POLITECNICO
MILANO 1863

Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Dipartimento di Scienze e Tecnologie Aerospaziali

Performance Study of an Immersed Boundary DNS Code applied to the Flow around a Confined Circular Cylinder

candidato: **Gregorio Frassoldati**
matricola 876004

relatore: **Professor Maurizio Quadrio**

Tesi Magistrale in Ingegneria Aeronautica

Anno Accademico 2017/2018

Ringraziamenti

Desidererei ringraziare tutti coloro che hanno preso parte, direttamente o indirettamente, agli sforzi e al lavoro che hanno portato al compimento di questa tesi e del mio percorso di studi. Un primo ringraziamento va al Professor Maurizio Quadrio, un maestro presente quando richiesto, pronto e puntuale nelle sue risposte, e diretto nelle sue osservazioni; mi ha sempre dato modo di poter riflettere su me stesso e sul mio lavoro in modo chiaro e senza giri di parole.
In secondo luogo vorrei ringraziare i suoi collaboratori, che mi hanno sostenuto e risposto quando le domande mi sembravano troppe e troppo difficili; quindi grazie di cuore a Jacopo, Alessandro e Vanessa.
Assieme a loro, non posso dimenticarmi di chi mi ha sostenuto dietro le quinte di questo lavoro: la mia famiglia, i miei amici e i miei affetti tutti. Ai miei genitori, un modello di riferimento per quanto riguarda l'impegno e l'etica professionale, va un pensiero speciale. Alle mie sorelle e ai miei nipoti una menzione d'onore per il percorso di crescita che continuano a condividere con me.
A Myriam, che mi ha accompagnato e sostenuto in questi mesi dandomi serenità e fiducia, va un caldo ringraziamento per avermi sopportato fin qui.
Ai compagni di studio con cui ho condiviso il carico di questi anni va un un ultimo pensiero, ed in particolare all'amico Claudio, in memoria delle serate estive (e non) tra birra, patatine fritte, derivate ed integrali.

Sommario

La presente tesi è stata redatta ed affrontata cercando di perseguire un duplice scopo; di investigazione e ricerca fluidodinamica così come, al contempo, di studio prestazionale delle indagini numeriche effettuate.
In questa trattazione è stato studiato il problema della corrente turbolenta attorno ad un cilindro, confinato tra due pareti parallele di un canale piano periodico. L'elevato bloccaggio, oltre che la presenza stessa del corpo tozzo all'interno di un channel flow, contribuiscono a rendere il presente studio peculiare per il problema trattato e dunque per la fenomenologia della corrente.
Il problema è stato affrontato mediante simulazione numerica diretta, utilizzando due software profondamente diversi per algoritmi risolutivi ed elaborazione delle superfici. I risultati sono stati confrontati al fine di verificare se e quanto sia possibile un guadagno prestazionale in termini di costo computazionale utilizzando uno schema a contorni immersi, con il fine ultimo di migliorare le odierne possibilità di affrontare simulazioni DNS a fini di ricerca.
Questo confronto ha riguardato anche i risultati fluidodinamici delle simulazioni, mettendo in luce limiti e possibilità di entrambi i codici.
Difatti, il riferimento fornito da un codice opensource, verificato e ampiamente diffuso come OpenFOAM, ha permesso di portare avanti un confronto critico; le sue prestazioni ed il suo funzionamento sono stati analizzati e comparati a quelli di un codice a contorni immersi innovativo (chiamato CPL). Questo ha reso possibile evidenziare delle strategie più efficienti o più accurate, sia dal punto di vista della gestione del calcolo parallelo, che nella risoluzione delle equazioni di Navier-Stokes.

[1]

[1] **parole chiave:** simulazione numerica diretta, contorni immersi, corpo tozzo, bloccaggio, cilindro, CFD, turbolenza, scia

Abstract

Aim of the present thesis is both to investigate a particular flow and to study the performace of the numerical analysis.
The work consists in the study of the flow around a circular cylinder - confined within a periodical channel flow. The high blockage ratio - due to the presence of the body between the two confining walls - contributes to make peculiar the phenomena taking place.
The problem has been addressed via direct numerical simulation using two different softwares. These two programs differ in two important features: the algorithms used for solving the differential equations and the method used to process the surfaces.
The main objective is to verify if a performance gain is achievable using an immersed boundary scheme. Therefore, the computational cost of an immersed boundary code as well as of a classical one - OpenFOAM - was evaluated. The results are comprehensive of the analysis of the parallelization in both codes.
OpenFOAM indeed provided a useful reference for the flow study and the computing perfomance. It represents the current state of the art and it was used compared to the immersed boundary code - called CPL. This comparison included the results of the study of the fluid dynamics phenomena. This englightened the advantages and disadvantages of both softwares in terms of efficiency and accuracy.
The ultimate purpose is to develop the current possibilities to carry on DNS simulations in academic and industrial research.

[2]

[2]**keywords:** DNS, Immersed Boundary, bluff body, blockage ratio, cylinder, CFD, turbulence, wake

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Goal of this study is to apply an innovative method to the flow analysis in the specific scenario of the flow around a circular cylinder.
The main focus concerns the time gain in terms of computational cost by the immersed boundary DNS algorithm in comparison to a classical one.
The literature available about the phenomena confirms the results here acquired. This fact validates the consistency of a new algorithm to run direct numerical simulations. The data here reported would also represent a useful database, since this type of flow is still marginally addressed by numerical research.
Between the codes chosen to pursue this comparison, OpenFOAM is used as standard whilst CPL is the immersed boundary code.

1.1 Direct Numerical Simulations

There are two types of flows in nature: laminar flows and turbulent flows. The latter concerns a large part of the nowadays fluid dynamics research. Turbulent flows are in fact the most diffuse type of flows and the most interesting in industrial applications.
Navier-Stokes equations - which govern the (newtonian) fluid dynamics - do not present an analytical solution in turbulent situations. As explained in [15], this means that the fluid motion cannot be predicted beforehand, as only its statistics can be. Computational fluid dynamics is a common way to deal with these problems. However, into current engineering problems the Reynolds number is high enough to make the discretization incredibly dense and - thus - the computational cost intolerable.

A high computational cost is unbearable in industries or wherever the time is a key factor. That is why the most part of nowadays studies performed in industries and technological-frameworks address this problem modeling the turbulence. This reduces the computational cost in return for a less accurate solution. These models necessarily imply some sort of time-averaging, even when just for the smaller turbulence structures (like in large eddy simulations). That choice jeopardizes the possibility to investigate the physical phenomena at the base of turbulent flows.
Large eddy simulations are indeed widely spreading wherever a better resolution or a more reliable time-changing database is needed. Their advantage with respect to Reynolds Averaged Navier-Stokes equations simulations - or their unsteady version, URANS - are remarkable. This also comes with an acceptable increase in computational cost, since the turbulence structures most expensive to solve - the smaller ones - are modeled.
However, this again comes with the impossibility to study many of the turbulence features, which are related to small scales structures interacting with surfaces, boundaries and so on.

The main strategy available nowadays to study the flow's instantaneous data are direct numerical simulations. This type of simulations avoids the modeling phase and proceeds by discretization in order to have an accurate and reliable solution. The discretization must be refined enough to capture all the turbulence scales.
The Reynolds number used in this very study - as it will be explained - is higher than other similar past DNS studies, but still distant from the technological range. Indeed this is not the immediate purpose of this project.

DNS are today an academic research tool, and their outcome is - among others - mainly referred to geometries interesting theoretical-wise. Nevertheless, the accuracy of these simulations can be valuable in every scientific environment, and innovative techniques - as the immersed boundary method - could represent an essential tool from the progress viewpoint. Moreover, new pioneering studies have already started to apply this strategy into a diverse variety of problems.

1.2 Structure and reasons of the investigation

The investigation in this study has been realized through the editing of two DNS codes - OpenFOAM and CPL - , making them suitable to solve the same problem. They were programmed with the same mesh refinement and with the very same setup. From the results computed in the two simulations a comparison has been carried out, trying to examine the reasons beneath the differences between them.
One purpose was indeed to validate the simulations, in order to proceed with a second comparison on the performaces. The main objective of the whole project although would be the analysis of an advanced methodology, the immersed boundary method, when applied to the study of the flow around a bluff body.
The time gain this type of method should guarantee would be an important achievement. Of course this has to come with the accuracy of the results obtained with the direct simulation.

Chapter 2

Problem definition

2.1 Literature bibliographic references

The scientific literature references in this peculiar area are both rare and diverse. Some of the papers collected in the editing of this thesis represented an important starting point. In this section we will analyze the elements more focused on this particular subject.
A brief comparison of the references can be found in tab. 2.1 at the end of this section.

The resemblance of these studies regards mainly the Reynolds number investigated and the blockage ratio of the setup.
The Reynolds number (Re) is a dimensionless quantity, defined as the ratio of inertial forces to viscous forces. It comes in handy when it is needed to predict flow patterns in various flow situations. As a matter of fact, the same domain can present different types of flows as this parameter change value.

$$Re = \frac{U \cdot \delta}{\nu} \tag{2.1}$$

As depicted in (2.1), the inertial forces are represesented by the product of a reference velocity U and a reference length δ. On the other hand, the viscous forces are represented by the cinematic viscosity ν.
The blockage ratio can be defined as $\beta = D/2h$, D being the diameter of the cylinder and h the semi-height of the channel.

2.1.1 'Numerical Investigations of Flow Over a Confined Circular Cylinder'

In [11], the flow around a confined cylinder has been studied. The value of the blockage ratio is 0.5. The investigation has been carried through a DNS study performed on OpenFOAM. The dimensions of both the cylinder and the surrounding domain resemble the ones of our study. In these simulations the highest Reynolds number investigated is equal to 500.

A quantitative analysis of the flow investigates lift and drag coefficients - as well as Strouhal number - as a function of the Reynolds number. The phenomena generated by the interaction between the cylinder wake and the boundary layer of the walls have been briefly described.
The vortex shedding phenomenon have been described also related to the deformed structures that the blocking effects generates.
In [11] is also interesting to analyze the criteria used in OpenFOAM to discretize the domain and solve the governing equations.

2.1.2 'A Numerical Analysis of Fluid Flow around Circular and Square Cylinders'

[19] focused on the behaviour of the wake behind cylinders. Its study validates a numerical setup and building procedure for this kind of problems. However, its conclusion - although validated via literature references - have been reached with a RANS study (performed on ANSYS Fluent). This makes these results less relevant and useful with respect to other studies.
The cylinder shape effects were briefly investigated to enlighten the different phenomena between circular and square section. However, the most interesting section contains the results the wake length as a function of the Reynolds number.

2.1.3 'A numerical investigation of wall effects up to high blockage ratios on two-dimensional flow past a confined cylinder'

The results of [17] are just qualitatively useful in this study. In fact, the two-dimensional setup prevents the simulation to develop physical phenomena close to the ones which this study addresses.
Nevertheless, there are some interesting results about the transition from

symmetric to asymmetric vortex shedding as the blockage ratio varies.

2.1.4 'Numerical study of the blockage effects on viscous flow past a circular cylinder'

In [1] the finite element technique was used for the solution of steady and unsteady flows around a circular cylinder at Re = 106 for blockage ratios of 0.05, 0.15 and 0.25. It is shown that the flow remains steady in the form of two standing vortices behind the cylinder. The size of the vortices is seen to be reduced by the increase in blockage ratio. Differently, the separation angle increases with the blockage ratio.
The most interesting results concern the blockage effects for the accuracy of the results of a numerical solution. When meshes with high blockage ratios are used for the reduction of computational time required, the values of the parameters may be considerably different from those of an unbounded solution.

2.1.5 'Stability of weak confined wake behind a cylinder in fully developed turbulent channel flow'

[10] investigates the instability of turbulent wake flow in a confined turbulent channel. Confined wakes are found to retain their mean velocity profile for a considerable downstream distance.
The basic aim of this paper was to investigate the correlation between the turbulence in the wake region and the inflection points in the wake region using stability theory, and d/h = 0.2 was chosen.
The numerical results aren't particularly useful though, since they were obtained by solving the Orr-Sommerfeld equation. Anyway, the experimental results are interesting and were obtained by introducing organized disturbances in the wake and tracking these downstream.

2.2 Introductive formulation of the problem

Into the channel in which our simulation takes place, it is straight-forward to define a cartesian coordinate system. The streamwise coordinate will be x, the spanwise y and the wall normal coordinate will be z. The velocity components along these directions will thus be called u, v and w; the velocity vector will be recalled as $\mathbf{u}$.

It has to be specified that in the case of a channel flow at constant flow rate the relevant Reynolds number becomes the so-called *bulk Reynolds number*.

$$Re = \frac{U \cdot \delta}{\nu} \rightarrow Re_B = \frac{U_B \cdot D}{\nu} \tag{2.2}$$

This is the case of our simulation. As presented in (2.2), the reference velocity is the *bulk velocity* (U_B). It can be defined as the average velocity of the single exact velocities at different locations in the same cross-section of the flow. Moreover, the characteristic length becomes D - the cylinder's diameter.
This nomenclature being established, the governing equations of the system can be written in a handy way. In the simulations is considered an incompressible flow and the problem is addressed non-dimensionally. Therefore, it will be used a version of the Navier-Stokes equations like the one reported in (2.2). The system consists in the mass conservation equation and the momentum equation.

$$\begin{cases} \nabla \cdot \mathbf{u} = 0 \\ \\ \frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \frac{1}{Re} \nabla^2 \mathbf{u} + \mathbf{F} \end{cases}$$

Author	**Paper**	**Blockage ratio**	**Reynolds number**	**Method**
Anagnosto-poulos et al.	Numerical study of the blockage effects [...] [1]	0.05 - 0.25	106	irrotational numerical simulation
Yuce et al.	A numerical analisys of fluid flow around [...] [19]	0.125	$2 - 4 \cdot 10^6$	RANS
Sahin et al.	A numerical investigation of wall effects [...] [17]	0.1 - 0.9	0 - 280	velocity-only finite volumes simulation
Kumar et al.	Stability of weak con-fined wake behind a [...] [10]	0.1	14200	experimental and theoreti-cal
Mathupriya et al.	Numerical Investiga-tions of Flow Over a [...] [11]	0.5	200 - 500	DNS

Table 2.1: *main articles published about confined cilinders' wake*

2.3 General Setup of the Problem

The general setup is built as a classical, periodical and smooth-surfaced channel flow, in which a bluff body is built-in. The numerical simulation treats the flow directly, without any modeling.
The bluff body is a circular cylinder and it is confined with an high blockage ratio. The peculiar flow around the body at a turbulent Reynolds is the main focus of the study. The cylinder axis is directed spanwise, along the y-axis. The incompressible flow motion is treated in a non-dimensional way, and peculiar boundary conditions are adopted inside the carried DNS.

Geometry The geometry is made out of two parts. The first one is a classical-layout channel with the streamwise coordinate x as main dimension. The upper and lower walls are considered physical walls. The second part is a cylinder whose axis is spanwise-oriented. More in depth, the channel has dimensions 6π x $8D$ x $2D$ x D, D being the characteristic length of the cylinder - the diameter. As can be seen in 2.1, the cylinder is placed at 1/6 of the length of the channel, symmetrically with respect to the wall-normal dimension.

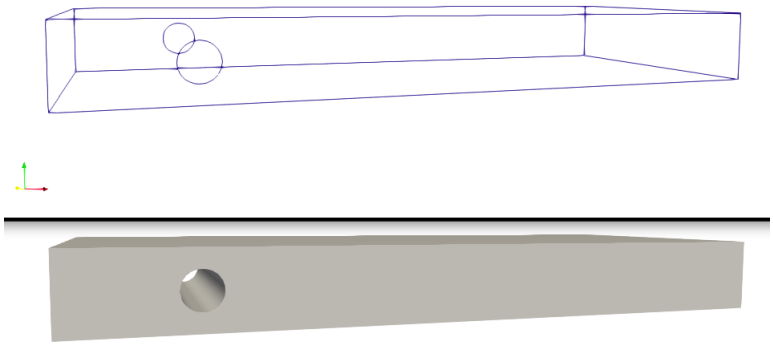


Figure 2.1: *Geometry of the domain object of the study: edges (top picture) and surfaces (bottom picture)*

The problem is addressed by the codes in a non-dimensional way, constructing the domain with a diameter $D = 1$, thus making the total height of the channel equal to 2 and the whole domain 6π x 8 x 2. A Reynolds

number equal to 750 is imposed in both simulations.
The reference length here coincides with the diameter D, thus making the blockage ratio equal to $D/(2h)$ - as h is conventionally half of the total height of the channel. In our geometry the blockage ratio is equal to 0.5 which is a rather high value, also compared with other studies performed in this field. It leads to particular and characteristic phenomena in the structure of the wake.

Fluid Dynamics Quantities The results presented in this work are obtained with conveniently imposed values of $U_B = 1$, $D = 1$ and $Re = 750$. It can be derived from (2.2) that the non-dimensional cinematic viscosity will be equal to $\nu = 1/750$.
This bulk velocity value is streamwise oriented; this is particularly relevant since the flow is bound by a Constant Flow Rate (CFR) condition, imposed indeed through the U_B on the entire domain.
It is worth reminding that here units of measure have no physical meaning. The goal to achieve was the similarity in the behaviour of the flow garanteed by the sole value of the Reynolds number. However, the very first simulation performed on OpenFOAM had the reference viscosity of air at 20°C (thus $\nu = 1.5x10^{-}5[m^2/s]$); the geometry was built trying to maintain a $\frac{U_B}{D}$ ratio similar to the one present in CPL - therefore $D = 0.1125[m]$ and $U_B = 0.1[m/s]$. This would have meant a comparison of results which would not have been as direct as the one we're portraying here.

Boundary Conditions In streamwise and spanwise directions the boundary conditions are periodical - *cyclic* on OpenFOAM - . Wall boundary conditions are imposed over the physical surface of the upper and lower surfaces of the confining box, as well as on the cylinder surface.
Concretely, the wall boundary condition translates into no-slip condition for velocity and a zero gradient condition for pressure.

2.4 Ideal results

A bluff body can be defined as a body that presents separated flow over a substantial part of its surface. This comes as a result of its shape with respect to the incident flow, if not its overall shape. The section perpendicular

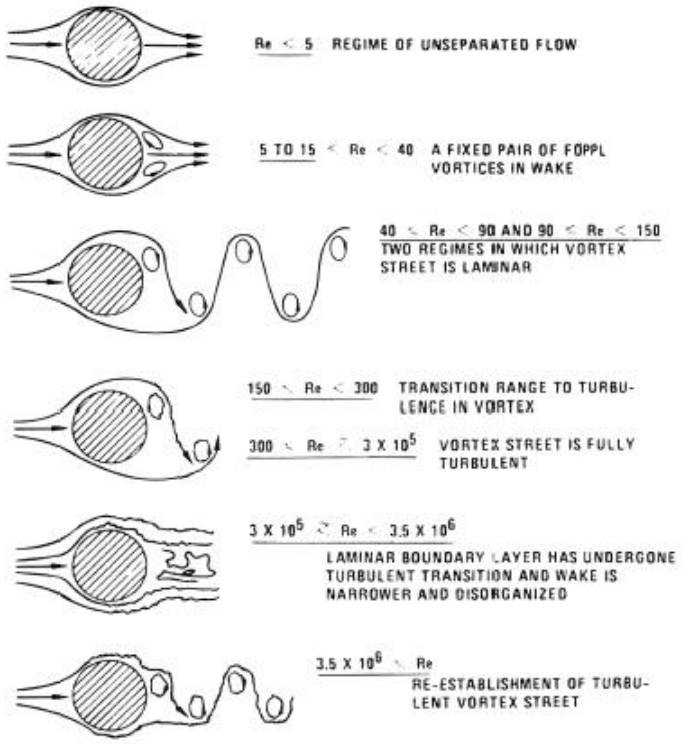


Figure 2.2: *Brief scheme of a circular cylinder wake behaviour as a function of the Reynolds number*

to the main direction of the flow determines the reference surface and the reference shape of the body.
Both simulations gave back a flow description coherent with the structures and phenomena expected from the theoretical results. In particular, the flow around a cylinder for Reynolds numbers around 750, is expected to present a large separation behind the body, with the periodic detachment of turbulent eddies.

Without interferences from the nearby walls we would normally see turbulent eddies detaching from the cylinder surface at a periodical vortex shedding frequency, giving the wake a precise and organized structure. The word "organized" has to be managed carefully though, since from the theory it is well known that the flow develops a turbulent wake starting from

Reynolds numbers close to 300.
Indeed, from Reynolds about 200 the formation of large scale instabilities transform the unsteady, periodical wake into turbulent. In this situation, the Von Karman wake still holds, with the difference that the eddies are not anymore laminar. Further on, increasing the Reynolds number, even small scale instabilities arise.
Ideally speaking, the separation point where the vortices generate is expected to be seen on the lateral surface of the cylinder at about 80 degrees with respect to the mean velocity direction.
These phenomena are generally recollected while considering the vortex pairing. Along with that, we expect the laminar boundary layer separation to generate a turbulent sheet: the shear layer separates, the transition to turbulent takes place and then the vortex originates. As explained in [6], the vortex tubes are generated from the vortex sheets decay because of the Kevin-Helmholtz instability.

However, it is important to recall that from this ideal situation we can foresee only a few preliminary characteristics. There are many important features which will determine a very different situation from the ideal one; the periodic boundary conditions, the presence of the walls and the blockage ratio are the most important.

Chapter 3

Description of the two numerical codes

In this study two different codes have been used. The first one - OpenFOAM - represents the current state of the art in the scientific landscape, whereas the latter one - CPL - is internally developed and profoundly innovative.
They address the problem and its solving process in different ways. The differences regard the geometry ad surfaces management, the solving schemes and even parallelization and processors communication. These differences make CPL's performances superior in comparison with OpenFOAM's, making the immersed boundary method a concrete enhancement in solving techniques.
From the viewpoint of direct simulations, this represents an important and needed progress.

3.1 OpenFOAM

Originally born as a partial differential equations solver, *OpenFOAM (**Open** **F**ield **O**peration **A**nd **M**anipulation* is one of the most valid softwares available nowadays for computational fluid dynamics. It consists essentially in a C++ toolbox, it is conveniently opensource and completely editable in its libraries and codes, thus allowing for an (expert) user tailor-made changes for the current problem. Furthermore it provides a full integration with the graphic software *Paraview*, a programmable and useful tool to data visualization (see [9] for further details).
Thanks to its opensource nature, it possesses a widespread and constantly-

updated library, making it suitable for the most diverse variety of problems. It goes withouth saying that the choice between all of these possibilities must be driven by a thorough knowledge of the problem to be analyzed.

3.1.1 Discretization

Surface management The surface management is in OpenFOAM a procedure made out of several consecutive steps. At first, the user needs to build a sorrounding box that has to comprehend internally the entire domain object of study. In this part, the box is built with a staggered cartesian subdivision of the space, using the tool *BlockMesh*. Many possibilities are available here to build the geometry more suitable for the case. Here the user has to specify the discretization, the eventual melding between different boxes, as well as the intended boundary conditions for the surfaces the box is made out of.
Then follows the introduction of all the objects that the user desires inside the domain; the most direct method is the construction of an ascii-written stl file.
The last part is the usage of *SnappyHexMesh*, a meshing tool in which the stl can be placed inside the domain, the desired parts can be considered inside or outside the field and so on. This utility proceeds by two steps. At first the geometry is refined where specified - still as cartesian cells, with *CastellatedMesh* - and then, the final mesh is refined and made out of hexaedral cells.
Here is possible to specify the desired resolution, smoothing criteria and geometrical tresholds.
In this study the first mesh build via *BlockMesh* has dimensions $6\pi x8x2$ and cells $400x170x100$, for a total 6.8M cells.

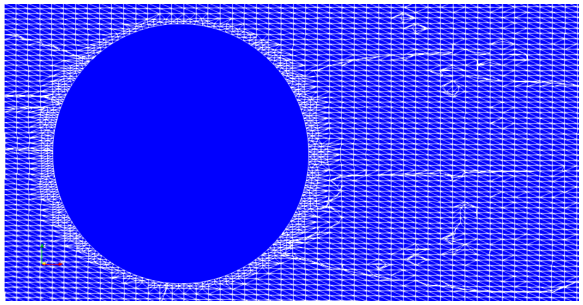


Figure 3.1: *magnification of the cells structure on the cylinder surface, y-normal section of the OpenFOAM simulation's domain*

Then the stl file of the cylinder was comprehended via *SnappyHexMesh* and positioned at 1/6 of the streamwise total length. Its surface was refined with a level of refinement equal to 2, meaning that each surface cell has been discretized one time into 4 sub-cells (as indicated through the *nCellsBetweenLevels* OpenFOAM variable). Then the entire mesh encountered the final refinement into hexaedral cells.

3.1.2 Solver

OpenFOAM is provided with many possibilities in terms of resolution schemes and simulation types. It contains several laminar schemes, RANS schemes, LES schemes and even a DNS dedicated option - suitable only for cubic domains, *dnsFoam*.
Through the editing of its dictionaries the user can choose the options that suit the most his needs. In particular, to run a DNS simulation for this study was necessary, in the *turbulenceProperties* dictionary, to specify a *laminar* type of simulation. This way the program would not model any of the flow structures, whatever the scale. Of course in order to do this in a proper manner, the resolution needs to be high enough to capture all the turbulence scales.

The pimpleFoam solver *pimpleFoam* is an incompressible, transient solver which can include turbulence models as well as customized finite volume options. It requires mandatorily two input fields, kinematic pressure and velocity. Its algorithm, called indeed *pimple* combines two different OpenFOAM algorithms, them being *piso* and *simple*.
The piso (***P****ressure* ***I****mplicit with* ***S****plitting of* ***O****perators*) algorithm is designed to solve Navier-Stokes equations in unsteady problems and is made

out of various steps. After the setting of boundary conditions, the discretized momentum equation is solved to compute an intermediate velocity field, making possible the computation of mass fluxes at the cells faces. Then the pressure equation is solved and take place the correction of mass fluxes and velocities on the basis of the pressure field. Then the boundary conditions are updated and the time step increased and the loop ends; of course the central steps of the algorithm are repeated the necessary number of time to correct for non-orthogonality.
The simple (***S**emi-**I**mplicit **M**ethod for **P**ressure **L**inked **E**quations*) algorithm starts with the construction of the momentum equations. Then, differently from the piso algorithm, it operates a under-relaxation of the momentum matrix before solving the momentum equations for velocities. From here, it constructs and solves the pressure equation, the correction of the flux takes place and the pressure itself is under-relaxed. Finally, the velocity is corrected through the pressure solution; here as well, of course, in case the convergence is not reached, the central steps are repeated before proceeding with the time step. Here, however, the momentum corrector step is perfomed only once (differently to what happens in piso). See [7] for further details.
The pimple algorithm combines the best of both methods in a dynamic way in order to increase its time performance without sacrifing accuracy.

FvSolutions and FvSchemes In OpenFOAM there is the possibility to edit the choice of the scheme used to solve the partial differential equations. This can be done once the solver has been specified, if the most suitable choice for the current simulation is not the default one.
In our specific case, through the library *FvSolutions*, the schemes chosen were GAMG (***G**eneralized geometric-**A**lgebraic **M**ulti-**G**rid*) for the pressure, with a DICGaussSeidel (***D**iagonal **I**ncomplete-**C**holesky/LU with **G**auss-**S**eidel*) smoother. For the velocity it has been used a smoothSolver - a solver that uses a smoother - with a symGaussSeidel (*symmetric Gauss-Seidel*) smoother (see [4] for further details).
Then through *FvSchemes* it was possible to set the numerical schemes to be used for each single finite difference calculation. For the gradient, laplacian and divergence operators the designed schemes were different applications of the linear Gauss scheme - namely a stardard finite volume discretization of Gaussian integration with a linear interpolation of values from cell centres to face centres. The backward scheme - transient, second order implicit,

partially unbounded - was chosen for the time derivatives.

3.1.3 Imposed conditions

Conditions It is important to enlighten one aspect about this flow: it takes long time to bring it up to speed. The turbulence must develop sufficiently in order to observe its structures and phenomena. In order to do this, the simulation has been carried on for about 30 time units, without the imposition of the CFR condition and with a uniform velocity and pressure field. In the entire domain the initial conditions were $\mathbf{u} = (\ 1\ 0\ 0\ )^T$ and $p = 0$.
Then the CFR condition, with $U_B = 1$, was imposed from that istant on and the simulation proceeded for 200 time units. It is worth reminding that the turbulence has no memory, this meaning that a turbulent flow does not possess characteristic determined by what generated its turbulent structures. Therefore, with a streamwise max lenght of 18.85 (6π) and being the velocity bulk imposed with unitary value, 200 time units were enough for the flow to reach capacity and steady statistics.

Implementation In OpenFOAM it is possible to edit ad-hoc libraries whenever the standard ones are not suitable to the specific needs of the simulation. Imposing the CFR condition onto the domain is one of those situations. Therefore, a particular library called *FvOptions* was edited and included into the *system* directory - which is the directory contanining the simulation main features.
Specifically, the edited version of this dictionary provides a correction based on the pressure gradient, acting through a momentum term to force the flow rate to be constant. The purpose is to reach a reference average velocity - U_B - on the whole domain.
An adjustable timestep was considered during the simulation, as specified inside the *ControlDict*, the dictionary dedicated to simulation control.

3.1.4 Parallelization logics

On OpenFOAM, when running a simulation in parallel, the geometry must first be decomposed into individual geometries for each MPI process. These separate geometries are connected together with special processor boundary

patches.
The utility here used for decomposition, called *decomposePar*, is a common method to decompose domains and subsequently distribute the fields. In its dedicated dictionary, *decomposeParDict*, it is required a certain group of specifications since OpenFOAM offers a variety of decomposition methods and routines. If a decomposition method requires additional configuration controls, these can be specified in a dedicated dictionary.
For the concerns of this study, the so-called *simple* decomposition was good enough; it comprehends as input the number of subdomains and the number of processors in each direction (x/y/z). A basic geometry like ours can be efficiently managed this way.

Moreover, it is worth mentioning that the reconstruction process has to be explicitly carried via the *reconstructPar* utility; in our comparison this is a secondary but still important issue. In fact, running a parallel simulation on OpenFOAM yields to as many directories as the number of processors, and the output results are saved there. It is a proper domain decomposition that takes place, with new boundary conditions generated so that each core can run its own simulation. In order to rebuild a whole set of results, the reconstructPar utility is needed. The more processors the simulation was split onto, the more time this operation requires.

3.2 CPL

The *CPL* language is a programming language developed by professor Paolo Luchini of University of Salerno. It is an high-level language based on the C language, with an intuitive and innovative syntax and memory management. It is inherently designed for direct numerical simulations and the code used in this very study is structured in various libraries, each one appointed for a specific task.
Even though it does not really consists of a structured software but instead of libraries, executable files and a specific *makefile* utility, for sake of simplicity we will generically refer to CPL as to the ensemble of codes edited in this language.
Through the compiler is thus possible to build, from five different modules, the code to run the simulation. Inside the parallelbcs modules (*parallelbcs.cpl* and *parallelbcs.h*) the definitions of the parallel algorithms and boundary conditions are found. Inside *iofiles.cpl* the input and output

phases are treated, both in terms of initial field (when present) and between different time steps. *timestep.cpl* manages the resolution scheme and then the main module - *cylinder.cpl* - defines and run the simulation, gathering all other utilities and subroutines.

3.2.1 Solver

The DNS code, through a particular version of the immersed boundary method, is able to simulate an incompressible flow around a solid object confined inside a channel. The geometry of both the channel and the object can be specified inside the code itself. One of the distinguishing aspect is that the governing equations are solved into physical space; this way the introduction of Fourier transform is avoided. Thus the solver takes advantage of the simplicity of second order finite differences.
The solving scheme proceeds with an algorithm which subdivides each temporal step into substeps, in order to simplify the numerical complexity associated with the operator; this type of scheme is called operator-splitting.
More in depth, we consider the forcing term $\mathbf{F}$ at right hand side of (3.1) necessary in order to make the fluid flow through the channel; in our non dimensional formulation the term represents a force per unit of volume.

$$\frac{\partial \mathbf{u}}{\partial t} = -u \cdot \nabla u - \nabla p + \frac{1}{Re}\nabla^2 \mathbf{u} + \mathbf{F} \tag{3.1}$$

The temporal discretization is based with the Runge-Kutta scheme (or the Adams-Bashforth when specified); the Kutta's third order method divides each timestep into three sub-steps assigning each one of them a coefficient. The spatial discretization takes place on the staggered cartesian grid built by the IB method; the implemented scheme is the second order finite differences. These methods together govern the time progress of the simulation through the computation of the discrete derivatives (see [16] for further details).
During the resolution, firstly the velocity is obtained solving the momentum equation; in fact, all velocity terms (apart from the time derivative)

are computed at the previous step, thus making the velocity explicit. Since all non-linear terms are referred to the previous-step-computed variables, the solution of the equation can now be calculated via Gauss substituion, since we have a (peculiar) linear system of equations. The pressure however, remains explicit.
Here takes place the second part of the scheme: the velocity solution just computed is substituted into the continuity equation, thus leading to an equation where the pressure is the only unknown; the adimensionalized pressure equation.

$$\nabla^2 p = \nabla(-\mathbf{u} \cdot \nabla \mathbf{u} + \mathbf{F}) \tag{3.2}$$

This equation can be discretized to become a discrete laplace equation, that is then resolved through the so-called pressure correction.

3.2.2 Discretization

The domain discretization is fundamental to define the finite differences used to solve the numerical version of the governing equation.
In the CPL's simulation it has been considered a staggered cartesian discretization of the domain, which leads to a mesh made of cubic cells.

More in depth, the velocity components are defined halfway between the nodes in which the pressure field is evaluated and cosidered. After the temporal and spatial discretization (exploiting them with the kroenecker δ_{ij}, defined equal to one if i=j, 0 otherwise) we get the first part of the resolution system in the form of:

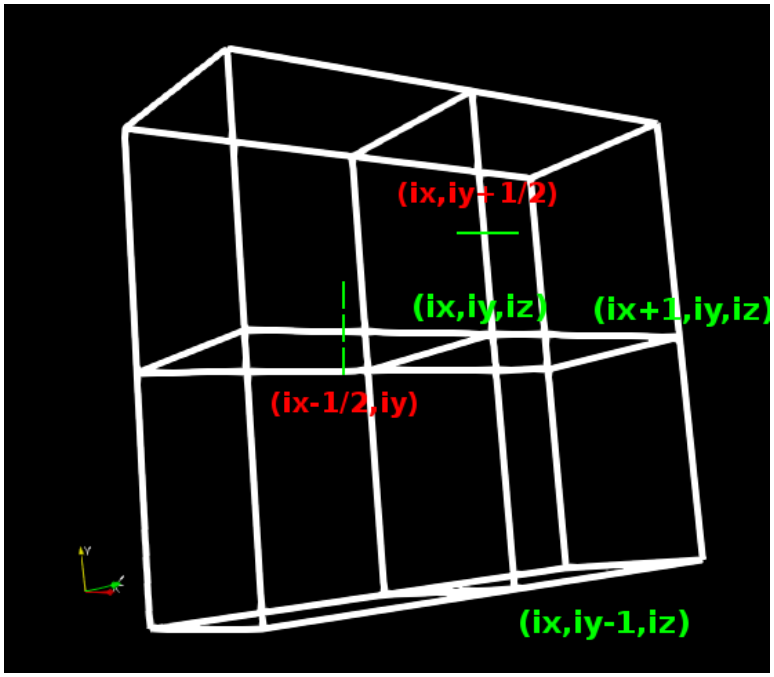


Figure 3.2: *nodes and cells of the staggered grid on the x-y plane*

$$\begin{cases} \frac{u_j^{n+1}(ix_j+1/2)-u_j^{n+1}(ix_j-3/2)}{2\Delta x_j} = 0 \\ \\ u_k^{n+1} = u_k^n - \Delta t u_j^n \frac{u_k^n(ix_j+1-\delta_{jk}/2)-u_k^n(ix_j-1-\delta_{jk}/2)}{2\Delta x_j} + \\ + \frac{\Delta t}{Re} \frac{u_k^n(ix_j+1-\delta_{jk}/2)-2u_k^n+u_k^n(ix_j-1-\delta_{jk}/2)}{\Delta x_j^2} + \\ -\Delta t \frac{p^{n+1/2}(ix_k+1)-p^{n+1/2}(ix_k+1)-p^{n+1/2}(ix_k-1)}{2\Delta x_k} + \Delta t F_k^n \end{cases} \quad k=1,2,3$$

The unknowns here are the velocity at the next step and the pressure at the next half-step (namely x_k^{n+1} and $p^{n+1/2}$).
That being said, the resolution of this first part (via Gauss linear substitution) provide us the velocity from the momentum equation, and then, substituting that into the continuity equation, as already explained we're left with the pressure equation, in which the pressure remains the only unknown.
Its discretization leads to a discrete Laplace equation, solved through the so called pressure correction:

$$\frac{p^{n+1/2}(ix_k+1)-2p^{n+1/2}+p^{n+1/2}(ix_k-1)}{\Delta x_k^2}=$$
$$=\frac{1}{2\Delta x_k}[-u_j^n(ix_k+1-\delta_{jk}/2)(\frac{\partial u_k}{\partial x_j})^n(ix_k+1/2)+$$
$$+u_j^n(ix_k-1-\delta_{jk}/2)(\frac{\partial u_k}{\partial x_j})^n(ix_k-3/2)+F_k^n(ix_k+1)-F_k^n(ix_k-1)] \tag{3.3}$$

The immersed boundary method The immersed boundary method has been first introduced by Peskin in [13] and is nowadays a useful tool for the treatment of fluid-structure interaction problems. The code requires a staggered cartesian grid, built without considering the immersed boundary of all the surfaces nor the geometry of the body inside of the box. The general idea is to modify the governing equations in the proximity of the solid contour as to consider the relative position of the boundary and the staggered grid.
Here lies the biggest liability of this method, namely the lower resolution control with respect to the boundary-fitted method. Of course this downside comes with an important advantage, which is the incredibly low memory usage; since the grid is staggered, there is no need to archive all cells and points position, and also the resolution of the equations will require, thanks to this fact, a much much lower random access memory by the computing machine.

The discretized equations (3.2.2) (3.3) need the introduction of a correction of the forcing term, since it is there induced the contribution of the surfaces' presence. Since the walls are not coordinate fitted, their boundaries are not known in terms of nodes from pressure and velocity.
The method therefore must proceed correcting the linear system with coefficients related and determined by the relative position between the staggered cartesian grid and the contour of the body. The influence will be considered

as to include the non-slip condition.

In the immersed boundary method version here implemented takes advantage of a linear interpolation between the grid points where the solution is computed and the points of the contour, then considering an interpolated solution to compute the governing equation's residual.

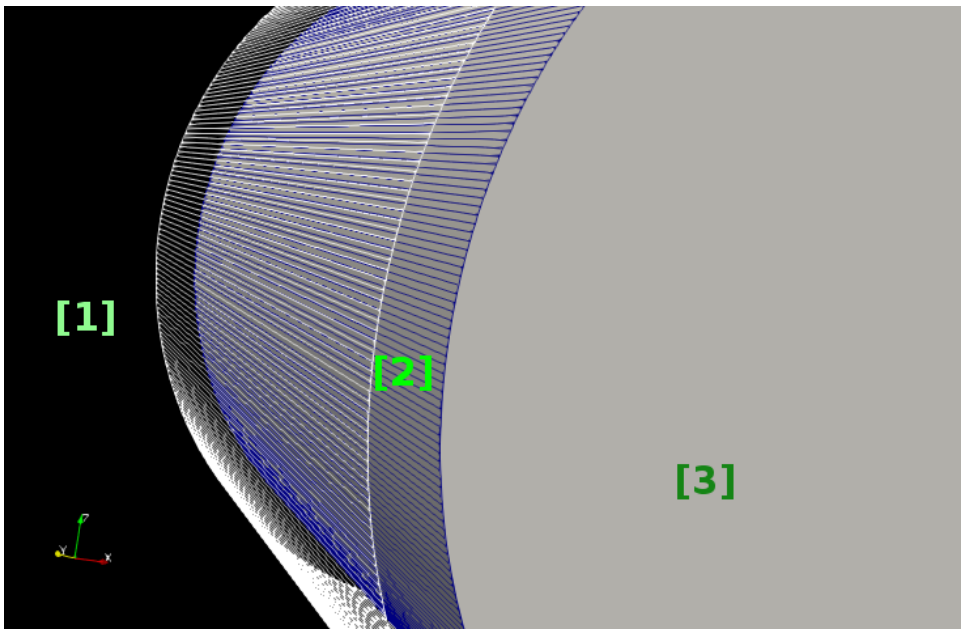


Figure 3.3: *Example scheme of the three regions near the cylinder surface*

More in detail, the method ideally divides the domain into three distinct regions, as depicted in 3.3. One region is away from walls, where the calculation can be performed without any boundary corrections; one lies where the calculation needs to be corrected as to include the interaction between fluid and solid wall; the last one lies inside the body. It is worth mentioning that once the code initiates the computation and evaluates the geometry, the code recognizes the points of the third region and exclude them from further calculations eliminating the values of the fields on them.

The present case application The residual computed in the equation is considered in an implicit form in order to ensure stability, and so, insted of

modifying the forcing term directly, in our specific case the surface presence is denoted into the coefficients of a unknown. The unknown considered is the discrete laplacian term, in which, as the surface approaches, the correction coefficient increases cancelling off the unknown (since it is the denominator of the term).

The linear interpolation yield to the possibility of a better approximation, instead of a stepped treatment of the surface; in order to do so, the discrete laplacian evaluation proceeds through a bisection algorithm. Indeed, considering it into our finite cartesian grid, its computation depends on the fields' value found on each of the seven points composing it.

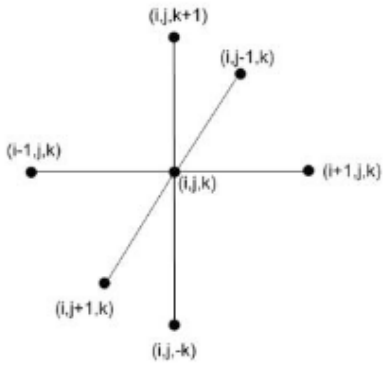


Figure 3.4: *seven points stencil representing the discrete laplacian*

Then, considering the geometry of our bluff body (namely a cylinder indefinite and straighted-axis along y), it is straightforward to comprehend why the variability of the values along the laplacian arms is contemplated along the x and z axes.

Therefore, the code itself is implemented in such a way that only the x and z coordinates can vary as to include the possibility of the presence of solid walls along their directions. Whenever, along those axes, a solid contour approaches, the bisection algorithm evaluates on each of the arms of the laplacian its relative position and then calibrates the correction coefficient along those arms which encounter the surface.

3.2.3 Imposed conditions

The conditions imposed on the CPL simulation were the most possible close to the ones implemented within the OpenFOAM simulation.

Conditions The conditions imposed at the beginning of the data production were an already developed and operational turbulent flow, characterized by a constant flow rate condition inside the domain, periodic boundary conditions streamwise and spanwise and variable time step (Δt).

Implementation To generate the initial conditions for the flow, it has been created one specific utility, called "interp.cpl". This utility generates a executable file which takes as input a turbulent flow of a generic geometry and interpolates and stretches its fields into the selected domain, in terms of cells and points, dimensions of the channel, coordinates and so on. It also takes into account the desired viscosity (which, as already said, in CPL represents the inverse of the Reynolds number). This allowed the simulation to start from an already developed turbulent situation; then the data production started after 200 time units and went on for 100 time units saving the results on a unitary basis.

As to impose the constant flow rate condition, a subroutine was implemented and inserted within the "timestep.cpl" utility, which contains all the time-solving and discretization tools needed by the main code to proceed with the simulation.
The subroutine proceeds by two quantities (UFR and VFR) which have to be specified at the beginning, and then considers them as a reference to correct the flowrate evaluated at each time step (along both streamwise and spanwise directions); the difference between them is then considered to generate correction terms that through a momentum quantity will modify the pressure gradient in the domain.
Of course the solid surfaces are firstly considered as to maintain the correct boundary conditions on walls surfaces as the simulation goes on.
It is worth mentioning that whenever the CFR condition is not specified during the initial and boundary conditions, the code proceeds considering a constant pressure gradient simulation (CPG).

In order to exploit the code maximum performance, via evaluation of the cfl number, a variable time step was implemented inside the code. The ***C****ourant–* ***F****riedrichs–* ***L****ewy* condition is a necessary condition for convergence while solving numerically partial differential equations (as Navier-Stokes are). In the numerical analysis of explicit time integration schemes, it states that the time step must be less than a certain value (namely, the

CFL number) as to avoid incorrect results. The numerical domain of dependence of any point in space and time (as determined by initial conditions and the solving scheme) must include the analytical domain of dependence (wherein the initial conditions have an effect on the exact value of the solution at that point); in essence, this ensures that the scheme can access the information required to form the solution.
This way the maximum value for Δt can be computed and the simulation speed up its resolution. Starting from an externally specified maximum value for the cfl number (a conservative value of 0.5 was considered in our case), each time step the code evaluates its instant value and then compute the Δt as a ratio between the two quantities.

3.2.4 Parallelization logic

The code comprehends the dictionaries *parallelbcs.cpl* and *parallelbcs.h* that allows and governs the parallelization of the calculation. They allow the main code to to split the grid along x and y (streamwise and spanwise) and it is written as to minimize the data exchange between processors.

The communication between cores has indeed been optimized: each single core starts solving the inner part of its grid portion, whilst in the meantime it communicates with the other processors the informations about the boundaries. Then, in a latter moment, it will solve the outer portion. Indeed, the algorithm manage the communication in such a way that each single processor will be able to compute the second order differences of the solving scheme, and this sequence of steps it has been proven to be the most efficient.

Moreover, CPL has the intrinsic capability to produce a single, unified output. Contrarily to what OpenFOAM does, the fields data are in fact produced into whole, all-domain comprehending output files. This, together with the possibility to edit completely and easily the saving criteria (for example iteration based rather than time based), is a straightforward advantage with respect to the OpenFOAM's functioning.
In the overall requested time for one simulation, the simplicity of this code allows to avoid complex and non-negligible operations, increasing its perfomances under the whole time-saving perspective.

Code	**Surface management**	**Parallelization logic**	**Main advantages**	**Main disadvantages**
OpenFOAM	Boundary-fitted	ad hoc utilities for decomposition and recomposition, MPI running	Complete, reliable and opensource	not very user-friendly w.r.t. other widespread softwares
CPL	Immersed-boundary	inbuilt and automatic logic for decomposition, recomposition and logic	Task-oriented, efficient	requires a profound editing for new domains' studies

Table 3.1: *brief schematic comparison of the two codes*

Chapter 4

Results

The simulation of the flow around a confined cylinder was comprehensive of a large data recollection. Moreover, post processing operations have been performed in order to evaluate the phenomena taking place and compare the results given by the two codes.

4.0.1 Expected results

The periodic boundary conditions consist in a numerical stratagem where the flow exiting from one boundary surface will re-enter from its coupled surface. In the present case, the flow exiting from one of the x-normal surfaces re-enters from the other one and the same happens for the two y-normal surfaces.
In the study of turbulent flows, this type of boundary condition is usually implemented in order to study the development of wall turbulence and its phenomena in channel flow studies. In our case we wanted to study the periodic features in a handy and reliable way. Therefore a full speed condition had to be reached in order to develop a consistent state of turbulence.
This was ensured in our case by the periodic boundary conditions along with the turbulent initial conditions.

The blockage ratio in our study is particularly elevated, as already said: 0.5 represents the fact that the perpendicular surface of the object that the flow encounters is half of all the channel's transversal area.
It is straightforward to comprehend that this fact will profoundly affect the behaviour of the flow. The region of the domain where the flow is bound to pass, through the section between the cylinder and the two physical walls,

will indeed act - firstly - as a convergent. Hence the flow will experience a sensible acceleration and decrease in pressure. This will hold until the widest perpendicular section of the cylinder - the central one - is reached by the flow.
There the area available for the flow to pass is at its minimum, similarly to the throat section of a nozzle. After that, the cross-sectional area available starts to increase progressively, leading to a divergent-like behaviour, generating the opposite phenomenon: a velocity decrease and a pressure increase.

Therefore, the phenomena taking place along the bluff body wake only partly reflects the linear and neat ones explicated above.
The acceleration of the flow entails the structure of the streamlines to realign and also the local Reynolds number to increase. We can clearly observe from the visualizations that the consequences of this phenomenon will reflect also on the shear layers of the bottom and top walls.

These boundary layers are turbulent and proceeding downstream they will detach, generating a complex interaction with cylinder's own detached shear layer. This interaction will reflect on the increased, unsteady, three-dimensional turbulence characterizing the wake.
The main features described in the literature are present, but slightly different in some cases.
The harmonic pattern generating on the cylinder along y-axis direction is observable in its first expressions. The vortex shedding phenomenon, along with its characteristic frequencies, is present but not straightforward observable as it would be in a case with less developed three-dimensional structures.

Among the several data graphs and visualizations disposed from now on, it can be recollected the turbulence structure in the flow. In particular, the mean profiles data help our understanding of the wake behaviour downstream the cylinder. The decay of the largest scales of turbulence is easily observable, and the decrease in the kinetic energy content can be derived from there.
The three dimensional phenomena influence the data recollection giving a very convoluted and twisted behaviour of the instantaneous quantities, denoting the complexity of the eddies' time-changing disposition.

Data collection The database was acquired saving the fields at each time unit of both simulations. The simulations proceeded from instant 200 to 300; therefore one hundred instantaneous fields constitute the available data.
All the data reported in the visualizations have been processed and obtained from those saved fields. The mean fields and the standard deviation fields have been computed over 100 time units as reference.

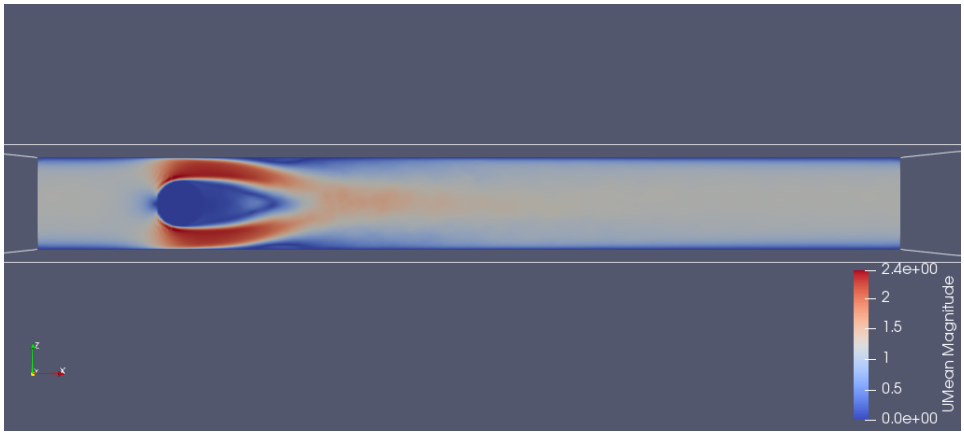
4.1 Visualizations

The first thing to consider is that the immersed boundary code does not build the geometry of the domain around the cylinder, as already said. It will be then immediate to understand that, while in OpenFOAM visualizations the cylinder's body is not part of the domain, in CPL visualization it will be. All fields are null inside of it, and therefore the viewer should be able to identify it.
The most part of the visualizations displayed below pertains to sections effectuated in the middle of the domains' dimensions along y and z axes.

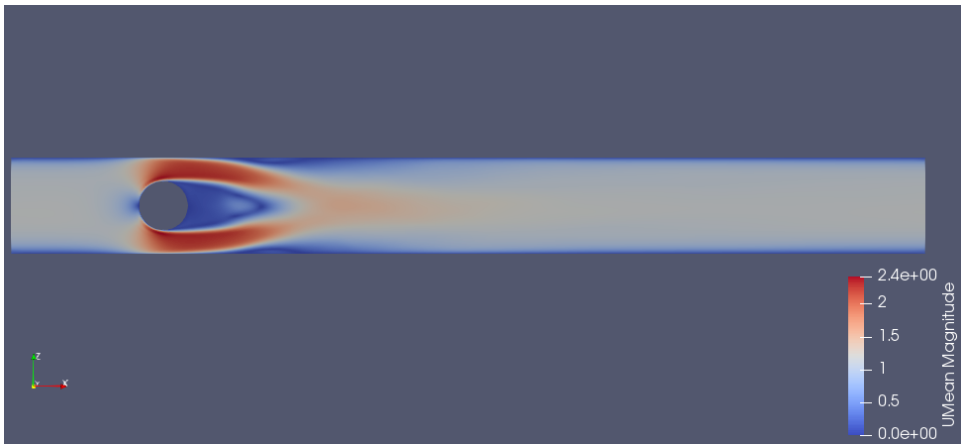
All the quantities here reported in figures are non dimensional and scaled with the reference quantities used to implement the equations from their original version.
For the very same reason, the eventual presence of negative pressure values is not an index of incorrect simulations - nor, of course, of negative pressure inside the domain.

4.1.1 Velocity

Mean field is represented in the figures 4.1 4.2. The differences between the two codes' results are just qualitative, as can be seen from the mean profiles shown latter in this chapter.



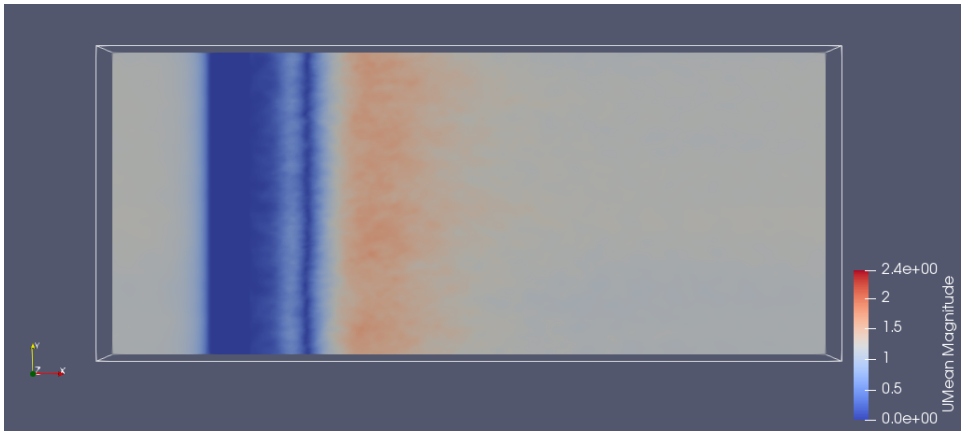
(a) *Time-averaged velocity field cpl, y-section*



(b) *Time-averaged velocity field openfoam, y-section*

Figure 4.1: *Time-averaged velocity field, from time 200 to 300, y-section*

The solver implemented in OpenFOAM requires the imposition of a diverse set of tolerance parameters and smoothing utilities, whose action can be detected into the different variance field present in the domain.

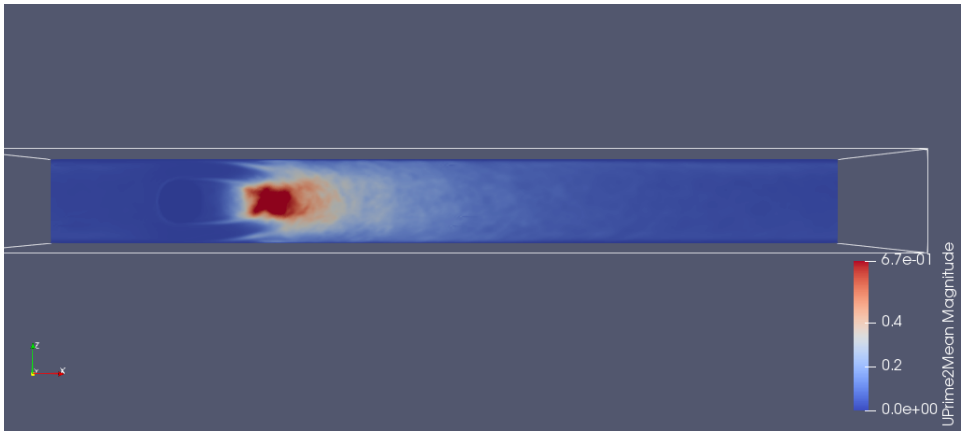


(a) *Time-averaged velocity field cpl, y-section*

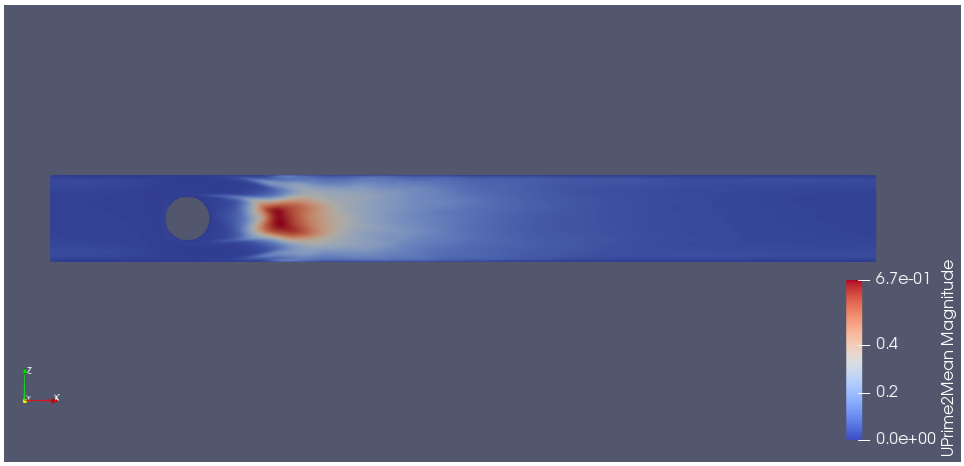
(b) *Time-averaged velocity field openfoam, z-section*

Figure 4.2: *Time-averaged velocity field, from time 200 to 300, z-section*

The variance for CPL is represented in fig. 4.4, while for OpenFOAM is depicted in fig. 4.3. OpenFOAM provides a runtime evaluation of the variance, and the standard deviation data have to be computed in the post processing phase. A comparative view of the standard deviation results can be found at paragraph *Mean fields profile*. The utilities that compute variance and standard deviation for CPL data has been implemented ad hoc in a handy way, thanks to the programming language's elevated plasticity.

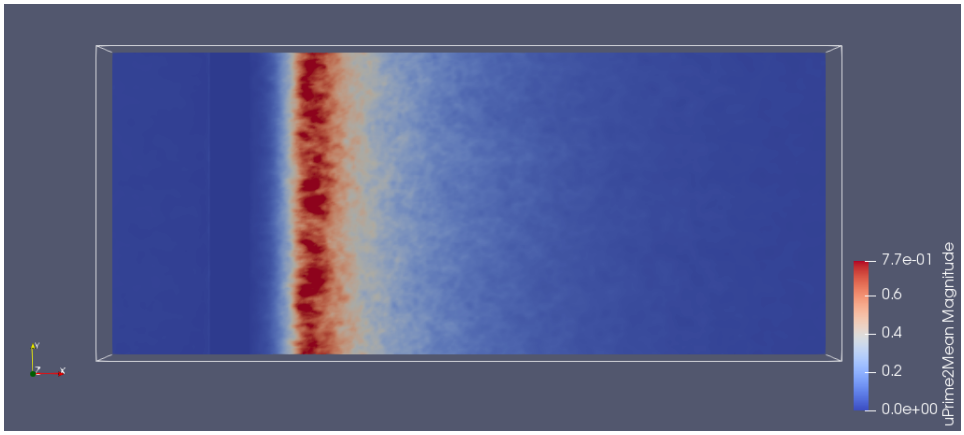


(a) *variance of velocity, y-section, cpl*

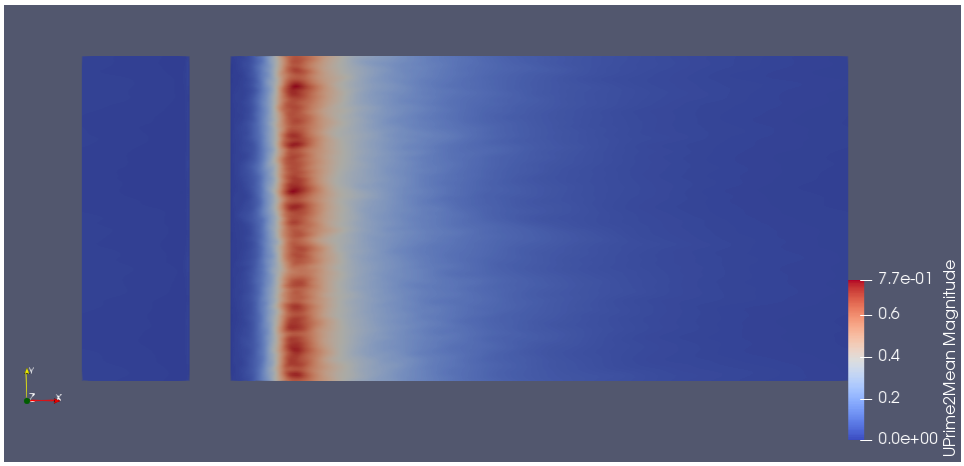


(b) *variance of velocity, y-section, openfoam*

Figure 4.3: *velocity magnitude variance*



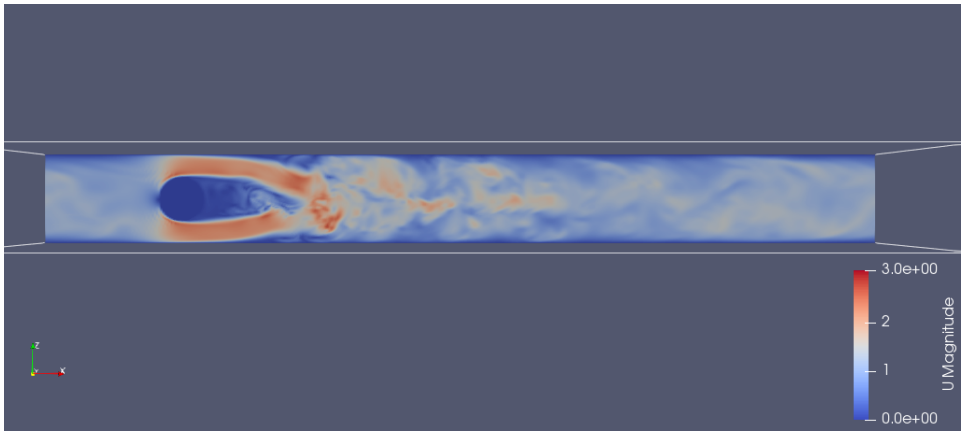
(a) *variance of velocity, z-section, cpl*



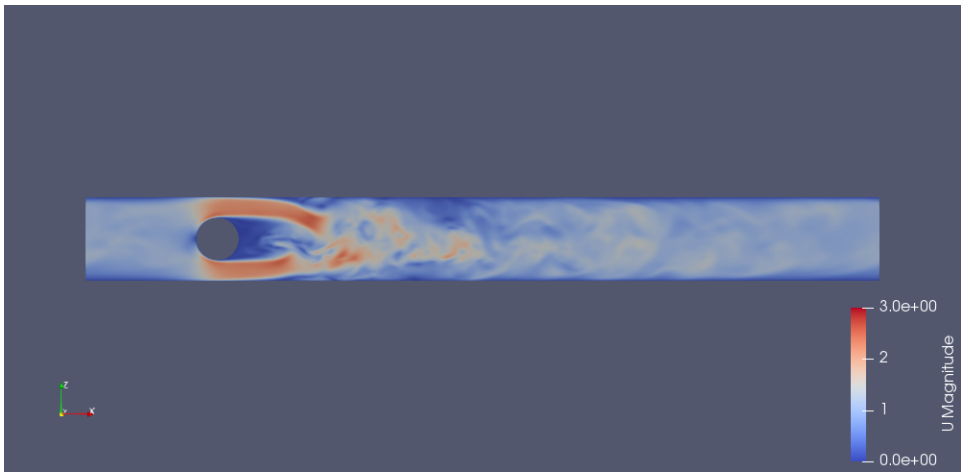
(b) *variance of velocity, z-section, openfoam*

Figure 4.4: *velocity magnitude variance*

Instant fields are briefly reported below in fig. 4.5 4.6 as to represent the flow structures and phenomena.
It is worth recalling that this data are instantaneous and therefore concern a single instant in the flow time history. Giving the turbulent nature of the current, it is embedded that differences are encountered between the two codes, as they are between different moments in the same simulation as well.



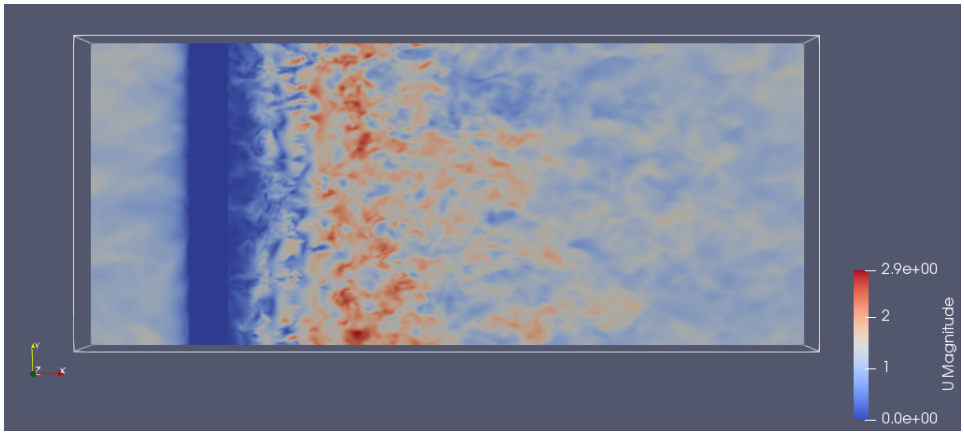
(a) *y-normal section, cpl simulation*



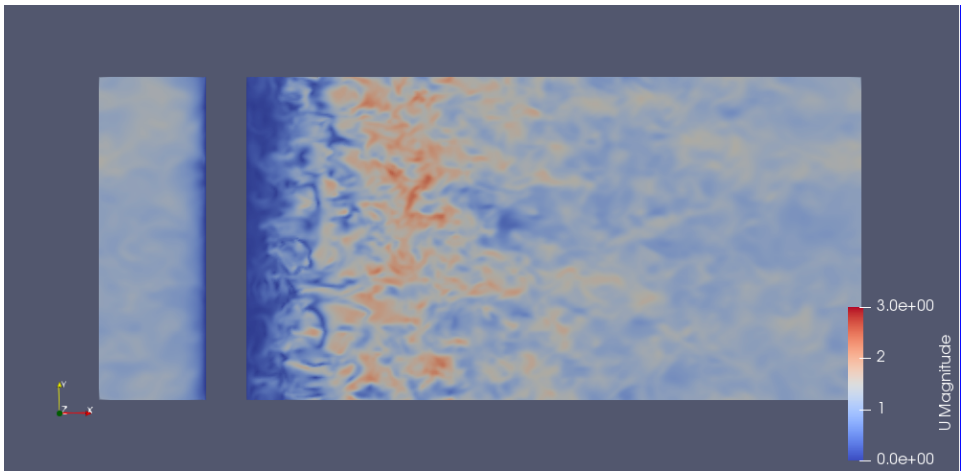
(b) *y-normal section, openfoam simulation*

Figure 4.5: *velocity magnitude at time = 200, starting time for the data acquisition*

The possibility of finding different range of velocities in different sections of the same simulation at the same time confirms the unsteady convoluted behaviour of the flow.
It is however important to recall that the data agree on the description of turbulent structures as well as on the values of the velocity field and their location in space.



(a) *z-normal section, cpl simulation*



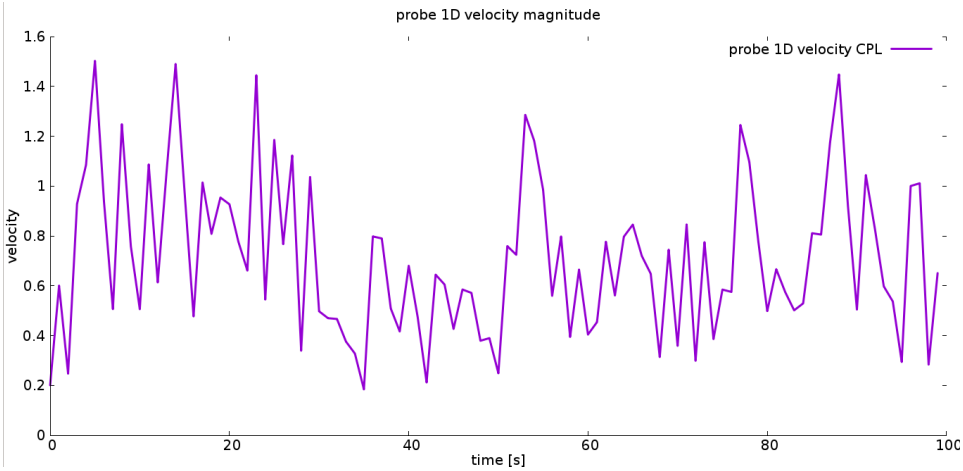
(b) *z-normal section, openfoam simulation*

Figure 4.6: *velocity magnitude at time = 200, starting time for the data acquisition*

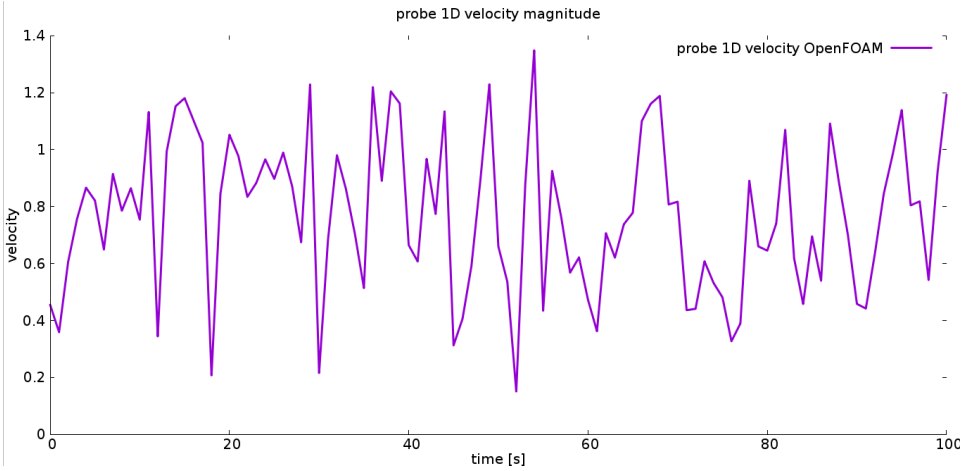
The time 200 has been chosen as an example, given that the simulation is unsteady and therefore any instantaneous rendering has the same value reporting these time-changing quantities.
From fig. 4.5 4.6 can be seen that the flow's velocity magnitude encounters big variation inside the domain. We can observe the highest values, around 3.0, develop into the convergent section of the domain; this is three times the reference value of U_B. These high velocities then proceed downstream generating the biggest scales eddies, which persist before decaying into smaller

scales eddies for about 5 times the characteristic length maximum. The persistency of these structures is influenced by the presence of the confining walls, as reported in [10].
Of course, along with that we can spot the velocity reaching its minimum values in the recirculatory area just behind the body. In the stagnation points the velocity is particularly low, as expected, but into the recirculation caused by the flow separation, the area of minimum velocity (around 0) is way wider and developed.
The flow then rapidly evolve into a turbulent flow analogous with the one that can be observed inside classical channel flow studies.

Probe The velocity evolution in the recirculatory area have been studied also by a probe located 1D downstream the cylinder's side (fig 4.7).
This position in particularly relevant in the evaluation of the wake's behaviour and therefore a comparison between the two codes can be performed.



(a) *cpl simulation*



(b) *openfoam simulation*

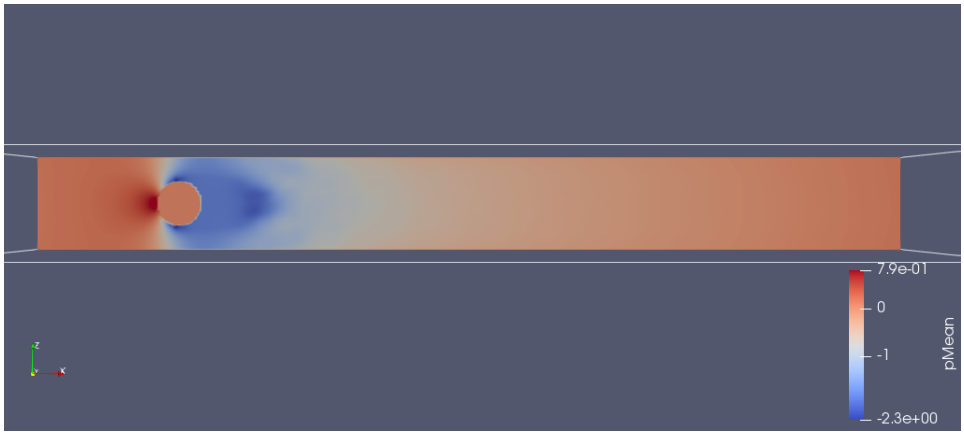
Figure 4.7: *velocity magnitude as a function of time*

We can see an overall similar behaviour although the instantaneous variation of this value are different from one simulation to another. This could represent a limited database acquisition, as the statistics behaviour depicted

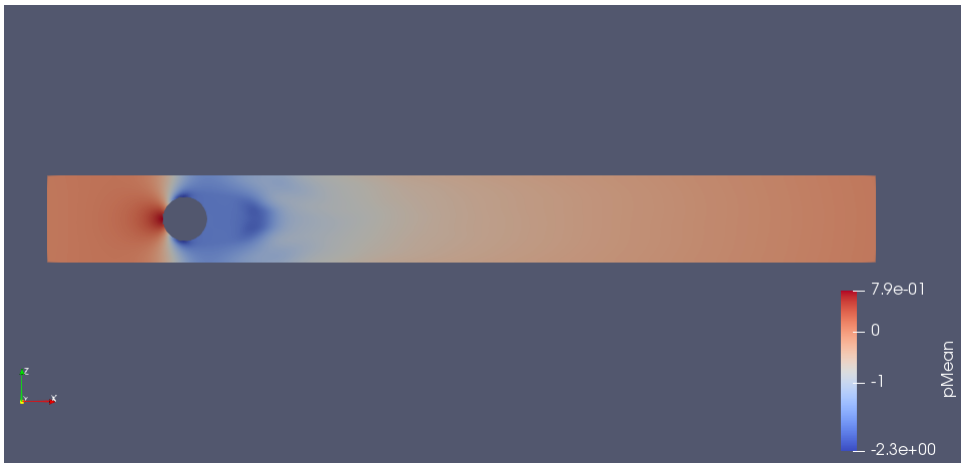
in fig. 4.19 can confirm.

4.1.2 Pressure

The mean field reported in fig. 4.8 4.9 has been evaluated with a time-averaging operation. All the same reasoning carried on in the paragraph about velocity visualization still holds for the pressure field.
It's important to describe how the pressure field evolves in the cylinder's wake. We can identify the recirculatory region for its low values, around -3 for instant fields, around -2.5 for mean fields. The pressure increase expected from the diverging action of the cylinder side is therefore spoiled by the turbulence phenomena, reflecting the behaviour proper of a bluff body.

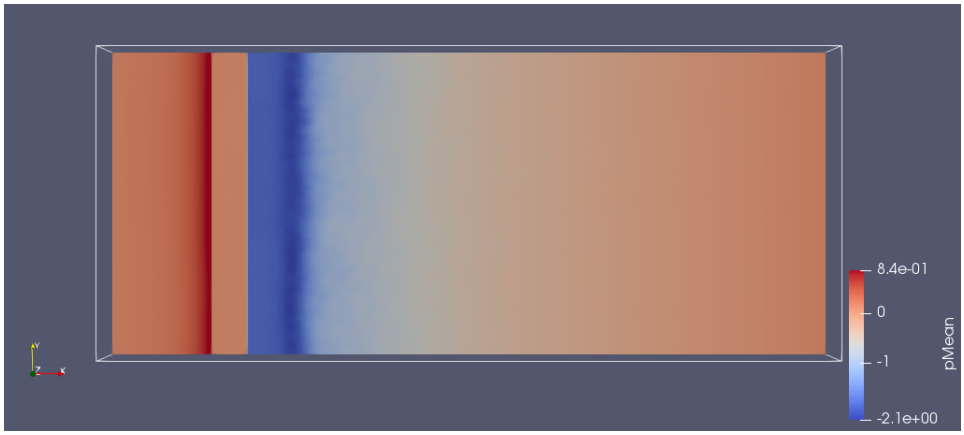


(a) *Time-averaged pressure field cpl, y-section*



(b) *Time-averaged pressure field openfoam, y-section*

Figure 4.8: *Time-averaged pressure field, from time 200 to 300*

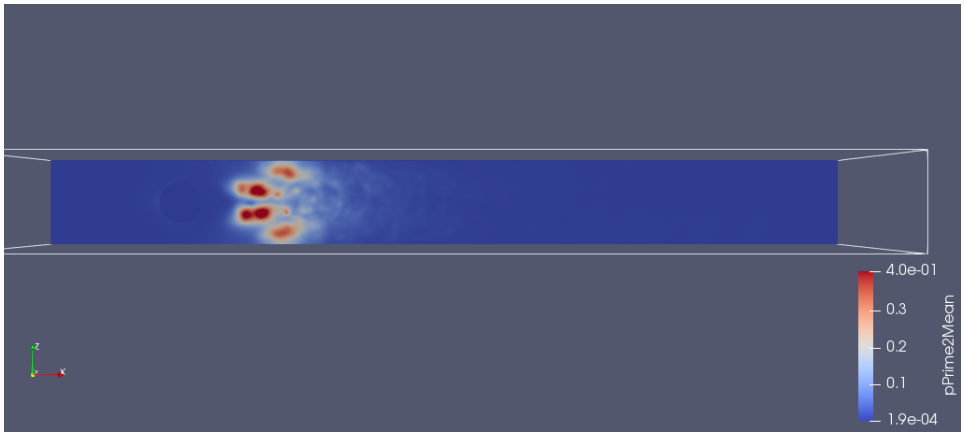


(a) *Time-averaged pressure field cpl, z-section*

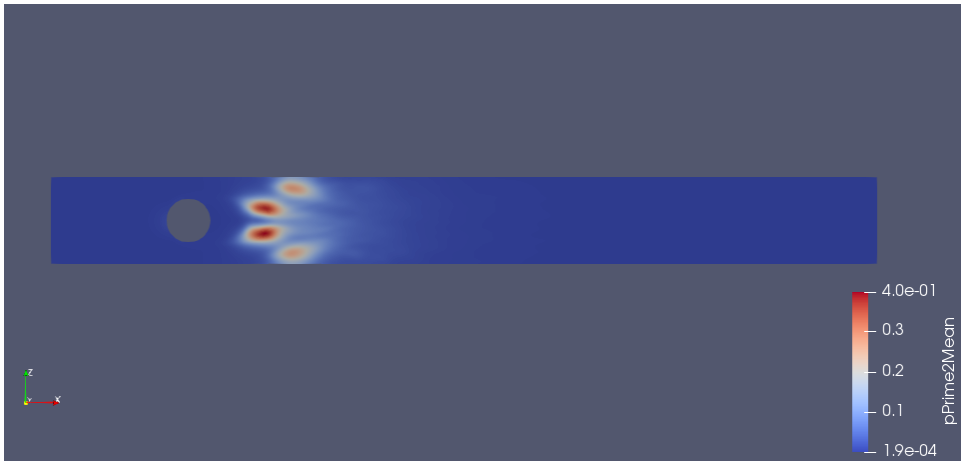
(b) *Time-averaged pressure field openfoam, z-section*

Figure 4.9: *Time-averaged pressure field, from time 200 to 300*

The variance results for CPL in fig. 4.11 are aligned with the velocity ones, as well as the variance results of OpenFOAM's simulation in fig. 4.10.

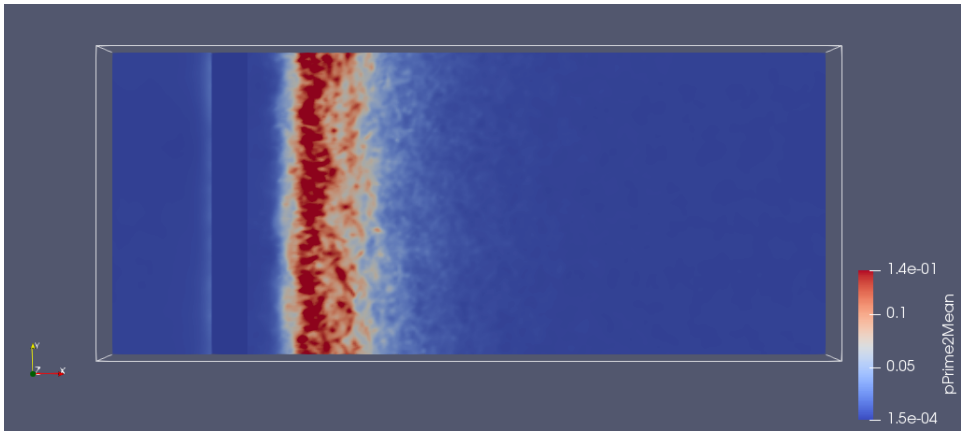


(a) *pressure variance, cpl, y-section*

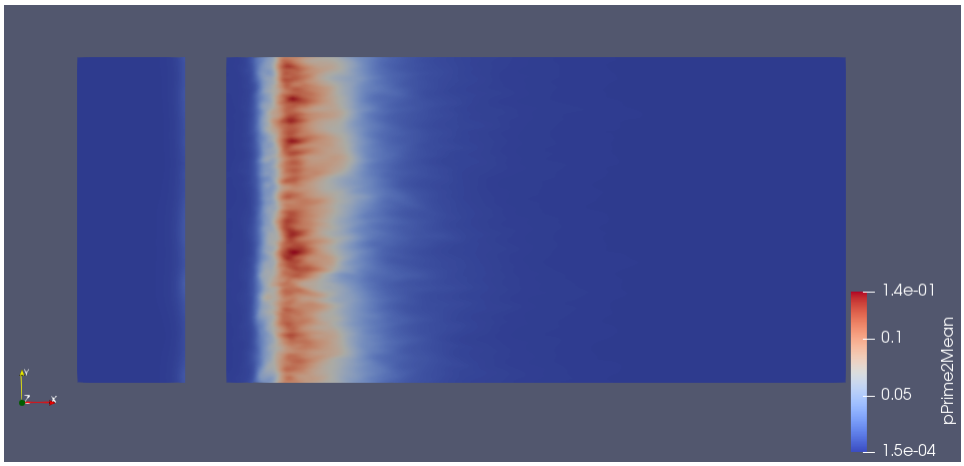


(b) *pressure variance, openfoam, y-section*

Figure 4.10: *variance of the pressure field*



(a) *pressure variance, cpl, z-section*

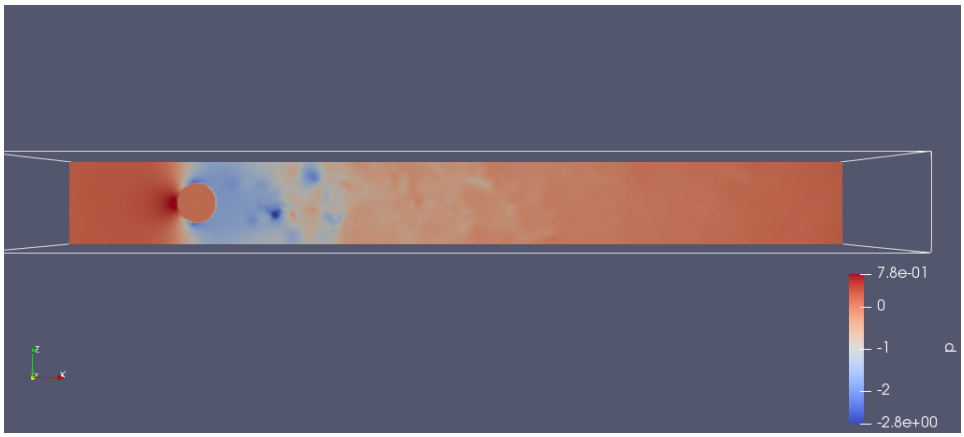


(b) *pressure variance, openfoam, z-section*

Figure 4.11: *variance of the pressure field*

Instant fields are reported in fig. 4.12 4.13 for the very same condition portraied for the velocity fields. Where narrow regions of low pressure are spotted, there lies the core of swirling structures.
We can easily recognize the stagnation points in front of the cylinder, diffused all over the surface front side. There the flow reaches its minimum velocity values whilst the pressure fields reaches its highest ones.
From the mean pressure field can be recognized the area where the biggest eddies, generated in the vortex shedding phenomenon, encounters just behind the limit of the recirculatory area. In that area, from the instantaneous

field visualization, we can also observe the pressure reach its minimum values in those points where the center of large eddies is located.

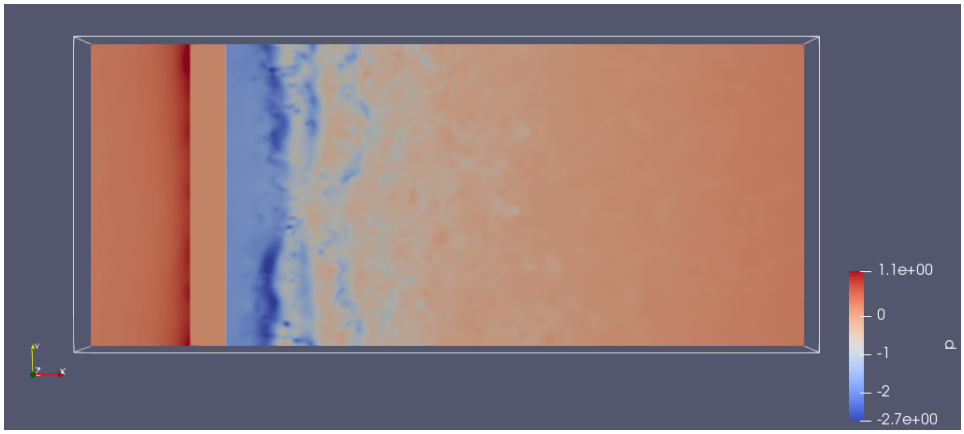


(a) *y-normal section, cpl simulation*

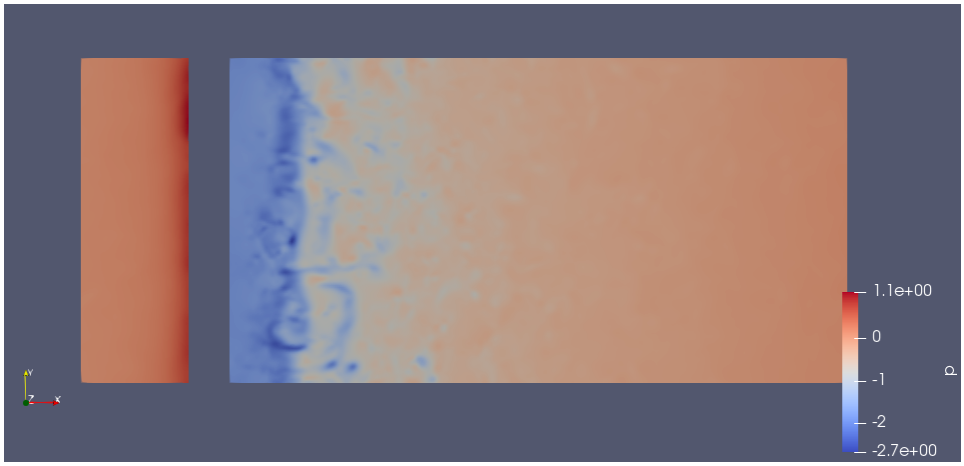
(b) *y-normal section, openfoam simulation*

Figure 4.12: *pressure field at time = 200, starting time for the data acquisition*

As was found into the velocity visualizations, along the y axis it is possible to observe that the vortex detachment is a three-dimensional phenomenon, and the y-dimension is developed enought to evaluate the first harmonic realizations of that.



(a) *z-normal section, cpl simulation*

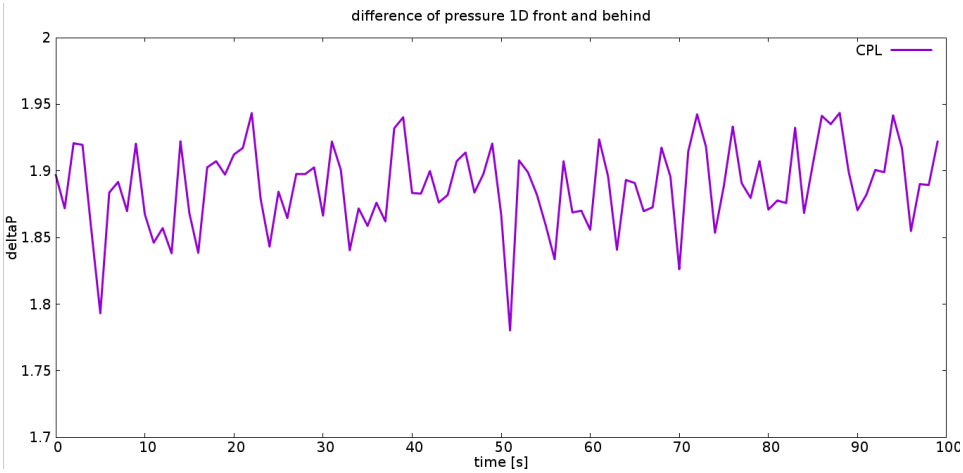


(b) *z-normal section, openfoam simulation*

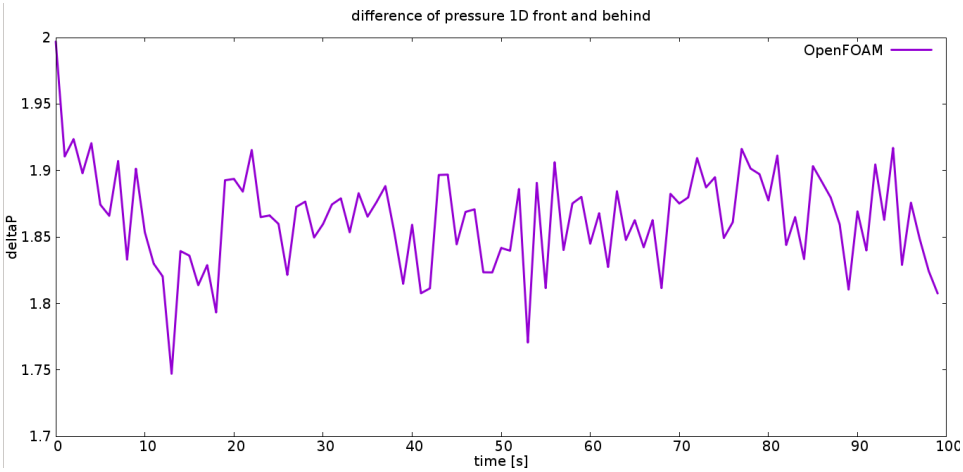
Figure 4.13: *pressure field at time = 200, starting time for the data acquisition*

Pressure difference The difference of pressure before and after the cylinder have been evaluated thorough all simulation; the unsteady nature of the flow make this values interesting to be viewed as a function of time. This in fact gives us the opportunity to study turbulent phenomena taking place in the cylinder wake development and is an important index to evaluate the drag.
In order to obtain the results shown in 4.14, we chose two reference surfaces. The first one lies at 1D upstream of the cylinder's side, the other 1D down-

stream; of course both of them are oriented x-normal.



(a) *cpl simulation*



(b) *openfoam simulation*

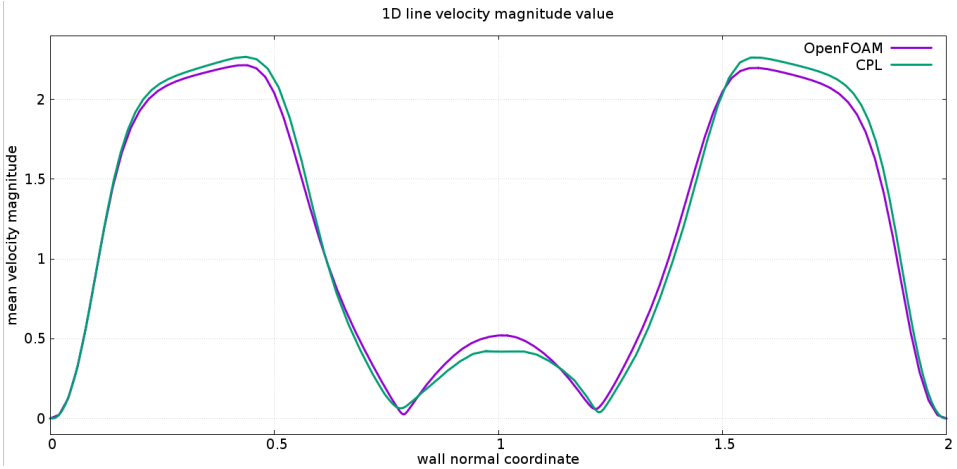
Figure 4.14: *pressure difference as a function of time*

To compute these results the pressure has been integrated over the surface and then normalized with the area. The results show time changing values for both simulations but also a relevant difference in range.
The non negligible variations in time suggests the presence of large, evolving turbulent structures, instead of a sole small scale turbulence, whose action would have been less unsteady when averaged on the whole surface.

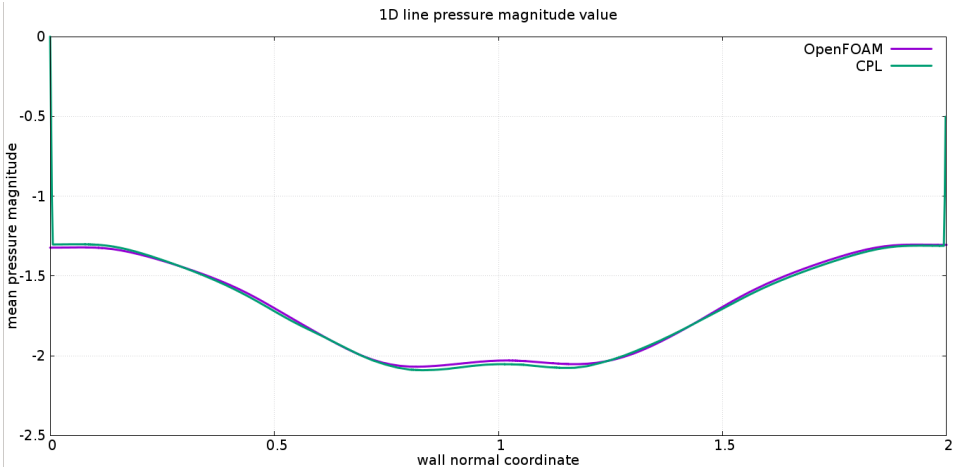
However, as expected by the theory, the pressure difference trend varies along a certain high value (1.85 more or less), resembling the almost costant behaviour of the high drag coefficient of a bluff body.
This values can be compared with 4.8 to better comprehension.

4.1.3 Mean fields profile

Through the paraview filter *PlotOverLine*, it was possible to collect the mean velocity magnitude profile and the mean pressure profile.
Starting from the temporal averaged fields - whose visualizations were reported before in this chapter - it was possible to select certain locations where plot the mean profiles.

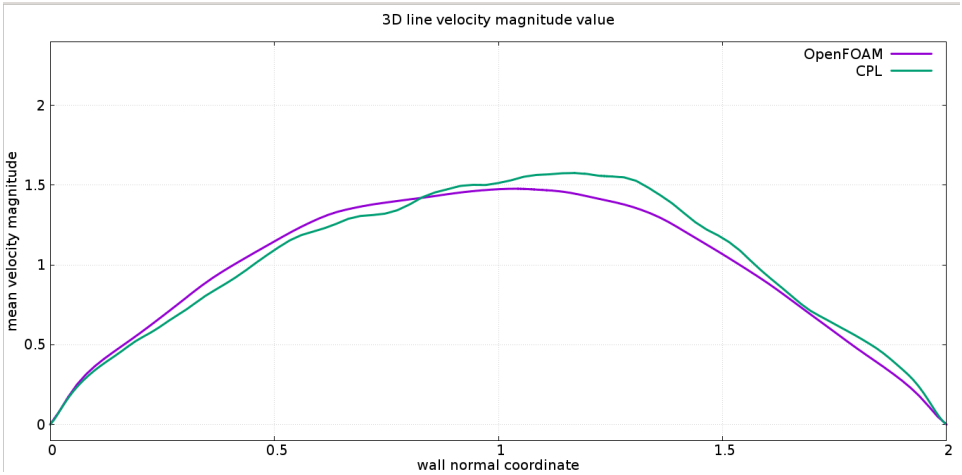


(a) *velocity*

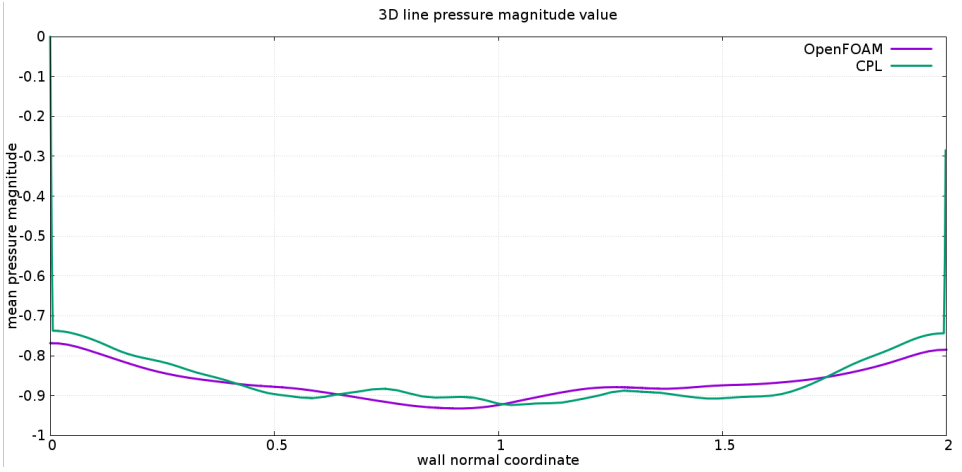


(b) *pressure*

Figure 4.15: *mean fields profile at 1D downstream the cylinder surface*

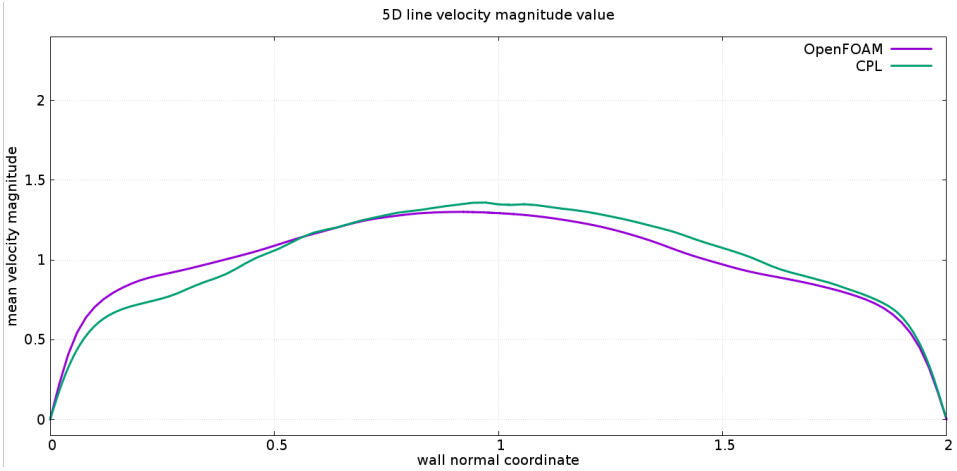


(a) *velocity*

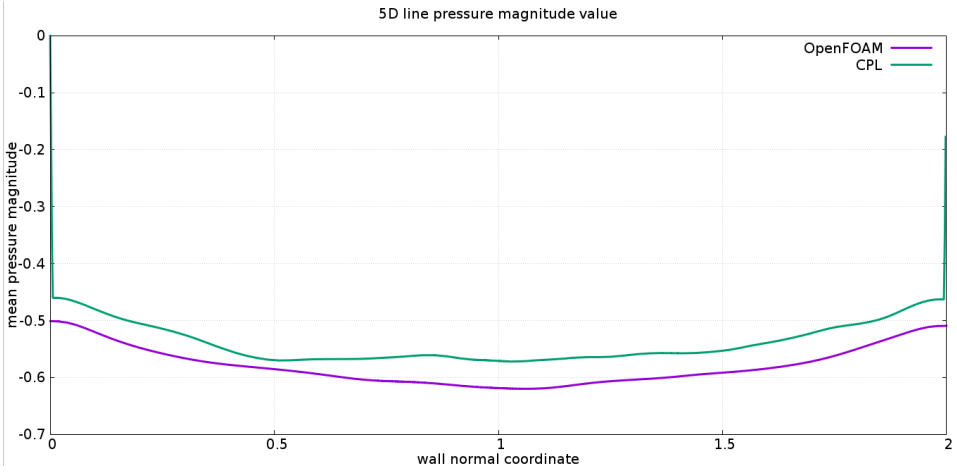


(b) *pressure*

Figure 4.16: *mean fields profile at 3D downstream the cylinder surface*



(a) *velocity*



(b) *pressure*

Figure 4.17: *mean fields profile at 5D downstream the cylinder surface*

The selected locations are of course the same for both OpenFOAM and CPL: centered along y axis, parallel to the z axis, these lines are found at 1/3/5 times the cylinder's diameter downstream with respect to the cylinder rear side, respectively.

It can be easily observed that the mean profiles coincide almost perfectly. The chosen locations for these evaluations are particularly relevant from the turbulence viewpoint; in fact, in those location the cylinder wake evolves

rapidly. For example, as can be seen from the plots, at 1D downstream the cylinder (fig. 4.15), it is present a wide recirculatory region. That region indeed coincide with smaller values of velocity magnitude, while on the sides of the channel the velocity is higher. This results confirm our expectations, since in this region the most external parts of the channel still possess a velocity field accelerated by the blocking effect.
Proceeding downstream, the largest turbulence structures decay progressively from fig. 4.16 to fig. 4.17, leading to a smaller scale turbulence.
We can observe this phenomenon by the increasing flatness of the velocity profile, which represents the mixing action taking place in the wake. Of course this behaviour still holds going on downstream. This perfectly reflects the channel mean velocity profile which become more and more flat, resembling the expected turbulence mean velocity profile for a turbulent channel flow. Confined channel flows - with developed turbulence - are in fact characterized by a flat velocity profile.

4.1.4 Standard deviation profile

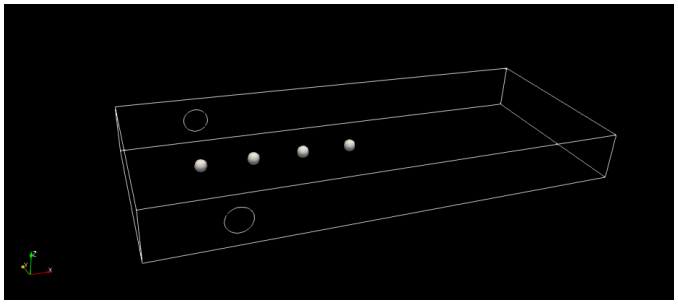
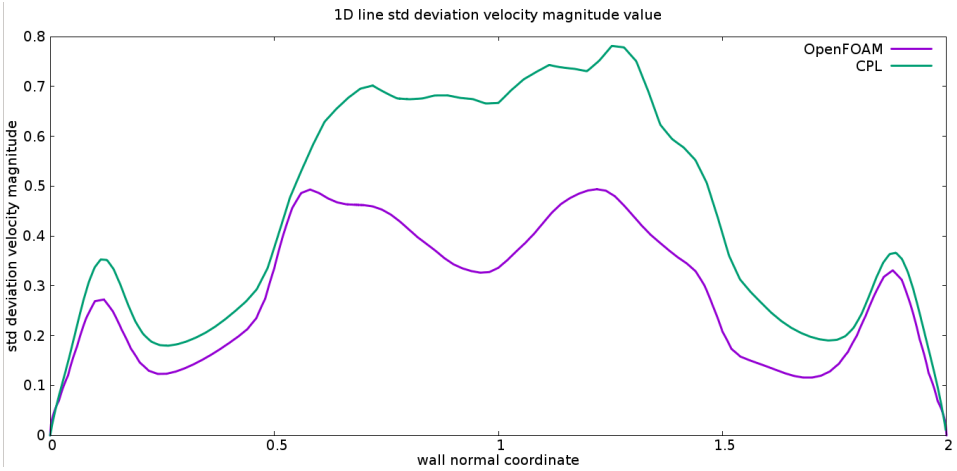
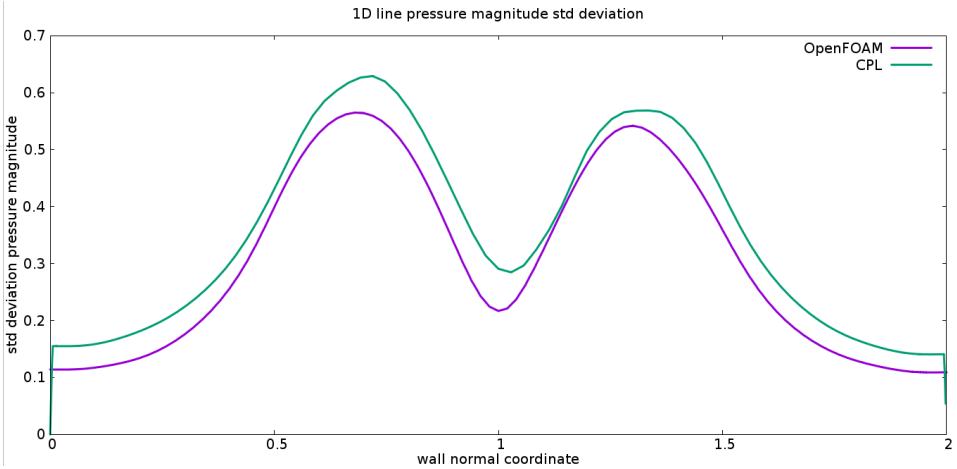


Figure 4.18: *lines location*

Through the paraview filter *PlotOverLine*, it was possible to collect the profiles for the standard deviation of both pressure and velocity magnitude.
The chosen positions and the plotting lines are of course the same as in the previous paragraph; these profile visualizations, along with those, will help the understanding of the surface visualizations found before in this chapter.

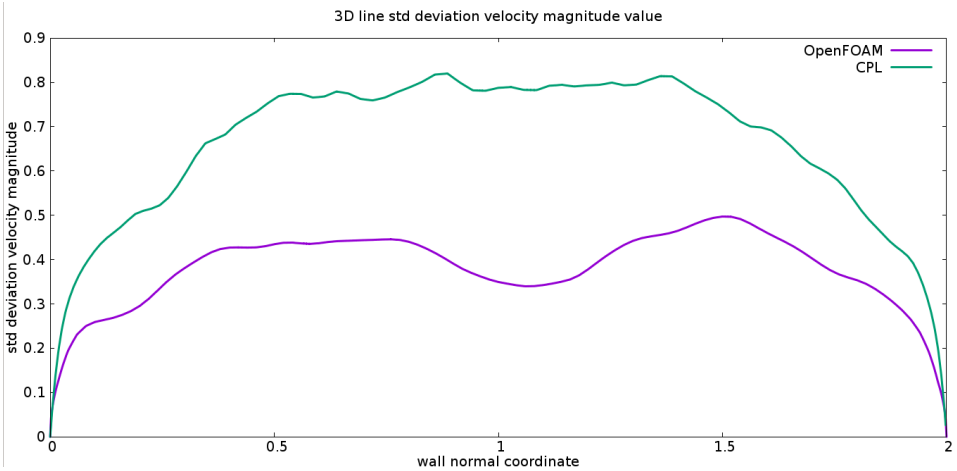


(a) *velocity*

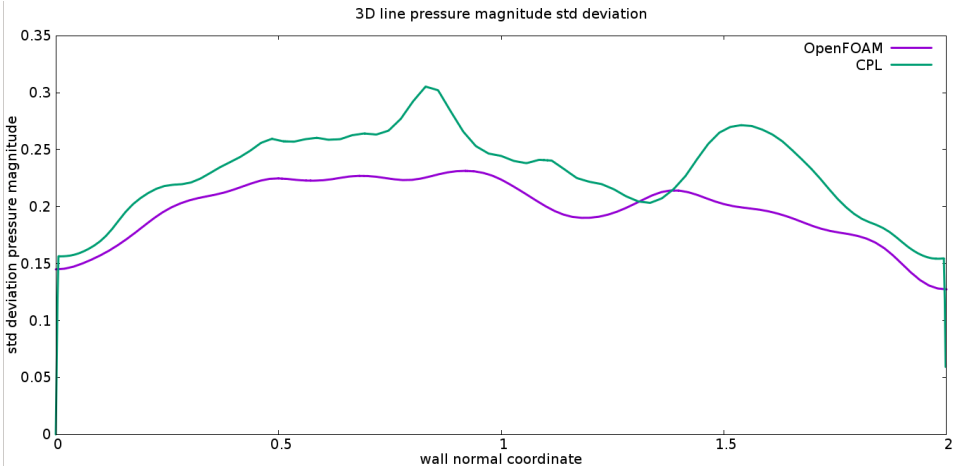


(b) *pressure*

Figure 4.19: *standard deviation profile at 1D downstream the cylinder surface*

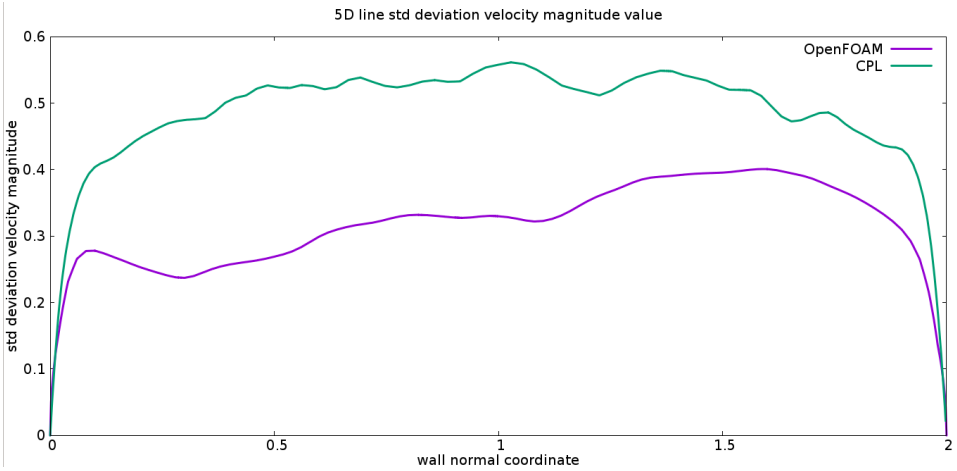


(a) *velocity*

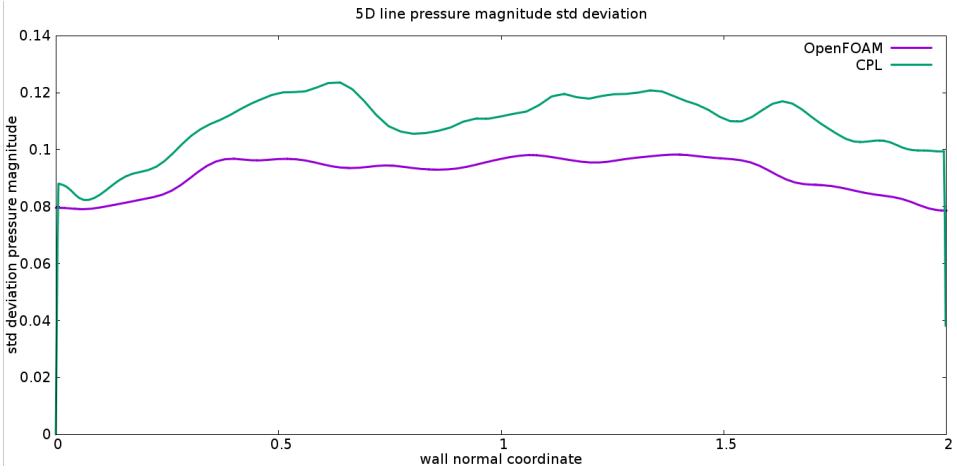


(b) *pressure*

Figure 4.20: *standard deviation profile at 3D downstream the cylinder surface*



(a) *velocity*



(b) *pressure*

Figure 4.21: *standard deviation profile at 5D downstream the cylinder surface*

It is straight-forward to observe that here, in spite of what observed for the mean fields, we have a different behaviour for the two simulations. This was foreseeable from the variance visualizations shown before in this chapter, but these graphs can confirm our analysis.
These differences in fig. 4.19 4.20 4.21 could be due to the smoothing action of the OpenFOAM solver, which acts cutting off the highest instantaneous values of the fields, mining the standard deviation computation. However it

is important to enlighten that the main reason is - with all probability - a limited data acquisition during the simulations.
The smaller values indicate a less sparse database for the discrete values of velocity and pressure, leading the simulations to different statistics in terms of skewness, probability distribution and so on.

Chapter 5

Performance of the codes

In this chapter the computational time results are evaluated, for both the codes and following different strategies.

5.1 Computational runtime as a function of the number of cores

The computational runtime is a key factor in computational fluid dynamics. Its values can determine whether or not a scheme or a method would be applied in the resolution of a problem; therefore into the technological world, the time factor influence new fields of research, the abandonment of a strategy and the spreading of new solutions.
As it is already been said, this factor is the main reason why the direct numerical simulations are still barely used outside academic research, even though they provide accurate and time-changing results.

It is then immediate to comprehend why an improvement in this way could represent an important step forward, in particular for the study of bluff bodies (which are particularly relevant in the engineering industry and problems).

Reference quantities The main reference value to be evaluated is the time per step. It represents the amount of seconds needed by the code to advance of one step during the solving process.
On this parameter is based our performance study as well as the comparison

between the two codes.
The ratio between the time per step for a serial calcutation and a parallel calculation is called speed-up parameter and is depicted in eq. 5.1. It would be namely defined as the ratio between the time required to complete 100 iterations in two cases, of course giving the very same result.

$$S_p = \frac{\Delta t_{serial}}{\Delta t_{parallel}} \tag{5.1}$$

Its value will of course vary along with the number of cores on which the computation is carried on, giving a curve which represents the ratio at which the code perfomance increase parallelizing the simulation.

Time step progression The time step progression is treated differently in the two codes.
The OpenFOAM dictionaries have been analyzed in the third chapter, and the chosen solvers have been described. These dictionaries have been edited in order to improve the performance, after some tests carried out in order to make the best choice possible.
In particular it has been chosen the pimpleFoam solver for the resolution and time progression. The tolerances set into the *fvSolution* dictionary are reported in the following schemes - as well as a brief recap of the used schemes.

For the pressure:

- solver GAMG;
- tolerance 1e-5;
- relTol 0.01;
- smoother DICGaussSeidel;

For the velocity:

- solver smoothSolver;
- smoother symGaussSeidel;

- tolerance 1e-05;
- relTol 0.1;

In the CPL simulation the chosen time progression scheme was the Runge-Kutta 3. It is a multistep scheme which divides each timestep into 3 parts to advance.
Because of this, the time that will reported further on in this chapter are related to the time needed for each one of the single sub-steps in comparison with one pimpleFoam step.

5.1.1 Hardware

The hardware used for this study (both the field-data production and the performance analysis) is the Marconi new Tier-0 system, co-designed by Cineca and based on the Lenovo NeXtScale platform, based on Intel Xeon Phi family alongside with Intel Xeon processor E5-2600 v4 (see [2] for further details).
The utilized part was a single node of the section A2, added at the start of 2017, equipped with the next-generation of the Intel Xeon Phi product family (Knights Landing), based on a many-core architecture, enabling an overall configuration of about 250 thousand cores with computational power of approximately 11Pflop/s. The system architecture provides each node with 68-cores Intel Xeon Phi 7250 CPU (Knights Landing) at 1.40 GHz and 16 GB/node of MCDRAM and 96 GB/node of DDR4.
To measure and evaluate the parallel speedup and performance, the number of requested cores was increased each time, whilst the RAM requested remained always equal to 86000 MB (the highest possible value for batch submitted jobs).

5.1.2 Results

The first thing worth precising is that the performed study is called strong scaling, namely the process where the number of cores changes over a pretermined discretized domain. More in depth, the total number of cells composing the domain is finite and fixed, and therefore whenever the number of processors increases, the fewer cells each processor will manage.
In order to have the most comprehensive and reliable data, the results have

been evaluated in different configurations and following different strategies. It is also important to remind that the geometry object of this study is characterized by 6.8 million cells which derive from the discretization of a global domain with dimensions $6\pi x8x2$ refined with $400x170x100$ cells on each of its dimensions respectively.

Data acquisition In order to collect the temporal data inherent to each calculation, it has been followed the following process:

- modify the code as not to include any postprocessing utility
- modify the saving logics of the code in order to avoid any undesired input/output operations corrupting the computational time measurement
- start the computation from a full speed solution for a certain amount of timesteps
- register the time to complete the first round
- start the computation from a operational condition solution for a certain amount of timesteps (smaller than the first one)
- register the time to complete the second round
- compute the difference between the two measurements as to cancel out the initial field reading operations and other unavoidable operations
- divide the neat measure by the number of iterations considered

The measurements here reported have been carried on multiple times, as to find eventual systematic errors and then evaluate an average between the single results.
In fact a certain liability of the time measures is inevitable, since running the very same application two times will give two different results. In order to limit this effect, also the time for iteration was computed on a large amount of timesteps before dividing it (e.g.: 100 iterations for the cpl measurements).

The performance of the parallelization is related to the scaling strategy, which determines the surface amount between subdomains. This parameter is in fact a relevant factor, determining a reduction in the code performance whether it is too high.
The single sub-domains related to each core should have a shape closer to a cubic one, improving the ratio between volume and surface of each one of them.
Therefore, the parallelization along the y-axis is not as profitable as the one along x. This way, the advantage of increasing the number of cores is soon overcomed by the time needed for the communication between them.

DNS codes comparison Below is reported the graph for the time per iteration comparison between the two DNS codes, for the optimized geometry setting of the parallelization logic.

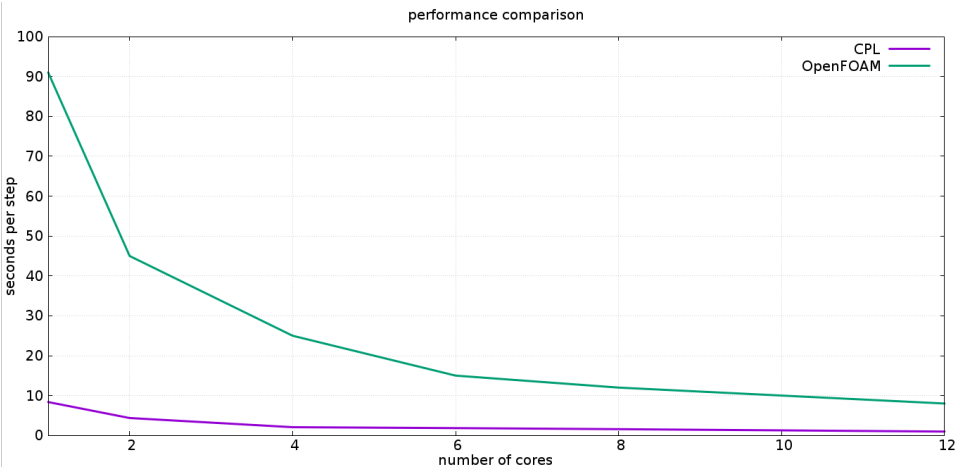


Figure 5.1: *Seconds per iterations as a function of the number of cores, for both CPL and OpenFOAM*

The results presented in fig. 5.1 are referred only to the computational time needed, meaning the required seconds per step. For the CPL code, the time has been obtained letting the simulation start from a operational field (at time 205 precisely) and, after a few iterations, measuring the time needed for one hundred interations. Measuring the time for two hundred iterations and than subtracting the one needed for one hundred, it was pos-

sible to subtract the time used by the machine to read the initial field and initiate the calculations. It is then immediate to divide the result by one hundred and then find the seconds per iteration.

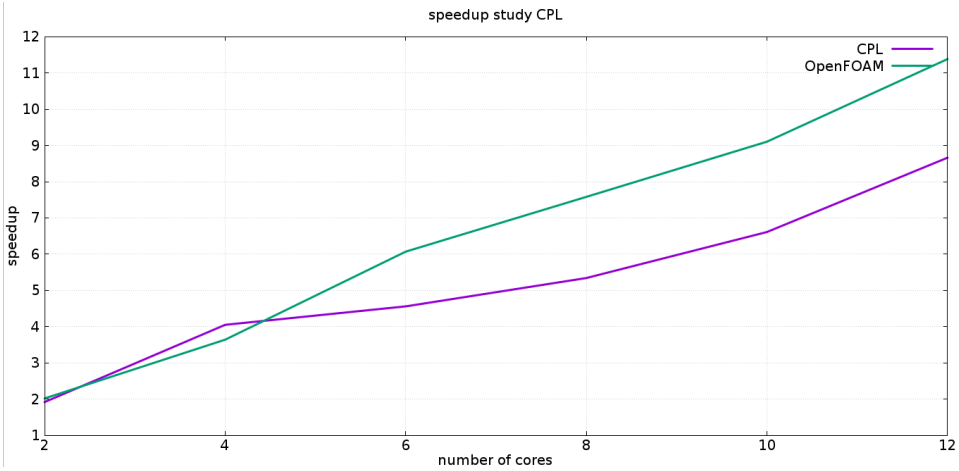
In order to have a better evaluation of the results, the data of figure 5.1 are also reported in 5.1. There can be also found the gain ratio between them.

The parallelization logic has been optimized for these final measures. Different and diverse attempts have been carried on for each setting of cores, on both OpenFOAM and CPL, in order to compare their performances at their very best setup. For example, the time per iteration needed with 4 cores was computed for all the possible setups; 2 processors along x and 2 along y, or 4 along x and 1 along y, and finally 1 along x and 4 along y.
At 12 cores the computational time per iteration was lower with a configuration of 6 cores along x and 2 along y. Before this value, keeping the simulation serial along the y-axis was the best choice. However, this difference with respect to the x-scaling results would probably hold on for bigger numbers of cores.

nCores	**CPL**	**OpenFOAM**	**gain ratio**
1	8.4”	91.1”	10.9
2	4.4”	45.0”	10.2
4	2.1”	24.9”	11.9
6	1.8”	15.0”	8.3
8	1.6”	12.1”	7.6
10	1.3”	10.2”	7.8
12	1.0”	8.1”	8.1

Table 5.1: *computational time per iteration of the different setups*

From this results it is immediate to evaluate the behaviour of the speedup



H

Figure 5.2: *speedup trend comparison*

parameter as a function of the number of processors.
It can be seen from fig. 5.2 that the perfomance of the codes are globally aligned, and that the overall time gain is about the same. The cpl speedup perfomances are not initially linear, with the curve slope changing sensibly at the beginning of the graph. However, as the number of cores increases we can notice that the ratio coincides with the one found in the OpenFOAM calculation.

The behaviour outlined in the graph is very close to linear, giving at the first stages of parallelization a conveniently predictable behaviour.
Moreover, we can conclude that the cutting-edge advantage given by the immersed boundary method is reliable and concrete. In fact, the cpl code continues to increase its perfomance increase for larger number of processors, echoing the OpenFOAM's behaviour.

The trend highlightened by fig. 5.1 then ensures that the better performances will hold whatever the structure employed to run the simulation.

Further comparison It is important to specify that the performance of the codes are also bound to the geometry of the domain. Indeed, along with the previous results, two other sets of measures have been computed to compare these two softwares.

A comparison of time gains for serial calculations has been performed, starting from a reduced geometry - 20 x 3 x 2 - discretized with about the same refinement - 400 x 100 x 100 cells. Two types of simulation have been carried on from this setup in order to study the features which influence the most the results.
In the first one a cylinder was built-in the channel, whilst in the second one the domain was just a classical - empty - channel flow. Both simulations started with an already developed turbulent initial condition, with a Reynolds number of 750.
The studies started from the beginning, redifining all the geometry parameters and optimizing both codes for either simulations.
The results briefly reported in tab. 5.2 are referred to the required seconds per iteration.

—	**CPL**	**OpenFOAM**	**gain ratio**
bluff-body	4.8”	66.4”	13.8
empty-channel	3.0”	110.4”	36.8

Table 5.2: *Comparison of gains of the IB code in different geometries addressed by serial simulations*

It is clear from these results that the geometry largely affects the immersed boundary performance, since each spatial dimension has to be scanned from the code affecting the usage of memory and the coordinate structures. In fact the immersed boundary strategy can be edited to proceed differently each time, depending on the shapes to be considered, as explained in the third chapter. This fact of course affects its performance, as it is clear from these last results.

Chapter 6

Conclusions and future developments

In this study a comparison between two different codes have been carried out.
The two codes have different solvers for the governing equation, different methods for the surface management, different parallelization schemes. In particular, an immersed boundary method - CPL -have been compared in both performace and field results with respect to a classical one - OpenFOAM.

Moreover, the study investigates a particular flow, in a non-trivial geometry.
The observed turbulence structures are consistent with the literature reviewed.
However, the results about the flow patterns suggest a limited sample for the data acquisition.
Mean fields are indeed coincident with one another although the second central moments differs in a non-negligible way. Therefore, it is difficult to evaluate the accuracy of the simulations.

The performance study carried out underlines a relevant time gain for the immersed boundary code. The gain ratio is about one order of magnitude; however, on this specific architecture, the parallel performances of CPL hold until a smaller number of cores.

Bibliography

[1] P. Anagnostopoulos, G. Iliadis, and S. Richardson, *Numerical study of the blockage effects on viscous flow past a circular cylinder*, International Journal for Numerical Methods in Fluids, (1996).

[2] Athlassian-News, *Cineca - hpc.* `https://wiki.u-gov.it/confluence/SCAIUS/`, 2019.

[3] A. Ben and H. Dou, *Simulation and stability study of the flow around a cylinder in infinite domain*, Procedia Engineering, (2015).

[4] CFD-Direct-Ltd, *Cfd direct.* `https://www.cfd.direct/openfoam/`, 2015-2019.

[5] J. G. Chen, Y. Zhou, R. A. Antonia, and T. M. Zhou, *Characteristics of the turbulent energy dissipation rate in a cylinder wake*, Journal of Fluid Mechanics, (2017).

[6] P. A. Davidson, *Turbulence: An Introduction for Scientists and Engineers*, Oxford University Press, 2004.

[7] ESI-Group, *Opencfd ltd.* `https://www.openfoam.com/documentation/`, 2016-2018.

[8] E. L. Houghton and P. W. Carpenter, *Aerodynamics for Engineering Students*, Butterworth Heinemann, 2003.

[9] Kitware, *Paraview.* `https://www.paraview.org/`, 2019.

[10] T. V. Kumar, P. K. Sen, S. Veeravalli, and M. Kumar, *Stability of weak confined wake behind a cylinder in fully developed turbulent channel flow*, Procedia Engineering, (2015).

[11] P. Mathupriya, L. Chan, H. Hasini, and A. Ool, *Numerical investigations of flow over a confined circular cylinder*, in Australasian Fluid Mechanics Conference, 2018.

[12] C. Monti, *Metodo dei contorni immersi per la simulazione numerica diretta di correnti turbolente su pareti non piane*, master thesis, 2016.

[13] C. Peskin, *The immersed boundary method*, Acta Numerica - 11, 2002.

[14] M. Pinelli, *Dns of a turbulent channel flow over a rough wall*, master thesis, 2017.

[15] S. B. Pope, *Turbulent Flows*, Cambridge University Press, 2000.

[16] A. Quarteroni, *Numerical Models for Differential Problems*, Springer, 2014.

[17] M. Sahin and R. G. Owens, *A numerical investigation of wall effects up to high blockage ratios on two-dimensional flow past a confined circular cylinder*, (2004).

[18] A. E. P. Veldman, *Boundary layers in fluid dynamics*, 2009.

[19] M. I. Yuce and D. A. Kareem, *A numerical analysis of fluid flow around circular and square cylinders*, Journal, (2016).