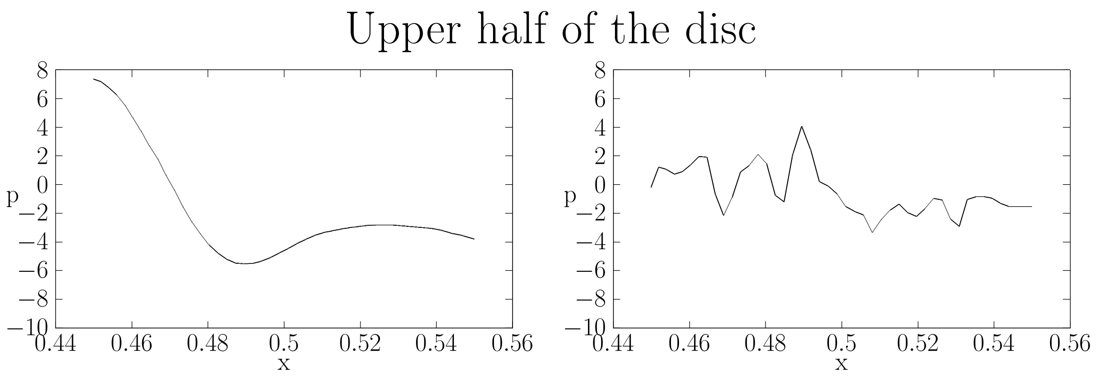


Figure 4.56: Plot of the pressure along the disc; fitted and unfitted algorithm; Re 100, upper half of the disc

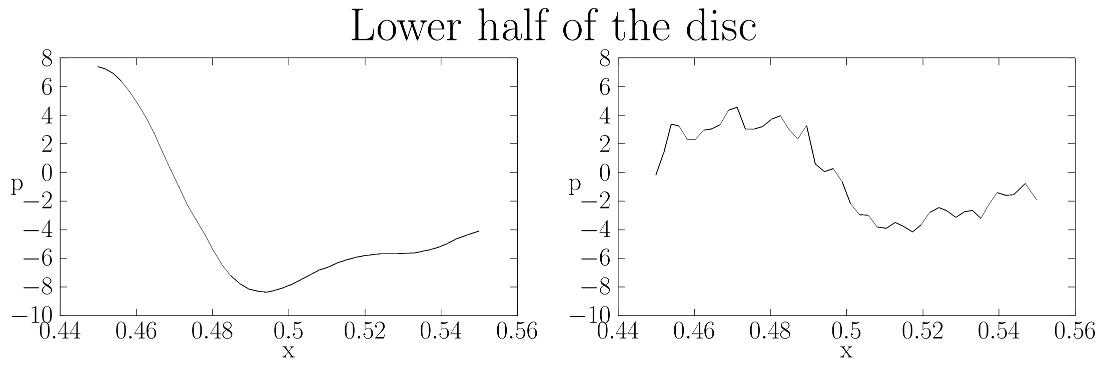


Figure 4.57: Plot of the pressure along the disc; fitted and unfitted algorithm; Re 100, lower half of the disc

Considering the results obtained from the velocity profiles first, we notice that, while for the "fitted" model we have values always very close to zero, both in vertical and horizontal direction and both for the upper and the lower semicircle of the disc, instead in the "unfitted" model the values result to be much higher, until reaching the values of the input velocity, that is 1.4 for Re = 40 and 3.5 for Re = 100. Above all, the major discrepancy of the velocity results occurs at the fluid-structure interface on the downstream side of the disc, as can be seen from the images Fig.4.36 and its zoom Fig.4.37 and Fig.4.47 and its zoom Fig. 4.48. We notice also that after the interaction with the disc, the unfitted model fails to reproduce a region where the velocity is lower, almost null, compared to that present in the rest of the domain.
Concerning the pressure, we notice that both in the case of Re = 40 and in that of Re = 100, the "Unfitted" model does not succeed to mimic the "Fitted" one. Especially in the case of Re = 100, we notice that in the reduced model no vortexes are formed, as it happens for the full model. In fact, the vortices are originated because ∇p along the circumference of the disc drops substantially and the negative pressure generates instability. In particular we observe a pressure drop from 8 to -6 for the upper semicircle and from 8 to -8 for the lower one. For

the reduced model, first of all, there is an important difference in the results of the upper and lower semicircles: the pressure profile of the inferior semicircle is more similar to how it should be, reaching a pick of almost 6 on the left part of the disc, decreasing gradually until a value of -6 on the left part. The pressure on the superior part of the disc, instead, has a fluctuating trend that doesn't allow the formation of a vortex to happen, fact that it is noticed instead for what concerns the lower part, shown with a zoom in the image Fig. 4.55.

Chapter 5

Conclusions

My thesis aims to create a 2D-0D dimensional model for the reduction of fluid-structure interaction. We want to reproduce the flow of an incompressible fluid (2D) around a disc (0D), which has to be modelled as a point due to the imposed zero dimensionality. This was our first problem to face and, in order to solve it, we created a suitable mathematical model that enforce the boundary conditions between the solid and the fluid by means of Lagrange multipliers. We proposed a numerical discretization to address our problem as a finite element problem and we used FreeFem++ as solver. We wrote the algebraic formulation in a matrix form in order to compute the numerical solution, obtaining a block matrix that represents the fluid structure interaction. We created an algorithm that uses Gauss point interpolation to find the intersection between the mesh of the fluid domain, which does not conform with the disc, and the disc itself. To model the disc we created a parameterized circumference, approximated by a set of points connected by segments. We have thus computed the intersection of each segment of the parameterized disc and the edges of the mesh of the fluid domain. It was also necessary to find a way to consider a pointwise velocity for the disc modelled as a point and, to deal with this issue, we replaced the velocity of the fluid with the average velocity of the fluid on the fluid-solid interface. We have consequently

carried out tests for the assessment of the reduced model, namely "Obstacle problem with Stokes flow", compared with a full model. we showed that, in this case, the two models produce comparable results. Then we modified the the computational model of the fluid structure interaction,to take into account of the constitutive equation for a disc which may move with a linear elastic constraint. We tested the respective algorithm reproducing different conditions around the fluid, such as the Couette's flow, and then we studied the movement of the disc. Finally, we completed the mathematical model considering a fluid governed by Navier-Stokes equations and we reproduced the same test of the previous case studying the movement of the disc to verify the validity of the code. The last test carried out consists in reproducing Von Kármán street and compare the results of the developed algorithm of the reduced model with the simulations based on the full 2D model. This last test gave the most interesting results because it highlights the weaknesses of the model and consequently suggest the problems to be solved to improve the model presented here.
In conclusion, the reduced method fails to capture the decrease in pressure which cause the origin of the vortexes of Von Kármán street. We need to find a more suitable formulation so that the pressure gradient is captured correctly.

Chapter 6

Code

```

int np;
np = 200;
real[int] xx(np+1);
real[int] yy(np+1);
int i;
int j;
int TNumber;
real[int] TN(500);
int ee;
int k;
int l;
int d;
int nh;
real[int] S1P0(2);
real[int] S1P1(2);
real[int] S2P0(2);
real[int] S2P1(2);
real[int] SP0(2);
real[int] SP1(2);
real[int] INTX(500);
real[int] INTY(500);
real[int] I2(2);
int TNumberw;
int TNumberq;
TNumberq= 500;
real[int] I0(2);
real[int] I1(2);
nh = 0;
real r = 0.05;

for (int cir = 0; cir < np+1; cir ++)
  {
    xx[cir] = r*cos(2*pi/np*cir)+0.5000005;
    yy[cir] = r*sin(2*pi/np*cir);
  }

border C01(t=0,1.0){x=t*4.25-1.0;y=-1.0;label=1;};
border C02(t=0,1.0){x=3.25;y=2.0*t-1.0;label=2;};
border C03(t=0,1.0){x=3.25-4.25*t;y=1.0;label=3;};
border C04(t=0,1.0){x=-1.0;y=1.0-2.0*t;label=4;};
border co(t=0,2*pi){x=cos(t)*0.3+0.5;y=sin(t)*0.3;};

border bo(t=0, 2*pi){x=r*cos(t) + 0.5; y=r*sin(t) ; label = 5; }

border C11(t=0,1.0){x=t*3.25-0.25;y=-0.5;};
border C12(t=0,1.0){x=3.0;y=1.0*t-0.5;};
border C13(t=0,1.0){x=3.0-3.25*t;y=0.5;};
border C14(t=0,1.0){x=-0.25;y=0.5-1.0*t;};

mesh Th = buildmesh(C01(140)+C02(40)+C03(140)+C04(40)+bo(90)+co
    (140)+C11(250)+C12(100)+C13(250)+C14(100));

plot(Th, [xx, yy], grey=false, wait = 0);

real SMALLNUM = 1.e-10;

func real perp(real [int] u, real [int] v) {
     return u[0]*v[1]-u[1]*v[0];
}

func real dot(real [int] u, real [int] v) {
     return (u'*v);
}

func int
        inSegment(real [int] P, real [int] SP0, real [int] SP1)
        {
            if (!(SP0[0] == SP1[0])) {    // S is not vertical
                if (SP0[0] <= P[0] && P[0] <= SP1[0])
                    return 1;
                if (SP0[0] >= P[0] && P[0] >= SP1[0])
                    return 1;
            }
            else {    // S is vertical, so test y coordinate
                if (SP0[1] <= P[1] && P[1] <= SP1[1])
                    return 1;
                if (SP0[1] >= P[1] && P[1] >= SP1[1])
                    return 1;
            }
            return 0;
         }
```

```

func int intersect2DSegments(real [int] S1P0, real [int] S1P1, real [int] S2P0, real [int] S2P1, real [int] I0, real [int] I1)
          {
              real [int]    u = S1P1 - S1P0;
              real [int]    v = S2P1 - S2P0;
              real [int]    w = S1P0 - S2P0;
              real    D = perp(u,v);

              // test if they are parallel (includes either being a point)
              if (abs(D) < SMALLNUM ) {          // S1 and S2 are parallel
                  if (perp(u,w) != 0 || perp(v,w) != 0) {
                      return 0;                  // they are NOT collinear
                  }
                  // they are collinear or degenerate
                  // check if they are degenerate points
                  real du = dot(u,u);
                  real dv = dot(v,v);
                  if (du==0 && dv==0) {          // both segments are points
                      if ((S1P0[0] != S2P0[0]) || (S1P0[1] != S2P0[1]))          // they are distinct points
                          return 0;
                      I0 = S1P0;                 // they are the same point
                      return 1;
                  }
                  if (du==0) {                   // S1 is a single point
                      if (inSegment(S1P0, S2P0, S2P1) == 0)  // but is not in S2
                          return 0;
                      I0 = S1P0;
                      return 1;
                  }
                  if (dv==0) {                   // S2 a single point
                      if (inSegment(S2P0, S1P0, S1P1) == 0)  // but is not in S1
                          return 0;
                      I0 = S2P0;
                      return 1;
                  }
                  // they are collinear segments - get overlap (or not)
```

```
               real t0, t1;                   // endpoints of S1 in eqn for S2
               real [int] w2 = S1P1 - S2P0;
               if (v[0] != 0.) {
                       t0 = w[0] / v[0];
                       t1 = w2[0] / v[0];
               }
               else {
                       t0 = w[1] / v[1];
                       t1 = w2[1] / v[1];
               }
               if (t0 > t1) {                  // must have t0 smaller than t1
                       real t=t0; t0=t1; t1=t;    // swap if not
               }
               if (t0 > 1 || t1 < 0) {
                   return 0;     // NO overlap
               }
               t0 = t0<0? 0 : t0;              // clip to min 0
               t1 = t1>1? 1 : t1;              // clip to max 1
               if (t0 == t1) {                 // intersect is a point
                   I0 = S2P0 + t0 * v;
                   return 1;
               }

               // they overlap in a valid subsegment
               I0 = S2P0 + t0 * v;
               I1 = S2P0 + t1 * v;
               return 2;
             }


             // the segments are skew and may intersect in a point
             // get the intersect parameter for S1
             real     sI = perp(v,w) / D;
             if (sI < 0 || sI > 1)              // no intersect with S1
                 return 0;

             // get the intersect parameter for S2
             real     tI = perp(u,w) / D;
             if (tI < 0 || tI > 1)              // no intersect with S2
                 return 0;
```

```

                I0 = S1P0 + sI * u;               // compute S1 intersect point
                return 1;
          }


for (k = 0; k < (xx.n-1); k ++) // loop on the segments
{

  TNumber = Th(xx[k],yy[k]).nuTriangle;
  TN[nh] = TNumber; // vector containing the triangle's numbers with points of the segment

  nh = nh +1;

  INTX[l] = xx[k]; // vector with x coordinates of my intersection
  INTY[l] = yy[k]; // vector with y coordinates of my intersection

  while (TNumber != Th(xx[k+1], yy[k+1]).nuTriangle) // while the element is different from the element of the next point
    {

    for(i = 0; i< 3; i++) // loop on edges of each triangle
      {

        if (i == 0) {d = 1;}
        if (i == 1) {d = 2;}
        if (i == 2) {d = 0;}

          S1P0[0] = xx[k];
          S1P0[1] = yy[k];
          S1P1[0] = xx[k+1];
          S1P1[1] = yy[k+1];
          S2P0[0] = Th[TNumber][i].x;
          S2P0[1] = Th[TNumber][i].y;
          S2P1[0] = Th[TNumber][d].x;
          S2P1[1] = Th[TNumber][d].y;

          real z = intersect2DSegments(S1P0, S1P1, S2P0, S2P1, I0, I1);

          if (i == 0) {ee = 2;}
          if (i == 1) {ee = 0;}
          if (i == 2) {ee = 1;}

        if (((abs(I0[0] - I2[0]) > SMALLNUM) || (abs(I0[1] - I2[1]) > SMALLNUM)) && ((I0[0] != 0) || (I0[1] != 0) ) ) // condition point different from the previews point found

        {

        TNumberw = Th[TNumber].adj((ee)); // adjacent triangle of triangle TNumber from edge ee

        l=l+1;

        INTX[l] = I0[0]; // xcordinate of intersection point
        INTY[l] = I0[1]; // ycordinate of intersection point

        I2[0] = I0[0];
        I2[1] = I0[1];

        I0[0] = 0;
        I0[1] = 0;

        TNumberq = TNumber; // previews triangle number to avoid to go back with the intesection already founq
        TNumber = TNumberw;

        TN[nh] = TNumber;

        nh = nh +1;

        break;

        }

        else

        {

        continue;

        }
    }
  }

        l = l+1;

        INTX[l] = xx[k+1];
        INTY[l] = yy[k+1];
}

        int s;
```

```
            int t;

            s=1;

            for(t = 0; t < INTX.n; t++ ){
            if ((INTX[t] != 0) && (INTY[t] != 0)){

            s = s+1; // number of the intersections points found

            }
            continue;
            }

            int tt;

            real[int] NTF(s-2);

            NTF = TN;

real[int, int] M(2,2);
real[int] q1dPointRefNewxy1(2);
real[int] q1dPointRefNewxy2(2);
real[int] csieta1(2);
real[int] csieta2(2);
real detM;
real[int] x1(2);
real[int] alphaminusx1fir(2);
real[int] alphaminusx1sec(2);
real[int] basisfunc1(s);
real[int] basisfunc2(s);
real[int] deralpha(2);
real[int, int] invM(2,2);
real[int] basitot(s);
int ii;
int jj;
int ni;
real[int] PP0(2);
real[int] PP1(2);
int ll;
real[int] N1(3);
real[int] N2(3);
ll = 0;
real[int] omega(2);
omega = [1, 1];
real norm2alpha;


// 2d basis functions

```

```
func real phif(real[int] xi, int l) {
 if ( l == 0 )
    return 1.-xi[0]-xi[1];
 else if ( l == 1)
    return xi[0];
 else
    return xi[1];
  }

// 1d quadrature points
real [int] q1dPointRef(2);
q1dPointRef[0]    = -sqrt( 1. / 3. );
q1dPointRef[1]    =  sqrt( 1. / 3. );
real q1dWeightRef =  1.;
real [int,int] q1dPoint(2,2);

int nv = Th.nv;
real[int, int] C(3*nv,2);
C = 0.;

for(ii = 0; ii < s - 2; ii ++) // loop on the segment
{
      PP0[0] = INTX[ii];
      PP0[1] = INTY[ii];

      PP1[0] = INTX[ii + 1];
      PP1[1] = INTY[ii + 1];

      for (jj = 0; jj < 2; jj ++)
      {

      q1dPointRefNewxy1[jj] = (PP0[jj]+PP1[jj])/2 + ((PP1[jj]-PP0
   [jj])/2)*q1dPointRef[0]; // first Gauss point traslated

      q1dPointRefNewxy2[jj] = (PP0[jj]+PP1[jj])/2 + ((PP1[jj]-PP0
   [jj])/2)*q1dPointRef[1]; // second Gauss point traslated

      deralpha[jj] = ((PP1[jj] - PP0[jj])/2);

      }

      M = [ [ (Th[NTF[ii]][1].x - Th[NTF[ii]][0].x) , (Th[NTF[ii
   ]][2].x - Th[NTF[ii]][0].x) ],
             [ (Th[NTF[ii]][1].y - Th[NTF[ii]][0].y) , (Th[NTF[ii
   ]][2].y - Th[NTF[ii]][0].y) ]
           ]; // Matrix of trasformation coordinate

      invM = [ [ M(1,1) , - M(0,1) ],
               [ - M(1,0) , M(0,0) ]
```

```
         ]; // inverse of M

      detM = 1./((M(0,0)*M(1,1)) - (M(0,1)*M(1,0)));

      x1 = [Th[NTF[ii]][0].x, Th[NTF[ii]][0].y];

      alphaminusx1fir = (q1dPointRefNewxy1 - x1);
      alphaminusx1sec = (q1dPointRefNewxy2 - x1);

      csieta1[0] = (alphaminusx1fir[0]*invM(0,0)+alphaminusx1fir
    [1]*invM(1,0))*(detM);
      csieta1[1] = (alphaminusx1fir[0]*invM(0,1)+alphaminusx1fir
    [1]*invM(1,1))*(detM);
      csieta2[0] = (alphaminusx1sec[0]*invM(0,0)+alphaminusx1sec
    [1]*invM(1,0))*(detM);
      csieta2[1] = (alphaminusx1sec[0]*invM(0,1)+alphaminusx1sec
    [1]*invM(1,1))*(detM);

      norm2alpha = sqrt((deralpha[0])^2 + (deralpha[1])^2); //
    norm 2 of the alpha trasformation

      for(int kk = 0; kk <3; kk++){
        N1[kk] = phif(csieta1, kk); // shape functions with first
     Gauss point
        N2[kk] = phif(csieta2, kk); // shape functions with
    second Gauss point
        ll = ll+1;
        int ig = Th[NTF[ii]][kk];
          C(3*ig,0) += (1./(2*pi*r))*(N1[kk]*norm2alpha*omega[0]
    + N2[kk]*norm2alpha*omega[1]); // sum of the basis functions
    integrated on each subsegment with two Gauss points
        C(3*ig+1,1) += (1./(2*pi*r))*(N1[kk]*norm2alpha*omega[0]
    + N2[kk]*norm2alpha*omega[1]);
      }
}

real T=10;
real dt = 0.05;
int ntime = T/dt;
real time=0;
real gamma = 0.01;
real m=1;
real k1=10;
real k2=10;
real d1 = 0;
real d2 = 0;
real d1old=0;
real d2old=0;
real d1oldold=0;
real d2oldold=0;
real mu=0.0035;

// Fespace
fespace Vh(Th, P1);
int nVh = Vh.ndof;
Vh u, v;
Vh uu, vv;
Vh uold, vold;
Vh p;
Vh pp;
Vh udata, vdata;
Vh nul;
Vh veln;

fespace Rh(Th,P0);
Rh tauK, uxK, uyK;

macro e11(u,v) dx(u) //
macro e22(u,v) dy(v) //
macro e12(u,v) ( (dx(v) + dy(u))*0.5) //
macro dn(u) ( dx(u)*N.x+dy(u)*N.y ) //
macro div(u,v) (dx(u)+dy(v)) //

func real negP(real tt) {
  if ( tt < 0 )
     return tt;
  else
    return 0.0;
}

real [int] auxf1(nVh);
real [int] auxf2(nVh);

fespace Yh(Th, [P1, P1, P1]);


// string flname1 = "x_plot_NS";
// string flname2 = "y_plot_NS";
// ofstream valuesx(flname1);
// ofstream valuesy(flname2);

udata = 3.5;
udata = 0;

for (int g = 0; g < ntime; g++){

time += dt;

```

```
varf aa ([u, v, p], [uu, vv, pp])
    = int2d(Th)( ((u*uu+v*vv)/dt))
    + int2d(Th)(2.0*mu*( e11(u,v)*e11(uu,vv) + e22(u,v)*e22(uu,vv) + 2.0*e12(u,v)*e12(uu,vv) ) )
        + int2d(Th)(uu*(uold*dx(u) + vold*dy(u)) + vv*(uold*dx(v) + vold*dy(v)))
        - int2d(Th)(p*(div(uu, vv)))
        + int2d(Th)(div(u, v)*pp)
    // + int2d(Th)(gamma*hTriangle^2*(dx(p)*dx(pp) + dy(p)*dy(pp) ) ) // Brezzi - Pitkaranta
      + int2d(Th)(0.5*(uu*u + vv*v)*div(uold,vold))
      + int2d(Th)( tauK*([uold*dx(u)+vold*dy(u) + dx(p), uold*dx(v)+vold*dy(v) + dy(p)]'*[uold*dx(uu)+vold*dy(uu)+dx(pp), uold*dx(vv)+uold*dy(vv)+dy(pp)])) // Total stabilization
      - int1d(Th, 2) ( 0.5*negP(uold * N.x + vold* N.y)*(u*uu + v*vv)  )    // backflow stabilization
    + int1d(Th, 1, 3)(10e+20*v*vv )
    + int1d(Th, 4)(10e+20*u*uu + 10e+20*v*vv) //parabolic profile
      ;

  uxK = uold;
    uyK = vold;
    tauK= 0.1*(1.0/(sqrt(4.0/(dt^2) +4.0*(uxK^2 + uyK^2)/(hTriangle^2) + 16.0*mu*mu/(hTriangle^4))));

//udata = 21.875*y*(2-y)*4; // parable
//vdata = 0; // parable
udata = 3.5; // couette
vdata = 0;

varf rhsfuu(unused, uu)
  //= int1d(Th, 4)(1*uu) //pressure
  = int1d(Th, 4)(10e+20*udata*uu) //parabolic profile
    //+ int1d(Th, 1, 3)(1*uu) // velocity null Gamma 1 e 3
  //+ int1d(Th, 3)(10e+20*uu) // Couette
  //+ int1d(Th, 1)(-10e+20*uu) // Couette
  + int2d(Th)((1./dt)*uold*uu )
  ;

varf rhsfvv(unused, vv)
    = int1d(Th, 1, 3)(0*vv) //velocity null Gamma 1 e 3
  + int1d(Th, 4)(vdata*vv) //parabolic profile
  + int2d(Th)((1./dt)*vold*vv )
  ;

auxf1 = rhsfuu(0,Vh);

auxf2 = rhsfvv(0,Vh);

```

```
matrix A = aa(Yh, Yh, solver=UMFPACK);

real[int, int] C4(2,2);
C4 = 0;

real[int, int] LLM(2,2);
real[int, int] KM(2,2);
real[int, int] STM(2,2);

LLM = [ [-1,0],
    [0,-1]
    ];

real[int, int] MLM(2,2);

MLM =  (1/dt)*[ [-1,0],
         [0,-1]
      ];

real[int, int] EV(C.n, 2);

EV = 0;

KM = [ [(m/(dt^2)) + k1, 0],
    [0, (m/(dt^2)) + k2]
    ];

matrix FS = [ [A, C, EV],
          [C', C4, MLM],
          [EV', LLM, KM]
            ];

set (FS, solver = sparsesolver );

real[int] rhs(FS.n);
real[int] sol(FS.n);
real[int] L1(1);
real[int] L2(1);

for (int mm = 0; mm < nVh; mm++)
  {
    rhs[3*mm] = auxf1[mm];
    rhs[3*mm+1] = auxf2[mm];
  }

rhs[FS.n-4]=(-d1old/dt);
rhs[FS.n-3]=(-d2old/dt);
rhs[FS.n-2]=(m*((2*d1old-d1oldold)/(dt^2)));
rhs[FS.n-1]=(m*((2*d2old-d2oldold)/(dt^2)));
```

```


sol = FS^-1 * rhs;

for (int nl = 0; nl < u.n; nl ++)
  {
    u[][nl] = sol[3*nl];
    v[][nl] = sol[3*nl+1];
    p[][nl] = sol[3*nl+2];
  }

L1 = sol[sol.n-4];
L2 = sol[sol.n-3];
d1 = sol[sol.n-2];
d2 = sol[sol.n-1];

plot([u,v], p, wait=0,value=true,coef=0.1);

uold=u;
vold=v;
d1oldold = d1old;
d2oldold = d2old;
d1old = d1;
d2old = d2;

cout << time << "= t" << endl;

}
```

References

[1] M. Fernández. Coupling schemes for incompressible fluid-structure interaction: Implicit, semi-implicit and explicit. *Computers Fluids*, 55:59–108, 2011.

[2] L. Formaggia, A. Quarteroni, and A. Veneziani. *Cardiovascular Mathematics: Modeling and simulation of the circulatory system.* MS&A. Springer Milan, 2010.

[3] G. Galdi. *An Introduction to the Mathematical Theory of the Navier-Stokes Equations: Volume I: Linearised Steady Problems.* Springer Tracts in Natural Philosophy. Springer New York, 2013.

[4] M. Gerouache. Etude numérique ed l'instabilité de bénard-karman derrière un cylindre fixe ou en mouvement périodique. dynamique de l'ecoulement et advection chaotique. *Ph.D. thesis Ecole Polytechnique de l'Université de Nantes*, 2000.

[5] R. Glowinski, T.-W. Pan, T. Hesla, and D. Joseph. A distributed lagrange multiplier/fictitious domain method for particulate flows. *International journal of Multiphase Flow*, 25:755–794, 1999.

[6] F. Hecht. New development in freefem++. *J. Numer. Math.*, 20(3-4):251–265, 2012.

[7] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, E. Constantinescu, D. Estep, K. Evans, C. Farhat, A. Hakim, G. Hammond, G. Hansen, J. Hill, T. Isaac, X. Jiao, K. Jordan, D. Kaushik, E. Kaxiras, A. Koniges, K. Lee, A. Lott, Q. Lu, J. Magerlein, R. Maxwell, M. McCourt, M. Mehl, R. Pawlowski, A. P. Randles, D. Reynolds, B. Rivière, U. Rüde, T. Scheibe, J. Shadid, B. Sheehan, M. Shephard, A. Siegel, B. Smith, X. Tang, C. Wilson, and B. Wohlmuth. Multiphysics simulations: Challenges and opportunities. *The International Journal of High Performance Computing Applications*, 27(1):4–83, 2013.

[8] F. Laurino and P. Zunino. Derivation and analysis of coupled pdes on manifolds with high dimensionality gap arising from topological model reduction. *Mathematical Modelling and Numerical Analysis*, 06 2019.

[9] A. Placzek, J. Sigrist, and A. Hamdouni. Numerical simulation of an oscillating cylinder in a cross-flow at low reynolds number: Forced and free oscillations. *Computers Fluids*, 38:80–100, 01 2009.

[10] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations.* Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2009.

[11] S. Salsa. *Partial Differential Equations in Action: From Modelling to Theory.* Universitext. Springer Milan, 2008.

[12] O. Steinbach. *Numerical Approximation Methods for Elliptic Boundary Value Problems: Finite and Boundary Elements.* Texts in Applied Mathematics. Springer New York, 2007.