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

This is the summary of the thesis.

Acknowledgements

I wish to thank the following persons...

Delft, The Netherlands
Date TBD

L. Manickathan B.Sc.

Contents

List of Figures

List of Tables

Nomenclature

Latin Symbols

c^2	Diffusion parameter	-
$\mathcal{E}$	Enstrophy	$\mathrm{m}^2\,\mathrm{s}^{-2}$
h	Nominal particle spacing	m
h_ν	Characteristic diffusion distance	m
$\mathbf{K}$	Biot-Savart kernel	-
$\mathbf{K}_\sigma$	Vortex blob kernel	-
$\mathbf{A}$	Vortex panel influence matrix	-
k_d	Diffusion frequency multiple	-
$\hat{\mathbf{n}}$	Normal vector	-
N_p	Number of particles	-
overlap	Overlap ratio of the blobs	-
p	Pressure	Pa
r	Radial position	m
$\hat{\mathbf{s}}$	Tangent vector	-
T	Cell of Finite Element mesh	-
t	Time	s
$\mathcal{T}_h$	Finite Element mesh	-
$\mathbf{u}$	Velocity	$\mathrm{m}\,\mathrm{s}^{-1}$
$\mathbf{u}_b$	Velocity of the body	$\mathrm{m}\,\mathrm{s}^{-1}$
$\mathbf{u}_\gamma$	Vortex sheet induced velocity	$\mathrm{m}\,\mathrm{s}^{-1}$
$\mathbf{u}_{\mathrm{ext}}$	External induced velocity	$\mathrm{m}\,\mathrm{s}^{-1}$

$\mathbf{u}^h$	Discrete velocity	$\mathrm{m\,s^{-1}}$
u_θ	Angular velocity	$\mathrm{m\,s^{-1}}$
$\mathbf{u}_\infty$	Free-stream velocity	$\mathrm{m\,s^{-1}}$
$\mathbf{u}_\phi$	Potential velocity	$\mathrm{m\,s^{-1}}$
u_r	Radial velocity	$\mathrm{m\,s^{-1}}$
$\mathbf{u}_{\text{slip}}$	Boundary slip velocity	$\mathrm{m\,s^{-1}}$
$\mathbf{u}_\omega$	Vortical velocity	$\mathrm{m\,s^{-1}}$
v	Test function	-
V	Trial vector function space	-
$\hat{V}$	Test vector function space	-
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 the particle	m

Greek Symbols

α_p	Circulation of the particle	$\mathrm{m^2\,s^{-1}}$
β_p	Corrected circulation of the particle	$\mathrm{m^2\,s^{-1}}$
Γ	Circulation	$\mathrm{m^2\,s^{-1}}$
Γ_b	Circulation of moving body	$\mathrm{m^2\,s^{-1}}$
γ	Vortex sheet strengths	s
Γ_c	Circulation of the vortex core	$\mathrm{m^2\,s^{-1}}$
Γ_γ	Circulation of vortex sheet	$\mathrm{m^2\,s^{-1}}$
Γ_ω	Circulation of the fluid	$\mathrm{m^2\,s^{-1}}$
Δt_c	Convection time-step size	s
Δt_d	Diffusion time-step size	s
ϵ	Relative error	-
ζ_σ	Smooth cut-off function of the blob	-
ν	Kinematic viscosity	$\mathrm{m^2\,s^{-1}}$
ξ	Scale relative position of particle to stencil node	-
ρ	Density	$\mathrm{kg\,m^{-3}}$
σ	Core size	m
τ	Scaled viscous time	$\mathrm{m^2}$
Ω	Fluid domain	-
$\partial\Omega$	Boundary of the domain Ω	-
Ω_E	Eulerian fluid domain	-
Ω_L	Lagrangian fluid domain	-

ω	Vorticity	s^{-1}
ω^h	Discrete vorticity field	s^{-1}

Abbreviations

1-D	One-Dimensional
2-D	Two-Dimensional
AD	Actuator Disk
BEM	Blade Element Momentum
CFD	Computional Fluid Dynamic
CG	Continuous Galerkin
CSVM	Constant-Strength Vortex Method
DG	Discontinous Galerkin
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
HELVPM	Hybrid Eulerian-Lagrangian Vortex Particle Method
ICNS	Incompressible Navier-Stokes
IPCS	Incremental Pressure Correction Scheme
LHS	Left Hand Side
LSTSQ	Least-Square solution method
MPI	Message Passing Interface
MRS	Modified Remeshing Scheme
PC	Population Control
PIV	Particle Image Velocimetry
PSE	Particle Strength Exchange
RWM	Random Walk Method
SCS	Simple Coupling Strategy
SRS	Simple Redistribution Scheme
VAWT	Vertical-Axis Wind Turbine
VPM	Vortex Particle Method

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 and is getting more popular, and have also become more affordable. Novel renewable technologies such as Vertical-Axis Wind Turbine (VAWT) are now a promising research field that can satisfy this growing demand.

VAWT are unlike the normal wind turbine, which are mounted on a mast away from the ground and generates energy by spinning perpendicular to the ground, figure 1.1. Whereas the VAWT, figure 1.1a, spins parallel to the ground with its hub located at the ground [56]. The advantages of the VAWTs are what makes them ideal for a source of renewable energy. As the turbine is located at the ground, it is easily accessible and is easily maintained. The second main advantage of the VAWT is the way it dissipates its wake. Near-wake experiments of Ferreira (2009) [46], and simulations of Vermeer (2003) [54] have shown that the fluid past the turbine is more turbulent. Due to this higher mixing, the flow is able to recover much earlier than the convectional wind turbines. This

(a) VAWT: Darrieus wind turbine[15]

(b) HAWT: Offshore wind turbine [16]

Figure 1.1: VAWT vs. HAWT

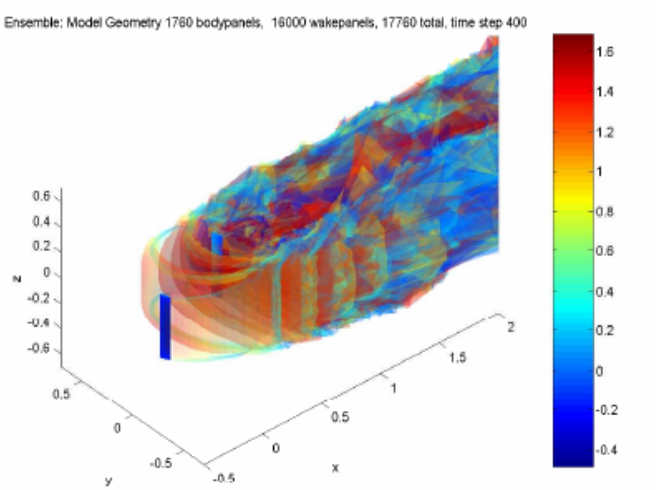


Figure 1.2: 3-D Unsteady Panel simulation of a Straight-bladed VAWT showing the strength of the shed vorticity. The VAWT blades interact with its own wake increasing the complexity of the wake geometry [22]

means that it possible to places VAWTs much closer to each other and so a VAWT farm can potentially give more power per area. Furthermore, VAWTs operate independent 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 passes through its own wake, complex wake-body interactions take places, figure 1.2. These have adverse effect 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 takes place [48]. This 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, the VAWT operates at 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 the research is to develop an efficient, reliable, and accurate numerical method for modeling the flow around a Two-Dimensional (2-D) VAWT, enabling to deduce its correct performance characteristics. The two approaches of investigating the flow around a turbine is by either using a numerical method to model the flow, or by performing a real-life experimental tests, such as a wind tunnel experiment.

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) [47], have shown that it was possible to acquire flow characteristics around the blade, and the simulations have been useful at validating the numerical methods for the VAWT. The downside to experimental investigation is that is it very expensive to

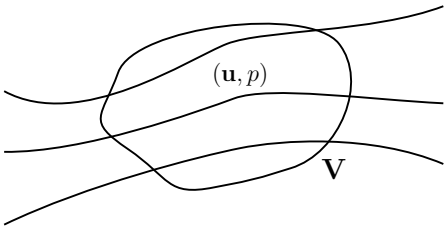


Figure 1.3: Eulerian formulation of the fluid. We observe a given volume $\mathbf{V}$ and evaluate the change in properties of the fluid (from incompressible flow: velocity $\mathbf{u}$ and pressure p) at time passes.

investigate all types of airfoil geometries, blade geometries and VAWT configurations. However, investigation this is vital in understanding the performance characteristics of VAWT. Furthermore, the model sizes are limited by the dimensions of the wind tunnel, and investigations with arrays of VAWTs in a wind tunnel is difficult.

Numerical methods are therefore a popular alternative as the cost of simulation is becoming progressively smaller, and the accuracy of the models are increasing day by day. In the research field, there exists many models with various orders of accuracy. 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 quick, however they lack the accuracy that are obtained by experimental simulation. Complex blade-wake interactions such as dynamic stalls and flow separations 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 Dynamic (CFD) methods discretizes the fluid into smaller regions and solves the Navier-Stokes equation in each region (or grids). This type of formulation is known as an Eulerian formulation as we are evaluating the change in flow property of a given region, figure 1.3. In order to fully resolve the flow around the turbine, we would have to discretize the fluid very small at the blade where the vortex cores are very small, and we would need large grids far away from the blades where the vortex cores are much larger. Therefore, we would need grid size of various order of magnitudes at various regions of the fluid. This requires a large number of grids, and furthermore as the blades are constantly moving, this introduction additional limitations when defining the problem in Eulerian formulation.

An alternative method is to use the vortex formulation of the Navier-Stokes equations, referred to as vorticity equation. This method is ideal because when describing it in Lagrangian formulation, the evolution of the vorticity is resulted from the interaction between vortices in the fluid. The removes the requirement of grids, figure 1.4. In addition, using simulation acceleration methods such as Fast Multipole Method (FMM) and parallel computation in Graphical Processing Units (GPU), they are much more efficient that typical CFD methods. However, vortex method cannot inherently take in 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.

So, 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

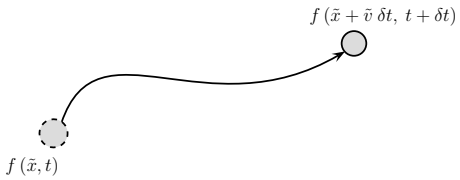


Figure 1.4: Lagrangian formulation of the fluid. We track the path of the individual fluid elements as time passes.

very efficient 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 advantage of both methods, we have decided to use a domain-decomposition method, referred to as Hybrid Eulerian-Lagrangian Vortex Particle Method (HELVPM). In this method, the Eulerian grid method will be used at the region around the blade (i.e. the near-wall region). The Lagrangian vortex method will be used in the wake region of the body. With proper coupling of these methods, we can ensure that this numerical method can capture not only the near-wake phenomenons 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 research that can help us accomplish our goal. The research question 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 wind tunnel experimental setup?*
- *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 the develop an efficient and accurate numerical method that is not only capable of capture 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. The investigation will be performed for 2-D geometries and the accuracy of the model needs to established first for simple problems before investigating more we can tackle complicated problems. In other words, the initial goal is to develop the hybrid vortex particle method and verify the approach. During this

process, the solver will be verified and validated against test cases starting from simpler problems and gradually developing more complex features.

The investigation is currently possible for 2-D simulations because full 3-D simulations are beyond the scope of the thesis. This is mainly due to the lack of research period that we have at hand. However, 3-D simulation can simply be extended from the accomplishments of the 2-D development work. A final goal would have been to investigate the VAWT performance, however due to time constrictions, it too was beyond the scope of the thesis. Thus we can now summarize the aim and the plan of the research.

Research aim and plan:

- *Develop the Hybrid method for capturing small-scale phenomenons and large scale phenomenons.*
- *Ensure this tool is efficient, reliable, and accurate.*
- *Verify and validate the tools with test cases.*

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 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 solves different domains of the fluid. The domain decomposition method is simply splitting the domain of interest and using the appropriate methods in each domain. For the problem of VAWT, as the boundary is non-trivial and is the source of vorticity, the full Navier-Stokes model will be used here, and away from the body where only the convection of the vorticity field is interested, the fast and efficient vortex particle method will be used, figure 1.5.

Several researches have already been done: Cottet and Koumoutsakos (2000a)[18], Guermond and Lu (2000) [25] simulated the advection dominated flows; Ould-Salhi et al. (2001) [41] blended the finite difference and vortex method together; Winckelmans et al. (2005a) [57] investigated the trailing vorticies; Daeninck (2006) [20] used a simplified coupling strategy, coupling Vortex Particle Method and Finite Diference Method; Stock (2010) [50] expanded Daeninck's strategy, coupling Vortex Particle Method and Finite Volume Method and modeled a 3-D rotor.

When evaluating the previous 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

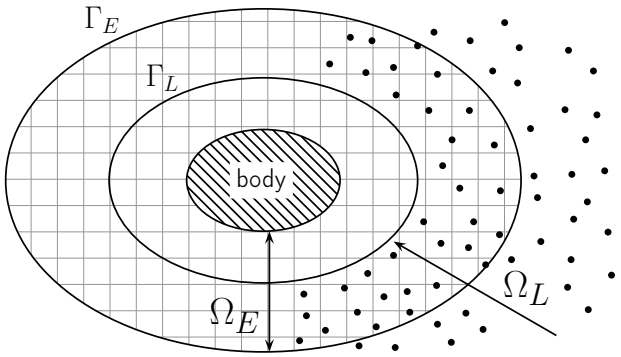


Figure 1.5: Standard domain decomposition using Schwartz iteration for coupling the two methods. Eulerian domain Ω_E (near the body), and Lagrangian domain Ω_L (away from the body). Figure is based on Guermond (2000) [25]

.

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 Ω_L and Ω_E respectively, match at the overlap region $\Omega_E - \Omega_L$. Figure 1.5 shows the domain decomposition using the Schwartz alternating method. The summary of a single iteration of the Schwartz alternating method is as follows:

- Determine the Eulerian boundary condition, the stream function ψ_{Γ_E} at the Eulerian boundary Γ_E, extracted from the Lagrangian stream function $\psi_L$in the Lagrangian domain Ω_L.
- Solve for the stream function ψ_E in the Eulerian domain Ω_E with the new boundary condition Γ_E.
- Determine the Lagrangian condition, the stream function ψ_{Γ_L} at the Lagrangian boundary Γ_L, extracted from the Eulerian stream function ψ_E in the Eulerian domain Ω_E.
- Solve the stream function ψ_L in the Lagrangian domain with the boundary conditions ψ_{Γ_L} at the Lagrangian boundary Γ_L.

This procedure is iterated until the stream functions of both the domains converges [41]. Onces the stream function is determined in both the domains, the velocity field can be obtained. Using the velocity field, we can evolve the vorticity field in both the domain.

As we realized now, the downside to this procedure is that we have to solve the stream functions in both Ω_E and Ω_L iteratively, till we converge to a solution. This makes the computation very expensive, especially when we are dealing with large number of vortex particles. Therefore, for this project, we are using the coupling techniques that is based on the research work of Daeninck (2006) [20] and Stock (2010) [50].

1.3.1 Simple coupling strategy

This approach will be referred to as the Simple Coupling Strategy (SCS). It is simpler that the Schwartz iterative method, as no iteration is needed for the coupling procedure.

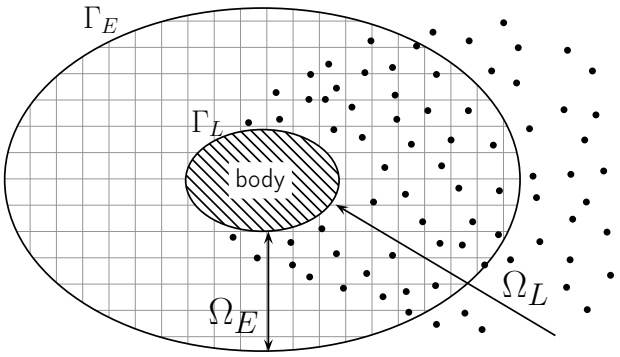


Figure 1.6: Modified domain decomposition without Schwartz alternating method. Lagrangian domain extends up to the surface of the body. Figure is based on Daeninck (2006) [20].

The basic procedure is to solve the vortex method in the full fluid domain using relatively coarse resolution of the near-wall region. Then we use the grid solver in the near-wall region to capture the detailed features of the boundary layer and transfer the vorticity field at this region to the vortex particles, figure 1.6. Therefore, the grid solver basically acts as the correction for the under-resolved regions of the Lagrangian method. The functionality of this strategy has been demonstrated by Daeninck and was found to be significantly faster than the Schwartz coupling strategy. The features of the simple coupling strategy can summarized as follows:

- Eulerian method is used to resolve the near-wall region, at the Eulerian domain Ω_E, enabling it to capture important features of the boundary layer (such as flow separation) with great accuracy.
- Lagrangian method is used to capture the wake, at the Lagrangian domain Ω_L, and to efficiently evolving the wake.
- The accurate solution of the Eulerian domain is transfered to the Lagrangian domain according to the coupling algorithm of Daeninck [20] and Stock [50].
- The boundary conditions for the Eulerian domain is retrieved from the Lagrangian domain.

The algorithm to the Simple Coupling Strategy (SCS) follows from Daeninck's doctoral thesis, [20]. Figure 1.7 shows the overview to the algorithm and can be summarized as follows:

1. **Correct Lagrangian:** Use the solution of the Eulerian domain Ω_E (in the near-wall region) to correct the solution of the Lagrangian domain Ω_L, that is overlapping the Eulerian domain.
2. **Evolve Lagrangian:** With the modified solution, evolve the Lagrangian solution from time step t_n to next time step t_{n+1}.

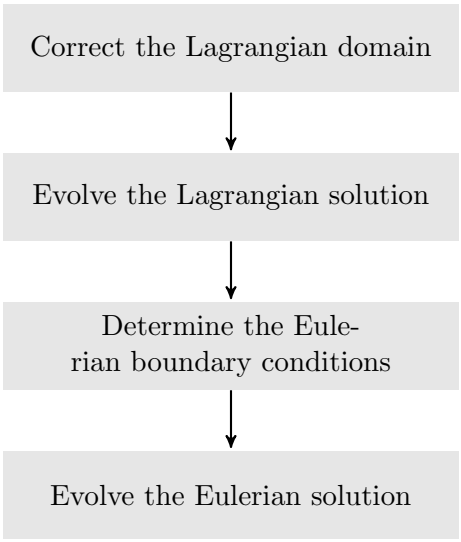


Figure 1.7: Flowchart of the simple coupling strategy. The flowchart shows the procedure to evolve both methods from t_n to t_{n+1}.

3. **Determine Eulerian boundary conditions:** Use the Lagrangian solution of time t_n and t_{n+1} to determine the boundary conditions of the Eulerian domain.
4. **Evolve Eulerian:** With the boundary condition, evolve the Eulerian solution from t_n to t_{n+1}.

This is the basic approach for coupling the Eulerian method in the Eulerian domain Ω_E with the Lagrangian method in the Lagrangian domain Ω_L without the iterative Schwartz algorithm.

Furthermore, the SCS handles the Lagrangian boundary condition differently from the classic hybrid method. Typically during the evolution process of the Lagrangian domain, the shedding of the vorticity is also defined in the Lagrangian method. However, in our coupling strategy, the Lagrangian method is under-resolved at the boundary and cannot be used to resolve the vorticity flux at the body. Instead, we use the Eulerian method to resolve the boundary, and the Eulerian method acts as the vorticity generator for the Lagrangian method. However, there as some assumptions that we must satisfy, for this coupling strategy to be valid:

- At t_n, before the evolution of both method to t_{n+1}, the Lagrangian solution matches Eulerian solution at the near-wall region.
- 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.
- Even though the Lagrangian domain is under-resolved in the near-wall region, it should be able to provide accurate boundary conditions for the Eulerian external boundary.

1.4 Verification and Validation Test Cases

The test-cases that are used for this thesis are summarized as given:

Lamb-Oseen vortex [34] [52]
Lamb-Oseen vortex test case is an analytical solution derived from the diffusion 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. This test case focuses on the evolution of the vorticity field and therefore is an ideal test case to verify and validate the convection and diffusion of the vorticity.

Clercx-Bruneau dipole [14]
The Clercx-Bruneau dipole test case is the simple case of dipole colliding with the wall. This test case will be used to verify and validate the coupling of the Eulerian and the Lagrangian method. This test cases focuses on the generation of vorticity from- the wall making it ideal to verify and validate the proper transfer of vorticity from the Eulerian domain to the Lagrangian domain.

Impulsively started cylinder [32] [9] [5] [35]
The impulsively started cylinder test case is used to analyse 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 [40]
The elliptic airfoil test cases 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 phenomenons 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 phenomenons.

1.5 Methodology

The initial steps of the development of the hybrid vortex methods is as follows:

1. Develop the vortex particle method
2. Validate the vortex particle method against a Lamb-Oseen convection test case.
3. Develop the vortex panel method to deal with the boundaries for the vortex particle calculation.
4. Validate the vortex panel method by solving a potential flow around a cylinder.
5. Develop the grid solver that is based on the Finite Element method.
6. Validate the grid solver against test cases: impulsively starting cylinder, dipole-Wall interaction.

Once all the components have been validated, the methods will be coupled and validated against similar test cases.

7. Couple vortex particle, vortex panel and grid solver together.
8. Validate it with the previous generated test case solution.
9. Introduce more complicated phenomenons: multiple geometry (i.e multiple grid meshes) and moving boundaries, if it feasible in the constraints of a master thesis.

If the coupled solver has been validated with the test cases, the final step will be to simulated the flow around a VAWT and investigating the performance vs. numerical and experimental data.

1.6 Thesis Outline

!!! To be done at the end !!!

Chapter 2

Lagrangian Domain: Vortex Particle Method

2.1 Introduction to Vortex Particle Method

Vortex Particle Method (VPM) is a branch of computational fluid dynamics that deals with the evolution of the vorticity of the fluid in a Lagrangian description. Typically, the fluid is viewed at a fixed window where the change in the fluid properties are evaluated. However, the Lagrangian formulation, regard the fluid as a collection of the particles carrying the property of the fluid.

Unlike the typical Eulerian method that require the discretization of all the fluid domain, VPM only needs fluid elements where there is vorticity. This means that the VPM are inherently auto-adaptive method that only simulated the flow of interest. Furthermore, with the computational acceleration methods such as Fast-Multipole Method (FMM) and parallel computation in Graphics Processing Unit (GPU), VPM are much more efficient at evolving the vorticity field that the typical Eulerian methods.

The main literature to the vortex method (the Lagrangian domain of our hybrid method), is the book of Cottet and Koumoutsakos, Vortex Methods: Theory and Practice [18]. It given an insight to the fundamentals of vortex method (specifically vortex particle method) and gives a summary on the hybrid methods.

2.1.1 Vorticity

Vorticity is the key subject of all vortex methods. Vorticity ω, the governing element of vortex particle method, is defined as

$$\omega = \nabla \times \mathbf{u}, \tag{2.1}$$

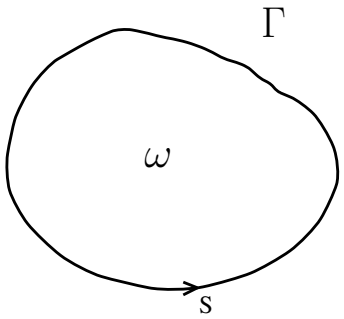


Figure 2.1: Definition of the circulation in the fluid

where $\mathbf{u}$ is the vector velocity field. The circulation Γ is defined by the stokes theorem,

$$\Gamma = \int_L \mathbf{u} \cdot d\mathbf{s} = \int_A \omega \cdot \mathbf{n} \; dA, \tag{2.2}$$

and represents the integral vorticity of the domain defined by the line $\mathbf{s}$. Figure 2.1 depicts the relation with velocity $\mathbf{u}$, vorticity ω and the circulation Γ in an arbitrary domain.

2.1.2 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 described by book Vortex Methods: Theory and Practice by Cottet and Koumoutsakos (2000) [18]. The 2-D incompressible Navier-Stokes momentum equation is given as,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u}, \tag{2.3}$$

relating the velocity field $\mathbf{u}(\mathbf{x}, t)$ to the pressure field $p(\mathbf{x}, t)$, the kinematic viscosity ν and density ρ. Furthermore, we also have to satisfy the incompressibility constraint given as,

$$\nabla \cdot \mathbf{u} = 0. \tag{2.4}$$

To attain the velocity-vorticity formulation, we should take the curl of the velocity-pressure formulation $\mathbf{u} - p$ of the Navier-Stokes equation. Taking the curl of the 2-D momentum equation 2.3, we derive the vorticity transport equation,

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \nabla^2 \omega, \tag{2.5}$$

which only relates the vorticity to the velocity enabling us to neglect the pressure field. Note that as we are dealing with the 2-D flow, we can neglected the 3-D stretching term. We also assume unit density, as we are dealing with incompressible flow, and it is trivial.

2.1.3 Viscous splitting algorithm

Vortex particle method 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 [12], showed that using the viscous splitting algorithm, it is possible to take in account of the viscous effects of the flow.

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

1. **Convection** (sub-step 1):
$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = 0; \tag{2.6}$$
2. **Diffusion** (sub-step 2):
$$\frac{\partial \omega}{\partial t} = \nu \nabla^2 \omega. \tag{2.7}$$

The first sub-step of the evolution deals with the convection of the vorticity. Note that, by convection we imply the advection of the vorticity field where the diffusion process is neglected. The second sub-step is where we deals with the diffusion of the vorticity field.

There are several advantage to this type of evolution. As the convection and diffusion are handled separately, there is minimum dispersion during the convection step and furthermore, the is no restriction of the advection CFL number, as shown by Wee (2006) [55].

There are many ways of dealing with the diffusion of the vorticity field. During this project, we initially used a modified interpolation kernel by Wee (2006) [55] that can simultaneously treat diffusion and remesh the vortex particles. Later, we used a simple redistribution scheme by Tutty (2010) [53], that was less constrained on time-step size for the diffusion giving us more flexibility during the simulation, see section 2.4.

2.2 Spatial Discretization: Generation of Vortex Blobs

Before we can convect and diffuse the vorticity field, we must discretize the vorticity field into Lagrangian fluid elements. For this project we used vortex blobs, that was introduced by Chorin [18]. Vortex blobs are mollified vortex particle carrying the local circulation of the fluid. They describes a smooth vorticity field, and this is important because non-modified vortex particles have asymptotic vorticity distribution and causes issues when particles approach each other.

2.2.1 Biot-Savart law

The velocity field can be decomposed using the Helmholtz decomposition given as,

$$\mathbf{u} = \mathbf{u}_\omega + \mathbf{u}_\phi, \tag{2.8}$$

where $\mathbf{u}_\omega$ is the rotational component of the velocity and $\mathbf{u}_\phi$ is the irrotational component, the solenoidal and the potential velocity respectively. In an unbounded flow we have $\mathbf{u}_\phi$ equal to the free-stream velocity $\mathbf{u}_\infty$, whereas for bounded flow, we must include the presence of the body, see section 2.5. The velocity can be related to the vorticity using the Biot-Savart law given as,

$$\mathbf{u}_\omega = \mathbf{K} \star \omega, \tag{2.9}$$

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

$$\mathbf{K} = \frac{1}{2\pi \left|\mathbf{x}\right|^2} \left(-x_2, x_1\right). \tag{2.10}$$

From the kernel, we see that as the distance to the kernel center approaches zero ($\mathbf{x} \to 0$), the kernel goes to infinity. This is a singularity and must be dealt with for a stable numerical scheme.

2.2.2 Discrete form of vorticity field

The spatial discretization of the fluid domain is done through N quadrature points. With the Biot-Savart law, we can treat these quadratures as discrete particles carrying the local quantities. The discrete vorticity field is given as,

$$\omega\left(\mathbf{x}, t\right) \simeq \omega^h\left(\mathbf{x}, t\right) = \sum_p \alpha_p\left(t\right) \delta\left[\mathbf{x} - \mathbf{x}_p\left(t\right)\right], \tag{2.11}$$

where α_p is the estimate of the circulation around the particle $\mathbf{x}_p$. We must note that ω^h is the discrete vorticity field and therefore in an approximation of the continuous vorticity field ω. The discrete form of the velocity is therefore written as,

$$\mathbf{u} \simeq \mathbf{u}^h = \sum_p \mathbf{K}\left[\mathbf{x} - \mathbf{x}_p\left(t\right)\right] \alpha_p\left(t\right). \tag{2.12}$$

Thus the discrete vorticity field is an N-body problem inducing velocity on each other. So, to evolve the vorticity field, we simply have to evolve the vortex particles with the induced velocity acting on it. This is the main advantage of the vortex particle method as there are efficient methods to evaluate the induced velocities. The N-body problem can be parallelized in GPUs, and furthermore the induced velocity calculations can be simplified using fast summation methods such as the Fast Multipole Method (FMM), reducing the problem from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, in an ideal case.

In order to deal with the asymptotic because of the Biot-Savart kernel, equation 2.10, we can use a mollified kernel, that gives as a smooth velocity distribution.

2.2.3 Mollified vortex kernels

A vortex particle with a mollified core has a non-zero core size σ. This mollified vortex particle is called vortex blob. The advantage of a vortex blob is that the with its smooth distribution of the vorticity, the singularity disappear, and the numerical instability when

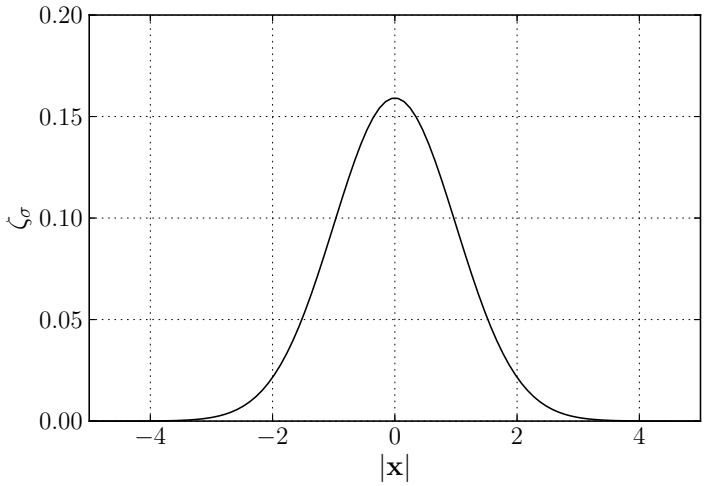


Figure 2.2: The smoothing function ζ_σ for a gaussian distribution with $k = 2$, $\sigma = 1.0$.

the blobs get too close to each other does not occur anymore. An ideal choice for a cutoff function that gives a smooth vorticity distribution is the Gaussian distribution, shown in figure 2.2. Gaussian kernels satisfy the requirement for smooth distribution, decaying quickly away from the blob. It is defined as,

$$\zeta_\sigma = \frac{1}{k\pi\sigma^2} \exp\left(-\frac{|\mathbf{x}|}{k\sigma^2}\right), \tag{2.13}$$

where k is either 1, 2 or 4 and it determines the width of the kernel, and σ is core-size of the blob. Note that smoothing function is chosen such that $\int \zeta = 1$, ensuring that circulation is conservated when mollified. So using the smooth cut-off function ζ_σ, the mollified Biot-Savart kernel $\mathbf{K}_\sigma$ is given as,

$$\mathbf{K}_\sigma = \mathbf{K} \star \zeta_\sigma. \tag{2.14}$$

The discrete mollified vorticity field, represented by vortex blobs is now defined as,

$$\omega^h(\mathbf{x}, t) = \sum_p \alpha_p(t)\, \zeta_\sigma\left[\mathbf{x} - \mathbf{x}_p(t)\right], \tag{2.15}$$

and similarly the discrete mollified velocity field is given as,

$$\mathbf{u}^h(\mathbf{x}, t) = \sum_p \mathbf{K}_\sigma\left[\mathbf{x} - \mathbf{x}_p(t)\right] \alpha_p(t). \tag{2.16}$$

Koumoutsakos and Chorin [18], explained that in order to ensure the smoothness of the velocity field, the vortex blobs needs to have an overlap with each other. The overlap ratio is defined as,

$$\text{overlap} = \frac{\sigma}{h}, \tag{2.17}$$

where h is the nominal particle spacing, and σ is the blob core size. Figure 2.3 shows the visual representation of the overlap ratio.

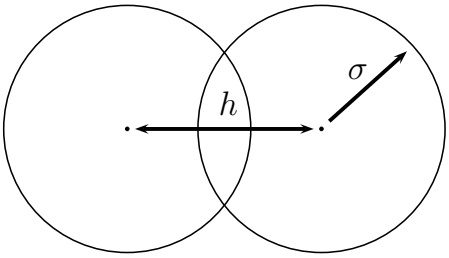


Figure 2.3: Vortex blob with overlap an σ/h

During the evolution of the vortex blobs, this overlap constraint is violated due to the strains in the flow. The vortex blobs cluster together at certain region and at others they disperse. This localized clustering effect is seen as a lagrangian grid distorting, and has to be treated using remeshing schemes, covered in section 2.3.1.

2.2.4 Vortex blob initialization

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

$$\alpha_p = \omega_p \cdot h^2, \tag{2.18}$$

meaning that the particle carry 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 area, given by equation 2.2, and therefore we will be conserving the circulation as all the circulation in the fluid is represented by the blobs.

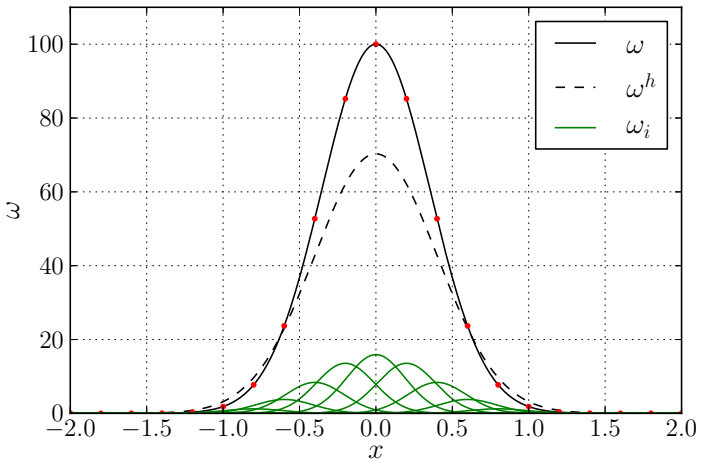


Figure 2.4: Mollified vorticity field of a gaussian vorticity distribution with overlap $= 1.0$, $\sigma = 0.19$, and $h = 0.19$. Vortex blob strengths was assigned using equation 2.18, sampling the exact vorticity [•, red dot]. Figure depicts the exact vorticity distribution ω [—, solid black], vorticity distribution of each blob ω_i [—, dashed green], and the mollified vorticity field from the blobs ω^h [- -, dashed black].

However, this type of initialization suffers some accuracy in the vorticity field itself as the vorticity field represented by the blobs is no longer the initial vorticity field, but the mollified vorticity field, equation 2.15. Barba and Rossi [1], has described this problem as gaussian blurring of the original vorticity field. Even though the particle have acquired the correct circulation strengths (i.e the local property), when evaluating the mollified vorticity field, we see that there is a mismatch with the original vorticity field, figure 2.4.

Another way of viewing this phenomenon is to say that the conservation of circulation is only valid globally (for an infinite domain), but once we try to conserve circulation locally (e.g. in a given sub-domain), is does not satisfy anymore. Figure 2.4 shows this effect for a simple gaussian initial vorticity distribution ω. The mollified vorticity field is given as ω^h and even though the integral of both function is the same (conservation of circulation is satisfied), the vorticity functions do not match. This is does not cause a lot of issues when we only dealing with vortex particle method, however once we start using domain decomposition methods, this problem is an issue. As the vorticity is the communication between both methods, we must have accurate vorticity distribution.

A common strategy, used by Koumoutsakos, Cottet, and other for recovering the initial vorticity field is to perform the Beale's method [3] [18].

Beale's Iterative Method

The Beale's method is particle circulation processing scheme where the circulation of the particles are modified such that the mollified vorticity field matches the indented vorticity field (the initial vorticity field). The recovery of the vorticity field is done by performing a discrete deconvolution,

$$\sum_{j}^{N} \beta_j \zeta_\sigma \left(\mathbf{x}_i - \mathbf{x}_j\right) = \omega_i, \tag{2.19}$$

where β_j is the circulation of the particles at positions $\mathbf{x}_j$ such that it matches the exact vorticity ω_i at the position $\mathbf{x}_i$ that we are evaluating. As we are try to solve for a N unknown problem, we must set up an N system of equations. Multiplying both sides with the area associated to the blobs, we get

$$\mathbf{A}_{ij}\beta_j = \alpha_i^{\text{exact}}, \tag{2.20}$$

where

$$\mathbf{A}_{ij} = \zeta_\sigma \left(\mathbf{x}_i - \mathbf{x}_j\right) \cdot h^2. \tag{2.21}$$

This is an $N \times N$ matrix containing the weights of the influence of each particle on each other. When we are dealing with large number of vortex blobs, we see that it is very expensive to invert the matrix $\mathbf{A}$, meaning that we would have to use a more efficient method. Furthermore as the deconvolution problem is a severely ill-condition problem [18], we should not directly invert the matrix. Beale's proposition to this problem was to iteratively solve for the solution,

$$\beta_j^{n+1} = \alpha_i + \beta_i^n - \mathbf{A}_{ij} \cdot \beta_j^n \tag{2.22}$$

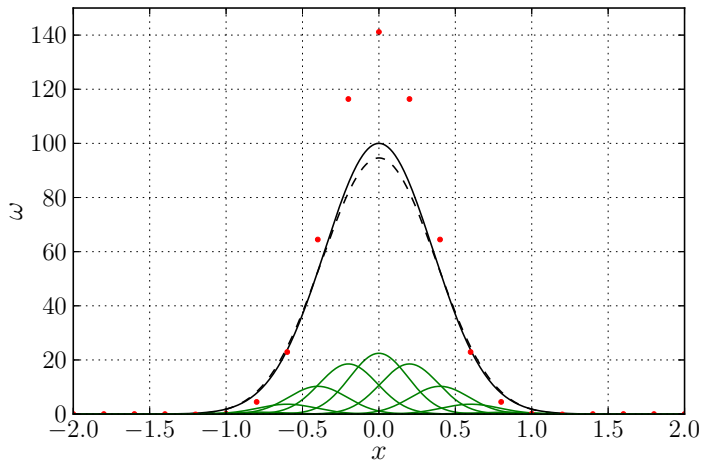


Figure 2.5: Mollified vorticity field after two Beale's iteration, with overlap $= 1.0$, $\sigma = 0.19$, $h = 0.19$. Figure depicts exact vorticity distribution ω [—, solid black], vorticity distribution of each blob ω_i [—, dashed green], the mollified vorticity field ω^h [- -, dashed black], and the corrected blob cell vorticity $\tilde{\omega} = \beta/h^2$ [•, red dot].

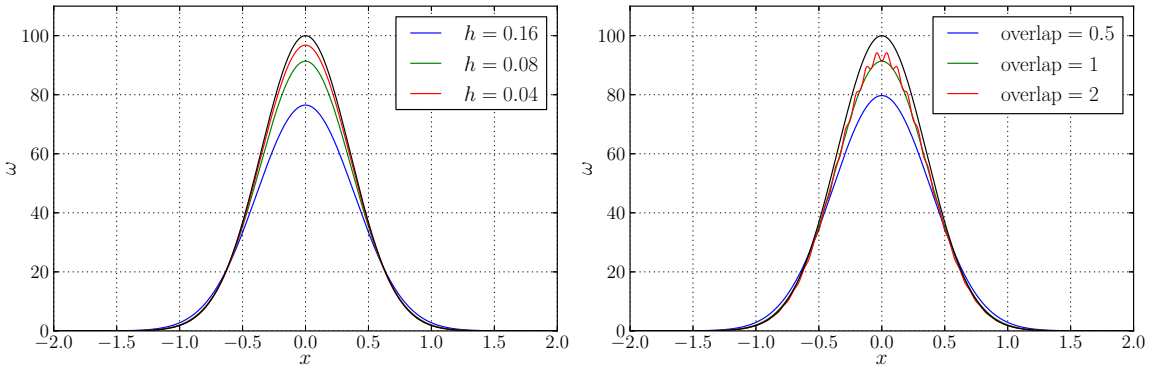
We see that with just two iterations, the error between the mollified and exact vorticity field reduces drastically, figure 2.5. Koumoutsakos and Cottet [18], had shown that there was a drastic improvement in the velocity with just two to three iterations. However, the average vorticity of the blobs cell $\tilde{\omega} = \beta/h^2$ (red dot in figure 2.5), is more peaky and no longer matches the initial vorticity distribution. This means that if we try to fix the mollified vorticity distribution to match the correct initial vorticity distribution, we corrupt the local circulation even more.

The another downside of using the Beale's correction method that it is only valid for an infinite domain as it performs a discrete deconvolution of a gaussian kernel with an infinite span. Therefore it applies to all of the fluid domain, meaning that if we are dealing with a decomposed domain with finite bounds, the Beale's correction cannot be used and would result in spurious results. So the Beale's correction can and should only be used for correcting the vorticity field of the whole fluid domain.

Convergence of particle discretization

An alternate method to reduce the gaussian blurring of the vorticity field is to reduce the overlap (i.e. increase the overlap number) of the vortex blobs, and also increase the spatial resolution. This approach does not solve the gaussian blurring problem, but only minimizes its effect.

Figure 2.6 shows mollified vorticity field results from modifying the spatial resolution parameters. Figure 2.6a 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.



(a) Convergence of h with overlap $= 1.0$ **(b)** Convergence of overlap with $h = 0.08$

Figure 2.6: Convergence of the vorticity initialization by modifying the spatial resolution. Figure depicts exact vorticity field ω [—, solid black], the initialized vorticity distribution with various parameters.

The second parameter we can adjust is the overlap of the blobs, as seen in figure 2.6b. The blob spacing h is set to $h = 0.08$, and we see that by increasing the overlap number, the mollified vorticity approaches the exact vorticity field. However, as shown by Koumoutsakos [18], if the overlap is too low, we lose the smooth recovery of the vorticity field. This can be seen when the overlap $= 2.0$. We see that the mollified vorticity field has a fluctuating solution. This would results in non-smooth velocity field which is an acceptable solution.

Therefore, to minimize the error between the mollified vorticity field and the exact vorticity field, we will use an overlap $= 1.0$, and reduce the nominal blob spacing h to a minimum. The advantage of this approach is that we can employ this correcting in a finite domain unlike the Beale's correction. However, as this correction method only minimizes the effect and not directly solve the problem of mismatched vorticity fields, it is recommended that we find a solution to this problem in future.

2.3 Convection of vortex blobs

The convection equation 2.6 of the viscous-splitting algorithm, is solved as system of ODEs, where,

$$\frac{\mathrm{d}\mathbf{x}_p}{\mathrm{d}t} = \mathbf{u}\left(\mathbf{x}_p\right), \tag{2.23}$$

with

$$\frac{\mathrm{d}\alpha_p}{\mathrm{d}t} = 0. \tag{2.24}$$

This problem in now solved using a Lagrangian formulation where the vortex blobs is used to discretize the vorticity field. Using the Biot-Savart law, equation 2.12, we can determine the induced velocities acting on each particles. The calculation of the induced velocities an N-body problem and is parallelized using GPU hardware and simplified using an FMM approach.

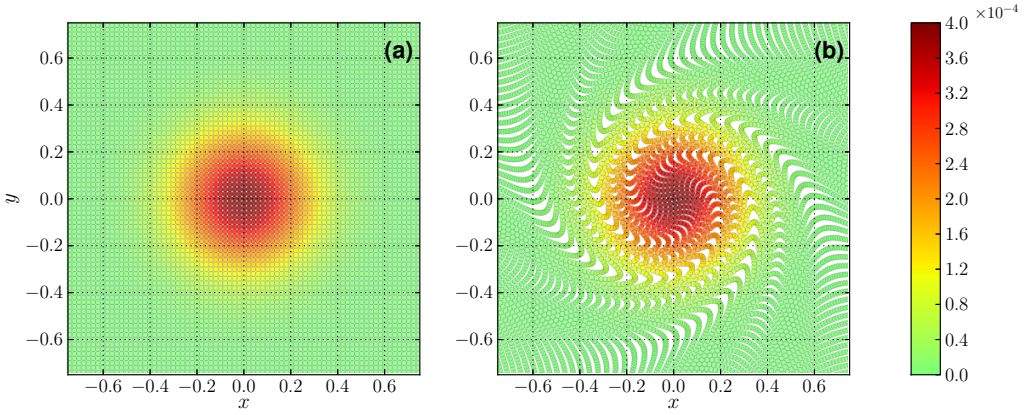


Figure 2.7: Lagrangian distortion of the vortex blobs after 100 time steps. The initial vorticity field is $\omega(\mathbf{x}, 0) = \exp(-12|\mathbf{x}|)$ with $\Delta t = 0.1$, $\sigma = 0.02$, and overlap $= 1.0$. Figure depicts **(a)** the initial distribution of the vortex blobs, and **(b)** the final distribution of the vortex blobs after 100 time steps.

Once we determine the induced velocity acting on each vortex blobs, they can be convected using the equation 2.23. To retain accuracy during the convection process, we used a 4^{th} order Runge-Kutta method, an explicit time marching scheme. As the diffusion is done at the next sub-step, the strengths of the particles do not change during the convection process.

2.3.1 Remeshing scheme: Treating lagrangian grid distortion

During the convection step, the vortex blobs will start to cluster together at certain regions of the flow, where as at the other regions, we see that there are no blobs. This clustering and dispersing effect of the blobs is due to the high strains in the flow, figure 2.7. This means that the overlap ratio is not satisfied everywhere in the flow. As we have seen before, in figure 2.6b, overlap ratio has a large impact on the mollified vorticity field and must be satisfied. In order to treat this Lagrangian grid distortion, we can remesh the vortex blobs on to a uniform grid, regaining our intended overlap ratio.

The remeshing is done by interpolating the strengths of the vortex blobs from distorted lagrangian grid on to a uniform grid. The strengths of the blobs of the new uniform grid is defined as,

$$\alpha_p = \sum_q \tilde{\alpha}_q W\left(\frac{x_p - \tilde{x}_q}{h}\right), \tag{2.25}$$

where the strengths of the blobs $\tilde{\alpha}_q$ of the distorted Lagrangian grid $\tilde{x}_q$ is transfered to the regular Lagrangian grid x_p using the interpolation kernel W. Figure 2.8 shows the remeshing of one vortex blob of the distorted grid on to the structured uniform grid. During this transfer, we must ensure that the properties of the fluid are conserved. The interpolation kernel is constructed by ensuring that the circulation, the linear impulse, and the angular impulse of the fluid is conserved. For this project, we used the M_4' interpolation kernel.

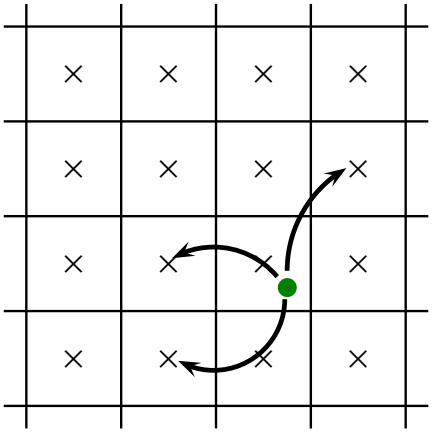


Figure 2.8: Remeshing of a single vortex blob [•, green dot] onto a uniform grid defined by the (4×4) 2-D stencil.

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 [39]. For a One-Dimensional (1-D) problem, the M'_4 interpolation kernel is given as,

$$\mathrm{M'}_4(\xi) = \begin{cases} 1 - \frac{5\xi^2}{2} + \frac{3|\xi|^3}{2} & |\xi| < 1, \\ \frac{1}{2}(2 - |\xi|)^2 (1 - |\xi|) & 1 \leq |\xi| < 2, \\ 0 & 2 \leq |\xi|, \end{cases} \tag{2.26}$$

where

$$\xi = \frac{x_\nu - x_i}{h}, \tag{2.27}$$

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, and with the $m = 4$, it has 4 support nodes in each dimension. Figure 2.9 shows the distribution of the kernel. For the two dimensional problem, the 2-D interpolation kernel is just the tensor product of the 1-D interpolation kernel, and will have a $4^2 = 16$ support nodes, as seen in figure 2.8.

2.4 Diffusion of Vortex Methods

So far, we have ignored the viscous effect of the flow, and in order to simulate a real flow, we must perform the diffusion of the vorticity field. Chorin simulated a viscous flow using the viscous splitting algorithm. The flow is segregated to inviscid and viscous component and during the second sub-step we deal with the diffusion of the vorticity, as show in equation 2.7. The diffusion term of the viscous splitting algorithm can be solved as a system of ODEs, similar to the convection step, given as,

$$\frac{\mathrm{d}\mathbf{x}_p}{\mathrm{d}t} = 0, \tag{2.28}$$

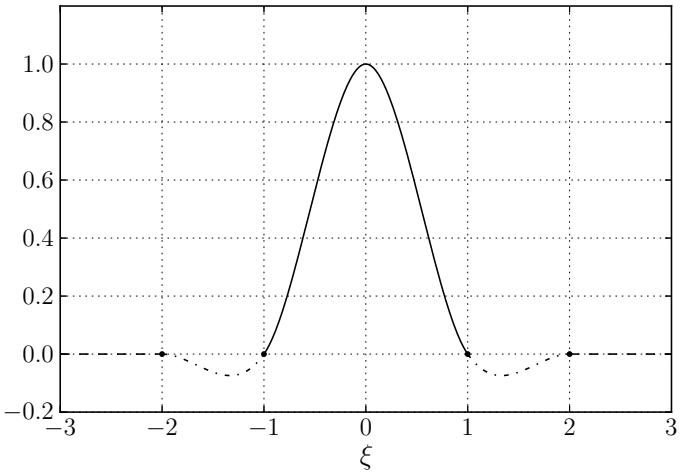


Figure 2.9: M_4' interpolation kernel, a third-order, piecewise smooth, B-spline kernel by Monaghan [39]

with

$$\frac{\mathrm{d}\alpha_p}{\mathrm{d}t} = \nu \Delta \alpha_p. \tag{2.29}$$

So in the diffusion step, we fix the position of the vortex blobs, and only modify the strengths of the particles. This process mimics the diffusion of vorticity. Chorin in 1973 [12], initially employed the Random Walk Method (RWM), which generates and disperse vorticity using pseudo-random number algorithm. However, this method suffers some limitations in accuracy, and since then methods such as Particle Strength Exchange (PSE) method [21], has been a common approach to treat the diffusion.

Particle Strength Exchange

The Particle Strength Exchange method, first proposed by Mas-Gallic in 1989 [21], showed that diffusion can be treated for a particle method with an isotropic, or an anisotropic viscosity by approximating the diffusion operator (a Laplacian) with an integral operator, and discretize the method with particles [49].

Vortex Redistribution Method

$$h_\nu = \sqrt{\nu \Delta t_d} \tag{2.30}$$

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. This means that the VRM scheme (and also the PSE) requires a search algorithm to determine the particles that are within the zone of influence. A direct evaluation would require an $\mathcal{O}\left(N^2\right)$ evaluation. However, this can be optimized using a search tree algorithm, speeding up the search to an $\mathcal{O}\left(\log N\right)$.

2.4.1 Modified remeshing for treating diffusion

From further investigation of the VRM, Ghoniem and Wee saw that the VRM is similar to the remeshing strategy that we used to counter the lagrangian distortion during the convection process. So, Ghoniem and Wee in 2006 [55], proposed to combine the remeshing and the diffusion together in one process. The application of this methodology was later validated by Speck in 2011 [49]. This Modified Remeshing Scheme (MRS), performs diffusion by modifying the interpolation kernel of the remeshing scheme. The key advantage of the MRM is that, now we are dealing with a uniform grid, and no longer requires a search algorithm to find the particles in the zone of influence. This significantly reduced the computational cost, making this diffusion scheme practical for a large scale simulations.

Ghoniem and Wee rederived the interpolation kernel that can perform both the remeshing and the diffusion simultaneously. The kernel transfers the correct fraction of circulation during the remeshing such that it satisfies the diffusion equation. The M'_4 kernel for treating the diffusion is given as,

$$\mathrm{M}'_4(\xi, c) = \begin{cases} 1 - \frac{5\xi^2}{2} + \frac{3|\xi|^3}{2} - c^2\left(2 - 9\xi^2 + 6|\xi|^3\right) & |\xi| < 1, \\ \frac{1}{2}(2 - |\xi|)^2(1 - |\xi|) - c^2(2 - |\xi|)^2(1 - 2|\xi|) & 1 \leq |\xi| < 2, \\ 0 & 2 \leq |\xi|, \end{cases} \tag{2.31}$$

where

$$c^2 = \frac{\nu \Delta t_d}{h^2}, \tag{2.32}$$

is a non-dimensional number that corresponds to the transfer weight for the diffusion. This additional terms in the interpolation kernel accounts for the diffusion process. When $c \to 0$, the interpolation kernel simply turns in to the standard remeshing kernel. 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, we have to ensure that,

$$\frac{1}{6} \leq c^2 \leq \frac{1}{2}. \tag{2.33}$$

However, we see that this c^2 constraint imposes a direct constraint on the maximum and the minimum Δt_d. This would mean that the diffusion time step size Δt_d will be a multiple of the convection step Δt_c,

$$\Delta t_d = k_d \cdot \Delta t_c, \tag{2.34}$$

so the diffusion of the vortex blobs will not be done at every step of the evolution process. This is a problem for hybrid method as we couple the Lagrangian and the Eulerian method 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 should take care of.

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 the diffusion is performed at every step. This was a feasible

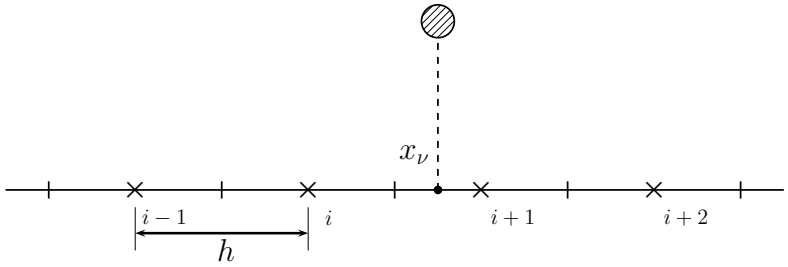


Figure 2.10: One dimensional simple redistribution scheme, diffusing the vortex blobs at $x_i \leq x_\nu \leq x_{i+1}$, onto the four stencil points $k = i-1, \ldots, i+2$, with a grid spacing h.

solution for our initial investigations with low Reynolds number flows, however when had to deal with the high Reynolds number flows, where the convection time step have to be small, we needed a scheme that was not constrained by the minimum diffusion time step.

Simple redistribution scheme

The simple redistribution scheme, based on the Shankar and Van Dommelen [44], developed by Tutty in 2010 [53], makes it possible to remesh and diffuse the vorticity after every convection step. The strengths of the particles after the remeshing and diffusion is given as,

$$\alpha_i^{n+1} = \sum_k \alpha_k^n W_{ki}^n, \tag{2.35}$$

where W_{ki}^n is the fraction of circulation from vortex blob k transferred to vortex blob i by the redistribution during the time step n. Like the MRS for treating diffusion, section 2.4.1, this Simple Redistribution Scheme (SRS) works by transferring the strengths of the particles to a fixed nodes rather than the neighboring nodes, figure 2.10. Therefore, if we use a uniform nodes, we do not need to employ a search algorithm to determine the neighboring nodes of the interpolation kernel. This makes the scheme as efficient as the MRS.

The fractions W_{ki}^n are calculated by conserving the vorticity, the center of the vorticity, the linear, and the angular momentum of the vortex blobs [53]. For the two dimensional problem, the redistribution fractions are simple tensors products of the x, y one dimensional redistribution fractions, given as,

$$W_{kl} = F_k G_l, \quad k = i-1, \cdots, i+2, \quad l = j-1, \ldots, j+2, \tag{2.36}$$

giving it a 16 point stencil. The one-dimension redistribution fractions for x-direction is a linear combination of the two basis solution for a conservative redistribution,

$$F_k = (1-\Delta) \cdot f_k + \Delta \cdot g_k, \quad k = i-1, \ldots, i+2, \tag{2.37}$$

having a four point stencil, as show in figure 2.10. The basis solution of redistribution

are

$$f_{i-1} = \frac{1}{2}(1 - f_i - \xi) \tag{2.38a}$$

$$f_i = 1 - 2\left(\frac{h_\nu}{h}\right)^2 - \xi^2 \tag{2.38b}$$

$$f_{i+1} = \frac{1}{2}(1 - f_i + \xi) \tag{2.38c}$$

$$f_{i+2} = 0 \tag{2.38d}$$

and

$$g_{i-1} = 0 \tag{2.39a}$$

$$g_i = \frac{1}{2}(1 - g_{i+1} - \xi_1) \tag{2.39b}$$

$$g_{i+1} = 1 - 2\left(\frac{h_\nu}{h}\right)^2 - \xi_1^2 \tag{2.39c}$$

$$g_{i+2} = \frac{1}{2}(1 - g_{i+1} + \xi_1) \tag{2.39d}$$

where ξ is given by equation 2.27. The $\xi_1 = \xi - 1$ is the distance between the k^{th} stencil node x_k and the vortex blob that is to be diffused with $x_i \leq x_\nu \leq x_{i+1}$. In the above equation, h_ν is the characteristic diffusion distance during the time Δt_d,

$$h_\nu = \sqrt{\Delta t_d \cdot \nu}. \tag{2.40}$$

The only constraint imposed for a positive redistribution fraction is,

$$\frac{h_\nu}{h} < \frac{1}{\sqrt{2}}. \tag{2.41}$$

giving us the maximum time step size constraint of,

$$\Delta t_d < \frac{h^2}{2\nu}. \tag{2.42}$$

So we see that, this scheme only posses a constraint on the maximum diffusion time step size. For high Reynolds number flows, the convection time step is usually much smaller than the diffusion time step, meaning that now we can ensure that diffusion is performed with every convection step (i.e $k_d = 1$).

2.5 Boundary conditions at solid boundary

So far, we have only dealt with unbounded flow. During bounded flow simulation, we must impose an addition constraint for the boundary, to simulate the flow with a geometry. Using the Helmholtz decomposition, we can have decompose the velocity field in to the rotational and the irrotational components, equation 2.8. We can use the potential component to prescribe the boundary conditions at the solid wall boundary,

$$\mathbf{u}_\phi = \nabla\Phi. \tag{2.43}$$

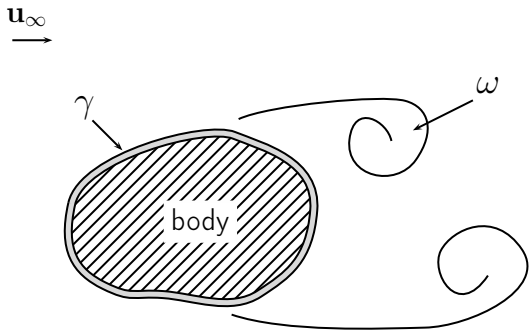


Figure 2.11: Extended vorticity field separated into vorticity in the fluid and the vortex sheet distribution confined to the body.

The incompressibility constraint results in a Laplace's equation for the potential field and a unique solution is obtained by enforcing the wall boundary conditions,

$$\mathbf{u}_b \cdot \hat{\mathbf{n}} = (\mathbf{u}_\omega + \nabla\Phi) \cdot \hat{\mathbf{n}}, \tag{2.44}$$

enforcing the no-through flow at the solid boundary wall, moving at $\mathbf{u}_b$, where $\hat{\mathbf{n}}$ is the normal vector of the solid boundary. The approach for determine the solution to the Laplace's equation is by Green's function formulation. This approach required a singularity distribution over the body resulting in the appropriate boundary condition. Doublets and/or source panels are used to attain the required potential such that equation 2.44 is satisfied.

Linked boundary conditions

However Koumoutsakos, Leonard and Pepin in 1994 [33], suggested to use vortex sheets to enforce the boundary conditions. This alternate approach of enforcing the solid boundary condition is by not to decompose the velocity field into potential and rotational but to consider the solid boundary as an extension of the vorticity field through vortex sheets γ, figure 2.11. Due to the non-zero tangential velocity at the surface, a sudden discontinuity in the velocity field can be considered as vortex sheet. So to enforce the boundary conditions of the solid wall, we must satisfy the no slip velocity at the boundary,

$$\mathbf{u} \cdot \hat{\mathbf{s}} = \mathbf{u}_b \cdot \hat{\mathbf{s}}. \tag{2.45}$$

Koumoutsakos [31], relied only on the vortex sheets to enforce the no-slip velocity. Koumoutsakos, Leonard and Pepin's paper in 1994 [33] stated that, by satisfying no-slip boundary condition, it directly satisfies the no-through boundary conditions, as these boundary conditions are linked. This was also been stated by Shiels [45] and have been further employed by Cooper, Mar and Barba in 2009 [17]. So enforcing the no-slip boundary condition directly satisfies the no-through constraint at the surface.

2.5.1 Boundary integral equations

The Lagrangian method that we are using for the hybrid scheme, is modified according Stock [50]. The Lagrangian method under-resolved the vorticity field in the near-wall

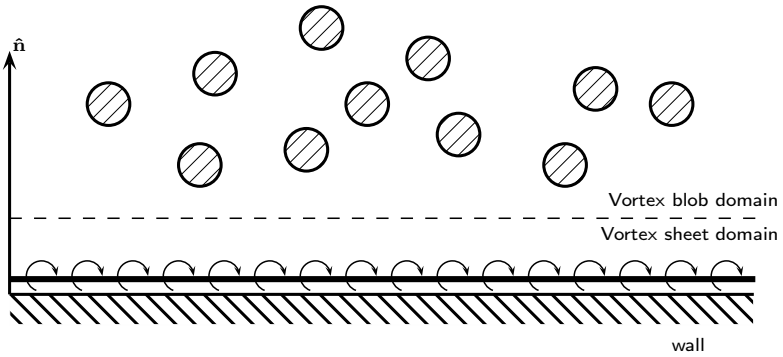


Figure 2.12: Extended vorticity field: Vortex sheet being an extension to the vorticity field (resolved by the vortex blobs), capable of capturing the body bounded vorticity distribution.

region. Furthermore, the vorticity of the fluid is segregated between the vortex blob domain and the vortex sheet domain, as seen in figure 2.12. 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. The extended velocity field (of the extended vorticity field) is summarized as,

$$\mathbf{u} = \mathbf{u}_\omega + \mathbf{u}_\gamma + \mathbf{u}_\infty \tag{2.46}$$

where the $\mathbf{u}_\gamma$ denotes the velocity field induced by the vortex sheet. Enforcing the no-slip boundary conditions, we state that,

$$(\mathbf{u}_{\text{ext}} + \mathbf{u}_\gamma) \cdot \hat{\mathbf{s}} = \mathbf{u}_b \cdot \hat{\mathbf{s}} \tag{2.47}$$

where $\mathbf{u}_{\text{ext}} = \mathbf{u}_\omega + \mathbf{u}_\infty$ is the velocity field induced from the vortex blob 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,

$$(\mathbf{u}_{\text{ext}} - \mathbf{u}_b) \cdot \hat{\mathbf{s}} = \mathbf{u}_\gamma \cdot \hat{\mathbf{s}}. \tag{2.48}$$

Note that as the boundary condition is enforced inside the body (under the vortex sheet), the velocity induced by the vortex sheet $\mathbf{u}_\gamma$ is the negative of equation 2.48. Koumoutsakos [31], 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,

$$-\frac{\gamma(s)}{2} + \frac{1}{2\pi} \oint \frac{\partial}{\partial n} \left[\log \left|\mathbf{x}(s) - \mathbf{x}\left(s'\right)\right|\right] \gamma\left(s'\right) ds' = \mathbf{u}_{\text{slip}} \cdot \hat{\mathbf{s}}. \tag{2.49}$$

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 we have to cancel. The Left Hand Side (LHS) of the equation states at the point s, the velocity discontinuity is due to the vortex sheet of that point and integral of all the other vortex sheets around the body. However, the equation 2.49 is not unique and accepts arbitrary solution for the vortex sheet strengths, therefore we must imposed an

additional constraint on the strength of the vortex sheet. The addition constraint is a constraint on the circulation of the fluid. To ensure conservation of circulation,

$$\Gamma = \Gamma_{\text{body}} + \Gamma_{\text{fluid}} = 0, \tag{2.50}$$

stating that the total circulation in the Lagrangian domain is the sum of the circulation of the body and the circulation of fluid, which should be equal to zero. As the body is in the domain of the vortex sheet, the circulation of the body is represented by the vortex sheet,

$$\Gamma_{\text{body}} = \Gamma_{\gamma_{\text{body}}}. \tag{2.51}$$

The circulation of the body is due to the motion of the body traveling at $\mathbf{u}_b$. One can consider the body as filled with uniform vorticity due to the rotation of the body. Therefore the circulation of a moving body can calculated simply integrating the "vorticity" inside the body,

$$\Gamma_b = \iint_{body} \nabla \times \mathbf{u}_b \; dA. \tag{2.52}$$

Furthermore, as we are using the Lagrangian method according to Stock [50], the boundary layer of the body is not resolved with the vortex blobs, as explained in figure 2.12. Therefore, the vortex sheet has to carry this neglected circulation of the boundary layer, and the circulation of the fluid is given as,

$$\Gamma_{\text{fluid}} = \Gamma_{\gamma_{\text{BL}}} + \Gamma_{\omega}, \tag{2.53}$$

where $\Gamma_{\gamma_{\text{BL}}}$ is the circulation of the boundary layer region represented by the vortex sheet, and Γ_{ω} is the total circulation captured by the vortex blobs. Combining equation 2.51 and 2.53 into equation 2.50, we derive the net circulation of the Lagrangian domain,

$$\Gamma = \Gamma_{\gamma_{\text{body}}} + \Gamma_{\gamma_{\omega}} + \Gamma_{\omega} = 0, \tag{2.54}$$

where the total circulation represented by the vortex sheet is given as,

$$\Gamma_{\gamma} = \Gamma_{\gamma_{\text{body}}} + \Gamma_{\gamma_{\text{BL}}}. \tag{2.55}$$

Thus, the constraint imposed on the net circulation of the vortex sheets is given as,

$$\Gamma_{\gamma} = -\Gamma_{\omega}. \tag{2.56}$$

ensuring that the total circulation of the fluid is zero. The total circulation of the vortex sheet is calculated by integrating the vortex sheet,

$$\Gamma_{\gamma} = \oint_{S} \gamma\,(s) \; ds. \tag{2.57}$$

So we have to solve for the vortex sheet satisfying both the equation 2.49 and the equation 2.57, which can be done using a panel method.

2.5.2 Panel method for treating no-slip boundary condition

Equation 2.49 is solved by discretizing the body into M vortex panels, giving us a system of M equation to determine the M unknowns of the strength of the vortex panels. This is the panel method, which has been extensively demonstrated by Katz and Plotkin [30]. 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 non-linear vortex panel. For this project, we have used a constant-strength vortex distribution that discretized the vortex sheet into straight segments, classified as Constant-Strength Vortex Method (CSVM).

Panel methods are constructed by discretizing the integral equation 2.49, and forming a system of M equations to solve the M unknowns of the vortex panel,

$$\underbrace{\begin{pmatrix} -\frac{1}{2} & a_{12} & \cdots & a_{1M} \\ a_{21} & -\frac{1}{2} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & -\frac{1}{2} \end{pmatrix}}_{\mathbf{A}_{MM}} \underbrace{\begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_M \end{pmatrix}}_{\vec{\gamma}} = \underbrace{\begin{pmatrix} \text{RHS}_1 \\ \text{RHS}_2 \\ \vdots \\ \text{RHS}_M \end{pmatrix}}_{\overrightarrow{\text{RHS}}}, \tag{2.58}$$

where A_{MM} contains the coefficients of the tangential induced velocity of vortex panels $\vec{\gamma}$ acting on each other. The $\overrightarrow{\text{RHS}}$ is given as,

$$\text{RHS} = \mathbf{u}_{\text{slip}} \cdot \hat{\mathbf{s}}, \tag{2.59}$$

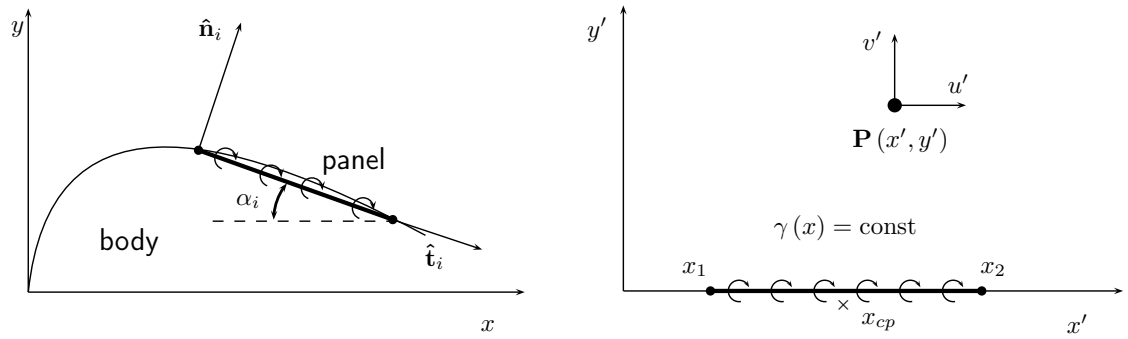
and contains the boundary condition to the system of equations. In addition, we have another constraint on the net circulation of the vortex panels, equation 2.57. In the discrete form, the equation is given as,

$$\sum_{i}^{M} \gamma_i \Delta s = \Gamma_\gamma, \tag{2.60}$$

and results in a $M+1$ system of equations for solving the M unknown strengths of the vortex panels,

$$\underbrace{\begin{pmatrix} -\frac{1}{2} & a_{12} & \cdots & a_{1M} \\ a_{21} & -\frac{1}{2} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & -\frac{1}{2} \\ \Delta s_1 & \Delta s_2 & \cdots & \Delta s_M \end{pmatrix}}_{\mathbf{B}_{(M+1)M}} \underbrace{\begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_M \end{pmatrix}}_{\vec{\gamma}} = \underbrace{\begin{pmatrix} \text{RHS}_1 \\ \text{RHS}_2 \\ \vdots \\ \text{RHS}_M \\ \Gamma_\gamma \end{pmatrix}}_{\overrightarrow{\text{RHS}}}, \tag{2.61}$$

the last equation imposing the circulation constraint on the panels. However, now we see that we have a set of $M+1$ equations with M unknowns, making it an overdetermined problem. The approach to solve such a problem is either by using a Least-Square solution method (LSTSQ), introducing a new unknown, or as used by Katz, eliminating an equation. Enforcing the no-slip boundary condition at each panel location is a vital criteria, and therefore we used the LSTSQ method for solving the problem.



(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 2.13: The two coordinate system of the panel method problem. The figure depicts **(a)** the global panel coordinate system, and **(b)** the local panel coordinates system, as defined by Katz and Plotkins [30].

Constant-Strength Vortex Method

The Constant-Strength vortex method (CSVM) is based on the flat (straight) discretization of the vortex sheet, where the panel have constant vortex strength. To solve the strengths of the panel problem, we enforce the Dirichlet velocity boundary conditions at the collocation point x_{cp}, that is located just below the vortex sheet, shown in figure 2.13b. The coefficient a_{ij} of the influence matrix $\mathbf{A}$ is defined as,

$$a_{ij} = \hat{\mathbf{u}}_{ij} \cdot \hat{\mathbf{t}}_i, \tag{2.62}$$

which represents the tangential influence coefficient of 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.

Figure 2.13a 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 2.13b. In general, 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 system) is given as,

$$u'_{ij} = \frac{\gamma_j}{2\pi}\left[\tan^{-1}\frac{y'_i - y'_{j,2}}{x'_i - x'_{j,2}} - \tan^{-1}\frac{y'_i - y'_{j,1}}{x'_i - x'_{j,1}}\right], \tag{2.63a}$$

$$v'_{ij} = -\frac{\gamma_j}{4\pi}\ln\frac{\left(x'_i - x'_{j,1}\right)^2 + \left(y'_i - y'_{j,1}\right)^2}{\left(x'_i - x'_{j,2}\right)^2 + \left(y'_i - y'_{j,2}\right)^2} \tag{2.63b}$$

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 2.13. The transformation of this vector

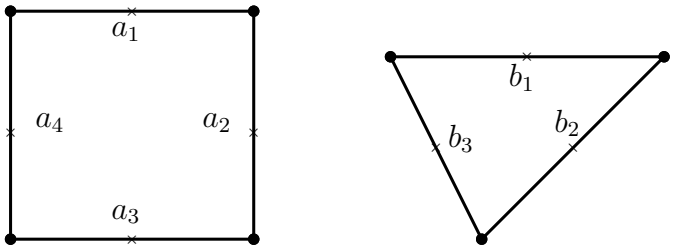


Figure 2.14: Multi-body panel problem: two bodies with different numbers of panels. The figure depicts a square body with 4 panels (a_1, a_2, a_3, a_4), and a triangular body with 3 panels (b_1, b_2, b_3).

(u'_{ij}, v'_{ij}) to the global coordinates is given as,

$$\begin{bmatrix} u_{ij} \\ v_{ij} \end{bmatrix} = \begin{bmatrix} \cos\alpha_j & \sin\alpha_j \\ -\sin\alpha_j & \cos\alpha_j \end{bmatrix} \cdot \begin{bmatrix} u'_{ij} \\ v'_{ij}. \end{bmatrix} \tag{2.64}$$

Note that when $i = j$, the influence coefficient becomes $a_{ij} = -1/2$, represented by the diagonal terms of equation 2.58. If we are dealing with multiple panel bodies (i.e. multiple geometries), as seen in figure 2.14, the panel problem can be solved by constructing a global influence matrix,

$$\underbrace{\begin{pmatrix} c_{a_1a_1} & \cdots & c_{a_1a_N} & c_{a_1b_1} & \cdots & c_{a_1b_M} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{a_Na_1} & \cdots & c_{a_Na_N} & c_{a_Nb_1} & \cdots & c_{a_Nb_M} \\ c_{b_1a_1} & \cdots & c_{b_1a_N} & c_{b_1b_1} & \cdots & c_{b_1b_M} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{b_Ma_1} & \cdots & c_{b_Ma_N} & c_{b_Mb_1} & \cdots & c_{b_Mb_M} \\ \Delta s_{a_1} & \cdots & \Delta s_{a_N} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \Delta s_{b_1} & \cdots & \Delta s_{b_M} \end{pmatrix}}_{\begin{pmatrix} C_{aa} & C_{ab} \\ C_{ba} & C_{bb} \\ \Delta s_a & 0 \\ 0 & \Delta s_b \end{pmatrix}} \underbrace{\begin{pmatrix} \gamma_{a_1} \\ \vdots \\ \gamma_{a_N} \\ \gamma_{b_1} \\ \vdots \\ \gamma_{b_M} \end{pmatrix}}_{\begin{pmatrix} \gamma_a \\ \gamma_b \end{pmatrix}} = \underbrace{\begin{pmatrix} \mathrm{RHS}_{a_1} \\ \vdots \\ \mathrm{RHS}_{a_N} \\ \mathrm{RHS}_{b_1} \\ \vdots \\ \mathrm{RHS}_{b_M} \\ \Gamma_{\gamma,a} \\ \Gamma_{\gamma,b} \end{pmatrix}}_{\begin{pmatrix} \mathrm{RHS}_a \\ \mathrm{RHS}_b \\ \Gamma_{\gamma,a} \\ \Gamma_{\gamma,b} \end{pmatrix}} \tag{2.65}$$

where the diagonal matrices (C_{aa}, C_{bb}), are the self-induction matrix of the panel body, $a \to a$ and $b \to b$ respectively. The non-diagonal terms (C_{ab}, C_{ba}) are the inter-induction matrix containing the panel influence of $b \to a$ and $a \to b$ respectively. The final two rows of the LHS matrix contains the circulation constraint for each body.

2.6 Validation of Lagrangian method

During the validation study, we are will not be able to validate the vorticity generation of the Lagrangian boundary. This is due to the nature of the hybrid method that we are

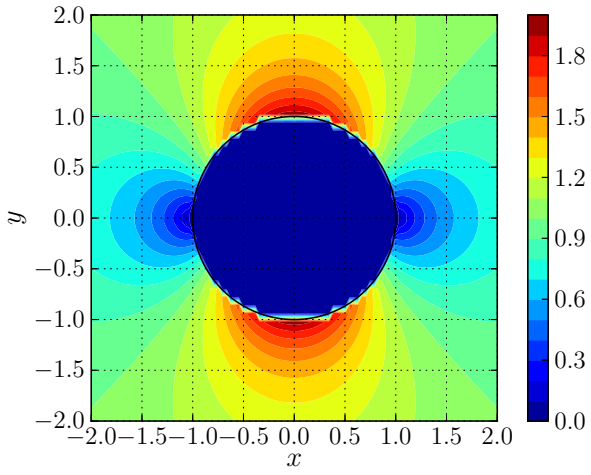


Figure 2.15: Panel method solution: the potential velocity field around a unit cylinder with $R = 1$, $\mathbf{u}_\infty = (1, 0)$, and $N_{\text{panels}} = 100$. The figure depicts the magnitude of velocity field $\|\mathbf{u}\|$, with a zero velocity inside the body.

using for this project. The generation of the vorticity from the boundary is defined in the Eulerian method, which is then interpolated onto the Lagrangian domain. Therefore, test cases that relies on the generation of the vorticity from the boundary cannot be used to validate our Lagrangian method. Thus the validation study for the Lagrangian method was done separately; first ensuring no-through flow on vortex panels is properly handled; second ensuring that the vorticity of the flow is properly handled by the vortex blobs.

2.6.1 Error analysis of panel method

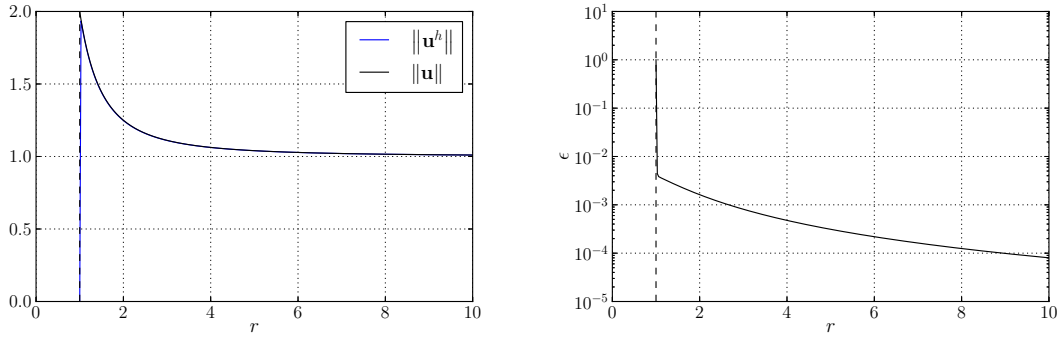
The validation of the panel method was done by performing a convergence study of a cylinder. The vortex panels where used to simulate a potential flow around a cylinder and the solution of the panels were compared with the analytical solutions.

To test the solution of the vortex panels with the analytical solution, the problem was first ran for the parameters in the table 2.1. The velocity field of the potential flow solution is shown in figure 2.15. The figure shows the magnitude of the velocity, and we see that it shows the velocity field of a 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, and for a higher sampling resolution, this will vanish. In order

Table 2.1: Panel study parameters

Parameters	Value	Description
R	1 m	Radius of cylinder
$\mathbf{u}_\infty$	1 m s^{-1}	Free-stream velocity
N_{panels}	100	Number of panels



(a) Comparison of the velocity field.

(b) Error in the velocity field

Figure 2.16: Comparison of the velocity field along the y-axis, $0 \to 10$. 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 2.67.

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 as,

$$u_r = u\left(1 - \frac{R^2}{r^2}\right)\cos\theta, \tag{2.66a}$$

$$u_\theta = -u\left(1 + \frac{R^2}{r^2}\right)\cos\theta, \tag{2.66b}$$

where u_r and u_θ are the radial and the angular velocity respectively. The equation 2.66 is a function of the radius from the center of the cylinder (in our case $r_0 = [0, 0]$) and the radius of the cylinder R.

The velocity field of the vortex panel was compared with this analytical solution along the y-axis from $y = 0 \to 10$. Figure 2.16a plots the magnitude of analytical velocity $\|\mathbf{u}\|$ and the vortex panel velocity field $\|\mathbf{u}^h\|$. 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 $2 \to 0$ at the surface.

The figure 2.16b shows the relative error ϵ between the numerical and the analytical solution,

$$\epsilon = \frac{\|\mathbf{u} - \mathbf{u}^h\|}{\|\mathbf{u}\|} \tag{2.67}$$

where $\mathbf{u}$ is the analytical solution and the $\mathbf{u}^h$ is the numerical (panel method) solution. Ignoring the solution right at the surface ($r = R$), we see that the error between the numerical and the analytical solution reduces from $\epsilon = 5 \times 10^{-3} \to 8 \times 10^{-5}$ as we go from $r = 1 \to 10$. 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

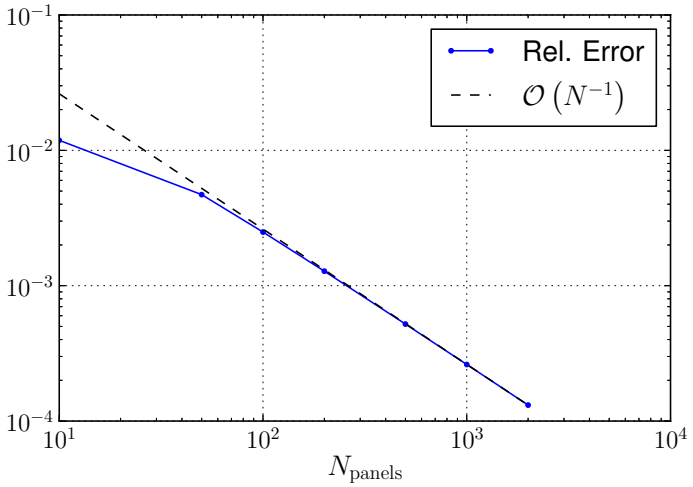


Figure 2.17: Convergence plot of the Constant-Strength Straight Vortex panels. The figures depicts the converges of the relative error ϵ at an $\mathcal{O}\left(N^{-1}\right)$.

next to the panels, we have the largest error. This is because the vortex panels discretizes the body using a first-order approximation (straight panels) and also discretizes the vortex sheet strength using a first-order approximation.

The convergence analysis of the vortex panels was done by determine the error of the vortex panel velocity field w.r.t to the analytical solution for the number of panels $N = 10 \rightarrow 1000$, figure 2.17. The error of the velocity field was determine at $R = 1.5$, and we see that the error converges at an $\mathcal{O}(N)$. This validates that the vortex panel that we have used is a 1^{st} order vortex panel.

2.6.2 Evolution of the 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 was used to compare against the analytical results, which we used to determine the accuracy of our vortex blobs.

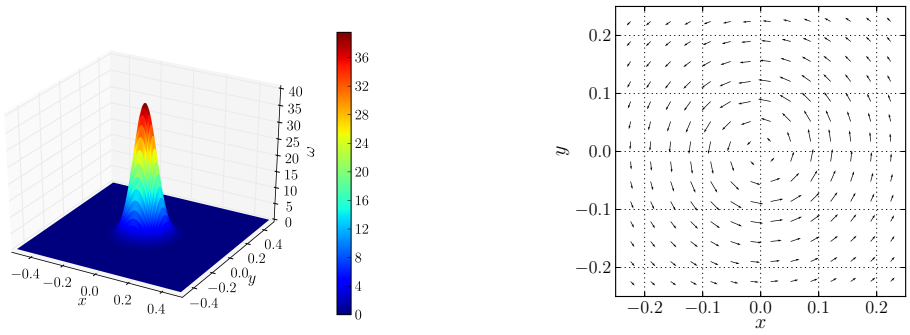
Lamb-Oseen vortex

The Lamb-Oseen vortex is a simple solution of unbounded laminar Navier-Stokes equation of a vortex core diffusing as time passes. It was first demonstrate by Lamb and Oseen [52]. The vorticity distribution ω of the core at a given time is defined as,

$$\omega(\mathbf{x}, \tau) = \frac{\Gamma_c}{4\pi\tau} \exp\left(-\frac{r^2}{4\tau}\right), \tag{2.68}$$

and is a function of core strength Γ_c, the scaled viscous time τ, and distance from the core center r. The scaled viscous time is defined as,

$$\tau \equiv \nu t. \tag{2.69}$$



(a) Vorticity field **(b)** Velocity Field

Figure 2.18: Lamb-Oseen Vortex problem with $\Gamma_c = 1$, $\tau = 2 \times 10^{-3}$, and $\nu = 5 \times 10^{-4}$. The figure depicts **(a)** the vorticity distribution, and **(b)** the velocity distribution.

At $\tau = 0$, the Lamb-Oseen vorticity distribution is a Dirac delta function, and the core starts to diffuse and expand as the time progresses. The velocity field of the Lamb-Oseen vortex is given as,

$$u_\theta = \frac{\Gamma_c}{2\pi r}\left[1 - \exp\left(-\frac{r^2}{4\tau}\right)\right] \tag{2.70a}$$

$$u_r = 0 \tag{2.70b}$$

where u_θ is the circumferential velocity, and u_r, the radial velocity is zero. Figure 2.18 shows the vorticity distribution, and the velocity field of the Lamb-Oseen vortex with a unit core strength $\Gamma_c = 1$, at scaled viscous time $\tau = 2 \times 10^{-3}$. As the time progresses, the core of the Lamb-Oseen vortex will diffuse and we use this analytical solution to valid our vortex blobs.

The lagrangian method ran the case of Lamb-Oseen vortex simulation according to the parameters tabulated in table 2.2. The problem was initialized by discretizing the vorticity field of the Lamb-Oseen vortex with $\Gamma_c = 1$, $\nu = 5 \times 10^{-4}$, and $\tau_0 = 2 \times 10^{-3}$, the initial scaled viscous time. The vorticity field was discretized over the domain $[x, y]$-domain $[-0.5, 0.5]^2$ using vortex blobs with $\sigma = 0.01$, and overlap $= 1$, giving it a fixed blob spacing of $h = 0.01$. We employed the standard initialization method of vortex blobs from section 2.2.4.

However as explained in section 2.2.4, we have to take in account of the gaussian blurring of the original vorticity field due to the initialization process. This posses a problem when evaluating the error between the numerical and the analytical solution. Barba [2], had also encountered the problem before when investigating with a Lamb-Oseen vortex. The solution to the problem was to apply a "time-shift correction", to compensate for the gaussian blurring, solving the problem of this very particular discretization of the diffusion 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 $k = 2$).

Table 2.2: Summary of the parameters for the Lamb-Oseen vortex evolution. Table shows the parameters of Tutty's diffusion method [53]

Parameters	Value	Unit	Description
Γ_c	1	$\mathrm{m^2\,s^{-1}}$	Core strength
Ω	$[-0.5, 0.5]^2$	m	Initial particle domain
$\mathbf{u}_\infty$	$[0, 0]$	$\mathrm{m\,s^{-1}}$	Free-stream velocity
ν	5×10^{-4}	$\mathrm{kg\,s^{-1}\,m^{-1}}$	Kinematic viscosity
(τ_0, τ_f)	2×10^{-3} to 2.5×10^{-3}	$\mathrm{m^2}$	Initial and final scaled viscous time
(t_0, t_f)	4 to 5	s	Initial and final simulation time
$\Delta t_c = \Delta t_d$	0.01	s	Diffusion and convection time step size
$N_{\mathrm{t-steps}}$	100	-	Number of time integration steps
σ	0.01	m	Vortex blob core size
overlap	1	-	Overlap ratio
k	2	-	Gaussian kernel width spreading
f_{redist}	1	-	Redistribution occurance
$f_{\mathrm{popControl}}$	1	-	Population control occurance
$\Gamma_{\mathrm{threshold}}$	$(1 \times 10^{-14}, 1 \times 10^{-14})$	$\mathrm{m^2\,s^{-1}}$	Population control threshold

Barba determined that 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 $\sigma^2/2\nu$. The initial particles strengths α_0 of vortex blobs from the Lamb-Oseen vorticity field is given as,

$$\alpha_0 = \omega_0 \cdot h^2 = \left\{ \frac{\Gamma_c}{4\pi\nu\,(t - \sigma^2/2\nu)} \exp\left[-\frac{r^2}{4\nu\,(t - \sigma^2/2\nu)} \right] \right\} \cdot h^2. \tag{2.71}$$

This method was used to investigate the error growth of the vortex blob method. The vortex blobs where convected and diffused according to the parameters in table 2.2. For the primary and the main investigation, we used diffusion method proposed by Tutty [53]. 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 behaviour in the evolution. We will see that when coupling the Lagrangian method and Eulerian method, discontinuity in the problem introduces additional errors.

So, using the Tutty's diffusion method (the simple redistribution scheme, section 2.4.1), we convect and diffuse the vortex blobs with a $\Delta t_c = \Delta t_d = 0.01$. The time integration of the problem is done using a 4^{th}-order Runge-Kutta method (RK4), ensuring accurate convection of the blobs. After the convection process, we will have to treat the Lagrangian distortion of the blobs using the remeshing scheme discussed in section 2.3.1. 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, $f_{\mathrm{redist}} = 1$.

In additional, after the evolution process, we can perform a population control on the vortex blobs to keep the simulation as efficient as possible. The Population Control (PC)

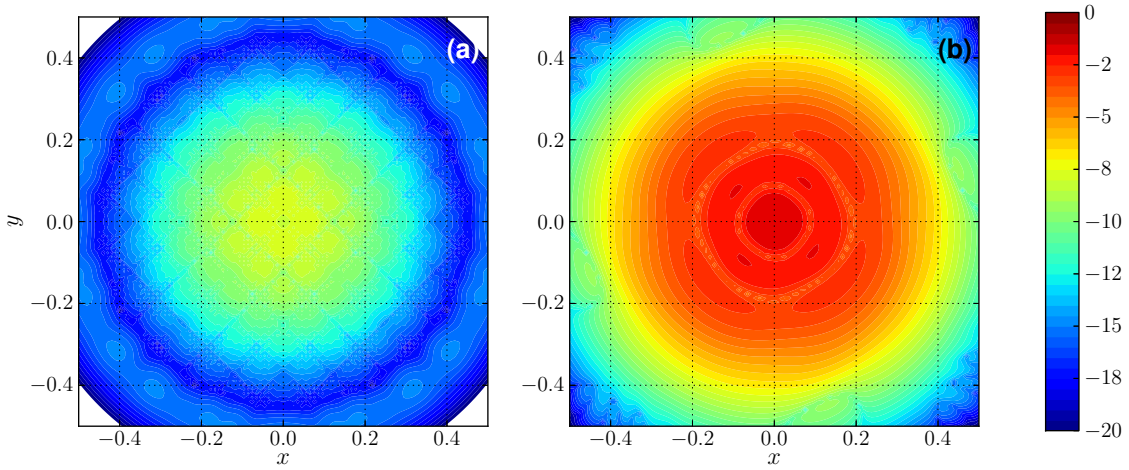


Figure 2.19: Relative error growth of Lamb-Oseen vorticity during the evolution (in logarithmic scale). The figure shows **(a)**, the initial relative error at $t_0 = 4$, and **(b)** the final relative error in vorticity at $t_f = 5$.

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 numerical precision, as they have minimal impact on the accuracy of the vorticity field, we can removed them. When performing the population control, we to ensure that the loss of total circulation is at minimal. We place a tolerance on the conservation of circulation, and from literature [2], we see that a cap of $\Gamma_{\text{threshold}} = 1 \times 10^{-14}$ is acceptable.

To verify whether our Lagrangian scheme is function according to theory, we evaluated the error growth of the simulation. Figure 2.19, 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 $t_0 = 4$ to $t_f = 5$, we see that the maximum relative error in vorticity increases from 10^{-8} to 10^{-2}. The error of the vorticity are predominantly localized at the center of the core. This is because, this is where we have the highest gradient in the vorticity, as seen from figure 2.18a. Therefore, with the current vortex blob discretization, we are not able to reconstruct this large gradient of the vorticity field.

Figure 2.20, shows the error growth of the Lamb-Oseen vortex from $t_0 = 4$ to $t_f = 5$. The figure plots the error growth of the vorticity and the velocity. Due to the relation of the vorticity and the velocity, the error of the vorticity is an order higher that the error in the velocity. The figure shows the both the maximum relative error, and the error in the $L^2 -$ norm. The maximum relative error (e.g. for vorticity), is determined as,

$$\left\| \omega^{\text{exact}} - \omega^{\text{discrete}} \right\|_{\infty} = \frac{\max\{\left| \omega^{\text{exact}} - \omega^{\text{discrete}} \right|\}}{\max\{|\omega^{\text{exact}}|\}}, \tag{2.72}$$

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

$$\left\| \omega^{\text{exact}} - \omega^{\text{discrete}} \right\|_{2} = \left(\sum_{i}^{N} \left| \omega^{\text{exact}} - \omega^{\text{discrete}} \right|^2 \cdot h^2 \right)^{\frac{1}{2}}, \tag{2.73}$$

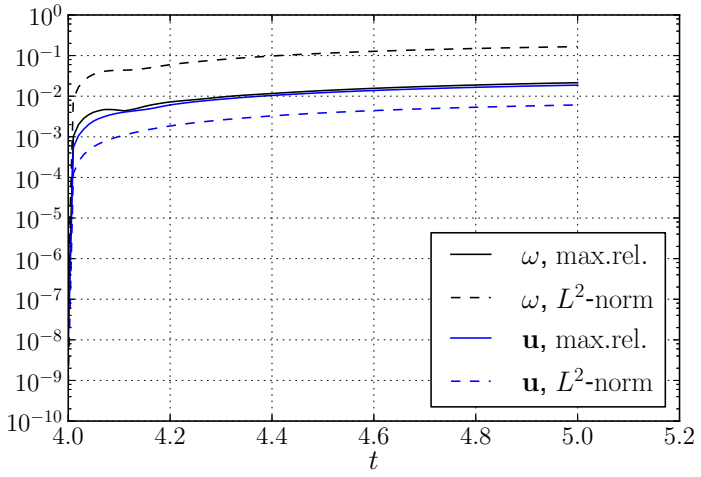


Figure 2.20: Relative error growth of Lamb-Oseen vortex during the evolution from $t_0 = 4$ to $t_f = 5$. Figure depicts the error in vorticity: maximum relative error [—, solid black], and error in $L^2 - \text{norm}$ [- -, dashed black]; and error in velocity: maximum relative error [—, solid blue], and error in $L^2 - \text{norm}$ [- -, dashed blue].

and the error in velocity is calculated using the same principle. Investigating the plot, we see that after the first iteration, there is sudden increase in the error, but as time progresses the error growth stabilizes. From literature, we see that this trend has also been observed by Barba [2] and Speck [49]. For comparison, we used similar parameters, and we observe that the sudden jump in error has been similar to the literature.

Comparison of diffusion schemes: SRS vs. MRS

To observe how both the diffusion scheme compare, we ran test cases with both diffusion schemes. From the simulation, we were able to observe that the Tutty's diffusion scheme, SRS, performed better that Wee's diffusion scheme, MRS. Figure 2.21 shows the error growth of maximum relative error in vorticity for both diffusion schemes. The solid lines represents the solution of the MRS and the dashed lines shows the SRS. The convection and the diffusion time steps where controlled by modifying the blob spacing h, and keeping the convection time step size $\Delta t_c = 0.01$.

At $h = 4 \times 10^{-3}$, both schemes have the diffusion time step $\Delta t_d = 0.01$ meaning that the diffusion occurs at every iteration $k_d = 1$. During this time we see that the MRS performs slightly better that the SRS scheme. This is because, the diffusion of the vortex blobs is directly incorporated into the remeshing process, whereas the Tutty's SRS segregates the diffusion and the remeshing into two subsequent process.

At $h = 5 \times 10^{-3}$, the diffusion constraint on the Wee's MRS changes the minimum allowable diffusion time step to $\Delta t_d = 0.02$, meaning that diffusion has to be performs at every second step, $k_d = 2$. Whereas for Tutty's SRS, we are not constraint by the minimum diffusion time step as so we can perform diffusion at $k_d = 1$. When investigating the growth in error for MRS, we see that the diffusion scheme has a large increase in the error, whereas the SRS does not have this limitation and is still able to diffuse at every step

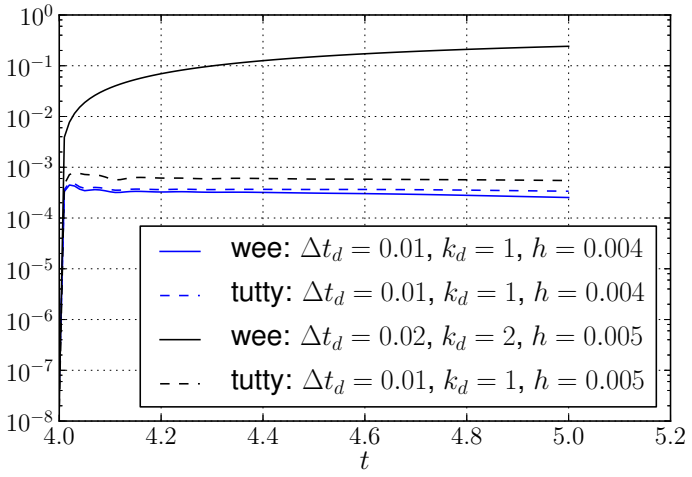


Figure 2.21: Comparison of Tutty's, simple redistribution scheme and Wee's modified interpolation method for treating diffusion. Figure depicts the growth in maximum relative error in vorticity from $t_0 = 4$ to $t_f = 5$ at $\Delta t_c = 0.01$. The Wee diffusion scheme with $\Delta t_d = \Delta t_c = 0.01$ [—, solid blue], and $\Delta t_d = 2\Delta t_c = 0.02$ [—, solid black]. The Tutty's diffusion scheme, $c^2 = 1/3$, with $\Delta t_d = \Delta t_c = 0.01$ [- -, dashed blue], and $\Delta t_d = \Delta t_c = 0.02$ [- -, dashed black].

and has only a slight increase in error, figure 2.21 (solid black line). The diffusion scheme over-estimates the diffusion, and results in an incorrect diffusion process. However, we see that Tutty's SRS still performs a stable diffusion (dashed black line).

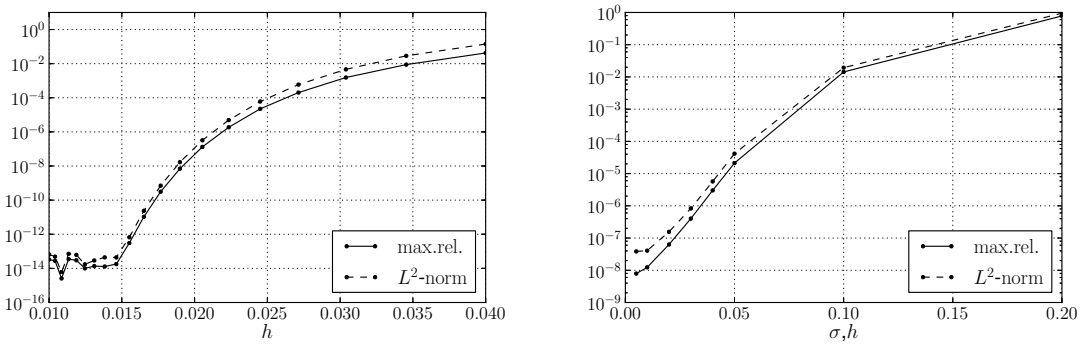
This verification of the diffusion scheme showed that if Wee's MRS diffuses at $k_d > 1$, it will over-estimate the diffusion of the vorticity. Therefore, when using this scheme we will have to ensure that $k_d = 1$, by adjusting the convective time step appropriately. However, for large Reynolds number flows this is not possible, and so we must rely on the more stable and versatile Tutty's SRS diffusion scheme.

2.6.3 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.

First, we investigate the convergence in spatial discretization. As we are dealing with vortex blobs, there are multiple ways of increasing the resolution. The straight forward method would be to increase the density of particles in a given area, i.e. reducing the blob spacing h and maintaining the core spreading σ. Figure 2.22a shows the convergence of the spatial discretization when the core size σ is maintained at $\sigma = 0.02$. At this case, the overlap ratio changes with the blob spacing, described by equation 2.17. At small blob spacing h, the error in vorticity quickly drops to near machine precision. When investigating the order of converges, we that the error converges in a non-linear fashion and similar results have been obtained by Barba [2].

Figure 2.22b, shows the convergence of error when the overlap ratio is fixed, overlap = 1.



(a) Error in vorticity vs. h with $\sigma = 0.02$ **(b)** Error in vorticity vs. h, σ with overlap $= 1$.

Figure 2.22: 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 is fixed.

In this test case, the core size scaled with the blob spacing, $h = \sigma$, and when increasing the spatial resolution, the error converges at non-linear fashion.

To investigate the convergence in temporal discretization, we determined the growth in Lamb-Oseen vorticity field after convecting with various convection step size Δt_c. Note that to perform this convergence study accurately, we had to rely on the Tutty's SRS diffusion scheme, so that $\Delta t_d = \Delta t_c$. Figure 2.23 shows the convergence of the error for various temporal discretization. With the combined convection and diffusion scheme, we see that error converges at $\mathcal{O}(\Delta t)$, meaning that the vortex blobs evolution is 1^{st}-order in time.

2.7 Summary of the Lagrangian method

In summary, we have investigated the Lagrangian domain 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,

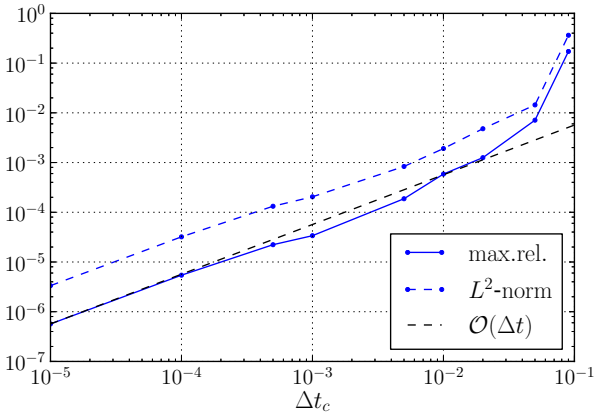


Figure 2.23: Error growth of Lamb-Oseen vorticity field after one-step.

as we require only to evolve the wake, and the generation of the vorticity is dealt with in the Eulerian domain. 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.

For the VPM, the choice of fluid elements was vortex blobs. This had non-zero core size, removing the singularity when performing Biot-Savart calculations. The strengths of these particles were initialized by assigning the local circulation strength to the particle, given by equation 2.18. When we perform coupling, we will see that the gaussian blurring of the original vorticity field during the initialization is the fundamental source of error. Strategies such as Beale's iterative method, cannot be used as it is defined for an unbounded domain. The only approach to minimize the gaussian spreading initialization error was to increase the overlap ratio to $overlap = 1$, and minimize the blob spacing h as much as possible. So, the proper handling of the initialization of the vortex blob strength is still an open question, and recommend that if solved can significantly improve the accuracy and the efficiency of the hybrid coupling. More on this will be discussed in section **??**. !!! citation !!!

The convection of the vortex blobs was done by using a 4^{th}-order Runge-Kutta time integration method. However, due to high strains in the fluid, we had to deal with the Lagrangian grid distortion of the vortex blob lattice. To deal with this, we used a M'_4 interpolation kernel that remeshed the particles onto a structured grid.

The diffusion of the vortex blobs was also import when simulation viscous flows. Initially, we employed the modified interpolation kernel by Wee [55], which integrated the diffusion process into the standard interpolation kernel. However, the method posed two problems: the diffusion time step Δt_d had a lower limit; the scheme over-estimated the diffusion when the diffusion did not occur at every step, $k_d > 1$. To overcome this problem, we used the simple redistribution scheme by Tutty [53], which enabled us to perform diffusion after every convection step. This also ensured that the diffusion process was continuous, which was important when performing the coupling algorithm.

To deal the body boundary in the flow, we used the vortex panels to enforce the no-through/no-slip boundary condition. Koumoutsakos, Leonard, and Pepin [33], have shown that enforcing no-slip boundary condition is equivalent to a no-through boundary condition, as they are linked. Constant-Strength Vortex panels, based of Katz [30], were used to treat the body boundary condition. This panel method was then verified and validated with the analytical solution of a potential flow around a cylinder. There was one difference between our vortex panel method and the standard boundary element method that is used in a typical VPM. Our panel method cannot be used to determine the vorticity generation at the body, as this is done in our Eulerian method. So we validated our Lagrangian method separately; first ensuring that the no-through boundary condition is satisfied at the vortex panels, second ensuring that the vorticity of the flow is handled properly by the vortex blobs.

To validate the vortex blob, we used the analytical solution of the Lamb-Oseen vortex problem. We verified that the Wee's MRS diffusion method indeed did not perform optimally for all parameters. The validation was done by investigating the error in the

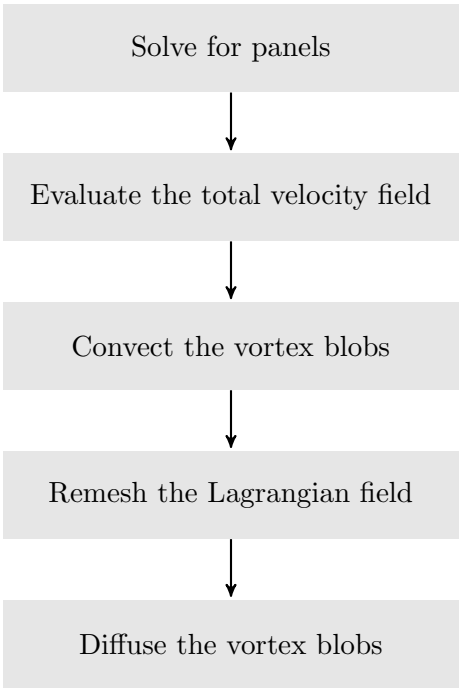


Figure 2.24: Flowchart of the Lagrangian method. The flowchart shows coupling between vortex panels and vortex blobs to evolve from t_n to t_{n+1}.

vorticity and the velocity field and was able to conclude that the vortex blobs performed according to literature of Barba [2].

Lagrangian method algorithm

The flowchart to one time step of the Lagrangian method is given by figure 2.24.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 satisfied the conservation of circulation, equation 2.50.
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 time step the vortex blobs from t_n to t_{n+1} to the new position with a convection time step Δt_c.
4. **Remesh the Lagrangian field:** Remesh the vortex blobs onto a structured square lattices 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 MRS or Tutty's SRS method.

The generation of the vorticity is dealt with in the Eulerian domain, which is explained in chapter 3. The vorticity is then transfered into the Lagrangian domain using the Hybrid coupling scheme, which was summarized in the introduction, chapter 1, and full elaborated in chapter 4.

Chapter 3

Eulerian Domain: Finite Element Method

Standard Computation Fluid Dynamics (CFD) method discretizes the fluid into smaller regions, known as grids, and solves the set of Navier-Stokes equations in this region. This type of formulation is referred to as an Eulerian method, as we are evaluating the change of flow property in a given volume.

For the hybrid method, we use the Navier-Stokes grid formulation in the near-body region. The advantage of using the Eulerian method at this region is that it is much more efficient in resolving the boundary layer than the Vortex Particle Method. We can directly enforce the wall boundary condition at the wall boundary of the Eulerian domain, solving the problem of vorticity generation of the body. In the hybrid coupling strategy, we can then interpolate this newly resolved near-wall solution on to the Lagrangian domain, where the vortex blobs can efficiently evolve the particles.

The various approaches to solve the fluid dynamics problem from a Eulerian reference frame. Finite Volume Method (FVM), Finite Difference Method (FDM), and Finite Element Method (FEM) are the common choice for solving the Navier-Stokes problem and differ by the way they approach to solve the problem. FVM divides the domain into volumes where it enforces the conservation of mass and momentum in each sub-domains. FDM divides the domain into nodes and use local Taylor expansions to approximate the partial differential equations. FEM divides the domain into elements and solves the problem using variational calculus. So in the end, the choice of Eulerian method does not have a direct impact on the coupling with the Lagrangian method as the purpose of the Eulerian method is only to efficiently, and accurately resolve the near-body region of the body.

We have decided to use the FEM packages provided by the FEniCS project as they have be already implemented efficient, multi-threaded algorithms for solving the partial differential equation. Furthermore, they provide extensive features for future developments such as adaptive mesh refinement, fluid-structure interaction, and efficient computation of turbulent flow.

3.1 Introduction to Finite Element Method

Finite Element Method (FEM) is numerical method to solve for the solution of a given partial differential equation. It is solved by describing it as a variation problem, giving us an approximate solution for the boundary value problem [6]. So the FEM approximates the unknown variables and converts the partial differential equations to a set of algebraic equations, which makes them easier to solve. It was traditionally used for solid mechanics (e.g for the analysis of aircraft structures [42]), but have since then used to solve fluid dynamics problems [26] [29] [27].

Finite element discretization

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

A finite element discretization in 2-D can be seen in figure 3.1. The figures 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 finite number of cells $\mathcal{T}_h = \{T\}$ of the fluid domain Ω, together makes the mesh of the Eulerian domain. As shown in figure 3.1, the cells of the finite element in 2-D, are made of simple geometrical shapes such as triangles or quadrilaterals. There are two approaches to discretize the domain: structured or unstructured mesh. The structured mesh has cells oriented in a structured pattern, and is the simplest approach in discretizing 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 computation. The downside to such discretization is that the mesh quality deteriorates as one increases the complexity of the domain. However, the FEM enables us to perform an unstructured discretization of the domain, as shown in figure 3.2. The figure shows

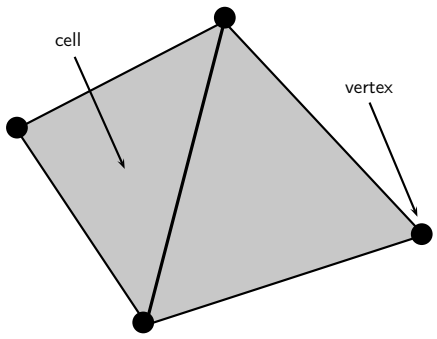
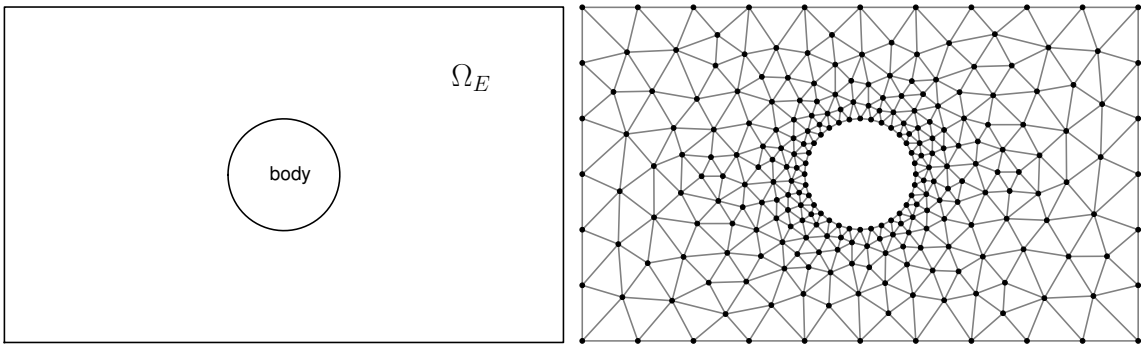


Figure 3.1: A two-dimensional finite element geometry. The cell represents the area of the element, and vertices are the edges of the cell.



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

Figure 3.2: Delaunay triangulation of the fluid around a cylinder resulting in unstructured mesh with controllable cell sizes.

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. This shows that even though the unstructured method formulation is more complicated that the structured formulation, we have the advantage that the mesh quality does not deteriorate as the domain becomes more complex.

There are several algorithms for mesh generation. The standard approach is to employ the Delaunay triangulation method derived from the Voronoi diagram concept [8]. This divides the domain into a set of triangles, as shown in figure 3.2. This type of mesh generation allows us to connect different shapes of boundary with each other. Furthermore, this triangulation method be controlled by predefining the boundary element nodes using a transfinite interpolation.

Finite element function and function space

The finite element is defined using a triple $(T, \mathcal{V}, \mathcal{L})$, as defined Ciarlet [13] and used by the FEniCS Project [37]. The domain Ω is divided into cells T, 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 $\mathcal{V}$.

When the domain Ω is divided into cells T, we can define the function and the function space of the finite element problem. For each cell, a local function space $\mathcal{V}$ can be defined to collectively construct the global function space V. Any given function $u \in V$ is a expressed in a linear combination of basis functions $\{\phi_1, \phi_2, ..., \phi_N\}$, of the function space V,

$$u(x) = \sum_{j=1}^{N} U_j \phi_j(x). \tag{3.1}$$

There are several types of finite element families: the Brezzi-Douglas-Marini, the Crouzeiz-Raviart, the Discontinuous Lagrange, the Hermite, and the Lagrange elements [37]. Each has its own advantage such as the Discontinous Lagrange, or Discontinous Galerkin (DG) element consists of discontinuous functions, which was originally introduced for solving hyperbolic problem by Reed and Hill in 1973 [43]. The method was able to conserve

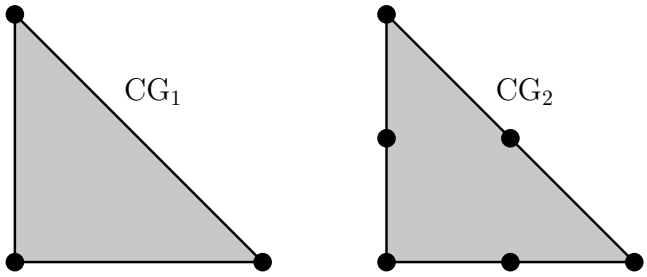


Figure 3.3: The Lagrange CG_q triangle for $q = 1, 2$. The triangles have 3 and 6 DOFs respectively (•, black dot).

mass at each element, had a high-order accuracy, and was robust in solving the advection problem. However for the current problem, we will rely on the Lagrange elements, also known as the Continuous Galerkin (CG), which are based on the Lagrange polynomials [10]. These elements are widely used and are the simplest to implement for our project.

Lagrange elements belong to the space H^1, which a Sobolev space containing functions u such that u^2 and $|\nabla u|^2$ have finite integral in the domain Ω [37]. The Lagrange element uses point evaluation for the degrees of the freedom, where a DOF in (x, y) denotes the point evaluation of the function u, $\ell(u) = u(x, y)$. We can have a Lagrange elements of various orders $q = 1, 2, ...$, where q is the degree of the Lagrange polynomial $\mathcal{P}_q$ on the domain at T. For the 2-D case, the dimension n of the finite element is given as,

$$n(q) = \frac{1}{2}(q + 1)(q + 2). \tag{3.2}$$

For $q = 1$, we have a simple Lagrange element CG_1, known as the Courant triange [19], with 3 DOFs. For a higher order finite element, we can set $q = 2$, giving us a Lagrange element CG_2 with 6 DOFs per cell. Figure 3.3 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. To describe our Eulerian problem of our hybrid scheme, we will rely on the CG_1 and CG_2 Lagrange elements.

Variational formulation

To solve a basic problem such as a Poisson equation numerically, we need to convert it into a variational problem. The methodology is followed from the FEniCS tutorial provide by Langtangen [37]. A 1D Poisson problem is given as,

$$\begin{aligned} -\nabla^2 u(x) &= f(x), \qquad x \text{ in } \Omega, \\ u(x) &= u_0(x), \qquad x \text{ on } \partial\Omega. \end{aligned} \tag{3.3}$$

We can transform equation 3.3 into a variational form by multiplying it with a test function v, and integrating it over the domain Ω,

$$-\int_\Omega \left(\nabla^2 u\right) v \, \mathrm{d}x = \int_\Omega f v \, \mathrm{d}x, \qquad \forall \; v \in \hat{V}. \tag{3.4}$$

In variational form equation 3.4, the function u is known as 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,

$$-\int_{\Omega} \nabla u \nabla v \, \mathrm{d}x = \int_{\Omega} fv \, \mathrm{d}x \qquad \forall \; v \in \hat{V} \tag{3.5}$$

This form is 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 function f and g in domain *Omega* is defined as,

$$\langle f, g \rangle = \int_{\Omega} fg \, \mathrm{d}x, \tag{3.6}$$

so we can simplify equation 3.5 to,

$$\langle \nabla u, \nabla v \rangle = \langle f, v \rangle, \qquad \forall \; v \in \hat{V}. \tag{3.7}$$

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

$$-\langle \nabla u_h, \nabla v \rangle = \langle f, v \rangle \qquad \forall \; v \in \hat{V}_h \subset \hat{V}, \tag{3.8}$$

where u_h is the discrete function in the discrete space V_h which is a subset of V, and the discrete function space $\hat{V}_h$ is a subset of $\hat{V}$. A common choice for the function space is linear triangular element with three nodes (Courant triangle), as shown in figure 3.3, 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 is zero, whereas the functions in the trial space is equal the boundary condition u_0. The equation 3.8 can be simplified as,

$$a\,(u, v) = L(v), \tag{3.9}$$

where,

$$a\,(u, v) = -\langle \nabla u, \nabla v \rangle, \tag{3.10}$$

and

$$L(v) = \langle f, v \rangle. \tag{3.11}$$

The variable $a(u, v)$ and $L(v)$ is the denoted as the bilinear and linear form, respectively. For simplicity, we will ignore the discrete notation (i.e $\{\cdot\}_h \rightarrow \{\cdot\}$). To solve for the discrete solution we substitute,

$$u = \sum_{j=1}^{N} U_j \phi_j, \tag{3.12}$$

the linear combination of the basis function ϕ_j, spanning the function space V, into $a\,(u, v)$. The test function is linear combination of the basis function $\hat{\phi}_i$, spanning the test space $\hat{V}$, which is defined as

$$v = \sum_{i=1}^{N} \hat{\phi}_i. \tag{3.13}$$

The test function v is taken to be zero at the boundary and one everywhere else. Substituting equation 3.12 and 3.13 into equation 3.9 gives,

$$\sum_{j=1}^{N} a(\phi, \hat{\phi}_i)\ U_j = L(\hat{\phi}_i). \tag{3.14}$$

Thus, we have to solve a linear system of equations given as,

$$\mathbf{A}U = b, \tag{3.15}$$

where $\mathbf{A}_{ij} = a(\phi_j, \hat{\phi}_i)$ is the coefficient matrix, and b is the Right-Hand Side (RHS) containing the knowns of the problem.

3.2 Solving the Finite Element problem

To solve this linear system of equations, equation 3.15, we used the FEniCS Project that had implemented a comprehensive library of finite elements,, high performance linear algebra, and a scripting interface in Python. The scripting interface in Python enabled us to perform computation similar to a Matlab interface, where we could focus on the development of the algorithm, with minimal time spend on the code generation and the debugging process. The Dolfin library from the FEniCS Project was used in python to define the finite element problem.

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 tools.

3.2.1 Introduction to FEniCS Project

The FEniCS Project is a collaborative work of various universities, that developed tools to perform automated finite element algorithms that can be used to solve the solutions of the partial differential equation. It was a project originated in 2003 with the research collaboration of University of Chicago and Chalmers University of Technology with Logg, Mardal, and Wells [37]. Since then, it has been expanded to various institutes such as Royal Institute of Technology, Simula Research Laboratory, University of Cambridge, and Delft University of Technology.

The consists of various libraries such as UFC, UFL, FIAT, INSTANT and mainly Dolfin. Dolfin is the core library aimed at automating the solution of partial differential equations using finite element method [38]. It uses automated code generation maintaining high level of mathematical expressions and internally providing efficient, multi-threaded performance using the Message Passing Interface (MPI). It used built-in linear algebra backend such as PETSc, TRILINOS/EPECTRA, uBLAS, and MTL4.

We used the Dolfin library in Python to solve the Poisson problem, equation 3.3. We took $f = 2$ and,

$$u(x) = u_0(x) = \sin x \cdot \cos y, \tag{3.16}$$

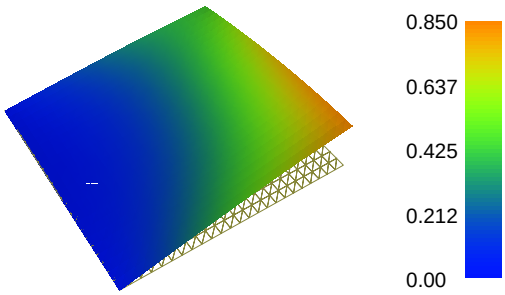


Figure 3.4: DOLFIN VTK plot of the Poisson solution, given by the problem, source code listing 3.1.

```
1  from dolfin import *
2
3  # Generate unit square mesh: 24 × 24
4  mesh = UnitSquareMesh(24, 24)
5
6  # Define Function space: 1st order, Continuous-Galerkin
7  V = FunctionSpace(mesh,"CG",1)
8
9  # Define boundary conditions
10 # u_0 = sin x · cos y
11 u0 = Expression("sin(x[0])*cos(x[1])")
12
13 def u0_boundary(x, on_boundary):
14     return on_boundary
15
16 # Define the boundary condition
17 # u(x) = u_0(x), x on ∂Ω
18 bc = DirichletBC(V, u0, u0_boundary)
19
20 # Define the variational problem
21 u = TrialFunction(V)  # Trial function
22 v = TestFunction(V)   # Test function
23 f = Constant(2.)      # f = 2
24 a = -inner(nabla_grad(u), nabla_grad(v))*dx # LHS: a = −∫ ∇u∇v dx
25 L = f*v*dx            # RHS: L = ∫ fv dx
26
27 # Solve the Poisson problem
28 u = Function(V)       # Define the solution
29 solve(a == L, u, bc)  # a(u, v) = L(v)
30
31 # Plot the result
32 plot(u)
```

Listing 3.1: 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 $f = 2$. The code is written in PYTHON using DOLFIN 1.2 library

on boundary $\partial\Omega$. The code generation of the finite element was automated with the DOLFIN library, leaving only the explicit expression of the problem in python. The source code to the Poisson problem is listed in listing 3.1, and we see that it is a direct explicit translation of the problem, with small overheads of constructing the finite element problem.

3.2.2 Mesh generation using GMSH

The proper generation of the fluid mesh is an important aspect of the Finite Element method. It is an important process, as an ill-construct mesh can be computationally very expensive, or might even problem with convergence. There have been literatures dedicated just to improve the mesh generation, for example by Hansen [28]. It focused on mesh enhancement techniques for elliptical methods, which enables to increase the quality of the data, and also the robustness of the simulation.

For the generation of the mesh, we use GMSH , an open-source software developed by Geuzaine and Remacle [23], which has implemented a user-friendly interface and fast algorithms. The GMSH implemented kernels that 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.

3.3 Solving Incompressible Navier-Stokes Equations

The finite element method will be used to describe the Eulerian domain of the hybrid scheme. In the Lagrangian domain, we have used the vorticity-velocity $\omega - \mathbf{u}$ formulation of the Navier-Stokes to describe the evolution of the vorticity in the wake. However, in the Eulerian domain, we will use the primitive variables, the velocity-pressure $\mathbf{u} - p$ formulation of the fluid, so that we can directly enforce the no-slip Dirichlet velocity boundary condition at the wall of the body.

3.3.1 Velocity-pressure formulation

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

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \nabla \cdot \sigma = f, \tag{3.17a}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{3.17b}$$

where σ is the Cauchy stress tensor defined as,

$$\sigma(\mathbf{u}, p) = 2\nu\epsilon(\mathbf{u}) - p\mathbf{I}. \tag{3.18}$$

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

$$\epsilon(\mathbf{u}) = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathrm{T}}\right). \tag{3.19}$$

```
# Define the trial and test function
omega = TrialFunction(W)
v = TestFunction(W)

# Define the variation problem for vorticity
a = inner(omega,v)*dx    #  ⟨ω, v⟩
b = inner(curl(u),v)*dx  #  ⟨∇×u, v⟩

# Pre-Assemble the LHS
A = assemble(a)

...

# During the time-stepping
omega = Function(W) # Define the function
B = assemble(b)     # Assemble b
solve(A, omega.vector(), B) # Solve for vorticity
```

Listing 3.2: The PYTHON implementation of the vorticity calculation

So the Cauchy stress tensor defines the stress at a point in fluid due to the velocity gradient and the pressure. The incompressible 2-D 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. Onces we solve for both these primitive variables, we can transfer this solution on the Lagrangian domain using the hybrid scheme. The hybrid scheme that we employ for this research, requires the transfer of the vorticity field from the Eulerian domain (i.e Eulerian grid), onto the vortex particles of the Lagrangian domain.

3.3.2 Determining the vorticity field

The coupling between the Eulerian to Lagrangian method is through the transfer of the vorticity field ω from the Eulerian domain to the Lagrangian vortex blobs. The vorticity field ω, is defined as,

$$\omega = \nabla \times u, \tag{3.20}$$

and is defined as the curl of the velocity field $\mathbf{u}$. The vorticity field ω is a scalar field that lies on the scalar-valued function space W. As the velocity field changes every at time t_n, we will have to evaluate the curl of the velocity at every step, $t_n, t_{n+1}, ...$ This brings the need for calculating the vorticity in an efficient manner, as we should not have to reconstruct the problem every step. To solve this problem in a efficient manner, FENICS has implemented a function that can pre-assemble (i.e pre-calculate) the fixed parameters of the problem. To do this, we must first define equation 3.20 in the variational (integral) form,

$$\int_\Omega \omega \cdot \mathbf{v} \, \mathrm{d}x = \int_\Omega (\nabla \times \mathbf{u},) \cdot \mathbf{v} \, \mathrm{d}x, \tag{3.21}$$

where $\omega = \sum_{j=1}^{N} \hat{\omega}_j \psi_j$, is a linear combination of basis function ψ_j, spanning the function space W. The variational form is summarized as

$$a(\omega, \mathbf{v}) = L(\mathbf{u}, \mathbf{v}) \tag{3.22}$$

where $a(\omega, v)$ contains the knowns of the problem, which are fixed during the simulation. This can be pre-calculated to optimize the problem. $(\mathbf{u}, \mathbf{v})$ is the unknown of the problem which has to be recalculated every time as it is a function of the current velocity. The PYTHON implementation of the algorithm is show in listing 3.2. Using the DOLFIN library, we can used the `assemble` function to pre-calculated the LHS of the problem (line 10). So using the algorithms of the hybrid coupling scheme, we can transfer this vorticity field of the Eulerian domain on the vortex blobs.

3.3.3 Taylor-Hood finite element family for solving ICNS

To solve the Incompressible Navier-Stokes (ICNS) problem, we must choose an appropriate finite element function spaces for the velocity $\mathbf{u}$, the pressure p, and the vorticity. When determining proper mix of finite elements, we have to ensure that it satisfies the Ladyzhenskaya-Babuška-Brezzi (LBB) compatibility condition, also known as the inf-sup compatibility condition [7]. When using the Lagrange finite element spaces, we have to ensure that the order of the velocity q_{vel}, is one order higher than the order of the pressure q_{pres},

$$q_{\text{vel}} = q_{\text{pres}} + 1. \tag{3.23}$$

Brezzi and Fortin [7] showed that if both are of same order, it will result to an unstable problem. To solve the ICNS problem, we will used the Taylor-Hood family [51], examined by Boffi [4]. The method use velocity order $q_{\text{vel}} = 2$ and pressure order $q_{\text{pres}} = 1$. We decided to choose this method, as it is the most conventional method, that is simple, and that shows a stable behavior.

In addition, we have to choose an appropriate function space for the vorticity field Ω. As vorticity is the curl of the velocity, to reduce interpolation error during the projection of the solution, we will used function space one order lower than the velocity, $q_{\text{vort}} = 1$. Table 3.1 shows the list of the function spaces, the finite element type and the order. In additional, 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 3.1: 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 also 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

3.3.4 Incremental pressure correction scheme

The algorithm to solve the Navier-Stokes problem was first demonstrated by Chorin in 1968 [11], referred to as the Chorin's projection method or sometimes known as the non-incremental pressure correction scheme. The process relied on first computing a tentative velocity by initially neglecting the pressure in the momentum equation of the Navier-Stokes problem, equation 3.17. 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 [24], 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 [37].

The detailed algorithms to the IPCS scheme, as demonstrated by the FEniCS Project [37], can be summarized as follows:

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

$$\begin{aligned}\langle D_t^n \mathbf{u}^\star, \mathbf{v}\rangle + \langle \mathbf{u}^{n-1} \cdot \nabla \mathbf{u}^{n-1}, \mathbf{v}\rangle + \langle \sigma(\mathbf{u}^{n-\frac{1}{2}}, p^{n-1}), \epsilon(\mathbf{v})\rangle \\ + \langle p^{n-1}\hat{\mathbf{n}}, \mathbf{v}\rangle_{\partial\Omega} - \langle \mathbf{v}\ \hat{\mathbf{n}} \cdot (\nabla \mathbf{u}^{n-\frac{1}{2}})^{\mathrm{T}}, \mathbf{v}\rangle_{\partial\Omega} = \langle f^n, \mathbf{v}\rangle,\end{aligned} \tag{3.24}$$

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

$$\mathbf{u}^{n-\frac{1}{2}} = \frac{\mathbf{u}^\star + \mathbf{u}^{n-1}}{2}, \tag{3.25}$$

   With the Dirichlet velocity boundary conditions at the boundary $\partial\Omega$, we can solve the equation 3.24. The additional term,

$$\langle \mathbf{v}\ \hat{\mathbf{n}} \cdot (\nabla \mathbf{u}^{n-\frac{1}{2}})^{\mathrm{T}}, \mathbf{v}\rangle_{\partial\Omega}, \tag{3.26}$$

   is resulted from the 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.

   The source code for solving the tentative velocity problem is shown in listing 3.3. First, we pre-define all the terms need 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, as it remains constant. During the time integration, we first assemble the RHS of the problem (line 26), then apply the Dirichlet velocity boundary condition (line 29) which consist of the wall boundary condition, and external Dirichlet velocity boundary condition (e.g. the free-stream). Finally, we can solve the problem using GMRES solver for solving the system of linear equations.

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

$$\langle \nabla p^n, \nabla q\rangle = \langle \nabla p^{n-1}, \nabla q\rangle - \langle \nabla \cdot \mathbf{u}^\star, q\rangle / \Delta t_n \tag{3.27}$$

```
# Before the time-stepping:

# Define: u^{n-1/2} = (u* + u^{n-1})/2
U  = 0.5*(u0 + u)

# Formulate the tentative velocity problem
F1 = (1/k)*inner(v, u - u0)*dx \
             + inner(v, grad(u0)*u0)*dx \
             + inner(epsilon(v), sigma(U, p0, nu))*dx \
             + inner(v, p0*n)*ds \
             - beta*nu*inner(grad(U).T*n,v)*ds \
             - inner(v, f)*dx

# Extract the LHS, and the RHS
a1 = lhs(F1)
L1 = rhs(F1)

# Pre-assemble the LHS
A1 = assemble(a1)

...

# During the time-stepping:

# Assemble the RHS
b = assemble(L1)

# Apply the Dirichlet velocity boundary condition b.c
[bc.apply(A1, b) for bc in bcVelocity]

# Solve for the Tentative velocity
solve(A1, u1.vector(), b, "gmres", "default")
```

Listing 3.3: The source code for solving the tentative velocity $\mathbf{u}^\star$, using the equation 3.24

is valid for all $q \in Q$. We use the previously calculated tentative velocity $\mathbf{u}^\star$ to determine the corrected 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 were 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 domain. This means that do not have to assume any pressure boundary condition.

The source code for solving the corrected pressure problem is shown in listing 3.4. As done for the tentative velocity, we formulate and pre-assemble the problems before the time loop. In the time loop, we only need to assemble the RHS, apply the boundary condition (if it exists), and finally solve for the corrected pressure. Using the corrected pressure, we can determine the corrected velocity field.

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

$$\langle \mathbf{u}^n, \mathbf{v} \rangle = \langle \mathbf{u}^\star, \mathbf{v} \rangle - \Delta t_n \langle \nabla (p^n - p^{n-1}), \mathbf{v} \rangle, \tag{3.28}$$

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.

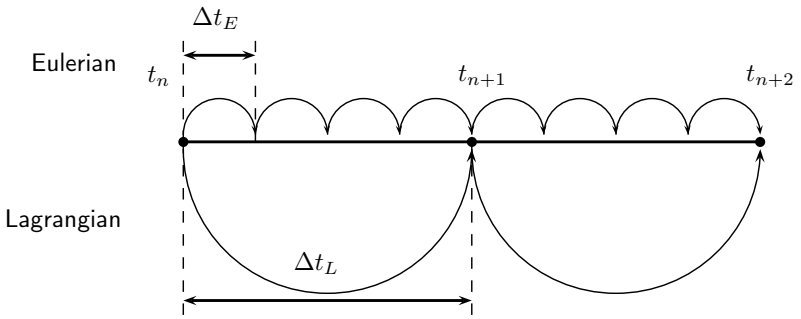


Figure 3.5: Eulerian multi-stepping to match the lagrangian Δt_L. The figures shows $\Delta t_L = 4\Delta t_E$ and required $k_E = 4$ iterations to time march from t_n to t_{n+1}.

The source code of the solving the corrected pressure problem in shown in listing 3.5. 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, apply the velocity boundary condition and finally solve for the corrected velocity field.

This algorithm was implemented using DOLFIN's Krylov GMRES solver with absolute and relative error tolerance of 10^{-25} and 10^{-12} respectively. The program structure was based on the collection of benchmark and solvers provided by the FENICS examples scripts [36]. The algorithm described above an explicit time marching scheme, also referred to as Forward Euler (FE), which is the simplest time marching scheme. Therefore, for the time marching scheme to be stable, we require the CFL number satisfy the following condition:

$$\mathrm{CFL} = \Delta t_{\max} \frac{\|\mathbf{u}\|_{\max}(\nu + \Delta h_{\min}\|\mathbf{u}\|_{\max})}{\Delta h_{\min}^2} \leq 1. \tag{3.29}$$

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}. When coupling with the Lagrangian method, we will see that $\Delta t_E \leq \Delta t_L$ (Lagrangian time step size is ideally larger than Eulerian time step size), meaning that we will have to perform k_E Eulerian sub-steps to reach the Lagrangian step, figure 3.5.

3.3.5 Determining the body forces

Once we solved for flow fields, it is a common strategy to verify the results using some sort of quantities. In aerodynamics, (especially in aerospace field), we like the evaluated the lift and drag coefficient of a given geometry and compare the results to available literature. In order to determine these coefficient, we must first determines the friction force and the pressure force acting on the no-slip boundary, or simply put, it is the total stress tensor σ acting on the surface of the body. The stress tensor σ is given by

$$\sigma(\mathbf{u}, p) = 2\nu\epsilon(\mathbf{u}) - p\mathbf{I}, \tag{3.30}$$

```
# Before the time-stepping:

# Formulate the pressure correction problem
a2 = inner(grad(q), grad(p))*dx          # ⟨∇q, ∇p^n⟩
L2 = inner(grad(q), grad(p0))*dx\        # ⟨∇ q, ∇ p^{n-1} ⟩ − ⟨ ∇ · u*, q⟩/Δ t_n
          - (1/k)*q*div(u1)*dx

# Pre-assemble the LHS
A2 = assemble(a2)

...

# During the time-stepping:

# Assemble the RHS
b = assemble(L2)

# Apply the Dirichlet velocity boundary condition b.c
if len(bcPressure) == 0: normalize(b)
[bc.apply(A2, b) for bc in bcPressure]

# Solve for the corrected pressure
solve(A2, p1.vector(), b)
if len(bcPressure) == 0: normalize(p1.vector())
```

Listing 3.4: The source code for solving the corrected pressure p^n using the equation 3.27

```
# 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 3.5: The source code for solving the corrected pressure p^n using the equation 3.27

```
...

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 3.6: The PYTHON implementation of the force calculation

where ϵ is the symmetric gradient, equation 3.19, and is a function of the velocity $\mathbf{u}$ and the pressure p acting on the surface.The lift coefficient and the drag coefficient is computed as,

$$L = \int_{\partial\Omega} [\sigma(\mathbf{u}, p) \cdot \hat{\mathbf{n}}] \cdot \hat{\mathbf{e}}_y \, \mathrm{d}s, \tag{3.31a}$$

$$D = \int_{\partial\Omega} [\sigma(\mathbf{u}, p) \cdot \hat{\mathbf{n}}] \cdot \hat{\mathbf{e}}_x \, \mathrm{d}s, \tag{3.31b}$$

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

$$\hat{\mathbf{e}}_x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \hat{\mathbf{e}}_y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{3.32}$$

The lift coefficient and the drag coefficient, C_l and C_d respectively, is the lift and drag normalized 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,

$$C_l = \frac{L}{\frac{1}{2}\|\mathbf{u}\|_\infty^2 c}, \qquad C_d = \frac{D}{\frac{1}{2}\|\mathbf{u}\|_\infty^2 c}. \tag{3.33}$$

Table 3.2: Summary of the parameters for the Lamb-Oseen vortex evolution.

Parameters	Value	Unit	Description
Γ_c	1	$\mathrm{m^2\,s^{-1}}$	Core strength
Ω	$[-1,1]^2$	m	Eulerian domain bounds
$\mathbf{u}_\infty$	$[0,0]$	$\mathrm{m\,s^{-1}}$	Free-stream velocity
ν	5×10^{-4}	$\mathrm{kg\,s^{-1}\,m^{-1}}$	Kinematic viscosity
(τ_0,τ_f)	2×10^{-3} to 2.5×10^{-3}	$\mathrm{m^2}$	Initial and final scaled viscous time
(t_0,t_f)	4 to 5	s	Initial and final simulation time
$h_{\min}$	$\frac{1}{100}\sqrt{2}$	m	Minimum mesh cell size
N_{cells}	200^2	-	Number of mesh cells
CFL	0.95	-	CFL number
$\|\mathbf{u}\|_{\max}$	1.5	$\mathrm{m\,s^{-1}}$	Maximum magnitude fo the velocity
Δt	0.001	s	Time step size
$N_{\text{t-steps}}$	1000	-	Number of time integration steps
ID_{fluid}	1	-	Fluid domain I.D
ID_{ext}	3	-	External Dirichlet velocity boundary I.D

3.4 Validation of eulerian method

To validate our Eulerian method, we will first investigate the problem of the Lamb-Oseen vortex. Then we will compare the results of the Clercx-Bruneau dipole collison at $Re = 625$. Finally, we will investigate the problem of the Impulsively started cylinder at $Re = 550$.

3.4.1 Lamb-Oseen Vortex

The Lamb-Oseen vortex is an analytical solution by Lamb and Oseen, describing the diffusion of a vortex core [52]. We solved the same problem as the one described in the Lagrangian validation problem 2.6.2. However, Eulerian domain use the primitive variables $\mathbf{u} - p$ for formulating the problem. Therefore, the initial conditions of the problem was the velocity field of Lamb-Oseen vortex,

$$u_\theta = \frac{\Gamma_c}{2\pi r}\left[1-\exp\left(-\frac{r^2}{4\tau}\right)\right] \tag{3.34a}$$

$$u_r = 0, \tag{3.34b}$$

where Γ_c is the vortex core strength, $\tau \equiv \nu t$ is the scaled viscous time, and r is the distance from the core center. The parameters of the simulation is tabulated in table 3.2. To make the Eulerian simulation comparable with the Lagrangian simulation, we tried to match the spatial discretization of the Eulerian domain with the Lagrangian domain. The figure 3.6 shows the domain of the problem, discretized the domain $\Omega = [-1,1]^2$ in a structure grid with the number of finite element cells $N_{\text{cells}} = 200^2$ in x and y direction, minimum cell size $h = \sqrt{2}/100$.

Furthermore, the Figure 3.6 also shows the boundary domains $\partial\Omega$ of the fluid domain. For the Lamb-Oseen problem, as we have the analytical solution of the velocity field for

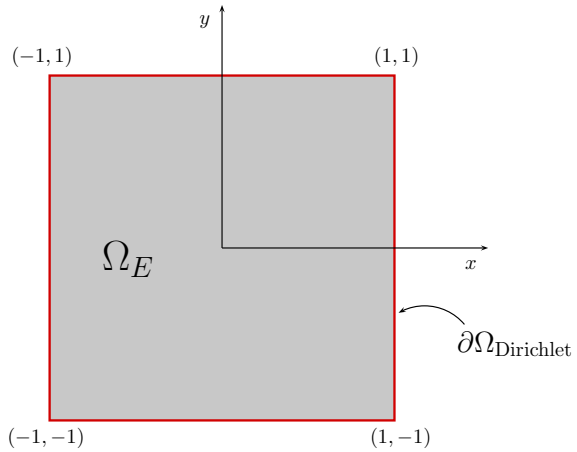


Figure 3.6: Eulerian domain for the Lamb-Oseen vortex problem. Figure shows the bound of the domain $\Omega = [-1, 1]^2$, identified as $\mathrm{ID}_{\mathrm{fluid}} = 1$; and the boundary domain $\partial\Omega$ [—, solid red], identified as $\mathrm{ID}_{\mathrm{ext}} = 3$, which is where the Dirichlet velocity boundary condition was applied.

all time, we can use this solution to prescribe the external domain boundary condition. So for this problem, we only need an external Dirichlet velocity boundary condition, at the boundary domain identified as, $\mathrm{ID}_{\mathrm{ext}} = 3$. This would imply that we do not need to explicitly apply pressure boundary condition, as we already have a velocity boundary condition. With all the boundary conditions, we can evolve the initial velocity distribution of the Lamb-Oseen vortex from $t_0 = 4$ to $t_f = 5$, using the IPCS algorithm described in section **??**.

We used CFL stability condition equation 3.29, to determine the time step size, $\Delta t = 0.001$. The Eulerian method time steps using a Forward Euler (FE) time marching and requires $N_{\mathrm{t-steps}} = 1000$ time steps. During the evolution, we evaluated the growth of the error in velocity, and in vorticity between the numerical results and the analytical solution. As mainly we are interested in the error in vorticity, as this is the quantity which will be interpolated onto the Lagrangian domain, figure 3.7 shows the initial and the final relative error in vorticity over the Eulerian domain. Opposed to the Lagrangian results,

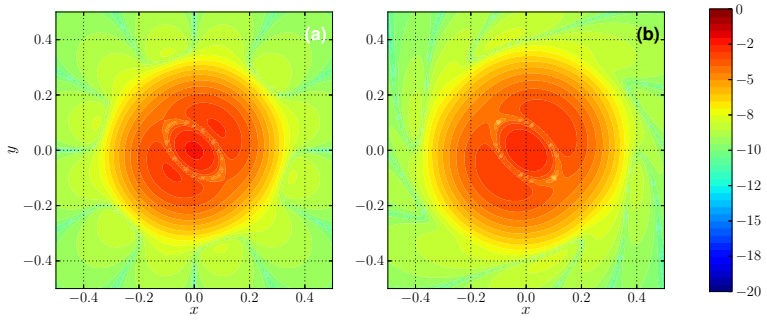


Figure 3.7: Relative error in vorticity field in logarithmic scale. Figure **(a)** shows the initial relative error in vorticity at $t = t_0$, and figure **(b)** shows the relative error in vorticity at the end of the time stepping $t = t_f$.

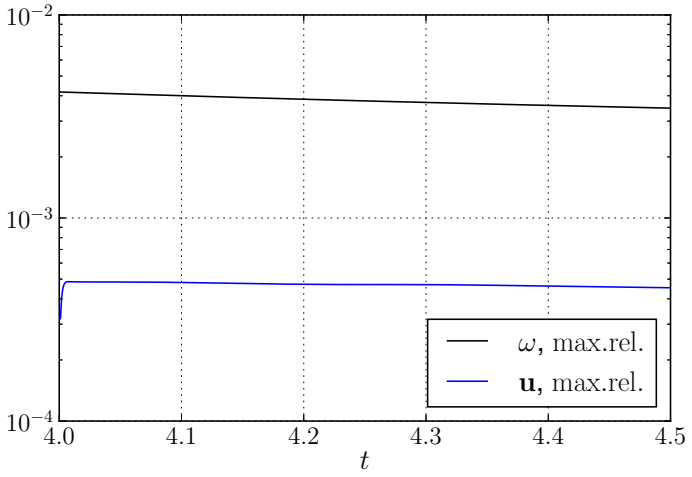
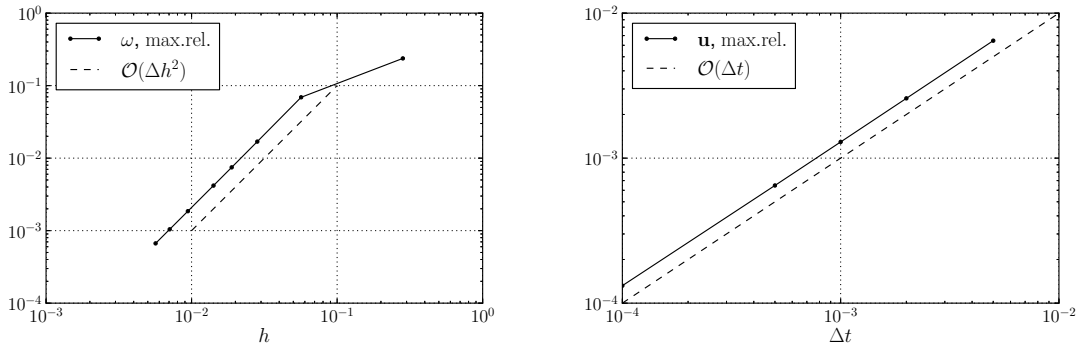


Figure 3.8: Eulerian Lamb-Oseen relative vorticity evolution

figure 2.19, we see that initial relative error in the vorticity field is larger. This is so because, the Lagrangian domain was initialized using the vorticity, where as the Eulerian domain was initialized using the velocity. To calculate the vorticity on the Eulerian domain, we had to project curl the velocity onto the function space of vorticity W. This process of initialization in the finite element domain and project of the vorticity introduced additional numerical error. However, the pattern of the relative error in vorticity, is similar to the Lagrangian solution, as the highest error is localized to the core center, where we have the highest gradient in vorticity.

As the time progress, we see that the error of the problem is stable and does not increase as observed for the Lagrangian domain, figure 3.7b. The growth of the relative error in velocity and vorticity can be observed in figure **??**. It shows that during the evolution of the Lamb-Oseen vortex, the relative error in velocity, and vorticity is stable. We see that due to the relation of vorticity to velocity, the error in vorticity is higher that velocity. However, the stability in the error of the solution is vital as it verifies that our method is stable and robust for solving the problem.



(a) Lamb-Oseen dx convergence

(b) Lamb-Oseen dt convergence

Figure 3.9: Lamb-Oseen convergence

In order to have an extensive understanding of the stability of our scheme, we investigated the convergence of the spatial and temporal discretization of the domain. To determine the convergence of space, the simulation was run for $h \approx 0.25$ to $h \approx 5 \times 10^{-3}$. Figure 3.9a shows the convergence of the relative error in vorticity, due to increase in spatial resolution. This validates that the scheme is 2^{nd}-order in space, due to second order function space CG_2 for the primitive variable, velocity.

To determine convergence in time, 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 primitive variable $\mathbf{u}$, converged at an order 1, figure 3.9. This is true to the theory, as we are employing a 1^{st}-order Forward Euler scheme for the time integration. Thus, we have verified with the analytical solution of the Lamb-Oseen vortex that our Eulerian method is implemented according to the theory, and perform in a robust manner.

3.4.2 Clercx-Bruneau dipole collison at $Re = 625$

To validate that our Eulerian method performs according to theory, and to provide a benchmark case for the hybrid coupling scheme, we decided to simulate a flow where the generation of the vorticity from the wall is the main focus. To generate this benchmark case, we used the numerical experiment benchmarked by Clercx and Bruneu in 2005 [], on the collison of the dipole with a no-slip boundary.

The reason for using the Clercx-Bruneau dipole for validation is because, it a common practise to use the dipole-wall collision test case to validate the numerical method. Ouldsalihi et al [] used this case to validate their Hybrid method that couples vortex particles with finite-difference method. Cottet et al. [] used the collision to tool to validate the vortex method.

Clercx and Bruneau have stated that dipole-wall interaction test cases are ideal for methods requiring the well-resolvedness of the thin boundary layer near the no-slip walls. The literature determined the convergence of the numerical solution by investigating quantities like the total kinetic energy, the enstrophy and the total angular momentum of the flow. It mainly focuses on the vorticity distribution and the vorticity flux at the no-slip boundary of the domain. Furthermore, this literature provided well-defined initial and boundary conditions, with smooth vorticity distribution as seen for the Lamb-Oseen vortex case.

The Clercx-Bruneau test case investiated the normal collision of a dipole with a no-slip wall. The parameters of the problem are tabulated in table **??**.

The collison of a dipole normal to wall, from Clercx nand Bruneau [], was an ideal choice as the literature has provided a systematic methodology to perform the simulation and have provide detailed results.

The Clercx-Bruneau dipole

3.4.3 Impulsively started cylinder at $Re = 550$

3.5 Summary

Table 3.3: Summary of the parameters for the Clercx-Bruneau normal collsion of a dipole with a no-slip wall [].

Parameters	Value	Unit	Description
Ω	$[-1,1]^2$	m	Eulerian domain bounds
$Re = UW/\nu$	625	-	Reynolds number
W	1	m	Half width of the domain
U	1	$\mathrm{m\,s}^1$	Characteristic velocity
ν	1	$\mathrm{m\,s}^1$	Characteristic velocity
Γ_c	1	$\mathrm{m}^2\,\mathrm{s}^{-1}$	Core strength
$\mathbf{u}_\infty$	$[0,0]$	$\mathrm{m\,s}^{-1}$	Free-stream velocity
ν	5×10^{-4}	$\mathrm{kg\,s}^{-1}\,\mathrm{m}^{-1}$	Kinematic viscosity
(τ_0, τ_f)	2×10^{-3} to 2.5×10^{-3}	m^2	Initial and final scaled viscous time
(t_0, t_f)	4 to 5	s	Initial and final simulation time
$h_{\min}$	$\frac{1}{100}\sqrt{2}$	m	Minimum mesh cell size
N_{cells}	200^2	-	Number of mesh cells
CFL	0.95	-	CFL number
$\|\mathbf{u}\|_{\max}$	1.5	$\mathrm{m\,s}^{-1}$	Maximum magnitude fo the velocity
Δt	0.001	s	Time step size
$N_{\text{t-steps}}$	1000	-	Number of time integration steps
ID_{fluid}	1	-	Fluid domain I.D
ID_{ext}	3	-	External Dirichlet velocity boundary I.D

Chapter 4

Hybrid Eulerian-Lagrangian Vortex Particle Method

Chapter 5

Verification and Validation of Hybrid Method

Chapter 6

Conclusion and Recommendation

6.1 Conclusion

6.1.1 Lagrangian domain

6.1.2 Eulerian domain

6.1.3 Hybrid method

6.2 Recommendations

6.2.1 Lagrangian domain

6.2.2 Eulerian domain

6.2.3 Hybrid method

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] Boffi, D. Three-Dimensional Finite Element Methods for the Stokes Problem. *SIAM Journal on Numerical Analysis 34*, 2 (Apr. 1997), 664–670.

[5] 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.

[6] Brenner, S. C., and Scott, L. R. *The Mathematical Theory of Finite Element Methods*. Texts in applied mathematics. Springer-Verlag, 2002.

[7] Brezzi, F., and Fortin, M. *Mixed and hybrid finite elements methods*. Springer series in computational mathematics. Springer-Verlag, 1991.

[8] Carey, G. F. *Computational Grids: Generations, Adaptation & Solution Strategies*. CRC Press, 1997.

[9] 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.

[10] Chen, Z. *The Finite Element Method: Its Fundamentals and Applications in Engineering*. World Scientific, 2011.

[11] Chorin, A. J. Numerical solution of the Navier-Stokes equations. *Mathematics of computation 22*, 104 (1968), 745–762.

[12] Chorin, A. J. Numerical Study of Slightly Viscous Flow. *Journal of Fluid Mechanics 57*, Part 4 (1973), 785–796.

[13] 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.

[14] 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.

[15] Commons, W. Darrieus windmill, 2007.

[16] Commons, W. Windmills d1-d4 (thornton bank), 2008.

[17] 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).

[18] Cottet, G. H., and Koumoutsakos, P. D. *Vortex Methods: Theory and Practice*, vol. 12. Cambridge University Press, 2000.

[19] Courant, R. Variational methods for the solution of problems of equilibrium and vibrations. *Bull. Amer. Math. Soc 49*, 1 (1943), 1–23.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] Hansen, G. A., Douglass, R. W., and Zardecki, A. *Mesh Enhancement: Selected Elliptic Methods, Foundations and Applications*. Imperial College Press, 2005.

[29] 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.

[30] Katz, J., and Plotkin, A. *Low-speed aerodynamics*. Cambridge Aerospace Series. Cambridge University Press, 2001.

[31] Koumoutsakos, P. *Direct numerical simulations of unsteady separated flows using vortex methods*. PhD thesis, California Institute of Technology, USA, 1993.

[32] 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.

[33] Koumoutsakos, P., Leonard, A., and Pépin, F. Boundary Conditions for Viscous Vortex Methods. *Journal of Computational Physics 113*, 1 (July 1994), 52–61.

[34] Lamb, H. *Hydrodynamics*. Cambridge university press, 1993.

[35] 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.

[36] Logg, A. NSBench in Launchpad. https://launchpad.net/nsbench, 2014.

[37] Logg, A., Mardal, K.-A., Wells, G. N., et al. *Automated Solution of Differential Equations by the Finite Element Method*, vol. 84. Springer, 2012.

[38] Logg, A., and Wells, G. N. DOLFIN: Automated finite element computing. *ACM Transactions on Mathematical Software 37*, 2 (2010).

[39] Monaghan, J. Extrapolating B-splines for interpolation. *Journal of Computational Physics 60*, 2 (Sept. 1985), 253–262.

[40] Nair, M. T., and Sengupta, T. K. Unsteady flow past elliptic cylinders. *Journal of Fluids and Structures 11*, 6 (1997), 555–595.

[41] 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.

[42] RAO, S. S. *The Finite Element Method in Engineering.* Elsevier Science, 2011.

[43] Reed, W., and Hill, T. R. Triangular Mesh Method for the Neutron Transport Equation. 10–31.

[44] Shankar, S., and Dommelen, L. A New Diffusion Procedure for Vortex Methods. *Journal of Computational Physics 127*, 1 (Aug. 1996), 88–109.

[45] Shiels, D. *Simulation of controlled bluff body flow with a viscous vortex method.* PhD thesis, California Institute of Technology, USA, 1998.

[46] 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.

[47] 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.

[48] 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.

[49] Speck, R. *Generalized Algebraic Kernels and Multipole Expansions for massively parallel Vortex Particle Methods.* PhD thesis, University of Wuppertal, Germany, 2011.

[50] 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).

[51] 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.

[52] Tryggeson, H. *Analytical Vortex Solutions to the Navier-Stokes Equation.* PhD thesis, Växjö University, Sweden, 2007.

[53] Tutty, O. R. A Simple Redistribution Vortex Method (with Accurate Body Forces). *ArXiv e-prints* (Sept. 2010).

[54] Vermeer, L., Sø rensen, J., and Crespo, A. Wind turbine wake aerodynamics. *Progress in Aerospace Sciences 39*, 6-7 (Aug. 2003), 467–510.

[55] 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.

[56] Wikipedia. Vertical-Axis Wind Turbine, July 2013.

[57] 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.