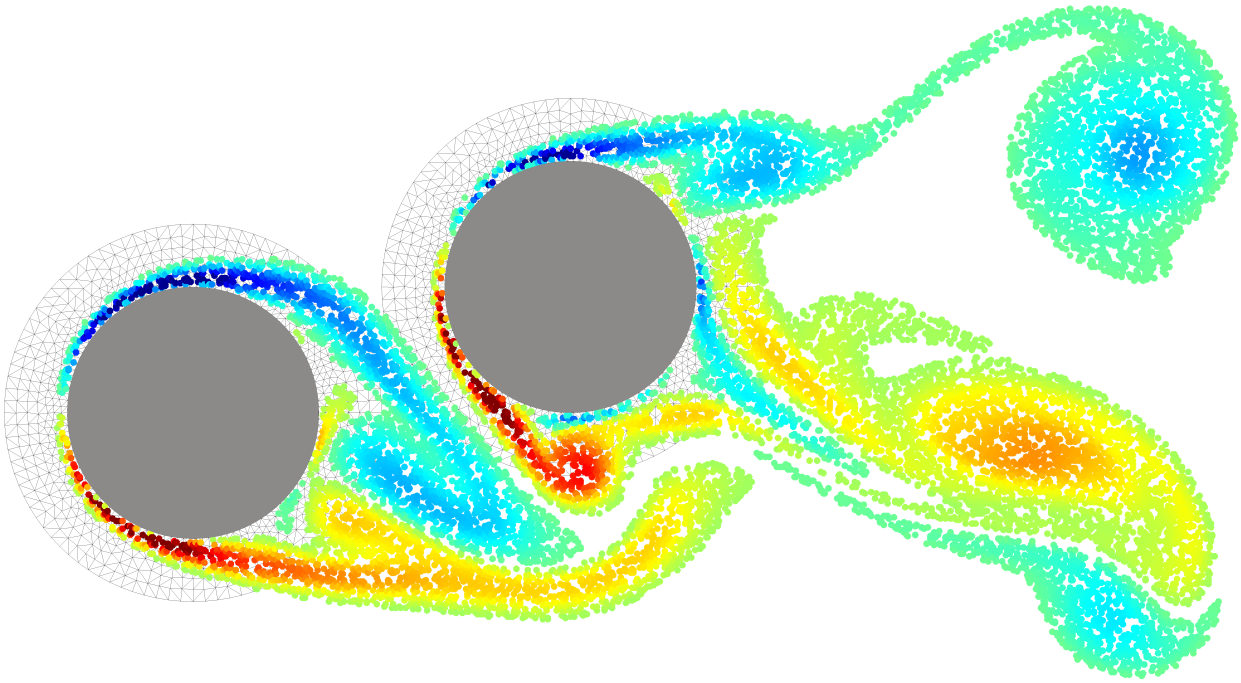
Master of Science Thesis



Hybrid Eulerian-Lagrangian Vortex Particle Method

A fast and accurate numerical method for 2D Vertical-Axis Wind Turbine

L. Manickathan B.Sc.

Date TBD

Faculty of Aerospace Engineering · Delft University of Technology

Hybrid Eulerian-Lagrangian Vortex Particle Method

A fast and accurate numerical method for 2D Vertical-Axis Wind Turbine

Master of Science Thesis

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

L. Manickathan B.Sc.

Date TBD

Faculty of Aerospace Engineering · Delft University of Technology

Delft University Of Technology
Department Of
Aerodynamics and Wind Energy

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Hybrid Eulerian-Lagrangian Vortex Particle Method"** by **L. Manickathan B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: Date TBD

Head of department: _____
prof.dr.ir. G.J.W. van Bussel

Academic Supervisor: _____
dr.ir. C.J. Simao Ferreira

Academic Supervisor: _____
dr.ir. A. Palha da Silva Clerigo

Summary

The wake geometry of a Vertical Axis Wind Turbine (VAWT) is unlike the standard Horizontal Axis Wind Turbine (HAWT). The blades of the turbine continuously passes through its own wake, creating complex wake-body interactions such as flow separation and dynamic stall, and convectional grid-based numerical method which is capable of describing such near-body phenomena fails at efficiently resolving the wake geometry. However, as these phenomena have a direct impact on the performance of the VAWT, it is paramount that there exists a numerical method that is not only capable of accurately resolving the small-scale near-body phenomena but also excels at efficiently resolving the unsteady large-scale wake geometry.

This was the goal of the research and the numerical method that satisfied these requirements was the domain decomposition method known as the Hybrid Eulerian-Lagrangian Vortex Particle Method (HELVPM), based on the doctoral thesis of Daeninck [24] and the additional study performed by Stock [61]. In the present study, we coupled an Eulerian Finite Element Method, which only resolves the near-body domain, with a Lagrangian Vortex Particle Method, which resolves the entire wake. The advantage of such fluid domain segregation was that the Eulerian method could focus on accurately describing the near-body features whereas the Lagrangian method could focus on efficiently evolving the wake using simulation acceleration methods such as Fast Multipole Method (FMM) and parallel computation in Graphical Processing Units (GPU).

The present study initially developed, verified and validated the Finite Element method and the Vortex Particle method separately ensuring it performs according to the theory. These methods were then coupled using the algorithm of Daeninck [24] and Lagrangian correction strategy developed by Stock [61]. However, during the study we determined that additional modifications to the coupling strategy is required to ensure conservation of circulation. Furthermore, it was determined that the spatial resolution of numerical method at the overlap region, where coupling takes place, plays a crucial role in the accuracy of the coupling.

Even though the hybrid method of the present study sacrificed some efficiency to ensure an accurately coupled scheme, we must not that it is still at its infancy. With the help of

advanced techniques such as varying particle core size, higher order time marching scheme in the Eulerian method, and boundary element method acceleration techniques such as FMM and/or GPU calculation, the hybrid method has the potential to substantially outperform standard grid based methods. In conclusion, the hybrid method that has been developed here has the potential to accurately describing the near-wake phenomena and efficiently evolving the wake of a VAWT.

Acknowledgements

The work I present here would not have been possible without the support of my supervisors dr.ir. C. (Carlos) J. Simao Ferreira and dr.ir. A. (Artur) Palha da Silva Clerigo. I want to thank Carlos for the constant encouragement and ensuring that I didn't loose myself in the detail and to have a global picture. Thank you for also giving me fresh perspective when it seems impossible and making me to think outside the box. I want to thank Artur for his daily supervision and guidance to teach me all the intricate details for developing this numerical method. I also want to thank him for putting aside time to work hand-in-hand to find solutions to the problems I faced and supporting me throughout the thesis.

I want to thank my friends: Chid, Thor, Oliver, Mark, Rob, Alberto, Hector and Dieter for spending time together at the university, tackling the challenge of finishing the master thesis together. You guys gave me joy during the work and it was fun working beside you guys. I want to thank my other friends you made by my last few years in Netherlands an enjoyable experience.

Finally, I want to thank my family: Daddy, Mummy, Chechi, and Denis Chettan. Thank you for your patience and your unending love throughout my life.

Delft, The Netherlands
Date TBD

L. Manickathan B.Sc.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

AD	Actuator Disk
BEM	Blade Element Momentum
CFD	Computional Fluid Dynamics
CG	Continuous Galerkin
CSVP	Constant-Strength Vortex Panel
DOF	Degrees of Freedom
FDM	Finite Difference Method
FEM	Finite Element Method
FE	Forward Euler
FMM	Fast Multipole Method
FVM	Finite Volume Method
GPU	Graphical Processing Units
HAWT	Horizontal-Axis Wind Turbine
HELVPM	Hybrid Eulerian-Lagrangian Vortex Particle Method
ICNS	Incompressible Navier-Stokes
IPCS	Incremental Pressure Correction Scheme
ISC	Impulsively Started Cylinder
lhs	left hand side
LSTSQ	Least-Square solution method
MPI	Message Passing Interface
NS	Navier-Stokes

PC	Population Control
PIV	Particle Image Velocimetry
PSE	Particle Strength Exchange
RWM	Random Walk Method
VAWT	Vertical-Axis Wind Turbine
VPM	Vortex Particle Method
WRS	Wee Remeshing Scheme

Chapter 1

Introduction

Conventional energy resources such as fossil fuels and nuclear energy are not only limited but also pose adverse effects on the environment. Therefore, we are striving to find a cheap and renewable source of energy. Wind energy is such source of energy, getting more popular and more affordable. Novel wind turbine designs such as the Vertical-Axis Wind Turbine (VAWT) are now a promising research field that can satisfy this growing demand.

VAWTs are unlike the normal wind turbines, which are mounted on a mast away from the ground and generate energy by spinning perpendicular to the ground, figure 1.1, whereas the Horizontal-Axis Wind Turbine (HAWT), spins parallel to the ground with its hub located at the ground, figure 1.1b. The VAWT has it's generator located at the ground, allowing it to be easily accessible and maintained. However, the main advantage is the early wake dissipation of VAWTs. Near-wake experiments of Ferreira (2009) [57], and simulations of Vermeer (2003) [66] have shown that the fluid past the turbine is more turbulent. Due to this higher turbulence, the flow is able to recover much earlier than convectional wind turbines. This allows the turbines to be placed much closer, potentially

(a) VAWT: Darrieus wind turbine[18]

(b) HAWT: Offshore wind turbine [19]

Figure 1.1: VAWT vs. HAWT

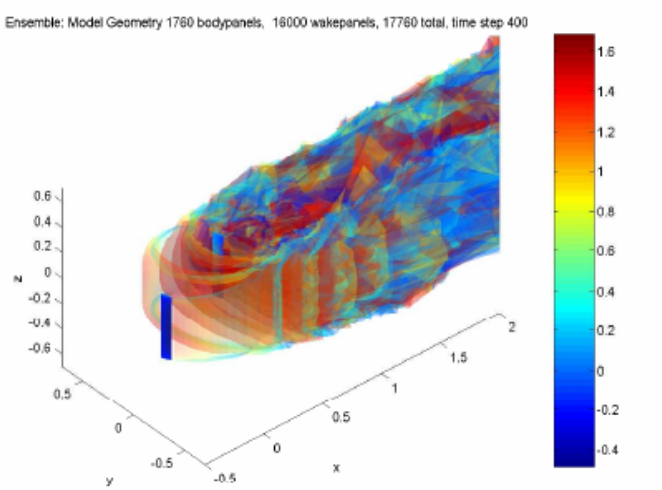


Figure 1.2: 3-D Unsteady Panel simulation of a Straight-bladed VAWT showing the strength of the shed vorticity. The VAWT blades interact with their own wake increasing the complexity of the wake geometry, source: Dixon et. al [27].

outputting more power per ground. Furthermore, VAWTs operate independently of the flow direction, and can operate at low wind speeds (i.e. at low tip-speed ratios).

However, there are some limitations that we must take into account. As the blades pass through their own wake, complex wake-body interactions take place, figure 1.2. These have adverse effects on the blade structure, making it more susceptible to fatigue. As the blade is constantly pitching, flow behaviors such as dynamic stall and constant vortex shedding take place (Ferreira [59]). These complex fluid behaviors makes it hard to predict the performance of a VAWTs and this is one of the reasons why VAWTs are not widely used.

In addition, a VAWT operates at a large Reynolds, number making accurate numerical methods computationally very expensive. So we see that we require a numerical method that can not only reproduce accurate results, but is also efficient at modeling the flow around the turbine.

1.1 Motivation and Goal

The goal of this research is to develop an efficient, reliable and accurate numerical method for modeling the flow around a Two Dimensional (2D) VAWT, enabling to compute the correct performance characteristics. The two approaches of investigating the flow around a turbine are by either using a numerical method to model the flow, or by performing an experimental test, for example in a wind tunnel.

To understand the unsteady aerodynamic behavior, Particle Image Velocimetry (PIV) has been a useful tool to visualize the flow around the turbine. PIV was used by Ferreira et al. (2007) [58], showing that it is possible to measure the flow characteristics around the blade. The downside to experimental investigation is that it is very expensive to investigate all types of airfoil geometries, blade geometries and VAWT configurations.

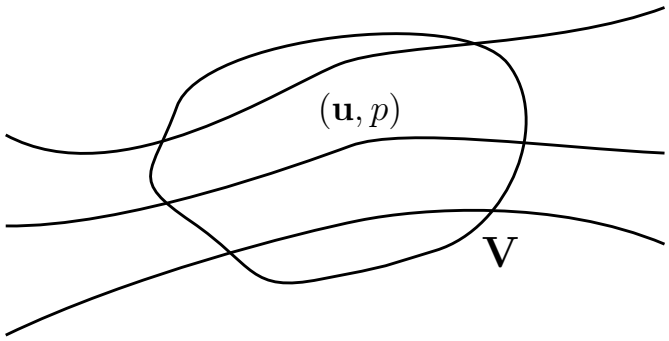


Figure 1.3: Eulerian formulation of the fluid. We observe a given volume $\mathbf{V}$ and evaluate the change in properties of the fluid, velocity $\mathbf{u}$ and pressure p at time passes.

However, investigating this is vital in understanding the performance characteristics of VAWT. Furthermore, it is difficult to perform experiments on arrays of wind turbines in a wind tunnel.

Numerical methods are therefore a popular alternative as the cost of simulation is becoming progressively smaller, and the accuracy of the models is increasing day by day. Actuator Disk (AD) and Blade Element Momentum (BEM) models are the simplest models, built upon satisfying the momentum balance of the turbine with the fluid. The advantage is that they are very fast, however they lack the accuracy that is obtained by experimental simulation. Flow phenomena such as dynamic stall and flow separation cannot be modeled by these methods, and therefore we must rely on more powerful tools.

To ensure more accuracy, one has to solve the Navier-Stokes equation of the flow around the turbine without large simplifications. Computional Fluid Dynamics (CFD) methods discretize the fluid into smaller cells (or volumes) and solve the Navier-Stokes equations. This type of formulation is known as an Eulerian formulation as we are evaluating the change in flow property in a given cell/volume, figure 1.3. In order to fully resolve the flow around the turbine, we would need a fine mesh near the blade where we have small scale vorticies. However, far from the body, where these vorticies dissipated into low frequency vortical structures, we can have lower mesh resolution. This means that at various regions of the flow, we require mesh resolutions of various magnitudes. This becomes a problem when we have moving boundaries as the mesh has to be adapted depending on the location of the body.

An alternative method is to use a Lagrangian formulation of the Navier-Stokes equations, known as vortex methods. These methods employ vorticity transport equations which makes them ideal for describing the evolution of the wake vorticity. Furthermore, they do not require cells/volumes to describe the domain. In addition, they use simulation acceleration methods such as Fast Multipole Method (FMM) and parallel computation in Graphical Processing Units (GPU) making them orders of magnitude faster than the typical CFD methods. However, vortex method cannot inherently take into account the solid body. They require additional methods that can describe the effect of the body in the fluid and the vorticity generated from the body.

We see that Eulerian method is accurate when describing the blade-wake interaction but not efficient when describing multi-scale domains. The Lagrangian method is very efficient

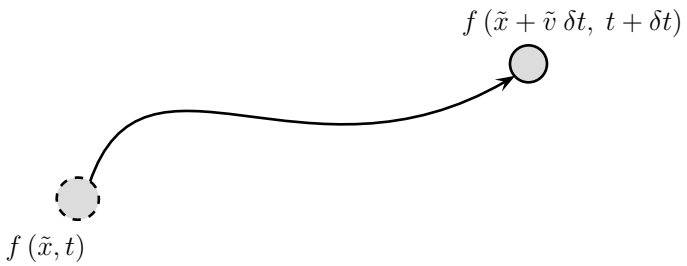


Figure 1.4: Lagrangian formulation of the fluid. We track the path of the individual fluid elements as time passes.

in evolving the vorticity of the fluid. Due to auto-adaptive nature of the Lagrangian method, it is an ideal choice when describing the multi-scale flow characteristics. However, it is not efficient in resolving the near-body region, where the vorticity is generated. Therefore, in order to use the advantages of both methods, we have decided to use a domain-decomposition method, referred to as Hybrid Eulerian-Lagrangian Vortex Particle Method (HELVPM).

For the HELVPM, the Eulerian grid method will be used at the near-walls, and the Lagrangian vortex method will be used in the wake region. With the proper coupling of these methods, we can ensure that this numerical method can capture not only the near-wake phenomena such as vortex shedding, dynamic stall, and the wake-body interaction, but also the large-scale flow structures such as the evolution of the VAWT wake.

1.2 Research Aim and Plan

We have formulated a research that can help us accomplish our goal. The research questions that are derived from the goal of the project is as follows:

Research Questions:

- *Is it possible to develop an efficient and accurate numerical method by an hybrid approach, with the vortex particle method solving the wake, and the Navier-Stokes grid solver solving the near-body region?*
- *Will it be able to predict similar performance characteristics and flow phenomena as observed from the experiments?*
- *Will it be capable of simulating the blade-wake interaction and the dynamic stall?*
- *Where are the errors and what are their sources?*

In order to answer the research questions, the goal of the project is to develop an efficient and accurate numerical method that is not only capable of capturing the small scale flow phenomena such as the dynamic stall and the vortex shedding, but is also efficient at modeling the evolution of the wake. Once the model has been developed, we will verify the approach and validate it against cases obtained from literature.

Research aim and plan:

- *Develop the hybrid method for capturing small-scale phenomena and large scale phenomena.*
- *Verify the efficiency, reliability, and the accuracy of the model.*
- *Verify and validate the model with test cases from literature.*

The innovativeness of this project is that such hybrid modeling has not been yet applied for the wind energy problem case. Through the parallelization of the vortex particle method in a GPU and employing solver acceleration techniques such as the FMM, this simulation could give an edge in the understanding the flow behavior of a VAWT.

1.3 Verification and Validation Test Cases

In order to assess the accuracy of this hybrid formulation, the following test cases have been used:

Lamb-Oseen vortex [43] [63]
Lamb-Oseen vortex test case is an analytical solution derived from the NS equation, and is a test case for unbounded flow (without any wall). This is the first model that will be used to validate the Lagrangian method and Eulerian method separately. As it describes an unbounded flow, we do not need to concern with the vorticity generation problem. This helps us focus on just the evolution of the vorticity field.

Clercx-Bruneau dipole [16]
The Clercx-Bruneau dipole test case is the simple case of a colliding dipole with a wall. This test case will be used to verify and validate the coupling of the Eulerian and the Lagrangian method in the presence of a solid wall. This test case focuses on the interaction of vorticity with the wall making it ideal to verify and validate the proper generation of vorticity and its transfer to the Lagrangian subdomain.

Impulsively started cylinder [41] [11] [6] [44]
The impulsively started cylinder test case is used to analyze the forces acting on the cylinder. This test case is used to verify and validate the lift and drag evolution of the cylinder exposed to free-stream flow.

Elliptic Airfoil [50]
The elliptic airfoil test case focuses on the flow separation past a lifting body. The elliptic airfoil is pitched at high angle of attack and the flow past the airfoil is comparatively unsteady and undergoes phenomena such as laminar separation bubble, flow separation and karman vortex shedding from the trailing edge of the airfoil. This helps us ensure the coupling strategy is accurate for complex flow phenomena.

1.4 Thesis Outline

Ch. 2 Hybrid Eulerian-Lagrangian Vortex Particle Method
The chapter introduces the hybrid Eulerian-Lagrangian vortex particle method. The chapter introduces the concept of domain decomposition and gives a summary of the coupling strategy developed by Daeninck [24] and the Lagrangian correction algorithm developed by Stock [61]. The chapter concludes which an overview of the algorithm for time stepping the hybrid method.

Ch. 3 Lagrangian Subdomain: Vortex Particle Method
The chapter introduces Vortex Particle Method used to solve the Lagrangian subdomain of the hybrid method. The chapters introduces the concept of vortex blobs for discretizing the vorticity in the fluid and vortex panels for discretizing the wall-bounded vorticity. The chapter concludes with a verification and validation of the Lagrangian method with a Lamb-Oseen vortex test case.

Ch. 4 Eulerian Subdomain: Finite Element Method
The chapter introduces Finite Element Method used to solve the Eulerian subdomain of the hybrid method. We investigate the methodology for solving the incompressible Navier-Stokes equations using the finite element approach. The chapter concludes with a verification and validation of the Eulerian method with the Lamb-Oseen vortex test case, impulsively started cylinder test case at $Re = 550$ if Koumoutsakos and Leonard [41], and the Clercx-Bruneau dipole collision of Clercx and Bruneau [16].

Ch. 5 Coupling Eulerian and Lagrangian Method
The chapter gives an indepth analysis on the coupling strategy of the Eulerian and the Lagrangian method. This chapter highlights the modification we implemented to the coupling strategy used by Daeninck [24] and Stock [61].

Ch. 6 Introduction to the Hybrid Solver: pHyFlow
The chapter introduces `pHyFlow`, the python based hybrid solver. The chapter gives an overview of the program structure and the class hierarchy.

Ch. 7 Verification and Validation of the Hybrid Method
The chapter focuses on the verification and the validation of the hybrid method. We use the test cases described in section 1.3, to achieve this.

Chapter 2

Hybrid Eulerian-Lagrangian Vortex Particle Method

2.1 Introduction to Hybrid Eulerian-Lagrangian Vortex Particle Method

The Hybrid Eulerian-Lagrangian Vortex Particle Method (HELVPM) is a domain decomposition method, where the Eulerian method and the Lagrangian method solve different regions of the fluid. The domain decomposition method simply splits the domain of interest and uses the appropriate method in each domain. The Eulerian formulation will be used at the near-wall region, where we need proper description of the vorticity generation at the boundary, and the Lagrangian formulation is used away from the body, where we only need to evolve the vorticity field. Figure 2.1 shows the decomposition of the domain into the gridded and the non-gridded region.

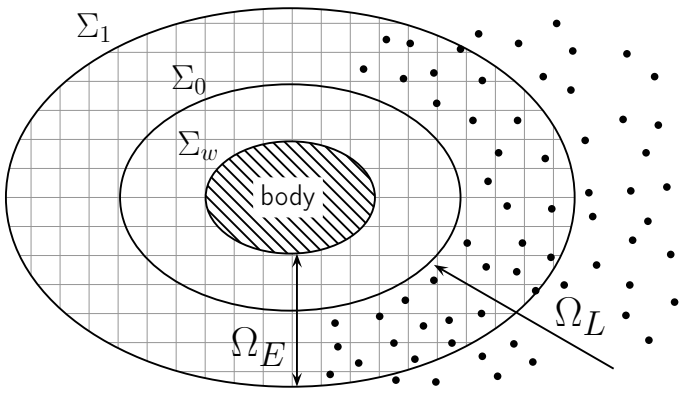


Figure 2.1: Standard domain decomposition using Schwartz iteration for coupling the two methods. Eulerian subdomain Ω_E (near the body), and Lagrangian subdomain Ω_L (away from the body). Figure is based on Guermond (2000) [31].

Several studies have already been performed regarding the domain decomposition. Cottet and Koumoutsakos (2000a)[22] summarized the theory behind the Eulerian-Lagrangian domain decomposition. Guermond and Lu (2000) [31] simulated the advection dominated, external flows. They used Schwartz type method to couple the two methods together. Ould-Salhi et al. (2001) [51] blended the finite difference and vortex method together to construct the hybrid method. Winckelmans et al. (2005a) [68] investigated the trailing vorticies. Daeninck (2006) [24] used a simplified coupling strategy without the Schwartz method, to couple Vortex Particle Method and Finite Difference Method together. Stock (2010) [61] expanded Daeninck's strategy, coupling Vortex Particle Method and Finite Volume Method and modeled a 3D rotor.

2.2 Convectional Coupling Strategy

When investigating the literature works, we see that not all domain decomposition methods are the same. The main difference between the methods is their coupling strategies. Most works employ the *Schwartz alternating method* to couple the vortex particle method and the grid solver. The Schwartz alternating method (or sometimes referred to as Schwartz iterative method), couples the vortex particle method and the grid solver by iteratively determining the boundary condition such that the stream functions in both domains, ψ_L and ψ_E in the Lagrangian domain Ω_L and the Eulerian domain Ω_E respectively, match at the overlap region $\Omega_E \cap \Omega_L$, shown in Figure 2.1. The summary of a single iteration of the Schwartz alternating method is as follows:

- Determine the Eulerian boundary condition, the stream function ψ_{Σ_1} at the Eulerian boundary Σ_1, extracted from the Lagrangian stream function ψ_L in the Lagrangian subdomain Ω_L.
- Solve for the stream function ψ_E in the Eulerian subdomain Ω_E with the new boundary condition Σ_1.
- Determine the Lagrangian condition, the stream function ψ_{Σ_0} at the Lagrangian boundary Σ_0, extracted from the Eulerian stream function ψ_E in the Eulerian subdomain Ω_E.
- Solve the stream function ψ_L in the Lagrangian subdomain with the boundary conditions ψ_{Σ_0} at the Lagrangian boundary Σ_0.

This procedure is iterated until the stream functions of both domains converge [51]. Once the stream function is determined in both the domains, the velocity field can be obtained. Using the velocity field, we can then evolve the vorticity field in the Lagrangian subdomain.

2.3 Simplified Coupling Strategy

As we realized now, the downside to this procedure is that we have to solve the stream function in both Ω_E and Ω_L iteratively, until we converge to a solution. This makes the

computation very expensive, especially when we are dealing with large numbers of vortex particles. Therefore for this project, we decided to use the coupling technique formulated by Daeninck (2006) [24] and expanded by Stock (2010) [61]. However, through the course of the present work, we will see that we have to perform a modification to the scheme, to ensure that the total circulation of the Lagrangian domain is conserved at all times (see section 5.1).

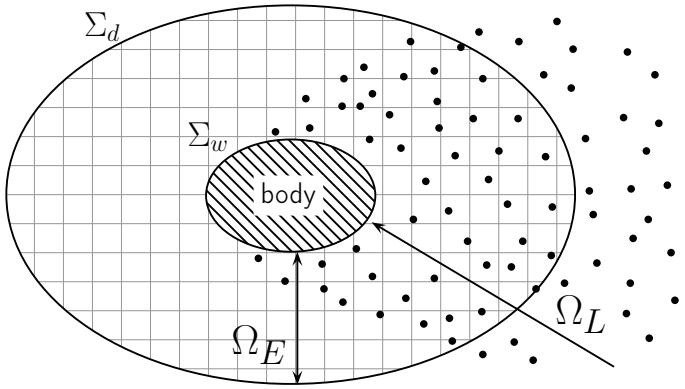


Figure 2.2: Modified domain decomposition without Schwartz alternating method. Lagrangian subdomain extends up to the surface of the body. Figure is based on Daeninck (2006) [24].

The simplified coupling strategy was first demonstrated in the doctoral thesis of Daeninck [24]. Daeninck showed that it is possible to couple the Lagrangian and the Eulerian methods without the use of Schwartz iterative method. Daeninck proposed an approach that satisfies the following statements:

- The Lagrangian vortex method solves the full fluid domain Ω_L (see Figure 2.2), but under-resolves the near-wall region Ω_E as it is less efficient at resolving the boundary layer of the flow.
- Eulerian method is used to resolve the near-wall region Ω_E, efficiently capturing the boundary layer features and flow separation.
- The Lagrangian subdomain in the near-wall region $\Omega_L \cap \Omega_E$ is corrected using the more accurate Eulerian solutions to compensate the aforementioned under-resolution.
- The boundary conditions for the Eulerian method is directly obtained from the evolved solution of the Lagrangian method.

The grid solver therefore essentially acts as the correction for the under-resolved regions of the Lagrangian method. The Lagrangian vortex method in the full fluid domain focuses only on capturing and efficiently evolving the wake.

Furthermore, the Lagrangian boundary condition is handled differently. Unlike the standard Lagrangian method, where the boundary sheds vorticity, in this strategy as the Lagrangian method is under-resolved at the boundary, it cannot be used to resolve the

vorticity flux at the body. Instead, the Eulerian method is used to solve vorticity generation from the wall boundary, and acts as the vorticity generator for the Lagrangian method. For this coupling strategy to be valid, there are some assumptions that we must satisfy:

- At t_n before the evolution of both methods to t_{n+1}, the Lagrangian solution matches the Eulerian solution at the boundary of the near-wall region Σ_d (see Figure 2.2).
- Even though the Lagrangian subdomain is under-resolved in the near-wall region, it should still be able to provide accurate boundary conditions for the Eulerian method at the external boundary Σ_d.
- After the evolution to t_{n+1}, the deviation of the Lagrangian solution (due to lack of vorticity flux at Lagrangian boundary), should be minimal.

Daeninck's simplified coupling strategy focused on the vorticity-velocity formulation for the Eulerian domain. However, he briefly showed that it is also possible to couple the Eulerian method with the velocity-pressure formulation. The advantage of using the velocity-pressure formulation is that it will be easier to extend to a 3D problem, unlike the vorticity-velocity formulation for the Eulerian method.

2.3.1 Coupling Algorithm

The coupling of the solvers was described in one global time stepping algorithm. As the Eulerian methods suffers from a larger stability constraint on the time step, and the Lagrangian time marching is computationally more expensive, a different time discretization for both methods was employed. The Lagrangian method and the Eulerian method had the time steps Δt_L and $\Delta t_E = \Delta t_L / k_E$, respectively, where k_E is the number of Eulerian substeps.

Assuming that we known the solutions of both solvers at t_n, the algorithm for the coupled time marching of the Eulerian method (with velocity-pressure formulation) and the Lagrangian method from t_n to $t_n + \Delta t_L$ is summarized as follows:

1. At t_n, **correct the Lagrangian solution** in the near-wall region $\Omega_L \cap \Omega_E$ from the Eulerian field, Figure 2.2. The vorticity in Ω_E is determined by taking the curl of the velocity field of the Eulerian method. The vortex particles strengths are determined by interpolating the vorticity from the Eulerian grid.
2. **Advance the Lagrangian method** from t_n to $t_n + \Delta t_L$, with the corrected Lagrangian solution. However, after the correction there exists a slip velocity at the solid wall Σ_w. Therefore to properly evolve the solution, the Lagrangian method needs to enforce the *no-slip* boundary condition at the wall. This is performed by computing the vortex sheet γ that cancels this slip velocity at the wall. At the end of the evolution, classic vortex methods diffuse the computed vortex sheet to the particles but in hybrid case, it is performed by the Eulerian method.

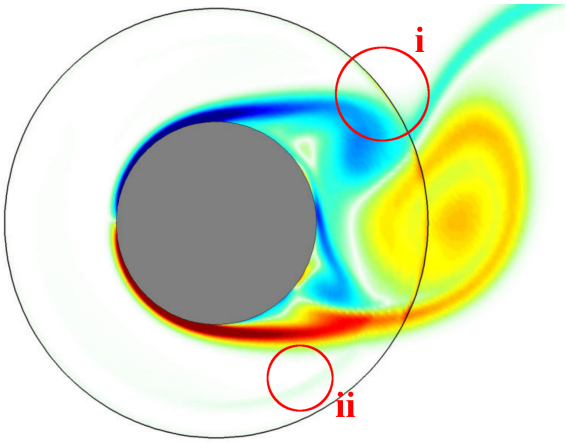


Figure 2.3: Error at the transition from the Eulerian and the Lagrangian domain, obtained from Daeninck [24]. The error is in the form of artificial vorticity at **i)** the Dirichlet boundary of the Eulerian domain Σ_d, and **ii)** the boundary of the Eulerian adjustment region shown in Figure 2.4.

3. **Determine the Eulerian boundary conditions** for the velocity field $\mathbf{u}$ at $t_n + \Delta t_L$ from the evolved Lagrangian solution at $t_n + \Delta t_L$. The Eulerian method requires the Dirichlet velocity boundary condition at Σ_d (the Eulerian Dirichlet velocity boundary) and the velocity boundary condition at the wall boundary Σ_w for a velocity-pressure formulation is simply the zero slip velocity, Figure 2.2.

4. **Advance the Eulerian method** from t_n to $t_n + \Delta t_L$ using k_E Eulerian substeps. The boundary conditions on $\mathbf{u}$ at each substep is obtained by linear interpolation of the boundary condition at t_n and $t_n + \Delta t_L$.

Daeninck notices that when using the velocity-pressure formulation for the Eulerian method, the transition from the Eulerian to the Lagrangian domain is more noticeable. Figure 2.3, shows the artificial vorticity at the boundary of the Eulerian domain Σ_d. This error is more prominent as the coupling is performed at the level of velocity, and we are

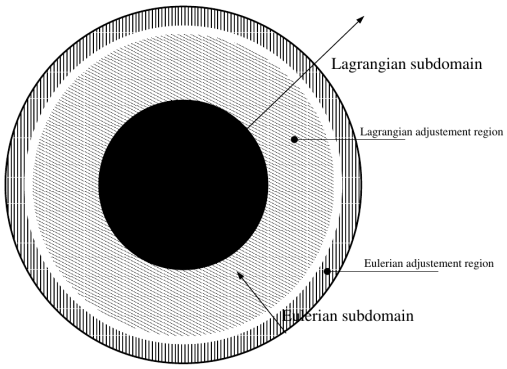


Figure 2.4: The domain decomposition and interpolation regions used by Daeninck [24]. The outer Eulerian domain is also modified to enhance the coupling of the methods.

looking at the curl of it, vorticity. To enhance the coupling of the Eulerian and the Lagrangian method, Daeninck further modified the Eulerian solution in the most external region of the Eulerian subdomain Ω_E interpolating the Lagrangian solution, and observed that it provided better results. Figure 2.4 shows the modified adjustments regions used by Daeninck in his work.

However, in the present work we will use the Lagrangian correction strategy by Stock [61] and introduce additional modifications to the Lagrangian correction step itself to reduce this mismatch.

2.3.2 Lagrangian Correction Step

The coupling strategy demonstrated by Daeninck [24], was studied and was further extended by Stock [61]. Stock's work focused on the overlap region $\Omega_E \cap \Omega_L$ (Figure 2.2) and correction of the Lagrangian solution. The following observations can be extracted from Stock's work:

- Eulerian solution is only assumed to be correct from the body surface Σ_w to somewhat inside of the outer Eulerian domain Σ_d. Therefore, the transfer of the Eulerian solution to the Lagrangian method should take in account the potential inaccuracy of the Eulerian solution at the outer boundary (depicted in the results of Stock, Figure 2.5).

- The very strong gradient in vorticity (vortex sheet) cannot be efficiently and accurately transfered to the Lagrangian method. This is especially problematic at high Reynolds number flows, and interpolating this vorticity from the Eulerian method to the Lagrangian method results in numerical problems. Therefore, to avoid the noise in the interpolation, the correction step has to ignore the region very near to the wall.

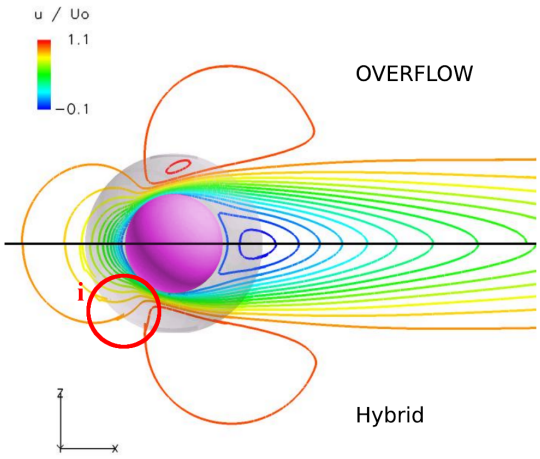
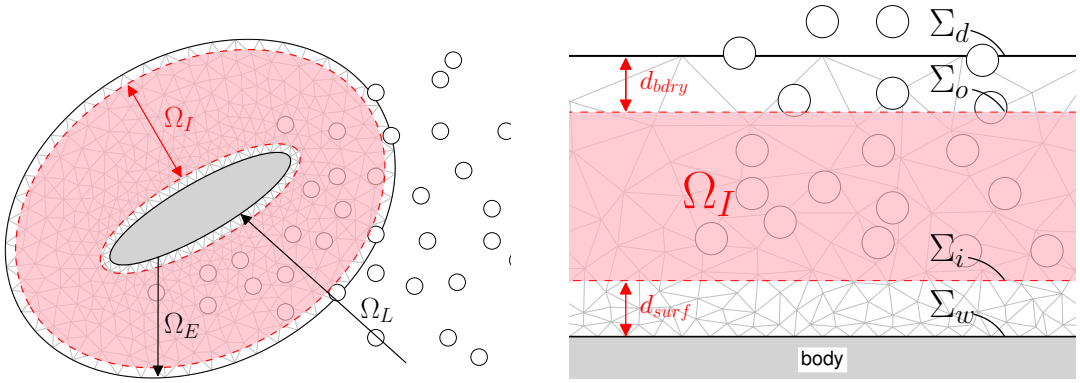


Figure 2.5: Mismatch in the Eulerian and the Lagrangian solution at the boundary Σ_d. The error is illustrated as **i)** slight mismatch in the velocity contours at the boundary.



(a) Definition of the Domains **(b)** Definition of the boundaries

Figure 2.6: Definition of the interpolation domain Ω_{int} for correcting the Lagrangian solution, with boundaries $\Omega_I : \partial\Omega_I = \Sigma_i \cup \Sigma_o$.

The resulting Lagrangian correction domain, or the interpolation domain Ω_I, using Stock's coupling approach, is shown in Figure 2.6. If the Eulerian domain Ω_E is defined by $\partial\Omega_E = \Sigma_w \cup \Sigma_d$, then the interpolation region Ω_I is defined by $\partial\Omega_I = \Sigma_i \cup \Sigma_o$. The outer boundary of the interpolation domain Σ_o is defined with an offset d_{bdry} from the Eulerian Dirichlet velocity boundary Σ_d, such that potential inaccuracy of the Eulerian solution is ignored (depicted in Figure 2.6b). Similarly, the inner boundary of the interpolation domain Σ_i is defined with an offset d_{surf} from the Eulerian wall boundary Σ_w such that the very strong vorticity (i.e the vortex sheet) is ignored (depicted in Figure 2.6b). The offsets d_{surf} and d_{bdry} were defined in the order of the Lagrangian vortex particle size.

The resulting Lagrangian correction step employed by Stock is summarized as follows:

1. Interpolate the vorticity of the Eulerian method from a non-uniformly structured (or an unstructured grid) onto a temporary uniformly structured Cartesian grid covering the entire Eulerian domain Ω_E. This is done to performed an easier transfer of the Eulerian solution to the Lagrangian domain. The interpolation ignores the very strong vorticity present in the boundary layer that would otherwise cause numerical problem.

2. Identify all the particles that are within the interpolation domain Ω_I. These particles will be ones that is corrected.

3. Correct or reset the strengths of the particles using the local particle area and the vorticity interpolated from the temporary structured Cartesian grid.

Using this approach, Stock demonstrated the feasibility of simulating a 3D compressible flow problem around a sphere at $Re = 100$, a finite airfoil at $Re = 1.5 \times 10^6$, and 4-Bladed advancing rotor at $Re = 865,500$.

2.4 Evolution of the Hybrid Method

In the present work, we will therefore employ Daeninck's simplified coupling strategy with the detailed Lagrangian correction approach of Stock. The algorithm for the evolution of the hybrid method is divided into four key components:

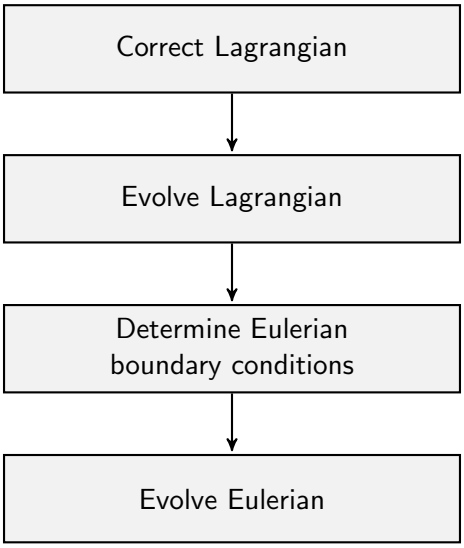


Figure 2.7: Flowchart of the simple coupling strategy. The flowchart shows the procedure to evolve both methods from t_n to t_{n+1}.

1. **Correct Lagrangian:** Use the solution of the Eulerian subdomain Ω_E, to correct the solution of the Lagrangian subdomain Ω_L, using the strategy of Stock (summarized in section 2.3.2). Chapter 5 provides a detailed analysis on the implementation of Stock's Lagrangian correction strategy. Furthermore, the chapter highlights the limitations of Stock's correction strategy and described the methodology to ensure conservation of total circulation, paramount for a valid numerical scheme.

2. **Evolve Lagrangian:** With the modified solution, evolve the Lagrangian solution from time step t_n to $t_n + \Delta t_L$. Chapter 3 provides the detailed investigation on the theory and the algorithm used to evolve the Lagrangian solution.

3. **Determine Eulerian boundary conditions:** Use the Lagrangian solution of time $t_n + \Delta t_L$ to determine the boundary conditions of the Eulerian subdomain at $t_n + \Delta t_L$. Chapter 5 describes the methodology used to determine the boundary conditions at each Eulerian substeps.

4. **Evolve Eulerian:** With the boundary condition, evolve the Eulerian solution from t_n to $t_n + \Delta t_L$ using k_E Eulerian substeps. Chapter 4 provides the detailed investigation on the theory and the algorithm of the Eulerian method used for the present work.

Figure 2.7 shows the flowchart of the evolution of the hybrid method. To ensure that the coupling of the hybrid method performs as explained in theory, we performed a verification and validation test on the functionality of each segregate methods.

Chapter 3

Lagrangian Method: Vortex Particle Method

In this chapter, we introduce the vortex particle method used to solve the Lagrangian subdomain of the hybrid method, and summarize the process for evolving the Lagrangian method, i.e. step 2 of the hybrid evolution, as shown in Figure 3.1.

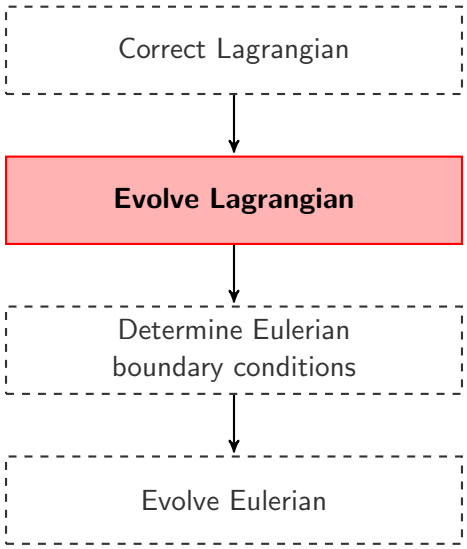


Figure 3.1: Flowchart of the hybrid evolution, focusing on the 2^{nd} step: Evolve the Lagrangian solution.

3.1 Introduction to the Vortex Particle Method

Vortex Particle Method (VPM) is a numerical method employed in computational fluid dynamics, dealing with the evolution of the vorticity in the fluid from a Lagrangian

description. In an Eulerian formulation, the fluid is viewed at a fixed window where the change in the fluid properties are evaluated. However, the Lagrangian formulation, regards the fluid as a collection of particles (or elements) carrying properties of the fluid (such as vorticity, mass, etc.).

Efficient discretization of the fluid domain becomes a difficult task for cases such as Vertical-Axis Wind Turbine (VAWT), where the wake geometry is complex and unsteady. Discretizing such wake using Eulerian formulation becomes difficult as it requires the adaption of the mesh over time for efficient computation. The VPM only needs fluid elements where there is vorticity meaning that the method is inherently auto-adaptive. This is one of the advantage of the VPM. Furthermore, with computational acceleration methods such as Fast-Multipole Method (FMM) and parallel computation in Graphics Processing Units (GPU) enables an efficient evolution of the vorticity wake.

However, the key advantage of the VPM is that it is ideal for capturing the resolving the long-time characteristics of the unsteady compact vortical structures that are shed off from the VAWT blades, as described by Stock [61], providing the motivation for using VPM for modeling the rotor wake.

A summary of the advantage of the Lagrangian vortex method w.r.t the Eulerian method was provided by Wee and Ghoniem [67]:

- Eulerian methods introduce dissipation, even in flows with zero velocity gradient. However such error is minimized when using a Lagrangian method.
- The numerical stability of the Lagrangian method is not restricted by the CFL condition.
- The effective support of the Lagrangian elements are a small fraction of the fluid domain. The support is confined to location of non-zero vorticity, making the method naturally grid adaptive.

The main literature on the VPM (the Lagrangian component of the hybrid method), is the book of Cottet and Koumoutsakos, Vortex Methods: Theory and Practice [22]. It gives an insight on the fundamentals of the vortex method (specifically VPM) and gives a summary on hybrid methods.

3.1.1 Vorticity

The vorticity ω is the governing element of the VPM. It is given by

$$\omega = \nabla \times \mathbf{u}, \tag{3.1}$$

where $\mathbf{u}$ is the velocity vector field. In 2D, the circulation Γ is defined by Stokes' theorem as,

$$\Gamma = \int_L \mathbf{u} \cdot \mathrm{d}s = \int_A (\nabla \times \mathbf{u}) \cdot \mathbf{n} \, \mathrm{d}A = \int_A \omega \cdot \mathbf{n} \, \mathrm{d}A, \tag{3.2}$$

and represents the flux integral of vorticity across the surface A, contoured by the line s. Figure 3.2 depicts this relation of velocity $\mathbf{u}$, vorticity ω and the circulation Γ.

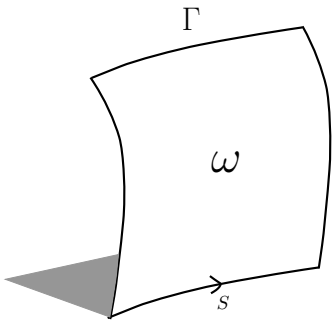


Figure 3.2: Definition of the circulation in the fluid.

3.1.2 Velocity-Vorticity Formulation of the Navier-Stokes Equations

The governing equation of the vortex particle method is the velocity-vorticity formulation $\mathbf{u} - \omega$ of the Navier-Stokes equations, as presented in Cottet and Koumoutsakos [22]. It is derived from the 2D incompressible Navier-Stokes momentum equation, given as,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u}, \tag{3.3}$$

relating the velocity field $\mathbf{u}(\mathbf{x}, t)$ to the pressure field $p(\mathbf{x}, t)$, the kinematic viscosity ν and density ρ, and satisfied the incompressibility constraint,

$$\nabla \cdot \mathbf{u} = 0, \tag{3.4}$$

The curl of the equation 3.3 is taken to obtain the velocity-vorticity formulation,

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \nabla^2 \omega, \tag{3.5}$$

giving the time evolution of the vorticity. Note that the pressure term p disappear from the equality.

3.1.3 Viscous Splitting Algorithm

The VPM was initially used to model the evolution of incompressible, inviscid flows. However, in order to simulate a real flow, we must also deal with the viscous behavior of the fluid. Chorin in 1973 [14], showed that using the viscous splitting algorithm, it is possible to take the viscous effects of the flow into account.

The viscous splitting algorithm is a fractional step method, where the viscous and the inviscid part of the vorticity transport equation, equation 3.5, are dealt with in two subsequent substeps,

1. **Convection** (substep 1):

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = 0; \tag{3.6}$$

2. **Diffusion** (substep 2):

$$\frac{\partial \omega}{\partial t} = \nu \nabla^2 \omega. \tag{3.7}$$

The first substep of the evolution deals with the convection of vorticity. The convection step is described in section 3.3. The diffusion of vorticity field is dealt with in the second substep. The diffusion of the vorticity field is dealt with in section 3.4

3.2 Spatial Discretization: Introduction to Vortex Blobs

The vorticity ω and the velocity $\mathbf{u}$ in the vorticity transport equation, equation 3.5, describes a continuous field. However, these variables needs to be discretized to perform a numerical integration required for the numerical simulation.

3.2.1 Discrete Form of Vorticity Field

The vorticity field is discretized by representing the continuous vorticity field by a summation of particle-type elements, as described by Barba [1]. The discrete vorticity field ω^h is represented by a linear combination of N basis function, given as,

$$\omega(\mathbf{x}, t) \simeq \omega^h(\mathbf{x}, t) = \sum_p^N \alpha_p(t)\, \delta[\mathbf{x} - \mathbf{x}_p(t)], \tag{3.8}$$

where α_p is the circulation carried by the particle at $\mathbf{x}_p$. We must note that ω^h is the discrete vorticity field and therefore in an approximation of the continuous vorticity field ω. The velocity $\mathbf{u}$ is related to the vorticity ω using the Biot-Savart Law.

3.2.2 Biot-Savart Law

A velocity field $\mathbf{u}$ that satisfies the incompressibility constraint, equation 3.4, can be decomposed using the Helmholtz decomposition,

$$\mathbf{u} = \mathbf{u}_\omega + \mathbf{u}_\phi + \mathbf{u}_\delta, \tag{3.9}$$

where $\mathbf{u}_\omega$ is the rotational (solenoidal) component of the velocity $\mathbf{u}$, $\mathbf{u}_\phi$ is the irrotational (potential) component, and $\mathbf{u}_\delta$ is the harmonic form. These components satisfying the following equality,

$$\nabla \cdot \mathbf{u}_\omega = \nabla \times \mathbf{u}_\phi = \nabla^2 \mathbf{u}_\delta = 0. \tag{3.10}$$

The divergence-free component $\mathbf{u}_\omega$ implies that there exists a stream function ψ, such that,

$$\mathbf{u}_\omega = \nabla \times \psi, \tag{3.11}$$

and therefore the vorticity ω of the velocity $\mathbf{u}$, is given as,

$$\omega = \nabla \times \mathbf{u} = -\Delta \psi \tag{3.12}$$

Similarly, there must exist a potential ϕ, such that,

$$\mathbf{u}_\phi = \nabla\phi. \tag{3.13}$$

For an incompressible and unbounded problem, the potential velocity $\mathbf{u}_\phi$ is simply the free-stream velocity $\mathbf{u}_\infty$ and the harmonic form $\mathbf{u}_b = 0$. In the case of the bounded problem with solid boundaries, the presence of the body must be taken into account, see section 3.5. For now, we will discuss the unbounded problem.

From the Poisson equation 3.12, the velocity is related to the vorticity by the Biot-Savart law, given as,

$$\mathbf{u}_\omega = \mathbf{K} \star \omega, \tag{3.14}$$

where $\star$ is the convolution of the vorticity with the 2-D Biot-Savart kernel $\mathbf{K}$ given by,

$$\mathbf{K} = \frac{1}{2\pi \left|\mathbf{x}\right|^2} \left(-x_2, x_1\right). \tag{3.15}$$

From the kernel, we see that as the distance to the kernel center approaches zero ($\mathbf{x} \to 0$), the kernel goes to infinity. The singularity of the kernel $\mathbf{K}$ is removed by mollifying the kernel distribution ensuring smooth velocity distribution.

3.2.3 Mollified Vortex Kernels

A mollified (or a regularized) vortex particle is called the vortex blob, and has a non-zero vortex core size σ. A smoothing function ζ is used to mollify the kernel $\mathbf{K}$, satisfying the constraint $\int \zeta = 1$, such that the circulation is conserved. An ideal choice for a smoothing function is the Gaussian distribution, given as,

$$\zeta_\sigma = \frac{1}{k\pi\sigma^2} \exp\left(-\frac{|\mathbf{x}|}{k\sigma^2}\right), \tag{3.16}$$

where typically k is either 1, 2 or 4, determining the width of the kernel, and σ being the core-size of the vortex blob.

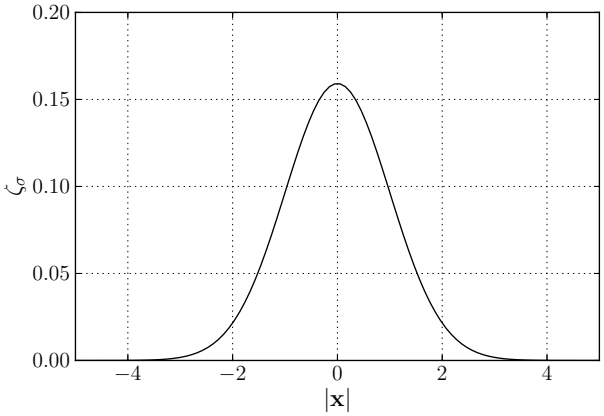


Figure 3.3: The smoothing function ζ_σ for a gaussian distribution with $k = 2$, $\sigma = 1$.

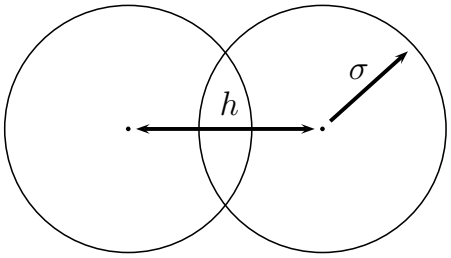


Figure 3.4: Vortex blob with an overlap ratio $\lambda = h/\sigma$

Figure 3.3 depicts the smoothing function ζ_σ with $k = 2$ and $\sigma = 1$, showing that the function decays quickly away from the center of the core. The mollified Biot-Savart kernel $\mathbf{K}_\sigma$ is given as,

$$\mathbf{K}_\sigma = \mathbf{K} \star \zeta_\sigma, \tag{3.17}$$

resulting in the discrete mollified vorticity field as,

$$\omega^h(\mathbf{x}, t) = \sum_p \alpha_p(t)\, \zeta_\sigma\left[\mathbf{x} - \mathbf{x}_p(t)\right], \tag{3.18}$$

and the discrete mollified velocity field as,

$$\mathbf{u}^h(\mathbf{x}, t) = \sum_p \mathbf{K}_\sigma\left[\mathbf{x} - \mathbf{x}_p(t)\right] \alpha_p(t). \tag{3.19}$$

Koumoutsakos and Chorin [22], explained that in order to ensure the smoothness of the velocity field, the vortex blobs need to have an overlap with each other. The overlap ratio λ is defined as,

$$\lambda = \frac{h}{\sigma}, \tag{3.20}$$

where h is the nominal particle spacing, and σ is the vortex blob core size. Figure 3.4 shows the visual representation of this definition.

The overlap constraint is violated during the convection of the vortex blobs. Due to the strains in the flow, the vortex blobs cluster together at certain regions and disperse at others. This localized clustering effect is seen as a Lagrangian grid distortion, which is dealt with using a remeshing technique, section 3.3.1.

3.2.4 Vortex Blob Initialization

Now the question arises on how should we initialize the particle's circulation strengths α_p of equation 3.18. The common approach, that is used as a standard, is to estimate the particles strength by quadrature,

$$\alpha_p = \omega_p \cdot h^2, \tag{3.21}$$

meaning that the particle carries the circulation of its local area. This might seem like a valid assumption as the circulation of a given area is the integral of the vorticity in the

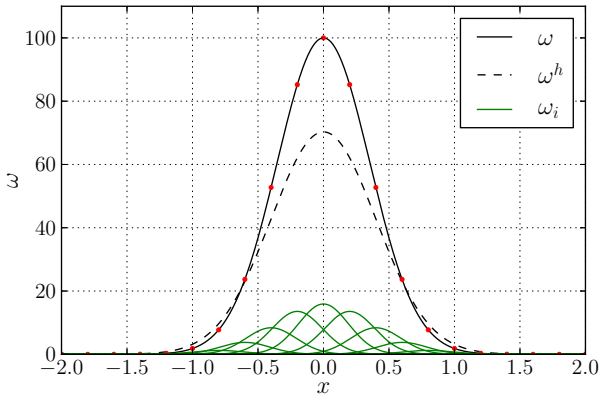


Figure 3.5: Mollified vorticity field of a Gaussian vorticity distribution by blobs with $\lambda = 1.0$, $\sigma = 0.19$, and $h = 0.19$. Vortex blob strengths were assigned using equation 3.21, sampling the exact vorticity [•, **red** dot]. Figure depicts the exact vorticity distribution ω [—, solid **black**], the vorticity distribution of each blob ω_i [—, solid **green**], and the mollified vorticity field from the blobs ω^h [- -, dashed **black**].

area, given by equation 3.2, and therefore we will be conserving the circulation as all the circulation in the fluid is represented by the blobs.

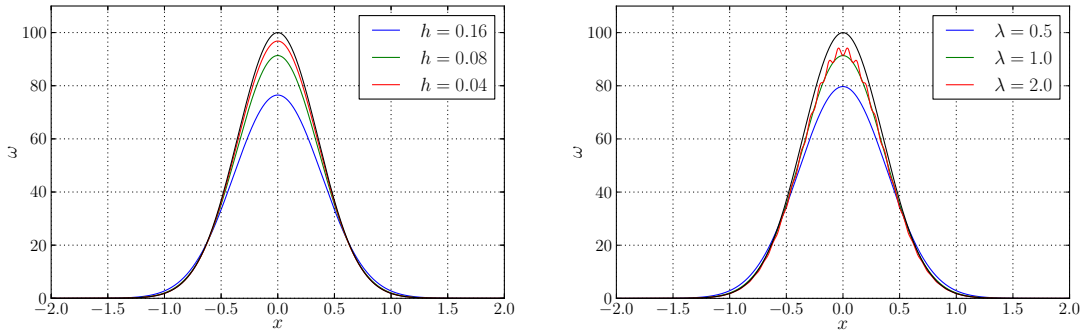
Figure 3.5 shows the initialization of the vorticity field using the equation 3.21. We observed that the mollified vorticity field ω^h, deviates from the original intended vorticity distribution ω. Barba and Rossi [1], have described this problem as Gaussian blurring of the original vorticity field due to use of mollified vortex kernel ζ_σ in equation 3.18. Even though the total circulation is conserved, locally we see that the circulation is not conserved.

This phenomenon causes issues during the coupling of the Lagrangian method and the Eulerian method, as particles in the overlap region are re-initialized at every step of the hybrid method. This error in initialization poses a challenge in the coupling of the method and is described in section 5.1.1.

A typical strategy for recovering the intended distribution is the Beale's Iterative Method [3], as used by Koumoutsakos and Cottet [22]. The method is a particle circulation processing scheme where the circulation α_p of the particles are modified iteratively such that the mollified vorticity field ω^h matches the original distribution ω. However, the Beale's method required the correction of the complete vorticity field in the fluid domain and cannot be used to correct only part of the fluid domain, as required for our decomposed domain method. Therefore, an alternate method is required to minimized the error in the particle initialization.

3.2.5 Minimization of Particle Discretization Error

An alternate method to reduce the Gaussian blurring of the vorticity field is to modify the overlap ratio λ, and the nominal particle spacing h. This approach does not remove the Gaussian blurring problem, but instead minimizes its discretization error.



(a) Convergence of h with $\lambda = 1.0$ **(b)** Convergence of λ with $h = 0.08$

Figure 3.6: Convergence of the spatial discretization h and λ of the initial vorticity distribution. Figure depicts the exact vorticity field ω [—, solid **black**], and various discretized vorticity distributions.

Figure 3.6 shows mollified vorticity field results from modifying the spatial resolution parameters. Figure 3.6a shows the convergence of the mollified vorticity field ω^h to the exact vorticity field ω by reducing the nominal particle spacing h. The overlap ratio is set to overlap $= 1$, meaning that the blob core-size σ is equal to h. We see that by reducing blob core size, and simultaneously increasing the number of particles, the mollified vorticity converges to the exact vorticity.

The second parameter we can adjust is the overlap of the blobs, as seen in Figure 3.6b. The blob spacing h is set to $h = 0.08$, and we see that by increasing the overlap ratio λ, the mollified vorticity approaches the exact vorticity field. However, as shown by Koumoutsakos and Cottett [22], if the overlap is low, we lose the smooth reconstruction of the vorticity field. This can be seen for $\lambda = 2.0$. We see that the mollified vorticity field has a fluctuating solution, and will result in a non-smooth velocity field.

Thus, to minimize the error in initializing the mollified vorticity ω^h from the vorticity ω, an overlap ratio of $\lambda = 1.0$, and a small nominal blob spacing h is required. In our hybrid method, this means that at the region where we initialize the vortex blobs (i.e inside the Eulerian subdomain), we require these conditions to be satisfied.

3.3 Convection in Vortex Particle Method

Convection of the vorticity is the first step of evolution of the vorticity from viscous splitting algorithm, from section 3.1.3. The convection of the vorticity is described by the first order hyperbolic equation, equation 3.6. The convection equation 3.6, is solved by the following system of ordinary differential equations,

$$\frac{\mathrm{d}\mathbf{x}_p}{\mathrm{d}t} = \mathbf{u}\left(\mathbf{x}_p\right), \tag{3.22a}$$

$$\frac{\mathrm{d}\alpha_p}{\mathrm{d}t} = 0, \tag{3.22b}$$

where the change in position of vortex blob $\mathbf{x}_p$ is due to the induction velocity $\mathbf{u}\left(\mathbf{x}_p\right)$ acting on it, and the strengths of the particles α_p are conserved.

The Biot-Savart Law, equation 3.19, is used to determine the induced velocities acting on each particle, resulting in an N-body problem. The calculation of the N-body problem is optimized by parallelizing the calculations in GPU hardware. The calculation is further optimized by using a fast summation method, the Fast Multipole Method (FMM), reducing the problem from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ (in the ideal case).

The time integration of equation 3.22 is performed using a 4^{th} order explicit Runge-Kutta method. The higher-order time integration ensures an accurate convection of the vortex blobs, resulting in minimum dissipation. However, the downside to employing a multi-stage method is that we require multiple evaluation of the induced velocity $\mathbf{u}(\mathbf{x}_p)$, when stepping from a given time t_n to the next time t_{n+1}.

After several steps of the convection of the vortex blobs, the overlap ratio λ will no longer be satisfied due to strains in the fluid, as described by Koumoutsakos and Chorin [22]. In section 3.2.5, we determined that to have an accurate reconstruction of the vorticity field, the vortex blobs must satisfy the overlap ratio λ at all times t. This introduced the need for a remeshing (or a regridding) scheme that can reconstruct the vortex blobs distribution to the original overlap ratio λ.

3.3.1 Remeshing Scheme: Treating Lagrangian Grid Distortion

The distortion of the Lagrangian grid is due to the clustering and dispersion of the vortex blobs. This clustering and dispersing effect of the blobs is due to the high strains in the flow. Figure 3.7 depicts the distortion of the Lagrangian grid after 100 convection steps. The final distribution shows gaps in vortex blob distribution and will result in an inaccurate representation of the vorticity field.

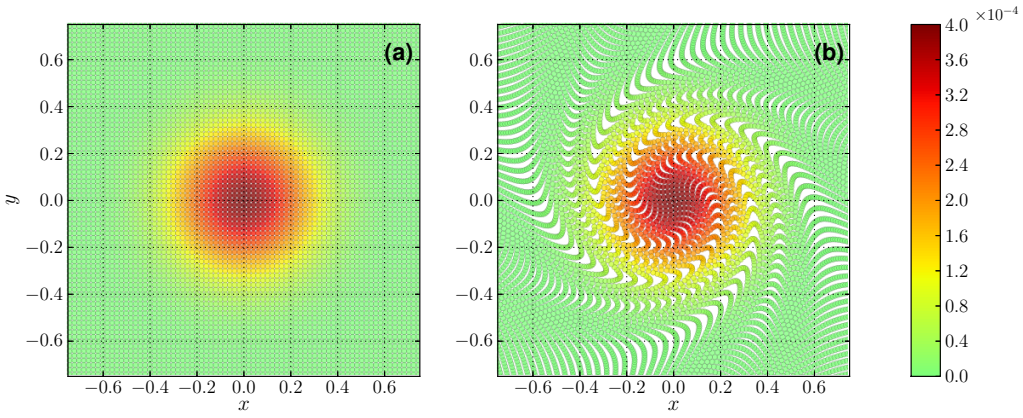


Figure 3.7: Lagrangian distortion of the vortex blobs after 100 time steps. The initial vorticity field is $\omega(\mathbf{x}, 0) = \exp(-12|\mathbf{x}|)$ with $\Delta t = 0.1$, $\sigma = 0.02$, and overlap $= 1.0$. Figure depicts **(a)** the initial distribution of the vortex blobs, and **(b)** the final distribution of the vortex blobs after 100 time steps.

A remeshing (or regridding) of this field is therefore required to retain proper distribution. It is done by interpolating the strengths of the vortex blobs from the distorted Lagrangian grid $\hat{\mathbf{x}}$ onto a uniform grid $\mathbf{x}$. The strengths of the blobs of the new uniform grid α_p is

determined by,

$$\alpha_p = \sum_q \tilde{\alpha}_q W\left(\frac{x_p - \tilde{x}_q}{h}\right), \tag{3.23}$$

where the strengths of the blobs $\tilde{\alpha}_q$ of the distorted Lagrangian grid $\tilde{x}_q$ are transfered to the regular Lagrangian grid x_p using the interpolation kernel W. Figure 3.8 shows an example of the remeshing of one vortex blob of the distorted grid on to the structured uniform grid with a kernel W with a 3×3 stencil. During this transfer, we must ensure that the properties of the fluid are conserved. The interpolation kernel is constructed by ensuring that the total circulation, the linear impulse, and the angular impulse of the fluid are conserved.

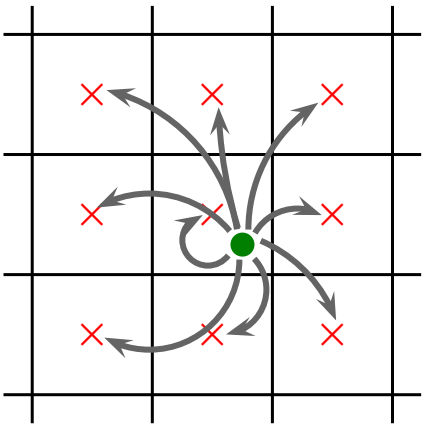


Figure 3.8: Remeshing of a single vortex blob [•, **green** dot] onto a uniform grid defined by the (3×3) 2-D stencil.

For the present work, the widely used M'_4 interpolation kernel, such as by Koumoutsakos and Cottet [22], Speck [60], and Barba [2]. The kernel is continuously differentiable ensuring conservation of total circulation, linear and angular impulse.

M'_4 Interpolation Kernel

The M'_4 interpolation kernel is an efficient interpolation kernel that has been used to reconstruct a smooth distribution, and was introduced by Monaghan in 1985 [48]. For a 1D problem, the M'_4 interpolation kernel is defined as,

$$\mathrm{M'}_4(\xi) = \begin{cases} 1 - \frac{5\xi^2}{2} + \frac{3|\xi|^3}{2} & |\xi| < 1, \\ \frac{1}{2}(2 - |\xi|)^2 (1 - |\xi|) & 1 \leqslant |\xi| < 2, \\ 0 & 2 \leqslant |\xi|, \end{cases} \tag{3.24}$$

where

$$\xi = \frac{x_\nu - x_i}{h}, \tag{3.25}$$

is a non-dimensional parameter, relating the position of the particle x_ν to the position of the i^{th} interpolation node x_i. The M'_4 is a third-order accurate, piecewise smooth, B-spline kernel. The kernel with the $m = 4$, has a 4 support nodes in each dimension.

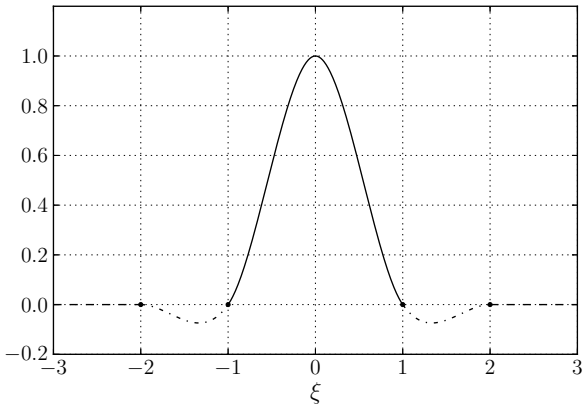


Figure 3.9: M_4' interpolation kernel, a third-order, piecewise smooth, B-spline kernel by Monaghan [48]

Figure 3.9 shows the distribution of the kernel. For the 2D problem, the 2D interpolation kernel is the tensor product of the 1D interpolation kernel, thus having a $4 = 16$ support nodes. The kernel has a compact support, making it ideal for an efficient $\mathcal{O}(N)$ remeshing.

3.4 Diffusion in Vortex Particle Method

Chorin [14], simulated the viscous flow using the viscous splitting algorithm, described in section 3.1.3. The viscous splitting algorithm segregated the vorticity transport equation, equation 3.5, to the inviscid and the viscous components, equation 3.6 and equation 3.7 respectively.

The flow is convected during the first substep, whereas in the second substep, we have to deal with the diffusion of the vorticity field, equation 3.7. The diffusion problem is solved using the following system of ODEs,

$$\frac{\mathrm{d}\mathbf{x}_p}{\mathrm{d}t} = 0, \tag{3.26a}$$

$$\frac{\mathrm{d}\alpha_p}{\mathrm{d}t} = \nu\Delta\alpha_p, \tag{3.26b}$$

where the position of the particles $\mathbf{x}_p$ is fixed, whereas the change in strengths of the particles α_p as depended on the kinematic viscous ν. Thus the diffusion of the vortex blobs requires only the modification to the strengths α_p of the particles.

Chorin in 1973 [14], initially employed the Random Walk Method (RWM), which generates and disperses vorticity using a pseudo-random number algorithm. However, this method suffered some limitations in accuracy, and since then methods such as Particle Strength Exchange (PSE) method [26], and Vortex Redistribution Method (VRM) [55] became popular choices for treating diffusion of the particles.

The VRM simulates the diffusion of the particles by redistributing fractions of circulations of the vortex blobs to each other, such that diffusion is appropriately modeled. These redistribution fractions f_{ij}^n are determined by solving a linear system of equations, that

conserves the moments of the particles (such as the total circulation, linear and angular impulse) and the diffusion of the flow.

The redistribution fractions f_{ij}^n, transfers portion of circulation α_p of the particle p to others within the diffusion radius, defined as,

$$h_\nu = \sqrt{\nu \Delta t_d} \tag{3.27}$$

where h_ν is the diffusion distance and is directly related to the kinematic viscosity ν and the diffusive time step Δt_d of the simulation.

For this work, we investigate two methods that employ this approach. The Wee Remeshing Scheme [67] implemented the VRM into the remeshing process, section 3.4.1. The advantage was that diffusion and remeshing can be performed simultaneously in a single process. However for some flow cases, this approach had an undesirable constraint on the diffusion time step Δt_d. Therefore, the approach of Tutty [64], the Tutty Remeshing Scheme, was employed which had a desirable constraint on the diffusion time step Δt_d, section 3.4.2. The scheme was used to perform diffusion at every step of the evolution, which is important for proper coupling of the Eulerian and the Lagrangian methods.

3.4.1 Wee Remeshing Scheme

Ghoniem and Wee [67], observed the similarities between the VRM and the standard remeshing strategy described in section 3.3.1. They proposed to combine the remeshing and the diffusion of the vortex blobs together in one single process. The application of this methodology was later investigated by Speck [60]. This approach, referred to as the Wee Remeshing Scheme (WRS), implements the diffusion of the vortex blobs in the interpolation kernel W of equation 3.23.

The key advantage of the WRS is that, now we are dealing with a uniform grid, and does not require a search algorithm to find the particles in the zone of influence, equation 3.27. This significantly reduces the computational cost, making this diffusion scheme practical for large scale simulations.

The modified M'_4 kernel for treating the diffusion is given as,

$$\mathrm{M}'_4\left(\xi, c\right) = \begin{cases} 1 - \frac{5\xi^2}{2} + \frac{3|\xi|^3}{2} - c^2\left(2 - 9\xi^2 + 6|\xi|^3\right) & |\xi| < 1, \\ \frac{1}{2}(2 - |\xi|)^2\,(1 - |\xi|) - c^2(2 - |\xi|)^2\,(1 - 2\,|\xi|) & 1 \leqslant |\xi| < 2, \\ 0 & 2 \leqslant |\xi|\,, \end{cases} \tag{3.28}$$

where

$$c^2 = \frac{\nu \Delta t_d}{h^2}, \tag{3.29}$$

is a non-dimensional number that corresponds to the transfer weight for the diffusion. The constant c^2 is a function of the kinematic viscosity ν, diffusion time step Δt_d and the remeshing grid spacing h. This additional term in the interpolation kernel accounts for the diffusion process. When $c \to 0$, the interpolation kernel simply turns into the standard remeshing kernel, equation 3.24.

Ghoniem and Wee also investigated the error growth and the stability properties of the interpolation kernel in the Fourier space and have determined that for a stable diffusion and remeshing, the following constraint has to be satisfied,

$$\frac{1}{6} \leqslant c^2 \leqslant \frac{1}{2}. \tag{3.30}$$

However, we see that this c^2 constraint imposes a direct constraint not only on the maximum diffusion time step Δt_d, but also imposes a constraint on the minimum step size. This would mean that the diffusion time step Δt_d will be sometimes larger than the convection step Δt_c,

$$\Delta t_d = k_d \cdot \Delta t_c. \tag{3.31}$$

where $k_d \geqslant 1$ and is an integer. This is a problem for the hybrid method as the Lagrangian method and the Eulerian method are coupled at every step. If the Lagrangian method does not perform diffusion at every step, from the Eulerian method's point of view, it would seem that the Lagrangian vorticity diffuses in a discontinuous fashion. This discontinuous behavior of the Lagrangian method (w.r.t. the Eulerian method), can cause stability issues during the coupling process, and therefore should be avoided.

We could minimize this problem by modifying the Δt_c such that it matches the diffusion time step (i.e $\Delta t_c = \Delta t_d$), so that the diffusion is performed at every step. This was a feasible solution for low Reynolds number flows, however for high Reynolds number flows, where the convection time step has to be small, we need a scheme that is not constrained by the minimum diffusion time step.

3.4.2 Tutty Remeshing Scheme

The Tutty Remeshing Scheme (TRS), developed by Tutty in 2010 [64], was based on the VRM of Shankar and Van Dommelen [55]. The scheme it possible to remesh and diffuse the vorticity after every convection step. The strengths of the particles after the remeshing and diffusion α_i^{n+1}, are given as,

$$\alpha_i^{n+1} = \sum_k \alpha_k^n W_{ki}^n, \tag{3.32}$$

where W_{ki}^n is the fraction of circulation transferred from vortex blob k (old) to the new vortex blob i (new), Figure 3.8. Tutty [64], explained that the fractions W_{ki}^n are calculated by imposing a conservation of vorticity, center of vorticity, linear, and angular momentum of the vortex blobs, given as,

$$\sum_k W_{ki}^n = 1, \tag{3.33a}$$

$$\sum_k W_{ki}^n (x_i - x_k) = \sum_k W_{ki}^n (y_i - y_k) = 0, \tag{3.33b}$$

$$\sum_k W_{ki}^n (x_i - x_k)^2 = \sum_k W_{ki}^n (y_i - y_k)^2 = 2h_\nu^2, \tag{3.33c}$$

$$\sum_k W_{ki}^n (x_i - x_k)(y_i - y_k) = 0, \tag{3.33d}$$

where h_ν is the characteristic diffusion distance associated to the time Δt_d,

$$h_\nu = \sqrt{\Delta t_d \cdot \nu}. \tag{3.34}$$

Similar to the the TRS, described in section 3.4.1, the TRS transfers the strengths to a known set of new nodes rather than the neighboring blobs (removing the requirement for a search algorithm). Figure 3.10 shows the 1D redistribution of the vortex blob located at $x_i \leqslant x_\nu \leqslant x_{i+1}$ to the stencil nodes x_k, where $k = i-1, \cdots, i+2$. The solution to the redistribution is given by the following equations,

$$f_{i-1} = \frac{1}{2}(1 - f_i - \Delta) \tag{3.35a}$$

$$f_i = 1 - 2\left(\frac{h_\nu}{h}\right)^2 - \Delta^2 \tag{3.35b}$$

$$f_{i+1} = \frac{1}{2}(1 - f_i + \Delta) \tag{3.35c}$$

$$f_{i+2} = 0 \tag{3.35d}$$

and

$$g_{i-1} = 0 \tag{3.36a}$$

$$g_i = \frac{1}{2}(1 - g_{i+1} - \Delta_1) \tag{3.36b}$$

$$g_{i+1} = 1 - 2\left(\frac{h_\nu}{h}\right)^2 - \Delta_1^2 \tag{3.36c}$$

$$g_{i+2} = \frac{1}{2}(1 - g_{i+1} + \Delta_1) \tag{3.36d}$$

where Δ is defined as,

$$\Delta = \frac{x_\nu - x_i}{h}, \tag{3.37}$$

as depicted in Figure 3.10. Note that $\xi_1 = \xi - 1$. The total redistribution fractions F_k, are the linear combinations of the functions f_k and g_k,

$$F_k = (1 - \Delta) \cdot f_k + \Delta \cdot g_k, \quad k = i-1, \ldots, i+2, \tag{3.38}$$

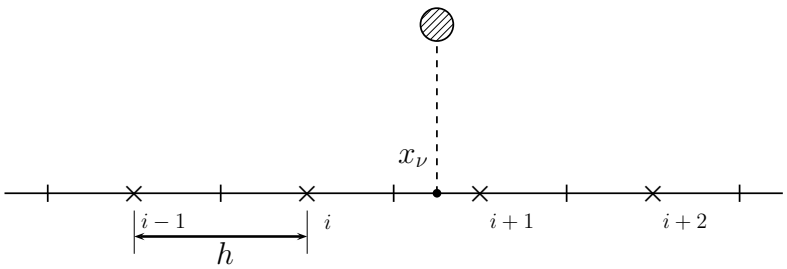


Figure 3.10: 1D Tutty Remeshing Scheme (TRS), diffusing the vortex blobs at $x_i \leqslant x_\nu \leqslant x_{i+1}$, onto the four stencil points $k = i-1, \ldots, i+2$, with a grid spacing h.

For the 2D, the redistribution fractions are simple tensors product of the x and y 1D redistribution fractions,

$$W_{kl} = F_k G_l, \quad k = i-1, \cdots, i+2, \quad l = j-1, \ldots, j+2, \tag{3.39}$$

with a 16 point stencil when $\xi = 1/2$.

The stability of the redistribution requires a positive redistribution fraction, $W_{kl}^n > 0$, imposing a direct constraint on the diffusive distance,

$$\frac{h_\nu}{h} < \frac{1}{\sqrt{2}}. \tag{3.40}$$

as explained by Tutty [64]. A resulting constraint is imposed on the maximum diffusion time step Δt_d,

$$\Delta t_d < \frac{h^2}{2\nu}. \tag{3.41}$$

Therefore, we observe that this scheme only poses a constraint on the maximum diffusion time step Δt_d, enabling us to perform the diffusion at every step of the evolution, equation 3.31.

When employing the Tutty's scheme, we require diffusion and redistribution to be performed at every step, i.e the diffusion frequency $f_{diff} = 1$ and the redistribution frequency $f_{redis} = 1$. In addition to the redistribution, a common approach to minimize the number of vortex blobs is to perform a population control. A population control removes particles with strengths $|\alpha|$ less than a pre-defined circulation threshold Γ_{loc}, and simultaneously ensuring that the total circulation removed is less that a pre-defined global threshold Γ_{glob},

$$\sum_i |\alpha_i| \leqslant \Gamma_{glob}, \tag{3.42}$$

where α_i is the strength of the removed particle i. Typically, the population control is performed in conjunction with the redistribution, i.e $f_{redis} = f_{pc} = 1$.

3.5 Boundary Conditions for Viscous Vortex Particle Method

For incompressible viscous flows, the solid boundary is the vorticity generator of the flow. So far, we have only dealt with unbounded flow. For bounded flow simulation, we must enforce the boundary condition of the flow. In 2D, the boundary condition for an immersed body, translating at velocity $\mathbf{u}_b(t)$ with an angular velocity $\Omega(t)$ about it's center of mass $\mathbf{x}_b$ is given as,

$$\mathbf{u}(\mathbf{x}_s) = \mathbf{u}_s \tag{3.43}$$

with,

$$\mathbf{u}_s = \mathbf{u}_b + \Omega(t) \times (\mathbf{x}_s - \mathbf{x}_b), \tag{3.44}$$

where $\mathbf{u}_s$ is the velocity at the surface point $\mathbf{x}_s$. However, in the present work, we deal with stationary bodies and so $\mathbf{u}_b = 0$ and $\Omega(t) = 0$. The boundary condition, Equation 3.43, is usually expressed as,

$$\mathbf{u} \cdot \hat{\mathbf{s}} = \mathbf{u}_s \cdot \hat{\mathbf{s}}, \tag{3.45}$$

and is referred to as the *no-slip* boundary condition. The paper of Koumoutsakos, Leonard and Pepin [42] stated that, by satisfying no-slip boundary condition directly satisfies the no-through boundary conditions, as these boundary conditions are linked (*Linked boundary condition*). This was also stated by Shiels [56] and have been further employed by Cooper, Mar and Barba in 2009 [20].

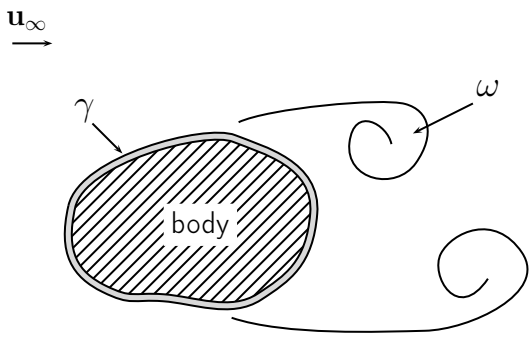


Figure 3.11: Extended vorticity field separated into vorticity in the fluid and the vortex sheet distribution confined to the body.

Typically in an inviscid flow, the boundary condition is enforced after performing the Helmholtz decomposition of the velocity, equation 3.9. The rotational component $\mathbf{u}_\omega$ represents the velocity due to the vorticity in the flow, whereas the harmonic form $\mathbf{u}_h$ is used to take in account of the presence of the body. Koumoutsakos, Leonard and Pepin in 1994 [42], uses the Helmholtz decomposition to divide the vorticity into:

- the vorticity field in the fluid ω,
- the vortex sheet distribution on the boundary γ from the harmonic form,

Figure 3.11 depicts this extended vorticity and the division of the vorticity field to the two sub-categories. The resulting velocity field throughout the domain is given as,

$$\mathbf{u} = \mathbf{u}_\omega + \mathbf{u}_\gamma + \mathbf{u}_\infty \tag{3.46}$$

where $\mathbf{u}_\omega$ is velocity field induced by the vorticity in the flow, $\mathbf{u}_\gamma$ is the velocity field induced by the vortex sheet and $\mathbf{u}_\infty$ is the free-stream velocity.

The vortex sheet distribution γ on the boundary is defined by the boundary integral equations, which will be used to enforce the boundary condition.

3.5.1 Boundary Integral Equations

The Lagrangian method that we are using for the hybrid scheme, is modified according to Stock [61]. The Lagrangian method under-resolved the vorticity field in the near-wall

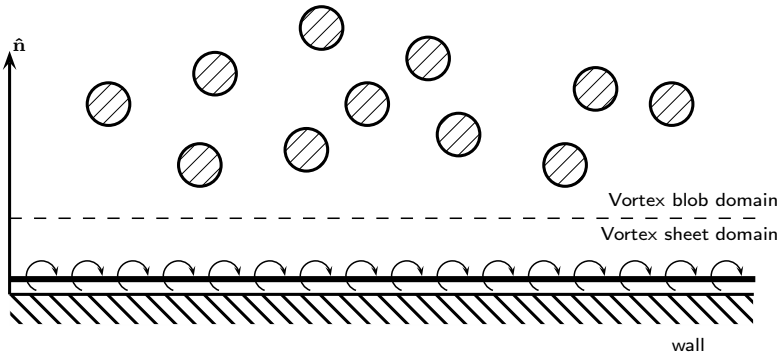


Figure 3.12: Extended vorticity field: Vortex sheet being an extension to the vorticity field (resolved by the vortex blobs), capable of capturing the body bounded vorticity distribution.

region. Furthermore, the vorticity of the fluid is segregated between the vortex blob domain and the vortex sheet domain, as seen in Figure 3.12. The figure shows that, very near the wall the vorticity of the fluid is represented by the vortex sheet. In other words, the vortex sheet is an extension to the vorticity represented by the vortex blobs.

Enforcing the no-slip boundary conditions, equation 3.46, we have that,

$$(\mathbf{u}_{\text{ext}} + \mathbf{u}_{\gamma}) \cdot \hat{\mathbf{s}} = \mathbf{u}_s \cdot \hat{\mathbf{s}} \tag{3.47}$$

where $\mathbf{u}_{\text{ext}} = \mathbf{u}_{\omega} + \mathbf{u}_{\infty}$ is the velocity field induced from the vortex blob domain (i.e external to vortex sheet domain). The equation states that the tangential component of the total velocity acting on the body should be equal to the tangential velocity of the body. So the induced velocity of the vortex sheet is given as,

$$(\mathbf{u}_{\text{ext}} - \mathbf{u}_s) \cdot \hat{\mathbf{s}} = \mathbf{u}_{\gamma} \cdot \hat{\mathbf{s}}. \tag{3.48}$$

Koumoutsakos [39], expressed the relation of the vortex sheet strengths to the no-slip boundary condition at the surface of the body (inside the body) through the Fredholm integral equation of the second kind,

$$-\frac{\gamma(s)}{2} + \frac{1}{2\pi} \oint \frac{\partial}{\partial n} \left[\log \left|\mathbf{x}(s) - \mathbf{x}\left(s'\right)\right|\right] \gamma\left(s'\right) ds' = \mathbf{u}_{\text{slip}} \cdot \hat{\mathbf{s}}. \tag{3.49}$$

where $\gamma(s)$ is the strength of the vortex sheet, and $\mathbf{u}_{\text{slip}} = (\mathbf{u}_{\text{ext}} - \mathbf{u}_b)$, is the slip velocity that needs to be canceled. The left hand side (lhs) of the equation states that at the point $\mathbf{x}_s$, the velocity discontinuity is due to the vortex sheet of that point and integral of all the other vortex sheets on the body. However, equation 3.49 is singular and accepts non-unique solution.

An additional constraint is obtained from Kelvin's circulation theorem stating that the circulation must be conserved at all times. This imposed a direct constraint on the total circulation of the vortex sheet, defined as,

$$\Gamma_{\gamma} = \oint_S \gamma(s) \; ds. \tag{3.50}$$

where Γ_γ is the integral of the vortex sheet strengths γ. The total circulation of the vortex sheet Γ_γ is determined during the hybrid coupling of the Lagrangian method to the Eulerian method, see section 5.1.2.

A potential alternative for removing the singularity associated to the boundary integral equation was demonstrated by Koumoutsakos [40]. He employed a spectral decomposition of the kernel in the Fredholm equation and obtained a set of equations that is well-conditioned, even when the number of panels is increased.

3.5.2 Discretization of Integral Equations using Vortex Panels

The panel method approach, exposed by Katz and Plotkin [37], is used to solve the set of equations, equation 3.49 and equation 3.50. Katz and Plotkins have shown several types of panel distributions with various orders of accuracy; from 0^{th} order point vortex or up to 2^{nd} order linear vortex panel. For this work, we have used a constant-strength vortex distribution that discretizes the vortex sheet into straight segments, classified as Constant-Strength Vortex Panel (CSVP).

Equation 3.49 is solved by discretizing the body surface into M vortex panels, giving us a system of M equation to determine the M unknowns of the strength of the vortex panels.

The integral equation 3.49 is discretized and is given in the matrix form as,

$$\mathbf{A} \cdot \vec{\gamma} = \overrightarrow{\text{RHS}}, \tag{3.51}$$

where $\mathbf{A}$ is an $M \times M$ matrix, containing the coefficients a_{ij}, the influence of vortex panel j on the vortex panel i. $\vec{\gamma}$ is an $M \times 1$ vector contains the strengths γ_i of the vortex panel i and $\overrightarrow{\text{RHS}}$ contains,

$$\text{RHS}_i = \mathbf{u}_{\text{slip}} \cdot \hat{\mathbf{s}}_i. \tag{3.52}$$

The additional constraint 3.50, is similarly discretized and is given as,

$$\sum_{i}^{M} \gamma_i \Delta s_i = \Gamma_\gamma, \tag{3.53}$$

where Δs is the length of the vortex panel i. Combining the equations, we have a system of $M + 1$ equations with M unknowns, given in the matrix form as,

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MM} \\ \Delta s_1 & \Delta s_2 & \cdots & \Delta s_M \end{pmatrix}}_{\mathbf{B}_{(M+1)M}} \underbrace{\begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_M \end{pmatrix}}_{\vec{\gamma}} = \underbrace{\begin{pmatrix} \text{RHS}_1 \\ \text{RHS}_2 \\ \vdots \\ \text{RHS}_M \\ \Gamma_\gamma \end{pmatrix}}_{\overrightarrow{\text{RHS}}}, \tag{3.54}$$

Since we now have a set of $M + 1$ equations with M unknowns, we have to solve the problem either by using a Least-Square solution method (LSTSQ), or by eliminating an equation as used by Katz, or by the spectral decomposition of the kernel in the Fredholm

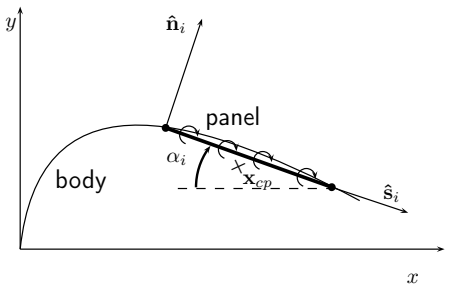
equation 3.49, as used by Koumoutsakos [22]. In this work, we opted for the LSTSQ method that the simplest to implement, however Koumoutsakos showed that to remove the singularity associated to the Fredholm equation 3.49, the spectral decomposition method should be used. The singularity becomes a problem with large number of panels, or thin panel geometries.

3.5.2.1 Constant-Strength Vortex Panel

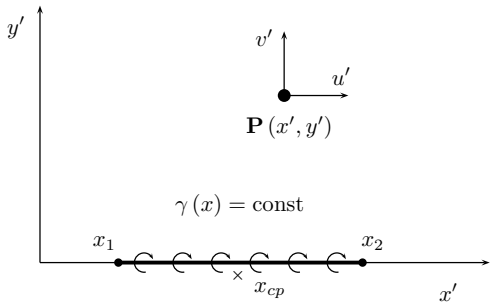
The Constant-Strength vortex panel (CSVP) is based on the flat (straight) discretization of the vortex sheet, where the panels have constant vortex strength. To solve the strengths of the panel problem, we enforce the Dirichlet velocity boundary conditions at the collocation points x_{cp}, that is located just below the vortex sheet, shown in Figure 3.13b. The coefficient a_{ij} of the influence matrix $\mathbf{A}$,equation 3.51, is defined as,

$$a_{ij} = \hat{\mathbf{u}}_{ij} \cdot \hat{\mathbf{t}}_i, \tag{3.55}$$

which represents the tangential influence coefficient of the j^{th} panel on the i^{th} panel. The influence coefficient is determined by prescribing the strengths of the vortex panels $\hat{\gamma}_i = 1$, resulting in an induced velocity $\hat{\mathbf{u}}_{ij} = (\hat{u}, \hat{v})_{ij}$ for a unit strength panel.



(a) Panel discretization of the body in the global cartesian coordinates system (x, y) with the local panel coordinates system rotated by α_i.



(b) Constant-Strength Vortex panel in the local panel coordinate system (x', y') inducing the velocity $\mathbf{u}' = (u', v')$ on the point P.

Figure 3.13: The two coordinate system of the panel method problem. The figure depicts **(a)** the global panel coordinate system, and **(b)** the local panel coordinate system, as defined by Katz and Plotkin [37].

Figure 3.13a shows the discretization of the body into panels in the global coordinates system, defined by (x, y), where each panel is rotated by an angle α_i w.r.t to the global coordinate system. Rotating the axis (x, y) by α_i, we arrive at the local panel coordinate system (x', y'), as shown in Figure 3.13b. Katz and Plotkin [37] showed that, the induced velocity of the vortex panels are calculated in the local panel coordinate system, where the induced velocity of the vortex panel j on the collocation point i (in the panel coordinate

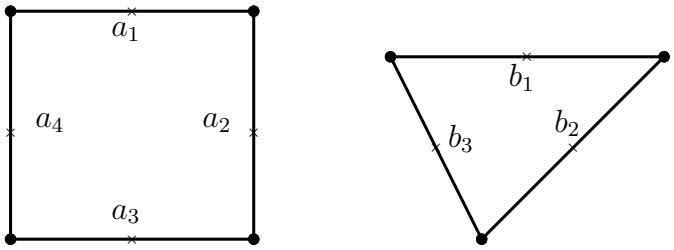


Figure 3.14: Multi-body panel problem: two bodies with different numbers of panels. The figure depicts a square body with 4 panels (a_1, a_2, a_3, a_4), and a triangular body with 3 panels (b_1, b_2, b_3).

system) is given as,

$$u'_{ij} = \frac{\gamma_j}{2\pi}\left[\tan^{-1}\frac{y'_i - y'_{j,2}}{x'_i - x'_{j,2}} - \tan^{-1}\frac{y'_i - y'_{j,1}}{x'_i - x'_{j,1}}\right], \tag{3.56a}$$

$$v'_{ij} = -\frac{\gamma_j}{4\pi}\ln\frac{\left(x'_i - x'_{j,1}\right)^2 + \left(y'_i - y'_{j,1}\right)^2}{\left(x'_i - x'_{j,2}\right)^2 + \left(y'_i - y'_{j,2}\right)^2} \tag{3.56b}$$

where $(x'_1, y'_1)_j$ and $(x'_2, y'_2)_j$ are the coordinates of the panel start and end points in its local panel coordinate system, as shown in Figure 3.13b. The transformation of this vector (u'_{ij}, v'_{ij}) to the global coordinates is given as,

$$\begin{bmatrix} u_{ij} \\ v_{ij} \end{bmatrix} = \begin{bmatrix} \cos\alpha_j & \sin\alpha_j \\ -\sin\alpha_j & \cos\alpha_j \end{bmatrix} \cdot \begin{bmatrix} u'_{ij} \\ v'_{ij}, \end{bmatrix} \tag{3.57}$$

corresponding to a rotation of α, as shown in Figure 3.13a.

If we are dealing with multiple panel bodies (i.e. multiple geometries), as seen in Figure 3.14, the panel problem can be solved by constructing a global influence matrix,

$$\underbrace{\begin{pmatrix} c_{a_1a_1} & \cdots & c_{a_1a_N} & c_{a_1b_1} & \cdots & c_{a_1b_M} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{a_Na_1} & \cdots & c_{a_Na_N} & c_{a_Nb_1} & \cdots & c_{a_Nb_M} \\ c_{b_1a_1} & \cdots & c_{b_1a_N} & c_{b_1b_1} & \cdots & c_{b_1b_M} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{b_Ma_1} & \cdots & c_{b_Ma_N} & c_{b_Mb_1} & \cdots & c_{b_Mb_M} \\ \Delta s_{a_1} & \cdots & \Delta s_{a_N} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \Delta s_{b_1} & \cdots & \Delta s_{b_M} \end{pmatrix}}_{\begin{pmatrix} C_{aa} & C_{ab} \\ C_{ba} & C_{bb} \\ \Delta s_a & 0 \\ 0 & \Delta s_b \end{pmatrix}} \underbrace{\begin{pmatrix} \gamma_{a_1} \\ \vdots \\ \gamma_{a_N} \\ \gamma_{b_1} \\ \vdots \\ \gamma_{b_M} \end{pmatrix}}_{\begin{pmatrix} \gamma_a \\ \gamma_b \end{pmatrix}} = \underbrace{\begin{pmatrix} \mathrm{RHS}_{a_1} \\ \vdots \\ \mathrm{RHS}_{a_N} \\ \mathrm{RHS}_{b_1} \\ \vdots \\ \mathrm{RHS}_{b_M} \\ \Gamma_{\gamma,a} \\ \Gamma_{\gamma,b} \end{pmatrix}}_{\begin{pmatrix} \mathrm{RHS}_a \\ \mathrm{RHS}_b \\ \Gamma_{\gamma,a} \\ \Gamma_{\gamma,b} \end{pmatrix}} \tag{3.58}$$

where the matrices (C_{aa}, C_{bb}), are the self-induction matrices of each of the single vortex panel problem. The non-diagonal terms (C_{ab}, C_{ba}) are the inter-induction matrices containing the panel influence of body b on body a and body a on body b, respectively. The final two rows of the LHS matrix contain the circulation constraint for each body, defined by equation 3.53.

3.6 Evolution of the Lagrangian method

The algorithm for evolving the full Lagrangian method is summarized in this section. The full Lagrangian method is the coupled vortex blobs and the vortex panels. Note that for our hybrid scheme, the panel does not act as the source of the vorticity in the Lagrangian method (which is done by the Eulerian method), but instead simply enforces the *no-slip* boundary condition for the vortex blobs.

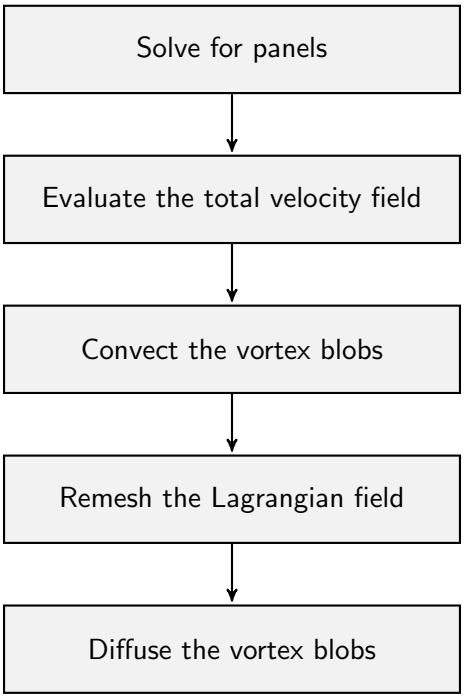


Figure 3.15: Flowchart of the Lagrangian method. The flowchart shows coupling between vortex panels and vortex blobs to evolve from t_n to t_{n+1} (without taking into account of the vorticity generation at the boundary).

The flowchart of one time step of the Lagrangian method is given by Figure 3.15.The algorithm to the Lagrangian method can be summarized as follows:

1. **Solve for panels:** Determine the strengths of the vortex panels γ, such that the no-slip boundary condition at the collocation points of the vortex panels is enforced. When determining the strengths, we also have to ensure that the total circulation of the vortex panels satisfies the conservation of circulation, equation 3.50, which we investigate during the hybrid coupling, section 5.1.2.

2. **Evaluate the total velocity field:** Evaluate the total velocity field $\mathbf{u}$, which is the sum of velocity field induced by the vortex blobs $\mathbf{u}_\omega$, the velocity field induced by the vortex panels $\mathbf{u}_\gamma$, and the free-stream velocity field $\mathbf{u}_\infty$.

3. **Convect the vortex blobs:** Use the velocity field to convect the vortex blobs from t_n to t_{n+1} to the new position. The strength of the vortex panels remain constant during this step.

4. **Remesh the Lagrangian field:** Remesh the vortex blobs onto a structured square lattice using the M'_4 interpolation kernel.

5. **Diffuse the vortex blobs:** Diffuse the vortex blobs using the Δt_d diffusion time step, by modifying the strengths of the vortex blobs according to Wee's WRS or Tutty's TRS method.

The generation of the vorticity is dealt with in the Eulerian subdomain, which is explained in chapter 4. The vorticity is then transfered into the Lagrangian domain using the Hybrid coupling scheme, which was summarized in the introduction, chapter 1, and fully elaborated in chapter 5.

3.7 Validation of Lagrangian method

In this chapter, we have investigated the Lagrangian component of the Hybrid method. The Lagrangian method is used to just evolve the vorticity field, where as the Eulerian method is used to properly formulate the vorticity generation at the boundary. The resolved boundary solution of the Eulerian method is then transfered onto our Lagrangian method. Therefore, the Lagrangian method that is implemented here does not require the generation of the vorticity from the boundary.

Thus, during the validation of the Lagrangian method, we focus on two test cases: Potential flow around a cylinder and Lamb-Oseen vortex evolution. The potential flow around a cylinder test case is used to verify and validate the vortex panel solver that is used enforcing the no-through flow for the vortex blobs. The Lamb-Oseen vortex evolution test cases is used to verify and validate the convection method and the diffusion methods implemented for the evolution of the vortex blobs.

Note that to investigate the coupling of the vortex blobs and the vortex panels, we require the proper definition of the vorticity flux from the boundary, requiring the Eulerian method as well. Therefore, the handling of the vorticity flux is investigated in fully coupled method, in chapter 7.

3.7.1 Error Analysis of Vortex Panels

The vortex panels was verified and validated on the test case of the potential flow around a cylinder. To test the convergence of the solution of the vortex panels, a comparison was made with the analytical solution for the parameters in table 3.1.

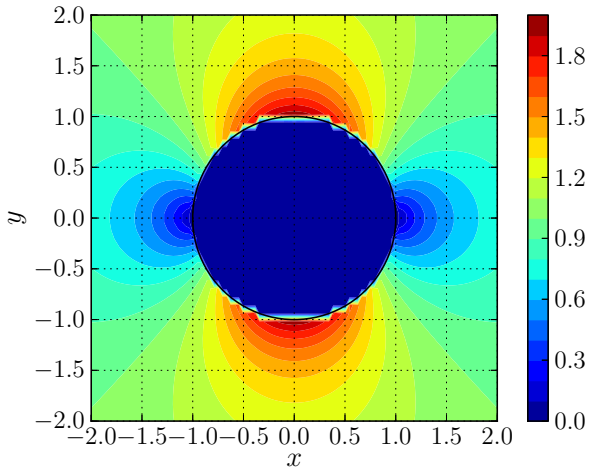
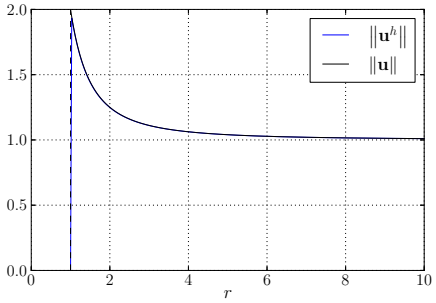
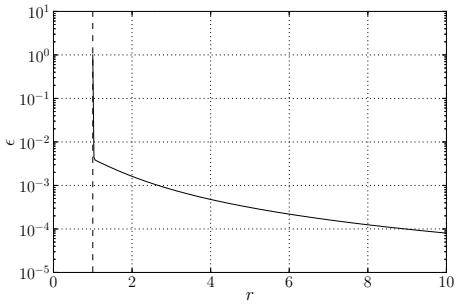


Figure 3.16: Panel method solution: the potential velocity field around a unit cylinder with $R = 1$, $\mathbf{u}_\infty = (1, 0)$, and $N_{\text{panels}} = 100$. The figure depicts the magnitude of velocity field $\|\mathbf{u}\|$, with a zero velocity inside the body.



(a) Comparison of the velocity field.



(b) Error in the velocity field

Figure 3.17: Comparison of the velocity field along the y-axis, $y = 0$ to $y = 10$. Figure **(a)** shows both the solutions, the numerical $\|\mathbf{u}^h\|$ [—, solid **blue**] and the analytical solution [—, solid **black**]. Figure **(b)** shows the relative error ϵ in velocity between the solution, given by equation 3.60.

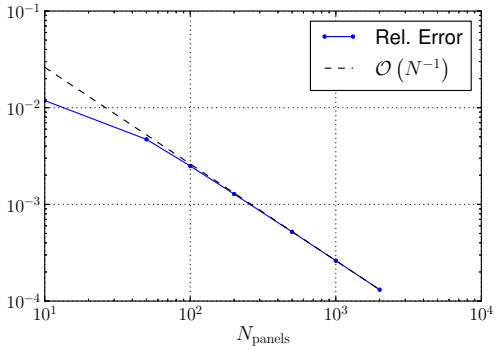


Figure 3.18: Convergence plot of the Constant-Strength Straight Vortex panels. The figures depicts the converges of the relative error ϵ at an $\mathcal{O}\left(N^{-1}\right)$.

An example of the numerical solution is shown in Figure 3.16. The figure shows the magnitude of the velocity $\|\mathbf{u}\|$, and it shows the velocity field of the potential flow solution, with an infinitely thin boundary layer, stagnating to zero velocity inside the body.

The jagged velocity field around the surface of the cylinder is simply due to the sampling resolution of the field. For a higher sampling resolution this will vanish. In order to determine the accuracy of the solution, the velocity field of the panel solution was compared with the analytical solution. The analytical velocity field around a cylinder is given in cylindrical coordinate centered in the cylinder as,

$$u_r = u_\infty \left(1 - \frac{R^2}{r^2}\right) \cos\theta, \tag{3.59a}$$

$$u_\theta = -u_\infty \left(1 + \frac{R^2}{r^2}\right) \cos\theta, \tag{3.59b}$$

where u_r and u_θ are the radial and the angular velocity respectively for a given free-steam velocity u_∞. Equation 3.59 is a function of the distance to the center of the cylinder r and the radius of the cylinder R and is valid for $r \geqslant R$

The velocity field of the vortex panel was compared with the analytical solution along the y-axis from $y = 0$ to $y = 10$. Figure 3.17a plots the magnitude of analytical velocity $\|\mathbf{u}\|$ and the vortex panel velocity field $\left\|\mathbf{u}^h\right\|$. Comparing the solutions of the plot we see that the solution of the vortex panels and the analytical potential flow solution matches everywhere except at the surface. This happens because the potential flow solution has a slip velocity (i.e non-zero velocity) at the surface of the body, whereas the vortex panels solves for a no-slip boundary condition at the collocation points of the surface. This explains the sudden drop of the velocity from $\|\mathbf{u}\| = 2$ to $\|\mathbf{u}\| = 0$ at the surface.

Figure 3.17b shows the relative error ϵ between the numerical and the analytical solutions,

$$\epsilon = \frac{\left\|\mathbf{u} - \mathbf{u}^h\right\|}{\|\mathbf{u}\|} \tag{3.60}$$

where $\mathbf{u}$ is the analytical solution and the $\mathbf{u}^h$ is the numerical solution. Ignoring the solution right at the surface ($r = R$), we see that the error between the numerical and the analytical solution reduces from $\epsilon = 5 \times 10^{-3}$ to $\epsilon = 8 \times 10^{-5}$ as we go from $r = 1$ to $r = 10$. This behavior of the error tells us that the solution of the constant-strength vortex panels gets more accurate as we go further away from the panels; right next to the panels, we have the largest error.

The convergence analysis of the vortex panels was done by determining the error of the vortex panel velocity field w.r.t to the analytical solution for the number of panels $N = 10$

Table 3.1: Panel study parameters

Parameters	Value	Description
R	1 m	Radius of cylinder
u_∞	1 m s^{-1}	Free-stream velocity
N_{panels}	100	Number of panels

to $N = 1000$, Figure 3.18. The error of the velocity field was computed at $(x = 0, y = 1.5)$, and we see that the error converges at with $\mathcal{O}(N)$. This validates that the vortex panel that we have used is a 1^{st} order vortex panel.

3.7.2 Error Analysis of Vortex Blobs

In order to verify and validate the vortex blobs, we simulate the evolution of a Lamb-Oseen vortex. The results of the simulation were used to compare against the analytical ones, which we used to determine the accuracy of our vortex blobs.

The Lamb-Oseen vortex is a solution of the Navier-Stokes equation, corresponding to the viscous evolution of a laminar vortex core on an unbounded domain, first derived by Lamb and Oseen [63]. The vorticity distribution ω of the core at a given time is defined as,

$$\omega(\mathbf{x}, \tau) = \frac{\Gamma_c}{4\pi\nu(t+\tau)} \exp\left(-\frac{r^2}{4\pi\nu(t+\tau)}\right), \tag{3.61}$$

and is a function of core strength Γ_c, the simulation time $t \in [0, \infty[$ and distance from the core center r. The constant τ in the equation 3.61 defines to initial width of the Lamb-Oseen vortex, where if $\tau = 0$, we have Dirac delta distribution.

The velocity field, corresponding to equation 3.61, in cylindrical coordinate is defined as,

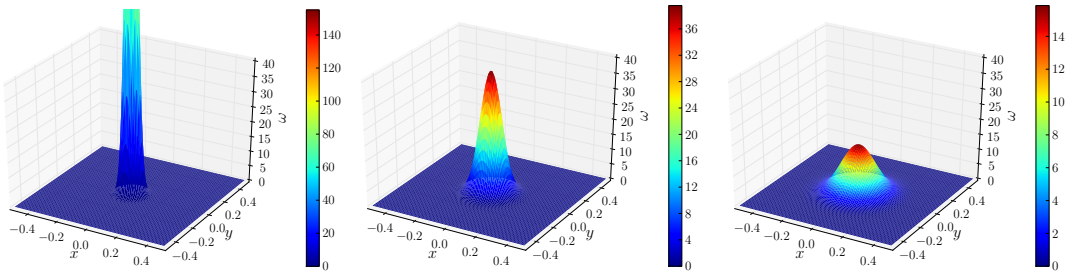
$$u_\theta = \frac{\Gamma_c}{2\pi r}\left[1 - \exp\left(-\frac{r^2}{4\pi\nu(t+\tau)}\right)\right] \tag{3.62a}$$

$$u_r = 0 \tag{3.62b}$$

where u_θ is the circumferential velocity, and u_r, the radial velocity is zero. Figure 3.19 shows the vorticity distribution ω for various initial time constant τ. We see that for small τ, the distribution approaches a Dirac delta distribution. Therefore, for this investigation we decided on $\tau = 4$ ensuring a non-peaky distribution, which was also investigated by Barba [2]. Therefore the literature of Barba [2] will serve as the validation data for our Lamb-Oseen investigation.

The Lagrangian method was applied to the Lamb-Oseen vortex test case with parameters tabulated in table 3.2. The vorticity field was discretized over the domain $[x, y]$ - domain $[-0.5, 0.5] \times [-0.5, 0.5]$. This was adequate as the circulation outside this domain was less that the threshold $\Gamma_{loc} \leq 10^{-14}$. The spatial discretization was performed according to the standard initialization method of vortex blobs, described in section 3.2.4.

However as explained in section 3.2.4, we have to take in account of the Gaussian blurring of the original vorticity field due to the initialization process. This poses a problem when evaluating the error between the numerical and the analytical solution. This problem has also been encountered by Barba [2], when investigating the Lamb-Oseen vortex. The solution to the problem was to apply a "time-shift correction", to compensate for the



(a) $\tau = 1$ **(b)** $\tau = 4$ **(c)** $\tau = 10$

Figure 3.19: The vorticity ω distribution of the Lamb-Oseen vortex problem with $\Gamma_c = 1$ and $\nu = 5 \times 10^{-4}$ in the domain $[-0.5, 0.5] \times [-0.5, 0.5]$. The figure depicts distribution for various initial time constant τ, determining the peakiness of the distribution.

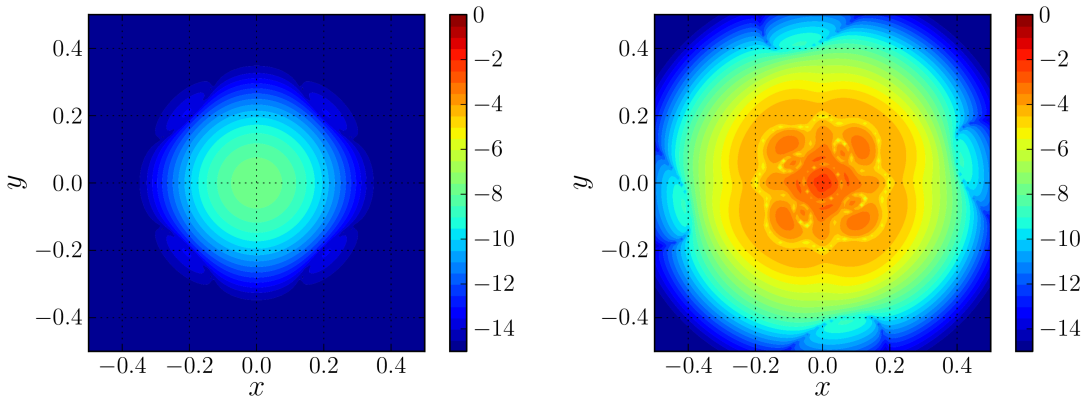
Gaussian blurring, solving the problem of this very particular discretization of the Navier-Stokes equation. Therefore, this is a special method and this approach can only be used for the Lamb-Oseen vortex problem.

The "time-shift correction" is derived by determining the diffusion effect caused by the discretization of the diffusion equation using the Gaussian vortex blobs (with $k = 2$). Barba [2], determined that the discretization of the diffusion equation (i.e. the Lamb-Oseen vortex) reconstructs the vorticity field that has been diffused by a time $\sigma^2/2\nu$. So when initializing the particles with a certain strength, we will have to reverse the time by $\sigma^2/2\nu$. Thus, the corrected initial particles strengths α_i^o of vortex blobs from the Lamb-Oseen vorticity field is given as,

$$\alpha_i^o = \omega_i^o \cdot h^2 = \left\{ \frac{\Gamma_c}{4\pi\nu\left(t + \tau - \sigma^2/2\nu\right)} \exp\left[-\frac{r_i^2}{4\nu\left(t + \tau - \sigma^2/2\nu\right)} \right] \right\} \cdot h^2. \tag{3.63}$$

Table 3.2: Summary of the parameters for the Lamb-Oseen vortex evolution. This table shows also the parameters of Tutty's diffusion method

Parameters	Value	Unit	Description
Γ_c	1	$\mathrm{m^2\,s^{-1}}$	Core strength
Ω	$[-0.5, 0.5] \times [-0.5, 0.5]$	m	Initial particle domain
$\mathbf{u}_\infty$	0	$\mathrm{m\,s^{-1}}$	Free-stream velocity
ν	5×10^{-4}	$\mathrm{kg\,s^{-1}\,m^{-1}}$	Kinematic viscosity
τ	4	s	Lamb-Oseen time constant
t	0 to 1	s	Simulation time
$\Delta t_c = \Delta t_d$	0.01	s	Diffusion and convection time step size
N_t	100	-	Number of time integration steps
σ	0.01	m	Vortex blob core size
λ	1	-	Overlap ratio
k	2	-	Gaussian kernel width spreading
f_{redis}	1	-	Redistribution frequency
f_{pc}	1	-	Population control frequency
$(\Gamma_{loc}, \Gamma_{glob})$	$(1 \times 10^{-14}, 1 \times 10^{-14})$	$\mathrm{m^2\,s^{-1}}$	Population control thresholds



(a) $t = 0$ **(b)** $t = 1$

Figure 3.20: Relative error growth of Lamb-Oseen vorticity during the evolution (in logarithmic scale) using the parameters tabulated in table 3.2. The figure shows **(a)**, the initial relative error at $t = 0$, and **(b)** the final relative error in vorticity at $t = 1$.

This method was used to investigate the error evolution of the vortex blob method. The vortex blobs where convected according to the procedures in section 3.3. The diffusion of the vortex blobs was performed using the schemes described in section 3.4. We investigated the accuracy of the Tutty's scheme (TRS) and Wee-Ghoniem scheme (WRS) in section 3.7.2.1. For the general investigation however, we employed the Tutty's diffusion method. The advantage of this approach is that the we can perform diffusion after every convection step. This makes the method less prone to time integration error and eliminates any discontinuous behavior in the evolution. We will see that when coupling the Lagrangian method and Eulerian method, discontinuity in the problem introduces additional errors.

The convection and diffusion was performed according to the time integration parameters tabulated in table 3.2. The vortex blobs where convected using a 4^{th}-order Runge-Kutta method (RK4) for a high order time integration. After the convection substeps, the Lagrangian distortion was treated using the remeshing scheme discussed in section 3.3.1. Generally, the remeshing is typically done every 10 iterations [2]. However, as our diffusion scheme and hybrid method requires structured lattice of vortex blobs for efficient calculations, we will remesh after every step, $f_{\text{redist}} = 1$.

In addition to the evolution of the vortex blobs, we performed a population control to minimized the number of vortex blobs, as described by Barba [2]. The Population Control (PC) removes vortex blobs that have very small circulation strengths. After the diffusion and remeshing, we will be left with vortex blobs with strengths close to the numerical precision, as they have minimal impact on the accuracy of the vorticity field, we can removed them. When performing population control, we need to ensure that the loss of total circulation is below the acceptable global threshold, Γ_{glob}. We used $\Gamma_{glob} = 10^{-14}$ as used by Barba [2] for the similar investigation.

To verify whether our Lagrangian scheme is performs according to theory, we evaluated the error evolution of the simulation. Figure 3.20, shows the initial and the final relative error in vorticity. We see that initially we have a maximum relative error around 10^{-8}, located at the center of the Lamb-Oseen core. After 100 time integration steps from $t = 0$

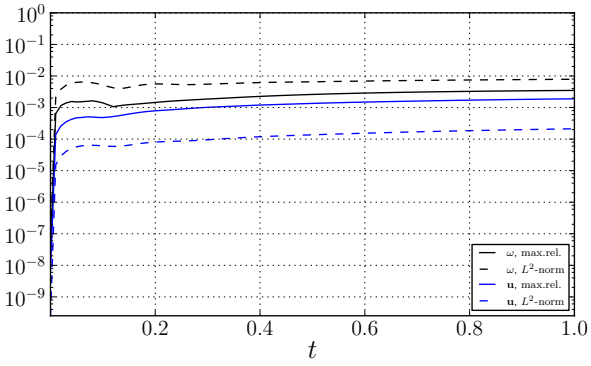


Figure 3.21: Relative error growth of Lamb-Oseen vortex during the evolution from $t = 0$ to $t = 1$ using the parameters in table 3.2. This figure depicts the error in vorticity: maximum relative error [—, solid **black**], and the error in $L^2 - \text{norm}$ [- -, dashed **black**]; and error in velocity: maximum relative error [—, solid **blue**], and error in $L^2 - \text{norm}$ [- -, dashed **blue**].

to $t = 1$, we see that the maximum relative error in vorticity increases from 10^{-8} to 10^{-2}. The errors of the vorticity are predominantly localized at the center of the core, where we have maximum vorticity, Figure 3.20.

Figure 3.21, shows the maximum relative error, equation 3.64, and the L^2-norm, equation 3.65, error evolution of vorticity and velocity from $t = 0$ to $t = 1$. Similar investigation was performed by Barba [2] and Speck [60]. Due to the relation of the vorticity and the velocity, equation 3.1, the error of the vorticity is higher that the error in the velocity. The figure shows both the maximum relative error, and the error in the $L^2 - \text{norm}$. The maximum relative error (e.g. for vorticity), is defined as,

$$\left\| \omega^{\text{exact}} - \omega^{\text{discrete}} \right\|_{\infty} = \frac{\max\{\left| \omega^{\text{exact}} - \omega^{\text{discrete}} \right|\}}{\max\{\left| \omega^{\text{exact}} \right|\}}, \tag{3.64}$$

where ω^{exact} is the analytical vorticity field, equation 3.61, and ω^{discrete} is the numerical vorticity field from the vortex blobs. The error in the $L^2 - \text{norm}$ of the vorticity is calculated as

$$\left\| \omega^{\text{exact}} - \omega^{\text{discrete}} \right\|_{2} = \left(\sum_{i}^{N} \left| \omega^{\text{exact}} - \omega^{\text{discrete}} \right|^2 \cdot h^2 \right)^{\frac{1}{2}}, \tag{3.65}$$

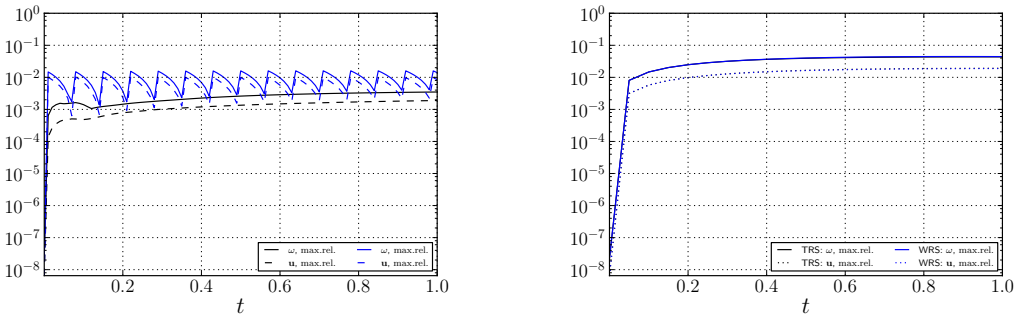
and the error in velocity is calculated using the same principle. Investigating the figure, we see that after the first iteration, there is a sudden increase in the error, but as time progresses the error growth reduces. From literature, we see that this trend has also been observed by Barba [2] and Speck [60]. For comparison, we used similar parameters, and we observe that the sudden jump in error is similar to the literature.

3.7.2.1 Comparison of Diffusion Schemes: WRS vs. TRS

To observe how both the diffusion schemes compare, we ran the same test case with both diffusion schemes. From the simulation, we were able to observe that Tutty's diffusion scheme (TRS), produced less error than Wee's approach (WRS).

Figure 3.22 shows the evolution of maximum relative error in vorticity, equation 3.64 for both diffusion schemes. Figure 3.22a shows the evolution of error for convective time step $\Delta t_c = 0.01$. The diffusion scheme TRS enables us to perform diffusion in conjunction with the convection, $\Delta t_d = \Delta t_c = 0.01$. This was possible due to the favorable constraint on the diffusion time step, equation 3.41.

The Wee's diffusion scheme WRS, however is constraint by equation 3.30 and equation 3.31. Therefore the diffusion time step Δt_d for the given convective time step $\Delta t_c = 0.01$ is $\Delta t_d = k_d \cdot \Delta t_c = 0.07$, where the diffusion frequency $k_d = 7$. We observe from the figure that performing diffusion at every other instant creates an oscillatory behavior. This behavior is not ideal when coupling with the Eulerian method as the oscillatory diffusion of the VPM will add additional error in coupling.



(a) $\Delta t_c = 0.01$ **(b)** $\Delta t_c = 0.05$

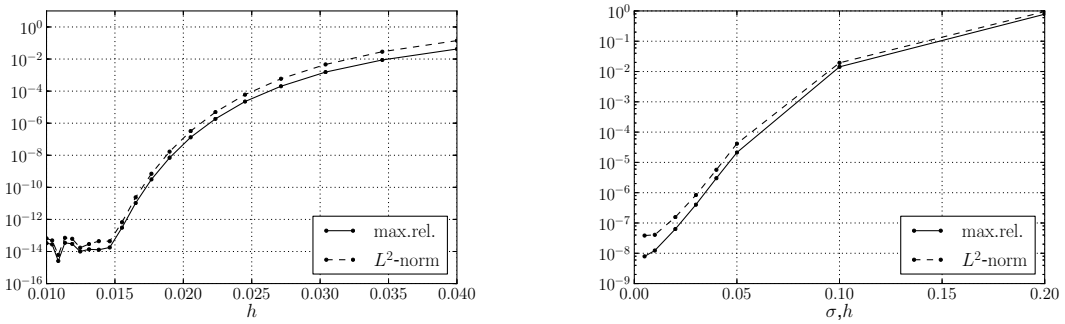
Figure 3.22: Comparison of Tutty's scheme TRS, and Wee-Ghoniem scheme WRS for treating diffusion, depicting the evolution of maximum relative error in vorticity, equation 3.64 from $t = 0$ to $t = 1$. The Figure **(a)** shows TRS performing diffusion at every step, $\Delta t_d = \Delta t_c = 0.01$ and WRS performing diffusion at every 7^{th} step, $\Delta t_d = k_d \cdot \Delta t_c = 7 \times 0.01$; **(b)** shows TRS performing diffusion at every step, $\Delta t_d = \Delta t_c = 0.05$ and WRS performing diffusion at every step, $\Delta t_d = k_d \cdot \Delta t_c = 1 \times 0.05$.

However, when modifying the convective time step to $\Delta t_c = 0.05$, Figure 3.22b, we observe that the error of WRS matches the TRS. At this convective time step Δt_c, the WRS has a diffusion time step $\Delta t_d = k_d \cdot \Delta t_c = 0.05$, where the diffusion frequency $k_d = 1$ now. Therefore, the WRS performs diffusion at every step and we see that WRS performs similarly to TRS.

The conclusion to this investigation is that WRS is useful if we are able to match the convective time step Δt_c to the diffusion time step Δt_d. However, this may not be possible for high Reynolds number flows where the convective time step is critical. The TRS outperforms WRS in this regard and should produce less error when coupling with the Eulerian method.

3.7.3 Convergence Study of the Viscous Vortex Method

Finally, we can perform a converge study, to validate that our scheme works according to the theory. For a scheme that is numerically stable, the error due to discretization must converge as the resolution of the discretization increases.



(a) Error in vorticity vs. h with $\sigma = 0.02$

(b) Error in vorticity vs. σ, h with $\lambda = h/\sigma = 1$.

Figure 3.23: Convergence in spatial discretization of the vortex blobs. Figure **(a)** shows the convergence by fixing the core size σ and **(b)** shows the convergence when overlap ratio $\lambda = h/\sigma = 1$.

First, we investigate the convergence for spatial discretization. As we are dealing with vortex blobs, there are multiple ways of increasing the resolution. The straightforward method would be to increase the density of particles in a given area, i.e. reduce the blob spacing h and maintaining the core spreading σ. Figure 3.23a shows the convergence of the spatial discretization when the core size σ is maintained at $\sigma = 0.02$. For this case, the overlap ratio changes with the blob spacing, described by equation 3.20. For small blob spacing h, the error in vorticity quickly drops to near machine precision. When investigating the order of convergence, we see that the error converges in a non-linear fashion and similar results have been obtained by Barba [2].

Figure 3.23b, shows the convergence of the error when the overlap ratio is fixed, $\lambda = 1$. In this test case, the core size scaled with the blob spacing, $h = \sigma$, and when increasing the spatial resolution, the error converges non-linearly.

To investigate the convergence in temporal discretization, we determined the evolution of the maximum relative error in vorticity, equation 3.64, at $t = 1$ for various convective

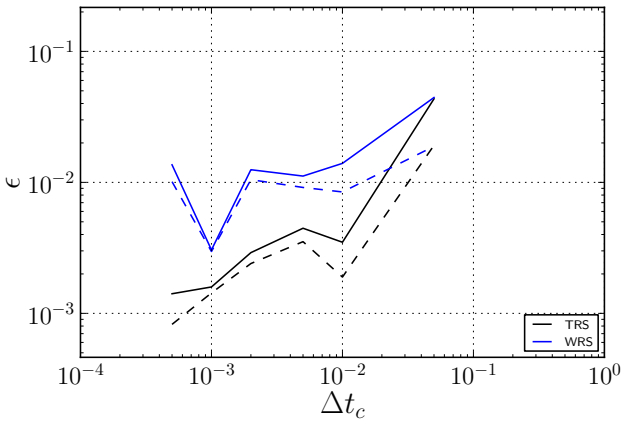


Figure 3.24: Convergence of the error in velocity [—, solid] and the error in vorticity [- -, dashed] due to temporal discretization of the vortex blobs. The figure compares the convergence rate of TRS (**black**) vs. WRS (**blue**)

time step Δt_c, shown in Figure 3.24. At $t = 1$, we see that the error reduces as we increase the temporal resolution meaning that we have a convergent scheme. We also observe that the error produced by WRS is higher than TRS and that it converges at a lower order than TRS. This again validates that TRS performs better than the WRS as it produces less error.

3.8 Summary

In summary, we have investigated the Lagrangian subdomain of our hybrid method in this chapter. The Lagrangian method was used to described the evolution of the wake past the geometry. Vortex Particle Method (VPM) was an ideal choice to describe the wake, as we require only to evolve the wake, and the generation of the vorticity is dealt with in the Eulerian subdomain. Unlike the Eulerian method, VPM only required the fluid elements where there was vorticity, meaning that the VPM was inherently auto-adaptive. Using the Population Control method, we were able to remove vortex blobs where they were not needed. Furthermore, the computation of the these elements were accelerated using an FMM, and simultaneously was parallelized using a GPU hardware.

In section 3.1, an introduction to the VPM was given. We determined advantage of the Lagrangian method w.r.t to the Eulerian method for resolving the wake for the VAWT. The velocity-vorticity formulation of the Navier-stokes equations is the governing equation of the VPM and we investigated the viscous splitting algorithm in section 3.1.3.

The viscous splitting algorithm enabled to perform diffusion and convection of the fluid is segregated steps. The discretization of the fluid through vortex blobs was investigated in section 3.2. These fluid elements has non-zero core size, removing the singularity when performing Biot-Savart calculations.

In section 3.2.4, we investigate the initialization of the vortex blobs. The proper initialization of the vortex blobs is a key factor for accurate coupling of Lagrangian method and the Eulerian method. The strengths of these particles is initialized by assigning the local circulation strength to the particle, as in equation 3.21. When the coupling is performed, it will be seen that the Gaussian blurring of the original vorticity field during the initialization is the fundamental source of error, section 5.1.1. Strategies such as Beale's iterative method, cannot be used as it is defined for an unbounded domain. The only approach found to minimize the Gaussian spreading initialization error is to increase the overlap ratio to $\lambda = 1$, and minimize the blob spacing h as much as possible, while keeping the computational effort to an acceptable level. The optimal strategy for the initialization of the vortex blob strengths is still an open question, and if solved can significantly improve the accuracy and the efficiency of the hybrid coupling

In section 3.3, we investigated the convection of the vortex blobs. The convection is performed using a 4^{th}-order Runge-Kutta time integration method. However, due to high strains in the fluid, the Lagrangian grid distortion of the vortex blob lattice has to be dealt with, section 3.3.1. For this reason, we used a M'_4 interpolation kernel that remeshed the particles onto a structured grid.

In section 3.4, we investigated two diffusion models for the vortex blobs. The WRS diffusion model developed by Wee and Ghonien [67], integrated the diffusion process into

the standard interpolation kernel. This reduces computation cost, however the model was unfavourable constraint on the diffusion time step equation 3.31. The constraint limits the minimum diffusion step size and results in a discontinuous diffusion in time, as shown in Figure 3.22a. To overcome this problem, we used the TRS diffusion model by Tutty [64], which enabled us to perform diffusion after every convection step, section 3.4.2. This also ensured that the diffusion process was continuous, which was important when performing the coupling algorithm.

In section 3.5, we investigated the handling of the *no-slip* boundary conditions for the viscous VPM. The boundary integral equations was used to enforces the wall boundary conditions in the Lagrangian method. We used the Constant-Strength Vortex panels, based on Katz [37], to discretize the integral equations. The panel method was then verified and validated with the analytical solution of a potential flow around a cylinder in section 3.7.

In section 3.7, we also verified and validated the implementation of the vortex blobs to analytical solution of the Lamb-Oseen vortex problem. We determined the evolution of the error for various spatial discretization and temporal discretization. The validation concluded that the implementation performed according to the literature, see example Barba [2].

3.9 Chapter Nomenclature

Latin Symbols

$\mathbf{A}$	Vortex panel influence matrix	-
c^2	Diffusion parameter	-
$\mathcal{E}$	Enstrophy	$\mathrm{m}^2\,\mathrm{s}^{-2}$
f_{pc}	Population control frequency	-
f_{redis}	Redistribution frequency	-
h	Nominal particle spacing	m
h_ν	Characteristic diffusion distance	m
k	Gaussian kernel width spreading	-
k_d	Frequency of vortex blob diffusion	-
$\mathbf{K}$	Biot-Savart kernel	-
$\mathbf{K}_\sigma$	Vortex blob kernel	-
$\hat{\mathbf{n}}$	Unit normal vector	-
N	Number of vortex blobs (particles)	-
λ	Overlap ratio	-
p	Pressure	Pa
r	Radial position	m
$\hat{\mathbf{s}}$	Unit tangent vector	-
t	Simulation time	s
$\mathbf{u}$	Velocity	$\mathrm{m}\,\mathrm{s}^{-1}$

$\mathbf{u}_b$	Velocity of the body	$\mathrm{m\,s^{-1}}$
$\mathbf{u}_\gamma$	Vortex sheet induced velocity	$\mathrm{m\,s^{-1}}$
$\mathbf{u}_{\mathrm{ext}}$	External induced velocity	$\mathrm{m\,s^{-1}}$
$\mathbf{u}^h$	Discrete velocity	$\mathrm{m\,s^{-1}}$
$\mathbf{u}_\infty$	Free-stream velocity	$\mathrm{m\,s^{-1}}$
$\mathbf{u}_\phi$	Free-stream velocity	$\mathrm{m\,s^{-1}}$
u_r	Radial velocity	$\mathrm{m\,s^{-1}}$
u_θ	Angular velocity	$\mathrm{m\,s^{-1}}$
$\mathbf{u}_{\mathrm{slip}}$	Boundary slip velocity	$\mathrm{m\,s^{-1}}$
$\mathbf{u}_\omega$	Vorticity velocity	$\mathrm{m\,s^{-1}}$
W	Interpolation kernel weight	-
$\mathbf{x}$	Position vector	m
$\mathbf{x}_\nu$	Position vector of particle to be diffused	m
$\mathbf{x}_p$	Position vector of vortex blob (particle)	m

Greek Symbols

α_p	Circulation of the particle	$\mathrm{m^2\,s^{-1}}$
Δt_c	Convection time step size	s
Δt_d	Diffusion time step size	s
ϵ	Relative error	-
Γ	Circulation	$\mathrm{m^2\,s^{-1}}$
Γ_{loc}	Particle circulation threshold	$\mathrm{m^2\,s^{-1}}$
Γ_{glob}	Total circulation threshold	$\mathrm{m^2\,s^{-1}}$
ν	Kinematic viscosity	$\mathrm{m^2\,s^{-1}}$
ω	Vorticity	$\mathrm{s^{-1}}$
$\tilde{\omega}$	Vortex blob cell vorticity	$\mathrm{s^{-1}}$
ω^h	Discrete vorticity field	$\mathrm{s^{-1}}$
Ω	Fluid domain	m
ρ	Density	$\mathrm{kg\,m^{-3}}$
σ	Core size	m
τ	Lamb-Oseen time constant	s
ξ	Scale relative position of particle to stencil node	-
ζ_σ	Smooth cut-off function of the blobs	-

Chapter 4

Eulerian Method: Finite Element Method

In this chapter, we introduce the Finite element Method used to solved the Eulerian subdomain of the hybrid method, and summarize the process for evolving the Eulerian method, step 4 of the hybrid evolution, Figure 4.1.

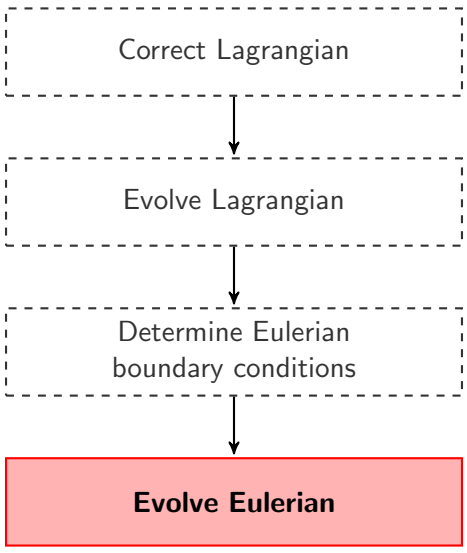


Figure 4.1: Flowchart of the hybrid evolution, focusing on the 4^{th} step: Evolve the Eulerian solution.

Standard Computation Fluid Dynamics (CFD) methods discretize the fluid into smaller grids, and solve the set of Navier-Stokes (NS) equations in these regions. Eulerian methods use this type of formulation.

For the hybrid method, we use the Navier-Stokes Eulerian formulation in the near-body region. The advantage of using this formulation in this region is that it is much more

efficient in resolving boundary layers than the Vortex Particle Method (VPM). We can directly enforce the wall boundary condition at the wall boundary of the Eulerian subdomain, solving the problem of vorticity generation of the body.

The most common approaches to solve the fluid dynamics problem on an Eulerian reference frame are Finite Volume Method (FVM), Finite Difference Method (FDM), and Finite Element Method (FEM). FVM divides the domain into volumes where it enforces the conservation of mass and momentum in each grid. FDM divides the domain into nodes and uses local Taylor expansions to approximate the partial differential equations. FEM divides the domain into elements and solves the problem using variational calculus.

For the current study, we have decided to use the FEM package provided by the FEniCS project as it has implemented efficient, multi-threaded algorithms for setting up and solve a finite element problem. Furthermore, it provides extensive features for future developments such as adaptive mesh refinement, deformable meshes, and efficient computation of turbulent flow.

4.1 Introduction to the Finite Element Method

The Finite Element Method (FEM) is a numerical method to obtain approximations to partial differential equations. The equations are solved by writing them as a variational problem, giving us an approximate solution for the boundary value problem [7]. Therefore the FEM approximates the unknown functions and converts the partial differential equations into a set of algebraic equations, which makes them suitable to be solved numerically. It was traditionally used for solid mechanics (e.g for the analysis of aircraft structures [52]), but have since been used to solve fluid dynamics problems [32, 35, 33].

4.1.1 Finite Element Discretization

The finite element method solves a problem by dividing the domain of interest into smaller cells known as *elements*. These *elements* are connected at the vertices which are called nodes or nodal points. We use these sets of nodes and elements to represent the variation in the field such as the displacement, the velocity, the pressure or the temperature using simple functions, known as basis functions. Thus, we have transformed the problem of interest into a finite number of Degrees of Freedom (DOF). We combined the set of equations of the elements into a global system of equations to solve for the unknowns.

A finite element discretization in 2D can be seen in Figure 4.2. The figure shows two connected elements, where the cells represent the area of the element, and the vertices of the cell represents the nodes of the element. The set of all the cells $\mathcal{T}_h = \{T\}$ in the fluid domain Ω, constitutes the mesh of the Eulerian domain. In Figure 4.2, the cells of the finite element in 2D, are made of simple triangles. There are two approaches to discretize the domain: structured or unstructured meshes. The structured mesh has cells oriented in a pattern, and is the simplest approach to construct the mesh. The advantage of such a discretization is that it is possible to make a simple data structure which can be used to perform efficient computations. The downside to such discretization is that it is very difficult to construct structured mesh in complex domains with several holes. However,

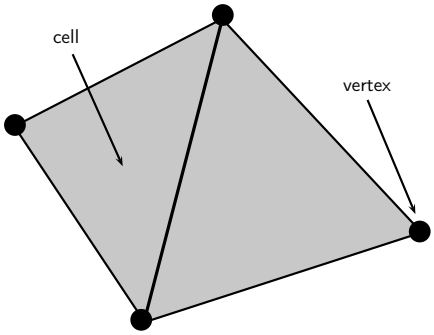


Figure 4.2: A two-dimensional finite element geometry. The cell represents the area of the element, and vertices are the edges of the cell.

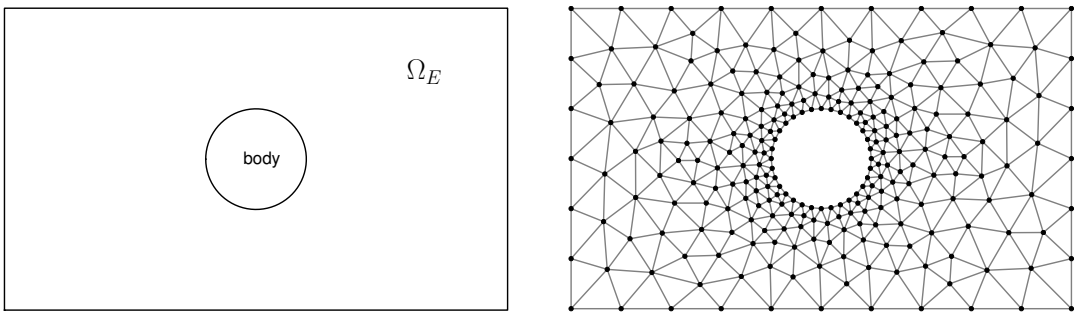
the FEM enables us to perform an unstructured discretization of the domain, as shown in Figure 4.3. The figure shows the unstructured discretization of the fluid domain around the cylinder Ω_E, connecting the rectangular outer boundary of the fluid to the circular no-slip boundary of the body in a simple fashion. Although the unstructured approach gives rise to less efficient discretization, its geometrical flexibility has advantages that surpasses the disadvantage.

There are several algorithms for mesh generation. The standard approach is to employ the Delaunay triangulation method derived from the Voronoi diagram concept [9]. This divides the domain into a set of triangles, as shown in Figure 4.3. This type of mesh generation allows us to connect boundaries with difference shapes together.

4.1.2 Finite Element Functions and Function Spaces

The finite element is defined using a triplet $(T, \mathcal{V}, \mathcal{L})$, as defined in Ciarlet [15] and used in the FEniCS Project [46]. T is a tessellation of the domain Ω, the space $\mathcal{V} = \mathcal{V}(T)$ is a finite dimensional function space on T of dimension n, and $\mathcal{L} = \{\ell_1, \ell_2, ..., \ell_n\}$ is the set of degrees of freedom forming the basis for the dual space $\mathcal{V}'$ of $\mathcal{V}$.

Once we perform the tessellation, we can define the functions and the function spaces of the finite element problem. For each cell, a local function space $\mathcal{V}$ can be defined to



(a) Fluid domain Ω_E around the cylinder **(b)** Delaunay triangulation of the fluid

Figure 4.3: Delaunay triangulation of the fluid around a cylinder resulting in unstructured mesh with controllable cell sizes.

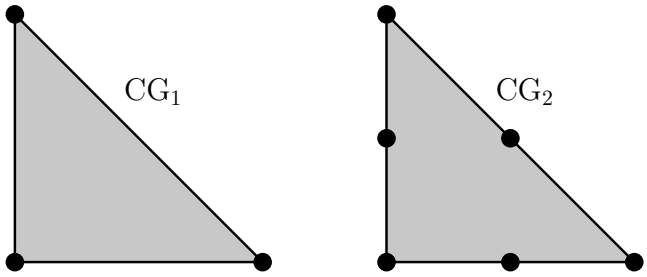


Figure 4.4: The Lagrange CG_q triangle for $q = 1, 2$. The triangles have 3 and 6 DOFs respectively (•, black dot).

collectively construct the global function space V. Any given function $u \in V$ is expressed as a linear combination of basis functions $\{\phi_1, \phi_2, ..., \phi_N\}$, of the function space V,

$$u(x) = \sum_{j=1}^{N} U_j \phi_j(x). \tag{4.1}$$

There are several types of finite element families: the Brezzi-Douglas-Marini, the Crouzeiz-Raviart, the Discontinuous Lagrange, the Hermite, and the Lagrange elements [46]. For the current study, we will rely on the Lagrange elements, also known as the Continuous Galerkin (CG), which are based on the Lagrange polynomials [12]. These elements are widely used and are the simplest to implement for our project.

Lagrange elements belong to the space H^1, which is a Sobolev space containing functions u such that u^2 and $|\nabla u|^2$ have finite integral in the domain Ω [46]. The Lagrange element uses point evaluation for the degrees of freedom, where a DOF in (x_i, y_i) denotes the point evaluation of the function u, $\ell_i(u) = u(x_i, y_i)$. We can have Lagrange elements of various orders $q = 1, 2, ...$, where q is the degree of the Lagrange polynomial $\mathcal{P}_q$. For the 2D case, the dimension n of the finite element is given as,

$$n(q) = \frac{1}{2}(q+1)(q+2). \tag{4.2}$$

For $q = 1$, we have a simple linear Lagrange element CG_1 or linear interpolant with 3 DOFs, known as the Courant triange [23]. For a higher order finite element, we can set $q = 2$, giving us a Lagrange element CG_2 with 6 DOFs per cell. Figure 4.4 shows the two Lagrange triangles CG_1 and CG_2 for $q = 1$ and $q = 2$ respectively. The Courant triangle has the DOFs located at the vertices of the cell, and the higher order CG_2 has 3 additional DOFs, all located midway between the vertices. Our Eulerian method of our hybrid scheme, is based on the CG_1 and CG_2 Lagrange elements.

Variational Formulation

To solve a problem such as the Poisson equation numerically with FEM, we need to convert it into a variational problem. The methodology is followed from the FEniCS tutorial provided by Langtangen [46]. A 1D Poisson problem is given as,

$$\begin{aligned} -\nabla^2 u(x) &= f(x), \qquad x \text{ in } \Omega, \\ u(x) &= u_0(x), \qquad x \text{ on } \partial\Omega. \end{aligned} \tag{4.3}$$

We can transform equation 4.3 into a variational form by multiplying it with a test function v, and integrating it over the domain Ω,

$$-\int_{\Omega} \left(\nabla^2 u\right) v \, \mathrm{d}x = \int_{\Omega} fv \, \mathrm{d}x, \qquad \forall \; v \in \hat{V}. \tag{4.4}$$

In equation 4.4, the function u is the trial function, and is what we are trying to approximate. The trial function u lies in the trial function space V, and the test function v lies in the test function space $\hat{V}$. When performing integration by parts, the test function v is required to be zero at regions where u is known. So, the additional terms cancel and we get,

$$-\int_{\Omega} \nabla u \nabla v \, \mathrm{d}x = \int_{\Omega} fv \, \mathrm{d}x \qquad \forall \; v \in \hat{V} \tag{4.5}$$

Equations 4.4 and 4.5 are referred to as the *weak-form* of the original Poisson equation and is valid for all v in the trial space $\hat{V}$. An inner product of any two functions f and g in domain Ω is defined as,

$$\langle f, g \rangle = \int_{\Omega} fg \, \mathrm{d}x, \tag{4.6}$$

so we can rewrite equation 4.5 as,

$$-\langle \nabla u, \nabla v \rangle = \langle f, v \rangle, \qquad \forall \; v \in \hat{V}. \tag{4.7}$$

In order to solve this continuous problem numerically, we must transform it into a discrete variational problem,

$$-\langle \nabla u_h, \nabla v \rangle = \langle f, v \rangle \qquad \forall \; v \in \hat{V}_h \subset \hat{V}, \tag{4.8}$$

where u_h is the approximate solution function belonging to the discrete function in the discrete space V_h which is a subset of V. Similarly the test discrete function space $\hat{V}_h$ is a subset of $\hat{V}$. The linear triangular element, shown in Figure 4.4 is used as the function space, where $\hat{V}_h$ and V_h are described by piecewise linear functions of the triangle. At the boundary, the functions in the test space are zero, whereas the functions in the trial space are equal to the boundary condition u_0. In the Langtangen [46], u is used to denote/write the solution of the discrete problem, ignoring the subscript h of u_h. Therefore, to have a one-to-one relation with literature, we will employ the same notation from here on. The equation 4.8 can be simplified as,

$$a\left(u, v\right) = L(v), \tag{4.9}$$

where,

$$a\left(u, v\right) = -\langle \nabla u, \nabla v \rangle, \tag{4.10}$$

and

$$L(v) = \langle f, v \rangle. \tag{4.11}$$

The variable $a(u, v)$ and $L(v)$ is the denoted as the bilinear and linear form, respectively. Since, $u \in V$, it can be written as a linear combination of the basis functions $\{\phi_i, ..., \phi_N\}$ of V, with $\text{span}\{\phi_i, ...\phi_N\} = V$, we can express u as,

$$u = \sum_{j=1}^{N} U_j \phi_j. \tag{4.12}$$

Similarly, the test function v can be written as linear combination of basis functions $\{\hat{\phi}_i, ..., \hat{\phi}_N\}$, with span$\{\hat{\phi}_i, ..., \hat{\phi}_N\} = \hat{V}$,

$$v = \sum_{i=1}^{N} V_i \hat{\phi}_i. \tag{4.13}$$

Since equation 4.9 has to valid for all $v \in \hat{V}$, and $\hat{V}$ can be written as a linear combination of the basis functions, equation 4.9 must be valid for each of the basis functions. Therefore, equation 4.9 can be expressed as,

$$a(\sum_{j=1}^{N} U_j \phi_j, \hat{\phi}_i) = L(\hat{\phi}_i), \qquad \forall\ \phi_i, i = 1, ..., N. \tag{4.14}$$

and simplifies to,

$$\sum_{j=1}^{N} U_j a(\phi_j, \hat{\phi}_i) = L(\hat{\phi}_i), \qquad \forall\ \phi_i, i = 1, ..., N. \tag{4.15}$$

This is an algebraic system of equations:

$$\mathbf{A}U = b, \tag{4.16}$$

where $\mathbf{A}_{ij} = a(\phi_j, \hat{\phi}_i)$ and the Right-Hand Side (RHS) b_i is given by $b_i = L(\hat{\phi}_i)$.

4.2 Solving the Finite Element Problem

To solve the finite element problem, we used Dolfin , the finite element library of the FEniCS Project. This library uses high performance linear algebra kernels, and provide a scripting interface to Python. The Python scripting environment helps us to focus on the development of the theory (i.e the high-level algorithms). In order to generate the mesh of the fluid domain, we used Gmsh, a three-dimensional finite element mesh generator which proves a fast, light and user-friendly meshing tool.

4.2.1 Introduction to FEniCS Project

The FEniCS Project is a collaborative work of various universities that developed tools to perform automated finite element algorithms, which can be used to solve partial differential equations. It was a project originated in 2003 with the research collaboration of University of Chicago and Chalmers University of Technology from Logg, Mardal, and Wells [46]. Since then, several other groups have joined such as the Royal Institute of Technology, Simula Research Laboratory, University of Cambridge and Delft University of Technology.

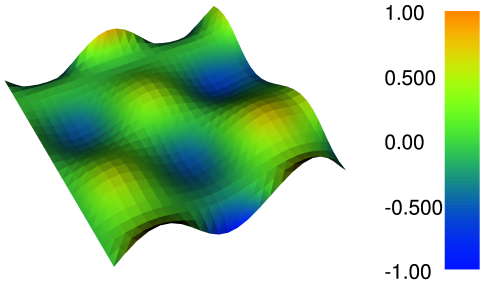


Figure 4.5: DOLFIN VTK plot of the Poisson solution, given by the problem, source code listing 4.1.

FENICS consists of various libraries such as UFC, UFL, FIAT, INSTANT and DOLFIN. DOLFIN is the core library aimed at automating the solution of partial differential equations using the finite element method [47]. It uses automated code generation thus maintaining high-level mathematical expressions but still providing efficient, multi-threaded performance (with Message Passing Interface (MPI)) internally.

We used the DOLFIN library wrapped in PYTHON to set up and solve the finite element problem. For example, we can demonstrate the procedures of solving the Poisson problem, equation 4.3. We can take $f = 2 \cdot \sin x \cdot \cos y$ with the boundary conditions,

$$u(x) = u_0(x) = \sin x \cdot \cos y, \qquad (x, y) \in \partial\Omega. \tag{4.17}$$

The finite element code generation is automated with DOLFIN, leaving only the explicit expression of the problem in python, see source code listing 4.1. Figure 4.5 shows the VTK plot of the solution for the Poisson problem.

4.2.2 Mesh Generation using GMSH

The generation of the mesh is achieved by GMSH, an open-source software developed by Geuzaine & Remacle [29], which has implemented a user-friendly interface and fast algorithms. The GMSH implemented kernels use BLAS and LAPACK linear algebra packages in C++ for fast computation. Furthermore, it allows for scriptability making it ideal to integrate it with our current PYTHON code project for future automation.

```python
from dolfin import *

# Generate unit square mesh: 24 × 24
mesh = UnitSquareMesh(24, 24)

# Define Function space: 1st order, Continuous-Galerkin
V = FunctionSpace(mesh,"CG",1)

# Define Dirichlet boundary conditions expression
# u_0 = sin x · cos y
u0 = Expression("sin(10*x[0])*cos(10*x[1])")

# Function that defines the boundary points
def u0_boundary(x, on_boundary):
    return on_boundary

# Define the boundary condition
# u(x) = u_0(x), x on ∂Ω
bc = DirichletBC(V, u0, u0_boundary)

# Define the variational problem
u = TrialFunction(V)  # Trial functions
v = TestFunction(V)   # Test functions

# f = 100· sin(x)· cos(y)
f = Expression('100*sin(10*x[0])*cos(10*x[1])')

# LHS: a = − ∫ ∇u∇v dx
a = -inner(nabla_grad(u), nabla_grad(v))*dx

# RHS:  L = ∫ fv dx
L = f*v*dx

# Solve the Poisson problem
u = Function(V)       # Define the solution
solve(a == L, u, bc)  # a(u, v) = L(v)

# Plot the result
plot(u, interactive=True)
```

Listing 4.1: A complete program for solving the Poisson problem and plotting the solution. The Poisson problem is given as $-\nabla^2 u = f$, where $u_0 = \sin x \cdot \cos y$ on the boundary and $f = 2 \cdot \sin(x) \cdot \cos(y)$. The code is written in PYTHON using DOLFIN 1.2 library

4.3 Solving Incompressible Navier-Stokes Equations

Using the DOLFIN library for constructing the finite element problem, we can now solve the flow in the Eulerian subdomain of our hybrid scheme. The Eulerian method use the primitive variables velocity-pressure $\mathbf{u} - p$ to describe the flow.

4.3.1 Velocity-Pressure Formulation

The velocity-pressure $\mathbf{u} - p$ formulation of the fluid flow problem, is the standard formulation of the Navier-Stokes equations. The 2D incompressible Navier-Stokes equations of a fluid with unit density (i.e $\rho = 1$) is given as,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \nabla \cdot \sigma = f, \tag{4.18a}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{4.18b}$$

where σ is the Cauchy stress tensor defined as,

$$\sigma(\mathbf{u}, p) = 2\nu\epsilon(\mathbf{u}) - p\mathbf{I}. \tag{4.19}$$

The Cauchy stress tensor is a function of pressure p, the fluid kinematic viscosity ν, and the symmetric gradient ϵ defined as,

$$\epsilon(\mathbf{u}) = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathrm{T}}\right). \tag{4.20}$$

describing the stresses in the fluid due to the velocity gradient and the pressure. The incompressible 2D Navier-Stokes equations have two unknowns, the vector velocity field $\mathbf{u}$, that lies on the vector-valued function space V, and the scalar pressure field p, which lies on the scalar-valued function space Q. Once we compute these two quantities we can determine the vorticity field, which we then transfer to the Lagrangian subdomain.

4.3.2 Determining the Vorticity Field

The coupling between the Eulerian and the Lagrangian subdomain is done by transferring the vorticity field ω from the Eulerian subdomain to the Lagrangian vortex blobs. The vorticity field ω, is defined as:

$$\omega = \nabla \times u, \tag{4.21}$$

where the vorticity ω lies on the scalar-valued function space X. However, as $\nabla \times \mathbf{u} \notin X$, because $\mathbf{u} \in V$, we require a projection from the velocity function space V onto the vorticity function space X. In weak form, this requires the following equality to be satisfied,

$$\int_\Omega \omega \cdot v \, \mathrm{d}x = \int_\Omega (\nabla \times \mathbf{u}) \cdot v \, \mathrm{d}x, \qquad \forall \, v \in \hat{X}. \tag{4.22}$$

To determine the vorticity at every step of the simulation $t_n, t_{n+1}, ...$, we have to perform this projection at every step. Thus, to solve this problem in an efficient manner, we can

pre-assemble (i.e pre-calculated) the knowns of the problem using the `assemble` function of DOLFIN .

Using the inner product rule, defined by equation 4.6, equation 4.22 can be rewritten as,

$$-\langle \nabla\omega, \nabla v\rangle = \langle \nabla \times \mathbf{u}, v\rangle, \qquad \forall\ v \in \hat{X}. \tag{4.23}$$

where $\hat{X}$ is the test function space of the trial function space X. Equation 4.23 can also be written as,

$$a(\omega, v) = L(\mathbf{u}, v), \qquad \forall\ v \in \hat{X}, \tag{4.24}$$

where $a(\omega, v) = -\langle \nabla\omega, \nabla v\rangle$, and $L(\mathbf{u}, v) = \langle \nabla \times \mathbf{u}, v\rangle$. Since ω can be written as a linear combination of basis functions $\{\psi_j, ..., \psi_N\}$, with span$\{\psi_j, ..., \psi_N\} = X$, we can express ω as,

$$\omega = \sum_{j=1}^{N} w_j \psi_j, \tag{4.25}$$

Similarly, v and $\mathbf{u}$ can be expressed in the linear form as $v = \sum_{i=1}^{N} V_i \hat{\psi}_i$ and $\mathbf{u} = \sum_{j=1}^{N} U_j \phi_j$, respectively.

As described in section 4.1.2, since equation 4.24 is valid for all $v \in \hat{X}$, and as $\hat{X}$ can be written as a linear combination of basis functions $\{\hat{\psi}_i, ..., \hat{\psi}_N\}$, equation 4.24 must be valid for each of the basis functions. Therefore, equation 4.24 can be expresses in the linear form as,

$$a(\sum_{j=1}^{N} w_j \psi_j, \hat{\psi}_i) = L(\sum_{i=j}^{N} U_j \phi_j, \hat{\psi}_i), \qquad \forall\ \hat{\psi}_i,\ i = 1, ...N, \tag{4.26}$$

and simplifying to,

$$\sum_{j=1}^{N} w_j \cdot a(\psi_j, \hat{\psi}_i) = \sum_{i=j}^{N} U_j \cdot L(\phi_j, \hat{\psi}_i), \qquad \forall\ \hat{\psi}_i,\ i = 1, ...N, \tag{4.27}$$

resulting in an algebraic system of equations,

$$\mathbf{A}w = b, \tag{4.28}$$

where $\mathbf{A}_{ij} = a(\psi_j, \hat{\psi}_i)$, and the Right-Hand-Side (RHS) is given as $b_i = L(\phi_i, \hat{\psi}_i)$. Since $\mathbf{A}$ does not change during the simulation, it can be pre-computed outside the time-marching loop, using the `assemble` function of DOLFIN , improving the efficiency of the vorticity calculation.

The PYTHON implementation of the algorithm is show in listing 4.2. Using the DOLFIN library, we can used the `assemble` function to pre-calculated the LHS of the problem (line 24). So using the algorithms of the hybrid coupling scheme, we can transfer this vorticity field of the Eulerian subdomain on the vortex blobs.

```
1  from dolfin import *
2
3  ...
4
5  # Define Function spaces
6  # X : scalar-valued vorticity function space W
7  X = FunctionSpace(mesh, 'CG', 1) # 1st order, Continuous-Galerkin
8
9  # Define the trial and test function
10 omega = TrialFunction(X) # vorticity ω∈X
11 v     = TestFunction(X)  # test function v∈X̂
12
13 ...
14
15 # Define the variation problem for vorticity
16 a = inner(omega,v)*dx    #  ⟨ω, v⟩
17 b = inner(curl(u),v)*dx  #  ⟨∇×u, v⟩
18
19 # Pre-Assemble the LHS
20 A = assemble(a)
21
22 ...
23
24 # During the time-stepping
25 omega = Function(X) # Define the function
26 B = assemble(b)     # Assemble b
27 solve(A, omega.vector(), B) # Solve for vorticity
```

Listing 4.2: The PYTHON implementation of the vorticity calculation using DOLFIN 1.2 library. Line 24 shows the use of `assemble` function to pre-assemble the knowns of the problem.

4.3.3 Taylor-Hood Finite Element Family for Solving ICNS

To solve the Incompressible Navier-Stokes (ICNS) problem, we must choose appropriate finite element function spaces for the velocity $\mathbf{u}$ and the pressure p by ensuring that we satisfy the Ladyzhenskaya-Babuška-Brezzi (LBB) compatibility condition, also known as the inf-sup compatibility condition, described in Brezzi and Fortin [8]. The Lagrange finite element space for velocity must be one order higher than the order of the pressure q_{pres},

$$q_{\text{vel}} = q_{\text{pres}} + 1, \tag{4.29}$$

for a stable formulation. Therefore, we have decided to use the Taylor-Hood family, introduced by Taylor and Hood [62] and verified by Boffi [5], that satisfies the inf-sup compatibility condition by using $q_{\text{vel}} = 2$ and $q_{\text{pres}} = 1$. We decided to choose this method, as it is the most conventional method, that is simple, and shows a stable behavior.

In addition, we have to choose an appropriate function space for the vorticity. As vorticity is the curl of the velocity, to reduce interpolation error during the projection of the solution, we will use a function space one order lower than the velocity, $q_{\text{vort}} = 1$. Table 4.1 shows the list of the function spaces, the finite element type and their orders. Additionally, we have included the variable names of the function space, trial functions and the test

functions, associated to the function element that we have chosen for the problem.

Table 4.1: Summary of the Lagrange element CG_q of order q, that was used for solving the incompressible Navier-Stokes problem. The variable names of the function space, the trial functions, and the test functions are tabulated together.

Variable	Finite element	Function space	Trial function	Test function
Velocity	CG_2	V	$\mathbf{u}$	$\mathbf{v}$
Pressure	CG_1	Q	p	q
Vorticity	CG_1	X	w	x

4.3.4 Incremental Pressure Correction Scheme

This algorithm to solve the NS problem was first demonstrated by Chorin in 1968 [13], and is referred to as Chorin's projection method or sometimes known as the non-incremental pressure correction scheme. The process relies in first computing a tentative velocity by initially neglecting the pressure in the momentum equation of the Navier-Stokes problem, equation 4.18. The velocity field is corrected by determining the pressure field satisfying a divergence free vector field. This method however does not satisfy the discrete incompressibility constraint exactly and so, Goda in 1979 [30], introduced an improved Incremental Pressure Correction Scheme (IPCS). The method computed the viscous term at the incremented time $(t_{n-1} + t_n)/2$, and used the stress formulation to determine the corrected pressure [46]. The detailed algorithms to the IPCS scheme, as presented in the FEniCS manual [46], can be summarized as follows:

1. **Compute the tentative velocity:** The tentative velocity $\mathbf{u}^\star$ is determined by solving,

$$\begin{aligned}\langle D_t^n \mathbf{u}^\star, \mathbf{v}\rangle + \langle \mathbf{u}^{n-1} \cdot \nabla \mathbf{u}^{n-1}, \mathbf{v}\rangle + \langle \sigma(\mathbf{u}^{n-\frac{1}{2}}, p^{n-1}), \epsilon(\mathbf{v})\rangle \\ + \langle p^{n-1}\hat{\mathbf{n}}, \mathbf{v}\rangle_{\partial\Omega} - \langle \mathbf{v}\ \hat{\mathbf{n}} \cdot (\nabla \mathbf{u}^{n-\frac{1}{2}})^{\mathrm{T}}, \mathbf{v}\rangle_{\partial\Omega} = \langle f^n, \mathbf{v}\rangle,\end{aligned} \tag{4.30}$$

   is valid for all $\mathbf{v} \in V$, where $\mathbf{u}^{n-\frac{1}{2}}$ is defined as,

$$\mathbf{u}^{n-\frac{1}{2}} = \frac{\mathbf{u}^\star + \mathbf{u}^{n-1}}{2}, \tag{4.31}$$

   With the Dirichlet velocity boundary conditions at the boundary $\partial\Omega$, we can solve equation 4.30. The additional term,

$$\langle \mathbf{v}\ \hat{\mathbf{n}} \cdot (\nabla \mathbf{u}^{n-\frac{1}{2}})^{\mathrm{T}}, \mathbf{v}\rangle_{\partial\Omega}, \tag{4.32}$$

   is results from integration by parts, when we evaluate the viscous term at $(t_{n-1} + t_n)/2$ and we use the stress formulation instead of the Laplacian formulation as done for the Chorin scheme. This difference ensures that the velocity profile at the inlet and the outlet of the domain is more accurate that the ones obtained for the Chorin scheme.

```
1  # Before the time-stepping:
2
3  # Define: u^{n-1/2} = (u* + u^{n-1})/2
4  U  = 0.5*(u0 + u)
5
6  # Formulate the tentative velocity problem
7  F1 = (1/k)*inner(v, u - u0)*dx \
8             + inner(v, grad(u0)*u0)*dx \
9             + inner(epsilon(v), sigma(U, p0, nu))*dx \
10            + inner(v, p0*n)*ds \
11            - beta*nu*inner(grad(U).T*n,v)*ds \
12            - inner(v, f)*dx
13
14 # Extract the LHS, and the RHS
15 a1 = lhs(F1)
16 L1 = rhs(F1)
17
18 # Pre-assemble the LHS
19 A1 = assemble(a1)
20
21 ...
22
23 # During the time-stepping:
24
25 # Assemble the RHS
26 b = assemble(L1)
27
28 # Apply the Dirichlet velocity boundary condition b.c
29 [bc.apply(A1, b) for bc in bcVelocity]
30
31 # Solve for the Tentative velocity
32 solve(A1, u1.vector(), b, "gmres", "default")
```

Listing 4.3: The source code for solving the tentative velocity $\mathbf{u}^\star$, using the equation 4.30.

The source code for solving the tentative velocity problem is shown in listing 4.3. First, we pre-define all the terms needed for the tentative velocity problem formulation (lines 3 to 16). We can also pre-assemble the LHS of the problem (line 19) outside of the time-integration loop, since it remains constant. During time integration, we first assemble the RHS of the problem (line 26), then apply the Dirichlet velocity boundary condition (line 29) which consists of the wall boundary condition, and external Dirichlet velocity boundary condition (e.g. the free-stream). Finally, we can solve the problem using a GMRES solver for solving the system of linear equation (line 32).

2. **Determine the pressure:** The pressure p^n is determined by solving,

$$\langle \nabla p^n, \nabla q \rangle = \langle \nabla p^{n-1}, \nabla q \rangle - \langle \nabla \cdot \mathbf{u}^\star, q \rangle / \Delta t_n \tag{4.33}$$

valid for all $q \in Q$. We use the previously calculated tentative velocity $\mathbf{u}^\star$ to determine the pressure. We can solve the problem using the Neumann pressure boundary condition at the pressure outlet of the domain. We define a boundary as the pressure outlet, if we do not know the velocity boundary condition at that boundary. This is true for the region where the exit flow is perturbed. However, for the coupled Eulerian method (that we will use), all the boundary conditions are available as a velocity boundary condition from the Lagrangian subdomain. This means that we do not have to assume any pressure boundary condition.

```
1  # Before the time-stepping:
2
3  # Formulate the pressure correction problem
4  a2 = inner(grad(q), grad(p))*dx          # ⟨∇q, ∇p^n⟩
5  L2 = inner(grad(q), grad(p0))*dx\        # ⟨∇ q, ∇ p^{n-1} ⟩ − ⟨ ∇ · u*, q⟩/Δ t_n
6            - (1/k)*q*div(u1)*dx
7
8  # Pre-assemble the LHS
9  A2 = assemble(a2)
10
11 ...
12
13 # During the time-stepping:
14
15 # Assemble the RHS
16 b = assemble(L2)
17
18 # Apply the Dirichlet velocity boundary condition b.c
19 if len(bcPressure) == 0: normalize(b)
20 [bc.apply(A2, b) for bc in bcPressure]
21
22 # Solve for the corrected pressure
23 solve(A2, p1.vector(), b)
24 if len(bcPressure) == 0: normalize(p1.vector())
```

Listing 4.4: The source code for solving the pressure p^n using the equation 4.33.

The source code for solving the pressure problem is shown in listing 4.4. As done for the tentative velocity, we can formulate and pre-assemble the problem before the time loop (lines 3 to 9). In the time loop, we only need to assemble the RHS (line 16), apply the boundary condition (if it exists, lines 18 to 20) and finally solve for the pressure (lines 22 to 24). Using the pressure, we can determine the corrected velocity field.

3. **Determine the corrected velocity:** The corrected velocity field u^n is determined by solving,

$$\langle \mathbf{u}^n, \mathbf{v} \rangle = \langle \mathbf{u}^\star, \mathbf{v} \rangle - \Delta t_n \langle \nabla (p^n - p^{n-1}), \mathbf{v} \rangle, \tag{4.34}$$

which is valid for all $\mathbf{v} \in V$. We correct the tentative velocity $\mathbf{u}^\star$ by the pressure difference to determine the correct velocity field. We will have to apply the Dirichlet velocity boundary condition at the boundary again, to solve for the problem.

The source code for solving the corrected velocity problem in shown in listing 4.5. We first initialize the problem, by formulating the problem and assembling the LHS outside the time loop (line 3 to 8). In the time integration loop, we assemble the RHS (line 15), apply the velocity boundary condition (line 18) and finally solve for the corrected velocity field (line 21).

The algebraic system of equations resulting form this algorithm have been solved with Dolfin's Krylov Gmres solver with an absolute and a relative error tolerance of 10^{-25} and 10^{-12} respectively. The program structure was based on the collection of benchmark solvers provided by the FEniCS [45]. In the algorithm described above an explicit time marching scheme, Forward Euler (FE) has been used. Therefore, for the time marching

```
# Before the time-stepping:

# Formulate the velocity correction problem
a3 = inner(v, u)*dx      # ⟨u^n, v⟩
L3 = inner(v, u1)*dx - k*inner(v, grad(p1 - p0))*dx # ⟨u*, v⟩ − Δt_n⟨∇(p^n − p^{n−1}), v⟩

# Pre-assemble the LHS
A3 = assemble(a3)

...

# During the time-stepping:

# Assemble the RHS
b = assemble(L3)

# Apply the Dirichlet velocity boundary condition b.c
[bc.apply(A3, b) for bc in bcVelocity]

# Solve for the corrected pressure
solve(A3, u1.vector(), b, "gmres", 'default')
```

Listing 4.5: The source code for solving the corrected velocity u^n using equation 4.34.

scheme to be stable, we require the CFL number to satisfy the following condition:

$$\mathrm{CFL} = \Delta t_{E,\max} \frac{\|\mathbf{u}\|_{\max}(\nu + \Delta h_{\min}\|\mathbf{u}\|_{\max})}{\Delta h_{\min}^2} \leqslant 1. \tag{4.35}$$

This gives us the direct constraint on the maximum Eulerian time step size $\Delta t_{E,\max}$ which is function of the CFL number, maximum fluid velocity in the Eulerian domain $\|\mathbf{u}\|_{\max}$, the fluid viscosity ν and the minimum mesh cell size Δh_{min}.

4.3.5 Determining the Body Forces

After we determine the flow fields, we can compute the lift and the drag generated by the body. To determine these parameters, we first need to determine the forces acting on the no-slip boundary, which can be determined from the stress tensor σ acting on the surface of the body, equation 4.19. The lift coefficient and the drag coefficient are computed by:

$$L = \int_{\partial\Omega} [\sigma(\mathbf{u}, p) \cdot \hat{\mathbf{n}}] \cdot \hat{\mathbf{e}}_y \, \mathrm{d}s, \tag{4.36a}$$

$$D = \int_{\partial\Omega} [\sigma(\mathbf{u}, p) \cdot \hat{\mathbf{n}}] \cdot \hat{\mathbf{e}}_x \, \mathrm{d}s, \tag{4.36b}$$

where $\hat{\mathbf{e}}_x$ and $\hat{\mathbf{e}}_y$ are the 2D unit Cartesian vectors,

$$\hat{\mathbf{e}}_x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \hat{\mathbf{e}}_y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{4.37}$$

such that lift perpendicular to the free-steam and the drag is tangential to it. The lift coefficient C_l and the drag coefficient C_d, are obtained by normalizing the lift L and

drag D forces with the dynamics pressure and reference length c (in 2D), where the lift perpendicular to the free-steam and the drag is tangential to it,

$$C_l = \frac{L}{\frac{1}{2}\|\mathbf{u}\|_\infty^2 c}, \qquad C_d = \frac{D}{\frac{1}{2}\|\mathbf{u}\|_\infty^2 c}. \tag{4.38}$$

```
...

def epsilon(u):
    "Returns symmetric gradient"
    return 0.5*(grad(u) + grad(u).T)

def sigma(u,p,nu):
    "Returns stress tensor"
    return 2*nu*epsilon(u) - p*Identity(u.cell().d)

# Define the normal function
n = FacetNormal(mesh)

# Define the unit vectors
eX = Constant((1.0, 0.0))
eY = Constant((0.0, 1.0))

# Define the line integrator
ds = Measure("ds")[boundaryDomains]
noSlip = 2 # No-slip boundary identification = 2

# Determine the forces
# Integrate the forces over the boundaryDomain == noSlip
L = assemble(inner(inner(sigma(u,p,nu), n), eY)*ds[noSlip]) # Lift
D = assemble(inner(inner(sigma(u,p,nu), n), eY)*ds[noSlip]) # Drag
```

Listing 4.6: The PYTHON implementation for calculating the lift force L and the drag force D acting on the no-slip boundary.

4.4 Evolution of the Eulerian method

The algorithm of evolving the Eulerian method is summarized in this section. The Eulerian method acts as the source of the vorticity for the Lagrangian method.

The flowchart of the Eulerian method is given by Figure 4.6.The algorithm to the Eulerian method can be summarized as follows:

1. **Mesh generation**: We generate the mesh of the fluid domain using GMSH before the iteration.
2. **Determine the boundary condition**: We determine the boundary conditions for the boundary domains: $\partial\Omega_E = \Sigma_w \cup \Sigma_d$. The Dirichlet velocity boundary condition is obtained directly the Lagrangian method and the Neumann pressure boundary condition is obtained from the velocity. This is possible as we known the complete velocity distribution at all $\partial\Omega_E$.

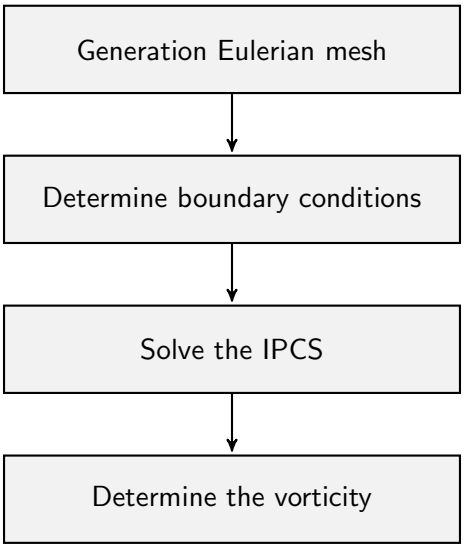


Figure 4.6: Flowchart of the Eulerian method.

3. **Solve the IPCS**: Using IPCS, time march from t_n to t_{n+1} to solve for the new velocity $\mathbf{u}$ and pressure p field.

4. **Determine the vorticity**: Using the algorithm described in 4.3.2, solve for the vorticity field ω at the time t_{n+1}.

Once we have the well-resolved vorticity ω of the near-body region, the vorticity is then transfered into the Lagrangian subdomain using the Hybrid coupling scheme, which was summarized in chapter 2, and fully elaborated in chapter 5.

4.5 Validation of Eulerian Method

To verify our Eulerian method, we first investigated the Lamb-Oseen vortex problem. The validation of the Eulerian method is done by investigating the Clercx-Bruneau dipole collision at $Re = 625$, comparing with the study of Clercx and Bruneau [16]. Furthermore, we investigated the problem of the impulsively started cylinder at $Re = 550$, where we verified and validated the lift and drag calculations with the literature of Koumoutsakos and Leonard [41], and Rosenfeld et al. [54].

4.5.1 Lamb-Oseen Vortex

The Lamb-Oseen vortex is an analytical solution by Lamb and Oseen, describing the diffusion of a vortex core [63]. The current investigation was performed similarly to the Lagrangian study in section 3.7.2.

Problem Definition

In section 3.7.2, the vortex blobs of the Lagrangian method required the vorticity formulation of the Lamb-Oseen vortex to set up the problem. However, as the Eulerian method use the primitive variables $\mathbf{u} - p$, we require the velocity formulation of the Lamb-Oseen vortex. The velocity field of the Lamb-Oseen vortex is defined as,

$$u_\theta = \frac{\Gamma_c}{2\pi r}\left[1 - \exp\left(-\frac{r^2}{4\pi\nu(t+\tau)}\right)\right] \tag{4.39a}$$

$$u_r = 0, \tag{4.39b}$$

where u_θ is the circumferential velocity, and u_r is the radial velocity. The velocity is a function of the vortex core strength Γ_c, the simulation time $t \in [0, \infty[$, the time constant τ, the kinematic viscosity ν, and the distance from the core center r.

The parameters used for the simulation are tabulated in table 4.2. To ensure a valid comparison of the present Eulerian method study with the Lagrangian method performed in section 3.7.2, we chose similar spatial discretization parameters.

Figure 4.7 depicts the domain for the present Lamb-Oseen vortex study. The fluid domain Ω_E is bounded by a Dirichlet boundary Σ_d such that $\partial\Omega_E = Sigma_d$. The Dirichlet boundary Σ_d is used to prescribe the analytical Dirichlet velocity boundary conditions for the Lamb-Oseen vortex problem.

The Eulerian method is time marched using the time integration parameters tabulated in table 4.2. To ensure a stable time integration, we enforced the CFL conditions, equation 4.35. During the evolution of the solution, we evaluated the growth of the error in velocity, and the error in vorticity between the numerical and the analytical solutions.

Table 4.2: Summary of the parameters for the Lamb-Oseen vortex evolution.

Parameters	Value	Description
Γ_c	1	Core strength
Ω	$[-1,1] \times [-1,1]$	Eulerian domain bounds
$\mathbf{u}_\infty$	0	Free-stream velocity
ν	5×10^{-4}	Kinematic viscosity
τ	4	Lamb-Oseen time offset
t	0 to 1	Simulation time span
Δt	0.001	Time step size
N_t	1000	Number of time integration steps
h_{grid}	$\approxeq 0.01$	Minimum mesh cell size
N_{cells}	161312	Number of mesh cells
CFL	0.95	CFL number
$\|\mathbf{u}\|_{\max}$	1.5	Maximum magnitude of the velocity
Time integration	FE	Forward Euler time integration method

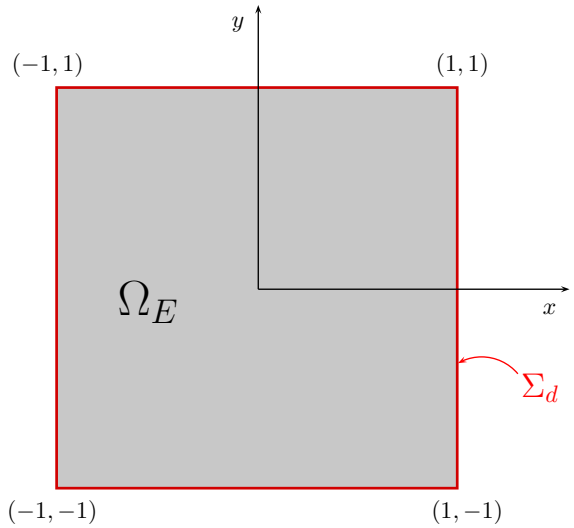


Figure 4.7: Eulerian domain Ω_E (**gray**) for the Lamb-Oseen vortex problem, bounded by the Dirichlet velocity boundary Σ_d (**red**), where the Dirichlet velocity boundary condition was applied. The parameters of the domain are tabulated in table 4.2.

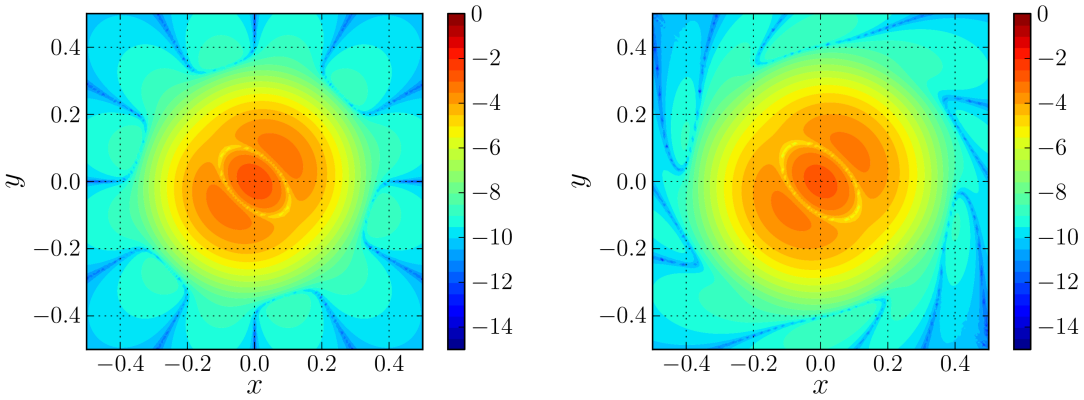
Results and Discussion

We are interested in the evolution of error in vorticity, as this is the quantity which will be interpolated onto the Lagrangian subdomain. Figure 4.8 shows the initial and the final relative error in vorticity over the Eulerian domain. Opposed to the Lagrangian results, Figure 3.20, we see that initial relative error in the vorticity field is larger. This is so because the Eulerian solution was initialized using the velocity and the vorticity is obtained by projecting $\nabla \times \mathbf{u}$ (see section 4.3.2) onto the scalar-valued function space W, whereas the Lagrangian solution was initialized directly from vorticity. This process of initialization in the Eulerian domain introduces additional numerical error in the vorticity. However, the pattern of the relative error in vorticity is similar to the Lagrangian solution, with the highest error at the core center, where we have stronger vorticity.

As time progresses, we see that the error in the solution does not increase as observed for the Lagrangian method, Figure 4.8b. Figure 4.9 shows this change in the maximum relative error in velocity and vorticity. The maximum relative error in vorticity is at all times higher than the maximum relative error in velocity, due to the error in projection.

To determine the convergence with spatial resolution, the simulation was run for $h \approx 0.25$ to $h \approx 5 \times 10^{-3}$. Figure 4.10a shows the convergence of the relative error in vorticity. This validates that the scheme is 2^{nd}-order in space, which is expectable due to the second order function space CG_2 for the primitive variable, velocity.

To determine convergence with time resolution, we ran the simulation with various time steps $\Delta t = 5 \times 10^{-3}$ to $\Delta t = 1 \times 10^{-4}$. As we performed the investigation, we saw that the error in the primitive variable $\mathbf{u}$, converged with order 1, Figure 4.10. This is expected, as have employed a 1^{st}-order Forward Euler scheme. Thus, we have verified against the analytical solution of the Lamb-Oseen vortex that our Eulerian method is well implemented and performs in a robust manner.



(a) $t = 0$ **(b)** $t = 1$

Figure 4.8: Relative error in vorticity (in logarithmic scale) using the parameters tabulated in table 4.2. The figure shows **(a)**, the initial relative error in vorticity at $t = 0$, and **(b)** the final relative error in vorticity at $t = 1$.

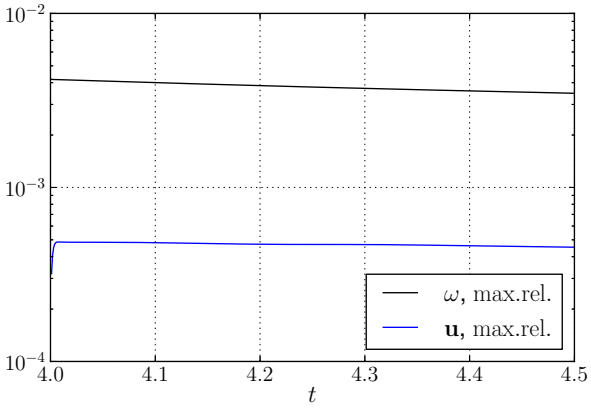
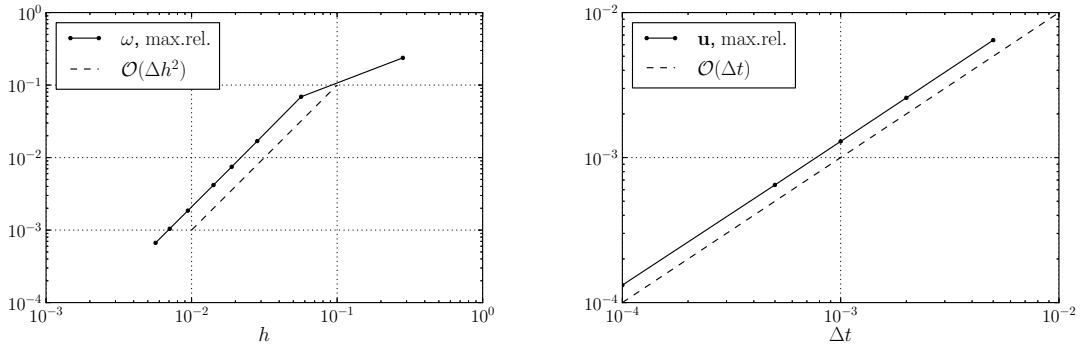


Figure 4.9: Evolution of the maximum relative errors from $t = 0$ to $t = 1$ using the parameters in table 4.2. The figure depicts maximum relative error in velocity [—, solid **blue**] and the maximum relative error in vorticity [—, solid **black**].



(a) Spatial convergence **(b)** Temporal convergence

Figure 4.10: Convergence in space and time. The figure depicts **(a)** convergence in space of $\mathcal{O}(\Delta h^2)$ and **(b)** convergence in time of $\mathcal{O}(\Delta t)$. The control parameters are tabulated in table 4.2.

4.5.2 Clercx-Bruneau Dipole Collision at $Re = 625$

The Eulerian method that we have developed here is to be used as a wall-bounded Eulerian solver that can resolve the vorticity production at the boundary for the Hybrid method. Therefore it is vital that the vortex interaction with the no-slip boundary is handled properly.

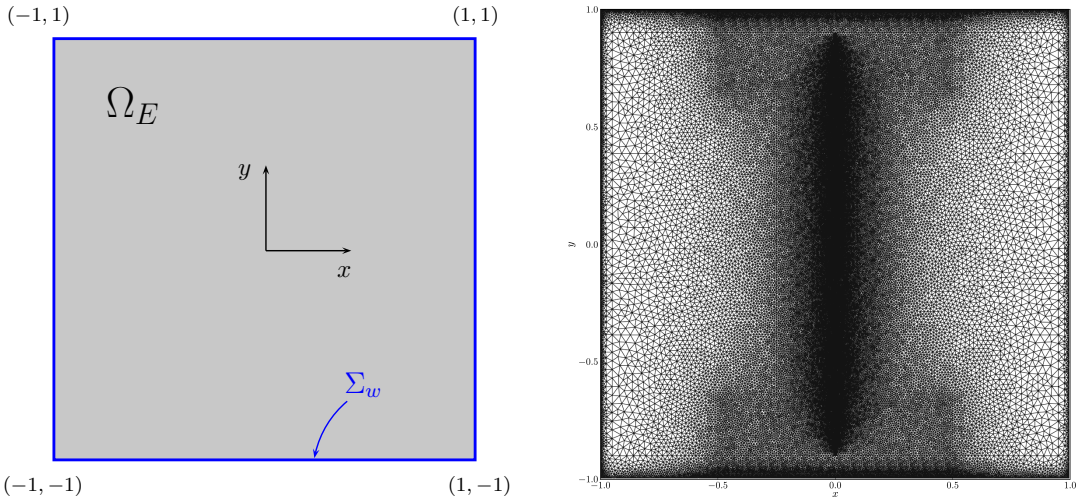
To determine the proper handling of the no-slip boundary, it is common practice to use a simple test of dipole colliding with the wall. In these test cases, one could observe how the no-slip boundary handles the incoming vortex and can be used to determine if the system is formulated appropriately. Ould-Salihi et al. [51] used this case to validate their Hybrid method that couples vortex particles with finite-difference method. Cottet et al. [21] used this test case to validate the vortex method. Therefore in a similar fashion, we have decided to use the dipole collision study by Clercx and Bruneau [16], performed using a Chevyshev pseudo-spectral method, to verify and validate our Eulerian method.

Problem Definition

Unlike other dipole test cases, Clercx and Bruneau provide well-defined initial and boundary conditions for the dipole vorticity field. Furthermore, they used a vorticity distribution that was continuous, which ensures a smooth velocity field for our Eulerian method using the $\mathbf{u} - p$ formulation. The literature provided results for the collision that we are interested: a normal collision with the dipole traveling perpendicular to the wall.

For the present study, we have decided to use the simpler case of $Re = 625$, where Re is the integral-scale Reynolds number defined as,

$$Re = \frac{UW}{\nu}, \tag{4.40}$$



(a) (*Not to scale*) Domain definition **(b)** Domain mesh

Figure 4.11: Domain of the Clercx-Bruneau dipole collision problem. The figure depicts **(a)** the definition of the domain with the fluid domain (**gray**) and the no-slip boundary (**blue**); and **(b)** the unstructured mesh of the domain with $N_{\text{vert}} = 48k$.

where U is the characteristic velocity of the flow, W is half width of domain, and ν is the kinematic viscosity and are chosen according to Clercx and Bruneau [16]. A low Reynolds number of $Re = 625$ is required as the Eulerian method currently only solves an incompressible laminar flow. However in future, an implementation of a turbulence method can enable a high Reynolds number investigation.

The domain Ω of the problem is square, $\Omega_E = [-1, 1] \times [-1, 1]$, as shown in Figure 4.11a. The problem is defined in a closed box, where the Eulerian domain is enclosed in a no-slip boundary $\partial\Omega = \Sigma_w$ where dipole collides and interacts.

The initial condition of the Clercx-Bruneau dipole is a smooth dipole velocity distribution with a positive monopole at $(x_1, y_1) = (0.1, 0)$ and the negative monopole at $(x_2, y_2) = (-0.1, 0)$, with each having a core radius $R = 0.1$. The velocity distribution for the combined monopole at $t = 0$ is given as,

$$u(\mathbf{x}, 0) = -\frac{1}{2}\omega_e(y - y_1)\exp\left\{-\left(\frac{r_1}{R}\right)^2\right\} + \frac{1}{2}\omega_e(y - y_2)\exp\left\{-\left(\frac{r_2}{R}\right)^2\right\}, \tag{4.41a}$$

$$v(\mathbf{x}, 0) = +\frac{1}{2}\omega_e(x - x_1)\exp\left\{-\left(\frac{r_1}{R}\right)^2\right\} - \frac{1}{2}\omega_e(x - x_2)\exp\left\{-\left(\frac{r_2}{R}\right)^2\right\}, \tag{4.41b}$$

where u and v are the velocities in the x and y direction, respectively. The characteristic vorticity magnitude $\omega_e = 299.528385375226$ is obtained from Renac et. al [53]. The radii r_1 and r_2 are the radial distances to the point $\mathbf{x}$ from the center of positive and the negative monopoles respectively. The corresponding vorticity distribution for the velocity distribution is given as,

$$\begin{aligned}\omega(\mathbf{x}, 0) = \quad & \omega_e\left[1 - \left(\frac{r_1}{R}\right)^2\right]\exp\left\{-\left(\frac{r_1}{R}\right)^2\right\} \\ & - \omega_e\left[1 - \left(\frac{r_2}{R}\right)^2\right]\exp\left\{-\left(\frac{r_1}{R}\right)^2\right\}.\end{aligned} \tag{4.42}$$

The Eulerian domain was discretized using an unstructured meshing method in GMSH (section 4.2.2), as shown in Figure 4.11a. The velocity distribution, equation 4.41, shows that the maximum velocity in the fluid will be along the y-axis (i.e $x = 0$). Therefore, to satisfy the CFL condition, we need the minimum cell size at the location of the maximum velocity. The resolution of the mesh in the region where the dipole and the wall interacts (i.e $-0.5 \leq x \leq 0.5$ and $0.5 \leq |y| \leq 1$) was also increased. Furthermore, in the region where there is no vorticity, we do not need high resolution (i.e $0.5 \leq |x| \leq 1$ and $-0.5 \leq y \leq 0.5$).

After initializing the velocity field in the discretized domain, the problem was evolved from $t = 0$ to $t = 2$. The time integration parameters are tabulated in table 4.3.

Results and Discussion

Figure 4.12 shows the evolution of the vorticity field at various instances, $t = [0, 0.25, 0.5, 0.75, 1.25]$. During the initial stages of the simulation, the initialized dipole travels along the y-axis towards the bottom no-slip boundary.

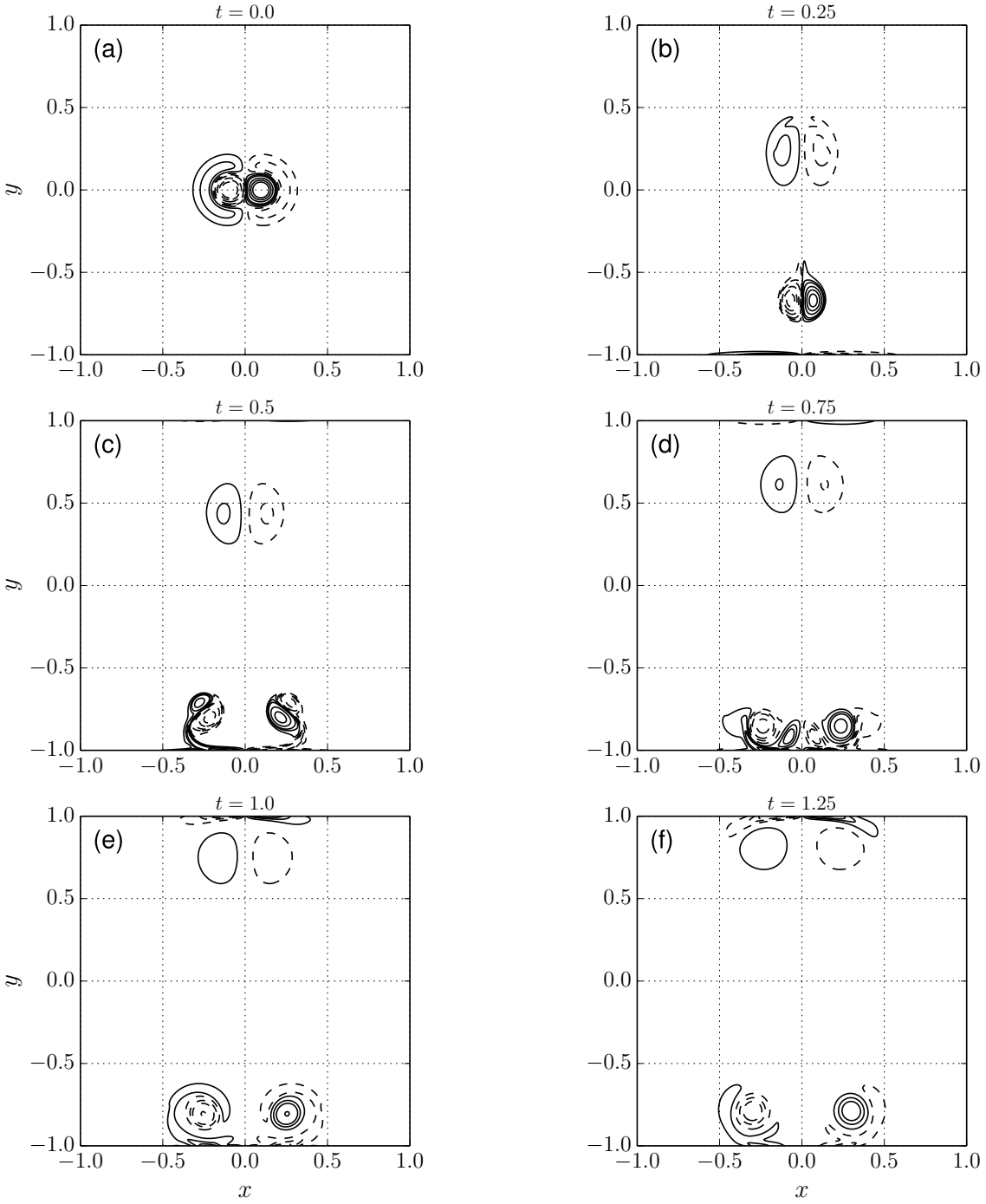


Figure 4.12: Vorticity contour plots of the Clercx-Bruneau dipole-wall collision at $Re = 625$ and $t = [0, 0.25, 0.5, 0.75, 1.0, 1.25]$ with vorticity contour levels at $[-320, -200, -100, -50, -10, 10, 50, 100, 200, 320]$. The figure depicts positive contours [—, solid **black**], and negative contours [- -, dashed **black**].

Table 4.3: Summary of the parameters for the Clercx-Bruneau dipole collision with a no-slip wall [16].

Parameters	Value	Description
Ω	$[-1,1]\times[-1,1]$	Eulerian domain bounds
Re	625	Reynolds number
U	1^a	Characteristic velocity
W	1^a	Half width of the domain
ν	1.6×10^{-3}	Kinematic viscosity
$(x,y)_{1,2}$	$(\pm0.1,0)$	Initial location of the monopoles
ω_e	299.5283853752226^b	Characteristic vorticity of the monopole
t	0 to 2	Simulation time span
CFL	0.95	CFL number
$\|\mathbf{u}\|_{\max}$	12	Maximum fluid velocity
Δt	1.25×10^{-5}	Time step size
N_{cells}	96142	Number of FE mesh cells
h_{grid}	3.6×10^{-3} to 5.1×10^{-2}	FE mesh cell size
N_t	$160,000$	Number of time integration steps

[a] Obtained from Clercx and Bruneau [16]
[b] Obtained from Renac et al. [53]

The dipole approaches the bottom boundary, where the no-slip boundary generates vorticity to ensure no-through flow, seen in Figure 4.12b. As the dipole approaches the wall, the vorticity filament at the wall rolls up and combines with the primary dipole forming two secondary dipoles, that is symmetric across the y-axis. Figure 4.12c shows the state of the vorticity field at $t = 0.5$ after the secondary dipoles are generated. This secondary dipole initially travels away from the bottom wall and later on approaches the wall again, colliding for a second time and creating a tertiary vortex, Figure 4.12d. The dipole stops convecting any further and diffuses as time progresses, as shown in Figure 4.12e and 4.12f, corresponding to $t = 1$ and $t = 1.25$.

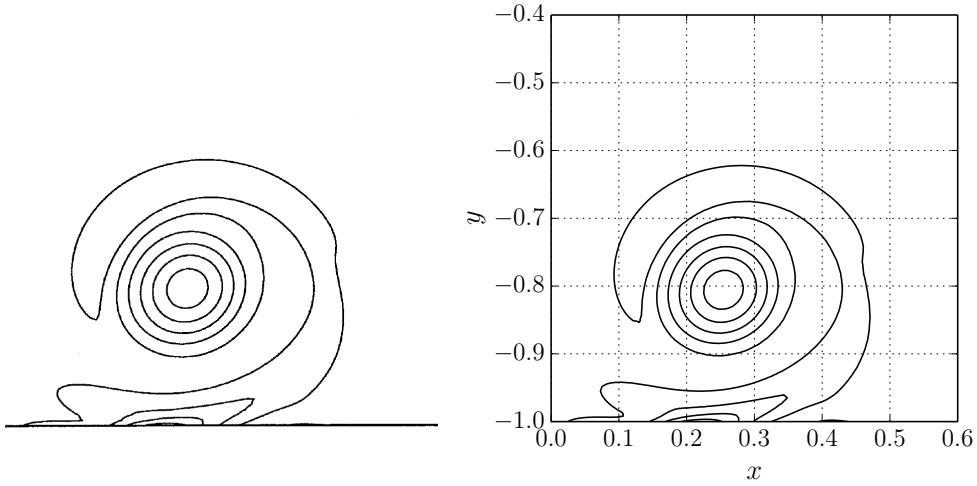
Figure 4.13 compares the vorticity contours of the computational domain close to the bottom wall, $0 \leqslant x \leqslant 0.6$ and $-1 \leqslant y \leqslant -0.4$, at $t = 1$. The positive vortex (solid black) is surrounded by the negative vortex (dashed black). Comparing the present study, Figure 4.13b with the study of Clercx and Bruneau [16], Figure 4.13a, shows that the shows that the shape of the vorticity contours is indistinguishable. This means that the present study matches very well with the pseudo-spectral simulation of the Clercx and Bruneau.

To determine the variation of the fluid properties as time progresses, Clercx and Bruneau investigated the evolution of the total kinetic energy E, the total enstrophy Ω, and the total palinstrophy P of the flow field. The total kinetic energy E of the flow can be determined with,

$$E(t) = \frac{1}{2}\int\int \mathbf{u}^2(\mathbf{x},t)\ \mathrm{d}x\mathrm{d}y, \tag{4.43}$$

and at $t = 0$, we have $E(0) = 2$. The total enstrophy of the flow is determined as,

$$\Omega(t) = \frac{1}{2}\int\int \omega^2(\mathbf{x},t)\ \mathrm{d}x\mathrm{d}y. \tag{4.44}$$



(a) Literature **(b)** Present study

Figure 4.13: Comparison of the vorticity contours at $t = 1$ with contour levels $[..., -50, -30, -10, 10, 30, 50, ...]$. The figure compares the plot obtained by **(a)** literature of Clercx and Bruneau [16] and **(b)** the present study.

The change in enstrophy of the flow field can give an insight into the dissipation rate in the fluid. At $t = 0$, the total enstrophy of the fluid is $\Omega(0) = 800$. The total palinstrophy P of the flow measures the gradient of vorticity and is given by,

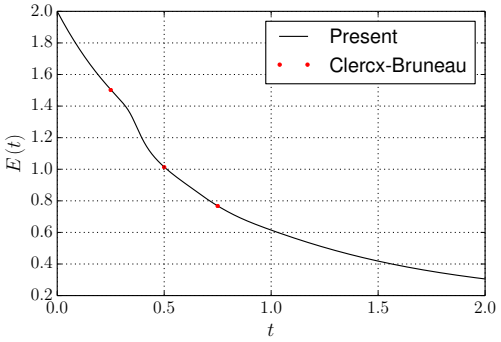
$$P(t) = \frac{1}{2} \int \int [\nabla \omega(\mathbf{x}, t)]^2 \ \mathrm{d}x\mathrm{d}y, \tag{4.45}$$

and gives an insight into the generation of vorticity at the no-slip boundary. Figure 4.14 compares the evolution of these time dependent parameters with the reference data provided by Clercx and Bruneau. Clercx and Bruneau [16] provide the values of the parameters at $t = 0.25$, $t = 0.5$ and $t = 0.75$.

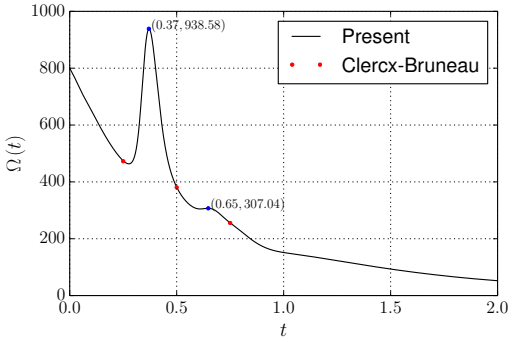
Figure 4.14a shows the evolution of the kinetic energy. The kinetic energy E reduced from $E(0) = 2$ to $E(2) \approx 0.3$. At $t = 0.4$, we have small kink representing the approach of the primary dipole at the wall. When plotting the reference data, we see that the variation in kinetic energy matches perfectly at $t = 0.25$, $t = 0.5$ and $t = 0.75$.

Figure 4.14b shows the evolution of enstrophy Ω in time. During the initial instants, enstrophy decreases linearly and at $t = 0.370$, there is sharp increase in the total enstrophy of the flow $\Omega(0.370) = 938.58$ (shown in blue). This indicates the initial impact of dipole with the no-slip wall. However, in literature [16], the initial peak occurs at $t = 0.371$ with a peak enstrophy of $\Omega(0.371) = 938.6$. Comparing these parameters we see that the enstrophy we calculate peaks earlier and with a larger values, meaning that our collision is slight early. However, the error in time of peaking is 0.3% with a 0.6% error in enstrophy, which is a significantly small error. We observe similar behaviour in the second collision, where the peak is our study is $\Omega(0.650) = 307.04$ (shown in blue), whereas in literature the peak is $\Omega(0.648) = 305.2$. Again the error is relatively small implying that our Eulerian method resolves the evolution of enstrophy very well.

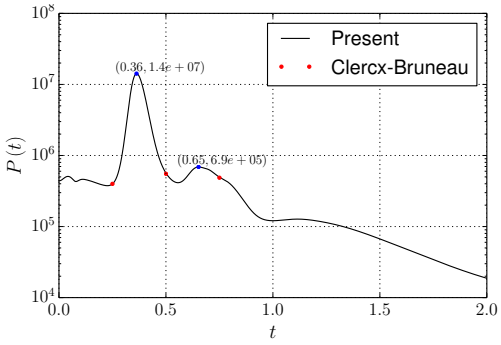
Figure 4.14c shows the evolution of palinstrophy P in time. When the dipole collides, vorticity is generated from the wall to ensure no-through boundary condition, which



(a) Kinetic Energy $E(t)$



(b) Enstrophy $\Omega(t)$



(c) Palinstrophy $P(t)$

Figure 4.14: Comparison of the fluid parameters from $t = 0$ to $t = 2$ with reference data obtained from Clercx and Bruneau [16] [•, **red** dot]. The figure shows the evolution of **(a)** the kinetic energy $E(t)$, **(b)** the enstrophy $\Omega(t)$, and **(c)** the palinstrophy $P(t)$.

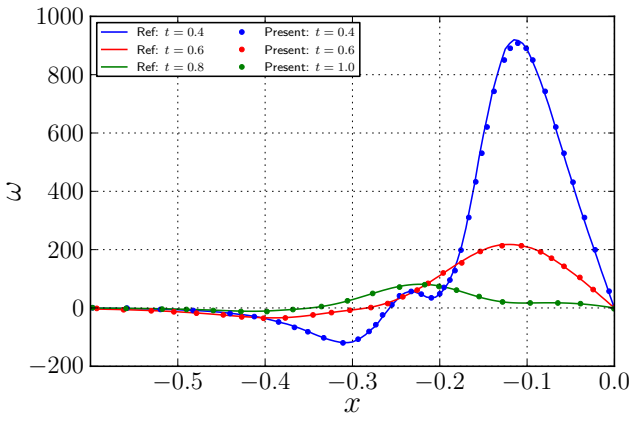


Figure 4.15: The vorticity generated at the bottom-left wall ($y = -1$, $-0.6 \leqslant x \leqslant 0$) at $t = 0.4$ [—, solid blue], $t = 0.6$ [—, solid red] and $t = 1$ [—, solid green]. The results are compared with Clercx and Bruneau [16] (dotted).

results in a sharp increase in the gradient of the vorticity. Similar to the evolution of enstrophy, we can observe to peaking at $t = 0.360$ and $t = 0.650$, with peak palinstrophy of $P(0.360) = 1.40\times10^7$ and $P(0.650) = 6.90\times10^5$, respectively (shown in blue). In literature however, the peak palinstrophy occurs at $P(0.361) = 1.39\times10^7$ and $P(0.652) = 6.78\times10^5$. This is an acceptable error and tells us the generation of vorticity in Eulerian method performs according to theory.

Figure 4.15 compares the vorticity along the boundary of the domain at $y = -1$ for $-0.6 \leqslant x \leqslant 0$. The solid lines represent the present data, and is compared with the dotted data obtained from Clercx and Bruneau [16]. The comparison is done for various time instances $t = 0.4$, $t = 0.6$ and $t = 1.0$ and we can finally validate that the Eulerian method accurately represents vorticity generation from the wall.

4.5.3 Impulsively started cylinder at $Re = 550$

Finally, we an impulsively started cylinder at $Re = 550$. This validation test ensured that we are able to determine correct forces acting on the body.

Problem Definition

The Impulsively Started Cylinder (ISC) test case simulates an impulsively started freestream flow around a cylinder. The test case focuses on the unsteady behavior of the separated flow past the cylinder. Various experimental and numerical investigations have been performed to study the flow characterstics. For this project we relied on the widely used and validated results of Koumoutsakos and Leonard [41]. They investigated the flow around the ISC using vortex method and provided extensive data on the vorticity profile behind the cylinder and the evolution of the lift and drag coefficients.

Figure 4.16 shows the domain of the ISC problem, where Figure 4.16a shows the definitions of the Eulerian domain Ω_E. The Eulerian domain Ω_E has the following boundary conditions: the no-slip wall boundary condition at Σ_w (solid blue) $\mathbf{u} = 0$, the freestream Dirichlet velocity boundary condition at Σ_d (solid red) $\mathbf{u}_\infty = [1, 0]$, and the pressure outlet Σ_p (solid green). Unlike the previous test cases we now require a pressure outlet boundary condition $\partial p/\partial \mathbf{n} = 0$, as the velocity field behind the cylinder perturbed and therefore free-stream boundary condition cannot be applied there. The boundaries of the Eulerian domain Ω_E are $\partial\Omega_E = \Sigma_w \cup \Sigma_d \cup \Sigma_p$.

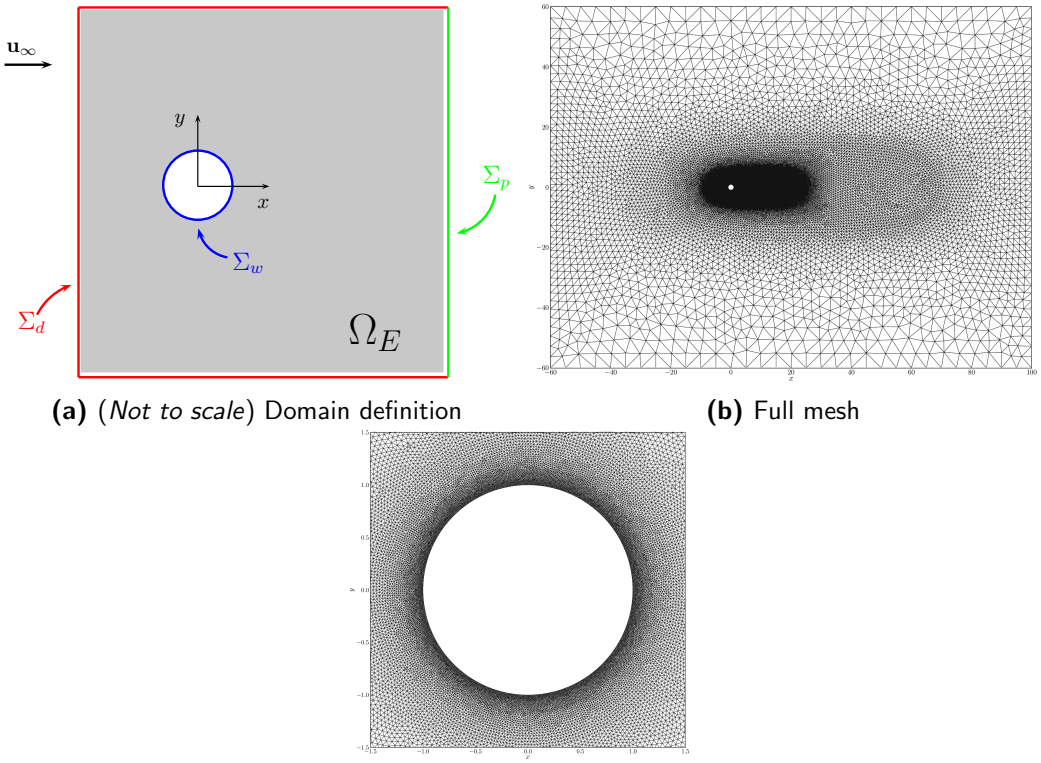
The Reynolds number Re of the flow dependent is defined as,

$$Re = \frac{UD}{\nu}, \tag{4.46}$$

and is a function of the freestream velocity U, the diameter of the cylinder D, and the kinematic viscosity ν. The normalized simulation time T is defined as,

$$T = \frac{U}{R}t, \tag{4.47}$$

where R is the radius of the cylinder.



(a) (*Not to scale*) Domain definition

(b) Full mesh

(c) Mesh near the surface

Figure 4.16: Domain of the ISC problem. The figure depicts **(a)** the definition, **(b)** the full domain mesh, and **(c)** the mesh near the surface.

Table 4.4: Summary of the parameters for the impulsively started cylinder test case for $Re = 550$.

Parameters	Value	Description
Ω	$[-60, 100] \times [-60, 60]$	Eulerian domain bounds
Re	550	Reynolds number
U	$1\hat{\mathbf{e}}_x$	Free-stream velocity
R	1	Radius of cylinder
D	2	Diameter of cylinder
ν	3.6×10^{-3}	Kinematic viscosity
T	0 to 40	Normalized simulation time
CFL	0.95	CFL number
$\|\mathbf{u}\|_{\max}$	2.5	Maximum fluid velocity
ΔT	1×10^{-3}	Time step size
N_{cells}	94352	Number of mesh cells
h_{grid}	9.7×10^{-3} to 7.4	FE mesh cell size
N_t	$40,000$	Number of time integration steps

The domain discretized with the highest mesh resolutions near the surface of the body, and in the near-wake region of the body, as shown in Figure 4.16b and Figure 4.16c. The discretization parameters are tabulated in table 4.4.

The simulation was started with an impulsively started free-stream boundary condition at the Dirichlet boundary Σ_d. The problem was evolved from $T = 0$ to $T = 40$ for $N_t = 40,000$ time steps. To validate the method against the reference data of Koumoutsakos and Leonard [41], we investigated the evolution of the vorticity field and the evolution of the forces acting on the body.

Results and Discussion

Figure 4.17 depicts the evolution of the vorticity at $T = [1, 3, 5, 7]$. The iso-vorticity contours of the present study are compared with the reference data obtained from Koumoutsakos and Leonard [41]. At $T = 1$, negative and positive vorticity is generated at the top and bottom of the cylinder, respectively. This results from satisfying the no-slip boundary condition. As time progresses, two primary vortices are formed behind the cylinder, increasing in shape as time advances. Comparing the vorticity contours, we can say that the contour lines match with the literature.

Using equations 4.36 to 4.38, we were able to calculate the lift and the drag force acting on the cylinder as time progresses, which we used to validate against the literature. Figure 4.18 shows the components of the drag force (friction drag $C_{d_{\text{fric}}}$, pressure drag $C_{d_{\text{pres}}}$) acting on the surface of the body. At $T = 0$, we have a singularity in the total drag C_d acting on the body due to the impulsive start of the flow. It then plunges to $C_d = 0.75$ at $T = 0.8$ and peaks again near $T = 3$ with $C_d = 1.3$. The dotted line is the data obtained from literature [41] and we see that the results of the simulation match well with the literature.

A final comparison was done for the evolution of the lift and the drag coefficients for a larger period ($T = 0$ to $T = 40$), which was used to determine the oscillatory behavior of the forces. For lower Reynolds number, the vorticity field is symmetric across the x-axis for a long time. This meant that the oscillatory behavior of the forces starts at a much later time. Therefore, we prescribed an artificial perturbation to the problem to create an asymmetry in the vorticity field which trips the wake. The perturbation was performed according to Leocointe and Piquet [44],

$$u_{\text{wall}} = \begin{cases} 0.15 & 3 \leqslant T \leqslant 3.5, \\ -0.25 & 3.5 \leqslant T \leqslant 5. \end{cases} \tag{4.48}$$

With this, we could ensure that we have a controlled behavior for the lift and drag, which we used to determine the amplitude and the frequency of the oscillation. Figure 4.19 compares the evolution of the lift and drag for $T = 0$ to $T = 40$. We see that our numerical scheme performs very similar to the literature [54]. However, there is a slight difference, which is due to the under-resolution of the Eulerian domain downstream of the cylinder, making the wake our Eulerian method more dissipative than the Lagrangian reference method used by Koumoutsakos and Leonard [41].

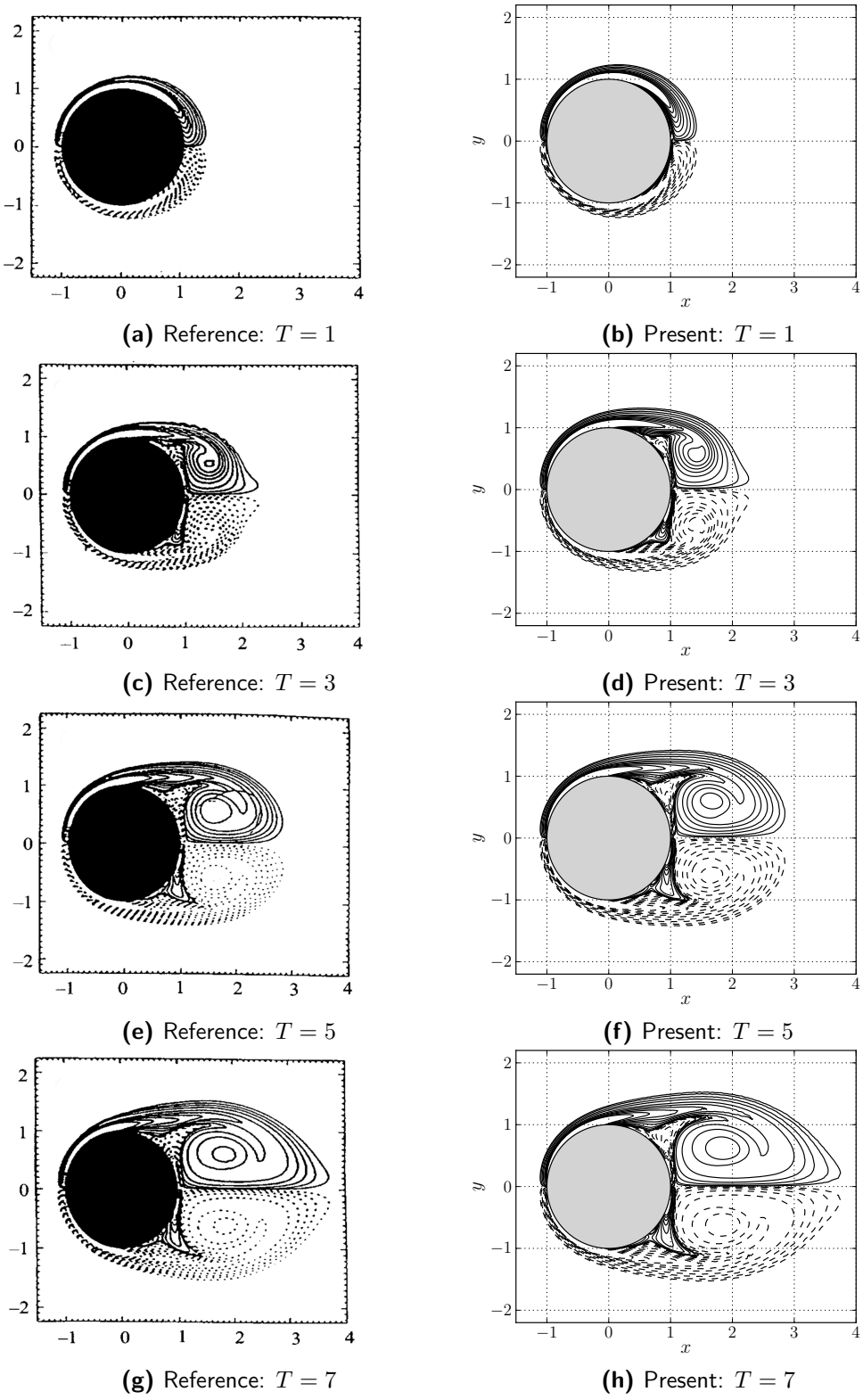


Figure 4.17: Comparison of the vorticity contours for $T = [1, 3, 5, 7]$ with contour levels $[-7, ..., -2, -1, -0.5, -0.2, -0.1, 0.1, 0.2, 0.5, 1, 2, ..., 7]$, showing negative vorticity (solid) and positive vorticity (dotted). The present study (right) is compared with plots obtained from Koumoutsakos and Leonard [41] (left).

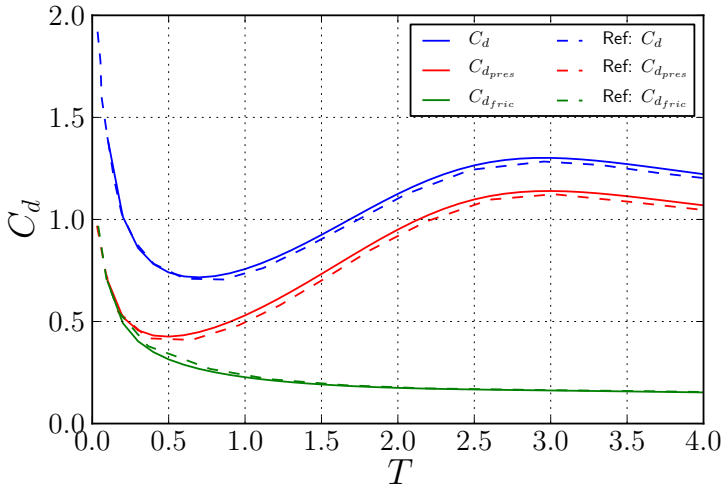


Figure 4.18: Evolution of drag force. The figure depicts the total drag coefficient C_d [—, solid blue], the pressure drag coefficient $C_{d_{\text{pres}}}$ [—, solid red] and the friction drag coefficient $C_{d_{\text{fric}}}$ [—, solid green]. The dotted lines indicate the data obtained from literature, Koumoutsakos and Leonard [41].

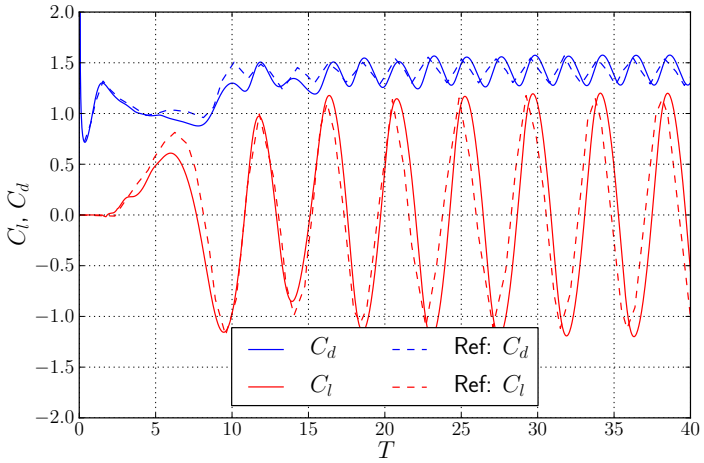


Figure 4.19: Evolution of the lift coefficient C_l and the drag coefficient C_d from $T = 0$ to $T = 40$ with artificial perturbation [44]. The dotted lines represent the data obtained from literature, Rosenfeld et al. [54].

4.6 Summary

In summary, we have investigated the Eulerian domain of our hybrid method in this chapter. The Eulerian method was used to resolve the near-body region where generation of vorticity from the *no-slip* boundary was of concern. The advantage of an Eulerian formulation is that it is much more efficient in resolving the boundary layer than the VPM described in chapter 3. We decided to use a Finite Element Method (FEM) to solve the incompressible laminar Navier-Stokes problem of the near-wall region.

In section 4.1, an introduced to the FEM was given. We investigated the theory of finite element discretization, and introduced the concept of functions and function spaces of a finite elements. Finally, we introduced the variational formulation for solving the PDE.

In section 4.2, we introduced FEniCS project, a collaborative work of various universities that developed tools to perform automated finite element algorithms. The Dolfin library of FEniCS project provided the finite element library for set up and solve the finite element problem. We used the Dolfin library wrapped in Python . It uses automated code generation thus maintaining high-level mathematic expressions but still provided an efficient, a multi-threaded performance. Gmsh mesh generation tool was used to generate the unstructured mesh of the fluid domain.

In section 4.3, we investigated the Incremental Pressure Correction Scheme (IPCS) for solving the incompressible Navier-Stokes equations. The scheme allows us to decouple the velocity $\mathbf{u}$ and pressure p from the momentum equation. Furthermore, we determined an approach for solving the vorticity field, which was computationally efficient.

In section 4.5, we verified and validated our implementation of the Eulerian method. A Lamb-Oseen Vortex test case was used to verify the implementation of the Eulerian method, and concluded that the method had a 1^{st}-order convergence in time and 2^{nd}-order convergence in space. To validate the calculation of vorticity and the vorticity production of the no-slip boundary, we used the high-fidelity numerical test case of Clercx and Bruneau [16]. The literature studied the collision of dipole with the wall, investigating the change in kinetic energy E, enstrophy Ω, and Palinstrophy P over time. Furthermore, we compared the generation of vorticity at the boundary, validating a consistent result.

The final test case involved simulating an impulsively started cylinder at $Re = 550$. We investigated the shedding of vorticity in time progress and observed that it matched the reference data provided by Koumoutsakos and Leonard [41]. We also investigated the long term evolution of the lift and drag of the cylinder, and observed that the frequency and the amplitude of the oscillation was similar to literature study of Rosenfeld et al. [54].

4.7 Chapter Nomenclature

Latin Symbols

$\mathbf{A}$	Coefficient matrix	-
b	Right-Hand-Side	-

c	Reference length (chord)	m
C_d	Drag coefficient	-
C_l	Lift coefficient	-
CFL	CFL number	-
$\hat{\mathbf{e}}$	Cartesian unit vector	-
E	Kinetic Energy	J
f	Source terms	-
$\mathbf{I}$	Identity matrix	-
$\hat{\mathbf{n}}$	Unit normal vector	-
N_{cells}	Number of FE mesh cells	-
N_t	Number of time integration steps	-
p	Pressure	Pa
	Trial function for pressure	-
$\mathcal{P}$	Lagrange polynomial	-
q	Degree of Lagrange polynomial $\mathcal{P}_q$	-
	Test function for pressure	-
Q	Scalar-valued function space for pressure p	-
Re	Reynolds number	-
t_n	Simulation time at n^{th} step	s
$\mathcal{T}_h$	Finite Element mesh	-
T	Cell of Finite Element mesh	-
$\mathbf{u}$	Velocity	$\mathrm{m\,s^{-1}}$
	Trial function for velocity	-
$\mathbf{u}^\star$	Tentative velocity	$\mathrm{m\,s^{-1}}$
v	Test function for velocity	-
V	Trial vector function space for velocity	-
$\hat{V}$	Test vector function space for velocity	-
w	Trial function for vorticity	-
x	Test function for vorticity	-
X	Scalar-valued function space for vorticity ω	-

Greek Symbols

Γ	Circulation	$\mathrm{m^2\,s^{-1}}$
Δh	Mesh cell size	m
Δt_E	Eulerian time step size	s
ϵ	Symmetric gradient	-
ν	Kinematic viscosity	$\mathrm{kg\,s^{-1}\,m^{-1}}$
ρ	Fluid density	$\mathrm{kg\,m^{-3}}$
σ	Cauchy stress tensor	Pa
Σ_d	Dirichlet velocity boundary	-

Σ_p	Pressure outlet boundary	-
Σ_w	No-slip wall boundary	-
ψ	Basis function	-
ψ	Basis function	-
ω	Vorticity	s^{-1}
Ω	Fluid domain	-
Ω_E	Eulerian fluid domain	-
$\partial\Omega$	Boundary of the domain Ω	-

Chapter 5

Coupling Eulerian and Lagrangian Method

Chapter 2 provided a detailed summary on Hybrid Eulerian-Lagrangian Vortex Particle Method, where we introduced the concept of coupling the Lagrangian and the Eulerian method. The chapter investigated the coupling algorithm without Schwartz iterative method used by Daeninck [24], and the Lagrangian correction algorithm used by Stock [61]. However, during the investigation and implementation of the algorithms, we observed some issues that had to be dealt with.

5.1 Modifications to the Lagrangian Correction Strategy

The Lagrangian correction step by Stock [61], based on the coupling strategy of Daeninck [24] had to be modified in the present work. During the investigation of the algorithm, it became apparent that without additional steps, the total circulation in the Lagrangian method will be not be conserved.

The two issues that causes an error in the total circulation (and subsequently introduces additional error in the coupling) are the following:

- **Vortex particle re-initialization**: In section 3.2.4, we observed that initializing the particles using the local particle volume and local vorticity causes a diffusive effect on the vorticity distribution due to the *smoothing error* of Gaussian kernels. Section 5.1.1 elaborates the cause and correction required for this problem.
- **Circulation of Vortex sheet**: In section 3.5, we saw that as the vortex panel problem is singular, we require an additional constraint on total circulation of the vortex sheet. It is an unknown and has to be determined from the solution of the Eulerian method. Section 5.1.2 elaborates the methodology for determining the strength.

- **Conservation of total circulation**: The Lagrangian correction step, consisting of re-initializing the vortex blobs inside the interpolation domain Ω_I (Figure 2.6), and the transfer of circulation to the vortex panels must ensure that the total circulation in the Lagrangian method is conserved. Section 5.1.3 elaborates the methodology used to explicitly ensure the conservation of circulation in the Lagrangian domain during the correction step.

5.1.1 Vortex Particle Re-initialization

The Lagrangian correction step requires the correction (or adjustment/modification) of the strength of the vortex particles inside the interpolation domain Ω_I, shown in Figure 2.6. However, when investigating this approach, it becomes apparent that the standard initializing approach of local particle volume and local vorticity is erroneous.

Concern with re-initialization method

The Lagrangian method discretizes the exact vorticity field ω by linear combination of N Gaussian basis functions, i.e vortex blobs. Let $\mathcal{P} = \{1, ..., N\} \subset \mathbb{N}$ be the set of particle indices of N particles p with position $\mathbf{x}_p$. The discrete continuous vorticity field $\hat{\omega}$ is given as,

$$\omega \approx \hat{\omega}(\mathbf{x}) = \sum_{p \in \mathcal{P}} \alpha_p \zeta_\sigma(\mathbf{x} - \mathbf{x}_p). \tag{5.1}$$

where α_p is the strength of particle $p \in \mathcal{P}$. In vortex method, it is typically a standard approach to initialize the particle strengths α_p using the local particle area h^2 and the local vorticity ω,

$$\alpha_p = \omega_p \cdot h^2. \tag{5.2}$$

where ω_p is the vorticity at particle position $\mathbf{x}_p$. This approach was also employed by Stock [61], to modify the strengths of the vortex blobs inside the correction region Ω_I. However, in section 3.2.4, we observed that this type of initialization introduces a *smoothing error*. Barba and Rossi [1] noticed that the standard initialization corresponds to a Gaussian blurring of the original vorticity field and is equivalent to the "diffusion" of the vorticity.

The accurate re-initialization of the Lagrangian vorticity field $\hat{\omega}^L$ in Ω_I using the Eulerian vorticity field ω^E must satisfy the following equality:

$$\omega^E = \hat{\omega}^L \qquad \text{in } \Omega_I. \tag{5.3}$$

Satisfying equation 5.3 ensures that the vorticity solution of the Eulerian method ω^E matches the blurred (mollified) Lagrangian vorticity field $\hat{\omega}^L$. Substituting equation 5.1 into Equation 5.3 gives:

$$\omega^E(\mathbf{x}_j) = \sum_{i=p \in \mathcal{P}} \alpha_i \zeta_\sigma(\mathbf{x}_j - \mathbf{x}_i) \qquad \text{in } \Omega_I, \tag{5.4}$$

simplifying to a system of linear equations,

$$\omega^E = \mathbf{A}_{ij}\alpha_i \qquad \text{in } \Omega_I \tag{5.5}$$

where $\mathbf{A}_{ij} = \zeta_\sigma(\mathbf{x}_j - \mathbf{x}_i)$ is a coefficient matrix. Therefore the strengths of the particles α_p must be obtained from equation 5.5 and the initialization of the particles using equation 5.2 is mathematically incorrect.

Equation 5.5 is a linear system of equations equating the strengths of each particle to the desired Eulerian vorticity distribution. However, inverting the matrix $\mathbf{A}$ is still an open equation in vortex method, as stated by Koumoutsakos and Cottet [22], and was investigated by Barba and Rossi [1]. The problem is that the matrix $\mathbf{A}$ is full and badly condition for direct inversion. For a global field interpolation (i.e for unbounded domain), one could use Beale's iterative method which uses a successive over-relaxation (SOR) for solving the equation 5.5. This method relies on iterative correction of all the particles in the full Lagrangian domain. However, in our case for initializing the strengths of the particles only in the sub-domain Ω_I of the Lagrangian domain Ω_L, it would require us to modify the strength of only those particles. In such case, Beale's iterative method is not valid and cannot be used. Therefore, Beale's method cannot be used to solve the problem of smoothing error.

In future, the key to solving this smoothing error might be in the research works of Barba and Rossi [1], where they try to reverse the blurring of the vorticity field by reversing the "diffusion" caused by the smoothing kernel. However, currently for our investigation the best solution is to minimize the error in equation 5.2.

Modification

The error ϵ due to the mismatch in the Eulerian vorticity field ω^E and the Lagrangian vorticity $\hat{\omega}^L$ in the interpolation domain Ω_I is defined as,

$$\epsilon = |\omega^E - \hat{\omega}^L| \qquad \text{in } \Omega_I \tag{5.6}$$

where the error is divided into,

$$\epsilon = \epsilon_\sigma + \epsilon_h, \tag{5.7}$$

the sum of the smoothing error ϵ_σ, and the discretization error ϵ_h. In section 3.2.5, we investigated the minimization of this error and observed that the error ϵ scales with particle resolution. With an overlap ratio of $\lambda = 1$ and a small particle core spreading σ, we can ensure that the initialization error inside the interpolation domain Ω_I is minimal. To determined an appropriate core spreading σ, we should target an initialization error of $\epsilon \leqslant 5\%$ ensuring a small error in coupling.

5.1.2 Circulation of Vortex sheet

The second concern with the implementation of Daeninck's coupling strategy and Stock's Lagrangian correction step is the uncertainty of the vortex sheet strengths. In section 2.3.1, we described that to time march the Lagrangian solution from t_n to $t_n + \Delta t_L$, we have to enforce a *no-slip* boundary condition at the wall by computing the satisfactory vortex sheet distribution γ.

To solve for the vortex sheet distribution γ that satisfy the no-slip boundary conditions, we discretized the boundary integral equation using vortex panels, as described in section 3.5.

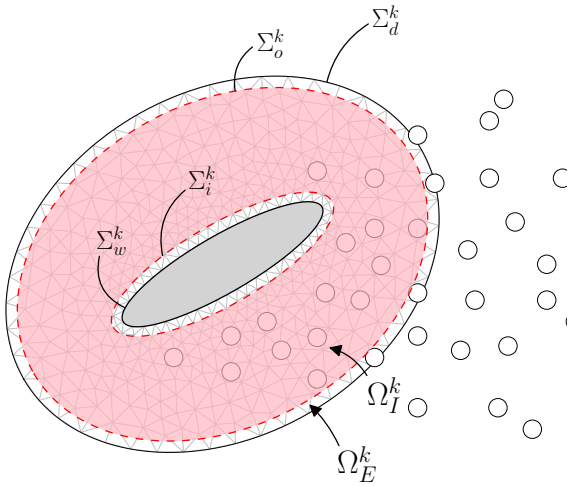


Figure 5.1: A domain decomposition with $k = 1, ..., N_E$ Eulerian subdomains. The figure shows the definition of the k^{th} interpolation domain Ω_I^k belonging to the k^{th} Eulerian domain Ω_E^k.

Koumoutsakos [39], related the vortex sheet strengths to the no-slip boundary conditions with Fredholm integral equation of the second kind, equation 3.49. However, as this equation is singular, we require an additional constraint to obtain a unique solution. Kelvin's circulation theorem imposes a direct constraint on the integral strengths of the vortex sheet, shown in equation 3.50, and can be used to find the unique solution.

Let us investigate a hybrid problem with N_E number of Eulerian subdomain and the k^{th} Eulerian domain $\Omega_E^k \subset \Omega_L$ is defined as shown in Figure 5.1. Stock [61] said that the Eulerian solution is assumed to be correct from the body surface Σ_w^k to only somewhat inside of the outer Eulerian domain Σ_d^k. More precisely, all the Eulerian solution within the domain Ω_{in}^k with $\partial\Omega_{in}^k = \Sigma_o$ as shown in Figure 5.2. During the Lagrangian correction step, Stock corrected the particles p with $\mathbf{x}_p \in \Omega_I^k$ using the more accurate Eulerian solution. So, any remaining Eulerian solution within Σ_o^k that was not transfered during the Lagrangian correction step should be resolved by the vortex sheet to ensure conservation of circulation.

In terms of circulation, we require that the Eulerian circulation $\Gamma_{in}^{E,k}$ and the corrected Lagrangian circulation $\hat{\Gamma}_{in}^{L,k}$ of the k^{th} correction region Ω_{in}^k must match to ensure conservation of circulation during the correction step, given by,

$$\Gamma_{in}^{E,k} = \hat{\Gamma}_{in}^{L,k} \qquad \text{in } \Omega_{in}^k, \tag{5.8}$$

The Lagrangian circulation $\hat{\Gamma}_{in}^{L,k}$ is decomposed as follows,

$$\hat{\Gamma}_{in}^{L,k} = \Gamma_{\gamma}^{L,k} + \hat{\Gamma}_{I}^{L,k}, \tag{5.9}$$

where $\Gamma_{\gamma}^{L,k}$ is the circulation of the vortex sheet and $\hat{\Gamma}_{I}^{L,k}$ is the circulation of the corrected vortex blobs in the interpolation domain Ω_I^k. The circulation of the particles $\hat{\Gamma}_{I}^{L,k}$ is determined directly from the sum of particles particle strengths $\hat{\alpha}_p^k$,

$$\hat{\Gamma}_{I}^{L,k} = \sum_p \hat{\alpha}_p^k \qquad \text{for } \mathbf{x}_p \in \Omega_I^k, \tag{5.10}$$

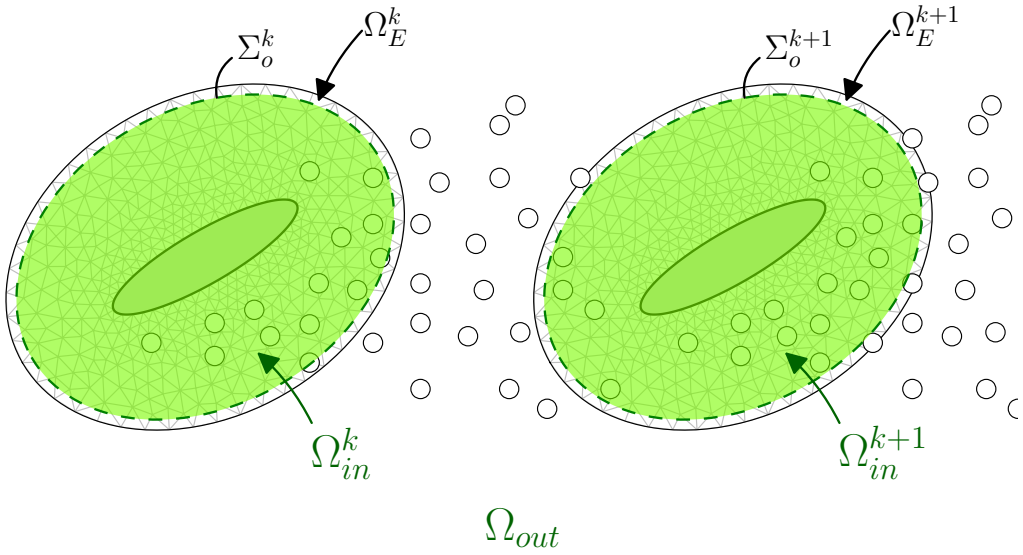


Figure 5.2: The segregation of the Lagrangian domain $\Omega_L = \Omega_{in} \cup \Omega_{out}$ into the region which is uncorrected Ω_{out} and the region which is corrected $\Omega_{in} = \bigcup_{k=1}^{N_E} \Omega_{in}^k$.

where $\hat{\alpha}_p$ is the strength of the corrected particles determined using equation 5.2. Substituting equation 5.9 into equation 5.8 helps us determine the circulation of the vortex sheet,

$$\Gamma_\gamma^{L,k} = \Gamma_{in}^{E,k} - \hat{\Gamma}_I^{L,k}. \tag{5.11}$$

Using this additional constraint on the circulation of the vortex sheet, we can solve the distribution of the vortex sheet that satisfies the no-slip boundary condition.

5.1.3 Conservation of Total Circulation

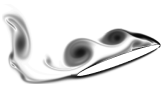
In section 5.1.1 we observed that the method used to determine the strength of particles inside the interpolation domain Ω_I (Figure 5.2) is not accurate for transferring the Eulerian solution to the Lagrangian domain. This approach is erroneous and has a detrimental effect on the total circulation of the Lagrangian domain Ω_L. To ensure that circulation is conserved during the correction step, we to perform an additional correction to the transfer.

The Lagrangian domain Ω_L can be divided into two section (as shown in Figure 5.2):

- Correction region Ω_{in}: The Lagrangian region where the correction takes place: $\Omega_{in} = \bigcup_{k=1}^{N_E} \Omega_{in}^k$, where the k^{th} correction region is defined by $\partial\Omega_{in}^k = \Sigma_o^k$, the outer boundary of the interpolation region Ω_I^k.
- Unmodified region Ω_{out}: The Lagrangian region that is outside the correction region and is therefore unmodified during the correction step: $\Omega_{out} = \Omega_L \backslash \Omega_{in}$.

The total circulation in the Lagrangian domain before the correction is given as,

$$\Gamma^L = \Gamma_{in}^L + \Gamma_{out}^L, \tag{5.12}$$



where $\Gamma^L_{in} = \sum_k \Gamma^{L,k}_{in}$ is circulation inside Ω_{in} before they are corrected, and Γ^L_{out} is the circulation in Ω_{out}. During the Lagrangian correction step, the circulation Γ^L_{in} is corrected and replaced with the corrected circulation $\tilde{\Gamma}^L_{in}$ determine from the Eulerian method, equation 5.8, resulting in a new total Lagrangian circulation $\tilde{\Gamma}^L$ given by,

$$\tilde{\Gamma}^L = \tilde{\Gamma}^L_{in} + \Gamma^L_{out}, \tag{5.13}$$

To ensure that the circulation is conserved during the correction step, we required $\Delta\Gamma = 0$. This gives us the requirement that,

$$\Gamma^L = \tilde{\Gamma}^L. \tag{5.14}$$

or, substituting equation 5.12 and equation 5.13 into equation 5.14 gives us the requirement that,

$$\Gamma^L_{in} = \tilde{\Gamma}^L_{in}, \tag{5.15}$$

and equivalently,

$$\Gamma^{L,k}_{in} = \tilde{\Gamma}^{L,k}_{in}. \tag{5.16}$$

This means that the circulation inside the correction region Ω^k_{in} before the correction should match the circulation after the correction. However, as there exists a slight error in the particle initialization, we introduce a small error in circulation ϵ^k during the transfer, resulting in,

$$\hat{\Gamma}^{L,k}_{in} = \tilde{\Gamma}^{L,k}_{in} + \epsilon^{L,k}_{in}, \tag{5.17}$$

where $\hat{\Gamma}^{L,k}_{in}$ is the erroneous circulation that was actually transfered and $\tilde{\Gamma}^{L,k}_{in}$ is the correct circulation that was supposed to be transferred. Substituting equation 5.17 into equation 5.15, we see that the error $\epsilon^{L,k}_{in}$ is written as,

$$\epsilon^{L,k}_{in} = \hat{\Gamma}^{L,k}_{in} - \Gamma^{L,k}_{in}. \tag{5.18}$$

and is simply the mismatch in the circulation during the correction step. This error can by modifying the new set of initialized particles in the interpolation domain Ω_I. Using equation 5.10, the final strengths of the particles $\tilde{\alpha}^k_p$ for particles $\mathbf{x}^k_p \in \Omega^k_{in}$ is given as,

$$\tilde{\alpha}^k_p = \alpha^k_p - \frac{\epsilon^{L,k}_{in}}{N}, \tag{5.19}$$

where α^k_p is the uncorrected strengths determined using the local particle area and local vorticity, equation 5.2, and N is number of particles inside the correction region Ω^k_{in}. Following this procedure in addition to Stocks Lagrangian correction strategy described in section 2.3.2, we will ensure that our hybrid scheme conserves circulation.

5.2 Modified Lagrangian Correction Algorithm

In this section we will investigate the algorithm used for the *Correct Lagrangian* step of our hybrid evolution, Figure 5.3. We first summarized the algorithm used by Stock [61] in section 2.3.2. However in section 5.1, we determine that the algorithm required some

modification to ensure minimum interpolation error and conservation of total circulation. Therefore, we will also describe the algorithm required to implement these modifications.

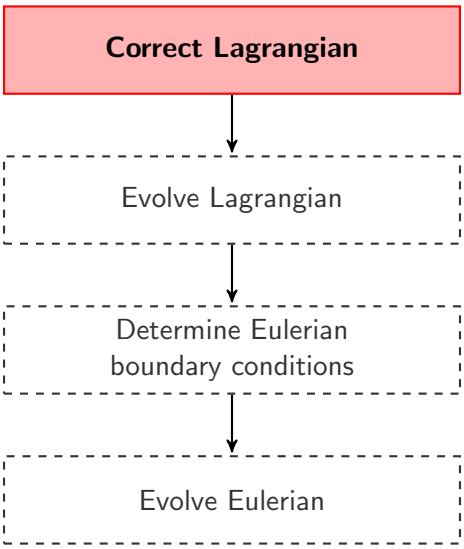


Figure 5.3: Flowchart of the hybrid evolution, focusing on the 1st step: Correct the Lagrangian solution.

The general approach we used to perform transfer of the solution from the Eulerian to Lagrangian domain is by substituting particles inside the zone of correction (i.e the correction region Ω_{in} as shown in Figure 5.1) with a correct set of particles that represent the more accurate Eulerian solution. We need to determine the vorticity at their positions so that we can use equation 5.2 to initialize the strengths of these new particles. The vorticity at these positions can be determined by interpolating the Eulerian solution onto their positions.

However, determining the location of the particle w.r.t the unstructured grid of the Eulerian method is computationally expensive. Therefore, we will first interpolate the Eulerian solution onto a regular grid. Performing a search algorithm on such grid is computationally less expensive. Furthermore, to ensure that the interpolation of vorticity from unstructured grid onto a regular structured grid is also efficient, the regular structured grid will be bounded to the Eulerian domain. In such case, the interpolation problem only needs to be setup once as the transfer weights remains unchanged throughout the simulation.

The final stages of the Lagrangian correction step is to ensure that the total circulation is conserved. We will therefore determine the strength of the vortex sheet and the total strength of the newly generated vortex blobs such that circulation is conserved.

The algorithm describing these procedure is organized and summarized as follows:

1. **Interpolate vorticity**: Interpolate the vorticity from the unstructured Eulerian mesh onto a structured grid that is bounded to the Eulerian domain Ω_E.

2. **Remove particles**: Remove particles that are inside the interpolation domain Ω_I.

3. **Generate particles**: Generate zero-strength particle inside the interpolation domain Ω_I.

4. **Assign strengths to particles**: Interpolate the vorticity from the structured grid onto the position of the newly generated particles. Using the standard particle initialization approach described in section 5.1.1, assign the strengths to the newly generated particles.

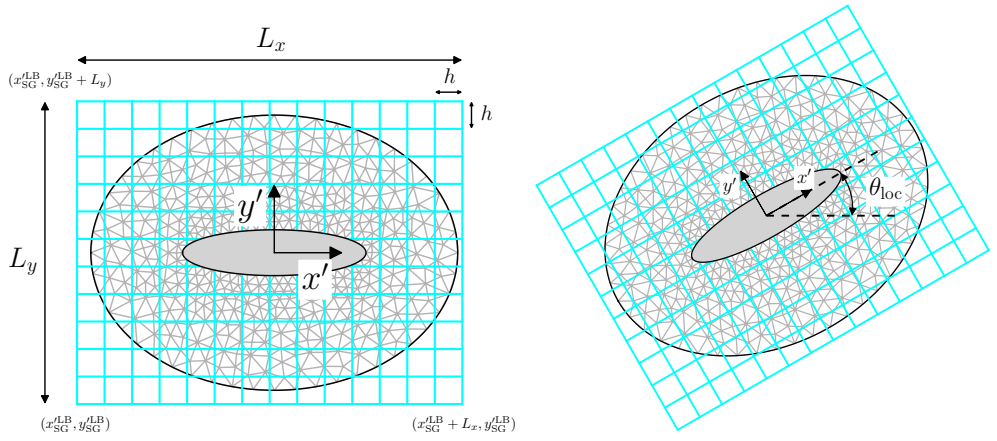
5. **Correct total circulation**: Using the approach described in section 5.1.2 and section 5.1.3, determine the total strength of the vortex sheet and total strength of the newly generated vortex blobs such that the total circulation is conserved.

The steps 1 to 4 will be described for the k^{th} Eulerian domain Ω_E^k. These steps simply needed to be iterated for all N_E number of Eulerian domains (i.e bodies) in the case of a multi-body problem. The last step of correcting the total circulation is performed for all the Eulerian domains simultaneously as we require the knowledge of total circulation. A detailed description of the each steps of the Lagrangian correction algorithm is given below.

5.2.1 Interpolate Vorticity

In this step, we will interpolate the vorticity from the unstructured grid of the Finite Element method onto a regular structured grid, that is bounded to the Eulerian domain Ω_E^k. The algorithm for interpolating the vorticity from the unstructured grid onto a structured grid is as follows:

1.a) **Setup a structured grid** (*only once*): A structured grid (SG) is created in the local coordinate system of the k^{th} Eulerian domain Ω_E^k. The grid is constructed such that it covers the Eulerian grid (SG) of the Eulerian method, as shown in Figure 5.4a.



(a) Local coordinate system **(b)** Global coordinate system

Figure 5.4: Structured grid (cyan) covering the entire Eulerian grid (**black**).

The parameters that define the structured grid in the local coordinate system are the starting point of the grid (the left-bottom corner) $(x'^{\mathrm{LB}}_{\mathrm{SG}}, y'^{\mathrm{LB}}_{\mathrm{SG}})$, the horizontal length of the grid L_x, the vertical length of the grid L_y, and the grid cell spacing h (which is equal to the nominal blob spacing). The lengths L_x and L_y are defined such that it is a multiple of the grid spacing h,

$$L_x = n_x \cdot h \tag{5.20a}$$

$$L_y = n_y \cdot h, \tag{5.20b}$$

where n_x and n_y are the number of cells in the horizontal and the vertical direction, respectively. The transformation of the structured grid follows that of the body and therefore the structure grid does not move w.r.t to the Eulerian grid. For more details on the transformation from the local coordinate system to the global, see appendix A. Figure 5.4b shows the structured grid bounded to the body in the global coordinate system.

1.b) **Determine the weights** (*only once*): The interpolation of the Eulerian vorticity ω from the Eulerian grid onto the structured grid is represented as,

$$\hat{\omega}_j = \sum_i \omega_i W_{ij}, \tag{5.21}$$

where $\hat{\omega}_j$ is the interpolated vorticity on the structured grid and W is the interpolation weights. As the structured grid is bounded to the Eulerian grid, the weights of the interpolation does not change throughout the simulation and only needs to be calculated once. This makes the interpolation problem computationally very efficient.

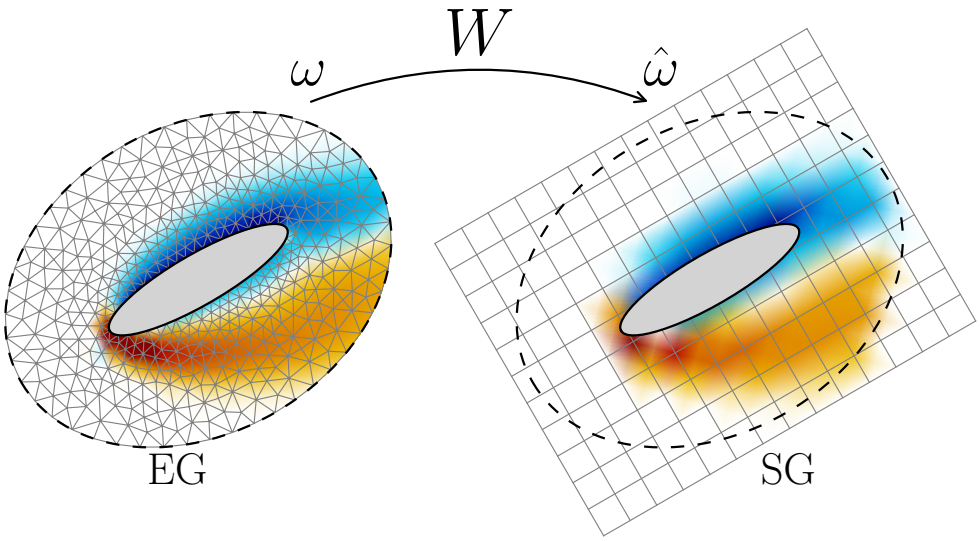


Figure 5.5: Interpolation of the vorticity ω from the Eulerian grid onto the structured grid using equation 5.21.

1.c) **Interpolate the vorticity**: This step needs to be perform at every Lagrangian correction step. The vorticity $\hat{\omega}$ on the structured grid is calculated by simply solving the pre-assembled problem, defined by equation 5.21.

To construct the interpolation problem, we used a tree search algorithm from the CGAL library [10], included in FEniCS and adapted for fast repeated evaluation by Mortenson (Fenicstools [49]). The algorithm probes the vorticity function ω of the unstructured Eulerian grid at the nodes of the structured grid $(x_{\text{SG}}, y_{\text{SG}})$. Figure 5.5 shows a depicts the result of the transferring the vorticity from the unstructured grid to the structured grid. The vorticity on the structured grid can then be efficiently interpolated onto the positions of the particles.

5.2.2 Remove Particles

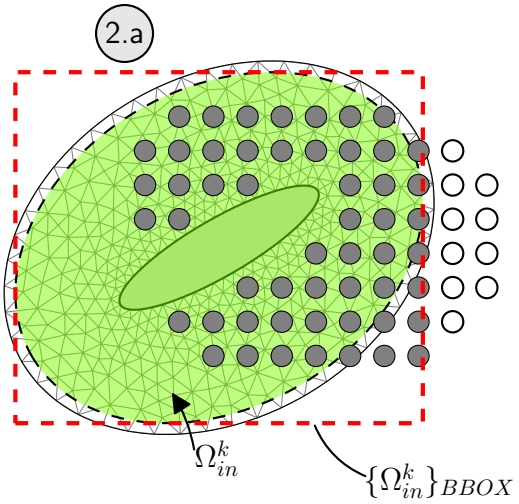
In this step, the uncorrected vortex particles inside the correction region Ω_{in} (as shown in Figure 5.2) will be removed. Let $\mathcal{P}$ be the set of particle indices with position $\mathbf{x}_p \in \Omega_L$. The algorithm for removing the particles p is as follows:

2.a) **Find the particles inside the BBOX of Ω^k_{in}**: Find the particles p that lie inside the minimum bounding box of the correction domain Ω^k_{in}. Let $\mathcal{P}^k_{\text{BBOX}}$ be the set of particles indices that are within the bounding box of the correction domain Ω^k_{in}, the gray particles shown in Figure 5.6a. The minimum bounding box (shown in red) is constructed using the outer boundary polygon of the correction region Σ_o. Performing this search algorithm is computationally efficient and reduces the number of particles of interest for the more expensive *point-in-polygon* search algorithm of the next step.

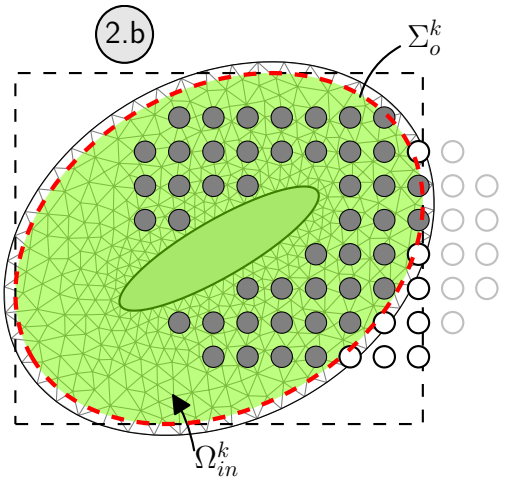
2.b) **Find the particles inside the correction region Ω^k_{in}**: Determine which of the particles in the set $\mathcal{P}^k_{\text{BBOX}}$ also lie within the correction domain Ω^k_{in}. Let $\mathcal{P}^k_{in} \subset \mathcal{P}^k_{\text{BBOX}}$ be the set of particles that also lie inside the correct region Ω^k_{in}, the gray particles in Figure 5.6b. The search algorithm is performed using a *point-in-polygon* test with the outer boundary Σ^k_o (shown in red).

2.c) **Remove particles inside correction region**: Remove the set of particles inside the correction region $\mathcal{P}^k_{in}$. These particles have a total circulation of $\Gamma^{L,k}_{in}$ (as used in Equation 5.12) which needs to be compensated later to ensure conservation of total circulation. Figure 5.6c shows the final set of particles after *remove particle* step.

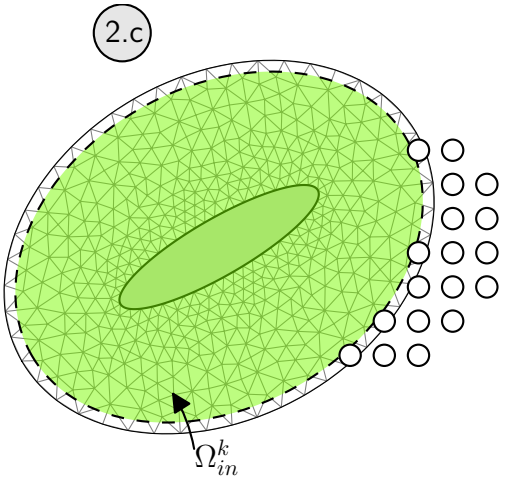
To perform the point-in-polygon test, we used the `pnpoly` function of `matplotlib`, the python 2D plotting library created by Hunter [34]. The function implemented the *point inclusion in polygon* test algorithm developed by Franklin [28]. The algorithm is based on the crossings test, which determines whether the point is inside the polygon by determining the number of the times a semi-infinite ray originating from the point intersects with the polygon.



(a) Step 2.a: Find the particles inside the BBOX of Ω^k_{in}

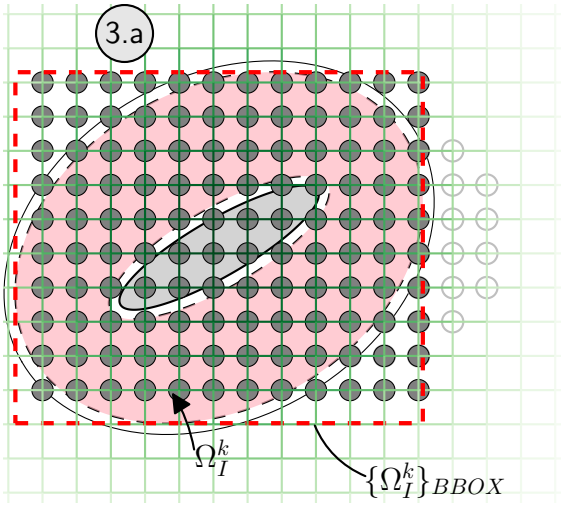


(b) Step 2.b: Among them, find the particles that are also inside the correction region Ω^k_{in}

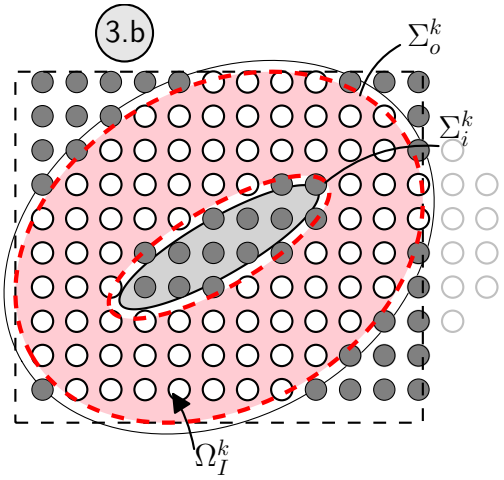


(c) Step 2.c: Remove the particles that are inside the correction region Ω^k_{in}

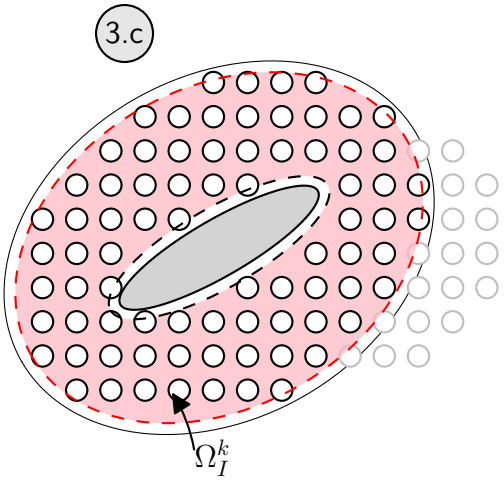
Figure 5.6: Figures depicts the computationally optimized algorithm for finding and removing the vortex particles inside the correction region Ω^k_{in}.



(a) Step 3.a: Generate particles inside the BBOX of the interpolation domain Ω_I^k matching the Lagrangian mesh



(b) Step 3.b: Among them, find the particles that are outside the interpolation domain Ω_I^k



(c) Step 3.c: Remove the newly generated particles that are outside interpolation domain Ω_I^k

Figure 5.7: Figures depicts the algorithm for generating vortex particles inside the interpolation domain Ω_I^k matching the Lagrangian mesh.

5.2.3 Generate Particles

In order to interpolate the solution from the Eulerian domain onto Lagrangian domain, we require the knowledge of the particle positions. In this step, we will generate zero-strengths particles inside the interpolation region Ω_I^k (Figure 5.1).

The algorithm for generating zero-strength particles inside the k^{th} interpolation domain Ω_I^k is as follows:

3.a) **Generate particles inside the BBOX of interpolation domain** Ω_I^k: Generate zero-strength particles inside the bounding box of the interpolation region Ω_I^k, the gray particles shown in Figure 5.7a. The particles are positioned at the nodes of the global Lagrangian remeshing grid (introduced in section 3.3.1) such that particles are equally distributed as satisfies the overlap ratio criteria. Let $\hat{\mathcal{P}}^k_{\text{BBOX}}$ be the set all newly generated particles (gray) inside the bounding box at the nodes of the Lagrangian method (green).

3.b) **Find newly generated particles inside interpolation region** Ω_I^k: We perform a point-in-polygon test for the newly generated particles $\hat{\mathcal{P}}^k_{\text{BBOX}}$, so that we can neglect the particles that are outside the interpolation region Ω_I^k. Let $\hat{\mathcal{P}}^k_{in} \subset \hat{\mathcal{P}}^k_{\text{BBOX}}$ be the set of particles (white) that are within the interpolation region Ω_I^k. Figure 5.7b shows the set of particles that are not inside the interpolation region $p_i \notin \hat{\mathcal{P}}_{in}$ (gray).

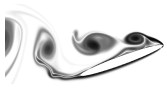
3.c) **Remove all the newly generated particles outside the interpolation region** Ω_I^k: Remove all the newly generated particles that outside the interpolation region $p_i \notin \hat{\mathcal{P}}_{in}$. Figure 5.7 shows the final distribution of the particles, with a uniformly distributed particles inside the interpolation region Ω_I^k without any gaps.

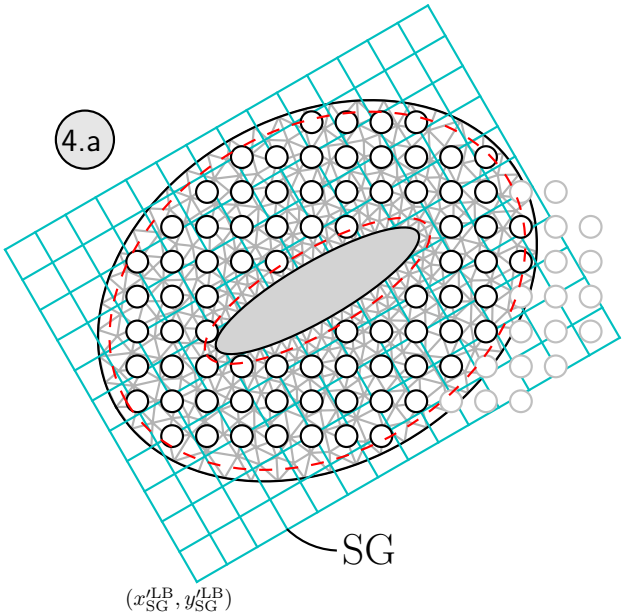
In combination with the *remove particle* step and the *generate particle* step, we can ensure that vortex blobs are uniformly distributed inside the interpolation region Ω_I^k without any gaps. If the initial distribution of the particles (as shown in Figure 5.6a) was not modified, we would see that due to the gaps in particle distribution not all vorticity could be interpolated to the Lagrangian method. However, now with the uniform distribution (as shown in Figure 5.7c), this is no longer the case.

5.2.4 Assign Strengths of Particles

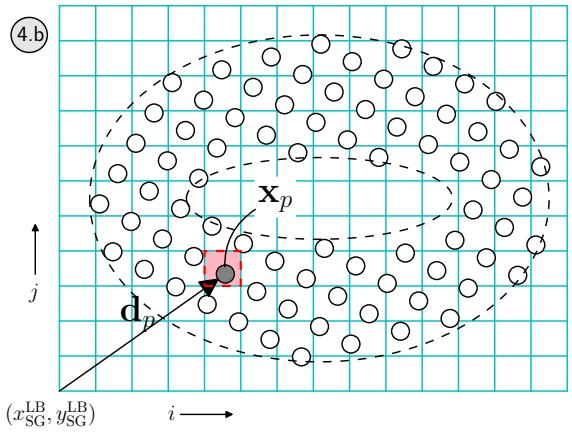
In this step, we will describe the algorithm for assigning strengths to the newly generated particles based on theory summarized in section 5.1.1. The strengths of the particle are determined by first interpolating the vorticity from the structured grid onto the position of the newly generated zero-strength vortex blobs, then using equation 5.2 to assign the strengths.

The algorithm for assigning the strengths of the newly generated particles inside the interpolation domain Ω_I^k is as follows:

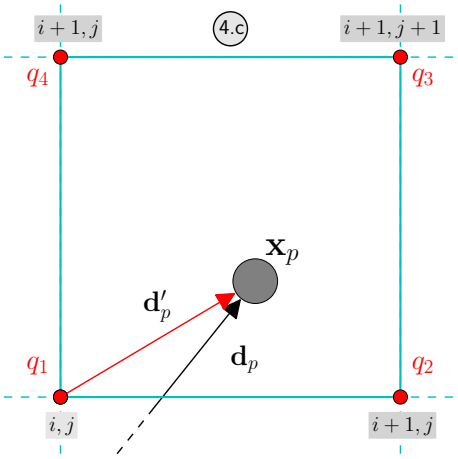




(a) Step 4.a: Determine the orientation of the SG w.r.t to the newly generated particles



(b) Step 4.b: Project vortex blobs onto the coordinate system of the SG



(c) Step 4.c: Determine the weights of the bilinear interpolation

Figure 5.8: Figures depicts the algorithm for assigning the strengths of the vortex particles inside the interpolation domain Ω_I^k.

4.a) **Determine the orientation of the SG w.r.t to the newly generated particles**: The interpolation algorithm starts by first determining the position of the structure grid SG w.r.t to the vortex blobs. The key parameters that define the position and the orientation of the structured grid in the global coordinate system is the position of the left bottom corner $(x_{\text{SG}}^{\text{LB}}, y_{\text{SG}}^{\text{LB}})$, and the rotational angle θ_{loc}, as described in section 5.2.1. Figure 5.8a, shows the global position of the structured grid w.r.t to the newly generated vortex blobs.

4.b) **Project the blobs into the coordinate system of the SG**: Once we know the position of the SG, we need to determine in which cells of the SG, the newly generated set of particles $\hat{\mathcal{P}}_{in}^{k}$ are located. To determine this, we need to project the vortex blobs onto the coordinate system of the structured grid. Figure 5.8b shows the projected orientation of the vortex blobs in the local coordinate system of the structured grid. Let $\mathbf{d}_p = |\mathbf{x}_p - \mathbf{x}_{SG}|$ be the offset of the particle p w.r.t to the left bottom corner of the structured grid. In the local coordinate system, the offset $\mathbf{d}_p$ is given by,

$$\mathbf{d}_p = \begin{bmatrix} d_{x,p} \\ d_{y,p} \end{bmatrix} = \begin{bmatrix} \cos\theta_{loc} & \sin\theta_{loc} \\ -\sin\theta_{loc} & \cos\theta_{loc} \end{bmatrix} \cdot \begin{bmatrix} x_p - x_{\text{SG}}^{\text{LB}} \\ y_p - y_{\text{SG}}^{\text{LB}} \end{bmatrix} \tag{5.22}$$

where $d_{x,p}$ and $d_{y,p}$ are the horizontal and the vertical distance from the left bottom corner, as shown in the Figure 5.8b.

In the local coordinate, the nodes of the structured grid are addressed by index i and j for x and y direction respectively. Let q_1, q_2, q_3, q_4 refer to the four nodes of the cell in which the particle p is located, as shown in Figure 5.8b). In this cases, the coordinates of these nodes in the local coordinate system are given as,

$$\begin{aligned} \mathbf{x}'_1 &= (i_p \cdot h, j_p \cdot h), \\ \mathbf{x}'_2 &= ([i_p + 1] \cdot h, j_p \cdot h), \\ \mathbf{x}'_3 &= ([i_p + 1] \cdot h, [j_p + 1] \cdot h), \\ \mathbf{x}'_4 &= (i_p \cdot h, [j_p + 1] \cdot h), \end{aligned} \tag{5.23}$$

where i_p and j_p are determined by a floor function,

$$i_p = \left\lfloor \frac{d_{x,p}}{h} \right\rfloor \tag{5.24a}$$

$$j_p = \left\lfloor \frac{d_{y,p}}{h} \right\rfloor . \tag{5.24b}$$

4.c) **Determine the weights of the bilinear interpolation**: Onces we determine in location of the particle p in terms of index i_p and j_p, we can set up a bilinear interpolation problem for transferring the vorticity from the nodes of the structured grid onto the position of the particle. The bilinear interpolation of vorticity is given as,

$$\hat{\omega}(\mathbf{x}_p) = \sum_{q=1}^{4} W_{q,p} \cdot \omega_q \tag{5.25}$$

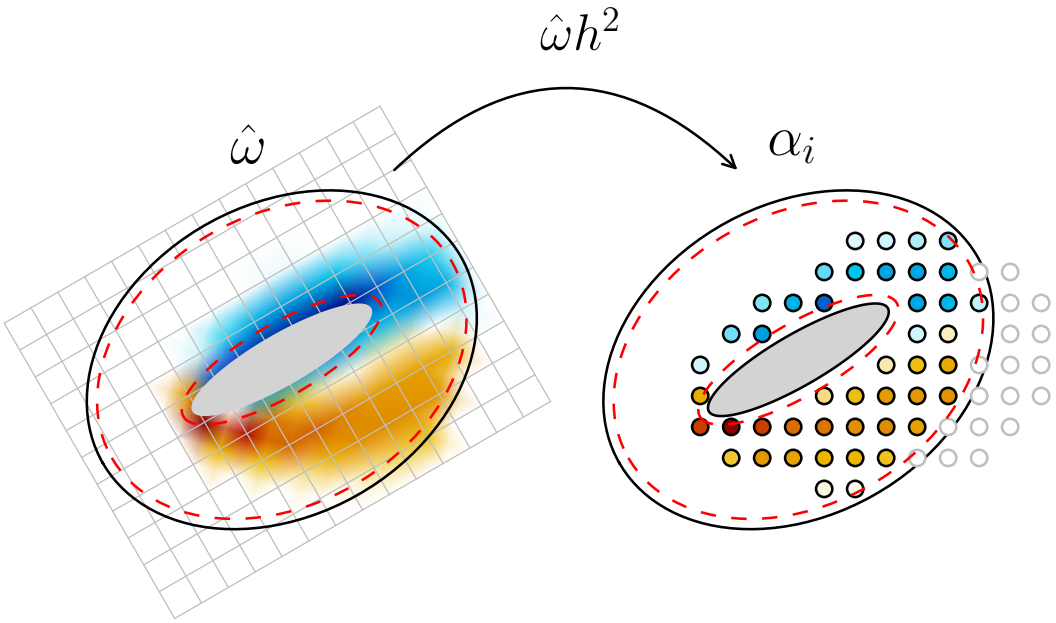


Figure 5.9: Bilinear interpolating the strengths from the structured grid SG onto the vortex blobs using equation 5.25

where $\hat{\omega}$ is the interpolated vorticity at the particle position $\mathbf{x}_p$. The weights of the bilinear interpolation $W_q p$ is given as,

$$\begin{aligned} W_{1,p} &= \frac{(d'_{x,p} - h)(d'_{y,p} - h)}{h^2}, \\ W_{2,p} &= \frac{-d'_{x,p}(d'_{y,p} - h)}{h^2}, \\ W_{3,p} &= \frac{d'_{x,p} d'_{y,p}}{h^2}, \\ W_{4,p} &= \frac{-d'_{y,p}(d'_{y,p} - h^2)}{h^2}, \end{aligned} \tag{5.26}$$

where $\mathbf{d}'_p = (d'_{x,p}, d'_{y,p})$ is given by,

$$\mathbf{d}'_p = \begin{bmatrix} d'_{x,p} \\ d'_{y,p} \end{bmatrix} = \begin{bmatrix} d_{x,p} - i_p \cdot h \\ d_{y,p} - j_p \cdot h \end{bmatrix} \tag{5.27}$$

such that $0 \leqslant d'_{x,p} \leqslant 1$ and $0 \leqslant d'_{y,p} \leqslant 1$ and reflects the offset from the four nodes.

4.d) **Interpolate the vorticity from *SG* onto the particle**: Finally, we can solve equation 5.25 to interpolate the vorticity from the nodes of the structured grid onto the pre-generated particle positions. Once we know the vorticity at their position, we can use equation 5.2 to assign the strengths of the particles. These particles will then have a total circulation of $\hat{\Gamma}_I^{L,k}$ according to equation 5.10. Figure 5.9 depicts the transfer of the Eulerian solution from the structured grid onto the particle.

However, as we described in section 5.1, we must still determine the vortex sheet strengths that satisfied a no-slip boundary condition and ensure conservation of circulation in the domain Ω_{in}^k due to the correction.

5.2.5 Correct Total Circulation

In this step, we will determine the strengths of the vortex sheet and the total circulation of newly generated vortex particles inside the correction region Ω_{in}^k such that the circulation conserved during the Lagrangian correction step. The algorithm is however depended on whether the problem is unbounded (without wall boundaries), or bounded (with wall boundary). If we do not have a wall boundary, the step to determine the vortex sheet is not applicable because there exists no vortex sheet.

The algorithm for correcting the total circulation inside Ω_{in}^k is as follows:

5.a) **Determine the strengths of the panels** (with solid wall): The total strength of the vortex panels $\Gamma_\gamma^{L,k}$ is determined from equation 5.11. The Eulerian circulation $\Gamma_{in}^{E,k}$ is the circulation within the boundary Σ_o^k (see Figure 5.2) and also includes the circulation within the Eulerian body Ω_b. In the case of a moving boundary, there exists a non-zero circulation within the body. The circulation $\Gamma_I^{L,k}$ is determined from step 4.d and represents the total circulation of the newly generated vortex particles inside the domain Ω_I^k. Once we determine the required strength for the vortex sheet $\Gamma_\gamma^{L,k}$ for all the k Eulerian domains, we can use the approach described in section 3.5 to solve the multi-body vortex panel problem, satisfy the no-slip boundary condition.

5.b) **Correct the strengths of particles to conserve circulation**: The final step of the *modified Lagrangian correction* algorithm is to ensure conservation of total circulation. This can be achieved by determining the adjusting the strengths of particles $\tilde{\alpha}_p$ using equation 5.19. Note that, in case we have unbounded problem, we have the strength of the vortex sheet $\Gamma_\gamma^{L,k} = 0$.

5.3 Determining Eulerian Substep Boundary Conditions

In this section, we elaborate the methodology for determining the Eulerian boundary conditions from the evolved solution of the Lagrangian method. Furthermore, we will explain how to determine the boundary conditions for the substeps of the evolution.

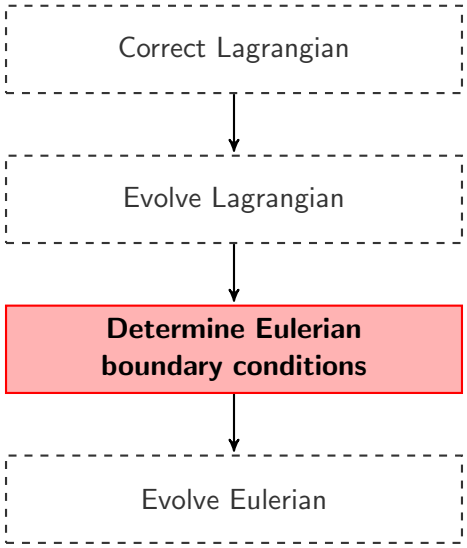


Figure 5.10: Flowchart of the hybrid evolution, focusing on the 3rd step: Determine the Eulerian boundary conditions.

We can determine the Dirichlet velocity boundary condition at the Eulerian boundary $\partial\Omega_E$ from the Lagrangian vorticity field ω at t_{n+1}. The velocity at the Eulerian boundary $\mathbf{x}_{bdry}$ is given by,

$$\mathbf{u}(\mathbf{x}_i, t_{n+1}) = \sum_p \mathbf{K}_\sigma[\mathbf{x}_i - \mathbf{x}_p(t_{n+1})]\alpha_p(t_{n+1}), \tag{5.28}$$

where $\mathbf{x}$ are the coordinates of the nodes of the Eulerian mesh that lie in the boundary Σ_d, as shown in figure 5.11.

5.3.1 Multi-step evolution

In section 2.3.1, we defined the Eulerian time step Δt_E and the Lagrangian time step, such that $\Delta t_E \leqslant \Delta t_L$. The main advantage of this different time step size is that the wake region, where the Lagrangian solver lies, can now be evolved with a much larger time step size.

We perform k_E Eulerian substeps within one Lagrangian time step t_{n+1},

$$t_n^k = t_n + k\Delta t_E, \tag{5.29}$$

where $k = \{0, ..., k_E\} \subset \mathbb{N}_0$, and is given by,

$$k_E = \frac{t_{n+1} - t_n}{\Delta t_E} = \frac{\Delta t_L}{\Delta t_E}. \tag{5.30}$$

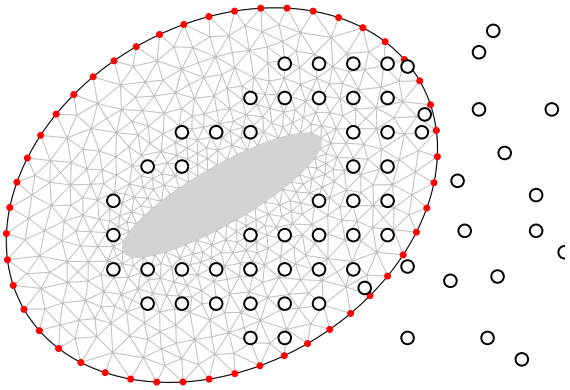


Figure 5.11: Dirichlet boundary conditions at boundary of Eulerian domain Σ_d. We evaluate the induced velocities from the Lagrangian solution at the nodes $\mathbf{x}_i \in \Sigma_d$ [●, red dot].

and we require that Δt_L is a multiple of Δt_E. When $k = 0$, we have $t_n^0 = t_n$ and when $k = k_E$, we have $t_n^{k_E} = t_{n+1}$. Figure 5.12 depicts the multi-stepping of the Eulerian solution from t_n^0 to $t_n^{k_E}$ to match the Lagrangian time. As the Eulerian solver requires boundary conditions at each substep, we have to perform an interpolation of the boundary conditions for each substep t_n^k. We can perform a linear interpolation of the boundary condition in order to determine the boundary conditions at each sub-step,

$$\mathbf{u}(t_n^k) = \mathbf{u}(t_n) + k\Delta\mathbf{u}, \tag{5.31}$$

where $\Delta\mathbf{u}$ is given as

$$\Delta\mathbf{u} = \frac{\mathbf{u}(t_n^{k_E}) - \mathbf{u}(t_n^k)}{k_E}, \tag{5.32}$$

and is a linear variation of the velocity between each substep.

Once that we have the boundary condition for each of the Eulerian substeps, we can use the approach described in section 4.4 to evolve the Eulerian method from t_n to t_{n+1} using k_E Eulerian substeps.

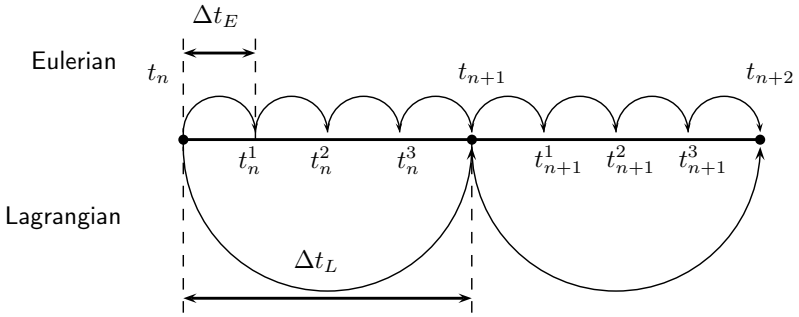


Figure 5.12: Eulerian multi-stepping to match the Lagrangian time step. The figure shows $\Delta t_L = 4\Delta t_E$ and requires $k_E = 4$ Eulerian substeps to time march from t_n to t_{n+1}.

Chapter 6

Introduction to the Hybrid Solver: pHyFlow

We have implemented the algorithms described in chapters 2 to 5 into `pHyFlow`, an acronym for **p**ython **Hy**brid **Flow** solver. `pHyFlow` functions as a fluid dynamics computational library written in python, that has implemented the Eulerian solver, the Lagrangian solver (without vorticity diffusion of panels). These solver can used as a standalone solver (for test purposes), or can be coupled together to make the Hybrid solver.

The features of `pHyFlow` can be summarized as follows:

- `pHyFlow` is a hybrid flow solver that uses Hybrid Eulerian-Lagrangian Vortex Particle Method to couple the Navier-Stokes grid solver and a vortex blob solver.
- The algorithms are written in Python , Cython [4], C, C++, and CUDA C/C++ for efficiency. All the high-level algorithms such as the definition of the problem, coupling of the solver, convection and diffusion of the problem are implemented in Python . The low-level algorithms such as the remeshing kernel and the routine for saving are written in the computationally efficient languages: Cython, C, and C++. The parallelizable routines such as the calculation of the induced velocities of the vortex blobs are written in CUDA C/C++ for the NVIDIA GPU hardware.
- `pHyFlow` uses several open-source libraries: FEniCS [46], Fenicstools [49], Scipy [36], Numpy [65], mpi4py [25], pyUblas [38], for performing the calculations; and PyVTK [38], H5py [17], Matplotlib [34] for plotting and efficient data storage.
- `pHyFlow` is maintained, and is available at the bitbucket online repositiory `https://bitbucket.org/apalha/phyflow2.0`.

6.1 Program structure

The `pHyFlow` library extends Python computing environment, where one can solve hybrid flow problems. To achieve this, we have implemented an Eulerian solver and a

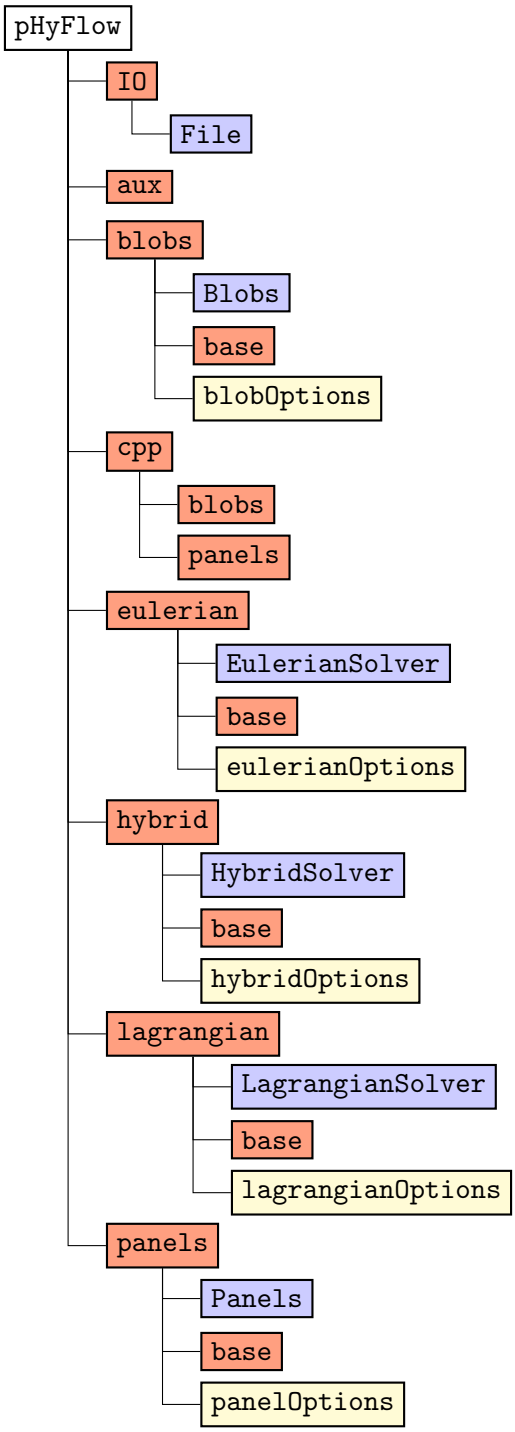


Figure 6.1: Flowchart of the pHyFlow library structured into modules, option script files, and classes.

Lagrangian solver (without panel diffusion scheme, which can be used as a standalone solver for verification and validation). The `pHyFlow` library is structured into several modules, categorized by their purposes. In each `module`, we defined a `class` that handles the functions in the module. To add flexibility in computation, we added an `option` file where the user can change the solver's options. Figure 6.1 shows the structure of the `pHyFlow` library, classified using a color code. The structure of the `pHyFlow` is as follows:

- `IO`: This module contains all the input/output functions for saving and plotting data. The `File` class handles the functions of the `IO` module.
- `aux`: This module contains all the auxiliary function of the library that does not belong to the fluid dynamics computation.
- `cpp`: The module that contains all the low-level compiled functions that have been wrapped for the use in `python`. This module contains the two main low-level algorithms for performing the induced velocity calculations for vortex blobs and vortex panels, and the remeshing algorithm for the vortex blobs.
- `blobs`: This module contains all the vortex blob related functions. Contains the class `Blobs` and the vortex blob solver object, handling all the vortex blobs operations. The algorithms of the vortex blobs defined in chapter 3 are implemented in this module.
- `panels`: This module contains all the vortex panel functions and is wrapped in the class `Panels`. The algorithms of the vortex panel defined in chapter 3 are implemented in this module
- `lagrangian`: This module contains all the vortex blob and vortex panel coupling functions. The vortex panel, vortex blob coupling algorithm described in chapter 3 is implemented in this module.
- `eulerian`: This module contains all the Navier-Stokes grid operations. The algorithms explained in chapter 4 are implemented in this module.
- `hybrid`: This module contains all the functions related to coupling of the Lagrangian and the Eulerian solver, summarized in section 5.2, 3.6, and 4.4. The functions are wrapped in the `HybridSolver` class and manage the global coupling process.

Figure 6.1 shows the structure of the `pHyFlow` library and is categorized into several modules of different purposes. It is structured in this manner such that one could employ the library for any general simulation purpose such as the hybrid case, or for non-hybrid cases (e.g. potential flow using vortex panels, full Eulerian grid simulation using Eulerian solver). This means that one could use a single module of `pHyFlow` library for the desired test case.

6.2 Hybrid class Hierarchy

However, the hybrid module relies on the functions of the Lagrangian module and the Eulerian module. Moreover, the Lagrangian module requires the function of vortex blob

module and the vortex panel module. Therefore, the hierarchy of the hybrid class is defined in a different manner, as shown in figure 6.2.

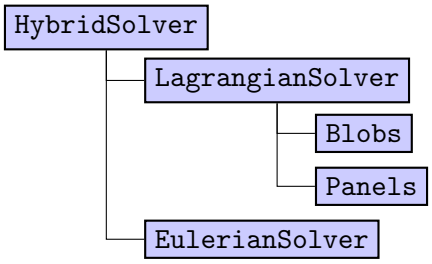


Figure 6.2: Flowchart of the `HybridSolver` hierarchy. The `HybridSolver` couples the `LagrangianSolver` class and the `EulerianSolver` class using the hybrid coupling schemes.

We use a bottom-up approach to construct the `HybridSolver` object, starting from the lower-level objects: `Blobs`, `Panels`. Then constructing the mid-level objects: `LagrangianSolver`, and `EulerianSolver`. Finally constructing the highest-level object: `HybridSolver`. The procedure for constructing the hybrid class is as follows:

1. Construct the lowest-level objects:
   (a) Construct the `Blobs` object using the vorticity field parameters, the vortex blob parameters, time step parameters, and population control parameters.
   (b) Construct the `Panels` object using panel geometry parameters.
2. Construct the mid-level solvers:
   (a) Construct `LagrangianSolver` object using the vortex blob object `Blobs` and the vortex panel object `Panels`.
   (b) Construct `EulerianSolver` object using the geometry mesh file, interpolation probe grid parameters, and the fluid parameters.
3. Construct the hybrid solver:
   (a) Construct `HybridSolver` object using the Lagrangian solver object `LagrangianSolver`, the Eulerian solver object `EulerianSolver`, and the interpolation parameters.

A detailed description of the parameters required for the construction of the objects, and the schematic of these objects are given in appendix B.

Chapter 7

Verification and Validation of Hybrid Method

This chapter focuses on the verification and the validation of the hybrid method. The implementation of the hybrid method was first verified using the analytical solutions of the Lamb-Oseen vortex problem, as the test case helped us quantify any errors in the coupling. The validation of the hybrid solver was performed using the test cases provided from literature. The Clercx-Bruneau dipole collision problem of Clercx and Bruneau [16], provide a detailed analysis on the evolution of the vorticity field and the generation of vorticity from the wall boundary. The impulsively started cylinder problem investigated by Koumoutsakos and Leonard [41], and Rosenfeld et al. [54] showed the evolution of the forces acting on an immersed cylinder in a free-stream flow.

The final stage of the validation was to show the feasibility of simulation the flow around a full VAWT. To perform this we investigated the flow past a stalled airfoil at $Re = 5000$, and the flow around a multi-body problem with two cylinders.

7.1 Lamb-Oseen Vortex Evolution

The Lamb-Oseen vortex test case simulates the evolution of a laminar vortex core in an unbounded domain. In section 3.7.2, we used this test case to verify and validate the implementation of the vortex blobs of the Lagrangian solver and in section 4.5.1, we used it to verify the implementation of the Eulerian solver. Therefore, in a similar fashion we employed this test case to verify the coupling of the hybrid solver.

The unbounded nature of the problem helped us to remove the influence of the vortex panel on the coupling strategy, and helps us focus on the vorticity field interpolation error discussed in section 5.1.1. Furthermore, with the analytical solution, we were able to quantitatively present the importance of ensuring conservation of circulation.

Moreover, we were able to quantify the influences of the discretization on the coupling. The parameters that determine the spatial discretization of the vortex blobs are the nominal particle spacing h, and the overlap ratio λ (see Figure 3.4). The spatial discretization of the Eulerian solver is regarded as a control variable for this test case as its impact was concluded in section 4.5.1. The parameters that determine the temporal discretization of the hybrid method are the time step size of the Eulerian solver Δt_E, and the time step size of the Lagrangian solver Δt_L. These were determined according to equation 5.30, with k_E being the number of Eulerian sub-steps.

7.1.1 Problem Definition

The Lamb-Oseen Vortex problem is defined by the vorticity field and the velocity field, equations 3.61 and 3.62, respectively. The hybrid solver is initialized by first assigning the strengths of the vortex blobs using equation 3.63. The Eulerian domain Ω_E is then initialized using the solution of the Lagrangian solver. Daeninck [24] used this approach to enhance the coupling between the methods ensuring minimum interpolation error.

Figure 7.1 shows the hybrid domain configuration for the Lamb-Oseen vortex problem with the Lagrangian domain Ω_L spanning the full fluid domain. The Eulerian domain Ω_E only resolves the center of the Lamb-Oseen core, $\Omega_E = [-0.5, -0.5] \times [-0.5, -0.5]$. The boundary of the Eulerian domain $\partial\Omega_E$ is a Dirichlet velocity boundary $\partial\Omega_E = \Sigma_d$ where the velocity boundary condition is applied, as described in section 2.3.1. The correction of the Lagrangian domain is performed in the interpolation domain Ω_I according to the procedures described in section 5.2. The outer boundary of the interpolation domain Σ_o is defined with an offset d_{bdry} from the Eulerian boundary Σ_d by a distance $d_{bdry} = 2 \cdot h$, where h is the nominal blob spacing. Similar choice was made by Stock [61] (discussed in section 2.3.2), and ensures that the potential inaccuracies at the outer Eulerian boundary is ignored during the interpolation procedure.

The spatial discretization of the Eulerian domain Ω_E is regarded as a control (i.e fixed) variable. Therefore, the parameter sensitivity analysis was performed by varying the

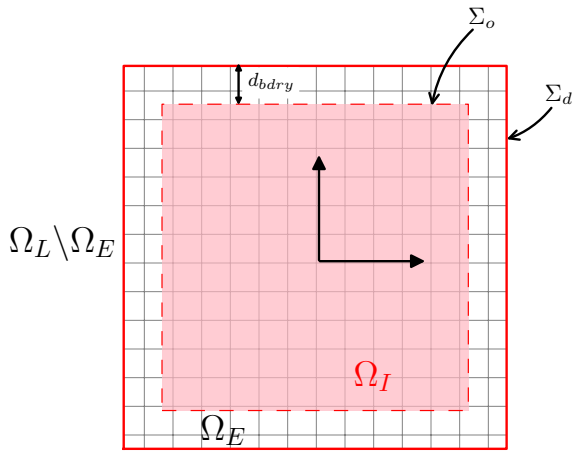


Figure 7.1: (*Not to scale*) The domain decomposition for the Lamb-Oseen vortex problem, Ω_E. The Eulerian domain is defined as $\Omega_E = [-0.5, 0.5] \times [-0.5, 0.5]$ with Dirichlet boundary Σ_d [—, solid red]. The parameters of the discretization are tabulated in table 7.1.

Table 7.1: Summary of the parameters for the Lamb-Oseen vortex evolution.

Parameters	Value	Unit	Description
Γ_c	1	$\mathrm{m}^2\,\mathrm{s}^{-1}$	Core strength
Ω	$[-0.5, 0.5] \times [-0.5, 0.5]$	m	Eulerian domain bounds
ν	0.001	$\mathrm{kg}\,\mathrm{s}^{-1}\,\mathrm{m}^{-1}$	Kinematic viscosity
τ	100	s	Lamb-Oseen time constant
λ	1	-	Overlap ratio
h	0.01	m	Nominal blob spacing
$(\Gamma_{loc}, \Gamma_{glob})$	$(1 \times 10^{-14}, 1 \times 10^{-14})$	-	Population control threshold
h_{grid}	0.007 to 0.016	m	FE cell diameter span
N_{cells}	26448	-	Number of mesh cells
Δt_L	0.001	s	Lagrangian time step size
Δt_E	0.001	s	Eulerian time step size
k_E	1	-	Eulerian sub-steps
$N_{\mathrm{t-steps}}$	1000	-	Number of time integration steps
t	0 to 1	s	Simulation time span
d_{bdry}	$2 \cdot h$	m	Interpolation boundary offset from Σ_o

discretization of the Lagrangian method only. The Eulerian domain is discretized with an unstructured mesh generated with GMSH (see section 4.2.2) having, $N_{cells} = 26448$ and grid size h_{grid} ranging from 0.007 to 0.0016.

The Lamb-Oseen vortex problem is defined according to the parameters tabulated in table 7.1. The center of the core is located at $(x, y) = (0, 0)$, where the Eulerian domain Ω_E is also centered. The parameters are chosen such that vorticity ω and velocity $\mathbf{u}$ is non-zero at the boundary of the Eulerian domain Σ_d, Figure 7.1. The evolution of the Lagrangian solver and the Eulerian solver is performed according to sections and 4.4 respectively.

7.1.2 Results and Discussion

The investigation of the Lamb-Oseen vortex problem is divided into three parts. The first part of the investigation concerns with comparing several stages of the hybrid coupling, defined in section 7.1.2.1. We compared the uncoupled scheme with the one-way coupled scheme and with the fully coupled scheme. These successive coupling investigation helped us determine the source of the error, and furthermore quantify the errors in the coupling process. The second part of the investigation, section 7.1.2.2 focuses on importance of conservation of circulation that was discussed in section 5.1.3. The results of the non-conserved and conserved scheme are compared to conclude the importance of conservation of circulation. During these two investigations, the parameters tabulated in table 7.1 are used. The third and final investigation is dedicated to the parameter sensitivity analysis, section 7.1.2.3. The parameters that determine the spatial and temporal discretization of the scheme is investigated to verify the convergence of scheme.

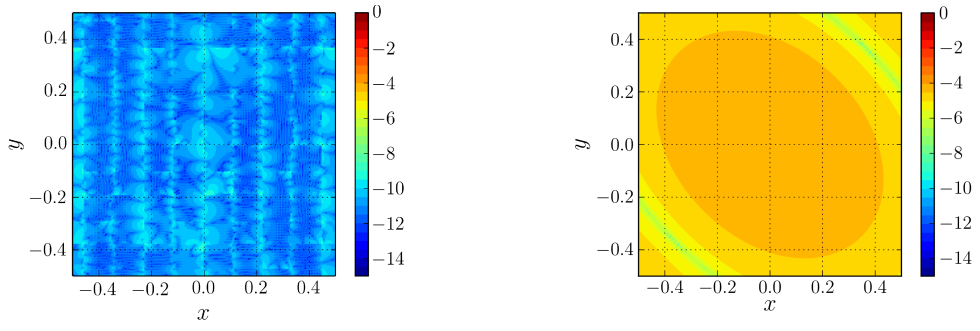
7.1.2.1 Uncoupled vs. One-way Coupled vs. Fully Coupled

The several stages of the hybrid coupling with the fully Eulerian test case, to verify the implementation of the hybrid algorithm. The three stages of the coupling are as follows:

- **Uncoupled**: In the uncoupled test case, we only focus on the Eulerian subdomain of the hybrid scheme, Figure 7.1. The boundary conditions for the Eulerian Dirichlet boundary are obtained directly from the analytical solution, equation 4.39. This test helps us quantify the base error in the FEM and furthemore serves as a benchmark for further investigation.
- **One-way coupled**: The one-way coupled test case is a partially coupled hybrid scheme. Here the Eulerian method obtains the boundary condition from the Lagrangian method and the Lagrangian subdomain remains unmodified throughout the simulation. The purpose of this test cases is to quantify any error due the transfer of the solution from Lagrangian to Eulerian method.
- **Fully coupled**: The fully coupled test case performs the full coupling strategy, according to the steps described in section 2.3.1. The Eulerian method is evolved using the Lagrangian solution. At the end of each time step, the Lagrangian solution inside the interpolation domain Ω_I, Figure 7.1, is corrected. This test case helped us quantify the error in transferring the Eulerian solution to the Lagrangian method.

Figure 7.2 depicts the initial relative error in velocity and vorticity inside the Eulerian domain Ω_E. The relative error in velocity is near machine epsilon $\epsilon \leq 10^{-10}$, but the error in vorticity is in the order of 10^{-5}. This occurs because the solution was initialized using the velocity field and the vorticity was determined numerically as described in section 4.3.2. Therefore, the increase in the error shown in figure is as expected.

Figure 7.3 shows the evolution of maximum relative error in vorticity ω and velocity $\mathbf{u}$ of the uncoupled, one-way coupled and the fully coupled cases inside the Eulerian domain Ω_E w.r.t. the analytical solution, equation 3.61. The initial observation shows that the error in velocity is two to three orders of magnitude lower than the error in vorticity. The figure shows that the uncoupled scheme has the lowest error in vorticity and velocity. This



(a) Velocity **(b)** Vorticity

Figure 7.2: Initial relative error at $t = 0$ inside the Eulerian domain Ω_E. The figure depicts **(a)** the relative error in velocity $\mathbf{u}$, and **(b)** the relative error in vorticity ω.

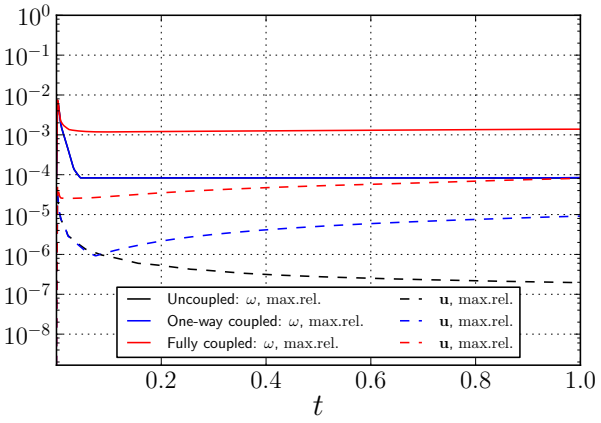


Figure 7.3: Evolution of the maximum relative error in velocity (dashed) and the maximum relative error in vorticity (solid), equation 3.64, from $t = 0$ to $t = 1$, using the parameters tabulated in table 7.1. The figure compares uncoupled case (**black**) vs. the one-way coupled case (**blue**) vs. the fully coupled case (**red**).

is expected as the boundary condition is directly obtained from the analytical solution and therefore minimum error is introduced. As time progresses, the error in velocity converges near 10^{-7} and the error in vorticity converges near 10^{-4}.

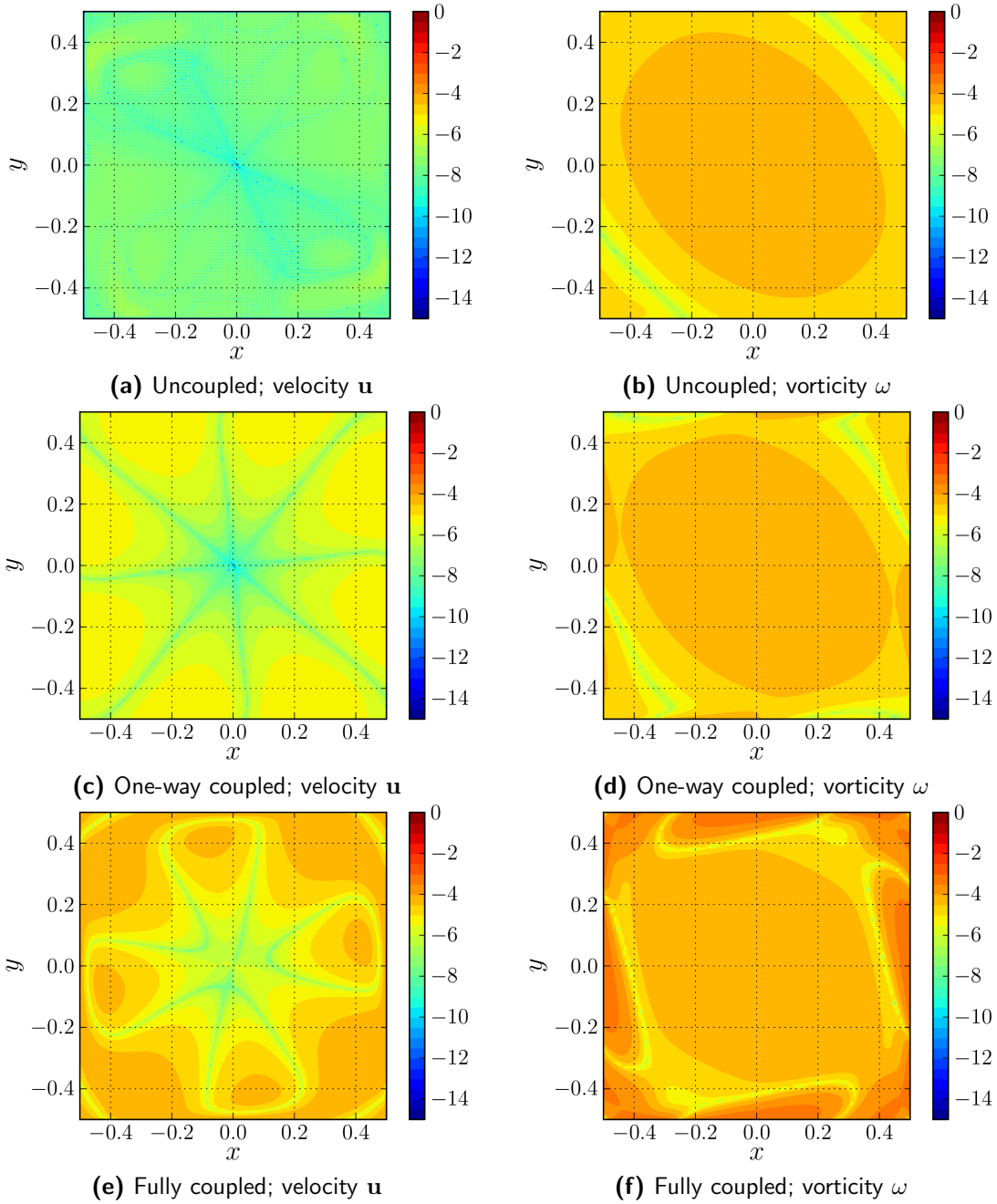
The one-way coupled case shows an increase in the relative error in velocity field $\mathbf{u}$ inside the Eulerian domain Ω_E. However, the difference is negligible at the initial stages of the simulation. This states that the analytical solution was well represented by the vortex blobs with a small discretization error. At $t = 1$, the error in velocity increases by two orders of magnitude from 10^{-7} to 10^{-5}, w.r.t to the uncoupled scheme. This was expected as the velocity boundary conditions for the Eulerian domain is obtained from the discrete solution of the Lagrangian method. In section 3.7.2, we observed the behavior of evolution of the Lagrangian solution and this behavior is introduced into the Eulerian method. To investigate the convergence of this error, we will later investigated the error for various Lagrangian discretizations.

The fully coupled case demonstrates that there is an additional source of error. Unlike the one-way coupled case, the increase in the error is observed from the start of the simulation, implying that this is due to the correction of the strengths of the vortex blobs. In section 5.1.1, we discussed that the re-initialization of the vortex blobs introduces smoothing error in the vorticity field (i.e the Gaussian blurring of the vorticity field). This causes the Lagrangian solution to further deviate from the analytical solution of the Lamb-Oseen vortex. The consequence of this was that at $t = 1$, the error in vorticity ϵ_ω increased from 10^{-4} to 10^{-3} and the error in velocity increased from 10^{-5} to 10^{-4}, w.r.t the one-way coupled case.

Figure 7.4 shows the relative error in velocity and vorticity inside the Eulerian domain Ω_E at $t = 1$, for the three stages of coupling. We observe that there is an increase in error when going from the uncoupled scheme to the one-way coupled scheme to the fully coupled. Comparing the uncoupled scheme to the one-way coupled scheme, an increase in error is observed at the boundary of the Eulerian domain Σ_d. Comparing the one-way coupled to fully coupled case, we see that there is an additional increase in the error

from the boundary. Therefore, the artificial vorticity generated from the boundary is due to the mismatch in the solutions of the Eulerian and the Lagrangian method. A larger mismatch in the solution will introduce strong artificial vorticity at the boundary.

Thus, we were able to determine the sources of the errors in the hybrid coupling. The error in vorticity, represented by the artificial vorticity near the outer Eulerian boundary



(a) Uncoupled; velocity $\mathbf{u}$ **(b)** Uncoupled; vorticity ω

(c) One-way coupled; velocity $\mathbf{u}$ **(d)** One-way coupled; vorticity ω

(e) Fully coupled; velocity $\mathbf{u}$ **(f)** Fully coupled; vorticity ω

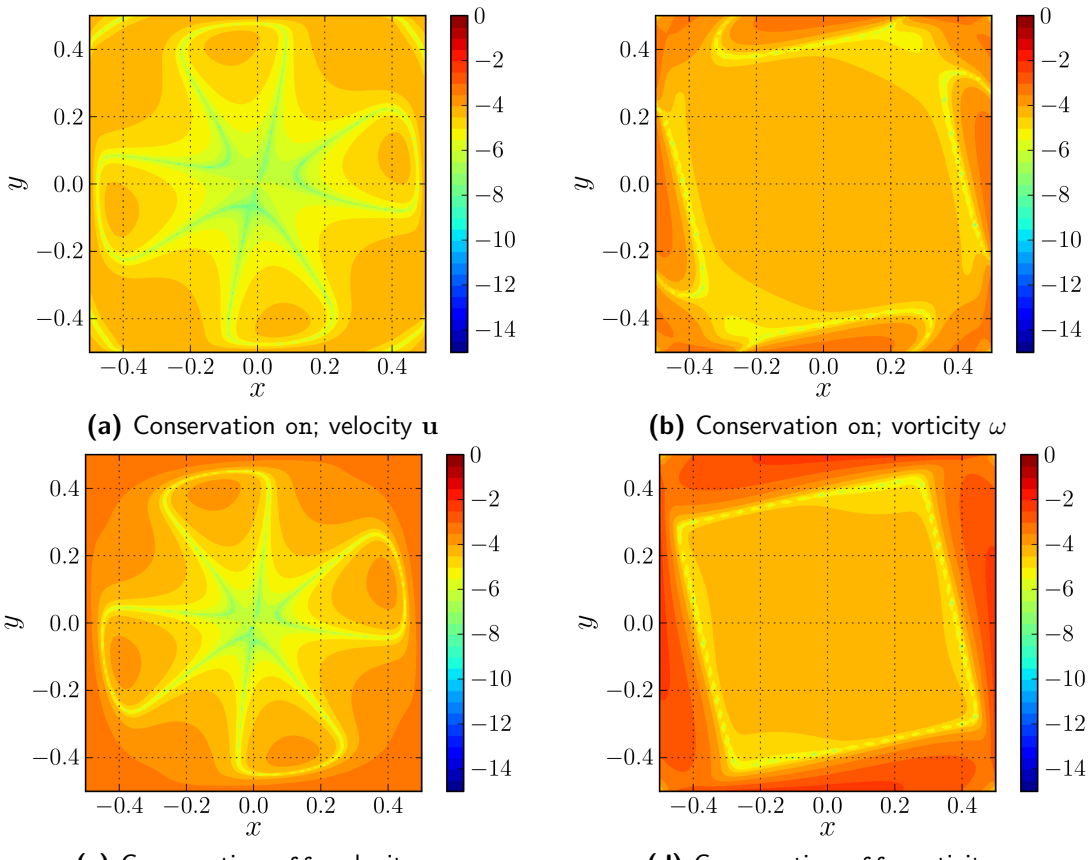
Figure 7.4: Plot of the relative error in velocity (*left*) and relative error in vorticity (*right*) in the Eulerian domain Ω_E at $t = 1$. The figure compares the error between **(a)(b)** the uncoupled, **(c)(d)** the one-way coupled, and **(e)(f)** the fully coupled cases.

Σ_o, is a culmination of the discretization error of the Lagrangian method and the Eulerian method. Therefore, if we were to use a higher resolution, we expect a greater matching of the Eulerian method and the Lagrangian method. Section 7.1.2.3, is dedicated to validate this claim.

7.1.2.2 Conservation of circulation

The approach for ensuring conservation of circulation during the coupling process was discussed in section 5.1.3. To validate the importance of conservation of circulation, we ran two simulations with and without the conservation of circulation during the transfer of vorticity from the Eulerian method to the Lagrangian method. The control variables of the simulation are the parameters tabulated in table 7.1.

Figure 7.5 compares the error in the Eulerian domain Ω_E of the coupling approach without the conservation circulation against the approach satisfying the conservation of circulation, at $t = 1$. We see that the scheme without the conservation has significantly larger error than the scheme with conservation. The maximum error is near the Eulerian boundary Σ_d and shows an increase in the artificial vorticity emanating from the boundary due



(a) Conservation `on`; velocity $\mathbf{u}$

(b) Conservation `on`; vorticity ω

(c) Conservation `off`; velocity $\mathbf{u}$

(d) Conservation `off`; vorticity ω

Figure 7.5: Plot of the relative error in velocity (left) and the relative error in vorticity (right) in the Eulerian domain Ω_E at $t = 1$. The figure compares the error between **(a)(b)** without conservation of circulation, and **(c)(d)** with the conservation of circulation.

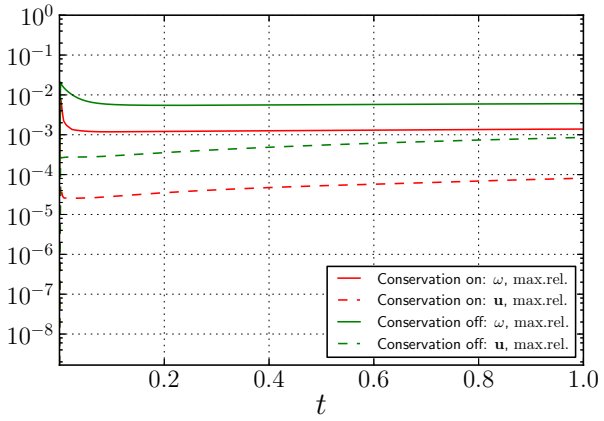


Figure 7.6: Plot of the maximum relative error in vorticity ϵ_ω [- -, dashed] and maximum relative error in vorticity $\epsilon_{\mathbf{u}}$ [—, solid], equation 3.64, from $t = 0$ to $t = 1$, using the parameters tabulated in table 7.1. The figure compares the coupling scheme with conservation of circulation (**red**) vs. the coupling scheme without conservation of circulation (**green**).

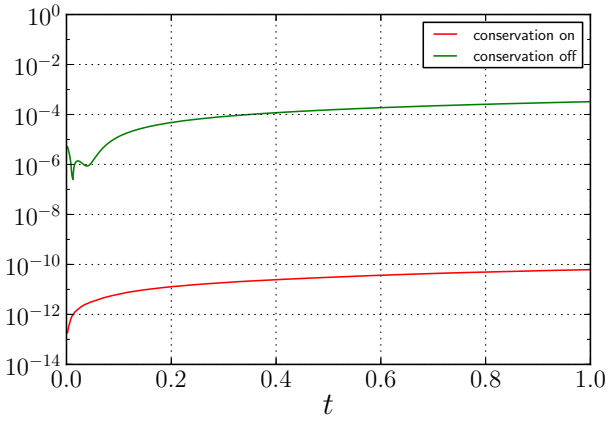


Figure 7.7: Plot of the error in total circulation ϵ_Γ of the Lagrangian method from $t = 0$ to $t = 1$. The figure compares the scheme with conservation of circulation [—, solid **red**], and the scheme without conservation of circulation [—, solid **green**].

to the larger mismatch in the solutions, Figure 7.5d. However, when we ensure that circulation is conserved, Figure 7.5b, the boundary produces significantly less error.

Figure 7.6 shows the evolution of the maximum relative error from $t = 0$ to $t = 1$, comparing the results of with and without the conservation of circulation. Observing the difference in the relative error in velocity and vorticity, we see that the scheme without the conservation of circulation produces larger errors at all times t. At $t = 1$, we observe that the scheme without the conservation of circulation has a relative error in vorticity near 10^{-2}, whereas with the conservation of circulation, the relative error is an order of magnitude lower, reaching only 10^{-3}. Similarly for velocity, the scheme without the conservation of circulation has the relative error approaching 10^{-3}, whereas with the conservation enabled, the error only reaches 10^{-4}.

Figure 7.7 shows the change in total circulation from $t = 0$ to $t = 1$ for the non-conserved and conserved scheme. It is apparent that without the conservation of circulation, the error in total circulation is significantly larger and approaches 10^{-3}. If the circulation is not conserved explicitly, the transfer of vorticity from the Eulerian method to the Lagrangian method introduces additional error in total circulation. By ensuring conservation of circulation, as described in section 5.1.3, we see that the error in total circulation is significantly smaller, near 10^{-10}.

Thus, we have determined that another potential source of error during the coupling of the two methods is the mismatch in the circulation. If we do not ensure conservation of circulation, the error in coupling is reflected by generation of large artificial vorticity, emanating from the outer Eulerian boundary Σ_o. The results validates this observation and therefore the algorithm described in section 5.1.3 is vital for valid hybrid scheme.

7.1.2.3 Parameter sensitivity analysis

The final question we have to answer is how discretization effects the coupling process. To determine this we varied the spatial and temporal resolution of the Lagrangian method and quantified the increase in the error.

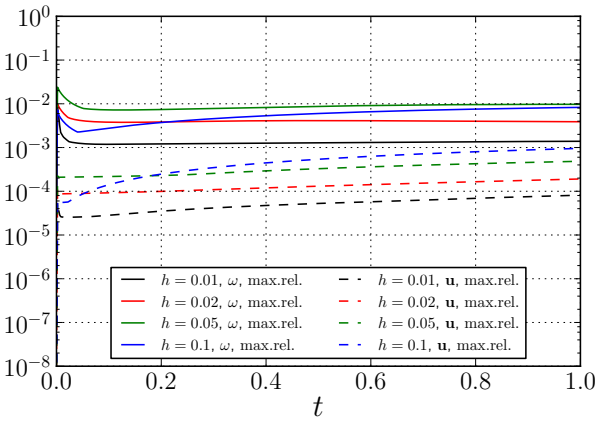


Figure 7.8: Evolution of the maximum relative error for various nominal blob spacing $h = [0.01, 0.02, 0.05, 0.1]$ from $t = 0$ to $t = 1$.

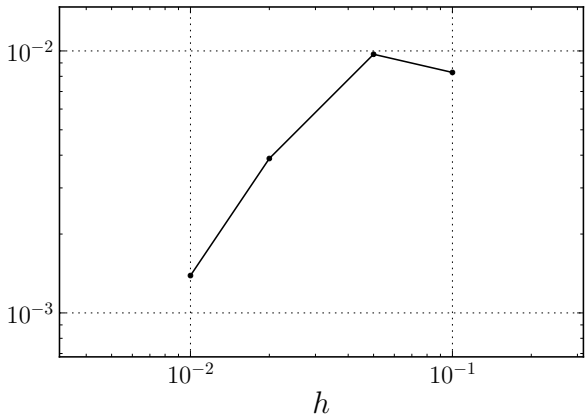


Figure 7.9: Convergence of the maximum relative error in vorticity due to the nominal blob spacing $h = [0.01, 0.02, 0.05, 0.1]$.

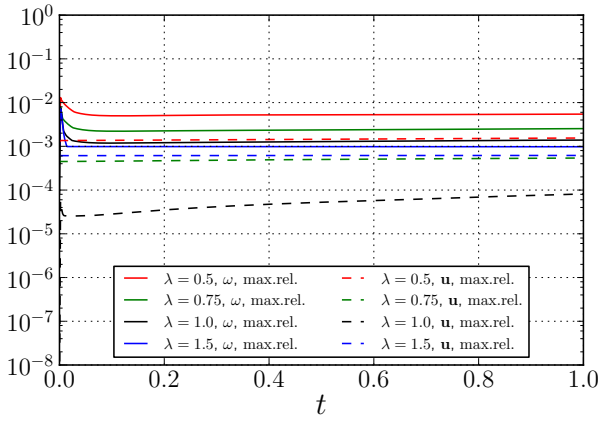
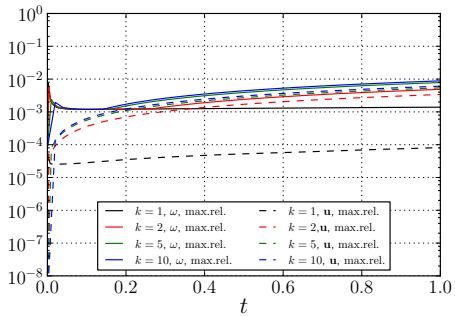
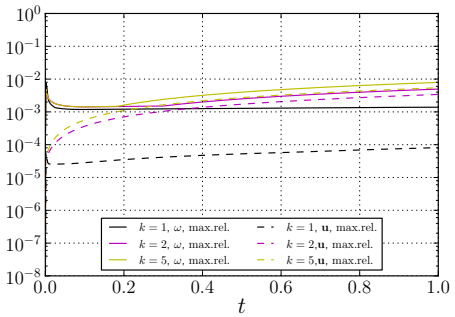


Figure 7.10: Evolution of the maximum relative error for various overlap ratios $\lambda = [0.5, 0.75, 1.0, 1.5]$ from $t = 0$ to $t = 1$. The figures shows the maximum relative error in vorticity [- -, dashed] and the maximum relative error in vorticity [—, solid].



(a) $\Delta t_L = [0.001, 0.002, 0.005, 0.01]$, $\Delta t_E = 0.001$



(b) $\Delta t_E = [0.001, 0.0005, 0.0002]$, $\Delta t_L = 0.001$

Figure 7.11: Evolution of the maximum relative error from $t = 0$ to $t = 1$ for various number of Eulerian sub-steps $k_E = [1, 2, 5, 10]$, modifying (**a**) the Lagrangian time step Δt_L, and (**b**) the Eulerian time step Δt_E. The figures shows the maximum relative error in vorticity [- -, dashed] and maximum relative error in vorticity [—, solid].

The spatial discretization of the Lagrangian method was modified by varying the nominal blob spacing h and the overlap ratio λ. The temporal discretization was modified by varying the Lagrangian time step size Δt_L. The control variables of the simulations are the ones tabulated in table 7.1.

Figure 7.8 shows the impact of varying the nominal blob spacing h on the coupling. The maximum relative error in vorticity and the maximum relative error in velocity is plotted from $t = 0$ to $t = 1$ for nominal blob spacing $h = [0.01, 0.02, 0.05, 0.1]$. The figure shows that increasing the spatial resolution of the Lagrangian method reduces the error. At $t = 1$, the minimum error is observed for $h = 0.01$ with the relative error in velocity at 10^{-4} and the relative error in vorticity at 10^{-3}. The maximum relative error is observed for $h = 0.1$, with 10^{-3} for relative error in velocity and 10^{-2} for relative error in vorticity. This implies that the growth in error is of order one. Figure 7.9 shows the variation in the maximum relative error in vorticity at $t = 1$ for various nominal blob spacing h. The figure indeed shows that the change in error due to the spatial discretization is of order one.

Figure 7.10 compares the evolution of the maximum relative error for various overlap ratios, $\lambda = [0.5, 0.75, 1.0, 1.5]$. We see that the minimum error in velocity and vorticity is observed for the overlap ratio $\lambda = 1$. As we move from this value, an increase in the error is observable. In section 3.2.5, we determined that to reduce the Gaussian blurring of the vorticity field from the Gaussian vortex kernels, we require an overlap ratio $\lambda = 1$ and a small nominal blob spacing h. The parameter sensitivity analysis on the spatial discretization validates this observation and states that to ensure minimum error in coupling, these criteria has to be satisfied.

Figure 7.11a shows the effect of modifying the Lagrangian time step size Δt_L w.r.t a fixed Eulerian time step size. With $\Delta t_E = 0.001$ and the number of Eulerian sub-steps $k_E = [1, 2, 5, 10]$, the Lagrangian time steps become $\Delta t_L = k_E \cdot \Delta t_E = [0.001, 0.002, 0.005, 0.01]$. In Figure 7.11b, we instead vary the Eulerian time step $\Delta t_E = \Delta t_L / k_E = [0.001, 0.0005, 0.0002]$, with $\Delta t_L = 0.001$ and $k_E = [1, 2, 5]$. We see that the error in coupling reduces as the resolution of the time discretization matches with each other, when $k_E = 1$. We see that when we use a higher number of Eulerian sub-step, regardless of varying the Lagrangian or the Eulerian time step, there is an increase in the error in coupling.

7.1.3 Conclusion

In section 7.1.2.1, we observed that moving from uncoupled to one-way coupled case, increases the relative error in velocity. The growth in error was mainly due to the time integration error of the Lagrangian method. When moving from one-way coupled to fully coupled scheme, there is a tangible increase in the relative error in vorticity and an additional increase in the error in velocity. The increase in this error was due to the re-initialization of the vortex blobs introducing an additional smoothing error at each correction step.

In section 7.1.2.2, we observed that conservation of circulation is vital in ensuring an accurate coupling strategy. The transfer of vorticity from the Eulerian method to the Lagrangian method, must be performed with a focus on the conservation of circulation, to ensure that the artificial vorticity at the Eulerian Dirichlet boundary Σ_d is minimal.

In section 7.1.2.3, we investigated the impact of varying the spatial and temporal discretization on the accuracy of the coupling. We determined there is an increase in error, if the Lagrangian method is spatially under-resolved w.r.t to the Eulerian method. An overlap ratio of $\lambda = 1$ was shown to have the minimum error during the coupling, as it ensures minimum Gaussian blurring. Varying the number of Eulerian sub-steps k_E, showed that the linear interpolation for the Dirichlet boundary condition is potential source of improvement.

Ultimately, the errors observed in the results can be explained and are consistent with the theory. We determined that in order to ensure a minimum error during the hybrid coupling, we must have a high spatial resolution at the zone of vorticity transfer (i.e the interpolation region Ω_I), such that the discretization error of the two methods have minimum effect on the overall accuracy. Furthermore, we determined that conservation of circulation is vital to ensure correct transfer of solution between both methods and the algorithms described in section 5.1 are vital to ensure a proper Lagrangian correction step.

7.2 Clercx-Bruneau Dipole Convection

In section 7.1, we determined the source of the errors when coupling the Eulerian and the Lagrangian solution of ta Lamb-Oseen vortex diffusion case. However, an important aspect of domain decomposition method is the passage of a vortex core through a decomposed domain as it is a common phenomena in a VAWT simulation. As the blades pass through its own wake, they will encounter a shed vortex core from the previous blades.

Therefore, the purpose of the Clercx-Bruneau dipole convection test cases is to determine whether the present hybrid method is capable of accurately describe a passage of a vortex core through a given Eulerian domain.

7.2.1 Problem Definition

To simulate the travel of a vortex core, we used the analytical expression of the Clercx-Bruneau dipole [16] as the initial condition and simulated its convection through an Eulerian domain. The hybrid domain decomposition of this investigation is depicted in Figure 7.12. The Eulerian domain Ω_E is finite with bounds $[-0.25, 0.25] \times [-0.5, 0.5]$. The Clercx-Bruneau dipole, defined by equation 7.12, is initialized outside the Eulerian domain $\Omega_L \backslash \Omega_E$ at $(x_1, y_1) = (-1, 0.1)$ and $(x_2, y_2) = (-1, -0.1)$, corresponding to the positive and negative cores, respectively. As the simulation progresses, the dipole convects along the x-axis, passing through the Eulerian domain.

The Eulerian and the Lagrangian domain are discretized according to the parameters shown in table 7.2. The simulation was first benchmarked using a Finite Element (FE) only simulation, and a Vortex Particle Method (VPM) only simulation, providing an basis for hybrid simulation. To ensure that FE only simulation was valid, the Eulerian domain Ω_E spanned up to the far-field of the dipole, where the vorticity and the induced velocity are zero. The Eulerian domain Ω_E of the FE only simulation spanned $[-3, 3] \times [-2, 2]$.

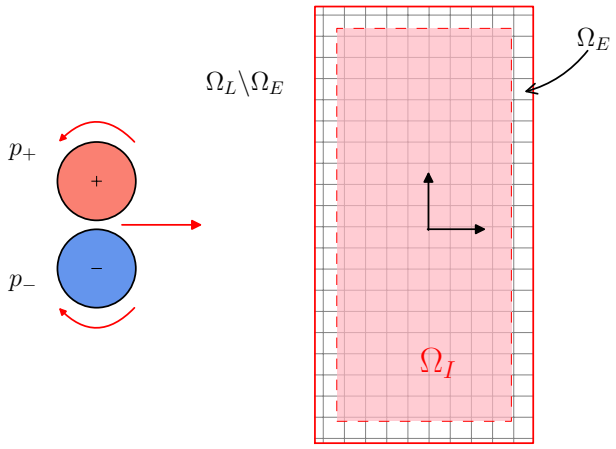


Figure 7.12: (*Not to Scale*) The domain decomposition for the Clercx-Bruneau convection problem, with the positive pole located at $p_+ = (x_1, y_1) = (-1, 0.1)$ and negative pole located at $p_- = (x_2, y_2) = (-1, -0.1)$. The parameters of the simulation are tabulated in table 7.2.

Table 7.2: Summary of the parameters for the Clercx-Bruneau dipole convection problem.

Parameters	Value	Unit	Description
Ω_E	$[-0.25, 0.25] \times [-0.5, 0.5]$	m	Eulerian domain bounds
Re	625	-	Reynolds number
U	1	$\mathrm{m\,s^{-1}}$	Characteristic velocity
W	1	m	Characteristic Length
ν	1.6×10^{-3}	$\mathrm{kg\,s^{-1}\,m^{-1}}$	Kinematic viscosity
$(x, y)_{1,2}$	$(-1, \pm 0.1)$	m	Initial location of the monopoles
ω_e	299.528385375226[a]	-	Characteristic vorticity of the monopole
λ	1	-	Overlap ratio
h	0.005	m	Nominal blob spacing
h_{grid}	≈ 0.007	m	FE cell diameter
N_{cells}	40000	-	Number of mesh cells
Δt_L	2.5×10^{-4}	s	Lagrangian time step size
Δt_E	2.5×10^{-5}	s	Eulerian time step size
k_E	10	-	Eulerian sub-steps
$N_{\text{t-steps}}$	2800	-	Number of time integration steps
t	0 to 0.7	-	Simulation time
d_{bdry}	$2 \cdot h$	m	Interpolation boundary offset

[a] Obtained from Renac et al. [53]

7.2.2 Results

To determine whether the hybrid method was capable of accurate describe the passage of the Clercx-Bruneau dipole through the Eulerian domain, we first compared the plot of the vorticity field of the hybrid method with FE only simulation. Figure 7.13 compares the vorticity field of the FE simulation and the hybrid simulation at $t = [0, 0.2, 0.4, 0.6]$.

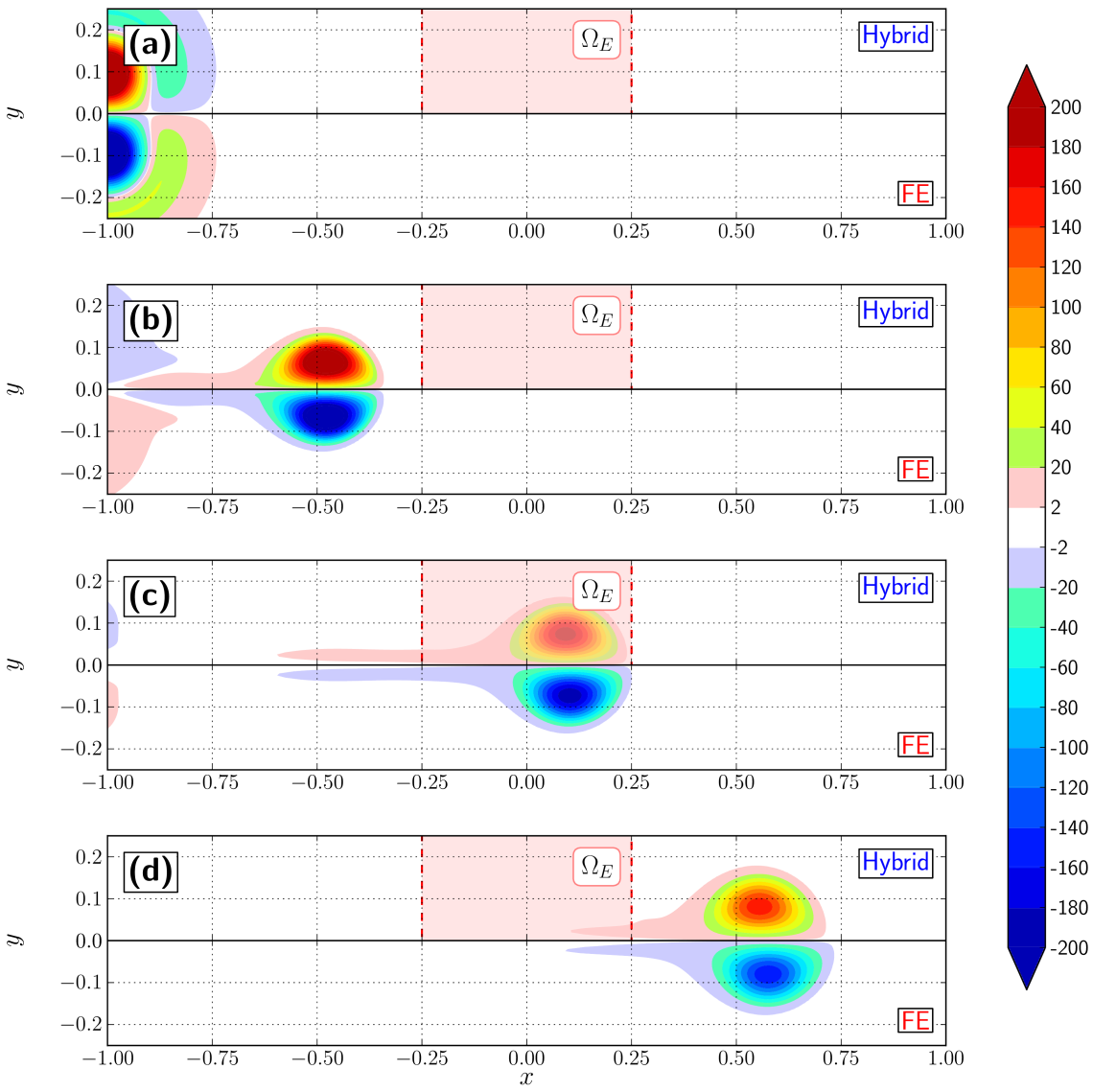
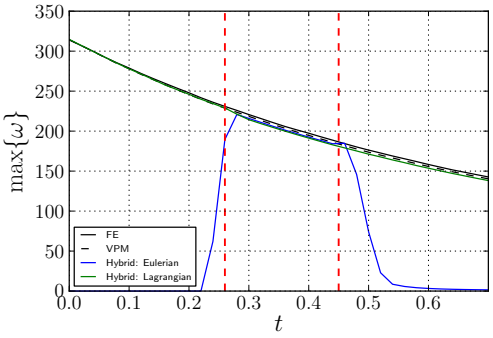


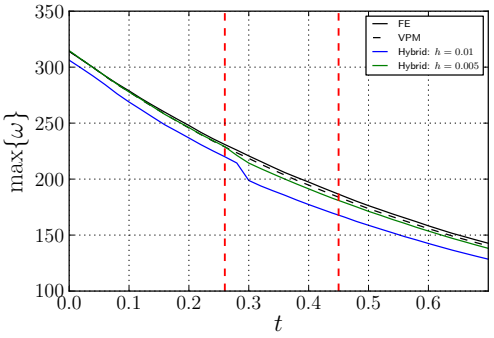
Figure 7.13: Plot of the Clercx-Bruneau dipole at $t = [0, 0.2, 0.4, 0.7]$ using parameters tabulated in table 7.2. The figure compares the hybrid simulation (top halves) against the FE only simulation (bottom halves).

The top half of each subplot belongs to the hybrid simulation, whereas the bottom half to the FE only simulation. The cores of the dipole enters the Eulerian domain at $t = 0.26$ and exits the domain at $t = 0.45$. We see that the plot of the vorticity field matches till $t = 0.4$. At $t = 0.7$ however, the vorticity distribution in the hybrid method is lagging w.r.t the FE only simulation.

To determine the cause of this phenomenon, we investigated the variation in the maximum vorticity ω_{max}. Figure 7.14a shows the evolution of the maximum vorticity ω_{max} from $t = 0$ to $t = 0.7$ in the Eulerian and the Lagrangian sub-domain of the hybrid simulation. We compared these results with the benchmark FE only and the VPM only simulations. At $t = 0.26$, the maximum vorticity in the Eulerian sub-domain starts to increase, signifying the entering of the vortex core. Similarly, at $t = 0.45$, the maximum vorticity starts to



(a) Comparison of the hybrid sub-domains



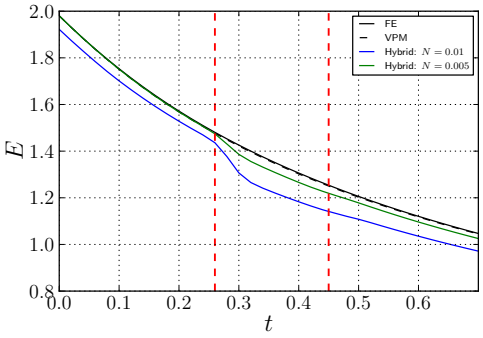
(b) Comparison with a less resolved hybrid method

Figure 7.14: Evolution of the maximum vorticity $\max\{\omega\}$ from $t = 0$ to $t = 0.7$. The solutions are compared with the benchmark results of FE only [—, solid black], and VPM only [- -, dashed black] simulations. The figure depicts (**a**) the maximum vorticity in the Eulerian and the Lagrangian sub-domain of the hybrid method, and (**b**) the maximum vorticity of hybrid method with nominal blob spacing $h = 0.01$ and $h = 0.005$.

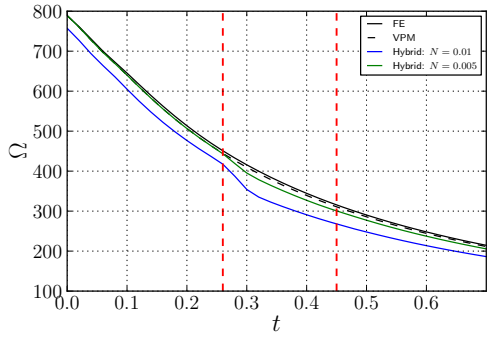
decrease, signifying the exiting of the vortex core. As the dipole enters the sub-domain, there is a drop in the maximum vorticity. A possible explanation of this phenomena is the interaction of the artificial vorticity at the boundary Σ_d with the solution. To confirm that the influence of the artificial vorticity scales with the resolution of the simulation (as determined in section 7.1), we performed a simulation with a lower resolution.

Figure 7.14b compares the evolution of maximum vorticity ω_{max} for nominal blob spacing $h = 0.005$ and a lower resolution of $h = 0.01$. We determine that the less resolved simulation indeed shows a larger drop in the maximum vorticity. This shows that the presence of the artificial vorticity indeed has a direct effect on the passage of the dipole through the boundary.

The evolution of the kinetic energy E and the enstrophy Ω shows the similar behavior, Figures 7.15a and Figure 7.15b, respectively. The figures shows that with increasing spatial resolution, the hybrid simulation is able to simulate similar variation in flow parameters



(a) Kinetic energy E



(b) Enstrophy Ω

Figure 7.15: Evolution of the (**a**) kinetic energy E and (**b**) enstrophy for the nominal blob spacing $h = 0.01$ and $h = 0.005$.

as the convection FE only simulation.

7.2.3 Discussion of results

In conclusion, the results of the Clercx-Bruneau dipole convection simulation showed that:

- As the resolution of the hybrid method decreases at the overlap region, the effects of the artificial vorticity becomes greater.
- A larger artificial vorticity at the boundary Σ_d, causes a greater lagging effect on the convection of the dipole.
- A larger artificial vorticity at the boundary Σ_d, causes a larger reduction in the maximum vorticity $\omega_{\max}$, the kinetic energy E and the enstrophy Ω at the entry of the dipole.

These results imply that for a less resolved Lagrangian method, the large artificial vorticity near the boundary has a significant diffusive effect on the vortex cores. The increased diffusion of the vortex cores causes the lagging phenomena and the larger drop in the maximum vorticity and the kinetic Energy. Therefore to accurate simulate the passage of a vortex core through the Eulerian domain, we need to minimize the influence of the artificial vorticity at the boundary. As the artificial vorticity is a representation of the difference in the Eulerian and the Lagrangian solutions at the outer Eulerian boundary, we need to ensure that they have a matching discretization near this region.

The results showed that with a matching spatial discretization, the hybrid method is indeed capable of simulating a passage of a dipole through the Eulerian domain. However, the limitation of the present study is that we have not investigated if these finding are similar for a different flow conditions such as for:

- a higher Reynolds number,
- a stronger vortex core,
- a large dipole convection velocity,
- a single monopole.

Therefore to have a better understanding, it is recommended that we perform further investigation related to this test case. These additional investigations should provide a stronger validation on whether the hybrid method is capable of simulation the passage of a vortex core. Though, in the context of the present study, as we are currently only concerned with laminar flow simulation with weak vortex cores, the observations we made for the present test case should be valid.

7.3 Clercx-Bruneau Dipole Collision

In this section, we study the Clercx-Bruneau dipole colliding with a solid wall. The main goal this investigation is it determine if the hybrid method is able to accurately deal with a wall bounded problem. A Finite Element (FE) only investigation was first performed in section 4.5.2 and was validated against the study of Clercx and Bruneau [16]. We determined that the FE only simulation was able to provide results matching with the literature. Therefore, the simulation will be used as a benchmark for the present investigation.

7.3.1 Problem Definition

The setup of the hybrid domain is as shown in Figure 7.16. The Eulerian sub-domain Ω_E resolves the near-wall region, and the Lagrangian sub-domain domain resolves the complete fluid domain and is bounded by the no-slip wall Σ_w (shown in blue). The Eulerian domain Ω_E extends from the wall Σ_w to the boundary Σ_d, where the velocity boundary condition from the Lagrangian method is prescribed. The parameters of the simulation are tabulated in table 7.3.

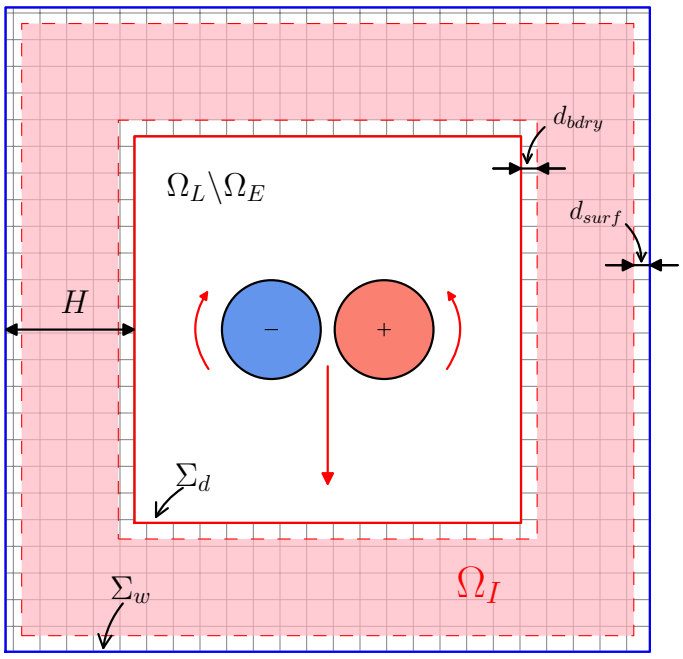


Figure 7.16: [*Not to Scale*] The domain decomposition for the Clercx-Bruneau dipole collision problem, with the positive pole at $p_+ = (x_1, y_1) = (0.1, 0)$ and negative pole at $p_- = (x_2, y_2) = (-0.1, 0)$. The parameters of the simulation are tabulated in table 7.3.

As we are dealing with the wall-bounded problem, we require the vortex panel method to enforce the boundary condition in the Lagrangian method. In section 2.3.2, we described the decomposition of the Lagrangian domain Ω_L into the vortex blob domain Ω_b and the vortex panel domain Ω_p.

Table 7.3: Summary of the parameters for the Clercx-Bruneau dipole collision.

Parameters	Value	Unit	Description
Ω	$[-1,1]^2$	m	Extend of Eulerian domain from wall Σ_d
H	0.2	m	Eulerian domain width
Re	625	-	Reynolds number
U	1	$\mathrm{m\,s^{-1}}$	Characteristic velocity
W	1	m	Characteristic Length
ν	1.6×10^{-3}	$\mathrm{kg\,s^{-1}\,m^{-1}}$	Kinematic viscosity
$(x,y)_{1,2}$	$(\pm 0.1, 0)$	m	Initial location of the dipole
ω_e	299.528385375226[a]	-	Characteristic vorticity of the monopole
λ	1	-	Overlap ratio
h	0.003	m	Nominal blob spacing
N_{panels}	400	-	Number of panels
h_{grid}	0.005 to 0.01	m	FE cell diameter
N_{cells}	58272	-	Number of mesh cells
Δt_L	2.5×10^{-4}	s	Lagrangian time step size
Δt_E	2.5×10^{-5}	s	Eulerian time step size
k_E	10	-	Eulerian sub-steps
$N_{\mathrm{t-steps}}$	4000	-	Number of time integration steps
t	0 to 1	-	Simulation time
d_{bdry}	$2 \cdot h$	m	Interpolation domain offset from Σ_d
d_{surf}	$3 \cdot h$	m	Interpolation domain offset from Σ_w

[a] Obtained from Renac et al. [53]

7.3.2 Results

The first step of the investigation was to compare the vorticity plots with the FE only simulation. Figure 7.17 shows the state of the dipole at $t = [0, 0.2, 0.4, 0.6, 0.8, 1]$. The figure compares the hybrid simulation (left half) with the FE only simulation (right half). Onces the dipole enters the Eulerian domain, at $t = 0.4$, we observe that difference in the solution stars to exist. The vorticity contours of $-10 \leqslant \omega \leqslant -1$ (gray contours) and $1 \leqslant \omega \leqslant 10$ (pink contours) shows that the near the core of the dipole there is a difference between the FE only simulation and the hybrid simulation. At $t = 0.6$, we see there exists less vorticity in between $-1 \leqslant 1$ (white regions). The possible explanation of this behavior is the existence of the artificial vorticity. These artificial vorticity, which has a maximum absolute magnitude less than 5, has an effect on the vorticity profile of the same magnitude.

Figure 7.18 compares the vorticity contour at $t = 1$ against the FE only simulation. We see there is a slight difference in the vorticity contour lines of hybrid solution, figure 7.18b. The shape of the contour lines near the wall is slightly different, and furthermore, the location of the core is also shifted.

To investigate further on the cause of this difference, we studied the change in maximum vorticity ω_{max}, the kinetic energy E, the enstrophy Ω, and the palinstrophy P, shown in Figure 7.19. The variation in maximum vorticity, Figure 7.19a, shows that the first peak

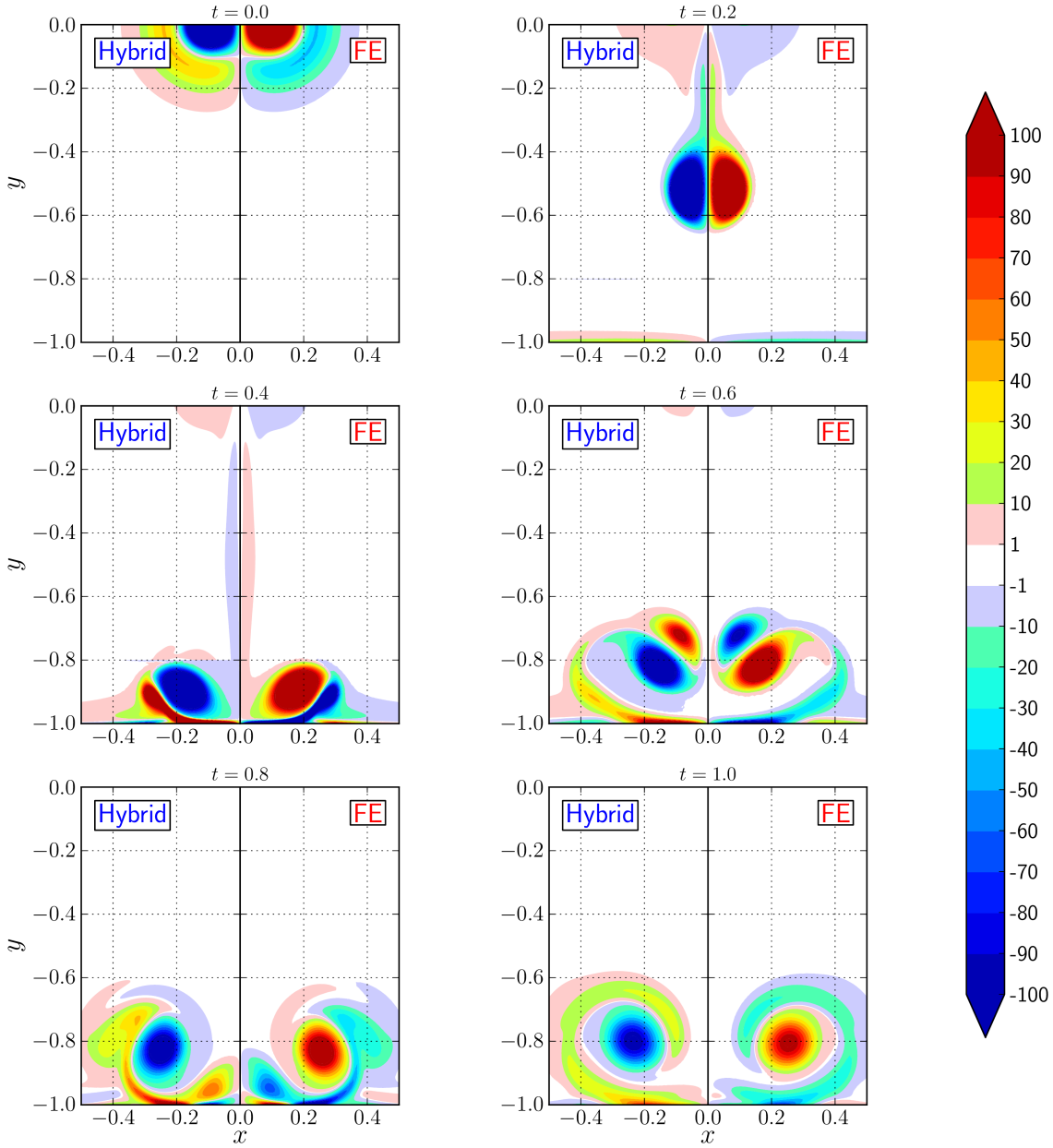
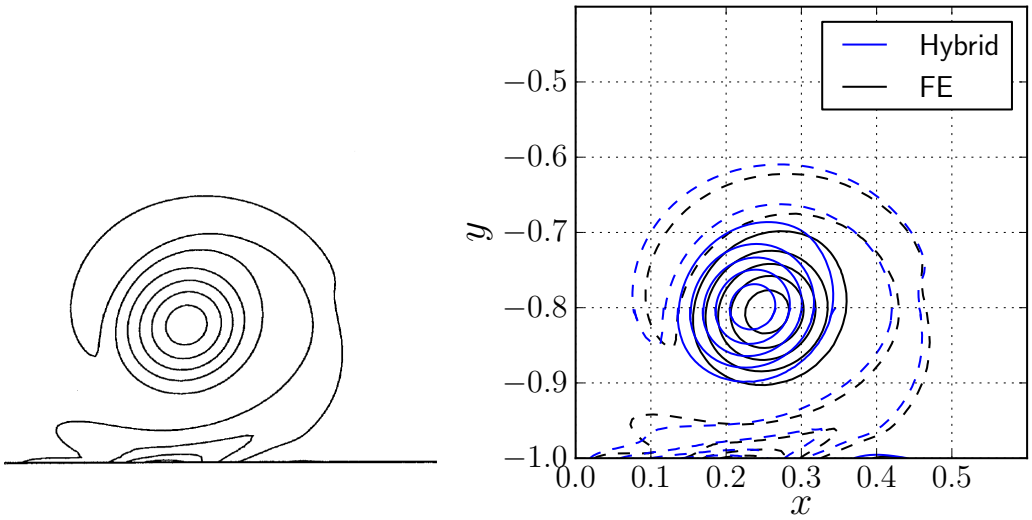
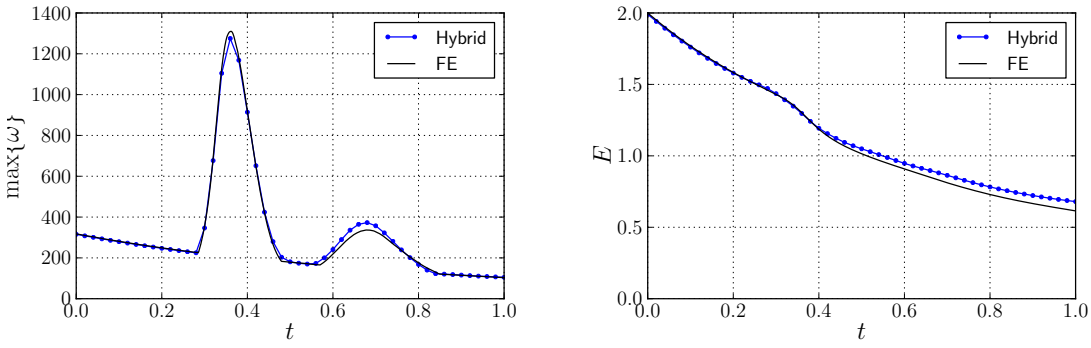


Figure 7.17: Plot of the dipole at $t = [0, 0.2, 0.4, 0.6, 0.8, 1]$, comparing the hybrid simulation (left half) and FE only simulation (right half).



(a) Literature **(b)** Present study

Figure 7.18: Comparison of the vorticity contours at $t = 1$. The figure compares the plot obtained by **(a)** literature, Clercx and Bruneau [16], and **(b)** the present study, the hybrid and FE only simulation.



(a) Maximum vorticity $\max\{\omega\}$ **(b)** Kinetic Energy $E(t)$

(c) Enstrophy $\Omega(t)$ **(d)** Palinstrophy $P(t)$

Figure 7.19: Variation in the fluid parameters from $t = 0$ to $t = 1$. The figure compares the hybrid results [—, solid blue] with the FE only [—, solid black] results.

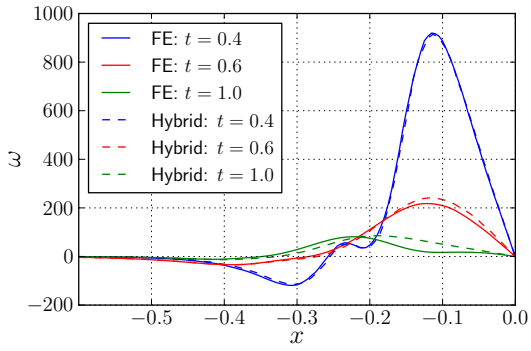


Figure 7.20: Compares the vorticity generated at the bottom-left wall ($y = -1$, $-0.6 \leqslant x \leqslant 0$) at $t = 0.4$ [—, blue], $t = 0.6$ [—, red] and $t = 1$ [—, green].

in the hybrid is the same as FE only simulation, at $t \approx 0.35$. However, the second peak in vorticity, at $t \approx 0.65$ is higher than the standard simulation. Between $t = 0.4$ and $t = 0.6$, the dipole exits and renters the Eulerian domain, as seen in Figure 7.17.

Figure 7.19b shows that, as the dipole leaves the Eulerian domain Ω_E from $t = 0.4$, the kinetic energy E is higher. At $t = 1$, the kinetic energy of hybrid is approximately 10% higher than FE only simulation. This was unexpected, as Figure 7.17 shows the vorticity contours are in good agreement with each other.

Investigating the enstrophy Ω , Figure 7.19c, and the palinstrophy P, we see that the difference is simulation is observable from only $t = 0.6$. Therefore, the variation in the simulation only occurs from the second collision of the dipole, when dipole re-enters the Eulerian domain. Therefore the cause the deviation must be the result of dipole entering the Eulerian domain.

Due to the increased kinetic energy E, we see that there is slight difference in the distribution of the vorticity field along $y = -1$, Figure 7.20. The figure shows when the kinetic energy of the simulations match, at $t = 0.4$, the distribution of the vorticity is the same. However as time progresses, the difference in the shape of the distribution becomes apparent. At $t = 1$, we see that the vorticity peak is wider and is due to the increased generation of vorticity from the wall to repel the dipole with higher kinetic energy E.

We investigated further with a higher resolved Lagrangian method, with a smaller nominal blob spacing h, a larger number of panels N_{panels}, and a smaller Lagrangian time step size Δt_L. However, these simulation did not provide significant improvement to the present results and was inconclusive.

7.3.3 Discussion of Results

In conclusion, the results of the dipole collision simulation showed that:

- The hybrid method is capable of dealing with the no-slip boundary.
- The results of the simulation are in good agreement upto $t = 0.5$.
- After $t = 0.5$, when the dipole re-enters, the hybrid simulation starts to deviate due to the presence of artificial vorticity.

- At $t = 1$, due to the higher kinetic energy E in flow, the location of the vortex core at $t = 1$ is different.

The present investigation verified and validated that the algorithm for determining the vortex panel strength described in section 5.1. We saw that the Eulerian method provided the necessary vorticity distribution into the fluid domain to oppose the oncoming dipole and the circulation of the vortex panels was able to be determined directly from the Eulerian domain.

The present results were obtained with a boundary offset of $d_{bdry} = 2 \cdot h$ as used by Stock [61]. However, the presence of the artificial vorticity had an impact on the collision of the dipole. During further investigation with impulsively started cylinder, in section 7.4, we determined that using such a small offset results in the amplification of the effect of the artificial vorticity, resulting in a larger mismatch in the solution. This provides a possible explanation of the surprising mismatch in the kinetic energy and the final distribution of the vorticity field at $t = 1$.

The limitation of the present test case is that it was not able to provide a full understanding of the collision of the dipole with the wall boundary. Therefore, for a more detailed understanding, a further investigation in regards to the varying d_{bdry} should be performed in future.

However, as the main purpose of this test case was to determine whether the hybrid method was capable of dealing with the wall boundary, the present test case provided the necessary confirmation. Furthermore the study gave us an understanding of the potential inaccuracies when dealing with wall bounded problems.

7.4 Impulsively Started Cylinder at $Re = 550$

An important aspect of VAWT simulation is the accurate calculation of the lift and the drag forces. This is important as they provided the necessary understanding of the performance of the VAWT.

Therefore in this section, we will study the flow around impulsively started cylinder (ISC) at $Re = 550$. Extensive study have been performed on this regard, such as by Koumoutsakos and Leonard [41], and Rosenfeld et al. [54]. With the help of these literatures and the FE only simulation performed in section 4.5.3, we can validate that the hybrid method is capable of accurately calculate the forces acting on the body.

7.4.1 Problem Definition

The description of the impulsively started test case was initially introduced in section 4.5.3. For the hybrid simulation, we performed similar investigation and compared the results with the benchmark study and the aforementioned literature. The parameters of the simulation are tabulated in table 7.4.

Figure 7.21 shows the domain decomposition of the hybrid simulation. The Lagrangian domain Ω_L resolves the full fluid domain, with an Eulerian domain Ω_E resolving the

Table 7.4: Summary of the parameters of the hybrid simulation for the impulsively started cylinder test case at $Re = 550$.

Parameters	Value	Unit	Description
Re	550	-	Reynolds number
$\mathbf{u}_\infty$	$[1, 0]$	$\mathrm{m\,s^{-1}}$	Freestream velocity
R	1	m	Radius to Eulerian boundary Σ_{wall}
R_{ext}	1.5	m	Radius to Eulerian boundary Σ_d
ν	3.6×10^{-3}	$\mathrm{kg\,s^{-1}\,m^{-1}}$	Kinematic viscosity
λ	1	-	Overlap ratio
h	0.008	m	Nominal blob spacing
h_{grid}	0.008 to 0.04	m	FE cell diameter
N_{cells}	32138	-	Number of mesh cells
N_{panels}	100	-	Number of panels
Δt_L	0.005	s	Lagrangian time step size
Δt_E	0.001	s	Eulerian time step size
k_E	5	-	Number of Eulerian sub-steps
$N_{\text{t-steps}}$	40000	-	Number of time integration steps
t	0 to 40	-	Simulation time
d_{bdry}	$0.1 \cdot R$	m	Interpolation domain offset from boundary Σ_d
d_{surf}	$3 \cdot h$	m	Interpolation domain offset from boundary Σ_{wall}

near-wall region of the cylinder. The interpolation domain Ω_I is defined by the inner boundary Σ_i and the outer boundary Σ_o. The inner boundary Σ_i, is defined with an offset $d_{surf} = 3 \cdot h$ from the cylinder wall Σ_w and the outer boundary Σ_o of the interpolation region Ω_I is defined with a larger offset $d_{bdry} = 0.1 \cdot R$ from the boundary Σ_d. We observed in the previous test cases that the main error of the hybrid scheme is the artificial vorticity generated at the Σ_d and therefore to minimize its influence, we have chosen a larger offset

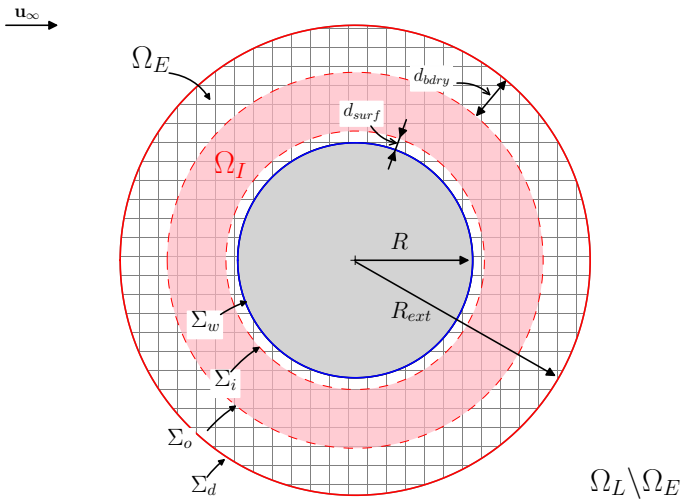


Figure 7.21: (*Not to Scale*) The domain decomposition for the Impulsively started cylinder. The parameters of the domain are tabulated in table 7.4.

than used by Stock [61].

7.4.2 Results

We investigated two aspects of the impulsively started cylinder problem. The first study focused on the impact of coupling parameters on the accuracy of the lift and the drag forces. The parameters of interest are the number of vortex panels N_{panels}, nominal blob spacing h, and time step size of the Lagrangian method Δt_L. The second study focused on the long run performance of the forces acting of the cylinder. Artificial perturbation was induced, as described in section 4.5.3, to induced vortex shedding at a smaller simulation time t.

Figure 7.22 shows the vorticity contour at $t = [1, 3, 5, 7]$. The plot compares the hybrid simulation (top halves) with the FE only simulation (bottom halves). The hybrid halves of the plot depicts the Eulerian sub-domain Ω_E in pink. After studying the figure, we observer that the vorticity contours of the hybrid simulation matches with the FE only simulation. The only difference between both simulation is the artificial vorticity at the Dirichlet boundary Σ_d. However, we must note that the magnitude of the artificial vorticity is in the range $|\omega| \leqslant 0.2$, and with the maximum vorticity in the domain $\max\{\omega\} = 32$, we have that the strength of the artificial vorticity is less than 1% of the maximum vorticity in the fluid. Therefore, we have ensured that the Lagrangian method and the Eulerian method has similar discretization at the region of overlap. In section 7.1, we determined that this is required to ensure an accurate coupling scheme.

Figure 7.23 shows the evolution of the drag coefficient C_d, the friction drag $C_{d_{fric}}$, and the pressure drag $C_{d_{pres}}$. The results are compared with the FE only simulation and the reference data obtained from Koumoutsakos and Leonard [41]. Observing the figure, we see that the hybrid simulation has a larger drag coefficient, due to the increased pressure drag $C_{d_{pres}}$.

Varying the Lagrangian time step Δt_L, Figure 7.24a, the nominal blob spacing h, Figure 7.24b, and the number of vortex panels, Figure 7.24c, we see that the accuracy of the simulation increases with increasing spatial and temporal resolution. The parameter sensitivity analysis showed that the simulation with $h = 0.00625$ showed the largest improvement.

However, during the end of the present study it became apparent that the parameters that has the largest impact is the interpolation domain boundary offset d_{bdry}. Figure 7.25 shows the impact of increasing the d_{bdry} of the interpolation domain. We see that with $d_{bdry} = 0.2R$, there is much larger agreement with the benchmark results. However, determined the appropriate d_{bdry} for a given flow condition becomes a challenging topic. As further increase of d_{bdry} means that less of the Eulerian solution is transferred to the Lagrangian domain. This could also cause a detrimental effect as the two method now becomes less coupled. Therefore, we propose a future research on determining the appropriate d_{bdry} such that the incorrect Eulerian solution at the boundary does not pose a concern to the coupling.

The second focus of the impulsively started cylinder is the long run evolution of the lift and the drag of the cylinder, from $t = 0$ to $t = 40$. We performed similar comparison as

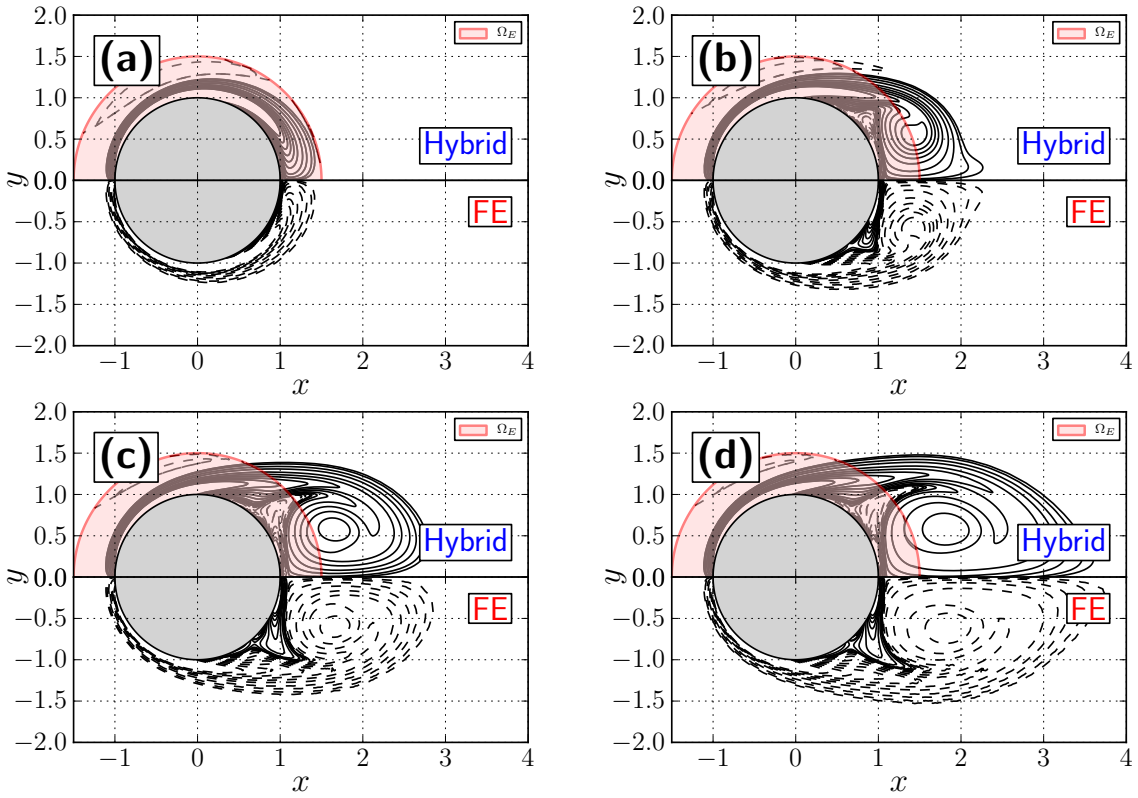


Figure 7.22: Comparison of the vorticity contours for **(a)** $t = 1$, **(b)** $t = 3$, **(c)** $t = 5$ and **(d)** $t = 7$ with contour levels $[-7, ..., -3, -2, -1, 0.5, -0.2, -0.1, 0.1, 0.2, 0.5, 1, 2, 3, ..., 7]$. The figures compares the hybrid simulation (top half) with FE only simulation (bottom half).

done in section 4.5.3. An artificial perturbation was induced according to Leocointe and Piquet [44]. Figure 7.26 compares the evolution of the lift coefficient C_l, and the drag coefficient C_d of hybrid simulation, with the FE only simulation, and the reference data from Rosenfeld et al. [54].

Investigating the evolution of drag shows that the hybrid simulation has higher drag. After $t = 5$, there is slight mismatch in the oscillation of the drag. However observing the amplitude fluctuation, we see that the simulation tend to fluctuate around $C_d = 1.4$. Observing the evolution of lift shows that the hybrid simulation has a larger initial amplitude. Furthermore, there exist a negative phase shift in the amplitude. However, at time progress, $t > 20$, we see that the frequency and the amplitude of the oscillation is similar to the reference data. A through investigation of the oscillation requires a longer simulation where the amplitude of the oscillation would become fixed. However, due to the lack of computational resources, a longer simulation than $t = 40$ with the current simulation parameters was not feasible.

Figure 7.27 compares the vorticity field of the hybrid simulation (left), and the FE only simulation (right) at time instances $t = [10, 20, 30, 40]$. We see that the initial vorticity distribution are in good agreement. As time progress, the simulations deviate away from each other and is simply due to the difference in the spatial discretization.

The difference in the spatial discretization becomes apparent in Figure 7.28, The figure

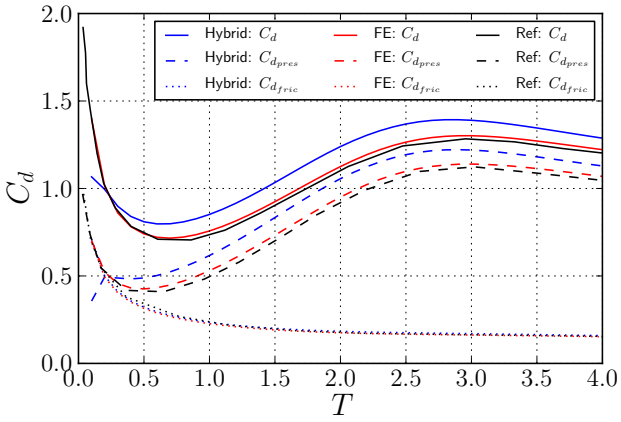
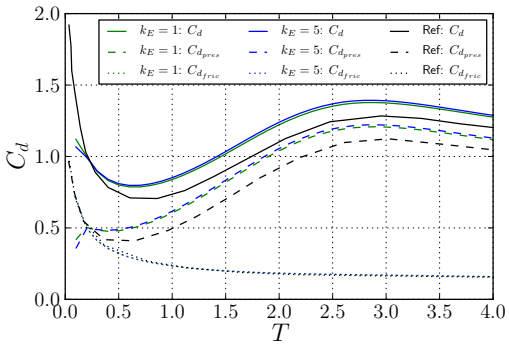
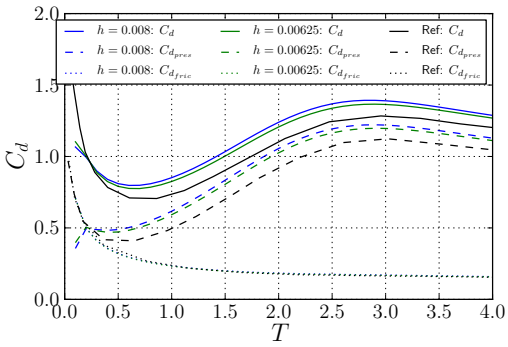


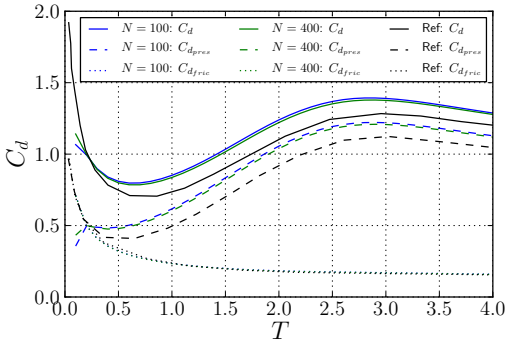
Figure 7.23: Evolution of the drag coefficient during the initial stages $t = 0$ to $t = 4$ with total drag coefficient C_d (solid), pressure drag coefficient $C_{d_{pres}}$ (dashed) and friction drag coefficient $C_{d_{fric}}$ (dotted). The figure compares results of hybrid simulation (**blue**), FE only simulation (**red**) and reference data (**black**) of Koumoutsakos and Leonard [41]



(a) Variation in Lagrangian time step size Δt_L: $k_E = 1$ with $\Delta t_L = \Delta t_E$ and $k_E = 5$ with $\Delta t_L = 5 \cdot \Delta t_E$



(b) Variation in nominal blob spacing h: $h = 0.008$ and $h = 0.005$



(c) Variation in number of panels N_{panels}: $N = 100$ and $N = 400$

Figure 7.24: Parameters sensitivity analysis on the drag evolution of the cylinder from $t = 0$ to $t = 4$, compared with literature data (**black**) obtained from Koumoutsakos and Leonard [41]

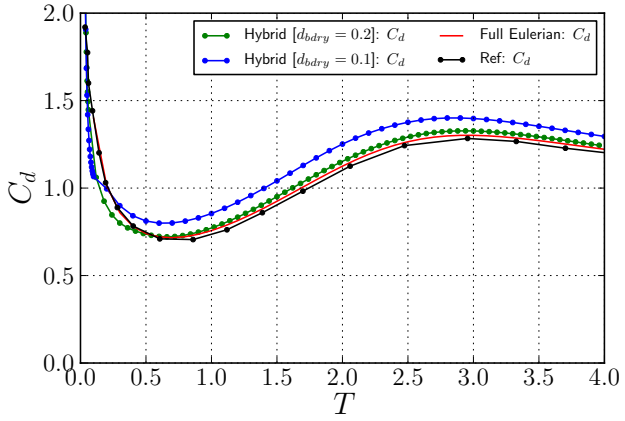


Figure 7.25: Variation in $d_{bdry} = [0.1R, 0.2R]$. Compared with benchmark FE only simulation and literature, Koumoutsakos and Leonard [41]

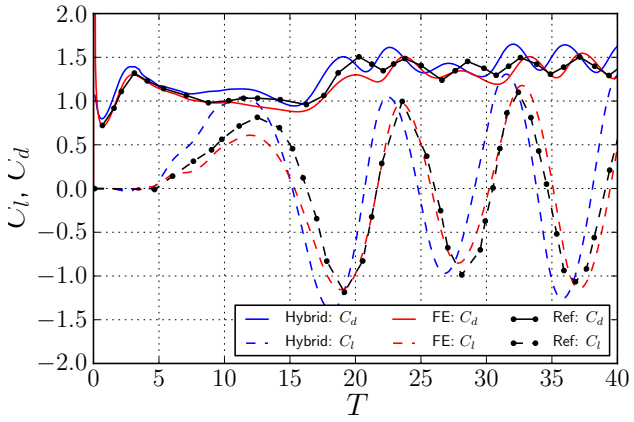


Figure 7.26: Evolution of the lift and drag coefficient from $t = 0$ to $t = 40$ with artificial perturbation [44]. The figure compares hybrid (**blue**), FE only (**red**), and the reference data (**black**) from Rosenfeld et al. [54].

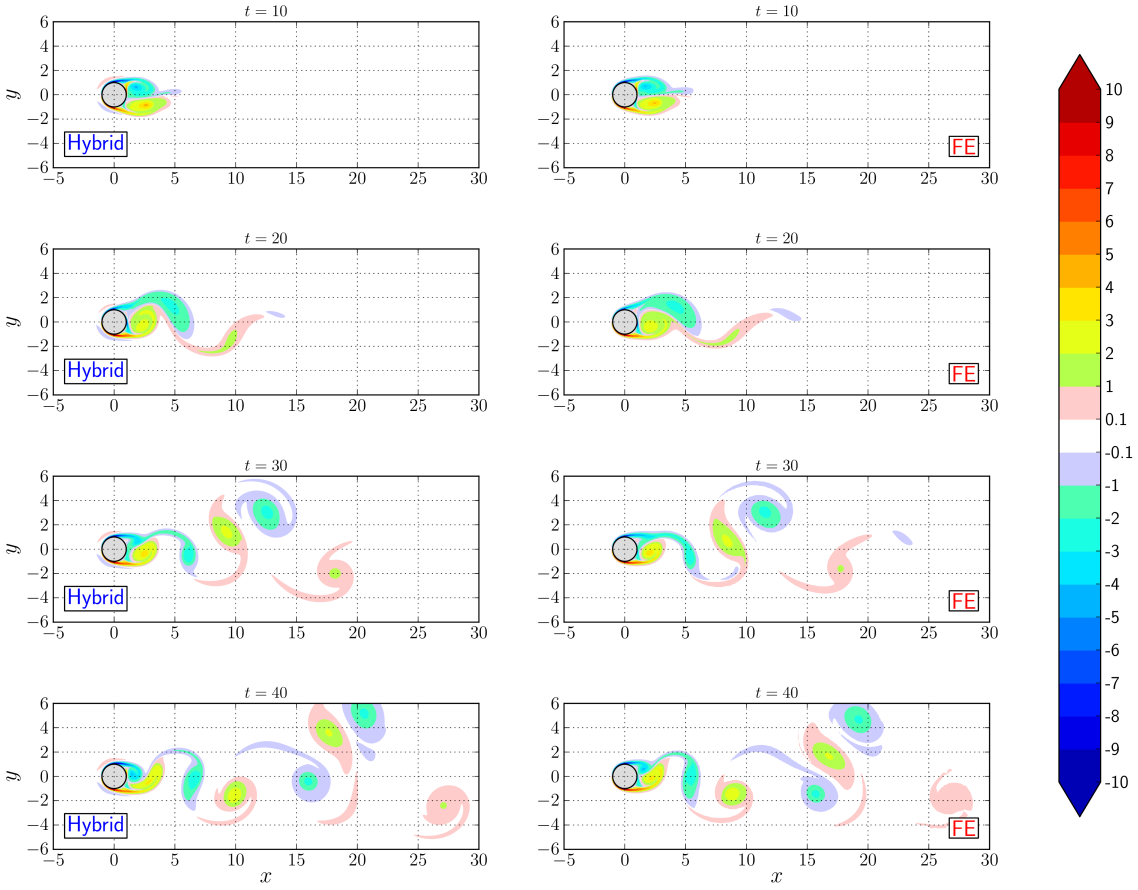
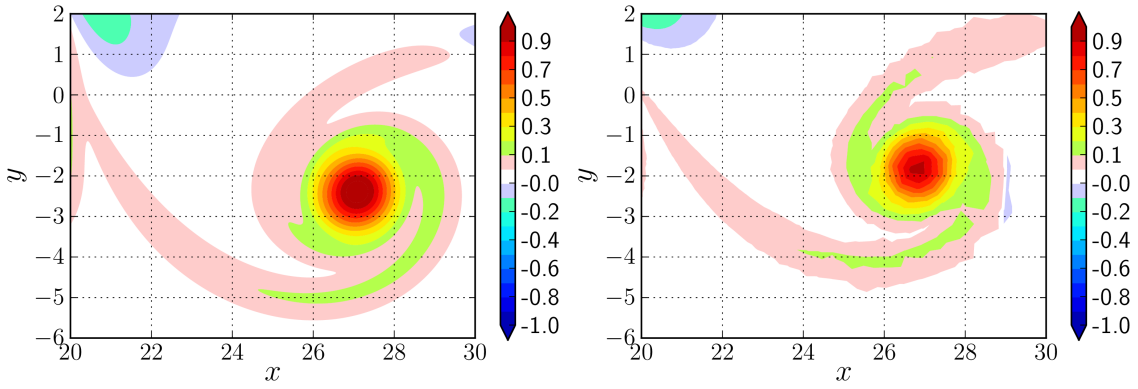


Figure 7.27: Plot of the vorticity field at $t = [10, 20, 30, 40]$, comparing the hybrid simulation (left) with the FE only simulation (right).



(a) Hybrid **(b)** FE only

Figure 7.28: Plot of the first shed vortex at $t = 40$, comparing (**a**) the hybrid simulation, and (**b**) the FE only simulation.

shows the state of the first shed vortex at $t = 40$, for the hybrid simulation and the FE only simulation. For an efficient FE only simulation, we had to under-resolve the mesh, away from the cylinder. However, with the auto-adaptive nature of the Lagrangian method, the hybrid simulation had a well defined vortex core at $D = 13$. Furthermore, we see that the dipole sustains is strength, unlike the FE only simulation with is more diffusive, because it is under-resolved.

7.4.3 Discussion of Results

In conclusion, the results of the impulsively started cylinder test cases showed that:

- When the hybrid method is under-resolved, the cylinder experience a higher drag.
- Increasing the boundary offset d_{bdry} from $0.1R$ to $0.2R$ improved the simulation results. However, an increased offset also means less coupled results as more Eulerian solution is ignored. This can have detrimental effect for high Reynolds number flows.

As the present study was limited with computational resource and time, a detailed investigation of the d_{bdry} could not be performed. A research in future in this regard could help us determine the ideal d_{bdry} for a given flow condition.

Due to the limited computation resources and time, the long run simulation was also performed for a shorter period. However, the results showed that even with less resolved method, the hybrid method is capable of observing similar oscillations as the well-resolved FE only simulation. In addition to this, when observing the starting vortex at $t = 40$, it became apparent that the hybrid method provides a well-resolved solution even at $D = 13$. This highlights the advantage of the auto-adaptive nature of the hybrid method and the ease of resolves an unsteady vortex core evolution.

If the hybrid method can provide additional accuracy without sacrificing its efficiency, for example with $d_{bdry} = 0.2R$, the hybrid method has the potential to substantially outperform the convectional grid methods for unsteady flow problems of a VAWT.

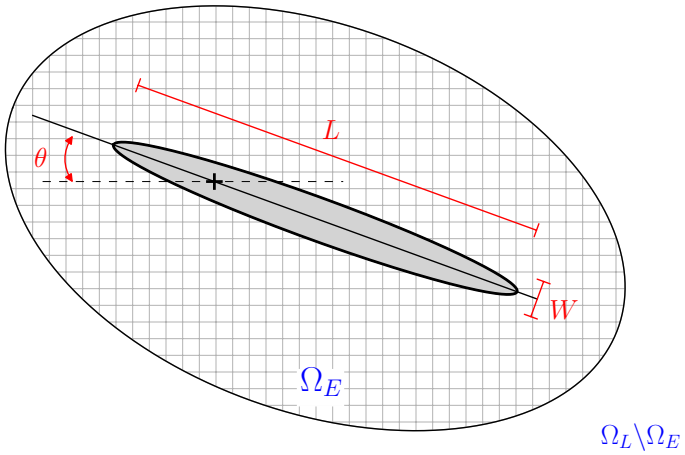


Figure 7.29: (*Not to scale*) The domain decomposition for the stalled elliptical airfoil. The parameters of the domain are tabulated in table 7.5.

7.5 Stalled Elliptic Airfoil at $Re = 5000$

The purpose of investigating the stalled elliptical airfoil at $Re = 5000$ is to demonstrate the feasibility of using the hybrid method to simulate a complex problem, such as a flow around a thin stalled airfoil. The FE only simulation is as a benchmark for the current study.

7.5.1 Problem Definition

The study of the stalled elliptical airfoil at $RE = 5000$, is performed similarly to the study of the impulsively started cylinder at $Re = 550$, see section 7.4. The main difference now is the mesh geometry (now representing a thin stalled airfoil), and a higher Reynolds number.

Figure 7.29 depicts the configuration of the hybrid simulation. We studied the flow around a thin airfoil with thickness to chord ratio t/c of 12%, having a chord length $L = 1$, and a maximum thickness $W = 0.12$. As we have a elliptical airfoil, the maximum thickness is located at $0.5L$. Furthermore, the airfoil is pitched to 20° about center of rotation $0.25L$, to create a stalled flow. The parameters of the simulation are summarized in table 7.5.

7.5.2 Results

The results shows the hybrid method is capable of simulating the flow past a lifting body. Figure 7.30 shows the evolution of the lift and the drag coefficient. We see that the lift and the drag coefficient are in good agreement upto $t = 2$. However, as time progress we see the results of the simulation deviate from each other. A possible explanation is the higher Reynolds number. When we deal with higher Reynolds number, we see that coupling procedure is more sensitive.

Figure 7.31 shows the plot of vorticity field at $t = [1, 2, 3, ..., 10]$. We see that thanks to the adaptive nature of the scheme, the shed vortices are well-defined. Furthermore, in

Table 7.5: Summary of the parameters of the hybrid simulation for the stalled elliptical airfoil test case.

Parameters	Value	Unit	Description
Re	5000	-	Reynolds number
$\mathbf{u}_\infty$	$[1, 0]$	$\mathrm{m\,s^{-1}}$	Freestream velocity
L	1	m	Chord length
W	0.12	m	Maximum thickness
ν	2×10^{-4}	$\mathrm{kg\,s^{-1}\,m^{-1}}$	Kinematic viscosity
λ	1	-	Overlap ratio
h	1.67×10^{-3}	m	Nominal blob spacing
h_{grid}	0.0014 to 0.008	m	FE cell diameter
N_{cells}	118196	-	Number of mesh cells
N_{panels}	400	-	Number of panels
Δt_L	1×10^{-4}	s	Lagrangian time step size
Δt_E	1×10^{-3}	s	Eulerian time step size
k_E	10	-	Number of Eulerian sub-steps
$N_{\text{t-steps}}$	10000	-	Number of time integration steps
t	0 to 10	-	Simulation time
d_{bdry}	$0.1 \cdot L$	m	Interpolation domain offset from boundary Σ_d
d_{surf}	$3 \cdot h$	m	Interpolation domain offset from boundary Σ_{wall}

the near-wall region, the Eulerian region is able to capture the complex stalled structure vorticity field behind the airfoil.

7.5.3 Discussion of results

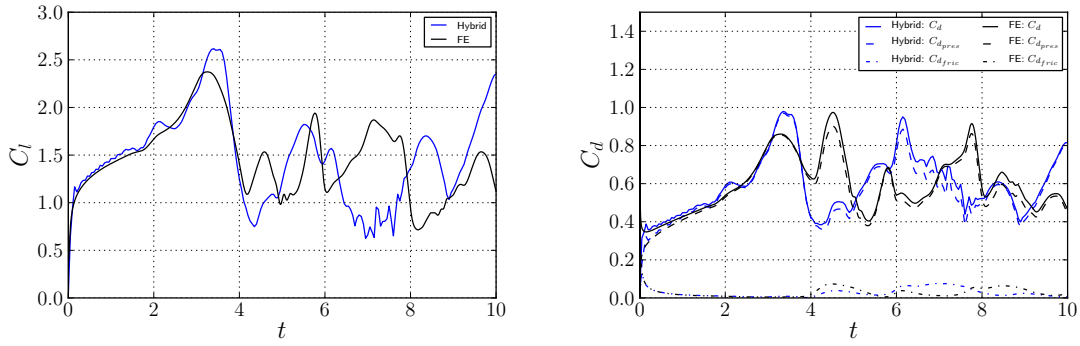
The results of the stalled elliptical airfoil at $Re = 5000$ showed that:

- The hybrid method is able to simulate the unsteady flow separation.
- A higher Reynolds number flow problem is still a challenge and accuracy has only achieved upto $t = 2$.

The simulation showed that the present hybrid method is capable of simulating the flow past an airfoil that undergoes a unsteady flow separation. This is a key aspect of the VAWT simulation and through this test case we have demonstrated the capability of simulating such problems.

However, we were only able to simulate up to $Re = 5000$ as we did not implemented any turbulence scheme. For Reynolds number regimes of the VAWTs, turbulence models such as URANS and LES are fundamental requirement for efficient and accurate computation of the flow.

Furthermore, an interesting extension of the present test case is to investigate the flow around a presently used VAWT airfoil such as a NACA0018 airfoil. This could enable us to perform a direct comparison with the competing numerical method used in simulating



(a) Hybrid **(b)** FE only

Figure 7.30: Forces

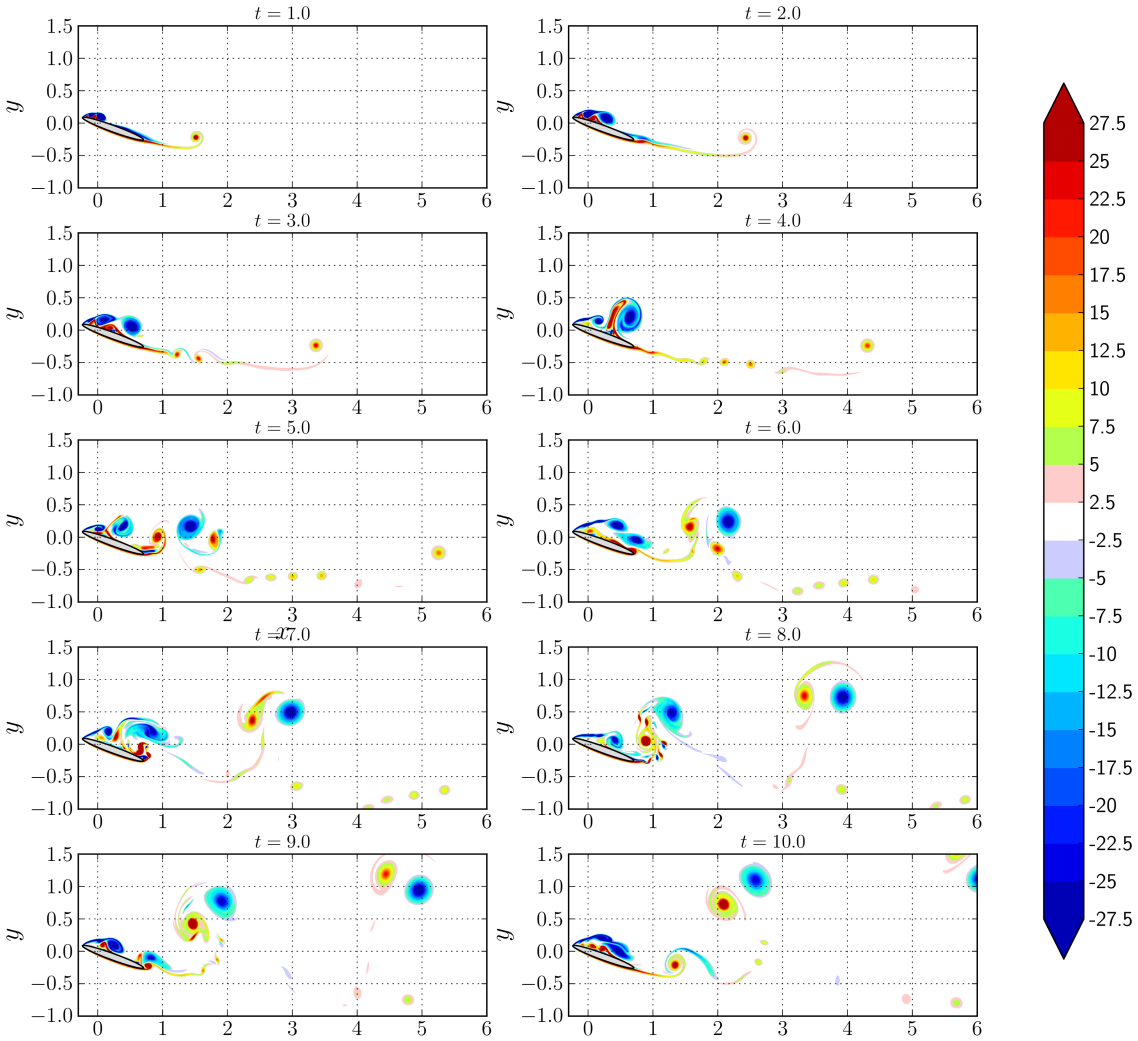


Figure 7.31: Plot of the vorticity contours at $t = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$.

the VAWT. However, a fundamental challenge of a full VAWT simulation is the multi-body moving problem. In section 7.6, we were able to shows that the present scheme is capable of dealing with multiple bodies. However, the implementation of the moving body require future research and can be achieved by the implementation of Arbitrary Lagrangian-Eulerian ALE scheme.

7.6 Multi-body problem

The hybrid method is ideal at simulating multi-body problems. As the Eulerian domains of each body is separated, each body can easily be modified without affecting each other. To demonstrate this, we simulated the flow around two impulsively started cylinders with segregated Eulerian domains.

7.6.1 Problem Definition

The parameters of the simulation are shown in table 7.4. The two cylinder configuration is represented by two Eulerian sub-domains, Ω_{E_A} and Ω_{E_B}, belonging to cylinder A and cylinder B respectively. The configuration of the two cylinder is depicted in Figure 7.32. The locations of the cylinder A and B are at $(x_A, y_A) = (-R_{ext}, 0)$ and $(x_B, y_B) = (R_{ext}, R)$, respectively. These positions ensure that the Eulerian sub-domains do not overlap, and the Lagrangian method is used to communicate between theses sub-domains.

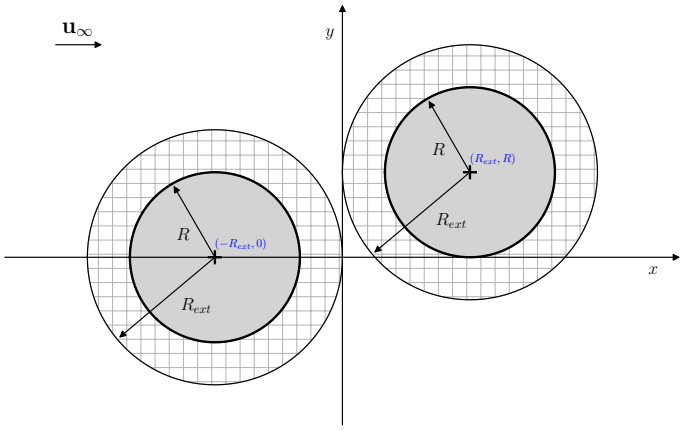


Figure 7.32: (*Not to Scale*) The hybrid domain decomposition configuration of two impulsively started cylinder test case.

7.6.2 Results

The simulation of multi-body problem was ran upto $t = 22$, reaching the limit of the computational resource. Figure 7.33 shows the plot of the vorticity at various time instances, at $t = [1, 4, 7, 10, 13, 16, 19, 22]$. The plot shows the transfer of vorticity from cylinder A to cylinder B with the help of the Lagrangian method. The Lagrangian method serves as the global communication tool for transferring the information between various Eulerian sub-domains.

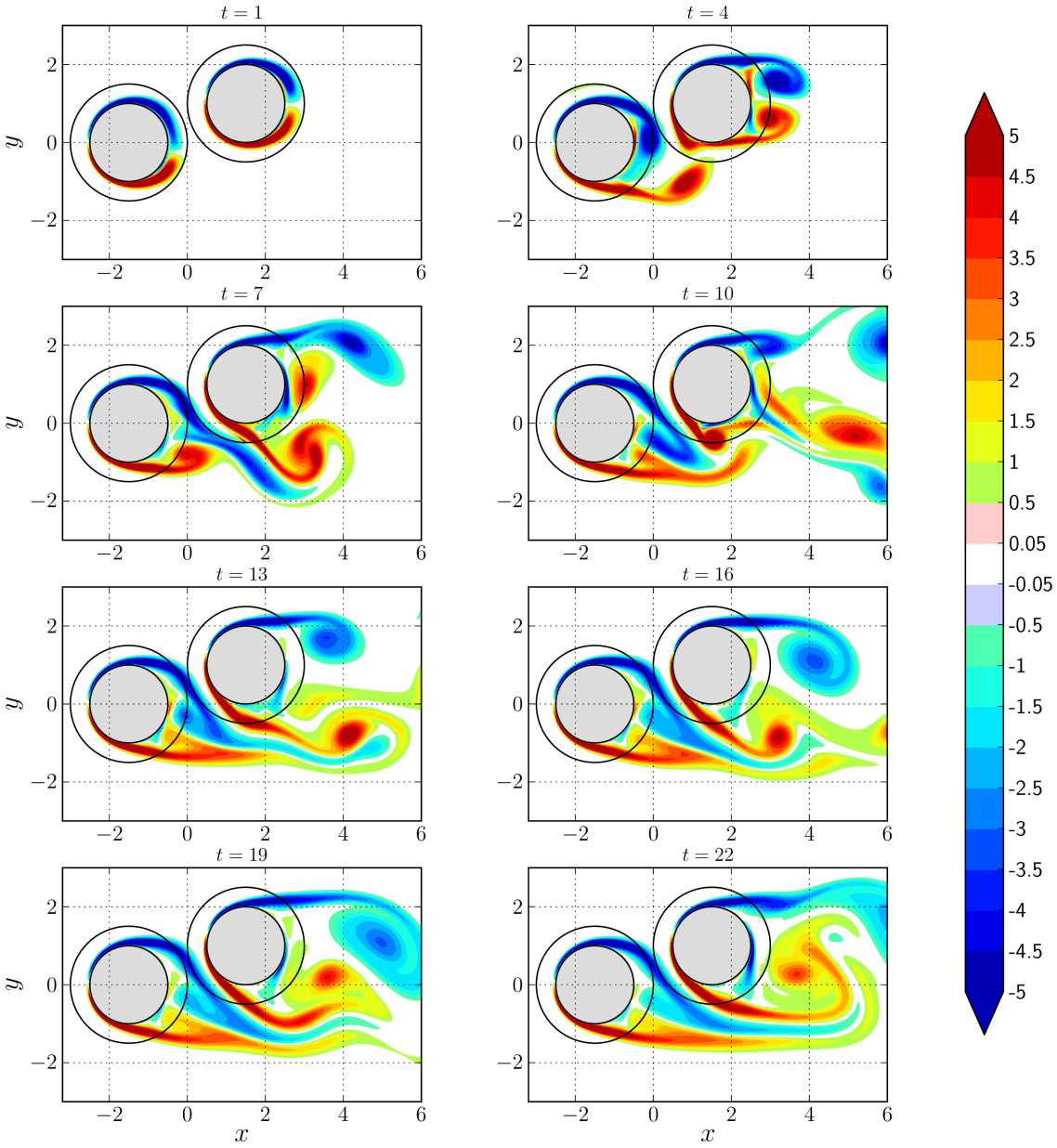


Figure 7.33: Plot of the vorticity contours at $t = [1, 4, 7, 10, 13, 16, 19, 22]$ at $Re = 550$.

7.6.3 Discussion of results

The simulation shows the feasibility of simulating a multi-body problem. The advantage of the hybrid method for such problems is that, in future when the moving body problem is implemented, the two bodies can move independently. In a convectional grid method, where the two methods have to share a single Eulerian domain, the determination of efficient mesh becomes a non-trival aspect.

In conclusion, we see that the hybrid method is capable of simulating a multi-body VAWT problem with greater ease than the convectional grid based method.

Chapter 8

Conclusion and Recommendation

The goal of the present research was to develop an efficient, reliable, and accurate numerical method for modeling the flow past a 2D Verticle Axis Wind Turbine (VAWT). The numerical method should enable us to have better understanding of the performance of the VAWT. The challenge in modeling the flow past a VAWT is the motion of the blades where it passes through its own wake creating flow phenomena such as flow separation, dynamic stall and other blade-wake interactions. The resulting wake geometry of VAWT becomes difficult to model and makes it hard to predict the performance of the VAWT. Therefore, the numerical method should be able to accurately simulate the near-body flow phenomena and should be efficient at evolving the wake.

The numerical method that satisfied these requirements was a Hybrid Eulerian-Lagrangian Vortex Particle Method (HELVPM) which couples a Finite Element Method (Eulerian method) that resolves the near-body region, and the Vortex Particle Method (Lagrangian Method) that resolves the wake. The advantage of an Eulerian method was that it was efficient at accurately describing the generation of vorticity from the body. This generated vorticity was then transfered to the Lagrangian method where it was efficiently evolved using Fast Multipole Method (FMM) and parallel computation in Graphical Processing Units (GPU). The hybrid method that was employed was based on the doctoral thesis of Daeninck [24] and the additional study on the Lagrangian correction algorithm by Stock [61]. Though additional modification where required to the coupling strategy to ensure a valid numerical method that conserves the total circulation in the fluid.

8.1 Conclusion

The present study implemented the strategy and performed test case simulations to ensure an accurate implementation (Chapter 7). During the verification and the validation of this hybrid method, we were able to derive some key conclusions:

- The coupling strategy employed by Daeninck and Stock required several modifications to ensure conservation of circulation (Chapter 5):

  - A standard particle initialization of local particle volume and local vorticity causes a diffusive effect for the vorticity field in the Lagrangian domain. This was due to the smoothing error of the Gaussian kernels for vortex particle. The accurate initialization of the particles is still an open question in vortex method, but for the present study we were able to minimize the error by adjusting the resolution of the spatial discretization.
  - Solving the boundary condition for the Lagrangian method required an additional constraint on the integral strength of the vortex panels, due to the singular nature of the vortex panel problem. The additional constraint on the strength was determined directly from the Eulerian method further ensuring conservation of circulation.
  - An error in the interpolation of vorticity from the Eulerian method onto the Lagrangian method introduces a an error in the total circulation of the Lagrangian method. Therefore, the correction of the particles within the interpolation region was performed with a focus on conservation of circulation.

- During the evolution of the hybrid method, if an error exists in the coupling, it is manifested by a generation of artificial vorticity from the external boundary of the Eulerian domain:

  - The transfer of the solution from Lagrangian to Eulerian method introduces the initial error. This error is small as the coupling is performed with the velocity which has errors an order lower than vorticity.
  - The transfer of the solution from Eulerian to Lagrangian produces a larger error as now we deal with the vorticity. As vorticity is the curl of velocity, the error in the velocity is amplified. The strength of the artificial vorticity due to this error will be therefore larger.
  - A mismatch in the circulation of the two method has an effect on the accuracy of the coupling. A mismatch produces artificial vorticity at the orders of magnitude similar to the vorticity in the flow. Therefore, ensuring that the circulation is conserved was paramount to a valid hybrid method.

- The hybrid method demonstrated that it is able to predict the evolution of the lift and drag forces according to theory. The error in the force coefficient is proportional to the resolution of the hybrid method in the overlap region where coupling takes place:

  - When employing an under-resolved hybrid method, the drag coefficient in the hybrid method is over-predicted. However as we increase the discretization within the interpolation region, we observe a convergence of the error.
  - For an under-resolved hybrid method, we observed a premature trigger in the unsteady behavior of the vorticity field. This occurs due to small generation of artificial vorticity in the under-resolved method but converges with increasing resolution at the overlap region. The simulations demonstrated the need for matching vorticity field from both methods at overlap region.
  - In the final stages of the present study, it was determined that the offset of the interpolation region from the outer Eulerian boundary d_{bdry} plays a pivot

role in the accuracy of the forces calculated. It was determined that increasing this offset can substantially increase the accuracy of the simulation. However, a smaller region of interpolation could have detrimental effect on the accuracy of the coupling. At high Reynolds number flows, the ideal parameters for the offset becomes a non-trivial question. Therefore it is recommended a dedicated research on the ideal method to deal with the other solution of the Eulerian method is performed in future.

- The present study concluded by demonstrating the feasibility for simulating the flow of a 2D VAWT through proof-of-concept test cases:
  - The simulation of the stalled elliptical airfoil at $Re = 5000$ showed that: a) the hybrid method can be extended to higher Reynolds number problem cases, b) the hybrid method is able to simulate the flow past a lifting body, and c) the hybrid method is able to simulate the stalled flow past a pitched airfoil. The limitation on the computational resource with the lack of turbulence modeling meant that the present study could only simulate for a short simulation time and a limited Reynolds number.
  - The simulation of the flow past two cylinders demonstrated that the present numerical method can easily be extended to a multi-body VAWT problem.

8.2 Recommendations

- **Vortex blob initialization**: An accurate initialization of the particles is still an open question. The standard approach of initializing the vortex blobs introduces additional error in coupling and if negated could improve the scheme substantially. A possible approach for overcoming this *blurring* of the vorticity field could be the investigation of Barba and Rossi [1].
- **Determine the relation of particle resolution to the flow conditions**: For the present study, the ideal particle resolution (overlap ratio λ and nominal blob spacing h) was determined by analyzing the final solution of the simulation (such as lift and drag). However, we recommend a thorough study on determine the relation for the particle size to the flow condition such as Reynolds number or the maximum vorticity in the fluid, in order for an accurate simulation.
- **Modify the outer Eulerian domain**: During the research, we observed that the any mismatch between the Eulerian and the Lagrangian method introduces an artifact near the outer Eulerian boundary, in the form of artificial vorticity. To deal with this artificial vorticity, we reduced the region of interpolation, ignoring the outer Eulerian boundary. However, as we now have a smaller interpolation region, this results in a weaker coupling of the methods. Therefore, an ideal approach to deal with the incorrect Eulerian solution is to directly modify the outer Eulerian domain such this artificial vorticity is negated.
- **Spatially varying vortex core sizes**: A vortex particle method with spatially varying vortex blob core size can substantially improve the computational efficiency

of the hybrid method. At the region of overlap, we could use vortex blobs with small core size ensuring minimum coupling error. As we move away from the body, the blob cores size can be scaled up with the size of shed vorticity.

- **Higher order time marching schemes for the Eulerian method**: A higher order time marching scheme for the Eulerian method can ensure a more accurate evolution of the Eulerian solution, thereby further increasing the accuracy in coupling. A higher order time marching scheme such as a 4^{th} order Runge-Kutta method could help us reduce the number of Eulerian substeps k_E.

- **Spectral decomposition of the kernel in the boundary integral equation**: Instead of the Kelvin's theorem for solving the no-slip boundary condition, Koumoutsakos [40] instead investigated the spectral decomposition of the kernel in the Fredholm equation. He demonstrated that we can then obtain a vortex panel problem that is well-conditioned, even when the number of panels is increased or the thickness of the body is decreased. Therefore, this approach will be advantages when dealing with larger problem set.

- **Multipole and GPU accelerated boundary element method**: The greatest computational limitation of the hybrid solver was the calculation of the vortex panel induced velocity on the vortex particles. When dealing with millions of particles, the direct calculation of this induced velocity substantially increases the computational time and resources. Simulation acceleration techniques such as Fast Multipole method or a GPU-accelerated boundary element method can help us tackle this challenge.

- **Turbulence modeling**: The present study could only research in the realms of laminar flow. However, as the flow around a VAWT is inherently turbulent, an implementation is turbulence modeling is needed. Turbulence model such as Unsteady Reynolds Averaged Navier-Stokes (URANS) or Large Eddy simulation (LES) could provide a potential solution to this problem.

- **Moving body**: The implementation of the moving geometry will enable us to investigate the dynamic stall behavior of the VAWT. The implementation can easily be extended by introducing the ALE formulation to the Eulerian method. Furthermore, this could open the possible for investigating fluid-structure interactions and introduce possibility for studying deforming VAWT blades.

- **Simulation of 3D geometries**: Research have shown that the 3D wake dynamics of a VAWT is fundamental in understanding the performance of the VAWT. Therefore, an extension of the present numerical method to 3D is recommended. The coupling approach described in the present study could used to couple a 3D Eulerian method with a 3D Lagrangian method.

Afterword

-summary, reflection, collaboration. -lessons learned -recommendations -insight into developing hybrid methods -thoughts, opinions

-issues dealth, how it was overcome

- things that would have been useful to know before tackling hybrid methoid

-letter

References

[1] Barba, L. A., and Rossi, L. F. Global field interpolation for particle methods. *Journal of Computational Physics 229*, 4 (2010), 1292–1310.

[2] Barba, L. A. L. Vortex method for computing high-Reynolds number flows: Increased accuracy with a fully mesh-less formulation.

[3] Beale, J. On the Accuracy of Vortex Methods at Large Times. In *Computational Fluid Dynamics and Reacting Gas Flows SE - 2*, B. Engquist, A. Majda, and M. Luskin, Eds., vol. 12 of *The IMA Volumes in Mathematics and Its Applications*. Springer New York, 1988, pp. 19–32.

[4] Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., and Smith, K. Cython: The best of both worlds. *Computing in Science and Engineering 13*, 2 (2011), 31–39.

[5] Boffi, D. Three-Dimensional Finite Element Methods for the Stokes Problem. *SIAM Journal on Numerical Analysis 34*, 2 (Apr. 1997), 664–670.

[6] Braza, M., Chassaing, P., and Ha Minh, H. Numerical Study and Physical Analysis of the Pressure and Velocity Fields in the Near Wake of a Circular Cylinder. *Journal of Fluid Mechanics 165* (1986), 79–130.

[7] Brenner, S. C., and Scott, L. R. *The Mathematical Theory of Finite Element Methods*. Texts in applied mathematics. Springer-Verlag, 2002.

[8] Brezzi, F., and Fortin, M. *Mixed and hybrid finite elements methods*. Springer series in computational mathematics. Springer-Verlag, 1991.

[9] Carey, G. F. *Computational Grids: Generations, Adaptation & Solution Strategies*. CRC Press, 1997.

[10] Cgal, Computational Geometry Algorithms Library. http://www.cgal.org.

[11] Chang, C. C., and Chern, R. L. Numerical study of flow around an impulsively started circular cylinder by a deterministic vortex method. *Journal of Fluid Mechanics 233* (1991), 243–263.

[12] Chen, Z. *The Finite Element Method: Its Fundamentals and Applications in Engineering*. World Scientific, 2011.

[13] Chorin, A. J. Numerical solution of the Navier-Stokes equations. *Mathematics of computation 22*, 104 (1968), 745–762.

[14] Chorin, A. J. Numerical Study of Slightly Viscous Flow. *Journal of Fluid Mechanics 57*, Part 4 (1973), 785–796.

[15] Ciarlet, P. G., and Raviart, P. A. General Lagrange and Hermite interpolation in $\mathbf{R}^n$ with applications to finite element methods. *Archive for Rational Mechanics and Analysis 46*, 3 (1972), 177–199.

[16] Clercx, H., and Bruneau, C.-H. The normal and oblique collision of a dipole with a no-slip boundary. *Computers & Fluids 35*, 3 (Mar. 2006), 245–279.

[17] Collette, A. *Python and HDF5*. O'Reilly Media, 2013.

[18] Commons, W. Darrieus windmill, 2007.

[19] Commons, W. Windmills d1-d4 (thornton bank), 2008.

[20] Cooper, C. D., and Barba, L. A. Panel-free boundary conditions for viscous vortex methods. In *19th AIAA Computational Fluid Dynamics Conference* (Dept. of Mechanical Engineering, Universidad Técnica F. Santa María, Casilla 110-V, Valparaíso, Chile, 2009).

[21] Cottet, G.-H., Koumoutsakos, P., and Salihi, M. L. O. Vortex Methods with Spatially Varying Cores. *Journal of Computational Physics 162*, 1 (July 2000), 164–185.

[22] Cottet, G. H., and Koumoutsakos, P. D. *Vortex Methods: Theory and Practice*, vol. 12. Cambridge University Press, 2000.

[23] Courant, R. Variational methods for the solution of problems of equilibrium and vibrations. *Bull. Amer. Math. Soc 49*, 1 (1943), 1–23.

[24] Daeninck, G. *Developments in Hybrid Approaches: Vortex Method with Known Separation Location; Vortex Method with Near-Wall Eulerian Solver; RANS-LES Coupling*. PhD thesis, Université Catholique de Louvain, Belgium, 2006.

[25] Dalcín, L., Paz, R., Storti, M., and D'Elía, J. MPI for Python: Performance improvements and MPI-2 extensions. *Journal of Parallel and Distributed Computing 68*, 5 (2008), 655–662.

[26] Degond, P., and Mas-Gallic, S. The Weighted Particle Method for Convection-Diffusion Equations. Part 1: The Case of an Isotropic Viscosity. *Mathematics of Computation 53*, 188 (Oct. 1989), 485–507.

[27] Dixon, K., Simão Ferreira, C. J., Hofemann, C., Van Bussel, G., and Van Kuik, G. A 3D unsteady panel method for vertical axis wind turbines. In *European Wind Energy Conference and Exhibition 2008* (2008), vol. 6, pp. 2981–2990.

[28] Franklin, W. R. Pnpoly-point inclusion in polygon test. http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html, 2006.

[29] Geuzaine, C., and Remacle, J.-F. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering 79*, 11 (2009), 1309–1331.

[30] Goda, K. A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows. *Journal of Computational Physics 30*, 1 (1979), 76–95.

[31] Guermond, J. L., and Lu, H. A domain decomposition method for simulating advection dominated, external incompressible viscous flows. *Computers & Fluids 29*, 5 (June 2000), 525–546.

[32] Guermond, J. L., Minev, P., and Shen, J. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering 195*, 44-47 (2006), 6011–6045.

[33] Guermond, J. L., and Shen, J. A new class of truly consistent splitting schemes for incompressible flows. *Journal of Computational Physics 192*, 1 (2003), 262–276.

[34] Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering 9*, 3 (2007), 90–95.

[35] Johnston, H., and Liu, J.-G. Accurate, stable and efficient Navier-Stokes solvers based on explicit treatment of the pressure term. *Journal of Computational Physics 199*, 1 (2004), 221–259.

[36] Jones, E., Oliphant, T., Peterson, P., et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2014-07-16].

[37] Katz, J., and Plotkin, A. *Low-speed aerodynamics*. Cambridge Aerospace Series. Cambridge University Press, 2001.

[38] Kloeckne, A. PyUblas: Hybrid numerical codes in Python and C++ integrate numpy and Boost.Ublas, 2008–. [Online; accessed 2014-07-16].

[39] Koumoutsakos, P. *Direct numerical simulations of unsteady separated flows using vortex methods*. PhD thesis, California Institute of Technology, USA, 1993.

[40] Koumoutsakos, P., and Leonard, A. Improved boundary integral method for inviscid boundary condition applications. *AIAA journal 31*, 2 (1993), 401–404.

[41] Koumoutsakos, P., and Leonard, A. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *Journal of Fluid Mechanics 296* (1995), 1–38.

[42] Koumoutsakos, P., Leonard, A., and Pépin, F. Boundary Conditions for Viscous Vortex Methods. *Journal of Computational Physics 113*, 1 (July 1994), 52–61.

[43] Lamb, H. *Hydrodynamics*. Cambridge university press, 1993.

[44] Lecointe, Y., and Piquet, J. On the use of several compact methods for the study of unsteady incompressible viscous flow round a circular cylinder. *Computers and Fluids 12*, 4 (1984), 255–280.

[45] Logg, A. NSBench in Launchpad. https://launchpad.net/nsbench, 2014.

[46] Logg, A., Mardal, K.-A., Wells, G. N., et al. *Automated Solution of Differential Equations by the Finite Element Method*, vol. 84. Springer, 2012.

[47] Logg, A., and Wells, G. N. DOLFIN: Automated finite element computing. *ACM Transactions on Mathematical Software 37*, 2 (2010).

[48] Monaghan, J. Extrapolating B-splines for interpolation. *Journal of Computational Physics 60*, 2 (Sept. 1985), 253–262.

[49] Mortensen, M. Fenicstools. https://github.com/mikaem/fenicstools, 2014.

[50] Nair, M. T., and Sengupta, T. K. Unsteady flow past elliptic cylinders. *Journal of Fluids and Structures 11*, 6 (1997), 555–595.

[51] Ould-Salihi, M. L., Cottet, G. H., and El Hamraoui, M. Blending Finite-Difference and Vortex Methods for Incompressible Flow Computations. *SIAM Journal on Scientific Computing 22*, 5 (Jan. 2001), 1655–1674.

[52] RAO, S. S. *The Finite Element Method in Engineering*. Elsevier Science, 2011.

[53] Renac, F., Gérald, S., Marmignon, C., and Coquel, F. Fast time implicit-explicit discontinuous Galerkin method for the compressible NavierStokes equations. *Journal of Computational Physics 251* (Oct. 2013), 272–291.

[54] Rosenfeld, M., Kwak, D., and Vinokur, M. A fractional step solution method for the unsteady incompressible Navier-Stokes equations in generalized coordinate systems. *Journal of Computational Physics 137* (1991), 102–137.

[55] Shankar, S., and Dommelen, L. A New Diffusion Procedure for Vortex Methods. *Journal of Computational Physics 127*, 1 (Aug. 1996), 88–109.

[56] Shiels, D. *Simulation of controlled bluff body flow with a viscous vortex method*. PhD thesis, California Institute of Technology, USA, 1998.

[57] Simão Ferreira, C. J. *The near wake of the VAWT: 2D and 3D views of the VAWT aerodynamics*. PhD thesis, Delft University of Technology, Netherlands, 2009.

[58] Simão Ferreira, C. J., Bijl, H., van Bussel, G., and van Kuik, G. Simulating Dynamic Stall in a 2D VAWT: Modeling strategy, verification and validation with Particle Image Velocimetry data. *Journal of Physics: Conference Series 75*, 1 (July 2007), 012023.

[59] Simão Ferreira, C. J., Kuik, G., van Bussel, G., and Scarano, F. Visualization by PIV of dynamic stall on a vertical axis wind turbine. *Experiments in Fluids 46*, 1 (Aug. 2008), 97–108.

[60] Speck, R. *Generalized Algebraic Kernels and Multipole Expansions for massively parallel Vortex Particle Methods.* PhD thesis, University of Wuppertal, Germany, 2011.

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

[62] Taylor, C., and Hood, P. A numerical solution of the Navier-Stokes equations using the finite element technique. *Computers & Fluids 1*, 1 (Jan. 1973), 73–100.

[63] Tryggeson, H. *Analytical Vortex Solutions to the Navier-Stokes Equation.* PhD thesis, Växjö University, Sweden, 2007.

[64] Tutty, O. R. A Simple Redistribution Vortex Method (with Accurate Body Forces). *ArXiv e-prints* (Sept. 2010).

[65] Van Der Walt, S., Colbert, S. C., and Varoquaux, G. The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering 13*, 2 (2011), 22–30.

[66] Vermeer, L., Sø rensen, J., and Crespo, A. Wind turbine wake aerodynamics. *Progress in Aerospace Sciences 39*, 6-7 (Aug. 2003), 467–510.

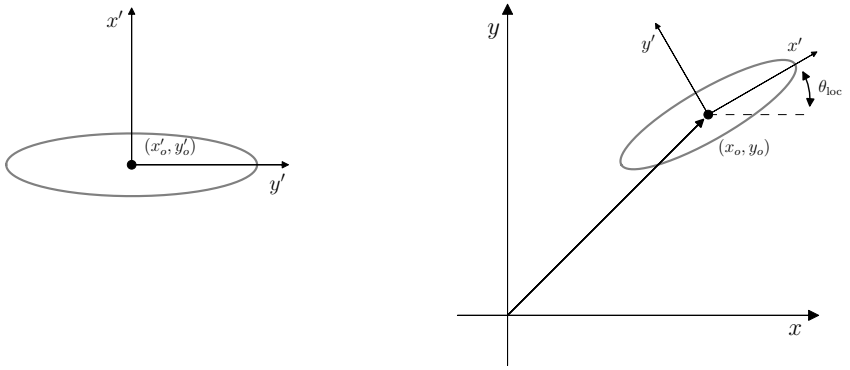
[67] Wee, D., and Ghoniem, A. F. Modified interpolation kernels for treating diffusion and remeshing in vortex methods. *Journal of Computational Physics 213*, 1 (2006), 239–263.

[68] Winckelmans, G., Cocle, R., Dufresne, L., and Capart, R. Vortex methods and their application to trailing wake vortex simulations. *Comptes Rendus Physique 6*, 4-5 (May 2005), 467–486.

Appendix A

Coordinate Systems

The geometries in the simulation are defined in their respective local coordinate systems $[x', y']$. Figure A.1a shows an elliptical geometry defined in its local coordinate system about its origin. The local origin point is defined such that it is the center of rotation and any rotation will be prescribed about this point.



(a) Local coordinate system $[x, y]'$ **(b)** Global coordinate system $[x, y]$

Figure A.1: Ellipse defined in **(a)** the local coordinate system and **(b)** the global coordinate system. The geometry is positioned using the displacement vector $[x_o, y_o]$ and rotated by θ_0 about the local origin point.

The body is then transformed to the global coordinate system $[x, y]$ by the displacement vector $\mathbf{x}_o = [x_o, y_o]$ (i.e the global position of the local origin) and a local rotation by θ_{loc} about the local origin $[x'_o, y'_o]$. The transformation of an arbitrary point $\mathbf{x}'_p = [x'_p, y'_p]$ in the local coordinate system to the global coordinate system is given as,

$$\mathbf{x}_p = \begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} \cos\theta_{loc} & -\sin\theta_{loc} \\ \sin\theta_{loc} & \cos\theta_{loc} \end{bmatrix} \cdot \begin{bmatrix} x'_p - x'_o \\ y'_p - y'_o \end{bmatrix} + \begin{bmatrix} x_o \\ y_o \end{bmatrix} \tag{A.1}$$

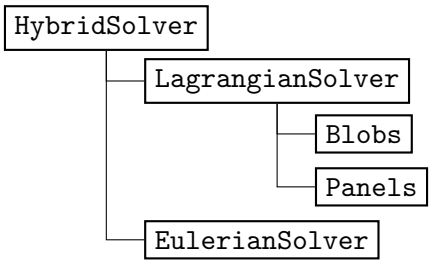
where $\mathbf{x}_p = [x_p, y_p]$ is the global position of the point. The Eulerian solver defines the body mesh in the local coordinate system is then transformed to global position using these parameters. Similarly, the panel geometry for the Lagrangian solver is defined in the same fashion. For a moving bodies problem, the displacement vector and the local rotation angle can be updated to prescribe the motion.

Appendix B

pHyFlow Code Structure

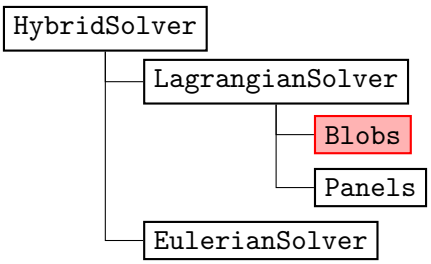
The document outlines the pHyFlow code structure. The pHyFlow functions are organized into several classes. The functions related to the vortex particles are placed inside the Blobs class. The functions related to the panel problem are inside Panels class. The LagrangianSolver class is made to couple the functions of the vortex blobs and the vortex panel together. The functions of the Eulerian domain are placed inside the EulerianSolver class, where the Navier-stokes grid problem is solved. Finally, coupling of all the problems are done with the HybridSolver class. Note, all the classes are capable of handling multi-body / multi-domain problem within them and LagrangianSolver class and the HybridSolver class only couples methods together.

pHyFlow Structure:

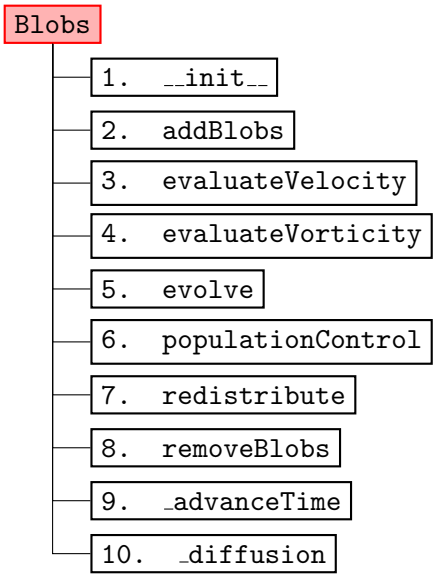


Blobs Class

The main structure of the Blobs class. This class contains all the function related to the calculation of the vortex blobs.



Class structure:



Attributes:

Attributes	Description
blobControlParams	The diffusion parameters. It is a dictionary containing all the parameters of the diffusion method used for the simulation. Contains: stepRedistribution, stepPopulationControl, gThresholdLocal, gThresholdGlobal.
computationMethod	computationMethod (tuple) with the type of Biot-Savart solver (direct, fmm) and the type of hardware to use (cpu, gpu).
deltaTc	The size of the convective time step Δt_c
deltaTd	The size of the convective time step Δt_d
diffusionParams	A dictionary containing all the parameters related to the computation of the diffusion step. Specifies the diffusion scheme and other specific parameters. Contains: method, c2.

`g`	The strength of the vortex blobs α.
`gThresholdGlobal`	Maximum value of variation of total vorticity due to the removal of blobs during population control.
`gThresholdLocal`	Minimum value of circulation to consider for each vortex blob when selecting blobs to remove during population control.
`h`	The size of the cell associated to the vortex blobs. Corresponds to the minimum spacing between the core of two neighboring cells. It is related to the core size of the blob, σ, and to the spacing h by the expression $Ov = h/\sigma$.
`integrationMethod`	`integrationMethod` (`fe`, `rk4`) the type of time integrator used: `fe` forward Euler, `rk4` Runge-Kutta 4^{th} order.
`nu`	The fluid kinematic viscosity, used to calculate the diffusion coefficient: `c2` and diffusion time step `deltaTd`, Δt_d.
`numBlobs`	The number of blobs.
`overlap`	The overlap ratio between neighboring blobs.
`plotVelocity`	A flag that defines if velocity is to be plotted or not.
`sigma`	The core size of the vortex blobs.
`stepDiffusion`	The frequency of diffusion steps.
`stepPopulationControl`	The frequency of population control.
`stepRedistribution`	The frequency of redistribution of blobs.
`timeIntegrationParams`	A dictionary containing all time integration parameters of the simulation. Contains the definition of the time integration scheme possibly additional parameters specific to the scheme.
`t`	The current time of the simulation.
`tStep`	The current time step of the simulation.
`velocityComputationParams`	A dictionary containing all the parameters related to the computation of induced velocities. Specifies computation scheme (direct or fmm) and hardware to use (cpu or gpu).
`vInf`	The free stream velocity.
`x`	The x coordinates of the vortex blobs.
`y`	The y coordinates of the vortex blobs.

Table B.1: Attributes of `Blobs` class and their description.

`__init__`

Description: Initialize the `Blobs` class with either the given input parameters or by a reading a `file` containing all the necessary parameters.

Input Parameters

File Name	Containing all the parameters to re-initalize the class.	
	— or —	
Parameters	Vorticity Field	: {`xBlob, yBlob, gBlob`} or {`wFunction, xBounds, yBounds`}
	Blob parameters	: `overlap, h`
	Time Step parameters	: `deltaTc, nu, stepRedistribution, integrationMethod, computationMethod`
	Population control parameters	: `stepPopulationControl, gThreshold`

Descriptions of the parameters:

Vorticity field		*Default*
`xBlob,yBlob`	: the x, y blob coordinates.	-
`gBlob`	: the circulation Γ_i associated to each of the vortex blobs.	-
	— *or* —	
`wExactFunction`	: the function that returns the exact value of vorticity ω at any given x, y coordinates.	
	Input parameters : `xEval,yEval`	
	Assigns : `-`	-
	Returns : `wEval`	
`xBounds, yBounds`	: the x, y bounds of the domain where the particles was originally distributed.	-

Blob parameters		*Default*
`overlap`	: the overlap ratio h/σ.	1.0
`h`	: the size of the cell h associated to the blobs. *Note:* Cells are square.	-

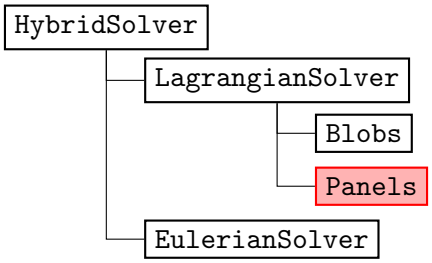
Time step parameters		*Default*
`deltaTc`	: the size of the convective time step Δt_c.	-
`nu`	: the fluid kinematic viscosity ν, used to calculate the diffusion coefficient c^2 and diffusion time step size ΔT_d.	-
`stepRedistribution`	: the redistribution step frequency.	1
`integrationMethod`	: the time integration method (`FE`: Forward euler , `RK4`: 4^{th} order Runge-Kutta).	RK4
`computationMethod`	: the calculation method to evolve the blobs, (`Direct`: Direct Method, `FMM`: Fast-Multipole Method) using (`CPU`, `GPU`).	{FMM, GPU}.

Population control parameters		*Default*
`stepPopulationControl`	population control step frequency	1.
`gThreshold`	: the tuple with minimum **and** maximum value of the circulation Γ_{min}.	-

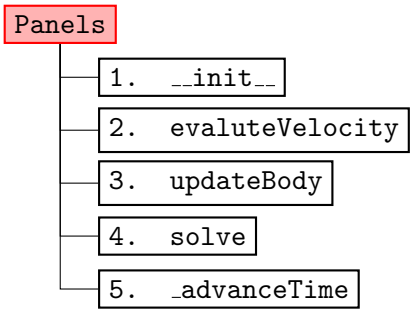
Free stream velocity		*Default*
`vInf`	: The free-stream velocity function, returning the velocity action on the vortex blobs.	-
	Input parameters : `t`	
	Assigns : `-`	-
	Returns : `vx,vy`	

`Panels` class

The main structure of the panel method class `Panels`. This class contains all the functions related to the calculation of panels.



Class structure:



Attributes:

Attributes	Description
`A`	The inter-induction matrix $\mathbf{A}$, the LHS of the problem.
`cmGlobal`	The global position vector for each of the $\mathbf{N}$ body, refining the position of the local panel $(0, 0)$ in the global coordinate system.
`deltaT`	The simulation time step size ΔT
`geometryKeys`	The dictionary containing all the parameters of the geometry. Contains: `xPanel` (the x coordinate of the $\mathbf{M}$ panel corners.), `yPanel` (The y coordinate of the $\mathbf{M}$ panel corners), `cmGlobal`, `thetaLocal`, `dPanel` (The off-set of the panel collocation point from the panel mid-point).
`nBodies`	The number of panel bodies.
`norm`	The x, y normal vector of each panel.
`normCat`	The global concatenated x, y component of the panel normal vector at each collocation points.
`nPanels`	The number of panels in each body/geometry.
`nPanelsTotal`	The total number of panels.
`panelKernel`	A string defining panel kernel type.
`problemType`	A string defining the panel problem is of a `moving` type or of a `fixed` type.
`solverCompParams`	The dictionary containing solver computation parameters.
`sPanel`	The vortex sheet strengths γ of $\mathbf{M}$ panels.
`t`	The current time t of the simulation.
`tang`	The x, y tangent vector of each panel.

tangCat	The global concatenated x, y component of the panel normal vector at each collocation points.
thetaLocal	The local rotation angle θ w.r.t to the local coordinate system. The rotational will be performed around the local reference point $(0, 0)$, i.e around the global center of rotation point cmGlobal.
tStep	The current step of the simulation.
velCompParams	A dictionary containing the velocity computation parameters, method and hardware.
xyCPGlobal	The global x, y coordinate of the panel collocation points.
xyCPGlobalCat	The global concatenated x, y coordinate of the panel collocation points.
xyPanelGlobal	The global x, y coordinate of the panel bodies.
xyPanelGlobalCat	The global concatenated x, y coordinate of the panel bodies.
xyPanelLocal	The local x, y coordinate of the panel bodies.

Table B.2: Attributes of Panels class and their description.

__init__

Input Parameters

File Name	Containing all the parameters to re-initalize the class.
Parameters	Panel coordinates : {xCP, yCP, xPanel, yPanel, cmGlobal, thetaLocal}
	External velocity : externVel

Description: Initialize the panels class with the given input parameters. In the case of a multibody problem, a list of panel coordinates can be given and internally it takes care of the inter-coupling.

Panel coordinates

xCP,yCP	: the local x, y-coordinates of the panel collocation points.
xPanel,yPanel	: the local coordinate of the panel edges. *Note*: Should have a closed loop (end with initial point coordinates).
cmGlobal	: the position of reference points of a given panel body.
thetaLocal	: the rotational angles of the panel body axes w.r.t to the global x-axis.

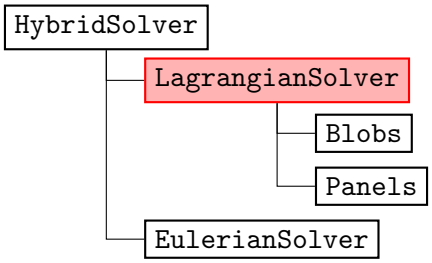
External velocity

externVel	: Reference to an external velocity **function** acting of the panels. For the panel case, the external velocity will the induced velocity of the blobs + freestream vortexBlob.evaluateVelocity.

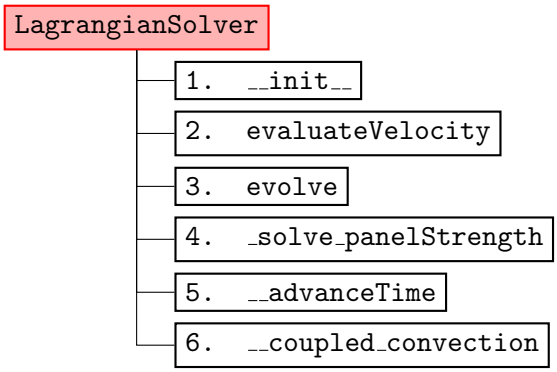
Input parameters : xCP,yCP
Assigns : -
Returns : vxCP,vyCP

LagrangianSolver Class

The main structure of the Blobs + Panels (LagrangianSolver) class. This class contains all the function related to the calculations of panel with vortex blobs.



Class structure:



Attributes:

Attributes	Description
deltaT	The inter-induction matrix **A**, the LHS of the problem.
gTotal	The total circulation of the Lagrangian domain.
t	The current time t of the simulation.
tStep	The current step of the simulation.
vInf	The x, y component of the free-stream velocity.
Blobs	The vortex blobs class Blobs.
Panels	The vortex panels class Panels.

Table B.3: Attributes of LagrangianSolver class and their description.

__init__

Input Parameters

File Name	Containing all the parameters to re-initalize the class.
Parameters	vortexBlobs : {vortexBlobs} class.
	panels : panels class.

Description: Initialize the `vortexMethod` class using **vortexBlob+panelMethod** classes.

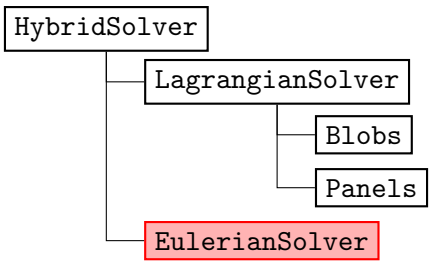
Input parameters:

`Blobs`: vortex particle class

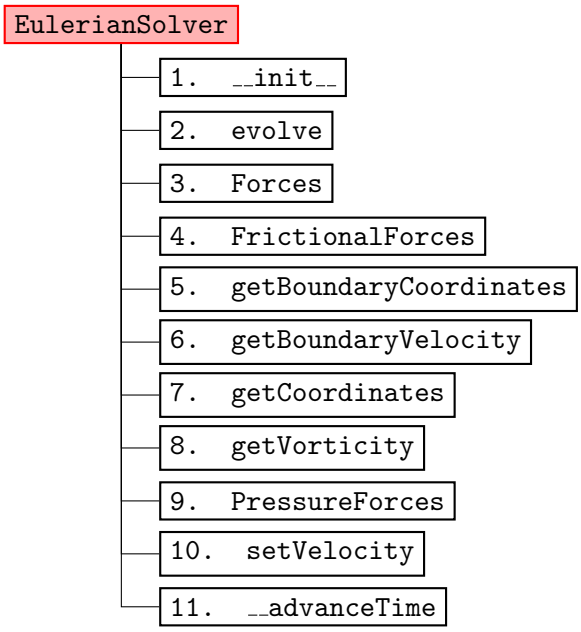
`Panels`: panel method class

EulerianSolver

The main structure for the Navier-stokes class `EulerianSolver`. This class contains all the functions related to computation of the Navier-stokes problem. Below is set of functions that acts as the interface to the class.



Class structure:



Attributes:

Attributes	Description
`deltaT`	The time step size Δt.
`deltaTMax`	The maximum allowable time step size $\max\{\Delta t\}$.
`cfl`	The CourantFriedrichsLewy condition stability number CFL.
`cmGlobal`	The x, y position of the mesh local reference point $(0, 0)$ in the global coordinates.
`hMin`	The minimum mesh cell size.
`nu`	The fluid kinematic viscosity ν.
`probeGridMesh`	The *local* x, y coordinates of the probe grid mesh.

probeGridParams	The dictionary containing all the parameters of the probe grid for extracting the vorticity data.
solverParams	The dictionary file containing all the solver parameters.
t	The current time of the simulation.
thetaLocal	The local rotational angle θ of the mesh domain. Therefore, the rotation will be done about local reference point $(0, 0)$, i.e cmGlobal in the global coordinate system.
tStep	The current step of the simulation.
uMax	The maximum fluid velocity max$\{\mathbf{u}\}$.

Table B.4: Attributes of EulerianSolver class and their description.

__init__

Description: Initialize the navierStokes class either using a fileName containing all the necessary parameter for initialization or by explicitly inputing the parameters.

Input Parameters

File Name	Containing all the parameters to re-initalize the class.	
	– or –	
Parameters	Mesh data	: mesh, boundaryDomains
	Geometry position	: cmGlobal, thetaLocal
	Fluid parameters	: uMax, nu
	Solver options	: cfl
	Probe grid parameters	: x0, y0, Lx, Ly, hx, hy

Description of the parameters:

Mesh data

mesh	: the mesh data file.
boundaryDomains	: the boundary mesh domain data file.

Geometry position

cmGlobal	: the x, y position of the geometry in global coordinates.
thetaGlobal	: the rotation angle (in rad) of the geometry in global coordinate system.

Fluid parameters

uMax	: the maximum fluid velocity U_{max}. Used to determine the maximum time step size Δt_{max}.
nu	: the fluid kinematic viscosity ν, for incompressible navier-stokes problem.

Solver options

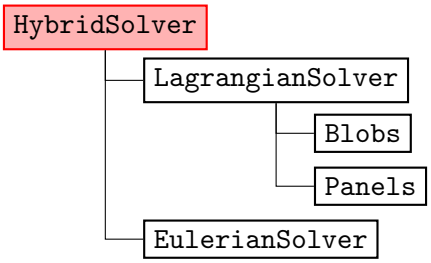
cfl	: the CFL stability parameter. If explicit time marching scheme, $CFL < 1$.

Probe grid parameters

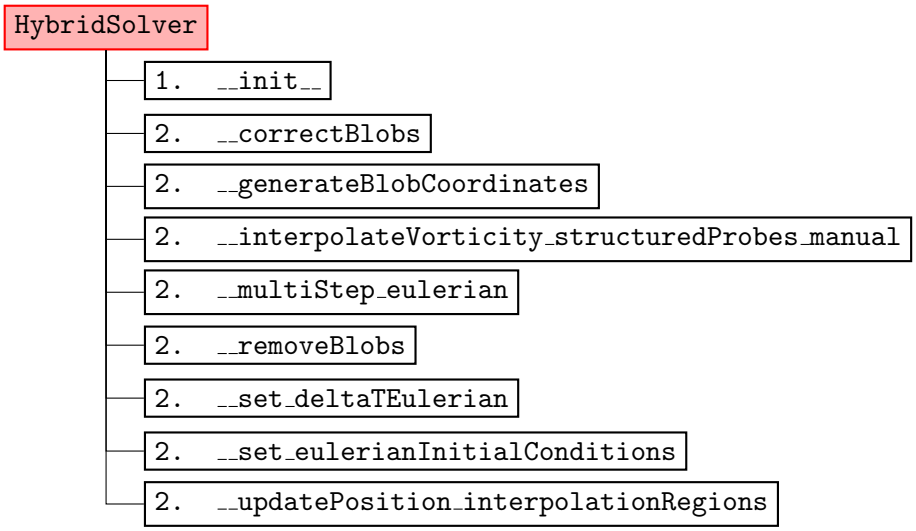
`x0,y0`	: the x, y coordinate of the origin of the probe grid.
`Lx,Ly`	: the x, y size (width and height) of the probing grid.
`hx,hy`	: the x, y spacing of the probe grid cell.

HybridSolver Class

The main structure for the hybrid class HybridSolver. This class contains all the functions related to computation of the hybrid problem.



Class structure:



Attributes:

Attributes	Description
deltaTEulerian	The time step size of the Eulerian sub-domain Δt_E.
deltaTLagrangian	The time step size of the Lagrangian sub-domain Δt_L.
nu	The fluid kinematic viscosity ν.
t	The current time t of the simulation.
tStep	The current step of the simulation.
vInf	The x, y component of the free-stream velocity.
interpolationRegion	The dictionary containing the surfacePolygon and boundaryPolygon defining the boundaries of the interpolation region for each Eulerian sub-domains. The geometry is identified by the keys of the Eulerian sub-domain found in multiEulerian. The coordinates are defined in local coordinate system of the Eulerian grid and will be transformed (rotated + moved) during the evolution step.

`lagrangian`	The Lagrangian solver class contains all the parameters related to simulation the flow in lagrangian sub-domain.
`multiEulerian`	The `multiEulerian` is solver class containing all the Eulerian sub-domains of the hybrid problem.

Table B.5: Attributes of `HybridSolver` class and their description.

`__init__`

Input Parameters

File Name	Containing all the parameters to re-initalize the class.	
Parameters	`vortexMethod`	: {`vortexMethod`} class.
	`navierStokes`	: `navierStokes` class.
	Interpolation region	: `xPolygon, yPolygon`
	Motion functions	: `T, cmGlobal, thetaGlobal, cmDotGlobal, thetaDotGlobal`

Description: Initialize the `hybrid` class using `LagrangianSolver` + `EulerianSolver` classes.

Input parameters:

`LagrangianSolver`: The vortex method containing `Blobs` and **Panels** classes which can already handle the multi-body problem.

`EulerianSolver`: The Navier-Stokes grid solver class (if multiple: list of `EulerianSolver` classes). The number of navier-stokes class has to be same as the number of vortex panels.

Interpolation Region: the Navier-Stokes class (if multiple: list of `EulerianSolver` classes). Should be equal to number of Navier-Stokes classes. The interpolation region should be defined as list of x, y coordinates of the polygon of the interpolation region.

Motion function: the function describing the motion of all the geometries in the hybrid class.

Interpolation Regions	
`xPolygon,yPolygon`:	the new x, y coordinate of the polygons description the interpolation region. The polygon should have a closed loop (end with starting coordinates) before continuing to the next polygon. In the case of multiple polygons, a list of `xPolygon,yPolygon` should be given and should be as many as the number of navier-stokes domain.

Motion function

`T`	: the current time.
`cmGlobal`	: a list of new positions of the geometries in the hybrid problem.
`thetaGlobal`	: a list of new rotational angle of the geometries in the hybrid problem.
`cmDotGlobal`	: a list of current displacement velocity of the geometries in the hybrid problem.
`thetaDotGlobal`	: a list of current rotational velocity of the geometries in the hybrid problem.

Input parameters : `T`
Assigns : `-`
Returns : `cmGlobal,thetaGlobal,cmDotGlobal,thetaDotGlobal`