Master of Science Thesis

Coupling of a Finite Volume solver to a Hybrid Lagrangian-Eulerian Vortex Particle Code

Jorge Mario Tamayo-Avendaño

8 of May, 2017

Faculty of Aerospace Engineering · Delft University of Technology

Coupling of a Finite Volume solver to a Hybrid Lagrangian-Eulerian Vortex Particle Code

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering at Delft University of Technology

Jorge Mario Tamayo-Avendaño

8 of May, 2017

Faculty of Aerospace Engineering · Delft University of Technology

Delft University Of Technology
Department Of
Aerodynamics, Wind Energy, Flight Performance and Propulsion

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Coupling of a Finite Volume solver to a Hybrid Lagrangian-Eulerian Vortex Particle Code"** by **Jorge Mario Tamayo-Avendaño** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 8 of May, 2017

Supervisor: _____
dr. ir. Carlos Simão Ferreira

Reader: _____
prof. Niels Sørensen

Reader: _____
dr. ir. Alexander van Zuijlen

Reader: _____
dr. ir. Artur Palha da Silva Clérigo

Acknowledgements

After completing this work I could not be more thankful to my supervisors Carlos and Niels for their confidence in my work and their constant and unconditional support throughout my project. Both their expertise in the field of computational mechanics and wind energy as well as their human and professional quality have contributed enormously to my education even before knowing the outcome of this thesis.

I am also very glad to have the support of the researchers Artur, Carlos Baptista, Mikko and Ye from the Aerodynamics and Wind Energy departments; their insight widened my horizon and helped me to get curious about the world of scientific computing and programming. I am now eager to learn more about it and for that I will always be thankful to them.

Being a student in Europe gave me the chance to experience many cultures and ways of thinking, I had the chance to meet a big group of brothers, who were born in a far away continent but that still gave me more than I would have expected; Piyush, Sohil and Faisal have been a constant source support; likewise I must thank my fellow Senior Ashok for his guidance through the completion of my thesis and *Bhai* Raunak, who in the most genuine way has introduced me to Indian culture, making me feel part of it.

Finally I need to say that without my mother, Amanda, I would not have had the chance of even dreaming with being part of the TU Delft family. She has given me all the values I have and will continue to be the main reason to carry on.

Delft, The Netherlands
8 of May, 2017

Jorge Mario Tamayo-Avendaño

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The current status of the wind energy industry has opened new research opportunities in the field of aerodynamics, addressing problems that usually require an intensive amount of resources and time to be analysed. The increasing level of detail in the study of wind turbines is reaching instances, in which turbulence and wake analysis are of great importance for the development of reliable studies in interdisciplinary areas of wind turbine technology. As this refining process moves forward, two essential requirements hold for any simulation tool: accuracy and efficiency. This means that suitable mechanisms of computation constitute an important demand and, most likely will continue to do so in the long term for the wind energy industry.

The aerodynamic behaviour of wind turbines has many levels of complexity and therefore can be analysed with different approaches, each with a particular degree of sophistication and accuracy. One important example is the study of wake shedding and interaction with solid bodies. Some times, this cases can be studied experimentally, as in the aviation industry or small wind turbine applications where scaling and availability of test facilities enable such methodologies. In other cases, involving subjects of larger scales, as in the analysis of large wind turbines or even wind farms, experimentation starts to offer its advantages and numerical analysis become the most convenient choice if not the only available.

Among the set of numerical tools that are often used to simulate the flow around airfoils, the Eulerian solvers based on the solution of the Navier-Stokes equations (NS) on meshed domains and Lagrangian solvers such as Vortex-Particle methods are prominent examples; in fact these two are the main components of the hybrid solver under consideration in this work. The Eulerian solvers, implemented with several techniques such as Finite Element Method, FEM, or the Finite Volume Method, FVM, have the advantage of being popular and robust; and usually give very good results in the prediction of boundary layer flows, an attractive characteristic for the purpose of this work. On the other hand, Lagrangian analysis, specifically with a Vortex Particle Method, represents a different alternative inside the spectrum of CFD, where the problem is formulated in terms of vorticity allowing an optimum prediction of flow structures with much larger dimensions than those found

in a boundary layer [14] and, with comparative advantages in terms efficiency and absence of recurrent drawbacks of standard Eulerian solvers, such as numerical diffusion [10].

The mentioned flow cases of large wind turbines and wind farms are of special interest because the simulation of the wake is a vital part of it. In the simulation of the generation, convection and interaction of a wake, many characteristic length scales are involved. When a case of this nature is analysed with a fully Eulerian or a fully Lagrangian solver, several drawbacks arise, due for example to the numerical diffusion or due to the lack of accuracy in simulating solid walls respectively. In response, solution techniques based on a combination of the two types of solvers have been proposed; these segregate the Eulerian solver to the near-wall region and the Lagrangian solver to the remaining portion of the domain.

In the present this methodology has already been studied and a substantial role inside the analysis is given not only to the Eulerian or Lagrangian components alone, but also to the component specifically addressing the process of information transfer between the domains of the main solvers during the simulation process.

The improvement on the implementation of the different solvers as well as the transfer of information between them both constitutes the framework in which this project has been defined. Taking as a starting point an existing hybrid solver, the tasks proposed here aim to expand the range of applicability of the solver, by implementing a more robust Eulerian solver and by ensuring that such a solver is correctly integrated in the evolution algorithm employed by the hybrid flow solver.

1.1 Research question, aims and objectives

Objective

The objective of the present research project is to determine an improved implementation of the hybrid flow solver pHyFlow, aiming to extend the applicability range of the software by substituting the current Eulerian component, used since early development stages, with a Finite Volume Method based solver and by testing the implementation of this component with simulations of typical flow cases.

Research Questions

At this preliminary stage the research project has been structured in a way that three central questions have been proposed:

- What are the main implications of coupling a FVM solver inside pHyFlow?
  - What modifications are required in the existing interpolation scheme and/or any other module that interacts directly with the Eulerian solver module?
  - To what size can the Eulerian sub-domain be reduced without degrading the quality of the solution?

  - Are there any reported or expected changes with respect to the quality of the solution or solution time?

- After verifying the modified version of pHyFlow (i.e. that with the FVM solver) How does the new solver performs?
  - Does the FVM solver works properly through the new interface (implementation in Python of OpenFOAM)?
  - Does the FVM performs as expected with respect to the FEM solver?
  - Does pHyFlow with the new Eulerian solver performs as expected with respect to the original framework?
  - What is the result after benchmarking the new version of pHyFlow with respect to selected flow cases?

- What can be concluded from the final version of pHyFlow?
  - What are the limitations of the new framework?
  - What final advantages are identified?

Chapter 2

Literature Review

2.1 Preamble

One of the most challenging applications of CFD in the present day is the prediction of a flow field with vortical structures and turbulence, two flow conditions that can exist separately or simultaneously depending on the approach of analysis to a specific case. The available simulation techniques vary in accuracy and execution time depending on the nature of the subject under study. Wind energy applications are interesting examples in terms of the diverse flow scenarios that can arise, such is the case of the study of a boundary layer in a wind turbine blade on one end of the spectrum of complexity and, the analysis of large scale wind turbine wakes on the other.

The study of viscous, turbulent flow around solid bodies is a complex task because on one hand, many length scales are involved and, on the other hand, the level of accuracy of a numerical method is often tailored to predict more accurately the flow field at the most fine scales, i.e. turbulent flow structures. In wind energy related flow cases, such as blade-vortex interactions, dynamic stall or the flow over a wind farm, a compromise is an essential goal, in order to allow an efficient and yet accurate prediction of the flow properties. The relevance of this key criteria is best illustrated by considering that the flow characteristics in the mentioned cases cover a relatively wide range of length scales and that all of them correspond, in the practice, to turbulent flow situations with different characteristics.

One of the approaches for analysing flows with multiple length-scales consists of setting a hybrid solution strategy that blends two different methods for the simulation of different flow regions of a same flow case, a Lagrangian method for the flow in the far-field of the domain and, an Eulerian method for the flow in the near-wall region. These regions are different from a phenomenological point of view and are solved by numerical tools, designed to work with best accuracy for the locally predominant flow conditions. This approach receives the name of *Lagrangian-Eulerian domain decomposition*, as reviewed in basic texts [7] and is the central research object of both the project and therefore the review presented here.

The domain decomposition technique has secondary subjects of interest that have been identified from the early stages of the project conception and planing. Both the Lagrangian and the Eulerian solver constitute each a subject of review as their detailed structure determines the way in which the coupling, the third relevant subject, is to be executed in pursue of a successful simulation tool in terms of the previously mentioned criteria. Many of the sources explored in this review present mostly knowledge on Vortex Particle Methods, VPM, the Lagrangian component of the hybrid solver. Other sources present information on the Eulerian component, for which a popular choice is the Finite Element Method, FEM. This project however considers a Finite Volume Method, FVM, solver, as the Eulerian component. Finally the most relevant topic is the coupling strategy between both solvers, and is at the centre of the present review.

One indicator of quality for a domain decomposition technique lies in the smoothness of any flow property across the boundary between the Lagrangian and Eulerian sub-domain; ensuring this condition is one key aspect during the execution of the project as more than one approach for its enforcement has been identified in the existing works. The correct matching of the flow solution between sub-domains, depends on the mechanism for information transfer and the evolution algorithm. This two aspects are of special care since the present review aims to shed enough light in the coupling of a FVM code to a PVM code in a framework for hybrid solution of the flow field around an airfoil and a cylinder at moderate to high Reynolds numbers.

The content of this chapter is structured mainly inside section 2.2, which contains the bulk of the survey distributed into subsection 2.2.1 with most of the information on the use of vortex methods and techniques for efficient computations. Subsequently the most relevant information on the coupling methods is contained in subsection 2.2.2 followed by a review on a couple of Finite Volume Methods applications in multiple domain cases in subsection 2.2.3. The final sections contain the review summary, in section 2.3 and the concluding remarks in section 2.4.

2.2 Survey

2.2.1 Vortex methods in fluid mechanics

The basis for the VPM model is the formulation of the Navier-Stokes equations in terms of the flow velocity and vorticity, $u - \omega$, according to Chorin [3], whose work is one of the earliest in the subject. The model proposed in [3] is composed, in simple words, of a transport equation for vorticity 2.1, and a second relation 2.2, known as the Poisson equation, linking ω and the stream function, ψ, which is inherently related to the flow velocity.

$$\frac{\partial \omega}{\partial t} + (\vec{u} \cdot \nabla)\,\omega = Re^{-1}\nabla^2\omega \tag{2.1}$$

$$\nabla^2\psi = -\omega \tag{2.2}$$

On the side of the Eulerian problem, viscosity dominated flows are considered following what can be described as a standard setting; the Navier-Stokes equations are formulated in terms of the flow velocity and pressure $u-p$, to be discretized in a meshed domain, with the wall boundary condition on the solid surface and the external boundary condition set from the coupled computations, see for example the work of Papadakis & Voutsinas [22].

With the existing basis for the vortex based methods, further work has been done, with adaptations in both the modelling and hardware resources that aim to increase the efficiency of computations and also their accuracy to describe a particular flow case of interest. Vortex methods have a relatively wide range of applications and are constituted from diverse conceptual elements as will be elaborated next.

The use of discrete particles as the base for modeling inviscid flows with vortical motion structures constitutes an approach for solving flows in a simple way but in cases for which detailed a discretization is required, i.e. a large number of particles, it becomes impossible to obtain smooth solutions. As pointed by Leonard [18], a way of dealing with the problems caused by the singular nature of particle vortices is the use of finite size elements, or blobs; elements with that are assigned a certain amount of vorticity distributed in a surrounding region. This conceptual approach to vortex methods allows to obtain vorticity fields that resemble closely the physical reality of the flow under study, as no singularities are included, and with induced velocities that, unlike with conventional particle vortices, remains bounded.

A recent work made by Yokota and Obi [28] presents a general review on the functionality of vortex methods including the aspects of accuracy, efficiency, application cases and existing acceleration techniques. A brief introduction to the $\omega - u$ formulation of the Navier-Stokes equations is made, similar to that in [3], and is followed by an analysis on the efficiency of the solution. The presented review offers interesting results regarding the computational effort of this model with different implementation and solution techniques. The use of Fast Multipole Methods, FMM, proposed in [11] as an alternative for the speeding of the computations in VPM is set to test in the computation of the Biot-Savart equation, that is implicitly contained in the governing equations of the VPM model. The findings of this comparison shows a reduction in the order of the computational cost from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, for a problem with N particle elements.

A second and relatively newer acceleration technique, the use of Graphical Processing Units, GPU, is also set to test by solving the same Biot-Savart calculation for several number of particles and, with a default VPM and a FMM solver. As expected, the computational effort reduction between the default and FMM implementations hold the same as mentioned previously, however, the use of the GPU reduces the computation time, according to the results presented by the authors, by almost two orders of magnitude. Further application of pure VPM by [28] in the simulation of turbulent flows, allows to demonstrate a level of accuracy that compares to a cell-based DNS method; however a large number of particles is required, approximately three orders of magnitude above the typical number of cells required by a FVM solver when predicting flows with moderate Re, i.e $Re \approx 1 \times 10^6$.

Both the application of GPU and FMM acceleration techniques seem to be at the centre of contemporary advances in the field as reflected now, in the work of Hu et al. [15]; who present a study of vortex rings in mutual interaction. A detailed analysis is done over the

algorithms that make the FMM implementation and their influence on the error of the method. A demonstration in the reduction of cost of the FMM accelerated VPM with respect to the baseline model is shown, arriving to conclusions that confirm those in [28], i.e. the FMM acceleration results in a cost reduction from second order to first order. Similarly the authors conclude that the approximated computation of particle interaction in the far-field, opposed to the exact computations in the exact near-field interaction, do not have a significant impact in the accuracy of the overall solution.

It is worth to mention here, that the use of hardware acceleration techniques other than GPU, have been explored by Sheel et al. [23] who compares the performance of a specifically tailored computer against a common parallel computation for the solution of an N-body problem. The results over the use of such specifically dedicated machines is favourable as explored also in [24]; however these computational resources are less available than the more recent GPU technique.

2.2.2 Coupling algorithm between Lagrangian and Eulerian components

The present object of study contemplates two modular solvers working simultaneously, the Lagrangian component, constituted by the vortex method, has been discussed already, and now a survey on different concepts for the coupling with the Eulerian component is presented; covering the relevant characteristics of each one.

One vortex based solution approach is the Vortex-in-Cell method, VIC, in which the solution of the Poisson equation takes place on a meshed domain in what is therefore an Eulerian solution step. This is a kind of classical hybrid approach in the sense that the Eulerian solution takes place in a domain that overlaps completely the Lagrangian domain, with the second being a region where the vorticity is concentrated and solved. The essence of this model proposed in [4] and employed in [18] consists on a different solution approach for each one of the governing equations.

The VIC has also been object of improvement by implementation along with a fast multipole method as is the case of Cocle et al. [5] who have used this improved method for the study of instabilities of vortex rings. Besides giving a glimpse on the capability of this particular method to account for infinity boundary conditions, good capabilities in the simulation of turbulence are also demonstrated. Nevertheless, the applications for which the VIC method has been used, in this review, deal with unbounded or partially unbounded flows. This somehow reflects a weakness since no demonstrated advantage seems to exist for the simulation of near wall flows.

The VIC model offers an interesting characteristic because when the Eulerian mesh covers the Lagrangian region completely, no boundary or interface exists, eliminating the need for matching properties at any interface, nevertheless it seems that such method is not popular in the prediction of flows with near wall boundaries. This situation puts in the spotlight a second kind of Lagrangian-Eulerian solvers, in which equations 2.1 along with the Poisson equation is solved in a fully Lagrangian discretisation while the Eulerian solution is applied strictly to the $u - p$ formulation of the Navier-Stokes equations.

The approach taken by Guermond et al. [14] has the characteristic of solving the Eulerian and Lagrangian problems over sub-domains that do not overlap mutually but instead share

a common interface, namely a line for a 2-D case. Similarly, the way in which information is transferred from one sub-domain to the other is distinctive. In order to model the flow in the Lagrangian domain, equations 2.1 and 2.2 take as boundary condition the vorticity values in the Eulerian domain, determined by a transmission condition that has the role of ensuring continuity of the values on each side of the interface. In the Eulerian sub-domain, the Navier-Stokes equations are formulated in terms of the stream function and the vorticity, $\psi - u$, and the boundary conditions are set differently for each variable since each one is modelled by a different kind of equation. The stream function is solved with a Dirichlet type boundary condition determined from a transmission condition of integral type. The boundary condition for the vorticity is set in a similar way as for the Lagrangian domain; with one expression determining the boundary and transmission conditions for each portion of the Eulerian domain through which the flow is entering or leaving; this is due to the kind of PDE modelling the vorticity and the specific requirements associated to it.

The fact that the information transmission happens at the same time that the boundary conditions are defined, is the relevant aspect in this case and marks one of the alternatives in terms of coupling of the two systems. Similarly, the approximation of the problem via hyperbolic PDEs makes that the definition of the boundary condition is linked to the direction in which the flow is crossing the interface between sub-domains; this might seem simple at a first glance, but when considering the case of flow with vortical structures external to the body under analysis, new questions can be formulated, specifically because vortical structures would imply a nonuniform boundary condition for the portion of the Eulerian sub-domain where the fluid is flowing outwards.

The use of hybrid solvers with overlapping domains, a different coupling concept, has been done in different works, Gao et al. [10] for example use it to analyse several flow cases, with the generation and reflection of shock waves being one of them. Another prominent example is that of Stock et al. [25] with a work that gives a good view on the interpolation of data during the hybrid simulation. The solutions in each sub-domain are consistent by means of suitable interpolation techniques and with certain restrictions. The importance of avoiding interpolation from points that are too close to either of the boundaries of the Eulerian domain is highlighted in this study; as reasons, the authors point different assumptions between the Lagrangian and Eulerian models for the outer boundary and the high gradients that exist in the near wall flow in the case of the wall boundary. The coupling algorithm in the referred work is defined in a way that the Lagrangian solution provides the essential information for the Eulerian boundary conditions; once the Eulerian solution is obtained, the vorticity in its own domain is updated and subsequently the Lagrangian solution moves forward to the next time step.

The work of Stock et al. [25] has a wide validation campaign that includes the analysis of a flow over a low-Re sphere, a wing tip vortex and a full four-bladed rotor. The analysis of the sphere with $Re = 100$, revealed that the flow field is not affected at the overlapping region, even in the recirculation bubble that exists immediately downstream the body of the sphere. In the case of the rotor analysis, the authors point a good solution behavior as convected vortices enter the Eulerian sub-domain after being transported along the Lagrangian wake region.

A coupling between a finite difference and a VPM has been performed by Papadakis and Voutsinas [22], in order to analyse the computational cost reduction that can be

obtained from a hybrid approach and the implementation of a Particle Mesh, PM, method in the Lagrangian solution procedure. Besides modifying the Lagrangian formulation which in principle considers incompressible flow, in order to comply with the compressible formulation of the Eulerian system, the authors have established a coupling procedure that aims to guaranty smoothness of the solution when moving to particle locations just outside the Eulerian sub-domain. Such a coupling has two fundamental parts, on one side, the Eulerian solution is used to correct the particle mesh solutions placed inside the Eulerian domain and, on the other side, the particle mesh solution is used to determine the boundary conditions for the solution of the Eulerian sub-domain. At the end of each time step, the latest available information is used to determine the Eulerian boundary condition, ensuring that the computation is strongly conservative. The analysis includes a comparative simulation of a symmetrical airfoil from which the authors have determined that when the Eulerian domain extends away from the wall to a distance between 1 and 1.5 times the chord, a good agreement is reached between hybrid and full Eulerian simulations.

- From Papadakis and Voutsinas [22] the implementation of matching sub-domains, in which the continuity of properties across the boundary is enforced by means of transmission conditions of integral type, can lead to a severe penalty in computation time when used to predict 3D flows.

- From the solution technique over the Lagrangian model, an approach is taken to resolve both the stream function and the velocity potential; instead of using the Green theorem, known in similar works, these quantities are obtained with Poisson solvers applied to structured grids, but later an interpolation takes place in order to assign the computed properties back to the particles on which the Lagrangian approach is centred. This approach is relevant to the present work because, a cost reduction alternative is presented by solving the core of the particle-based Lagrangian problem via a structured grid and, also because this methodology includes the interpolation between a structured grid and a set of unstructured particles by means of appropriate interpolation functions such as the M_4'.

- The consideration of viscous effects by Papadakis and Voutsinas is done via the Particle Strength Exchange, PSE, which according to their work, is a convenient choice for problems with constant viscosity. In their work, turbulence is included, this means that the existence of 'mechanical' dissipation due to turbulent viscosity could complicate the modelling of this phenomena, however, the Lagrangian sub-domain describes convection dominated flows, for which the viscosity of molecular nature can be considered as constant.

- The analysis presented in terms of computational cost leads to the conclusion that, it is difficult to see any improvement for the study of 2D cases. The efforts made by Papadakis and Voutsinas in solving the particle problem with a grid setting does not show a bigger improvement as does the reduction of the Eulerian domain size, under the argument that when reducing the size of the Eulerian domain, the errors inherent to the boundary regions propagate more rapidly throughout the sub-domain thus favouring convergence.

One popular coupling methodology between the Lagrangian and Eulerian sub-domains is the Schwartz iterative method, for which a comprehensive explanation regarding its use in hybrid computations can be found in [7]. This iterative method allows to solve the Poisson equation for ψ in each sub-domain boundary, setting as the boundary condition the computed value at the adjacent sub-domain at the immediately preceding time step. This procedure iterates until the solution for ψ converges and the velocity field derived from the stream function is found everywhere.

In the spirit of performing the same transfer of information between sub-domains, a method is proposed by Daenink [8] which uses no iterative procedure. Instead a simple interpolation of values is proposed to obtain the boundary conditions of the Eulerian sub-domain from the Lagrangian solution. This alternative constitutes a simpler choice than the Schwartz iterative method introduced earlier.

The use of finite difference methods along with VPM has been explored in multiple ways by Ould-Salihi et al. [20] in a study of two solution methods, the VIC, and the "Particle-grid" domain decomposition method, an approach in which the Eulerian solution is applied to the pressure-velocity formulation of the Navier-Stokes equations and that is proposed in [6, 7].The nature of this particular work, allows to observe the differences between VIC methods and the traditional Lagrangian-Eulerian domain decomposition, showing the broad extent to which Eulerian computation can be blended over the whole domain.

2.2.3 Finite Volume Methods in Hybrid Solvers

One interesting work is presented by Zang and Street [30] who develop and test a composite numerical method for the solution of the N-S equations in multi-grid domains. The particularity of their research is the use of an Eulerian solver for the entire domain which brings a rather important difference with respect to the present project: the transmission of information and the treatment it receives in order to ensure the mathematical consistency of the method happens as an integrated part of the algorithm in which the governing equations are solved. Contrasting with this, it has been already pointed that the present method executes the Lagrangian and Eulerian solutions as separate stages, with the transmission of information as a modular step between them. Another Important aspect is the need of a correction on the velocities at the boundaries between sub-grids, which despite being overlapped, have the same requirement of conservativeness inherent to the finite volume method.

The need for a correction for ensuring conservativeness of the method raises from the interpolation that is carried out in order to determine the boundary condition at the internal boundaries. The convergence and consistency of this method is thus determined by the accuracy of the interpolation scheme that takes care of projecting the information from the internal domain to the required boundary regions.

The imbalance in the mass flow rate through the boundaries is managed by correcting the values of the velocity by an amount which is proportional to the volume flux at each cell face. The particular development and application in [30] shows how the mass flux correction becomes part of the method as it is performed inside the predictor-corrector algorithm. More precisely during the solution of the Poisson equation for the pressure.

A relatively recent work by Burton and Eaton [2] presents a finite volume method implemented around an overlapping grid approach. In their study, the authors consider a fractional step method to solve the incompressible NS equations with a staggered grid for the arrangement of the flow variables and an implicit/explicit hybrid time scheme. These are contrasting items with respect to OpenFOAM [27], the Eulerian solver to work with in this project, in which the flow variables are arranged in a collocated grid and where an implicit time advancement scheme is used. The authors explore the time and spatial accuracy of the method paying attention to the size of the overlapping domain. The most relevant findings point to an unchanged time accuracy with respect to the size change of the overlapping region; this conclusion is not the same for the spatial accuracy which seems to be inconsistent when the the overlapping is reduced.

2.3 Analysis of the current literature state

The improvement process suffered by vortex based methods for the prediction of complicated flow cases includes several refinement stages, covering improvements in both accuracy and efficiency of computation. The application of hybrid solvers to the analysis of unbounded flows around solid bodies of arbitrary shape has proven to be a not so trivial objective. When a study case involves regions with convection dominated flow and regions with diffusion dominated flow in the same domain, the use of vortex methods alone is not an ideal alternative.

The basic VPM approach that was the central piece in early researches [3, 18, 7] has been extended by the use of hybrid approaches with the goal of reducing the computational cost. This is done in a first approach by solving the Poisson equation, part of the vorticity based model, in an Eulerian mesh [18, 5]. Further improvements are done in a second kind of treatment in which the Eulerian solution can include the Poisson equation as well but also and most importantly, the solution of the Navier-Stokes equations in the $u - p$ formulation, [22, 25], for a more appropriate description of the near-wall flow.

The use of the VIC method by Ould-Salihi et al. [20] has an interesting contrast with the first application case selected by Cocle et al. [5], an unbounded vortex ring. The contrast lies in the fact that the first author performs simulations of confined flows, namely a lid-driven cavity flow and several cases of vortex-wall collisions; the second author instead, simulates the instabilities around an unbounded vortex ring in which the far-field condition can be satisfied by setting a grid much larger than the region of discretised particles, this does not happen in the case of bounded flows, as the Eulerian domain for this kind of situation does not exceed the size of the Lagrangian domain. The difference in approaches for the application of the same model denotes its flexibility, highlighting the appropriateness depending on the nature of the flow being simulated.

The application of Finite Volume Methods to the analysis of problems with decomposed domains seems to be a rather unexplored topic when it comes to integration of FVM as the Eulerian component to vortex particle-based methods as the Lagrangian complement. In spite of what could be interpreted as a demarcation line of the current developments, study cases such as the prediction of flows around moving bodies and complex geometries have made possible to implement FVM for the solution of the N-S equations, making use

of individual sub-domains that are connected, similar to those in a hybrid vortex particle method.

The present survey on the current state of implementation of VPM along with CFD methods leads to identify one important situation: in general, the techniques that segregate domains into far and near-body meshes, and that use FVM formulations usually take this approach for the flow prediction everywhere [29, 30, 2]. No use of vortex particle method along with FVM has been identified, and the concept of 'overset' or overlapping meshes along with FVM aims generally to simplify the problem by enabling the use of structured meshes and by simplifying the mesh discretisation in problems involving the motion of one or more bodies inside the fluid.

The coupling strategies consist basically on three different approaches: the integral transmission conditions for matching sub-domains used in [14]; the coupling of overlapping sub-domains via the Schwartz iterative method, which presumably imposes some degree of extra computation effort but that at the same time has been analysed in more depth. Finally, the non-iterative method for overlapping domains, proposed in [8] seems to be the most simple approach of all, since no iterations are required, the interpolation performed by this method is expected to be an efficient and fast procedure.

2.4 Concluding remarks on the literature survey

A general review has been performed over the most important aspects of hybrid Lagrangian-Eulerian flow simulations via vortex particle methods; from a review that has given an insight into the standard vortex models and coupling methods to Eulerian modules.

It can be stated that the domain decomposition technique does not necessarily restricts to using traditional CFD solvers in the proximity of solid bodies; since methods such as the Finite Differences can be used to compute particle velocities as is the case of the VIC method; nevertheless the traditional CFD solutions, namely FVM, offers a much more valuable resource for the prediction of near-wall flows.

A general survey on hybrid Lagrangian-Eulerian solution methods has been done allowing to identify common methods for the increase in accuracy of the solution and for the increase in efficiency. Similarly it has been observed that up to some extent, the chronological order of the reviewed works point in their development to the application of more practical flow cases in both 2D and 3D.

Since the use of vortex methods has been dedicated to the prediction of flow far from solid walls, it can be said that the use of an Eulerian solution in the vicinity of this kind of boundary contributes to the hybrid model by offering a more adequate formulation. The Lagrangian model on the other hand, has been object of relatively recent improvements such as the use of vortex blobs, the combination with FMM and the solution with GPU hardware resources; aspects that increase the accuracy and efficiency of the solver respectively. Regarding hardware acceleration techniques, the present project must be restricted to the use of parallel computation and the use GPU power, since these are

available alternatives for which favourable reviews have been made.

From the varied coupling methods, the non-iterative coupling seems to be a good choice as a starting point. The reduction of time by avoiding iterative calculations may result in an advantage for the development of a new solver as is the case of the present project. Improvements by testing more advanced coupling techniques can be done upon this initial work.

Chapter 3

The Finite Volume Method

All flow properties in the domain surrounding the boundaries of solid bodies are predicted with an Eulerian solver. The present project makes use of OpenFOAM as the base implementation of the Finite Volume Method (FVM) and this chapter is dedicated to outline the components of such an approach. The chapter introduces in first place the governing equations for the prediction of incompressible and laminar flow in 2-D. Subsequently a presentation of the discretisation schemes and boundary conditions is done. In the latter aspect a key differentiation is made between the boundary conditions used in conventional external flow applications and the use inside the Hybrid solution approach presented in this project. The end of this chapter is dedicated to the description of the $p-V$ coupling and the solvers for each one of these flow variables.

3.1 Governing equations

The governing equations presented in this chapter describe the incompressible flow in the proximity of solid boundaries, the flow condition being targeted by the Eulerian component. The set of equations about to be presented can be derived by performing a Taylor series representation of the conserved quantities along with a subsequent balance over a differential volume of fluid. Such a procedure is omitted for the sake of brevity.

The first of the equations that model the viscous flow in 2-D is the equation of conservation of mass which is nothing but the representation of the continuity law: the rate of change of mass in a volume of fluid equals the net flow of mass through the boundaries of such a volume, the differential form of this equation is as follows:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \left(\rho u_i\right)}{\partial x_i} = 0 \tag{3.1}$$

The second set of equations, central to the model, describe the conservation of the linear momentum of the fluid volume, by relating the rate of change of linear momentum to the sum of the external forces acting on the fluid volume. Once more, the equation is presented in differential form as:

$$\frac{\partial\left(\rho u_i\right)}{\partial t}+\frac{\partial\left(\rho u_i u_j\right)}{\partial x_j}=\frac{\partial T_{ij}}{\partial x_j}+S_i \tag{3.2}$$

Note that the right-hand side in equation (3.6) represents the forces acting on the volume of fluid, with a source term represented by S_i and, the stress tensor represented by T_{ij} and defined as:

$$T_{ij}=-\left(p+\frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\right)\delta_{ij}+2\mu D_{ij} \tag{3.3}$$

The tensor D_{ij} accounts for the strain rates in the volume of fluid and is defined in the following way:

$$D_{ij}=\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right) \tag{3.4}$$

For applications of interest in the wind energy context, and, in general, low speed aerodynamics, the consideration of flow speeds no greater than 0.3 times the local speed of sound allow to assume that the change of ρ is negligible. In this case the form of the equation of conservation of mass changes to:

$$\frac{\partial u_i}{\partial x_i}=0 \tag{3.5}$$

The simplification of the equation for the conservation of momentum follows a similar path since ρ can be simplified from the transient and convective terms in the left-hand side of equation (3.6) and be set to divide the diffusive terms, i.e. those in the right-hand side of the same equation. Besides, the new form of the continuity equation can be used to cancel the second term in the definition of T_{ij} and one of the terms resulting from the differentiation of D_{ij}. After these simplifications the resulting equation for the conservation of momentum can be written again as:

$$\frac{\partial u_i}{\partial t}+\frac{\partial\left(u_i u_j\right)}{\partial x_j}=\nu\frac{\partial^2 u_i}{\partial x_j^2}-\frac{1}{\rho}\left(\frac{\partial p}{\partial x_i}-S_i\right) \tag{3.6}$$

The correct solution of equation 3.6 depends on the setting of appropriate boundary and initial conditions, a topic that is further addressed in section 3.3 after analysing the mathematical nature of the equations for the conservation of momentum in x and y directions. The canonical form of a second order partial differential equation (PDE) (3.7) is the

basis for the classification of the governing equations according to the behaviour of the solution in relationship to the boundary conditions. This classification follows standard texts on the subject (see [26, 9]) and consists in judging the value of the discriminant of the equations, determined from the coefficients of the higher order terms that arise when arranging the terms of the momentum conservation equation in the same way as those in equation 3.7.

$$a\frac{\partial^2\phi}{\partial x^2} + b\frac{\partial^2\phi}{\partial x \partial y} + c\frac{\partial^2\phi}{\partial y^2} + d\frac{\partial\phi}{\partial x} + e\frac{\partial\phi}{\partial y} + f\phi + g = 0 \tag{3.7}$$

Discriminant value	Type of Equation
$b^2 - 4ac < 0$	Elliptic
$b^2 - 4ac = 0$	Parabolic
$b^2 - 4ac > 0$	Hyperbolic

Table 3.1: Classification of mathematical behaviour for second order PDE's.

Despite being a widely used technique for the analysis of the behaviour of the PDE's, the method of the "characteristics" has been conceived for an equation with two independent variables, nevertheless, the conservation of momentum represented in equation 3.6 has actually three independent variables, with the third one being the time, t as the model describes a marching problem. Although this discrepancy does not limit the similarity between the behaviour of the governing equation and that of the canonical equation from which the classification is done [9], it is worth to highlight since a single equation can exhibit different behaviours depending on which variables are considered.

One can take for example the equation 3.6 which is seen to have an elliptic behaviour with respect to the spatial coordinates x and y, typical of a steady-state problem and on the other hand, it can exhibit a parabolic behaviour with respect to t, corresponding to one of the time-marching flow problems.

3.2 The FVM inside the Hybrid Framework

The introduction of the hybrid framework in the earlier chapters describes a set of modular components dedicated to solve the flow properties, each one in a specific region of the domain. In the present case the Eulerian solver is designated to solve flows closer to solid walls. In a hybrid method the Eulerian and Lagrangian solvers work together in an cycle that has been outlined by Palha et al. [21] in the following way:

1. Evolve Lagrangian solution.
2. Determine BC for Eulerian solver.
3. Evolve Eulerian solution.

4. Correct Lagrangian solution.

In this algorithm the Lagrangian solution is carried with a vortex particle method and is carried in the first step in a domain that extends all the way to the surface of solid bodies, modelled with boundary elements; the Lagrangian solution in the area of influence of the Eulerian domain is corrected as indicated in the last step. Now, the Eulerian solver implemented here with OpenFOAM, is addressed along this work in what concerns to the second step, the determination of the boundary conditions which is presented in the next sections. The Evolution of the Eulerian solution is finally covered by the remainder of the work.

3.3 Boundary Conditions

A significant number of CFD cases related to external incompressible and viscous flow share a common boundary condition selection that provides both stability and accuracy; in fact it is advised as a standard setting in both basic textbooks [26, 9] and code documentation [13]. The configuration that is considered throughout the work is already determined by the baseline Eulerian solver, PisoFoam, which is itself constructed on a collocated mesh arrangement for the velocity field. This formulation makes necessary to have a specification of boundary conditions for both p and $\mathbf{u}$.

In the surroundings of solid walls it is common to set no-slip boundary condition for velocity and a zero gradient boundary condition for the pressure. In flow regions located away from walls a reversed situation occurs, the pressure in that region is set as a fixed gradient value and the velocity as a fixed value; which directly implies that the distance from the body or for this case, disturbance, is so large that the diffusivity in the flow has dissipated any transient feature. The previous configuration is summarized clearly in table 3.2.

A representation of a domain layout for the typical case of an airfoil is included in 3.1, with a far-field boundary condition comprised of *inlet* and *outlet* regions, one of the most popular set of BC in simulation of external flow around solid bodies. Both the *wall* boundary, and the internal boundary, $\partial\Omega_E$, are shown in detail in figure 3.2; this is the boundary region for the Eulerian grid. Again it is necessary to bear in mind that this configuration is specific to the collocated arrangement of the flow variables on which pisoFoam is constructed.

Since the Eulerian grid is clearly smaller in relationship with the airfoil, the values of $\mathbf{u}$ and p are set under different assumptions in comparison with a far-field boundary condition. Two particular aspects draw the attention at this point:

- The flow field at the Eulerian boundary becomes relatively irregular when such boundary is placed closer to the airfoil or any other disturbance in the flow.
- The values of $\mathbf{u}$ on every segment of the Eulerian boundary should be defined with Dirichlet boundary conditions in order to simplify the flux correction associated to the pressure-velocity coupling and also to avoid introducing a new source of error during the information transfer from the Lagrangian solution field.

Boundary	Velocity BC	Pressure BC
Inlet	Dirichlet, (u, v)	Neumann, $\frac{\partial p}{\partial n} = 0$
Outlet	Neumann, $\frac{\partial \mathbf{u}}{\partial n} = 0$	Dirichlet, p
Wall	Dirichlet, $(0, 0)$	Neumann, $\frac{\partial p}{\partial n} = 0$

Table 3.2: Boundary condition specification for external flow with far-field conditions.

Boundary	Velocity BC	Pressure BC
Inlet	Dirichlet, (u, v)	Neumann, $\frac{\partial p}{\partial n} = H \cdot n$
Outlet	Dirichlet, (u, v)	Neumann, $\frac{\partial p}{\partial n} = H \cdot n$
Wall	Dirichlet, $(0, 0)$	Neumann, $\frac{\partial p}{\partial n} = 0$

Table 3.3: Boundary condition specification for external flow.

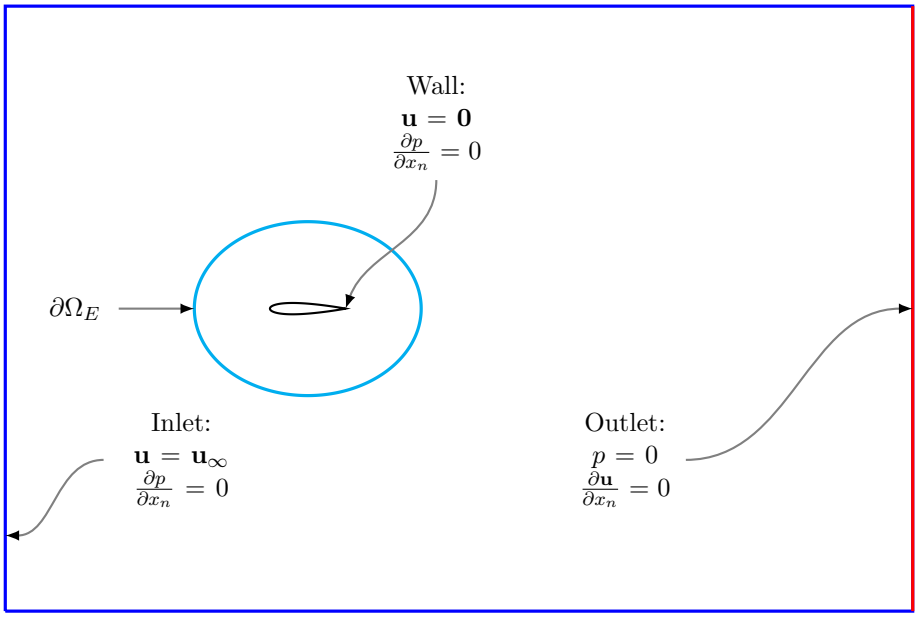


Figure 3.1: Domain and boundary layout for the full Eulerian Domain simulation.

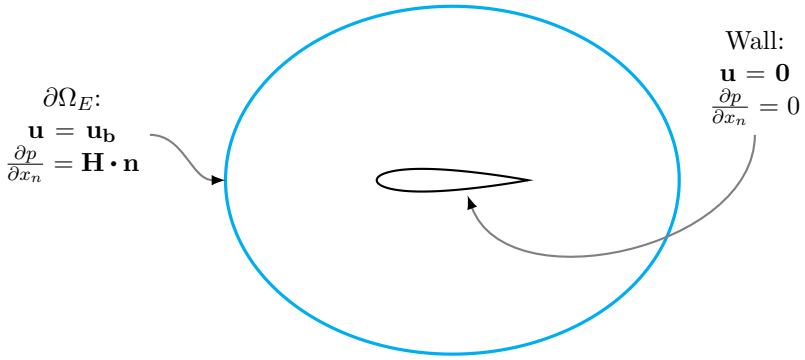


Figure 3.2: Domain and boundary layout for the Eulerian sub-domain simulation.

3.4 Finite Volume Discretisation

The Finite Volume discretisation of the NS equations is based on the use of the Gauss theorem of divergence for the integration of the momentum equation 3.6 over a control volume. The volume integral is then expressed in terms of an associated surface integral over the faces of a volume cell. This section presents then a description of the final result of this integration procedure for each term and a subsequent approximation for the derivatives or values of the transported variables. As will become clear, the spatial discretisation is treated first, followed by the time integration of the momentum equation. The implementation of the boundary conditions is also covered; this gives an integral overview of the process by which the linear system is assembled.

3.4.1 Approximation of surface integrals

A surface integral over a control volume with k faces can be defined as the summation of the integrals over the individual faces:

$$\int_S f \, ds = \sum_k \int_{S_k} f \, dS \tag{3.8}$$

Where the integrand f represents the face-normal flux vector which can be either convective, $f = \rho\phi\mathbf{U}\cdot\mathbf{n}$ or, diffusive, $f = \Gamma\nabla\phi\cdot\mathbf{n}$. Since this integral is performed on the surface of the control volume, the integral must be evaluated everywhere at the cell faces. The availability of this value depends on the variable arrangement on the mesh. For the case of OpenFOAM the collocated mesh provides the solved values directly at the cell centres, rather than at the faces.

In the most simple of the cases the integral, which equals the product of the area of the face and the mean of the value being integrated, can be approximated as the product of the area and the face centre value, f_k this is:

$$\int_{S_k} f \, dS = \overline{f}_k S_k \approx f_k S_k \tag{3.9}$$

According to Ferziger and Peric [9], this approximation provides second order accuracy and it can only be maintained if the method for computing the face-centre flux, f_k, is at least second order accurate as well.

3.4.2 Approximation of volume integrals

Volume integrals are computed via a second order accurate approximation as well, which is based on the approximation of the integral as a product between the volume of the cell and the value of the variable at the cell centre node:

$$Q_p = \int_\Omega q \, dV = \overline{q}\Delta\Omega \approx q_p\Delta\Omega \tag{3.10}$$

3.4.3 Approximation of derivatives

The approximation of derivatives have a fundamental role in the implementation of a Finite Volume method; solution properties such as the accuracy and convergence behaviour are directly influenced by the choices regarding this particular aspect. With the most elementary example being illustrated with the help of a familiar technique known as the Finite Difference Method in a one dimensional general case. Among the options for approximating a first order derivative of a function, f, at the location x_i, the use of Taylor series provides with a group of schemes whose simplest form is obtained from a three-point stencil and, identified by the indexes $i-1$, i and $i+1$. The variation of a flow property in space, is depicted by figure 3.3 along with the graphical representation of the exact first derivative at the point (x_i, f_i) and the approximations according to the different schemes.

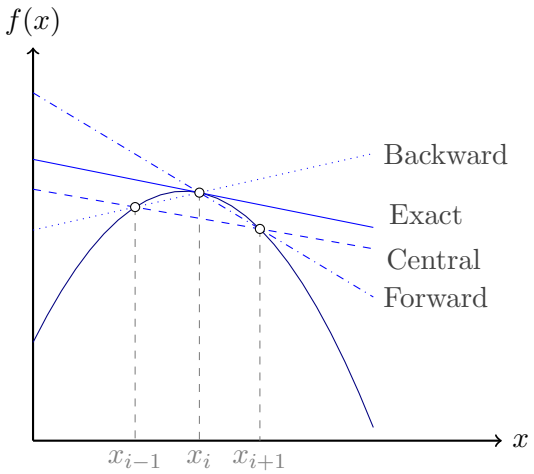


Figure 3.3: Basic differentiation schemes for Finite Volume spatial discretization.

The backward differencing scheme consists on a linear approximation to the first derivative in terms of the value of the flow property at the points i and $i-1$:

$$\left(\frac{\partial f}{\partial x}\right)_i = \frac{f_{i-1} - f_i}{x_{i-1} - x_i} + b \tag{3.11}$$

With b representing the higher order terms, originated from the Taylor series representation of f_i. Now the second scheme to be introduced is the Forward Differencing scheme, in which in a similar way, the derivative is approximated by taking an adjacent point, this time being x_{i+1}, which results in the following approximation:

$$\left(\frac{\partial f}{\partial x}\right)_i = \frac{f_{i+1} - f_i}{x_{i+1} - x_i} + b \tag{3.12}$$

The third possible way for the simple case shown here, is known as the centred differentiation scheme, in which both of the adjacent points to (x_i, f_i) are taken to construct the Taylor series expansion and finally to approximate the value of the derivative.

$$\left(\frac{\partial f}{\partial x}\right)_i = \frac{f_{i+1} - f_{i-i}}{x_{i+1} - x_{i-1}} + b \tag{3.13}$$

The truncation of the Taylor series expansions for each one of the above equations, gives rise to the approximations for the Backward, Forward and Centred schemes.

$$\left(\frac{\partial f}{\partial x}\right)_i \approx \frac{f_{i+1} - f_i}{x_{i+1} - x_i} \tag{3.14}$$

$$\left(\frac{\partial f}{\partial x}\right)_i \approx \frac{f_{i-1} - f_i}{x_{i-1} - x_i} \tag{3.15}$$

$$\left(\frac{\partial f}{\partial x}\right)_i \approx \frac{f_{i+1} - f_{i-i}}{x_{i+1} - x_{i-1}} \tag{3.16}$$

With this approximations a truncation error is introduced and its magnitude and behaviour depends on the mesh size primarily. The behaviour of this error for each particular scheme will determine its rate of convergence and will affect the rate of convergence of the overall method.

3.4.4 Scheme specification in the Eulerian Solver

A dictionary for the specification of the discretization schemes is defined with the entries for the different kind of terms and for the interpolation of quantities and flux-related computations. This list of schemes has been specified as required in the OpenFOAM dictionary format, which constitutes the file `fvSchemes`, as is done with the remaining parameters.

- Time scheme `Euler`
- Gradient Schemes `Gauss linear` (Default), `Gauss linear` (Pressure gradient)
- Divergence scheme `Gauss linear`
- Laplacian scheme `Gauss linear`
- Interpolation scheme `linear`
- Surface-normal gradient schemes `orthogonal`

Time schemes

The transient flow phenomena considered here are accounted for with three possible temporal discretisation schemes, which have been selected for discussion since these are the main choices in OpenFOAM. These time schemes are all implicit and bounded. The scheme denominated *Euler* is first order accurate in time; the *Euler Backward* and *Crank-Nicholson* schemes are instead second order accurate in time.

When the flow field is solved in time, the momentum equations are integrated in time between the instants m and $m+1$, with $t = m\Delta t$ and with Δt representing the time-step size. A look into equation 3.17, which accounts for any scalar ϕ and also for varying ρ, shows the approach for the time discretisation over a momentum equation that has been integrated in a C.V., following the general approach of [17].

$$\int_t^{t+\Delta t} \left[\frac{\partial}{\partial t} \int_{V_P} \rho\phi dV + \int_{V_P} \nabla \cdot (\rho u \phi) dV - \int_{V_P} \nabla \cdot (\rho \Gamma_\phi \nabla \phi) dV \right] dt = \int_t^{t+\Delta t} \left[\int_{V_P} S_\phi(\phi) dV \right] dt \tag{3.17}$$

Once all the terms have been integrated over the C.V, the procedure results the semi-discrete equation in 3.18. Here that the time derivative of the transient term exchanges order with the volume integral. The relevance of arranging the semi-discrete momentum equation in the way it has been done in equation 3.18 lies in the physical meaning on both sides. The right member contains the transient term whereas the left member, with the rest of the terms, contains all of the spatial discretisation terms. From this point the temporal discretisation requires an approach for approximating the time integration of the spatial terms and an approach for approximating the time derivative in the right member.

$$\int_t^{t+\Delta t} \left(\frac{\partial \rho\phi}{\partial t} \right)_P V_P dt = \int_t^{t+\Delta t} \left[\sum_f F\phi_f - \sum_f (\rho\Gamma_\phi)_f \mathbf{S} \cdot (\nabla\phi)_f + SuV_P + SpV_P\phi_P \right] dt \tag{3.18}$$

Under this purpose, equation 3.18 can be expressed in a simpler way, gathering all the terms with spatial operators into one single term.

$$\int_t^{t+\Delta t} \left(\frac{\partial \rho\phi}{\partial t}\right)_P V_P dt = \int_t^{t+\Delta t} A^*\phi \; dt \tag{3.19}$$

The transient term can be treated via the fundamental theorem of calculus, and considering that the cell volume, V_P, remains constant in time, the term can be expressed as:

$$\int_t^{t+\Delta t} \left(\frac{\partial \rho\phi}{\partial t}\right) dt = (\rho\phi)_P^{t+\Delta t} - (\rho\phi)_P^t \tag{3.20}$$

This is equivalent to the Euler Implicit approximation in which the derivative is approximated with a scheme analogous to to a forward finite difference scheme:

$$\frac{\partial}{\partial t}\int_V \rho\phi \; dV \approx \frac{(\rho_P\phi_P V)_{t+\Delta t} - (\rho_P\phi_P V)_t}{\Delta t} \tag{3.21}$$

This scheme employs information from both the solved instant t and the unknown time instant $t + \Delta t$ and provides a first order accuracy in time. Alternatively a higher order scheme can be used for the approximation, using now three different time levels, as the solution at the time instant $t - \Delta t$ is included in the stencil for temporal discretisation, providing a second order accurate solution in time.

$$\frac{\partial}{\partial t}\int_V \rho\phi \; dV \approx \frac{3\,(\rho_P\phi_P V)_{t+\Delta t} - 4\,(\rho_P\phi_P V)_t + (\rho_P\phi_P V)_{t-\Delta t}}{2\Delta t} \tag{3.22}$$

Observing now the terms with spatial operators, on the right hand-side of equation 3.18, a suitable discretisation for the time integrals is developed. Assuming that the integrand remains constant during the time interval [17], the approximation of the whole integral can be performed in terms of the integrand evaluated at one particular time instant that is determined by the discretisation scheme being used. Such schemes include for this project three basic options, which are part of the implementation in OpenFOAM and are present in most of the basic texts dedicated to Finite Volume theory.

The first of these schemes is the *Euler Explicit* method in which the terms with spatial operators are discretised explicitly, which means they are computed from the known solution state at time t. The computation of the solution at time $t + \Delta t$ is carried as:

$$\int_t^{t+\Delta t} A^*\phi dt = A^*\phi^t \Delta t \tag{3.23}$$

When the Euler Implicit scheme is used, careful attention must be given to the *Courant* number, Co, defined in terms of the reference velocity $\mathbf{U_f}$, the minimum mesh spacing $\mathbf{d}$ and Δt, as:

$$Co = \frac{\mathbf{U_f} \cdot \mathbf{d}}{|\mathbf{d}|^2 \Delta t} \tag{3.24}$$

According to the Courant-Friedrich-Levy (CFL) condition, the solution becomes unstable when $Co > 1$, a situation in which, during the lapse of one time step the actual flow propagates by a distance greater than the characteristic length of one cell. When geometric discretisation or the magnitude of Δt ensure a stable solution, the explicit Euler scheme should provide first order accuracy.

Another method for approximating the time integral makes use of the solution values at $t + \Delta t$ as can be observed in equation 3.25. This approximation corresponds to the *Euler Implicit* scheme which has first order accuracy in time.

$$\int_t^{t+\Delta t} A^* \phi dt = A^* \phi^{t+\Delta t} \Delta t \tag{3.25}$$

For an improved solution accuracy the *Crank-Nicholson* scheme (see equation 3.26) can be employed. This method proposes the discretisation as trapezoidal rule in which the values of the integrand are taken as the mean of both the current and new time instants.

$$\int_t^{t+\Delta t} A^* \phi dt = A^* \frac{1}{2} \left(\phi^{t+\Delta t} + \phi^t \right) \Delta t \tag{3.26}$$

Finally another second-order integration scheme with second-order accuracy in time is introduced, complementing an already complete list of basic one-step methods for the time integration of a transport equation of momentum or any arbitrary scalar. In this last scheme denominated *Backward Euler*, three time levels are involved via the approximation of the time derivative with second order accuracy from equation 3.22. The discretisation of the terms with spatial operators alone remains as done for the Euler Implicit scheme, this is, with the solution values of ϕ at the new or unknown time instant.

Note that the segregated solution of a coupled pressure-Velocity system implies explicit terms inside the equations being solved, therefore guaranteeing that the CFL condition is met is still necessary, even though the time-scheme being implemented is implicit on its own.

Gradient schemes

The gradient terms are evaluated explicitly with three alternatives; the first one selected here is the application of Gauss theorem over the integral of the gradient of interest on a finite volume. The other two options evaluate the gradient by using a least-squares

method or by using the surface normal gradient scheme. The gradients are then obtained as:

$$\int_V \nabla\phi \; dV = \int_S d\mathbf{S} \; \phi = \sum_f \mathbf{S}_f \phi_f \tag{3.27}$$

Divergence schemes

The implementation of the Eulerian solver PisoFoam based on OpenFOAM [12] has a distcretisation scheme specifically dedicated to divergence terms different from the convective terms, in which as can be seen, a divergence operator acts on a dependent variable ϕ being convected by the flow velocity. The generic divergence term is then discretised in an explicit way according to:

$$\int_V \nabla \cdot \phi = \int_S d\mathbf{S} \cdot \phi = \sum_f \mathbf{S}_f \cdot \phi_f \tag{3.28}$$

As might be inferred by this point, the values of ϕ_f can be provided either because they are directly available, or via interpolation from cell centre values in case these are the only available source of information. Both cases are accounted for in the "finite volume calculus" set of functions, compiled in the statically-linked library object `finiteVolume`.

Laplacian schemes

Laplacian term in OpenFoam is discretised first by integration over a finite volume and then by linearisation of the resulting form:

$$\int_V \nabla\cdot(\Gamma\nabla\phi)dV = \int_s d\mathbf{S} \cdot (\Gamma\nabla\phi) = \sum \Gamma_f \mathbf{S_f} \cdot (\nabla\phi)_f \tag{3.29}$$

The discretised form for the Laplacian scheme is explicit when the vector between the cell central nodes, is parallel to the common face between the cells in question. When the cells are not orthogonal, an explicit term is included in which cell centre gradients obtained by central differences are used for the evaluation.

$$\mathbf{S_f} \cdot (\nabla\phi)_f = |S_f| \frac{\phi_N - \phi_P}{|\mathbf{d}|} \tag{3.30}$$

Interpolation schemes

OpenFOAM offers several interpolation schemes for computing values at face centres based on the available fields at cell centres. The relevance of interpolation schemes comes to the foreground for implementations as the present, where a collocated grid arrangement

is a predefined choice. A set of generic schemes comprise a portion of all the available methods; the rest are designed specifically for the discretisation of divergence terms via the Gaussian scheme.

Surface-normal gradient schemes

A simple scheme using the centre values from the cell centre and the neighbouring centre values is used to evaluate the normal surface gradients which in the case of an orthogonal mesh consists in:

$$(\nabla\phi)_f = \frac{\phi_N - \phi_P}{|\mathbf{d}|} \tag{3.31}$$

Similarly as done for the Laplacian discretisation schemes, the computation of surface normal gradients on non-orthogonal meshes is done by adding an explicitly computed term with the interpolated gradients from the cell centres via CDS.

Convective term

The integration and discretisation of the convective term in the model is as follows:

$$\int_V \nabla \cdot (\rho\mathbf{U}\phi)dV = \int_S d\mathbf{S} \cdot (\rho\mathbf{U}\phi) = \sum_f \mathbf{S}_f \cdot (\rho\mathbf{U})_f\phi_f = \sum_f F\phi_f \tag{3.32}$$

Note that the mass normal flux at the face, F, is introduced and becomes an important factor when an upwind differencing scheme is selected. When a central differencing scheme is chosen, the face value of the scalar, ϕ_f, is computed by matching the gradient at the cell face to that of the $P-$ cell centre. Defining $f_x = \overline{fN}/\overline{PN}$ then:

$$\phi_f = f_x\phi_P + (1 - f_x)\phi_N \tag{3.33}$$

Instead, when an upwind scheme is used, the value of ϕ_f is determined in a bounded fashion depending on the direction of the mass flux:

$$\phi_f = \begin{cases} \phi_P & \text{if } F \geq 0 \\ \phi_N & \text{if } F < 0 \end{cases} \tag{3.34}$$

By this, the discretisation scheme is always ensured to be backwards which means that the flow properties propagate in the direction of the flow.

Discretisation of boundary conditions

Boundary conditions are subject of the discretisation procedure because they provide information for the dependent variables at the faces of boundary cells that belong to the boundary of the domain itself. The approximations that have been presented in the first part of this section show how the discretised form of a particular term may require either the value of the dependent variable or the value of its gradient. In that sense, the type of boundary condition in a particular boundary region requires a specific treatment.

In the case of **Dirichlet** boundary conditions the value of the dependent variable, ϕ_b, is specified at the boundary face. In consequence, for every term that requires the value of ϕ_f at one boundary face, the implementation of the boundary condition reduces to simply specifying $\phi_f = \phi_g$. Nevertheless, when the value of the gradient is required instead, as happens to be the case for gradient boundary conditions, the value of the gradient in question must be approximated, as:

$$\mathbf{S}_f \cdot (\nabla\phi)_f = |\mathbf{S}_f| \frac{\phi_b - \phi_P}{|\mathbf{d}|} \tag{3.35}$$

When a **Neumann** boundary condition is defined, the gradient of ϕ_b constitutes the information provided for the cell faces at the boundary. For the very generic case where the gradient is provided as a vector with three Cartesian components, the projection of the specified gradient on the face normal vector for every boundary face is then used as the boundary condition:

$$g_b = \left(\frac{\mathbf{S}}{|\mathbf{S}|} \cdot \nabla\phi \right)_f \tag{3.36}$$

In the situation that the terms require the values of ϕ_b to be specified at the boundary face, instead of the normal gradient, the following interpolation is used:

$$\phi_f = \phi_P + \mathbf{d} \cdot (\nabla\phi) \tag{3.37}$$

When the discrete terms are function of the gradients specified at a boundary face instead, the implementation of the boundary condition reduces to the substitution:

$$\mathbf{S}_f \cdot (\nabla\phi)_f = |\mathbf{S}_f| g_b \tag{3.38}$$

3.5 Pressure-Velocity Coupling

What is perhaps one of the fundamental aspects of the numerical solution of the N-S equations, the solution strategy for the pressure, is now outlined. The apparent need for

an equation to determine p, evident from equations (3.5) and (3.6), is treated with an auxiliary relationship that results from the combination of mass and momentum conservation equations as will be shown next. The relevance of pressure in an incompressible NS system, lies in the fact that this flow variable plays the role of a Lagrange multiplier, allowing for the mass conservation to be maintained. The details of this procedure follow those described in [9] which is based on the PISO algorithm as proposed originally by Issa [16].

3.5.1 Derivation of the Pressure equation

The computation of pressure in OpenFOAM is achieved by constructing a new equation that results from taking the divergence of the momentum equations and remembering the divergence-free nature of incompressible flow, resulting from the mass conservation equation. The application of the divergence over the N-S equations would initially result in:

$$\frac{\partial}{\partial x_i}\left[\frac{\partial u_i}{\partial t}+\frac{\partial\left(u_i u_j\right)}{\partial x_j}\right]=\frac{\partial}{\partial x_i}\left[\nu\frac{\partial^2 u_i}{\partial x_j^2}-\frac{1}{\rho}\left(\frac{\partial p}{\partial x_i}-S_i\right)\right] \tag{3.39}$$

Before proceeding further, the pressure is now normalized by the density as $p^* = p/\rho$ in order to follow the same problem setting as the one used in OpenFOAM, the star superscript is not used outside the context of the present section for practicality. Additionally, no body forces are considered, this allows to drop the symbol for the source term in the force contributions from the momentum equation. Now conservation of mass can be invoked once again to treat the transient term, having in mind that $u_i\left(t, x_i\right)$ is presumed to be continuous and in such case, the sequential order of differentiation in second order derivatives becomes irrelevant. After this, the equation changes to:

$$\frac{\partial}{\partial x_i}\left[\frac{\partial p^*}{\partial x_i}\right]=\frac{\partial}{\partial x_i}\left[-\frac{\partial\left(u_i u_j\right)}{\partial x_j}+\nu\frac{\partial^2 u_i}{\partial x_j^2}\right] \tag{3.40}$$

The convective and diffusive terms in the right hand side of equation (3.40) are grouped in the H_i operator, clearly dependent only on u_i and ν. The result is a Poisson equation for p^* which constitutes the auxiliary relationship for the coupling of velocity and pressure:

$$\frac{\partial^2 p^*}{\partial x_i^2}=\frac{\partial H_i}{\partial x_i} \tag{3.41}$$

The initial model is then casted once again as a group of three coupled equations consisting of the 2-D momentum equation and the Poisson equation for p^*:

$$\begin{aligned} \frac{\partial u}{\partial t} &= H_1 - \frac{\partial p^*}{\partial x} \\ \frac{\partial v}{\partial t} &= H_2 - \frac{\partial p^*}{\partial y} \\ \frac{\partial^2 p^*}{\partial x^2} + \frac{\partial^2 p^*}{\partial y^2} &= \frac{\partial H_1}{\partial x} + \frac{\partial H_2}{\partial y} \end{aligned} \tag{3.42}$$

The coupled nature of the pressure and velocity in the resulting model can be treated by means of two general methods, that are summarized by Jasak [17] in a compact way. First a so called group of *Simultaneous Algorithms* that work solving the equations simultaneously over the entire domain and that provide a convenient computational cost provided that both the size of the mesh and the number of coupled equations is not very large. The second group is formed by methods that adopting a *Segregated Approach*, allow to solve the coupled equations by means of a sequential solution algorithm. One common algorithm from this latter group consists of an implicit solver introduced at the beginning of the section, precisely denominated Pressure-Implicit Split-Operator, PISO, which is present in OpenFOAM's transient solvers icoFoam and pisoFoam and, which will be the centre of attention in the remainder of the section.

3.5.2 Solution algorithm and advancing the solution in time

The PISO solution algorithm consists of three basic steps summarized in the same way as [17]:

1. Momentum predictor.
2. Pressure solution.
3. Explicit velocity corrector.

The PISO algorithm conformed of these three basic steps is shown schematically in figure 3.4, with a break down of each element in greater detail according to the illustration of the procedure as done by Ferziger and Perić [9].

Take in the first place the equation of momentum, discretised for a volume with centre P and neighbouring volumes centred in l, as:

$$A_P^{u_i} u_{i,P}^{n+1} + \sum_l A_l^{u_i} u_{i,l}^{n+1} = Q_{u_i}^{n+1} - \left(\frac{\delta p^{n+1}}{\delta x_i} \right)_P \tag{3.43}$$

The implicitness reflected in equation (3.43) becomes clear when observing that both u and p in the resulting Poisson equation are evaluated at the later time instant, $t = t_{n+1}$. Both variables are unknowns in terms of the new time instant, therefore an iterative

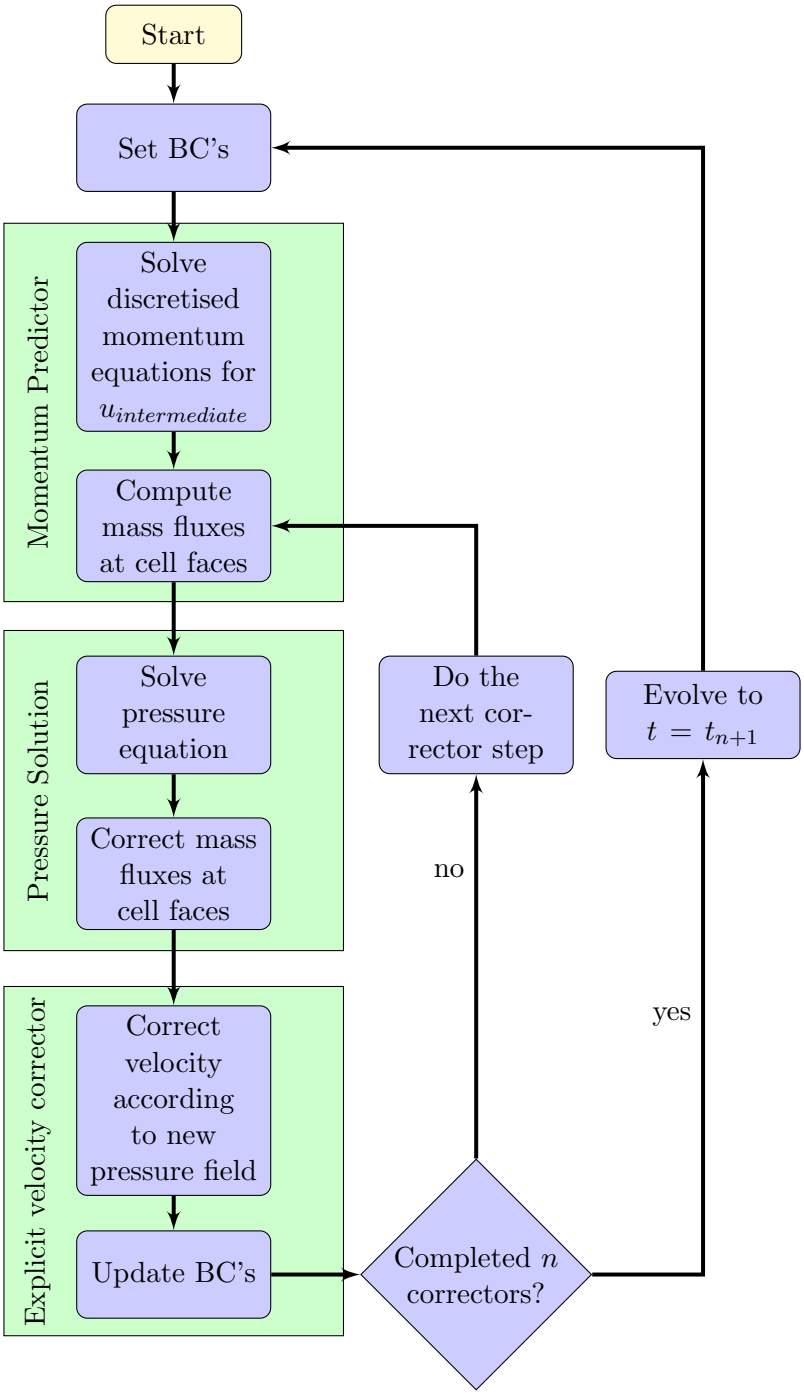


Figure 3.4: Flow diagram for the algorithm behind the PISO approach for $p - V$ coupling.

procedure is required for their computation and this begins by reformulating equation (3.43) into a linear expression:

$$A_P^{u_i} u_{i,P}^{m*} + \sum_l A_l^{u_i} u_{i,l}^{m*} = Q_{u_i}^{m-1} - \left(\frac{\delta p^{m-1}}{\delta x_i}\right)_P \tag{3.44}$$

A detail to observe in the expression above is the solution denoted with the new superscript, m, reached by using a previous estimation for the pressure gradient and the remaining terms represented by $Q_{u_i}^{m-1}$ which can contain terms related to either u_i^n, body forces, the velocity u_i^{m-1} or any other variable at the time t_{n+1}.

Up to this point the solution process can be continued by enforcing continuity or, taking the divergence, over $u_{i,P}^{m^*}$ as solved from equation 3.44. This should result in a Poisson equation from which to obtain an updated pressure estimate p^m. Subsequently a velocity that satisfies the conservation of mass is computed. Since no satisfaction of the momentum equation is guaranteed for the two updated values of u^m and p^m, the iteration process continues further until the momentum equation is satisfied up to an established tolerance, at that moment the estimate u^{m*} and the corresponding value of p are taken as the correct estimations for the time t_{n+1}. The PISO algorithm works in a slightly different way, as the definition of pressure and velocity is done in the form of a correction:

$$u_i^m = u_i^{m*} + u' \tag{3.45}$$

$$p^m = p^{m-1} + p' \tag{3.46}$$

When the definitions for u_i^{m*} and p^{m-1} taken from (3.46) are used in the discretised form of the momentum equation (see equation 3.43) a new relation is obtained:

$$u'_{i,P} = \tilde{u}'_{i,P} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p'}{\delta x_i}\right)_P \tag{3.47}$$

With:

$$\tilde{u}'_{i,P} = -\frac{\sum_l A_l^{u_i} u'_{i,l}}{A_P^{u_i}} \tag{3.48}$$

Now that both the velocity and pressure corrections are related via equation 3.47 the law of continuity is invoked over the definition for velocity from (3.47). After further substitution of u'_i by the expression shown in 3.48, a Poisson equation is obtained:

$$\frac{\delta}{\delta x_i}\left[\frac{1}{A_P^{u_i}}\left(\frac{\delta p'}{\delta x_i}\right)_P\right] = \frac{\delta\left(u_i^{m*}\right)}{\delta x_i} + \frac{\delta(\tilde{u}'_{i,l})}{\delta x_i} \tag{3.49}$$

The unknown velocity corrections open two possibilities at this point, one is continuing the solution process by neglecting $\tilde{u}'_{i,l}$, which leads to a slow convergence process as mentioned by [9], the other option is to consider the approximation:

$$\tilde{u}'_{i,l} = \frac{\sum_l A_l^{u_i} u'_{i,l}}{\sum_l A_l^{u_i}} \tag{3.50}$$

When this approximation is taken, equation (3.49) can be rewritten without the term that involves the velocity correction but, without neglecting it either:

$$\frac{\delta}{\delta x_i}\left[\frac{1}{A_P^{u_i}+\sum_l A_l^{U_i}}\left(\frac{\delta p'}{\delta x_i}\right)_P\right]=\frac{\delta\left(u_i^{m*}\right)}{\delta x_i} \tag{3.51}$$

Inspecting carefully the term in which the pressure gradient is present, the factor involves now the coefficients associated to neighbouring volumes in the denominator as a consequence of the approximation for the velocity corrections. The next step in the solution algorithm is yet another correction for velocity:

$$u''_{i,P}=\tilde{u}'_{i,P}-\frac{1}{A_P^{u_i}}\left(\frac{\delta p''}{\delta x_i}\right)_P \tag{3.52}$$

With:

$$\tilde{u}'_{i,P}=-\frac{\sum_l A_l^{u_i}u'_{i,l}}{A_P^{u_i}} \tag{3.53}$$

Note that u'_i is needed for this last correction; and it can be computed neglecting the first term in equation (3.47) as:

$$u'_{i,P}=-\frac{1}{A_P^{u_i}}\left(\frac{\delta p'}{\delta x_i}\right)_P \tag{3.54}$$

Once again, a divergence free velocity correction derives in the Poisson equation for the associated pressure correction:

$$\frac{\delta}{\delta x_i}\left[\frac{1}{A_P^{u_i}}\left(\frac{\delta p''}{\delta x_i}\right)_P\right]=\frac{\delta\tilde{u}'_{i,P}}{\delta x_i} \tag{3.55}$$

The second correction for the flow velocity that is finally computed according to equation (3.52) is the final result of the solution algorithm known as SIMPLE, used for the prediction of incompressible steady flow. Further velocity corrections can be performed in the very same way for an arbitrary amount of times, being this the differencing factor of the PISO algorithm.

3.6 Solution of linear systems for velocity and pressure

The discretisation of the N-S equations leave as a result a linear system of equations, in incompressible FV problems as the one treated here, one for u and another for p. The model linear system follows the shape presented in equation 3.56; where A is the matrix of coefficients that is assembled from the discrete equation on every cell centre node, ϕ represents in the same sense the vector of the variable being solved, with each element

representing the value at the centre of each cell and finally **Q** representing the vector in which the constant term of every discrete equation is stored.

$$A\phi = \mathbf{Q} \tag{3.56}$$

The discrete set of equations for each node in the domain is reduced then to the system of linear equations in 3.56 that can be solved in a variety of ways and for which, the approach of discretisation determines the shape of A. This is a relevant aspect of the solution of linearised equation systems because the shape of A, in terms of sparsity pattern or algebraic properties, will have an influence on how easily can the system be solved or how accurate or stable will the solution be for a particular solution method. The available solution methods can be put in two basic categories, these are: *Direct* and *Iterative*. In the first case, the system is solved in one single step, what seems not very troubling for a small system of equations but that becomes very expensive in number of operations when a mesh of realistic size is solved. Since the mesh sizes for which the discretisation error is reduced to an acceptable value imply large computation times with direct methods, the use of faster alternatives, which are, iterative methods, has become a predominant approach in CFD. It turns out that iterative methods offer more efficient ways of solving 3.56; since this solution methods are part of OpenFOAM, their basic characteristics will be covered in the discussions to follow.

Direct solution of linear systems

Despite being discarded from the Eulerian solution framework, the direct solution methods can be reviewed briefly so a reference can be set regarding what an undesired solution efficiency is like. Taking as a starting point the extense review in [9] three representative methods for direct solutions can be highlighted. One of them is the Gaussian Elimination algorithm, consisting of two parts, one 'forward elimination' stage and a 'backward substitution' stage; requires a number of operations proportional to $n^3/3$ for a system of equations with n unknowns. The method known as LU factorization is another way of solving the system and is part of one of the iterative techniques as well. The Tridiagonal Matrix Algorithm is in a way a specialized method because it is applied to matrices with three diagonals, a very specific pattern resulting from a specific discretisation scheme and, because the computational work is now scalable with n while resembling to the more general Gaussian elimination.

After covering some characteristics of standard direct solvers, their limitations for applications in problems with a large mesh become evident, therefore the use of *iterative methods* constitutes a very convenient alternative that covers all the solvers that are currently implemented in the Eulerian solver used here. The advantages of these iterative solvers is the improvement of the solution efficiency with relatively less constraints when compared to the direct methods.

Iterative solution of linear systems

In an iterative solution the value of ϕ is approximated by the value ϕ^n a number of times until convergence is reached. The estimation of the solution at an iteration n which is

carefully identified by a superscript with this symbol, satisfies the original equation only to a limited extent, such that a finite residual ρ^n exists as shown in equation 3.57, reflecting the difference between both members of equation 3.56 when the estimated solution is evaluated in equation 3.56.

$$A\phi^n = \mathbf{Q} + \rho^n \tag{3.57}$$

The residual for iteration n is inherent to the iteration error ϵ^n, which is defined as:

$$\epsilon^n = \phi - \phi^n \tag{3.58}$$

And is linked to ρ^n through the coefficient matrix A:

$$A\epsilon^n = \rho^n \tag{3.59}$$

The estimation of the solution for a new iteration $n+1$ is computed by splitting the matrix A into two different matrices, M and N on which the discussion of the chapter will be centred during in the following sections. For now, the iterative estimate is obtained according to:

$$M\phi^{n+1} = N\phi^n + B \tag{3.60}$$

At this point it is important to keep in mind that for the solution to be converged, the residual, ρ^n, must come to be zero, the same as ϵ^n and on the light of equation 3.60 it is not difficult to observe that when finally $\phi^{n+1} = \phi^n = \phi$, the role of the matrix decomposition becomes more clear because under these conditions $A = M - N$ and $B = Q$. Under this methodology the decomposition of A is crucial to the convergence characteristics of the whole solver; it has been shown that the higher is the similarity between M and A the faster the iterative solution converges. A very illustrative analysis is done in [9], precisely by showing how the iteration matrix, $M^{-1}N$, determines how fast the iteration error is reduced from iteration n to iteration $n+1$.

For an iterative solution method to be successful under the mentioned criteria, two conditions must be met during the application, these are:

- The system represented by equation 3.56 should be relatively simple to solve, which implicitly means that both A and N should be sparse matrices and the amount $N\phi^n$ should be easy to compute.
- The iterative estimate ϕ^n should converge fast enough to ϕ; for which M should be easily inverted, and its sparsity pattern should be diagonal, tridiagonal triangular, block tridiagonal or block triangular.

Some of the basic iterative methods and their characteristics are:

- Jacobi. Iterations proportional to n_x^2 or n_y^2. Higher work than Direct methods.
- Gauss-Seidel. Convergence is twice as fast than Jacobi but the improvement is not so convenient after all.
- Successive Over-Relaxation, SOR. When the key parameter of the method, ω, is optimized, the convergence of the solution can be reached after a number of iterations proportional to n_{x_i}.

On top of the advantages of this basic solvers, there are advanced methods that allow for an improved solution efficiency and robustness, these are:

- Incomplete LU decomposition and Stone's method.
- ADI and other splitting methods. Alternating Direction Implicit Method, methods for the solution of elliptic problems. Such elliptic problems, arise for example when a Laplace equation is added a transient term before being solved as a steady-state problem. The particular discretization of this type of equation results in tridiagonal matrix systems, this information is presented from Ferziger and Perić [9] who also point out that this method is particularly well suited for CFD applications were not only pressure correction equations but also equations involving convection can be treated successfully. The overall method efficiency can be such that the cost of the solution can be proportional to as low as n_{x_i} and even $\sqrt{n_{x_i}}$.
- Congugate gradient methods. An interesting picture of this category of solvers is outlined by Ferziger and Perić [9] by classifying two other groups of solvers that are often used together as part of a conjugate gradient method. These two types of methods are the so called, 'Newton-like' methods and the 'general' methods, with many of the former kind consisting in general terms of an optimization problem. For the conjugate gradient method a minimization of the function in question takes place by searching for minima over multiple search directions at the same time. Under favourable circumstances, which are: a non-singular coefficient matrix and the absence of round-off errors, the number of iterations until the convergence of the method would be equal, instead of proportional, to the size of the matrix.

  The rate of convergence of this kind of methods is dependent on the *conditioning* number of A, which can be described as its degree of singularity and which is defined as the ratio between the maximum and the minimum Eigenvalues of A: $\kappa = \lambda_{max}/\lambda_{min}$. Since a matrix with a bad condition number will be more difficult to invert, the solution method might turn time consuming instead of providing any advantage. Systems with unfavourable condition numbers are not the exemption in CFD problems, so in order to take advantage of conjugate gradient methods, an improvement over the algebraic properties of the solution matrix is taken. The procedure of *preconditioning* consists of improving the method by solving the problem for a matrix with a more convenient condition number than the original. This is off course achieved by modifying the problem in a way that preserves the symmetry of the original matrix.

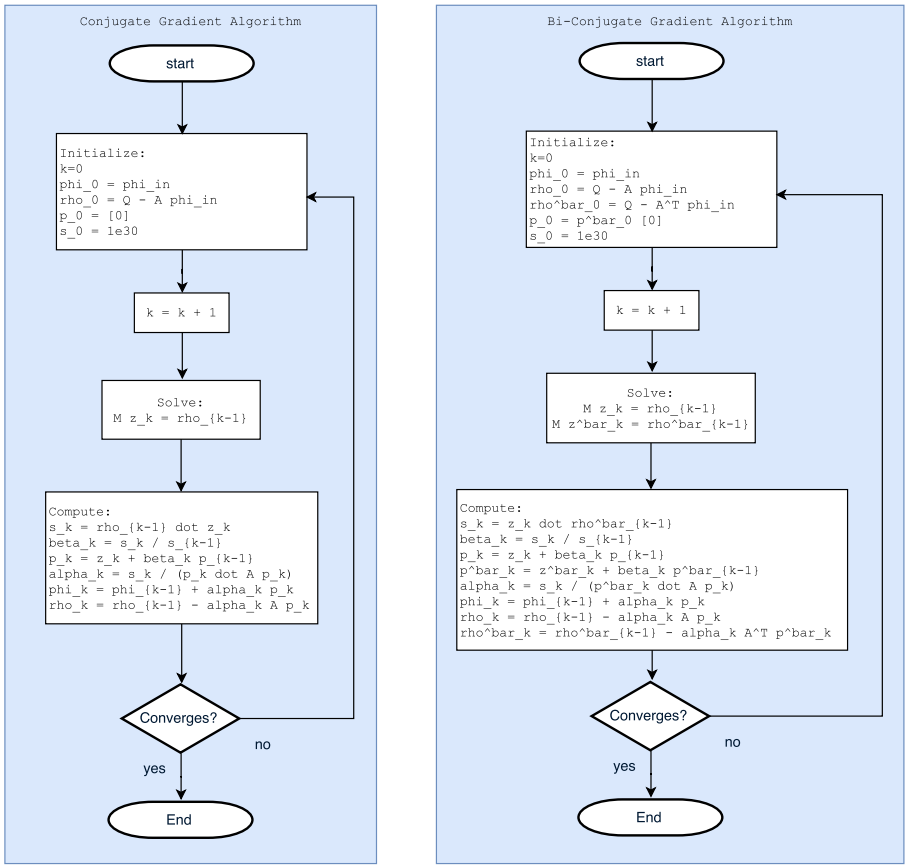


Figure 3.5: Detailed algorithms for conjugate gradient methods.

$$C^{-1}AC^{-1}C\phi = C^{-1}Q \tag{3.61}$$

A new matrix $C^{-1}AC^{-1}$, is considered as the system to be solved; furthermore, the matrix responsible of improving the conditioning number of the problem is C. In the algorithm for solution, the preconditioning matrix is obtained indirectly and its inverse, C^{-1}, is computed instead for direct use. This last task can be done by using incomplete Cholesky factorization of A, the original matrix.

So far the conjugate gradient method has a restricted range of applicability because it requires the coefficient matrix to be symmetric, a condition that is not impossible but that does not happen for all the equations that are commonly solved in CFD. The Poisson equation is a good example of an equation that yields a symmetric matrix after discretisation and this is the case for the pressure correction approach as noted by [9], an equation that has already been explored in section 3.5.

A modified version of the conjugate gradient method allows to use the same solution concept in linear systems with non-symmetric matrices. This modification receives the name of *Bi-conjugate Gradients* and consists on using the Pre-Conditioned Conjugate Gradient method on a symmetric system that is artificially assembled from

the coefficient matrix and its transpose in the following way:

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \cdot \begin{bmatrix} \psi \\ \phi \end{bmatrix} = \begin{bmatrix} 0 \\ Q \end{bmatrix} \tag{3.62}$$

As a remark on the previous two solution methods, it must be kept in mind that both give converged solutions after a similar number of iterations, however the Bi-Conjugate Gradient method is more computationally demanding than the simpler Conjugate Gradient method.

- Multi-grid methods.

  The concept behind this method lies in the treatment of two different meshes allowing to use a mesh with coarser cells to iterate a solution estimate that will in principle yield a good approximation for a subsequent solution in the finer, original mesh. The solution procedure in the coarse mesh has two special characteristics, in the first place, it reduces considerably the computational consumption, in second place it provides a smooth estimation of the error.

  Since the solution on the coarser mesh can be gradually simplified by reducing the number of cells, via an operation of clustering or agglomeration, the saving in computational resource for this stage can be extended up to a point in which the mesh is no longer a determining factor of the work scaling. This important saving makes possible to obtain a work scaling proportional to the number of cells when the multi-grid method is assessed as a whole, which is the sequential solution on both the coarse and fine mesh levels.

Chapter 4

Vortex Particle Method

An introduction to the basic concepts of vortex based models and the key elements for using vortex blobs in the modelling of flow field interactions is now presented. The chapter is comprised of an introduction to a different formulation of the Governing equations discussed along the chapter on Finite Volume Methods and the strategy used to model different flow phenomena. Subsequently the discrete forms of vorticity and velocity fields are introduced after giving a look to the role of the Biot-Savart Law and the Gaussian distribution in the constitution of the model.

4.1 Vorticity-Velocity formulation of the Navier-Stokes equations

One of the main pillars in a vortex particle method is the set of Navier-Stokes equations as they describe the convective and diffusive mechanisms governing the flow of fluid, pretty much in the same way as was covered in the FVM chapter. In the context of vortex methods however the Velocity-Vorticity formulation is considered, as the the vorticity, ω defined in equation 4.2 plays a key role in the definition of the Lagrangian flow field; for example it is central to the auxiliary Poisson equation from which the velocity field is constructed when mesh based solutions are considered.

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla)\,\omega - \nu \nabla^2 \omega = 0 \tag{4.1}$$

$$\omega = \nabla \times \mathbf{u} \tag{4.2}$$

$$\nabla^2 \mathbf{u} = -\nabla \times \omega \tag{4.3}$$

The boundary conditions for this problem lie in the assumption of a uniform velocity field far enough from the perturbation: $\mathbf{u}(x_b, t) \to \mathbf{U}_\infty$ and in consequence a decay of vorticity to zero: $\omega(x_b, t) \to 0$. When solid boundaries exist, boundary elements are required to model its influence on the flow field; on the other hand, when no solid walls are present, all of the vorticity in the flow field must be directly represented. Furthermore, the context in which the equations are solved allow to make an additional simplification, as the segregation of the vortex particle model to flow regions with a relatively small molecular diffusion, allow to attribute a predominantly convective nature. It is by splitting the evolution of the flow into two successive stages that the convective and diffusive phenomena are handled. The aim is now to satisfy the convective term of equation 4.1 in the first step and subsequently, the diffusive term of the momentum equation in the second step. Both stages are shown in detail in table 4.1 which reflects the Eulerian and Lagrangian formulation of the Viscous Splitting algorithm originally proposed by Chorin [3].

Stage	Eulerian Formulation	Lagrangian Formulation
1. Convection	$\frac{\partial \omega}{\partial t} = -\mathbf{u} \cdot \nabla \omega$	$\frac{d\mathbf{x}_p}{dt} = \mathbf{u}(\mathbf{x}_p)$ $\frac{d\omega_p}{dt} = 0$
2. Diffusion	$\frac{\partial \omega}{\partial t} = \nu \nabla^2 \omega$	$\frac{d\mathbf{x}_p}{dt} = 0$ $\frac{d\omega_p}{dt} = \nu \nabla^2 \omega(\mathbf{x}_p)$

Table 4.1: Detail of the viscous splitting strategy in Vortex Particle Methods.

4.2 Biot-Savart Law

A continuous velocity field can be constructed in terms of the convolution between the Biot-Savart kernel, $\mathbf{K}$ and the vorticity field. This corresponds to the Biot-Savart Law:.

$$\mathbf{u}(\mathbf{x}, t) = \int (\nabla \times \mathbf{G})(\mathbf{x} - \mathbf{x}')\omega(\mathbf{x}', t)d\mathbf{x}' = \int \mathbf{K}(\mathbf{x} - \mathbf{x}')\omega(\mathbf{x}', t)d\mathbf{x}' = (\mathbf{K} * \omega)(\mathbf{x}, t) \quad (4.4)$$

Close inspection of equation 4.4, reveals a definition for $\mathbf{K}$ in terms of the Green's Function, $\mathbf{G}$ applied to the Laplacian operator, following the introduction to the model in several works already mentioned in the literature review. For a two dimensional problem as happens to be the case here, $\mathbf{K}$ takes the form shown in equation 4.5, carrying along a singular behaviour when $|\mathbf{x}| \to 0$. This is an undesired situation as the values for velocity can become unbounded depending on the proximity among the particles. The standard solution to this problematic consists of employing a vorticity distribution in order to smooth the field. This is precisely done with a cutoff function in order to arrive to a mollified or regularised kernel, $\mathbf{K}_\sigma$, using again a convolution according to equation

4.6.

$$\mathbf{K} = \frac{1}{2\pi|\mathbf{x}|^2}(-x_2, x_1) \tag{4.5}$$

$$\mathbf{K}_\sigma = \mathbf{K} * \zeta_\sigma \tag{4.6}$$

In the present context as happens to be a common choice, the cutoff function, denominated ζ_σ, is a Gaussian distribution whose shape depends on the parameter σ_i, the core size of the particle or *blob*, and the constant k, which determines extent or scope radius of the cut-off function itself.

$$\zeta_\sigma(\mathbf{x}) = \frac{1}{k\pi\sigma^2}\exp\left(\frac{-|\mathbf{x}|^2}{k\sigma^2}\right) \tag{4.7}$$

Until this point, many of the elements pertaining the discretisation in the vortex particle method have been introduced, besides the parameters σ_i and ζ_i each blob element has a particular vorticity strength or circulation, Γ_i which denotes the amount of vorticity assigned to it.

4.3 Discrete form of vorticity and velocity fields

A discrete form of the vorticity field is built upon the contributions from each element and takes the form of a summation that has the form:

$$\omega(\mathbf{x}, t) \approx \omega^h(\mathbf{x}, t) = \sum_{i=1}^{N} \mathbf{\Gamma}_i(t)\zeta_{\sigma_i}(\mathbf{x} - \mathbf{x}_i(t)) \tag{4.8}$$

Furthermore, the velocity field is constructed in a similar way, making use of the mollified kernel from equation 4.6. The final form of the discrete velocity field is then:

$$\mathbf{u}^h(\mathbf{x}, t) = \sum_{i=1}^{N} \mathbf{\Gamma}_i(t)\mathbf{K}_\sigma(\mathbf{x} - \mathbf{x}_i(t)) \tag{4.9}$$

4.4 Boundary conditions

The interaction of the flow field with solid walls requires a new model to allow predicting the flow field in the near wall region. The approach in the pHyFlow framework works by using a panel method formulation in the surface of the body in question, then a vortex

sheet is considered over its surface and, the associated vorticity distribution applied over the Lagrangian field. From this construct, the no-slip velocity boundary condition is enforced in the Lagrangian component of the Hybrid framework, acting as an estimate since the latter Eulerian component is used in a correction step.

4.5 Initialization and Remeshing

The relative position of the blobs has a relevant effect on the quality of the predicted properties of a flow field, therefore the discretization faces a constraint known as the overlap ratio, $\lambda = h/\sigma$, the ratio between h, the distance between two blobs, and σ, the size of the blob core. The location of each blob at a particular instant is known to be determined by the convective action of the velocity field. Since there is no way to ensure an optimum overlap ratio at every time, the vortex particle method makes use of a remeshing stage; by which, with the help of an interpolation function, the blobs are periodically relocated and their vortex strengths reassigned to match their new position.

4.6 Time Evolution Algorithm

A solution algorithm for the VPM solver module is set by Manickathan's work [19], shown here in figure 4.1. For a given time step a solution is performed by solving for the panels set in the surface of the body, ensuring no-slip condition with respect to the existing flow field. Once a new velocity field is evaluated from the free-stream velocity and the total induced velocities from the vortex blobs and the panels, the blobs are convected for the current time step. Just after convection is evaluated, the final position of each vortex element must be checked in order to avoid breaking the overlapping margin; this is taken care of during the re-meshing stage with the required redistribution of vorticity. The diffusion step takes care of redistributing the vorticity strength, according to the transport of momentum being modelled and completes the solution for a time step.

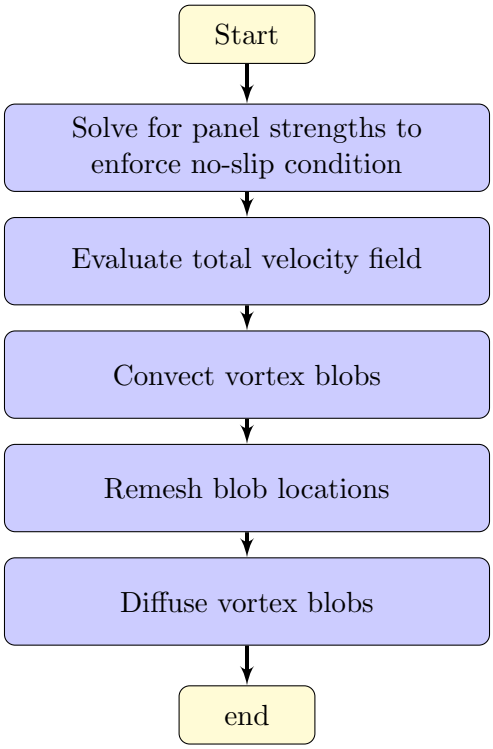


Figure 4.1: Flow diagram for the solution of the Vortex Particle Method.

Chapter 5

Coupling Strategy

The coupling strategy in a hybrid solver handles the most delicate operation of the global solution procedure: interpolating the flow field properties between the Lagrangian and Eulerian domains. Interpolation of flow field data in the context of hybrid vortex particle methods has several roles. One can list for example the correction of Lagrangian solutions in near body regions taking as a source a more accurate Eulerian prediction or the remeshing routines used to deal with the critical aspect of inter particle spacing in VPM. Given that the current project has been scoped to the implementation of a stand-alone FVM solver, under a modular approach so as to ensure the compatibility inside an hybrid framework, the current chapter is dedicated a third and equally essential role for interpolation: using Lagrangian far-field data for determining the boundary conditions in the near-body Eulerian solver.

In order to reproduce the hybrid framework, a far field simulation, predicted also with FVM is considered as the source case, since the integration of an OpenFOAM solver Eulerian component to the Lagrangian VPM solver has not been done in the current work. The direct coupling method is adopted, for setting the boundary condition in the Eulerian portion of the solver, following the work of [8, 21], this means that the determination of the boundary condition for the Eulerian sub-domain occurs directly as opposed to any iterative procedure. Since basic interpolation methods have been considered for this section, the core issue in the determination of the boundary conditions is ensuring the continuity of the flow field in the boundary. Such a requirement is the centre of the implemented solver and is set to test in two flow cases which are introduced in chapter 6.

5.1 Flow adjustment in OpenFOAM

For problems with 'adjustable' boundary conditions in at least one segment of the outer boundary, denominated *boundary patches* in the OpenFOAM context, the solution of the pressure equation includes an adjustment of the boundary face mass flux ϕ, as included

in the `adjustPhi.C` source file, included in the `finiteVolume` library. The construction of the `finiteVolume` library is performed with the `wmake` compiler using the `-libso` option, whereas the construction of the `customPisoFoam` solver is performed using the same compiler with no option specified.

The way in which OpenFOAM corrects flux imbalances is depicted in Figure 5.1, which contains a detailed schematic of the implementation of the function `adjustPhi`; invoked at the moment of solving for the pressure correction in the PISO loop.

One should consider that the natural way of guaranteeing continuity with a PISO-based solver lies in the correction of velocity in boundary patches where the velocity field is specified with a gradient instead of a fixed value. This is why the algorithm in figure 5.1 works with two definitions for the fluxes exiting the boundary or, where $\mathbf{u} \cdot \mathbf{dS} > 0$. One is Φ^{A}_{out}, an adjustable mass outflow and the other is Φ^{F}_{out} which on the contrary is associated to a fixed velocity boundary field.

In a problem where the entire boundary field for velocity is specified with a fixed value, the `adjustPhi` function will have the role of checking continuity in the boundary conditions instead of performing a velocity correction; since no adjustable mass is found, the algorithm outputs an error whenever the relative mass imbalance is higher than 1×10^{-8}.

If the flux can be corrected at any existing patch with adjustable flux, the function returns a boolean value determined by:

$$\frac{||\Phi_{in}||}{\Phi_T} < \epsilon_2 \ \wedge \ \frac{||\Phi^{F}_{out}||}{\Phi_T} < \epsilon_2 \ \wedge \ \frac{||\Phi^{A}_{out}||}{\Phi_T} < \epsilon_2$$

In the OpenFOAM implementation, the values of ϵ_2 and ϵ_1 are finite constants dependent on the precision level with which the solver and the library are compiled.

5.2 Enforcement of continuity at interface

The region enclosing the Eulerian domain is formed by a set of external faces, one for each peripheral control volume. For a collocated mesh arrangement the boundary conditions for $\mathbf{u}$ and p must be determined in the cell centres of all the faces comprised in $\partial\Omega_E$. Determining the required fields at those locations is a task that might introduce an error, specially in the context of hybrid solvers where the information is supposed to move from one sub-grid to the other during the solution algorithm. Such error is usually originated from non-conservative methods for determining the velocity fields at the boundaries and, as consequence, results in a boundary velocity field that violates the continuity law.

Knowing that the enforcement of mass conservation across $\partial\Omega_E$ is crucial for the FVM computations, an artificial correction is proposed, assuming that the error in the boundary mass flux is small in relation to the absolute mass flux permeating $\partial\Omega_E$. It is important to note that in the present context the term 'small' is rather subjective, since the size of the flux imbalance might depend on several aspects of the solution method, specially the kind of interpolation used to transfer the information from one domain to the other. This

flux error is in fact a mass imbalance, denominated here Φ_{net} and defined according to equation 5.1.

$$\Phi_{net} = \int_{\Omega_E} \nabla \cdot \mathbf{u} \; dV = \int_{\partial\Omega_E} \mathbf{u} \cdot \mathbf{n} \; ds \approx \sum_{k=1}^{N_b} (\mathbf{u} \cdot \mathbf{n})_k ds_k \tag{5.1}$$

The approach for correcting the velocity boundary condition in the Eulerian sub-domain began by proposing a modification in the velocity field at the boundary, by an amount on each face such that $\Phi_{net} \approx 0$ at least to machine precision; this is nothing but the continuity law over the boundary condition. The correction that must take place discretely on each local mass flux is outlined in equation 5.2.

$$\Phi_k^{new} = \Phi_k^{old} - \frac{\Phi_{net}}{S_n} |\Phi_k^{old}| \tag{5.2}$$

From equation 5.2 one can observe that the correction is proportional to Φ_{net} normalized by a weighing factor, S_n; this amount, determines how the flux imbalance is redistributed to a discrete level and its definition is an arbitrary choice. A first correction can be assembled according to equation 5.3 which results in a cell-area weighted correction.

$$S_n = \int_{\partial\Omega_E} ds \tag{5.3}$$

A second correction scheme can be set if S_n is defined as a flux-based weight, as specified now in equation 5.4. Once the model for distributing the magnitude of ϕ_{net} on a local level across the boundary. The modification on the velocity field is shown in equation 5.5. This velocity is essentially the correction on u_2, the face-normal component of the velocity vector.

$$S_n = \int_{\partial\Omega_E} |\mathbf{u} \cdot \mathbf{n}| \; ds \tag{5.4}$$

$$u_2' = u_2 - \frac{\int_{\partial\Omega_E} \mathbf{u} \cdot \mathbf{n} \; ds}{\int_{\partial\Omega_E} |\mathbf{u} \cdot \mathbf{n}| \; ds} |u_2| \tag{5.5}$$

The correction term is now the ratio of the local normal velocity to the absolute flux passing across; the improvement with respect to the initial correction makes it a more appropriate way of dealing with the non-conservativeness of the velocities. Works in the same field performed by Zang and Street [30] and Zahle [29] proposed this very same correction. Observing the correction as a way of keeping the proportionality between the global flux imbalance and the local flux correction gives an interesting glance to the concept behind it:

$$\frac{\Phi_{net}}{\Phi_{abs}} = \frac{\int_{\partial\Omega_E} \mathbf{u} \cdot \mathbf{n} \; ds}{\int_{\partial\Omega_E} |\mathbf{u} \cdot \mathbf{n}| \; ds} = \frac{u_2 - u_2'}{|u_2|} \tag{5.6}$$

After comparing both approaches in a simulation of the flow past an aerofoil with a curvilinear boundary with Dirichlet boundary conditions for the velocity it can be observed that the first introduces velocity changes in segments of the boundary where the velocity is predominantly tangential to the boundary surface; it makes sense to assume that in such regions, the normal flux is relatively small and in consequence the velocity correction should have a small effect. The second correction results in velocity fields with better appearance in those areas where the velocity vector is tangential to the boundary surface as no irregularities are evidenced after inspecting the results. It can be said that the area-weighted correction is not wrong but is inappropriate for regions where the velocity is tangential to the boundary face. Only when the velocity permeates a face orthogonally, the correction would yield a fair result. For this reason the second approach is retained in the subsequent part of the project.

5.3 Implementation of a Stand-Alone Eulerian Solver in OpenFOAM

The implementation described here is the central element of the project and consists of an Eulerian solver, `customPisoFoam` built upon the `pisoFoam` solver as a baseline, with its functionality extended by:

- Enabling the formulation of a pure Dirichlet problem for velocity via a non conservative interpolation method with a flux correction for guaranteeing mass conservation across the boundary.
- Enabling the formulation of the FVM with a pure Neumann problem for pressure.

The structure of the baseline solver can be described in terms of the momentum predictor-corrector three-step diagram that has been introduced in figure 3.4, using now the corresponding sections of the `pisoFoam.C` source file, written in OpenFOAM's C++ based high level syntax. Beginning by the momentum predictor as specified in listing 5.1 the linear system matrix for velocity is instantiated separately in the variable `UEqn` so it can be reused in a later step, the first velocity prediction is then carried by solving for `UEqn` equated to the negative of the pressure gradient which is computed explicitly from the available internal field as an initial guess.

Listing 5.1: Code segment for the momentum predictor step.

```
fvVectorMatrix UEqn
(
    fvm::ddt(U) + fvm::div(phi, U)
  + MRF.DDt(U)
  + turbulence->divDevReff(U)
 ==
    fvOptions(U)
);
solve(UEqn == -fvc::grad(p));
```

Once the momentum predictor is executed the solution continues in the pressure solver step as shown in listing 5.2 which begins by defining the auxiliary mass flux `phiHbyA` and velocity `HbyA`, checks for any mass imbalance in the flux field in order to correct it when required and finally solves the Poisson pressure equation instantiated in the `pEqn` linear matrix system.

Listing 5.2: Code segment for the pressure solution step.

```
// Define the mass flux
volScalarField rAU(1.0/UEqn.A());
volVectorField HbyA("HbyA", U);
HbyA = rAU*UEqn.H();

surfaceScalarField phiHbyA
(
    "phiHbyA",
    (fvc::interpolate(HbyA) & mesh.Sf())
  + fvc::interpolate(rAU)*fvc::ddtCorr(U, phi)
);

// Mass flux correction to ensure continuity before solving for p
adjustPhi(phiHbyA, U, p);

while (piso.correctNonOrthogonal())
{
    // Pressure corrector
    fvScalarMatrix pEqn
    (
        fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
    );
    pEqn.setReference(pRefCell, pRefValue);
    pEqn.solve(mesh.solver(p.select(piso.finalInnerIter())));
    if (piso.finalNonOrthogonalIter())
    {
        phi = phiHbyA - pEqn.flux();
    }
}
```

The last step in the PISO algorithm is the explicit velocity corrector which as shown in listing 5.3 consist in the subtraction of the pressure contribution from the guess of the velocity field.

Listing 5.3: Code segment for the velocity corrector step.

```
U = HbyA - rAU*fvc::grad(p);
U.correctBoundaryConditions();
```

Before going into details of the modifications that lead to the customized version of the solver, it is important to note that the mass flux taking part in the pressure solution step is denominated `phiHbyA`, representing a newly defined mass flux in terms of the H operator which just as the one introduced earlier in equation 3.41 represents in the OpenFoam context, the off-diagonal terms involving known velocities and additional terms other than ∇p. the term `rAU` represents the inverted diagonal coefficients of the velocity matrix `UEqn` defined in the momentum predictor; as a consequence, `phiHbyA` and also `HbyA` represent

the mass flux and the velocities respectively, without the force contribution associated to ∇p. This is reflected in the first statement of the velocity corrector, where the effect of pressure is included in order to obtain a divergence-free flow field.

5.3.1 Default method for flux correction in the baseline solver.

When a given velocity field results in a mass flux imbalance, mass continuity is ensured with a correction proportional to the imbalance itself, this operation is already included in the PISO-based OpenFOAM solvers, but requires at least one boundary patch to be specified with a Neumann boundary condition for **u**. The flux correction is carried according to figure 5.1 and, as one can see in the previous description of the baseline solver, it is an indispensable requisite prior to the computation of the pressure gradient that allows for a divergence free velocity field after the corrector step is executed.

In a simulation with either the `pisoFoam` or `icoFoam` solvers, if no boundary patches with Neumann BC for **u** exist, the default implementation of the `adjustPhi` function will halt the computation with an error prompt, as no means of correcting the flux for zero-divergence exists for that scenario.

5.3.2 Flux correction in the custom solver.

The implementation of the `customPisoFoam` solver preserves all of the structure of the baseline solvers but contains two modifications for the boundary velocity field and the mass fluxes, that correct the mass conservation error introduced to the flow field when the boundary conditions are interpolated from an existing solution on the large scale domain. A code section has been included to extract mesh information and the pressure boundary field as shown in listing 5.4. The `customPisoFoam` solver contains a code section introduced in listing 5.5 that modifies the boundary condition for velocity to make it divergence free right after the time loop begins.

Listing 5.4: Determination of boundary conditions from `customPisoFoam` solution for a far-field sized domain.

```
label eulerianPatchID = mesh.boundaryMesh().findPatchID("omegaE");
if(eulerianPatchID != -1)
{
    Info << "Eulerian boundary exists! ..." << endl;

    Create polyPatches for the boundaries:
    const polyPatch& nbPatch= mesh.boundaryMesh()[eulerianPatchID];

    Info<< "\n\nExtracted Face centre coordinates Vectors: \n"<<endl;
    const Field<vector>& fcs  = nbPatch.faceCentres();
    Info << "Face Centres:" << fcs << endl;

    Info<< "\n\nExtracted Face Normal Vectors: \n" << endl;
    const Field<vector>& fns  = nbPatch.faceNormals();
    Info << "Face Normals:" << fns << endl;
    Info << "Pressure BC field: " << p.boundaryField()[eulerianPatchID
        ] << endl;
}
```

Listing 5.5: Correction for $\mathbf{u}$ in the frame of reference of each boundary face.

```
const scalar totalBoundaryFlux = VSMALL + sum(mag(bphi));
const scalar netBoundaryFlux = sum(bphi);

    // Flux correction
forAll(bphi, patchi)
{
    fvsPatchScalarField& phip = bphi[patchi];
    forAll(phip, i)
    {
        phip[i] -= netBoundaryFlux*mag(phip[i])/totalBoundaryFlux;
    }
}

    // Velocity correction at the boundary
Field<scalar> sPhi(fn.size());
Field<scalar> cPhi(fn.size());
for(int k = 0; k<fn.size();k++)
{
    sPhi[k] = -fn[k][0]/mag(fn[k]);
    cPhi[k] = fn[k][1]/mag(fn[k]);
}

float Qnet = 0; float Qtotal = 0; float tmp = 0.0;

for(int k = 0; k < U.boundaryField()[numBound].size(); k++)
{
    tmp = fa[k]&U.boundaryField()[numBound][k];
    Qnet   += tmp;
    Qtotal += mag(tmp);
}

Tensor<scalar> R(1, 0, 0, 0, 1, 0, 0, 0, 1);

// Rotate velocity vectors to the cell-face reference frame
for(int k = 0; k < fn.size(); k++)
{
    R.xx() = cPhi[k];
    R.xy() = sPhi[k];
    R.yx() = -sPhi[k];
    R.yy() = cPhi[k];
    U.boundaryField()[numBound][k] = transform(R,U.boundaryField()[
        numBound][k]);
    U.boundaryField()[numBound][k][1] -= Qnet/Qtotal*mag(U.
        boundaryField()[numBound][k][1]);
    U.boundaryField()[numBound][k] = transform(R.T(),U.boundaryField()
        [numBound][k]);
}
```

The flux correction occurs later, as part of the modified version of the `adjustPhi` function. This flux correction has the purpose of supplying a divergence free flux, computed without the action of the pressure gradient for the subsequent solution of the pressure equation in the PISO algorithm. The implementation of the flux correction just before the solution of the pressure equation allows to ensure a divergence free flow either for a time-fixed boundary condition or for a time-varying boundary condition.

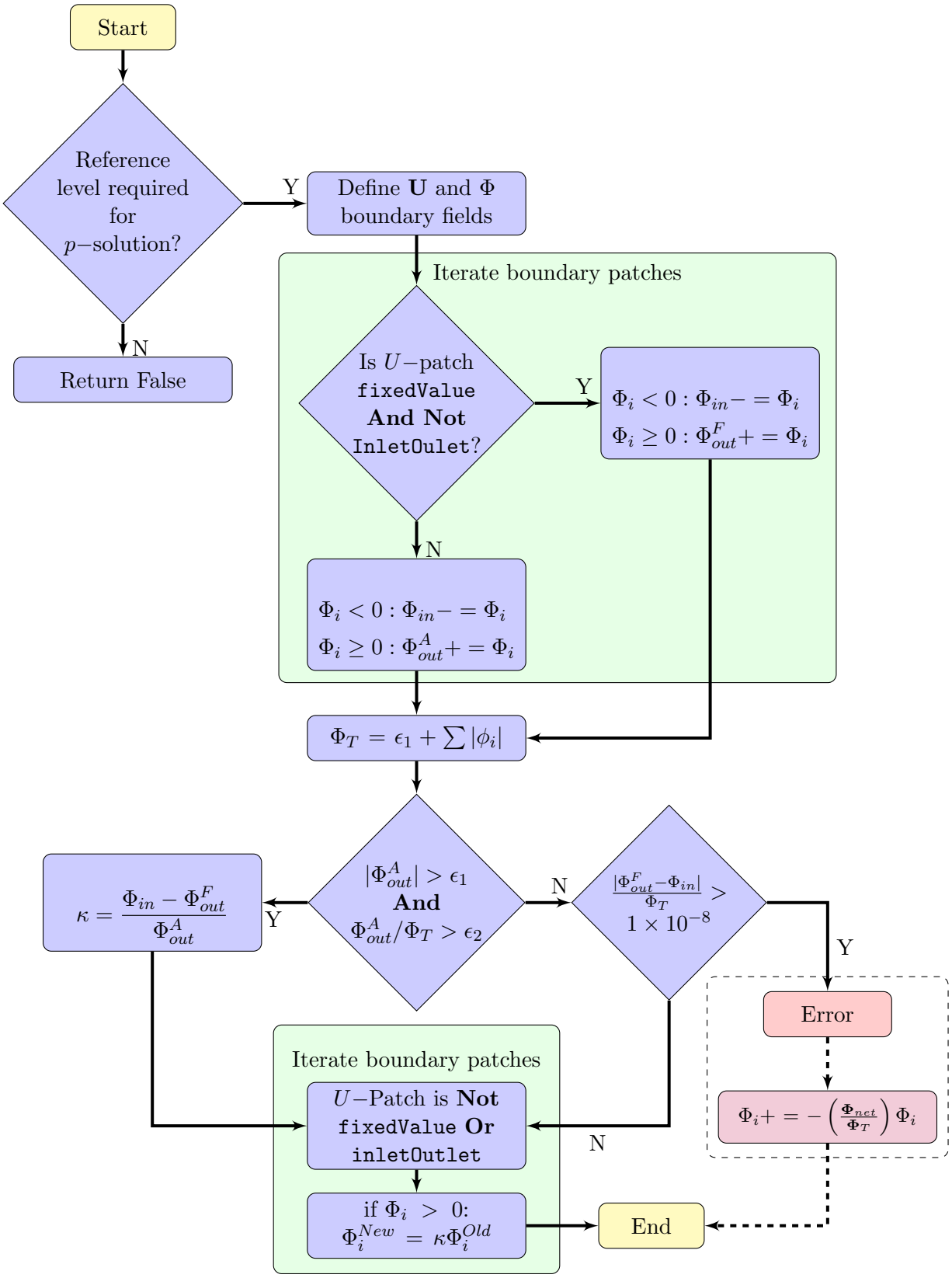


Figure 5.1: Flow chart for the `adjustPhi.C` original source file and the proposed modification enclosed in the dashed, '- -', rectangle.

Chapter 6

Benchmarking Simulation Cases

The simulation cases for the benchmarking of the customized `pisoFoam` solver application in OpenFOAM are presented next. The details of basic solution convergence with respect to the spatial discretisation as well as the boundary condition for pressure are discussed as part of the analysis of the uniform flow induced by a point vortex. The analysis on the proximity between flow and wall boundary conditions is developed in the context of FV simulations of a DU00-W-212 airfoil; this time working on a series of cases where the mesh is set to different sizes with respect to the airfoil chord length.

6.1 Point vortex flow with Dirichlet velocity BC

The modified version of the pisoFoam solver is set to test with the simulation of a 2-D and steady flow field in a squared domain, induced by the action of a point vortex located externally to the domain. The consideration of a point vortex is conceived in this section as a convenient way of testing a flow field with an exact solution for velocity and pressure; this facilitates the comparison with flow field variables as predicted by the Eulerian solver. For a point vortex of strength Γ the induced flow field components are computed as:

$$u = \frac{\Gamma}{2\pi} \frac{y - y'}{(x - x')^2 + (y - y')^2}$$

$$v = -\frac{\Gamma}{2\pi} \frac{x - x'}{(x - x')^2 + (y - y')^2} \tag{6.1}$$

An expression for the pressure field p can be deduced from the Bernoulli equation which for steady flow is a function of the freestream pressure and the velocity magnitude. Assuming now that the reference pressure is $p_\infty = 0$, the resulting equation is:

$$p = -\frac{|\mathbf{u}|^2}{2} \tag{6.2}$$

Since the velocity components are explicitly determined for any point in space, except the location of the vortex itself, both $\mathbf{u}$ and p can be computed everywhere for comparison. The methodology for the analysis of the computations with the Eulerian solver consist of recreating a solution scenario as it would happen in the hybrid framework. With this purpose a velocity field is computed at random locations around the domain as the base flow field for latter computation of the boundary conditions and sampling regions for error analysis. The velocity field is illustrated in figure 6.1; the sampling region and the location of the boundary points are shown as well. The set up of the simulation cases

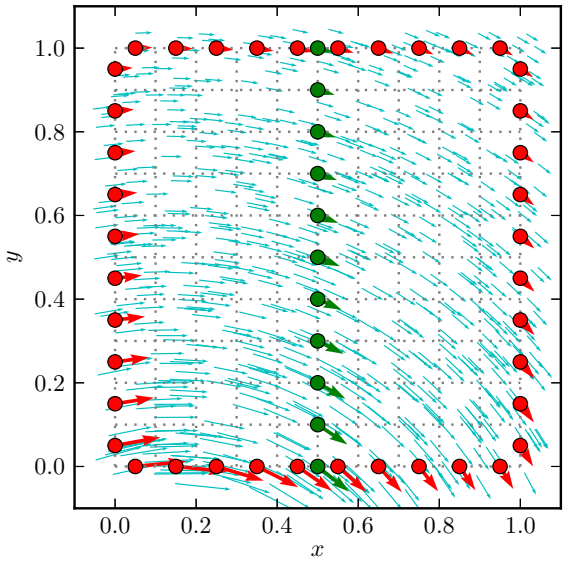


Figure 6.1: Potential flow field of a point vortex at $(x, y) = (0.1, -0.5)$ for $n = 10$.

is done with base in several parameters, with the location of the vortex element among them. A convenient choice is to place the vortex element outside the flow domain because, unlike a mollified vortex, the velocity becomes unbounded in the proximity of the vortex centre for the simple case considered here. Other parameters are:

$$\begin{aligned} \Gamma &= -3 \\ (x', y') &= (0.1, -0.5) \\ L &= 1\ m \\ p/\rho &= 1\ m^2/s^2 \\ \Delta t &= 1 \times 10^{-3} \end{aligned}$$

The simulations are organized in three main groups with each one corresponding to a method for determining the u boundary condition over $\partial\Omega_E$, given a set of values from an imaginary flow field which in the context of the hybrid solver corresponds to Ω_L. Following the approach from the hybrid solver framework, the boundary condition for velocity is set with fixed values in all the four boundaries of the squared domain in figure 6.1, using

an exact computation following equations 6.1 and using three interpolation methods for benchmarking: *nearest neighbour*, *linear* and *cubic*.

The comparison between the different set of boundary conditions is reflected in the convergence analysis of the solution method which is carried out by, taking a sample of the velocity field along the sampling region shown in figure 6.1 and, computing the absolute error with respect to the exact solution for the flow field at the corresponding locations. The computation of the absolute error for a set of four meshes with refinement levels of 10, 60, 100 and 600 cells in x and y directions alike. The error, computed as the $l^{\infty}-$ norm of the absolute error for each grid appears in figure 6.2.

For both u and v velocity components, the slope of the fitted linear polynomials is close to -1. This is verified in the postprocessing and represented graphically in figure 6.2. The most immediate indication of this results is that the spatial discretisation of the Eulerian solver has an order of convergence close to first order and above for the best case. This statement holds valid for the simulations with exact velocity BCs and for linear and cubic interpolated BCs; not only because the slopes of the fitted polynomials have values between 1.1 and 1.3 but because for the most refined cases, where the FV approximation is valid, the variation of the error has a much better tendency. On the other hand, the simulation cases with nearest neighbour interpolated BC's shows a slower convergence in the solution of both velocity components. At this point it is also interesting to note that the behaviour of the error for the v component is not so defined as that of the u component. Additional simulation cases in which the vortex is located in the same y coordinate as the sampling region show that the behaviour of the error in the v component is even more irregular than the u component. Based on this, it makes sense to assume that when the velocities get close to zero, as is the case for the v component in the latter case and, to a lesser degree in the original one, it is difficult to avoid loosing accuracy when the postprocessing requires the data to be handled in multiple platforms, such as paraFoam with the *vtk* format and python, with the latter being the platform in which the postprocessing is ultimately performed.

The discretisation schemes that are specified in the `fvSchemes` dictionary in each simulation are first order accurate, therefore the most important outcome of this exercise, as reflected by figure 6.2, indicate that the determination of the boundary conditions via interpolation from a known velocity field is not an impediment for preserving the order of convergence of the Eulerian solver as long as the interpolation method is accurate enough. In this case a linear and cubic interpolation method are taken as test methods that behave relatively well.

A qualitative analysis of the flow field variables is done by comparing the contours of u, v and p between the exact solution and two sets of numerically predicted fields in which two different specifications for the Neumann boundary condition for the Poisson equation for p are considered. The comparison of the individual velocity components is presented in figure 6.3 whereas the comparison of the density-normalized pressure is presented in figure 6.4. The results from the exact solution are included in both figures in the form of solid-colored contours and the results from numerical solutions are presented as superimposed contour levels with their respective label. In the case of figure 6.4, the pressure is plotted with the offset of the minimum pressure; with the intention of comparing the variation of the field with the same reference.

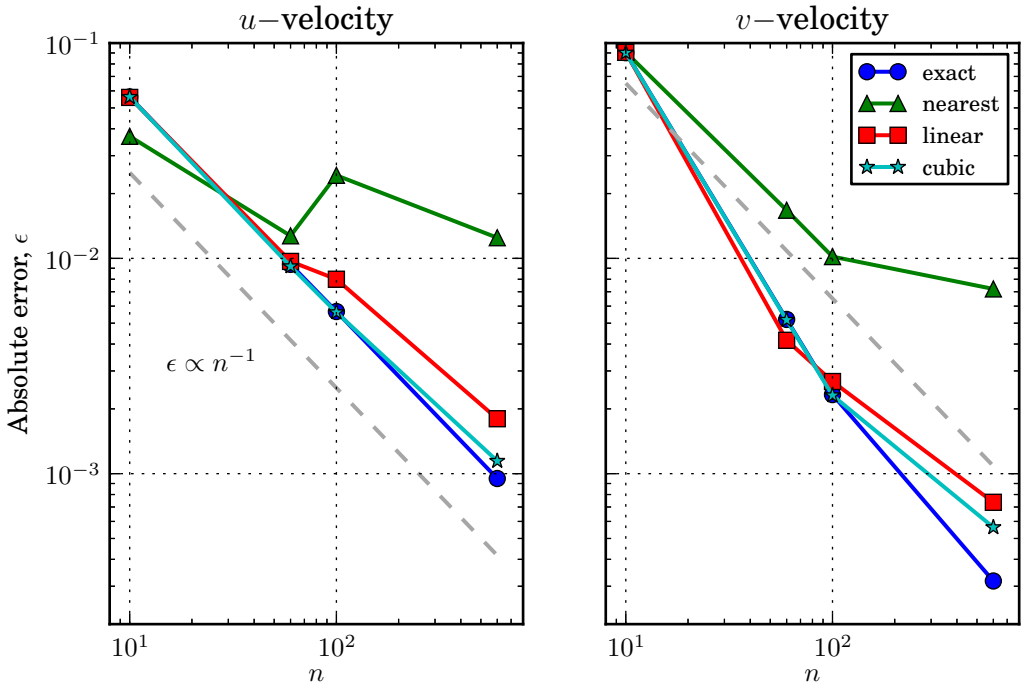
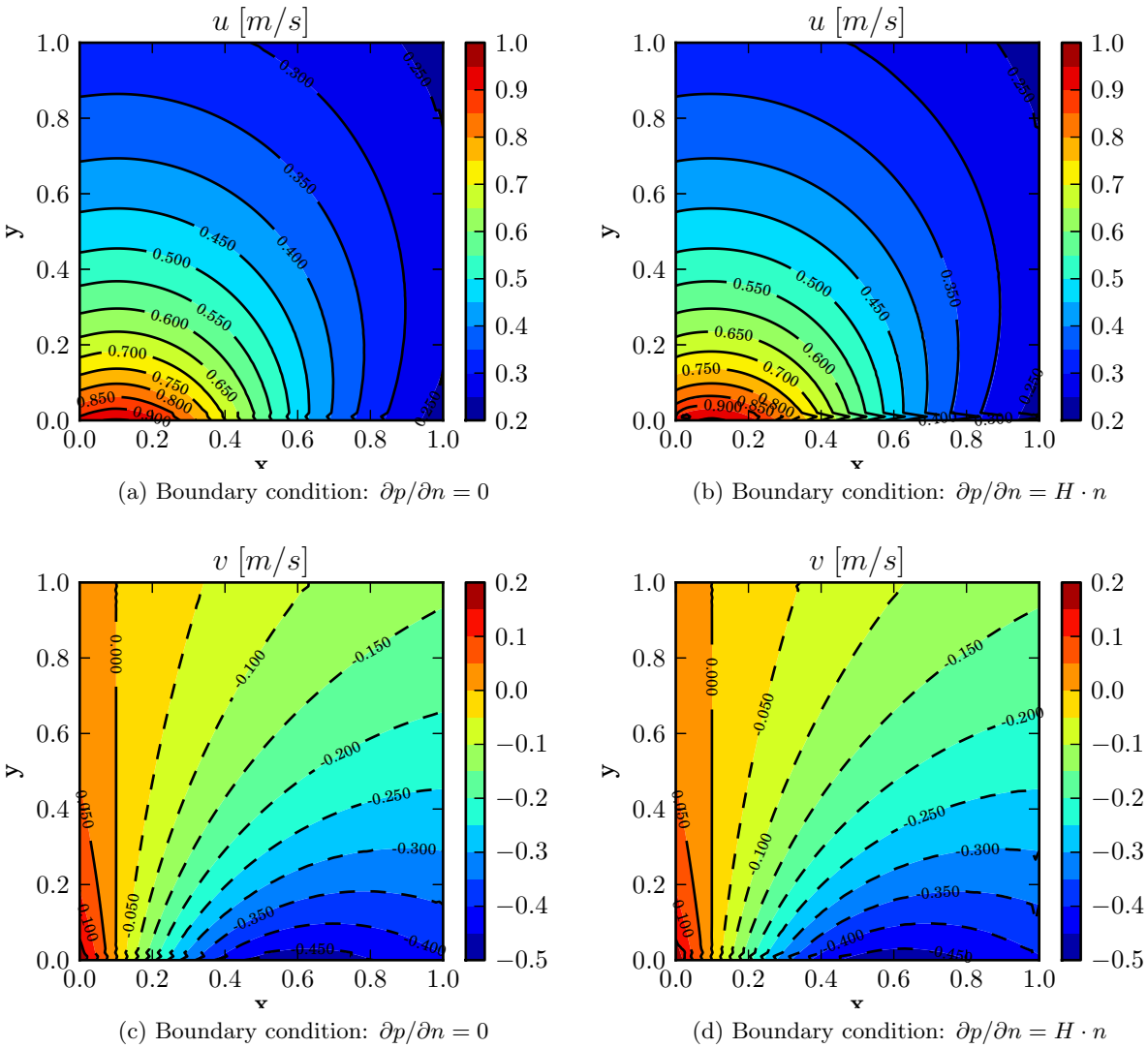


Figure 6.2: Consistency analysis on the FVM spatial discretisation set up.

The results show that the velocity components are predicted in agreement with the exact solution for most of the domain except the regions near the southern and eastern boundaries for figures 6.3 (b) and (d). Besides this aspect, which depends largely on the reconstruction of the pressure values at the boundaries and not on the solution of the pressure equation, there is barely a difference in the p node values between the heterogeneous and homogeneous Neumann BC for p. A different situation occurs for the predicted p field itself, since not only there is a clear difference between exact and numeric solutions everywhere in the domain but there's also a large discrepancy in the south-west cell pressure value; the same cell at which the pressure is set to an arbitrary value as is usually done.

Observing the current situation in perspective, it is important to recall that the heterogeneous Neumann BC for p is available as long as an estimate of the velocities exist; with this, the estimation of the required pressure gradient on a boundary is computed from the momentum equation. Despite having an explicit estimation for the boundary condition, the pressure solution deviates from the exact solution, in contrast to the homogeneous Poisson problem, for which the pressure is predicted in close resemblance to the reference case. A solution to this discrepancy could be a reconstruction of the pressure values across $\partial\Omega_E$ from the node solutions in Ω_E as proposed by Zahle [29].

The solution of the Poisson equation for p is seen to be significantly affected by the boundary condition specification. In the case of $\mathbf{u}$ on the other hand, the solution agrees with the exact velocity field except for the near boundary solution, for both boundary condition specifications for p. The fact that the pressure solution has a constant included due to the nature of the resulting linear system has been considered before comparing



(a) Boundary condition: $\partial p/\partial n = 0$

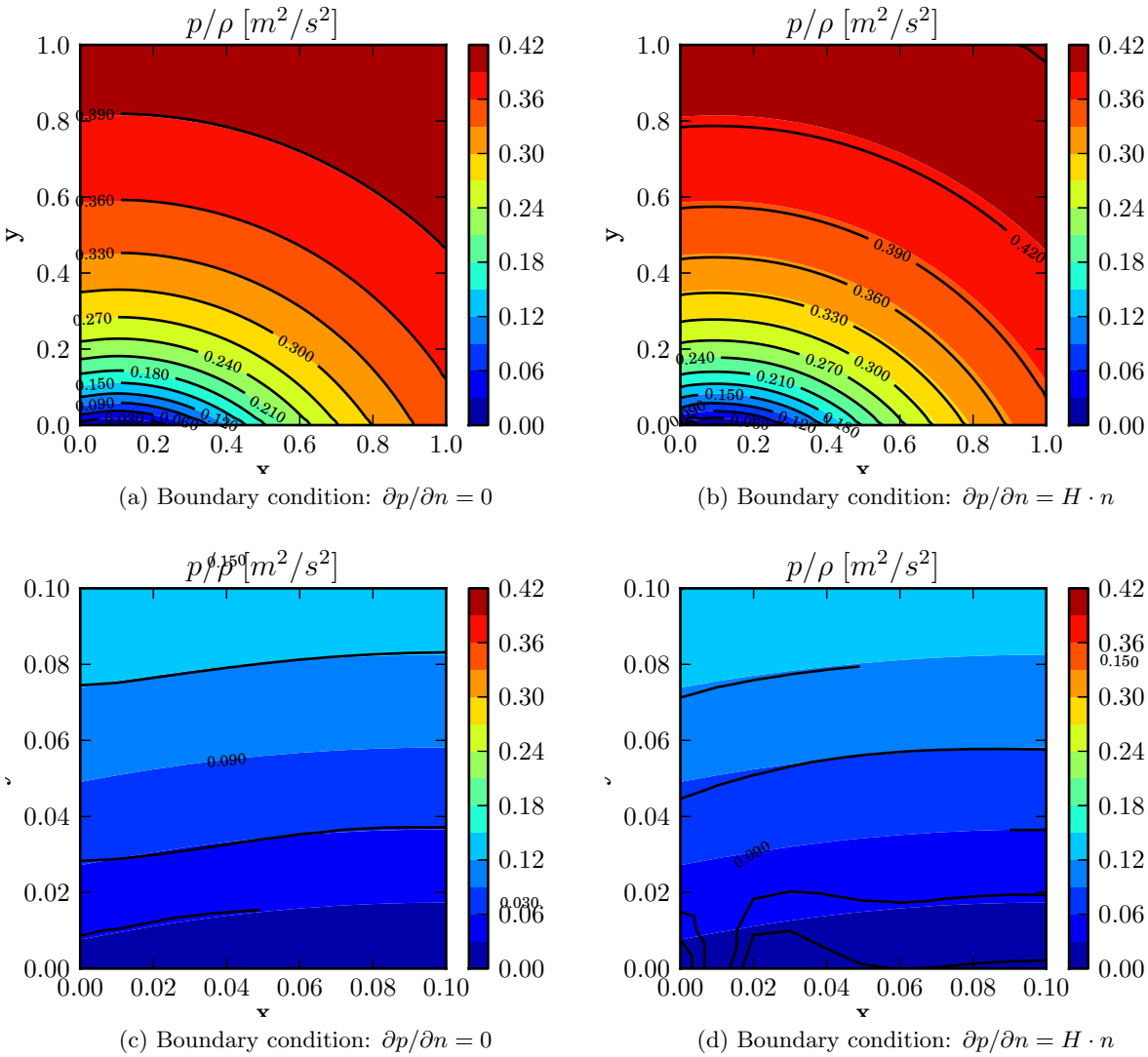
(b) Boundary condition: $\partial p/\partial n = H \cdot n$

(c) Boundary condition: $\partial p/\partial n = 0$

(d) Boundary condition: $\partial p/\partial n = H \cdot n$

Figure 6.3: Contour plots for velocity components. The Filled contours represent the exact solution of the potential field; black contour levels represent the numeric solution.

the pressure fields from the numerical solution to the exact values; in this sense, the boundary condition setting is a key aspect for the prediction of the loads since the pressure distribution over the boundaries will eventually be affected by the field on the whole domain. The absence of viscous transport and the existence of a single region for $\partial\Omega_E$ gives a rather simplistic focus to the present exercise which aims to address aspects inherent to the solver performance instead to the nature of the flow. It is necessary to point that the simulations are performed with a point vortex instead of using the VPM hybrid solver as the OpenFOAM Eulerian solver has been compiled as an individual application. In order to continue with the benchmarking the analysis extends further to the study of flow around an airfoil in analogous conditions to the ones considered in the current section.



(a) Boundary condition: $\partial p/\partial n = 0$

(b) Boundary condition: $\partial p/\partial n = H \cdot n$

(c) Boundary condition: $\partial p/\partial n = 0$

(d) Boundary condition: $\partial p/\partial n = H \cdot n$

Figure 6.4: Contour plots for p with offset of p_{min}. The Filled contours represent the exact solution of the potential field; black contour levels represent the numeric solution.

6.2 Flow around the DU 00-W-212 airfoil

In this section the solution of the N-S equations is performed in a flow domain around an airfoil in a simulation campaign containing five cases denominated 'target' cases and one more simulation denominated the 'source' case. According to the methodology defined for this part of the project the source simulation works as a reference solution from which boundary conditions for all the other 'target' cases, are obtained via linear interpolation; additionally the solution from the source simulation is the reference for comparative analysis of the solutions in each one of the target cases. In this order of ideas it is clear that the role of the source simulation the same as that of a Lagrangian solver, i.e. providing the boundary conditions for the Eulerian simulations in the near-wall sub-domain. On the

other hand, contrary to the ideal hybrid framework, both source and target solutions are obtained from a pure Eulerian solver. The consideration of steady state flow is retained for a clearer assessment of the solution on a domain with two boundary regions which are Ω_E and the airfoil wall, and with a variable domain size on each one of the five target cases.

6.2.1 Simulation approach and parameters

The flow conditions correspond to an intermediate Reynolds around $Re \approx \mathcal{O}(1 \times 10^5)$ which can provide an interesting case for analysis along with the selected angle of attack. Additionally this flow condition allows for comparison with available data such as the work of [1]. In the current simulation campaign the flow is assumed to be fully turbulent, predicted with the Spalart-Allmaras turbulence model and with no wall treatment; expecting a fair level of simplicity that enables to observe the behaviour of the steady state solution without dropping the customized time-varying solver as the centre of the analysis. The linear convection scheme specified for $\mathbf{u}$ in section 3.4 is retained for the transport equation for the kinematic turbulent viscosity, $\tilde{\nu}$. A more extensive review of the main parameters can be listed as:

$$
\begin{aligned}
U_\infty &= 26\ m/s \\
\alpha &= 8^\circ \\
c &= 1\ m \\
\Delta t &= 1 \times 10^{-5} \\
Re &= 130000
\end{aligned}
$$

The size of the Eulerian domain, Ω_E, in each target simulation case is quantified by δ which represents the distance between the airfoil 'wall' boundary and the outer flow boundary, $\partial\Omega_E$, in a direction normal to the airfoil surface. A detail of a target and source meshes are shown in figures 6.6 and 6.5 respectively. From the five geometries which have been used, the smallest domain has a boundary extension equivalent to $\delta/c = 0.1$, the largest domain on the other hand has an extension equivalent to $\delta/c = 0.5$.

6.2.2 Solver set up

A solution set up for the study cases included in this section is composed of the solver, tolerances and PISO control parameters. The solution details for the flow variables are specified in the `fVSolution` dictionary with the corresponding entries listed here in Table 6.1.

The remaining entries in the solution dictionary for every pisoFoam based solver consist in the solver settings for additional transported quantities, such as $\tilde{\nu}$; finally the entries for the PISO algorithm specify the number of correctors, set to 2 for sufficiently converged p solutions and both the pressure level reference value and location.

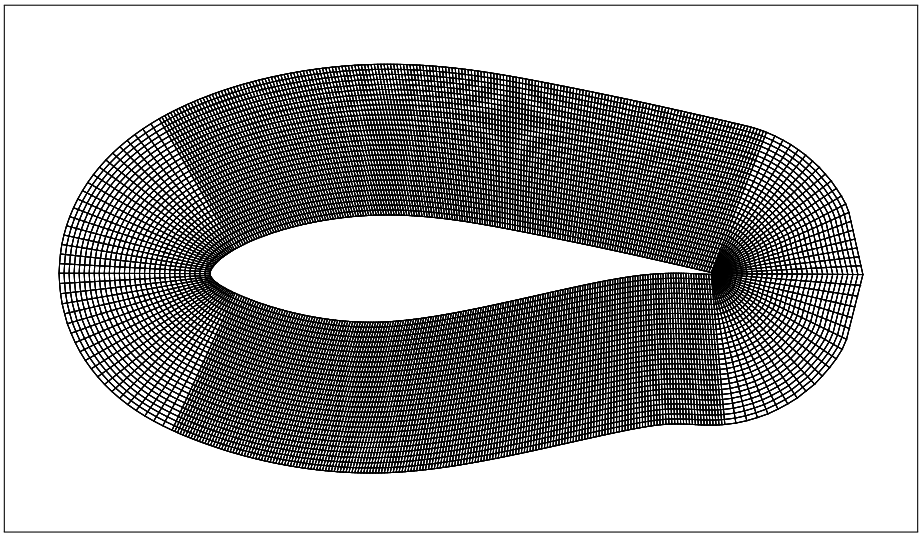


Figure 6.5: Structured 'target' mesh around a DU 00 W-212 airfoil.

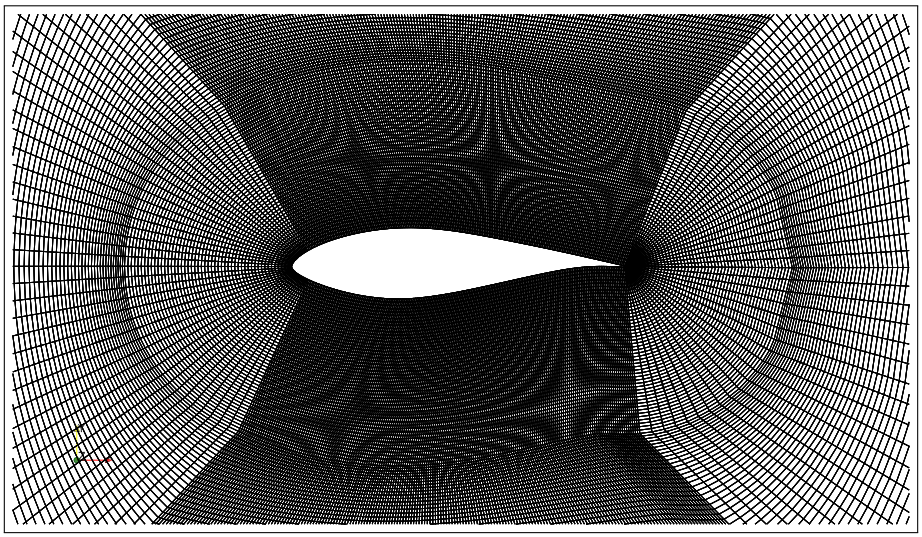


Figure 6.6: Structured 'Source' mesh around a DU 00 W-212 airfoil.

Variable	p	$\mathbf{u}$	$\tilde{\nu}$
Solver	PCG	PBiCG	Smooth Solver
Preconditioner/(Smoother)	DIC	DILU	(Gauss-Sidel)
tolerance	1×10^{-6}	1×10^{-5}	1×10^{-5}
relTol	0	0	0

Table 6.1: Solver specifications for velocity and pressure.

6.2.3 Influence of boundary-wall proximity

The analysis on the size of the Eulerian domain size for each one of the 'target' simulations is divided into the qualitative analysis of the pressure coefficient in the flow field and the quantitative analysis of the airfoil surface distribution for the pressure coefficients and the time history of the force coefficients. A primary aim of this analysis is to observe how does the pressure coefficient in the 'Target' simulations compares to the same results from the 'source' simulation, paying extra attention to the boundary regions of Ω_E where the Eulerian subdomain ends and, to the airfoil wall boundary off course, since the pressure distribution in this region is responsible for a portion of the resulting loads.

Several plots have been assembled between figures 6.7 and 6.11 showing how the C_p distribution behaves across the flow field from both the 'source' and 'target' cases. Given that the simulation has been performed with steady turbulent flow, using the Spalart-Allmaras one equation turbulence model all of the plots represent the flow field after convergence to steady state is reached. The main features in the source simulation, corresponding to the solid-colour contours, are the high pressure region just in the stagnation point of the airfoil, and the low pressure region in the upper surface, around the 25% of the chord length. The empty contours representing all of the target cases, show that in general, the pressure field near the stagnation point and the suction side of the airfoil follows the pressure field from the source case since the low and high pressure regions are located in the same areas. Nevertheless the similarity between the source and target simulations can be appreciated clearly since the contour levels for the target pressure field are shifted by an amount with respect to the source simulation. Under this results it can be pointed that the resemblance between the target cases among themselves appears to be less defined than the difference between any target case and the source case.

The analysis of the C_p over the surface of the airfoil stems from this initial comparison of the overall flow fields; aiming to identify the general shape of the C_p distribution and its resemblance to the reference distribution, taken once again from the Source simulation. All the comparative plots have been gathered in figure 6.12. For this case as for any analysis on the pressure solutions, it it known that for an independent target domain, the pressure solution will differ by an arbitrary constant, therefore the solutions are compared taking as a reference the minimum value of each field. The result from such a comparison shows a C_p distribution that resembles in shape and, more or less amplitude, to that of the reference simulation. The stagnation C_p for the reference case, which is set with the boundary conditions from figure 3.2, is close to unity. The stagnation C_p in the target simulations however does not, therefore the pressure values are displaced by an offset and matched at a specific location in the chord-wise direction so a comparison can be done.

The discrepancy between the target case and the source case C_p distribution is probably better reflected by computing the difference of the coefficient and visualizing it as a surface distribution as well. This information is presented in figure 6.13 which reflects how the pressure coefficient prediction for all of the target case keeps a close resemblance, which is, an absolute difference on the same order of magnitude. The results as one observes the error near the trailing or leading edge regions are not as good, since a pronounced difference is present in those areas; this is in fact recognizable from the plots of figure 6.12 alone. The C_p distributions near the trailing edge for any of the target simulation is shifted and with an apparent step change in the edge of the airfoil, at $x/c = 1$.

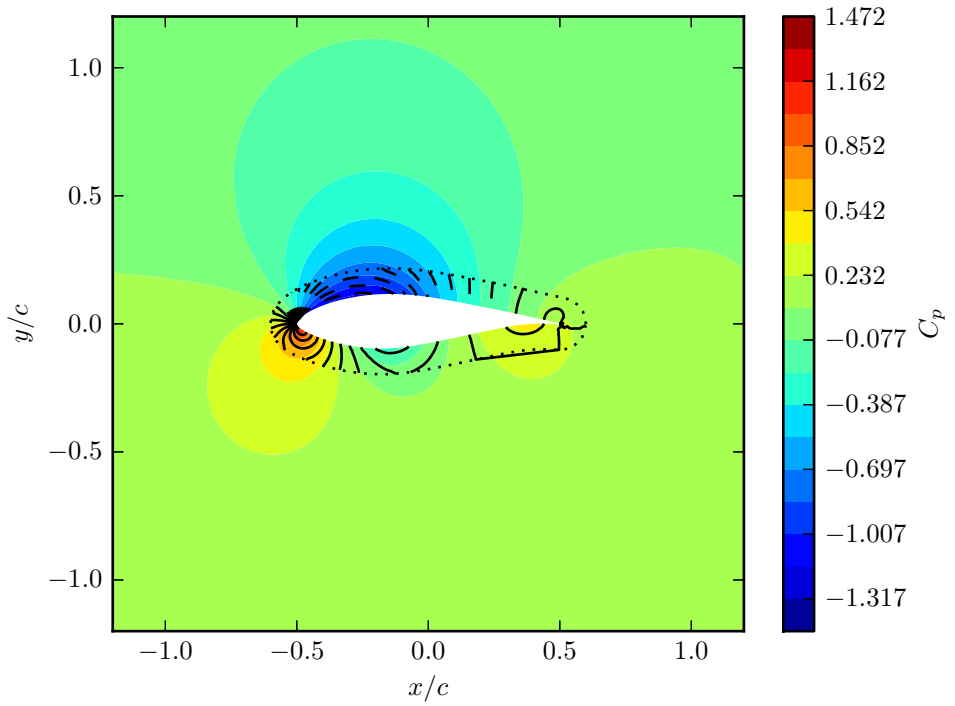


Figure 6.7: Comparison of C_p field for $\delta/c = 0.1$ with respect to the reference case.

The analysis on the force coefficient history has similarity with the pressure coefficient in the sense that there is both a general resemblance and a steady value is reached after a certain flow time. Both C_l and C_d values are presented in figure 6.14, and are the result of a simulation campaign where the target simulation cases have a Dirichlet boundary condition for p as well as for $\mathbf{u}$. The results included in figure 6.15 are obtained instead from a pure Neumann problem for the pressure, with fixed gradients specified for the wall and outer boundaries (see figure 3.2). As can be seen this exercise differs from that of section 6.1 because the flow field domain is delimited by two boundary regions instead of only one. As long as at least one of the regions is set with Neumann boundary conditions, in this case the wall with the zero gradient condition, a solution can be obtained.

The solutions for the first scenario, where the pressure is fixed at the outer boundary, show an overestimated C_d for all of the target cases. In fact the predicted value of $C_d = 0.065$ for the reference case, labeled as 'base' is considerably higher than the value reported in [1], where a $C_d = 0.015$ is computed; this means that the best prediction in this case is about 4 times the value taken from the literature. The values for C_l show a similar degree of variation among the target cases, but slightly greater than for the source simulation. The target simulations for both simulation sets show a variable lift coefficient; the simulations with larger domains present values closer to $C_l = 1.15$; which has a relative difference of about 4% in comparison to the value reported in [1]. The prediction of the lift coefficient seems to be then considerably better than the drag prediction, particularly for the target simulation cases. The base C_l in the series from figure 6.14 and figure 6.15 is not that close to the value reported in the literature, since both values differ by a relative error of

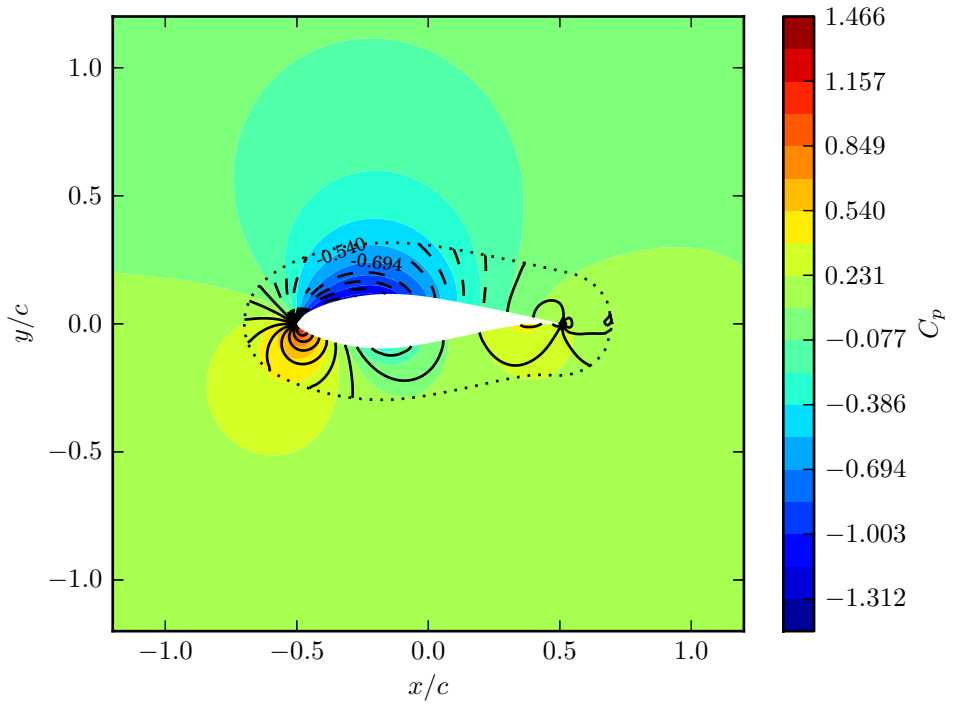


Figure 6.8: Comparison of C_p field for $\delta/c = 0.2$ with respect to the reference case.

about 16%.

6.3 Discussion

So far the consistency of the solver depends not only on the interpolation of the velocity boundary condition from the Lagrangian field, represented in this work by the source case, but it is also dependent on the interpolation or in a general sense, the computation of the boundary condition for the pressure. The satisfaction of the momentum equation when a Dirichlet velocity BC and a Neumann pressure BC are considered seems to be a necessary condition for the problem to be consistent for the pressure equation as well. By judging the convergence rates in the exercise of the point vortex flow, it is not difficult to observe that the convergence rates from cases with interpolated boundary conditions resulted in degraded convergence in relation to the case with computed BC's from an exact expression. Furthermore, the rate of convergence of the best case, corresponding precisely to the simulation with exact boundary conditions, barely resulted in a convergence rate of 1.3 approximately, even though the discretisation schemes in the case are second order accurate.

For both the cases of the point vortex and airfoil benchmarking cases, linear interpolation was specified, this means that a central differencing spatial discretisation is used and should yield second order accurate solutions. The time stepping discretisation was initially treated with an Euler implicit order, which brings the solution convergence to

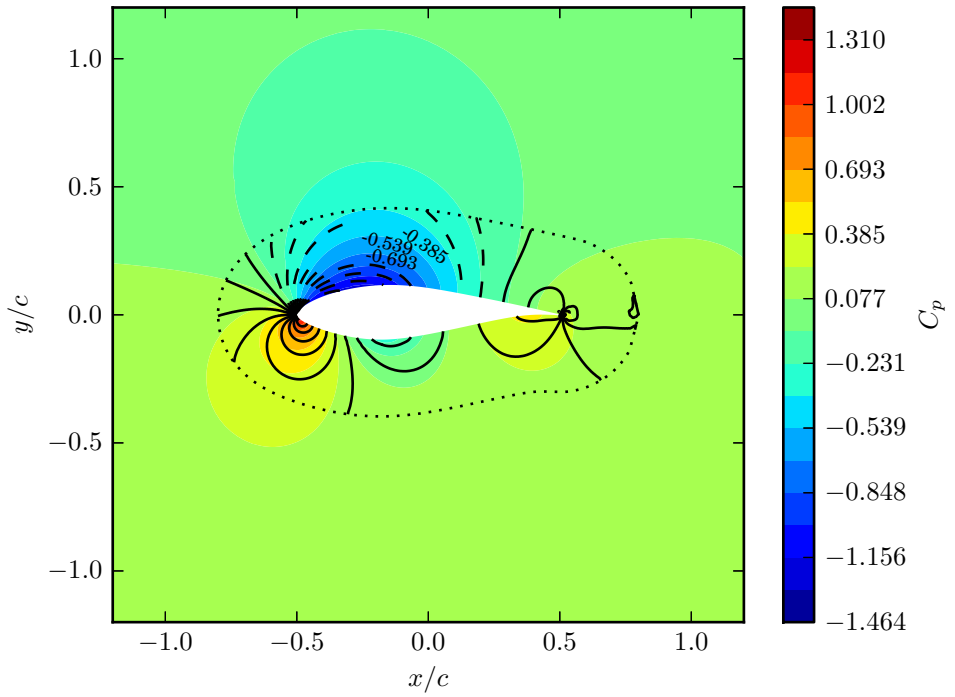


Figure 6.9: Comparison of C_p field for $\delta/c = 0.3$ with respect to the reference case.

first order and Subsequently a second order time stepping, the Crank-Nicholson scheme, was considered for the point vortex flow case. According to Issa [16], at least 3 pressure correction stages must be done so that the discretisation error introduced by the PISO algorithm reaches second order accuracy. Second order convergence is not reached since it was not possible to guarantee that the boundary conditions fulfilled the momentum equation evaluated on the boundaries; it is not certain whether the corrected pressure equation is yielding a consistent system for every PISO corrector step after the artificial flux correction.

It has been known from other authors that the interpolation from the Eulerian solution to the Lagrangian field, in the correction step of the hybrid solution algorithm, is done avoiding the wall and the boundary of the Eulerian domain. The results on both the consistency study and the simulations with DU00-W-212 airfoil show that the values in the regions adjacent to the Eulerian domain boundary, should be excluded from the set of interpolation values used in the hybrid Framework. This measure could counteract the introduction of error when the flow field values near the boundaries do not fulfil the boundary condition as happens to be the case in figure 6.3.

The error observed near the Eulerian domain boundary is however drastically reduced when $n \to \infty$; after paying closer attention to this situation it could be stated that the extent to which the Eulerian region is segregated for the interpolation of the correction step depends on the level of refinement of the Eulerian mesh and also how the velocity field is reconstructed between adjacent cell centres and the boundary itself.

The analysis on the extent of the Eulerian domain normal to the airfoil wall, shows that

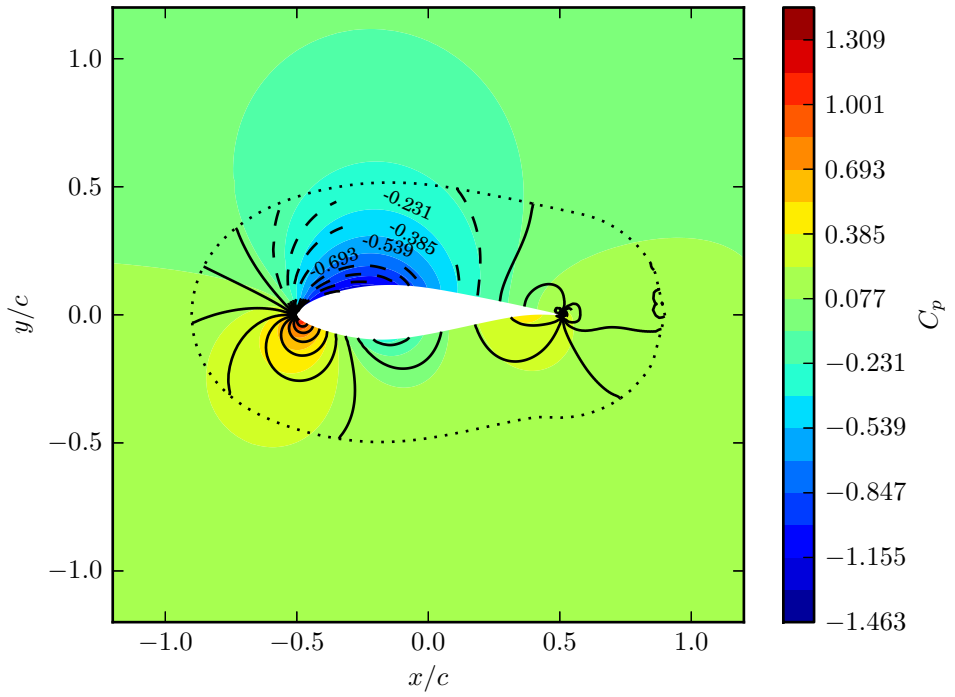


Figure 6.10: Comparison of C_p field for $\delta/c = 0.4$ with respect to the reference case.

the difference between the C_p distributions is notable for most of the airfoil surface, unfortunately, the pressure field in the trailing edge and the stagnation point is clearly sensitive to the boundary condition setting when this is interpolated between time steps. The shape of the C_p distribution is similar for all of the domain sizes in the airfoil simulation campaign, but the difference that results in the force coefficients, particularly drag, cannot be considered as reliable. Supposing that the boundary condition setting is tuned to match the pressure distribution in the Eulerian domain to that of an equivalent far-field simulation, the analysis could be extended considering that the size limit in the Eulerian domain would be conditioned to the mesh refinement level and, also considering how much of the solution field can then be selected for building the interpolation functions in the correction step. This latter idea is rather speculative and would be conditioned to further improvement of the accuracy in the prediction of aerodynamic loads via a reduced domain simulation.

6.3.1 The research questions in perspective

The prediction of flow properties with FVM offers the advantage of accounting for mass conservation when compared with other solution methods. In order to define the implications of integrating a FV solver to a hybrid solution framework, it is necessary to state that the solution methodologies for the $\mathbf{u} - p$ system already rely on the correction of mass flux to ensure continuity. As part of the hybrid solver an additional correction of the mass flux is required in order to ensure continuity at the Eulerian boundary after the

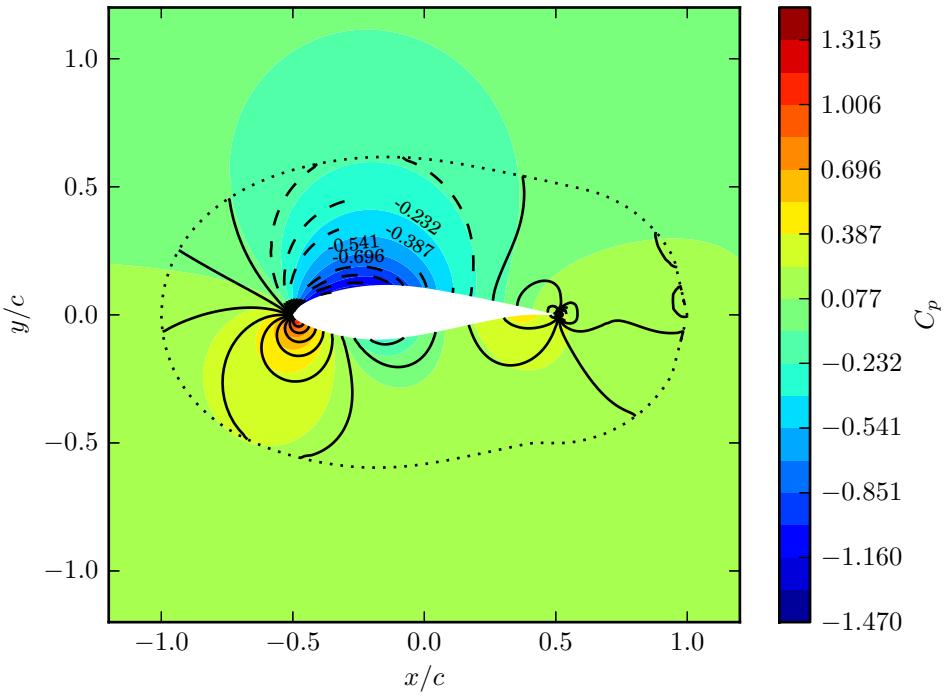


Figure 6.11: Comparison of C_p field for $\delta/c = 0.5$ with respect to the reference case.

u boundary condition is computed from the Lagrangian domain. This step takes place directly and once per time step; in consequence a discrepancy between the Lagrangian and Eulerian velocity fields at the interface $\partial\Omega_E$ will remain at the end of the solution, resulting in a non-conservative velocity field. The conservative nature of the FVM is then compromised when the coupling of the velocity solutions takes place according to the 'direct' approach for determining the velocity boundary condition at the sub-domain interface $\partial\Omega_E$.

The overall balance on the performance of the hybrid solver is completely centred on the performance of the stand-alone Eulerian solver implemented in OpenFOAM since no full coupling with the vortex method software is done. The solution of the first benchmarking exercise, concerning a flow field induced by a stationary point vortex, shows that the prediction of velocities can be performed in an isolated domain matching the exact velocity field; the prediction of the pressure field is less convincing as not only the mass conservation for the velocity boundary condition is to be preserved but also the momentum conservation for both the pressure and velocity boundary conditions simultaneously. The benchmarking case of the DU00-W-212 airfoil resulted in similar discrepancies for the solution of the pressure field. The prediction of the force coefficients also left several doubts that must be addressed for ensuring full functionality of the stand-alone solver; the correct solution of the flow in the boundary layer must be carefully guaranteed in order to filter out the effect of turbulence from the effect that the mass flux correction alone has on the reported differences between the force coefficients, particularly the coefficient for the drag force.

The final version of the openFOAM stand-alone solver has been used in the benchmarking of two steady-state cases. Although the customized version has been developed on the basis of a transient flow solver, the simulation of heavily transient flow structures resulted in discontinuities of the flow field in the region near to the interface. This problem is also triggered by the initial notion of implementing the custom solver with a boundary condition class in the OpenFOAM syntax that enables time variation. All these sophistications were dropped in order to achieve the working solver which has been finally tested, with stationary solution in time and consequentially, stationary boundary conditions.

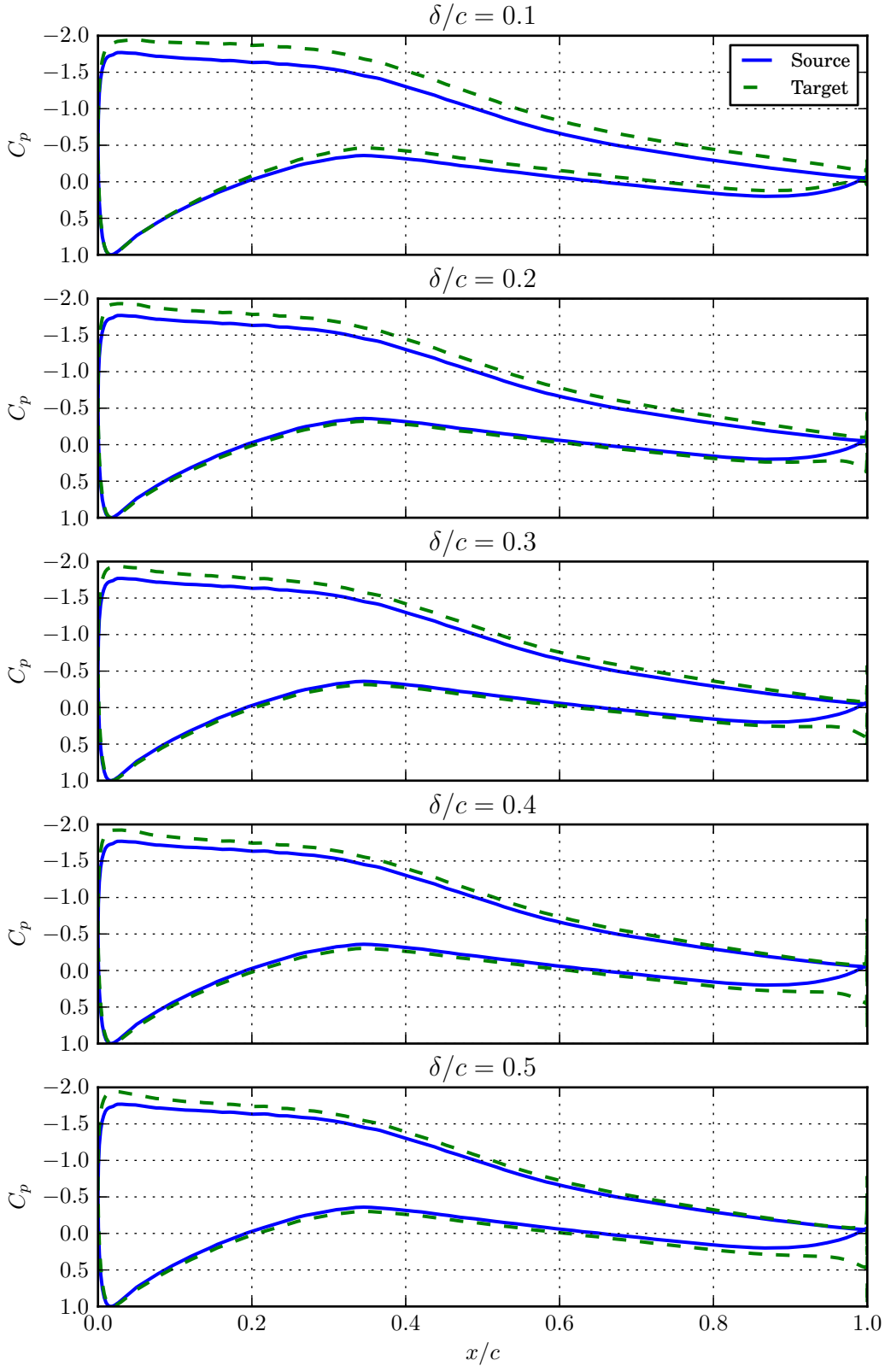


Figure 6.12: Surface C_p distribution variation with δ/c for source and target case results.

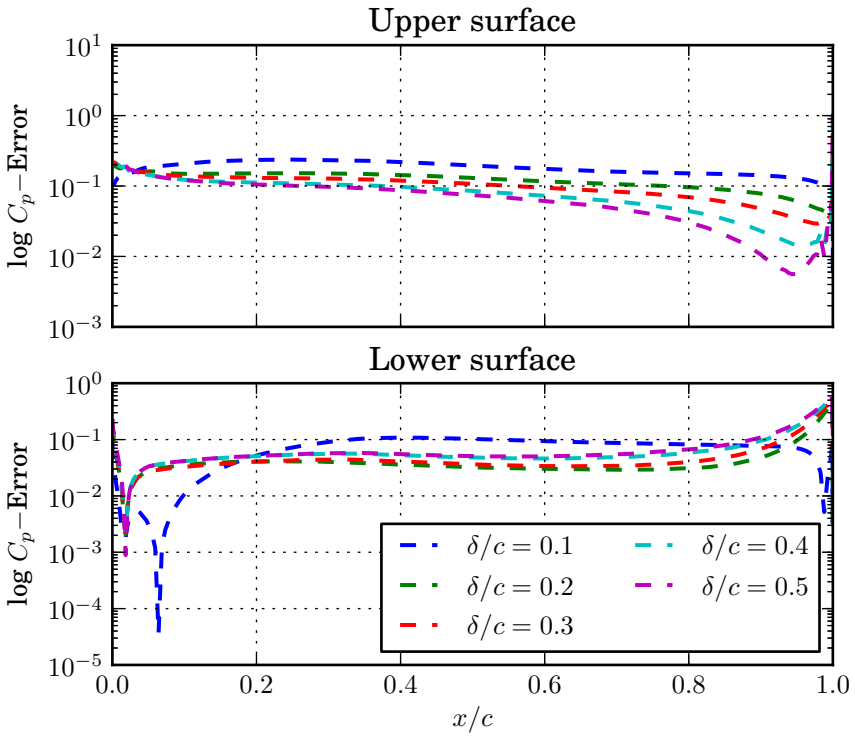


Figure 6.13: Plots of the absolute error in the C_p distribution across the airfoil surface.

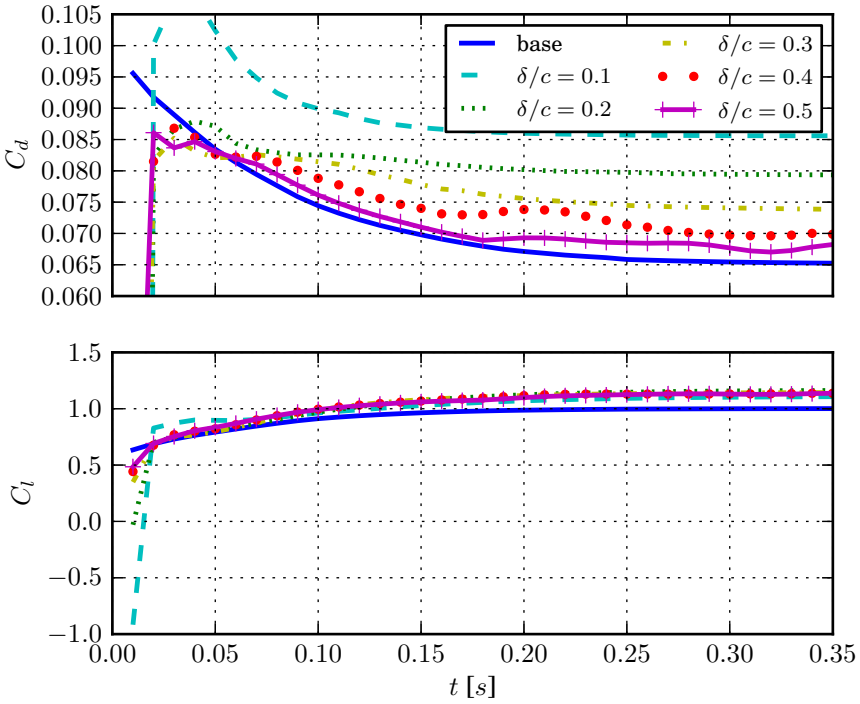


Figure 6.14: Comparative time history of force coefficients Dirichlet BC for pressure.

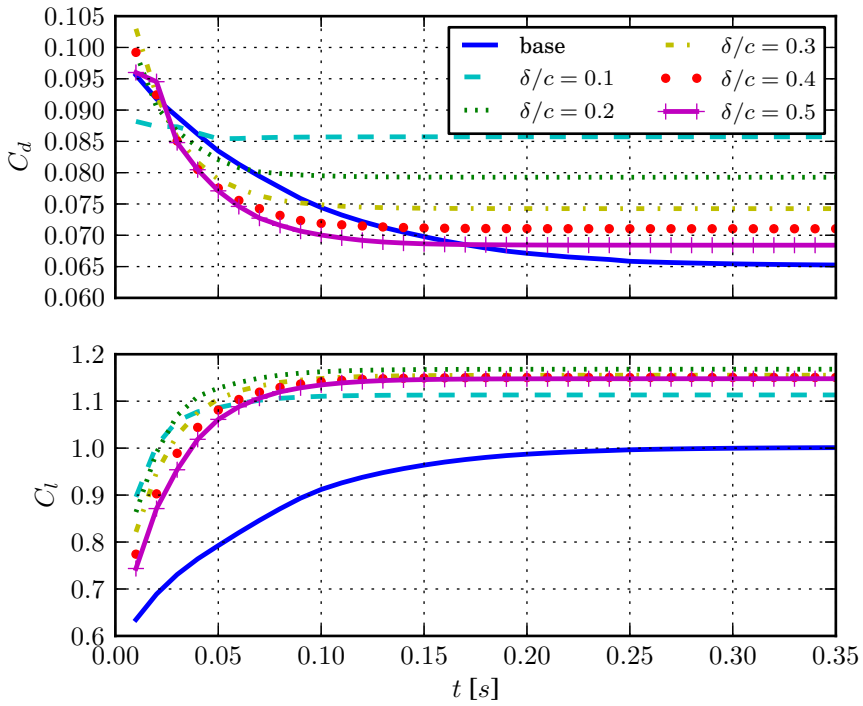


Figure 6.15: Comparative time history of force coefficients.

Conclusions

A PISO solver has been implemented taking as a baseline the pisoFoam solver, with a customized functionality for specifying non-conservative pure Dirichlet velocity boundary conditions. Using this version as a stand-alone module it has been possible to explore the consistency of the solver, analysing the accuracy of the velocity and pressure fields. The solution for the velocity components was proven to have an accuracy above 1, with some obstacles such as the possible inconsistency of the pressure equation, keeping the system to reach higher solution accuracy due to an unfulfilled set of boundary conditions for the central case, with pure Dirichlet velocity boundary conditions and pure Neumann pressure boundary conditions.

With the solution of a stationary vortex-induced flow field, for which an exact solution is available, it has been possible to judge the effect of the boundary conditions on the solution procedure. The extraction of boundary conditions for the Dirichlet velocity problem from an exact flow field computed at random locations was useful to observe that the consistency of the velocity solution depends on the interpolation method used to determine the boundary conditions, with base on the available data at the mentioned random locations. In the present analysis, both linear and cubic interpolation showed acceptable results while nearest-neighbour interpolation resulted in less accurate solutions. The relevance of the exercise lies in the fact that the operation of a hybrid flow solver has the same principle in which a Lagrangian flow field provides the boundary condition for the Eulerian flow field solution. The determination of the pressure boundary condition for the same case was explored by testing the solution quality of a homogeneous pressure Poisson problem and contrasting it with the solution for an heterogeneous counterpart which corresponds in fact to the expected solution.

From the behaviour of the pressure field in the point vortex exercise, it is inferred that the flux correction, central to the custom solver for ensuring mass-conservativeness in the velocity Dirichlet BC, is not necessarily being conservative in terms of the momentum equation, which may result in an inconsistent linear system when the equation for the corrected pressure at each PISO stage is solved.

Additional study of the flow field around an airfoil, shows that when the boundary conditions for pressure and velocity are available exactly at the locations where the Eulerian

boundary is defined the solution preserves general features in the flow field and the general shape of the pressure distribution. On the other hand, the comparison with reference simulations, both implemented in this project and the literature, show that the force coefficients are particularly sensitive. The drag coefficient prediction was found to be difficult since the difference with both reference values is quite broad. The development of the flow field around the trailing edge of the airfoil is an aspect to take into consideration since it is a likely cause of error. The prediction of an inconsistent pressure field in the trailing edge region may introduce artificial oscillations that affect the lift and drag predictions.

Recommendations

In addition to the modifications made to the **u** boundary condition, it is necessary to implement an analogous modification to the code that takes care of the p boundary condition using as a reference the conservation of momentum in the same way as mass conservation is considered when the velocity correction at the boundary is ensured.

Implementing a fully transient solver has presented several challenges, the first of them is the need for boundary condition of Neumann type that enables updating the p boundary condition on a time-wise basis. Additionally the iteration procedure for each time step must be carried such that the flow field at the end of each time step solution is converged to the point that it matches the time dependent boundary condition. Initial tests on this particular issue resulted in unbounded velocities due to an artificial increase in the vortex strength for the smallest vortices that reached the boundary in simulations with full Dirichlet **u** boundary conditions.

The possibility of changing the general boundary condition in openFOAM such that the prescribed values are determined at the boundary face locations would be a major improvement over the actual state of the project. The boundary condition class used for specifying the velocity values uses a linear interpolation for mapping the values in space, from an arbitrary set of locations specified via an auxiliary entry with point coordinates, to the boundary face centre locations. Considering that the extractions of values from a generic Lagrangian field is also done with interpolation, one could certainly assume that reducing the number of times the values undergo interpolation would at least reduce the computational effort if not also the accuracy degradation.

References

[1] C. Baptista, D. Baldacchino, C. Ferreira, A. van Zuijlen, G. van Bussel, and N. Sørensen. Comparison of Spalart-Allmaras and $k - \omega$ SST for high Reynolds steady airfoil simulations. 2016. 4th Symposium on OpenFOAM in Wind Energy, May 2-4, Delft, the Netherlands.

[2] T. M. Burton and J. K. Eaton. Analysis of a fractional-step method on overset grids. *Journal of Computational Physics*, 177(2):336–364, 2002.

[3] A. J. Chorin. Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, 57:785–796, 3 1973.

[4] I. P. Christiansen. Numerical simulation of hydrodynamics by the method of point vortices. *Journal of Computational Physics*, 13(3):363 – 379, 1973.

[5] R. Cocle, G. Winckelmans, and G. Daeninck. Combining the vortex-in-cell and parallel fast multipole methods for efficient domain decomposition simulations. *Journal of Computational Physics*, 227(21):9091 – 9120, 2008.

[6] G. H. Cottet. Particle-grid domain decomposition methods for the navier-stokes equations in exterior domains. *Lectures in Applied Mathematics*, 28:103–117, 1991.

[7] G. H. Cottet and P. D. Koumoustakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, 2000.

[8] G. Daeninck. *Developments in Hybrid Approaches*. PhD thesis, Universit catholique de Louvain, 2006.

[9] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer, 2002.

[10] B. Gao, S. S. Xu, and Z. N. Wu. A note on hybrid eulerian/lagrangian computation of compressible inviscid and viscous flows. *Journal of Computational Physics*, 226(1):1–16, 2007.

[11] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[12] C. J. Greenshields. *OpenFOAM, The Open Source CFD Toolbox - Programmer's Guide*. OpenFOAM foundation Ltd., 2015.

[13] C. J. Greenshields. *OpenFOAM, The Open Source CFD Toolbox - User Guide*. OpenFOAM foundation Ltd., 2015.

[14] J. L. Guermond, S. Huberson, and W. Z. Shen. Simulation of 2d external viscous flows by means of a domain decomposition method. *Journal of Computational Physics*, 108(2):343 – 352, 1993.

[15] Q. Hu, N. A. Gumerov, and R. Duraiswami. GPU accelerated fast multipole methods for vortex particle simulation. *Computers & Fluids*, 88:857 – 865, 2013.

[16] R. I. Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62(1):40 – 65, 1986.

[17] H. Jasak. *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. PhD thesis, Imperial College of Science, Technology and Medicine., 1996.

[18] A. Leonard. Vortex methods for flow simulation. *Journal of Computational Physics*, 37(3):289 – 335, 1980.

[19] L. Manickathan. Hybrid lagrangian-eulerian vortex particle method. http://www.lr.tudelft.nl/fileadmin/Faculteit/LR/Organisatie/Afdelingen_en_Leerstoelen/Afdeling_AEWE/Wind_Energy/Education/Masters_Projects/Finished_Master_projects/doc/LentoManickathan_r.pdf, December 2017. Delft University of Technology.

[20] M. L. Ould-Salihi, G. H. Cottet, and M. El Hamraoui. Blending finite-difference and vortex methods for incompressible flow computations. *SIAM Journal on Scientific Computing*, 22(5):1655–1674, 2001.

[21] A. Palha, L. Manickathan, C. S. Ferreira, and G. van Bussel. A hybrid eulerian-lagrangian flow solver. https://arxiv.org/abs/1505.03368v2, June 2015. arXiv:1505.03368v2.

[22] G. Papadakis and S.G. Voutsinas. In view of accelerating CFD simulations through coupling with vortex particle approximations. *Journal of Physics: Conference Series*, 524(1), 2014.

[23] T. K. Sheel and S. Obi. High-performance computing techniques for vortex method calculations. *Theoretical and Computational Fluid Dynamics*, 24(1-4):175–179, 2010.

[24] T. K. Sheel, K. Yasuoka, and S. Obi. Fast vortex method calculation using a special-purpose computer. *Computers & fluids*, 36(8):1319–1326, 2007.

[25] M. J. Stock, A. Gharakhani, and C. P. Stone. Modeling rotor wakes with a hybrid overflow-vortex method on a GPU cluster. 2010. 28th AIAA Applied Aerodynamics Conference.

[26] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics*. Longman Scientific & Technical, 1995.

[27] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in physics*, 12(6):620–631, 1998.

[28] R. Yokota and S. Obi. Vortex methods for the simulation of turbulent flows: Review. *Journal of Fluid Science and Technology*, 6(1):14–29, 2011.

[29] F. Zahle. *Wind Turbine Aerodynamics Using an Incompressible Overset Grid Method*. PhD thesis, Imperial College of Science, Technology and Medicine., 2006.

[30] Y. Zang and R.L. Street. A composite multigrid method for calculating unsteady incompressible flows in geometrically complex domains. *International Journal for Numerical Methods in Fluids*, 20(5):341–361, 1995.