

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



Copyright © L. Manickathan B.Sc.
All rights reserved.

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

Industrial Supervisor:

prof.dr.ir. I. Bennett

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

Summary	v
Acknowledgements	vii
List of Figures	xv
List of Tables	xvii
Nomenclature	xix
1 Introduction	1
1.1 Motivation and Goal	2
1.2 Research Aim and Plan	3
1.3 Introduction to Hybrid Eulerian-Lagrangian Vortex Particle Method . . .	4
1.3.1 Simple coupling strategy	5
1.4 Verification and Validation Test Cases	7
1.5 Methodology	8
1.6 Thesis Outline	8
2 Lagrangian Domain: Vortex Particle Method	9
2.1 Introduction to Vortex Particle Method	9
2.1.1 Vorticity	9
2.1.2 Velocity-vorticity formulation of the Navier-Stokes equations . . .	10
2.1.3 Viscous splitting algorithm	10
2.2 Spatial Discretization: Generation of Vortex Blobs	11
2.2.1 Biot-Savart law	11
2.2.2 Discrete form of vorticity field	12
2.2.3 Mollified vortex kernels	12
2.2.4 Vortex blob initialization	14

2.3	Convection of vortex blobs	17
2.3.1	Remeshing scheme: Treating lagrangian grid distortion	17
2.4	Diffusion of Vortex Methods	19
2.4.1	Modified remeshing for treating diffusion	20
2.5	Boundary conditions at solid boundary	23
2.5.1	Boundary integral equations	24
2.5.2	Panel method for treating no-slip boundary condition	26
2.5.3	Error analysis of panel method	29
2.6	Validation of Lagrangian method	31
2.6.1	Evolution of Lamb-Oseen vortex	33
2.6.2	Convergence study of the viscous vortex method	37
2.7	Summary of the Lagrangian method	38
3	Eulerian Domain: Finite Element Method	41
3.1	Introduction to Finite Element Method	42
3.2	Solving the Finite Element problem	45
3.2.1	Introduction to FEniCS Project	45
3.2.2	Mesh generation using GMSH	47
3.3	Solving Incompressible Navier-Stokes Equations	47
3.3.1	Velocity-pressure formulation	48
3.3.2	Incremental pressure correction scheme	48
3.3.3	Determining the vorticity field	50
3.3.4	Determining the body forces	51
3.4	Validation of eulerian method	52
3.4.1	Lamb-Oseen Vortex	52
3.4.2	Clercx-Bruneau dipole collison at $Re = 625$	52
3.4.3	Impulsively started cylinder at $Re = 550$	52
3.5	Summary	52
4	Hybrid Eulerian-Lagrangian Vortex Particle Method	55
4.1	Theory of Domain Decomposition Method	55
4.1.1	Advantage of domain decomposition	55
4.1.2	Assumptions and Limitations	55
4.1.3	Modified coupling strategy	55
4.2	Eulerian-Lagrangian coupling algorithm	55
4.2.1	Eulerian dirichlet boundary condition	55
4.2.2	Vorticity interpolation algorithm	55
4.3	Introduction to pHyFlow: Hybrid solver	55
4.3.1	Program structure	55
4.4	Summary	55

5	Verification and Validation of Hybrid Method	57
5.1	Error in coupling: Verification with Lamb-Ossen vortex	58
5.1.1	Generation of artificial vorticity	58
5.2	Clercx-Bruneau dipole convection at $Re = 625$	58
5.2.1	Comparison of vorticity contours	58
5.2.2	Variation in maximum vorticity	58
5.2.3	Variation in kinetic energy	58
5.2.4	Variation in enstrophy	58
5.3	Clercx-Bruneau dipole collision at $Re = 625$	58
5.3.1	Comparison of vorticity contours	58
5.3.2	Variation in maximum vorticity	58
5.3.3	Variation in kinetic energy	58
5.3.4	Variation in enstrophy	58
5.3.5	Variation in palinstrophy	58
5.4	Impulsively started cylinder problem at $Re = 550$	58
5.4.1	Evolution of the wake	58
5.4.2	Evolution of pressure and friction drag	58
5.4.3	Evolution of lift	58
5.5	Moving body	58
5.5.1	Error due to perturbation lag	58
5.6	Proof of concepts	58
5.6.1	Multiple cylinder case	58
5.6.2	Stalled airfoil at $Re = 5000$	58
5.7	Summary	58
6	Conclusion and Recommendation	59
6.1	Conclusion	59
6.1.1	Lagrangian domain	59
6.1.2	Eulerian domain	59
6.1.3	Hybrid method	59
6.2	Recommendations	59
6.2.1	Lagrangian domain	59
6.2.2	Eulerian domain	59
6.2.3	Hybrid method	59
	References	61

List of Figures

1.1	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.	2
1.2	Lagrangian formulation of the fluid. We track the path of the individual fluid elements as time passes.	3
1.3	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) [21]	5
1.4	Modified domain decomposition without Schwartz alternating method. Lagrangian domain extends up to the surface of the body. Figure is based on Daeninck (2006) [15].	5
1.5	Flowchart of the simple coupling strategy. The flowchart shows the procedure to evolve both methods from t_n to t_{n+1}	6
2.1	Definition of circulation of the fluid	10
2.2	Vortex blob with Gaussian distribution: $[k = 2, \sigma = 1.0]$	13
2.3	Vortex blob with overlap σ/h	13
2.4	Mollified vorticity field of an arbitrary vorticity function with overlap = 1.0, $\sigma = 0.19$, $h = 0.19$. Vortex blob strength has been assigned by equation 2.18, sampling at exact vorticity [\bullet , red dot]. Figure depicts exact vorticity distribution ω [—, solid], vorticity field of each blob ω_i [—, green dashed], the mollified vorticity field ω^h [- -, dashed].	14
2.5	Mollified vorticity field after two Beale's iteration, overlap = 1.0, $\sigma = 0.19$, $h = 0.19$. Figure depicts exact vorticity distribution ω [—, solid], vorticity field of each blob ω_i [—, green dashed], the mollified vorticity field ω^h [- -, dashed].	16
2.6	Convergence of vorticity by modifying the spatial resolution. Figure depicts exact vorticity field ω with [—, black] and various resolutions.	16
2.7	Lagrangian distortion of the vortex blobs after 100 steps. The initial vorticity field $\omega(\mathbf{x}, 0) = \exp(-12 \mathbf{x})$ with $\Delta t = 0.1$, $\sigma = 0.02$ and overlap = 1.0. Figure depicts (a) the initial and (b) the final distribution of the vortex blobs.	17

2.8	Remehsing of vortex blob (\bullet , green) on the uniform grid defined by the (4×4) 2-D stencil.	18
2.9	M'_4 interpolation kernel, a third-order peicewise smooth B-spline kernel by Monaghan [37]	19
2.10	One dimensional, Simple redistribution scheme, diffusing the vortex blobs to the four stencil points ($k = i - 1, \dots, i + 2$) with vortex blob positioned at $x_i \leq x_\nu \leq x_{i+1}$ and nominal blob spacing h	22
2.11	Extended vorticity field consisting of vorticity in the fluid and vortex sheet distribution around the body.	24
2.12	Extended vorticity field: Vortex sheet being an extension to the vorticity field (resolved by the vortex blobs) which is able to fully resolve the boundary vorticity	25
2.13	Vortex panel's global (a) and local (b) coordinates system definition as defined by Katz and Plotkins [26].	28
2.14	twoPanelBodies	28
2.15	Panel method solution: Potential flow velocity field around unit cylinder. The figure depicts $\ \mathbf{u}\ $ with zero velocity inside the body.	30
2.16	Comparison of the velocity field along the y -axis $0 \rightarrow 10$. 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 between the solution, given by equation 2.64	30
2.17	Convergence plot of the constant-strength straight vortex panels. The figures depicts the reduction is error at $\mathcal{O}(N)$	31
2.18	Lamb-Oseen (a) vorticity distribution, and (b) velocity distribution. $\Gamma_c = 1$, $\tau = 2 \times 10^{-3}$	32
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 $t_f = 5$	34
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], error in L^2 - norm [- -, dashed black]; and error in velocity: maximum relative error [—, solid blue], error in L^2 - norm [- -, dashed blue].	35
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 $\Delta t_d = \Delta t_c = 0.01$ [- -, solid blue], and $\Delta t_d = \Delta t_c = 0.02$ [- -, dashed black].	36
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.	37
2.23	Error growth of Lamb-Oseen vorticity field after one-step.	37
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}	39

3.1	A two-dimensional Finite Element geometry. The cell represents the area of the element, and vertices are the edge of the cell.	42
3.2	Delaunay triangulation of the fluid around a cylinder resulting in unstructured mesh with controllable cell sizes.	43
3.3	DOLFIN VTK plot of the Poisson solution, given by the problem, source code listing 3.1.	47
3.4	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}	49
3.5	Eulerian Lamb-Oseen relative vorticity evolution	53
3.6	Lamb-Oseen convergence	53

List of Tables

2.1	Panel study parameters	29
2.2	Summary of the Lamb-Oseen vortex evolution study parameters. Table shows the parameters of Tutty’s diffusion method [48]	33

Nomenclature

Latin Symbols

c^2	Diffusion parameter	-
\mathcal{E}	Enstrophy	$\text{m}^2 \text{s}^{-2}$
h	Nominal particle spacing	m
h_ν	Characteristic diffusion distance	m
\mathbf{K}	Biot-Savart kernel	-
\mathbf{K}_σ	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	m s^{-1}
\mathbf{u}_b	Velocity of the body	m s^{-1}
\mathbf{u}_γ	Vortex sheet induced velocity	m s^{-1}
\mathbf{u}_{ext}	External induced velocity	m s^{-1}

\mathbf{u}^h	Discrete velocity	m s^{-1}
u_θ	Angular velocity	m s^{-1}
\mathbf{u}_∞	Free-stream velocity	m s^{-1}
\mathbf{u}_ϕ	Potential velocity	m s^{-1}
u_r	Radial velocity	m s^{-1}
\mathbf{u}_{slip}	Boundary slip velocity	m s^{-1}
\mathbf{u}_ω	Vortical velocity	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}_ν	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	$\text{m}^2 \text{s}^{-1}$
β_p	Corrected circulation of the particle	$\text{m}^2 \text{s}^{-1}$
Γ	Circulation	$\text{m}^2 \text{s}^{-1}$
Γ_b	Circulation of moving body	$\text{m}^2 \text{s}^{-1}$
γ	Vortex sheet strengths	s
Γ_c	Circulation of the vortex core	$\text{m}^2 \text{s}^{-1}$
Γ_γ	Circulation of vortex sheet	$\text{m}^2 \text{s}^{-1}$
Γ_ω	Circulation of the fluid	$\text{m}^2 \text{s}^{-1}$
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	$\text{m}^2 \text{s}^{-1}$
ξ	Scale relative position of particle to stencil node	-
ρ	Density	kg m^{-3}
σ	Core size	m
τ	Scaled viscous time	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

AD	Actuator Disk
BEM	Blade Element Momentum
CFD	Computational Fluid Dynamic
CG	Continuous Galerkin
CSVM	Constant-Strength Vortex Method
DG	Discontinuous 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
HVM	Hybrid Eulerian-Lagrangian Vortex Particle Method
IPCS	Incremental Pressure Correction Scheme
LHS	Left Hand Side
LSTSQ	Least-Square solution method
MPI	Message Passing Interface
PIV	Particle Image Velocimetry
PSE	Particle Strength Exchange
VAWT	Vertical-Axis Wind Turbine
VPM	Vortex Particle Method
VRM	Vortex Redistribution Method

Chapter 1

Introduction

Conventional energy resources such as fossil fuels and nuclear energy are not only limited supply 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 turbines is now an interested research field.

Vertical-Axis Wind Turbine (VAWT) are unlike the normal wind turbine. Typical wind turbines are mounted on a mast away from the ground and generates energy by spinning normal to the ground. However, a VAWT spins parallel to the ground with its hub located at the ground [51]. The advantages of the vertical axis wind turbine are what makes them ideal for a source of renewable energy. As the turbine is located at the ground, it is easily accessible and can be easily maintained. The second main advantage of the VAWT is the way it dissipates its wake. Near-wake experiments of Ferreira (2009) [18], and simulations of Vermeer (2003) [49] have shown that the fluid past the turbine is more turbulent. Due to this higher mixing, the flow is able to smooth out much earlier. This 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 (low tip-speed ratios).

However, with these advantages also comes some drawbacks. As the blades passes through its own wake, complex wake-body interactions take places. These have adverse effect on the blade structure and therefore is more susceptible to fatigue. This happens because the blades are constantly pitching, and complex flow behaviour such as dynamic stall and constant vortex shedding occurs [44]. This complex fluid behaviours makes it hard to predict the performance of a VAWT and this is one of the reasons why VAWTs are not mainstream. In addition, as the VAWT operates at large Reynolds number, accurate numerical methods are computationally very expensive. Therefore, it is vital to have a good understanding of the flow structure evolution and the wake generation of the VAWT using not only an efficient method, but also an accurate one.

1.1 Motivation and Goal

The goal of the research is to develop an efficient, reliable, and accurate numerical method for modelling the flow around a 2D VAWT such that one is able to deduce the correct performance characteristics of the VAWT. The two main approaches of investigate the flow is either using a numerical method to simulate the flow or by performing real-life experimental tests.

To understand the unsteady aerodynamic behaviour, Particle Image Velocimetry (PIV) has been as useful tool to visualize the flow around the turbine. Ferreira et al. (2007) [17], have shown that it was possible to acquire flow characteristics around the blade, and these method had the accuracy to be used as validation tools. However, the downside to experimental investigation is that is it very expensive to investigate all types of configuration. Furthermore, the model sizes are limited by the dimensions of the wind tunnel and investigations with arrays of VAWT is difficult.

Numerical methods are a popular alternative as the cost of simulation 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 one of simplest models that are build upon satisfying the momentum balance of the turbine and the fluid. The advantage of theses models are that they are very quick, however they lack the accuracy that can be achieved by experimental simulation. Complex blade-wake interactions such as dynamic stalls and flow separations cannot be modeled by these methods.

To ensure more accuracy, one has to solve the Navier-Stokes equation of the flow around the turbine. Computational Fluid Dynamic (CFD) methods discretizes the fluid into smaller regions and solves the set of Navier-Stokes equation in each region (or grids). This type of formulation is known as an Eulerian method as we are evaluating the change in flow property of a given region, figure 1.1. In order to fully resolve the flow around the turbine, we would have to discretize the fluid at the order of the size of the vortex cores. As the vortex cores are very small (a fraction of size of the airfoil) near the blade, and very large far away from the blade (in order of size of the turbine), we must have grid size that adapts. This requires a large number of grids, as the blades are constantly moving, and makes it computation very expensive to solve especially for arrays of turbines.

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 La-

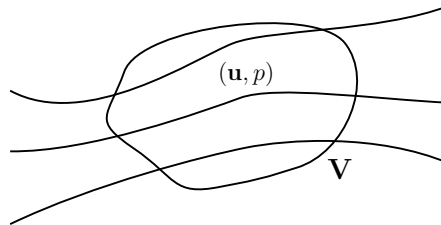


Figure 1.1: Eulerian formulation of the fluid. We observe a given volume \mathbf{V} and evaluate the change in properties of the fluid (from incompressible flow: velocity \mathbf{u} and pressure p) at time passes.

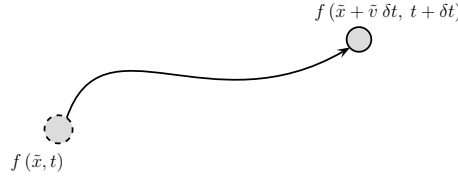


Figure 1.2: Lagrangian formulation of the fluid. We track the path of the individual fluid elements as time passes.

In the Lagrangian formulation, the vorticity evolution is evaluated as interaction between vortices, and this removes the requirement of gridding, figure 1.2. 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 than typical CFD methods and can easily be scaled to distributed computation. However, vortex methods cannot inherently take into account the solid body. They require additional methods that can describe the effect of the body in the fluid and the vorticity generated from the body.

So, we see that the Eulerian method is accurate when describing the blade-wake interaction but is not efficient when describing multi-scale domains. Whereas, the Lagrangian method is very efficient in the evolution of the vorticity of the fluid, and is an ideal choice when describing the multi-scale flow characteristics.

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 (HVM). In this method, the Eulerian grid method will be used at the region around the blade (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 phenomena such as vortex shedding, dynamic stall, and the wake-body interaction, but also capture the large-scale flow structures such as the evolution of the VAWT wake, with efficiency and accuracy.

1.2 Research Aim and Plan

Research Questions: *Is it possible to develop an efficient and accurate numerical method by a hybrid approach where the vortex particle method is used in the wake, and the Navier-Stokes grid solver is used at the near-body region? Will it be able to simulate real life performance characteristics of a vertical axis wind turbine? 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?*

Research aim and plan:

- Develop the Hybrid method for capturing small-scale phenomena and large scale phenomena.

- Ensure this tool is efficient, reliable, and accurate.
- Verify, Validate the tools with model problems.
- Apply the model to the 2D flow of VAWT.

In order to answer this 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 modelling the wake evolution of VAWT. The investigation will be performed for 2D geometries and the accuracy of the model needs to established first at simpler problems before continuing to more complex problem cases.

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 final goal is to perform a simulation of a VAWT in 2D, compute its performance and validating it against experimental data. By the gradual development of the complex elements of the simulation, one can investigate the feasibility of such approach. Also, the investigation is only done for 2D currently as a full 3D simulation might be difficult and might not be feasible for the master thesis yet.

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 behaviour of a VAWT.

1.3 Introduction to Hybrid Eulerian-Lagrangian Vortex Particle Method

The Hybrid Eulerian-Lagrangian Vortex Particle Method (**HVM**) 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 1.3.

Several researches have already been done: Cottet and Koumoutsakos (2000a)[14], Guermont and Lu (2000) [21] simulating the advection dominated flows, Ould-Salhi et al. (2001) [39] blending finite difference and vortex method together, Winckelmans et al. (2005a) [53] investigating the trailing vortices, Daeninck (2006) [15] implementing RANS and LES to the simulation, Stock (2010) [46] using GPU clusters for efficiency and Speck (2011) [45] implementing researching on multipole expansion and modified interpolation kernels.

When evaluating the previous works, we see that not all domain decomposition methods are the same. The main difference differencing between the methods is their coupling

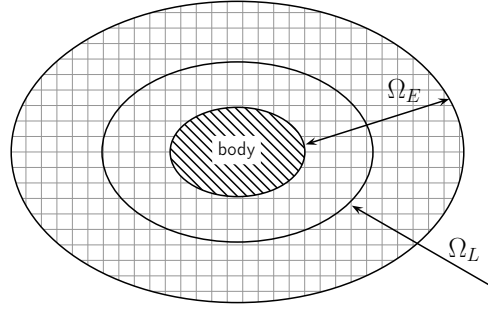


Figure 1.3: Standard domain decomposition using Schwartz iteration for coupling the two methods. Eulerian domain Ω_E (near the body), and Lagrangian domain Ω_L (away from the body). Figure is based on Guermond (2000) [21]

strategies. Most works employ the Schwartz alternating method to couple the vortex particle method and the grid solver. The Schwartz alternating method solve the grid solver initially and couples it with the vortex method by iteratively trying to satisfy the boundary conditions. However, for this project the coupling techniques that will be used is similar to Daeninck (2006) [15] and Stock (2010) [46].

1.3.1 Simple coupling strategy

This new approach is much simpler and only a single iteration is needed for the coupling. The basic procedure is to solve the vortex method in the whole domain using relatively coarse evaluation, then 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 1.4. 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, referred to as Eulerian

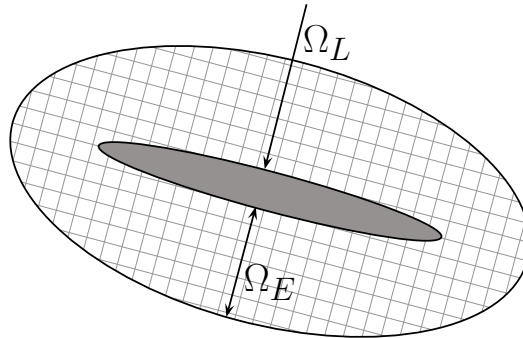


Figure 1.4: Modified domain decomposition without Schwartz alternating method. Lagrangian domain extends up to the surface of the body. Figure is based on Daeninck (2006) [15].

domain Ω_E . With this implementation, subtle features of the boundary layer (such as flow separation) can be resolved with great accuracy.

- Lagrangian method is used to capture the wake, referred to as Lagrangian domain Ω_L . Lagrangian method is used to efficiently evolve the wake.
- The accurate solution of the Eulerian domain is transferred to the Lagrangian domain according to the coupling strategies of Daeninck [15] and Stock [46].
- The boundary conditions for the Eulerian domain is retrieved from the Lagrangian domain.

Figure 1.5 shows the overview to the simple coupling strategy. The detailed algorithm to the simple coupling strategy follows from Daeninck [15].

The simple coupling algorithm 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 overlapping in the Eulerian domain.
2. **Evolve Lagrangian:** With the modified solution, evolve the Lagrangian solution from t_n to t_{n+1} .
3. **Determine Eulerian boundary conditions:** Use the Lagrangian solution of 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} .

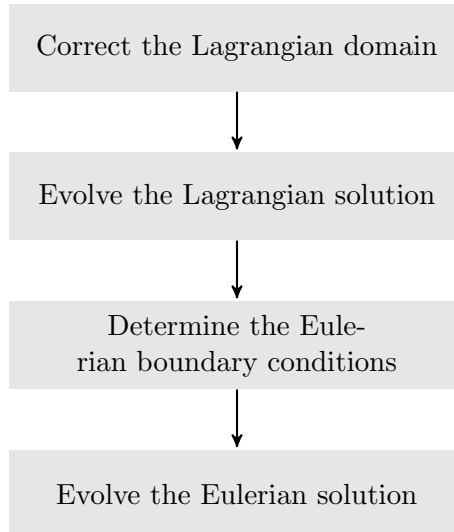


Figure 1.5: Flowchart of the simple coupling strategy. The flowchart shows the procedure to evolve both methods from t_n to t_{n+1} .

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. In addition, the second difference between the classical hybrid method is that the Lagrangian method handles the boundary conditions differently. Typically during the evolution process of the Lagrangian domain, vorticity is shed from body. However, in this coupling strategy, the Lagrangian method under-resolves the boundary and cannot be used to resolve the vorticity flux. Instead, we use the Eulerian method to resolve the boundary and interpolate the newly generated vorticity into the Lagrangian domain.

However, there are some assumptions that we must satisfy, for this coupling strategy to be valid:

1. At t_n , before the evolution of both methods to t_{n+1} , the Lagrangian solution matches Eulerian solution at the near-wall region.
2. After the evolution to t_{n+1} , the deviation of the Lagrangian solution (due to lack of vorticity flux at Lagrangian boundary), is minimal.
3. 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 [31] [47]

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 diffusion of the vorticity is an ideal to verify and validate the convection and diffusion of the vorticity.

Clercx-Bruneau dipole [12]

The Clercx-Bruneau dipole test case is the simple case of dipole colliding with a wall and is used to verify the coupling of the methods. This test case focuses on the generation of vorticity from the wall and is ideal to verify and validate the coupling.

Impulsively started cylinder [28] [7] [8] [33]

The impulsively started cylinder test case focuses on 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 [38]

The elliptic airfoil test cases focus on the flow separation past a lifting body. The elliptic airfoil is pitched at high angle of attack and the flow past the airfoil is comparatively unsteady and undergoes phenomena such as laminar separation bubble, flow separation and Karman vortex shedding from the trailing edge of the airfoil. This helps us ensure coupling process is sound for complex flow phenomena.

1.5 Methodology

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

- Develop the vortex particle method
- Validate the vortex particle method against a Lamb-Oseen convection test case.
- Develop the vortex panel method to deal with the boundaries for the vortex particle calculation.
- Validate the vortex panel method by solving a potential flow around a cylinder.
- Develop the grid solver that is based on the Finite Element method.
- 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.

- Couple vortex particle, vortex panel and grid solver together.
- Validate it with the previous generated test case solution.
- 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

!!! Update this section !!!

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 it is described as a function of space \mathbf{x} and time t . However, the lagrangian point of view regards the fluid as a collection of the particles carrying the property of the fluid.

Unlike the typical eulerian method that require 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 on Graphics Processing Units ([GPU](#)), VPM can be more efficient than typical eulerian methods.

2.1.1 Vorticity

Vorticity ω , the governing element of vortex particle method, is defined as

$$\omega = \Delta \times \mathbf{u}, \quad (2.1)$$

where \mathbf{u} is the velocity. The circulation Γ is defined as

$$\Gamma = \int_L \mathbf{u} \cdot d\mathbf{r} = \int_S \omega \cdot \mathbf{n} dS, \quad (2.2)$$

by the stokes theorem, as represents the integral vorticity of the domain, figure [2.1](#)

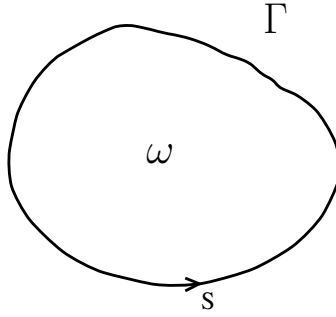


Figure 2.1: Definition of circulation of the fluid

2.1.2 Velocity-vorticity formulation of the Navier-Stokes equations

The governing equation of the vortex particle method is velocity-vorticity $\mathbf{u} - \omega$ formulation of the Navier-Stokes equations [14]. The 2-D incompressible Navier-Stokes momentum equation is given as

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

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

$$\nabla \cdot \mathbf{u} = 0. \quad (2.4)$$

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

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

which only relates the vorticity to the velocity enabling us to neglect the pressure field. Note that as we are dealing with the two dimensional flow, we neglected the stretching term.

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 behaviour of the fluid. Chorin [10] has shown that using the viscous splitting algorithm, it is possible to simulate a viscous flow.

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

- Sub-step 1: convection

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = 0; \quad (2.6)$$

- Sub-step 2: diffusion

$$\frac{\partial \omega}{\partial t} = \nu \nabla^2 \omega. \quad (2.7)$$

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 deal with the diffusion of the vorticity field.

There are several advantages to this type of evolution. As the convection and diffusion are handled separately, there is minimum dispersion during the convection and furthermore, there is no restriction of the advection CFL number [50].

There are many ways of dealing with the diffusion of the vorticity field. During this project, we use a modified interpolation kernel [50] that can simultaneously treat diffusion and remesh the vortex particles, see section 2.4.

2.2 Spatial Discretization: Generation of Vortex Blobs

In order to deal with the vorticity field, we must first discretize the vorticity to vortex particles. Vortex blobs have been first introduced by Chorin and is a mollified particle carrying the local circulation. Vortex blobs describe a smooth vorticity field and are ideal because of it does not cause singularity issues when particles approach each other.

!!! check for consistency, continuity !!!

2.2.1 Biot-Savart law

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

$$\mathbf{u} = \mathbf{u}_\omega + \mathbf{u}_\phi, \quad (2.8)$$

where \mathbf{u}_ω is the rotational component of the velocity and \mathbf{u}_ϕ is the irrotational component, solenoidal and potential velocity respectively. In an unbounded flow we have \mathbf{u}_ϕ equal to the free-stream velocity \mathbf{u}_∞ . 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

$$\mathbf{u}_\omega = \mathbf{K} \star \omega, \quad (2.9)$$

where the \star represents convolution of the 2-D kernel \mathbf{K} given by

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

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

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

where α_p is the estimate of the circulation around the particle \mathbf{x}_p with core size σ . We must not that ω^h is an approximately equal to ω of the fluid due to the discretization.

The discrete form of the velocity is therefore written as

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

Thus the discrete vorticity field is an N -body problem inducing velocity on each and implicitly evolving the vorticity field. This is one of the advantage of the vortex particle method as there are many ways to efficiently treat the problem. The N -body problem can be parallelized and can be accelerated using fast summation methods such as FMM, see ??.

However, like all N -body problem, equation 2.10 has a singularity when the particles approach each other and can result in numerical instability. To overcome this we can mollify the kernel, removing the singularity.

2.2.3 Mollified vortex kernels

A vortex particle with a mollified core, non-zero core-size, is referred to as vortex blobs. The advantage of the vortex blobs is that the with a smooth distribution of the vorticity, the singularity disappears and so numerical instability does not happen when blobs get too close to each other. An ideal choice for a cutoff function is a Gaussian distribution, figure 2.2.

Gaussian kernels satisfy the requirement for smooth distribution and decays quickly and is defined as

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

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

$$\mathbf{K}_\sigma = \mathbf{K} \star \zeta_\sigma. \quad (2.14)$$

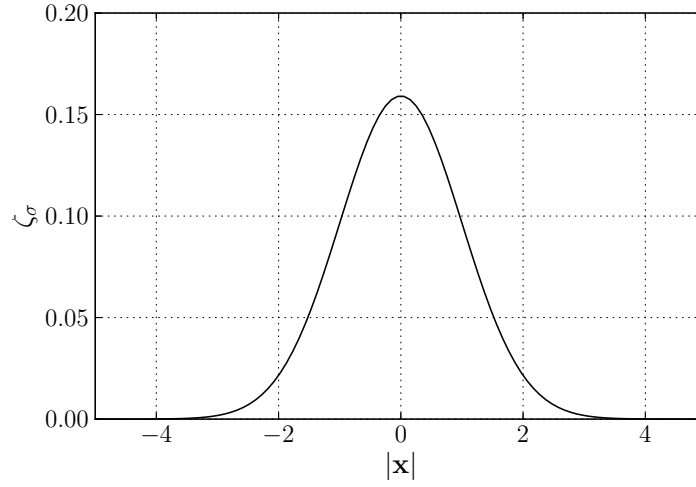


Figure 2.2: Vortex blob with Gaussian distribution: $[k = 2, \sigma = 1.0]$

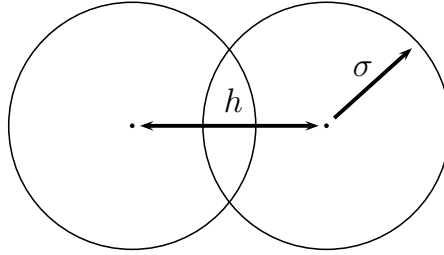


Figure 2.3: Vortex blob with overlap σ/h

The mollified vorticity field, represented by vortex blobs is given as

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

now representing the mollified vorticity field and equivalently, the mollified velocity field is given as

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

Koumoutsakos and Chorin [14], have shown that for proper communication between the particle, the particle needs to overlap,

$$\text{overlap} = \frac{\sigma}{h}, \quad (2.17)$$

where h is the nominal particle spacing, figure 2.3. If the particles fail to overlap, vortex blobs will also fail to recover the vorticity field. Such problems occurs when blobs are clustered due to high flow strain, leading to lagrangian grid distorting and must be treated, see section 2.3.1.

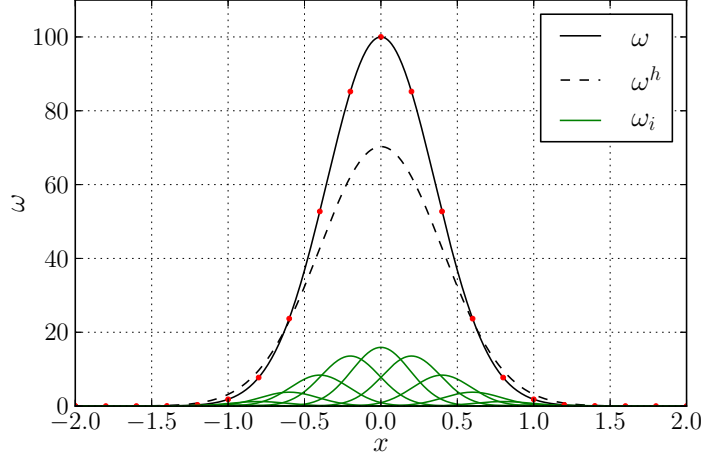


Figure 2.4: Mollified vorticity field of an arbitrary vorticity function with overlap = 1.0, $\sigma = 0.19$, $h = 0.19$. Vortex blob strength has been assigned by equation 2.18, sampling at exact vorticity [•, red dot]. Figure depicts exact vorticity distribution ω [—, solid], vorticity field of each blob ω_i [—, green dashed], the mollified vorticity field ω^h [- -, dashed].

2.2.4 Vortex blob initialization

Now the question arises on how to initialize the particle's circulation strengths α_p . A common approach that is used is to estimate the particles strength is to say that

$$\alpha_p = \omega_p \cdot h^2. \quad (2.18)$$

This might seem like a valid assumption as the circulation of a given area is the integral of the vorticity in the area, equation 2.2, however this is no longer valid when regularizing the vorticity field using mollified gaussian kernels, equation 2.15. Barba and Rossi [2], 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 in the evaluated vorticity field, figure 2.4.

Another way of viewing this characteristic is say the conservation of circulation is only valid globally, but not locally. A common standard for recovering the initial vorticity field is perform the Beale's method [4].

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 recovery of the vorticity field is done by performing a discrete deconvolution,

$$\sum_j^N \beta_j \zeta_\sigma(\mathbf{x}_i - \mathbf{x}_j) = \omega_i, \quad (2.19)$$

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 a N system of equations. Multiplying both sides with the area associated to the blobs, we get

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

where

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

is a $N \times N$ matrix containing the weights of the influence of each particle on each other. This matrix can be constructed by setting the Γ to one and determine the induced vorticity on each other. Furthermore, we see that it is not feasible to directly invert the matrix when we have large set of blobs but most importantly as the matrix \mathbf{A} is severely ill-conditioned [45], it should not be directly inverted. Beale's proposition to this problem was to iteratively solve for the solution,

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

We see that with just two iterations, the error between the mollified and exact vorticity field reduces drastically, figure 2.5. Koumoutsakos and Cottet [14], had shown that there was a drastic improvement in the velocity with just two to three iterations. However, we see that the cell vorticity of the blobs, directly evaluated from the particle strengths, equation 2.18, are more peaky and no longer matches the exact vorticity.

During the hybrid coupling algorithm, we see that this is the central source of coupling error between the eulerian and the lagrangian method, section ???. When performing the hybrid coupling, we need to recover the vorticity field transfered from the eulerian domain to the lagrangian domain in every step. So, beale's correction is not a viable solution for the hybrid method. Thus there is a need for an alternate method of recovering the vorticity field.

!!! add the reference to hybrid !!!

Convergence of particle discretization

An alternate, temporary method to reduce the gaussian blurring of the vorticity field is to reduce the overlap (i.e. increase the overlap ratio) of the vortex blobs and to the increase the spatial resolution.

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 blobs have overlap = 1 and so the blob core-size σ is equal to h . We see that as you reduce the size of the blob and increase the number of particles, the mollified vorticity converges to the exact vorticity. Therefore, an alternate method of reducing the gaussian blurring is to increase the spatial resolution.

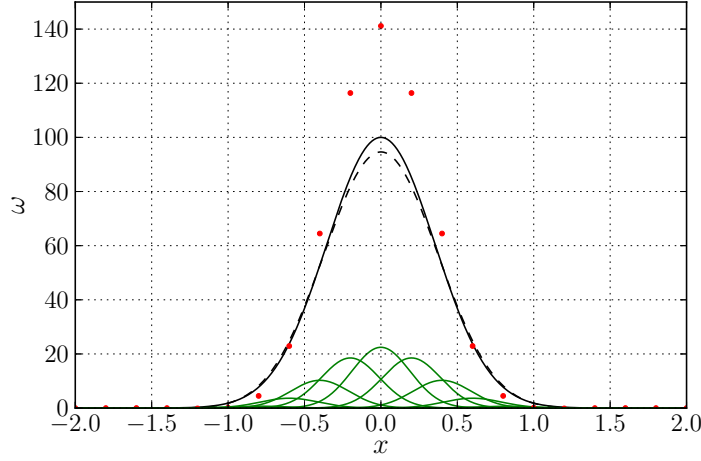


Figure 2.5: Mollified vorticity field after two Beale's iteration, overlap = 1.0, $\sigma = 0.19$, $h = 0.19$. Figure depicts exact vorticity distribution ω [—, solid], vorticity field of each blob ω_i [—, green dashed], the mollified vorticity field ω^h [- -, dashed].

Furthermore, we could also adjust the overlap of the blobs, figure 2.6b. The σ and h of the blob is 0.08 and we see that increasing overlap number (i.e reducing the overlap), helps us to recover the original vorticity field. However, as explained by Koumoutsakos [14], if the overlap is too low, we lose the smooth recovery of the vorticity field. This is apparent when overlap = 2.0, where we see that the mollified vorticity field is fluctuation.

Therefore, for the hybrid coupling, we set overlap = 1.0 and maximize the spatial resolution at the coupling zone.

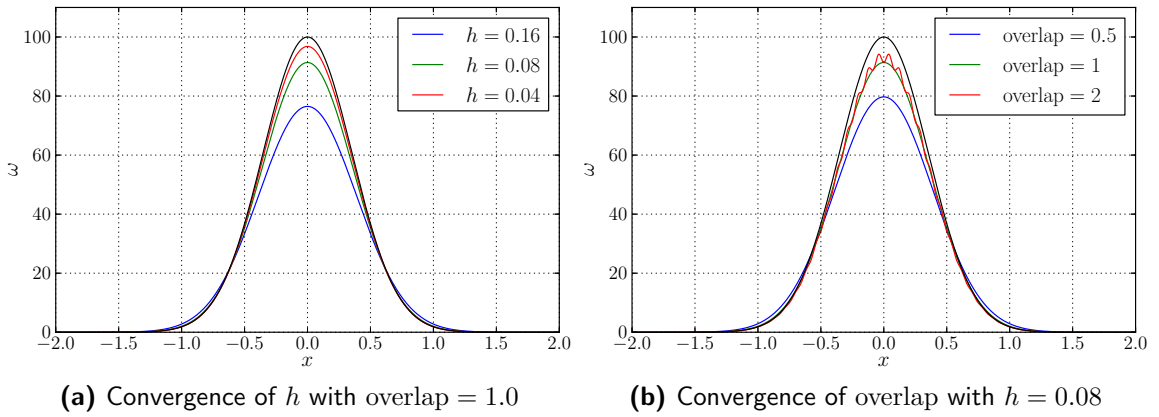


Figure 2.6: Convergence of vorticity by modifying the spatial resolution. Figure depicts exact vorticity field ω with [—, black] and various resolutions.

2.3 Convection of vortex blobs

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

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{u}(\mathbf{x}_p), \quad (2.23)$$

with

$$\frac{d\alpha_p}{dt} = 0. \quad (2.24)$$

As the diffusion is done at the next sub-step, we have to ensure that the circulation is conserved.

2.3.1 Remeshing scheme: Treating lagrangian grid distortion

During the convection step, we see that another source of error in the vorticity field is the lagrangian grid distortion. As we have seen before, when the vortex blobs fails to overlap, we are no longer able to reconstruct the correct the vorticity field, figure 2.6b. During the convection, due to the high strains in the fluid, the vortex blobs tent to clump together and creates regions where no vortex blobs are found, reducing the overlap of the blobs, figure 2.7.

We see that due to clustering of the vortex blobs, it fails to reproduce the correct vorticity field. A common strategy to overcome this problem is to remesh the vortex blobs to a uniform grid, so that we have a continuous vorticity field.

However, when transferring the vorticity from the old deformed grid to the new lagrangian uniform grid, we must satisfy the conservation laws of vorticity field. The interpolation methods is based on the conservation of linear impulse which directly implies the

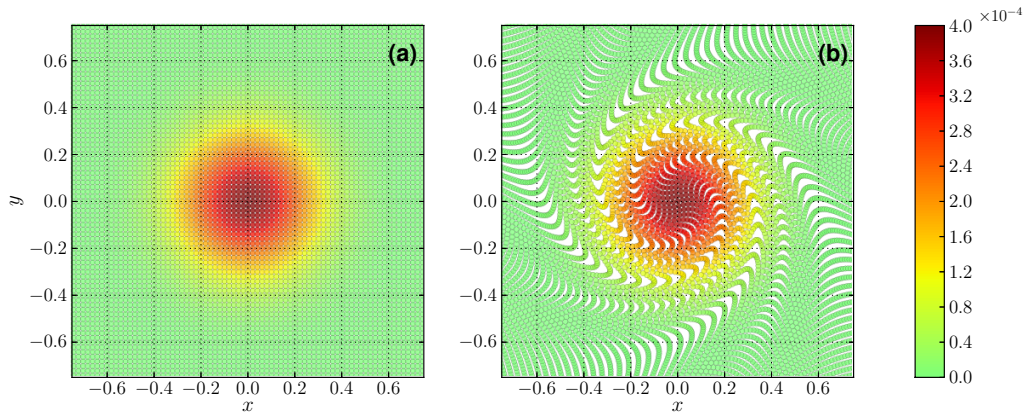


Figure 2.7: Lagrangian distortion of the vortex blobs after 100 steps. The initial vorticity field $\omega(\mathbf{x}, 0) = \exp(-12|\mathbf{x}|)$ with $\Delta t = 0.1$, $\sigma = 0.02$ and overlap = 1.0. Figure depicts (a) the initial and (b) the final distribution of the vortex blobs.

conservation of the total circulation [14]. The transfer of the particle strengths is given as,

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

where the strengths of the particles $\tilde{\alpha}_q$ of the distorted lagrangian grid \tilde{x}_q is transferred to the regular lagrangian grid x_p using the interpolation kernel, weighted W , giving us the remeshing particle strengths α_p . The transfer of strengths of one particles to it's interpolation nodes can be seen in figure 2.8.

M'_4 interpolation kernel

For lagrangian problem, we use the efficient interpolation kernel that has been used to reconstruct a smooth distribution interpolation, the M'_4 interpolation kernel, introduced by Monaghan [37]. In one dimension it is given as,

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

where

$$\xi = \frac{x_\nu - x_i}{h} \quad (2.27)$$

is the scale relative position of the particle x_ν to the i^{th} interpolation node x_i . The M'_4 is a third-order accurate piecewise smooth B-spline kernel, where $m = 4$ giving it 4 support nodes, figure 2.9. For the two dimensional problem that we have, the 2-D interpolation

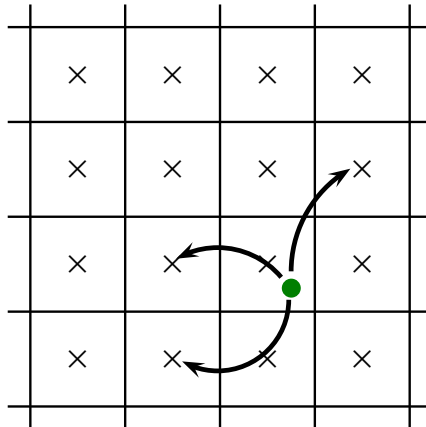


Figure 2.8: Remeshing of vortex blob (•, green) on the uniform grid defined by the (4×4) 2-D stencil.

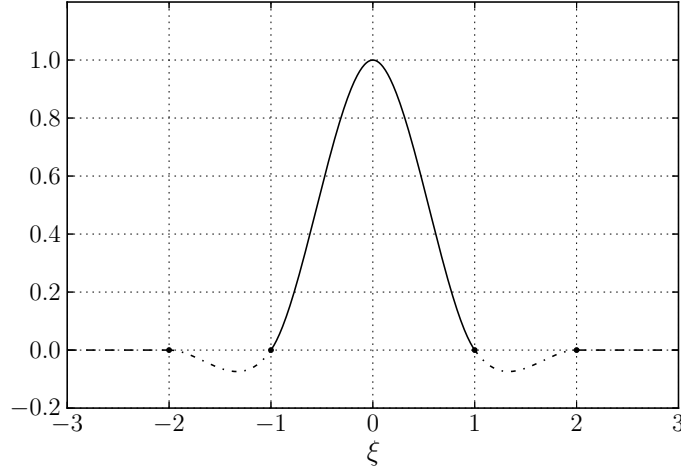


Figure 2.9: M'_4 interpolation kernel, a third-order piecewise smooth B-spline kernel by Monaghan [37]

formula is simply tensor product of the 1-D interpolation kernel equation 2.26, and results in $4^2 = 16$ support nodes, figure 2.8.

The interpolation kernel achieves the third-order accuracy as it also conserves the linear and the angular momentum of the vortex. Koumoutsakos [27] has investigated the drawback of the employing the remeshing strategy and have shown that there is approximately 4% decay in enstrophy of the flow due to sub-grid dissipation. Note that enstrophy \mathcal{E} , is defined as

$$\mathcal{E} = \frac{1}{2} \int_S \omega^2 dS \quad (2.28)$$

and is directly related to the kinetic energy of the fluid and gives an insight in the energy production and the dissipation of the fluid. Enstrophy is especially useful in turbulence flow investigation as it helps describe the energy cascade of the fluid.

2.4 Diffusion of Vortex Methods

So far, we have dealt with unbounded Euler flow, where we disregarded the viscosity of the flow. However, for real flow we must take in account of the diffusion of the vorticity. Chorin's approach to deal with the viscous term was to employ 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, equation 2.7. The equation can again be solved as a system of ODEs, similar to the convection step, where we say,

$$\frac{d\mathbf{x}_p}{dt} = 0, \quad (2.29)$$

with

$$\frac{d\alpha_p}{dt} = \nu \Delta \alpha_p. \quad (2.30)$$

Therefore in the diffusive step, we fix the position of the vortex blobs and only have to modify the strengths of the particles to mimic the diffusion process. Chorin initially employed a random walk method, which generates and disperse vorticity using pseudo-random number algorithm [10]. However, this method suffers some limitations in accuracy. Since then Particle Strength Exchange (PSE) method [16], has been a common approach that has been used to treat diffusion.

Particle Strength Exchange

The Particle Strength Exchange method, first proposed by Mas-Galic [16], showed that diffusion can be treated for a particle method isotropic and anisotropic viscosity by approximating the diffusion operator (laplacian) with an integral operator and discretizing the operator using particles. The PSE can be seen as circulation correction method, where during the diffusion step of the viscous splitting algorithm, the strengths of the particle are corrected such that it accounts for the diffusion.

Vortex Redistribution Method

An alternative method to simulate the diffusion is to use the Vortex Redistribution Method (VRM) [42]. The model simulates diffusion by distributing the fraction of circulation of the vortex blobs to each other satisfying the diffusion. The model is based on conserving the moments of the particles by satisfying a linear system of equations. The circulation of the particle are transfer to the nearby particles that are

$$h_\nu = \sqrt{\nu \Delta t_d} \quad (2.31)$$

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. Not that the diffusive time-step Δt_d is equal to the convective time-step Δt_c if the diffusion process is done during every time-step. However, we can easily adjust the diffusion time-step and perform diffusion at a multiple step of the convection.

This is vital as a VRM (and also the PSE) requires a search algorithm to determine the particles that are within the zone of influence. A direct evaluation required $\mathcal{O}(N^2)$ evaluation, however can be speed up to $\mathcal{O}(\log N)$ using search tree algorithm.

The downside of the this approach, as also for the standard remeshing approach is the global remeshing generates large computation data and scales with the number of particles N . Therefore for problems with large number of particles, a tree-structured remeshing would be more feasible strategy [52].

2.4.1 Modified remeshing for treating diffusion

From further investigation of the VRM, we see that it is similar to remeshing strategy used to counter the lagrangian distortion during the convection process. Therefore Ghoniem and Wee [50] has proposed to combine the remeshing and the diffusion. The application of this methodology was validated by Speck [45]. The diffusion is simulated by the

modifying the interpolation kernel of the remeshing process. The key advantage of the modified remeshing method is that now it deals with the uniform remeshing grid helping us eliminating the computational expensive research requirement. The second advantage as shown by the authors is the simplicity. The method only requires a slight modification to the original remeshing tool.

During remeshing, the heat equation is satisfied by transferring the correct fraction of circulation to produce the proper amount of diffusion. The M'_4 kernel was modified to treat the diffusion and is given by:

$$M'_4(\xi, c) = \begin{cases} 1 - \frac{5\xi^2}{2} + \frac{3|\xi|^3}{2} - c^2(2 - 9\xi^2 + 6|\xi|^3) & |\xi| < 1, \\ \frac{1}{2}(2 - |\xi|)^2(1 - |\xi|) - c^2(2 - |\xi|)^2(1 - 2|\xi|) & 1 \leq |\xi| < 2, \\ 0 & 2 \leq |\xi|, \end{cases} \quad (2.32)$$

where

$$c^2 = \frac{\nu \Delta t_d}{h^2}, \quad (2.33)$$

and corresponds to the transfer quantity for diffusion. The addition terms in the interpolation kernel accounts for the diffusion process. When $c \rightarrow 0$, the interpolation kernel turns to the classical non-diffusion kernel.

Similar to the VRM, we can perform the remeshing at a given multiple step k_d of the convection step. However, we have an additional constraint on the diffusion time-step Δt_d , equation 2.33. Wee and Ghoniem [50] also investigated the error growth and the stability properties of the interpolation kernel in the Fourier space and have determined that for M'_4 interpolation kernel, we must satisfy

$$\frac{1}{6} \leq c^2 \leq \frac{1}{2}. \quad (2.34)$$

to ensure amplification factor and the phase error does not grow. This will ensure the stability of the problem and will suppress any spurious oscillations and ensure that it is a non-negative interpolation kernel with non-negative redistribution fractions.

However, we see that constraint on c^2 not only imposes a maximum Δt_d but also a constraint on the minimum Δt_d . This means that in scenarios, the $\Delta t_d = k_d \cdot \Delta t_c$ and this might introduce sudden change in vorticity field. Therefore, during the coupling of the lagrangian and eulerian domains, we would like to diffuse the vorticity after every convection scheme.

Simple redistribution scheme

The simple redistribution scheme based on the Shankar and Van Dommelen [42], developed by Tutty [48], makes it possible to remesh and diffuse the vorticity after every convection step,

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

where W_{ki}^n is the fraction of circulation from vortex blob k transferred to vortex blob i by diffusion during the time step n . The fractions W_{ki}^n are calculated by conserving vorticity, center of vorticity, linear, and angular momentum of the vortex blobs [48]. In two dimensional problem the redistribution fractions are simple tensors products of the x, y one dimensional redistribution fractions,

$$W_{kl} = F_k G_l, \quad k = i - 1, \dots, i + 2, \quad l = j - 1, \dots, j + 2 \quad (2.36)$$

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

$$F_k = (1 - \Delta) \cdot f_k + \Delta \cdot g_k, \quad k = i - 1, \dots, i + 2 \quad (2.37)$$

having a four point stencil, figure 2.10. The basis solution of redistribution are

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

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

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

and

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

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

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

where ξ is given by equation 2.27, $\xi_1 = \xi - 1$ are the distances between the k^{th} stencil nodes x_k and the vortex blob that is to be diffused with $x_i \leq x_\nu \leq x_{i+1}$. Note, f_k and

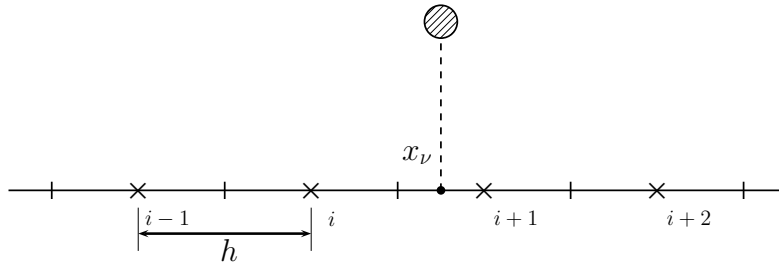


Figure 2.10: One dimensional, Simple redistribution scheme, diffusing the vortex blobs to the four stencil points ($k = i - 1, \dots, i + 2$) with vortex blob positioned at $x_i \leq x_\nu \leq x_{i+1}$ and nominal blob spacing h .

g_k is zero for all the other k . In the above equation, h_ν is the characteristic diffusion distance over the time Δt_d ,

$$h_\nu = \sqrt{\Delta t_d \cdot \nu}. \quad (2.40)$$

The only constraint imposed for positive redistribution fraction is

$$\frac{h_\nu}{h} < \frac{1}{\sqrt{2}}. \quad (2.41)$$

giving us the maximum time-step size constraint of

$$\Delta t_d < \frac{h^2}{2\nu}. \quad (2.42)$$

So now, we are able to perform diffusion, together with the 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 addition constraint of the boundary to the simulation to simulate any flow about a geometry. From Helmholtz decomposition, we can have decompose the velocity field to the rotation and the irrotation component, equation 2.8. With the Helmholtz decomposition, we can use the potential component to prescribe the boundary conditions at the solid wall boundary,

$$\mathbf{u}_\phi = \nabla \Phi. \quad (2.43)$$

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

$$\mathbf{u}_b \cdot \hat{\mathbf{n}} = (\mathbf{u}_w + \nabla \Phi) \cdot \hat{\mathbf{n}}, \quad (2.44)$$

and is defined as enforcing the no-through flow at the solid boundary wall, moving at \mathbf{u}_b . Note that the $\hat{\mathbf{n}}$ is the normal vector of the solid boundary. A classical 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 [29], have suggested to enforce the boundary conditions through vortex sheets. The alternative approach of enforcing the solid boundary condition is 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

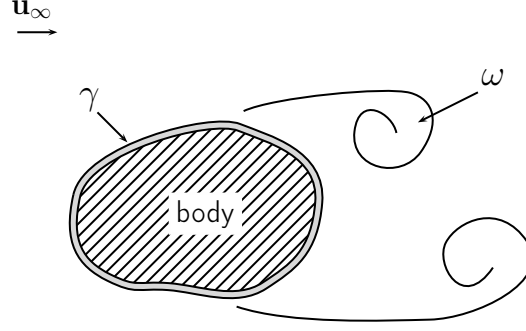


Figure 2.11: Extended vorticity field consisting of vorticity in the fluid and vortex sheet distribution around the body.

discontinuity in the velocity field can be considered as vortex sheet. Therefore, now to enforce the boundary conditions of the solid wall, we must satisfy the no slip velocity at the boundary,

$$\mathbf{u} \cdot \hat{\mathbf{s}} = \mathbf{u}_b \cdot \hat{\mathbf{s}}. \quad (2.45)$$

Koumoutsakos [30] only relies on the vortex sheet to enforce the no-slip velocity as no-slip boundary condition directly satisfies the no-through boundary conditions, also known as the linked boundary conditions. This was also proven by Shiels [43] and further investigated by Cooper, Mar and Barba [13] and shown that vortex sheet is the only necessary singularity element required to satisfy both the boundary conditions. Therefore, enforcing the no-slip boundary condition directly satisfies the no-through constraint at the surface.

2.5.1 Boundary integral equations

Thus the decomposed velocity field can be summarized as

$$\mathbf{u} = \mathbf{u}_\omega + \mathbf{u}_\gamma + \mathbf{u}_\infty \quad (2.46)$$

where the \mathbf{u}_γ denotes the velocity field induced by the vortex sheet. Applying the no-slip boundary conditions, we can say that

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

where $\mathbf{u}_{\text{ext}} = \mathbf{u}_\omega + \mathbf{u}_\infty$ is the velocity field induced from domain external of the body and the vortex sheet. For the tangential boundary condition, for a well-conditioned system of equations, the induced velocity from the vortex sheet can be summarized as

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

The equation states that underneath the vortex sheet, we must have a vortex sheet inducing the velocity field \mathbf{u}_γ to counter the slip velocity $\mathbf{u}_{\text{slip}} = (\mathbf{u}_{\text{ext}} - \mathbf{u}_b)$ such that we satisfy the no-slip kinematic boundary conditions. Note that, \mathbf{u}_γ of equation 2.48, is negative of equation 2.47 as we looking under the vortex sheet. Koumoutsakos [30], expressed the

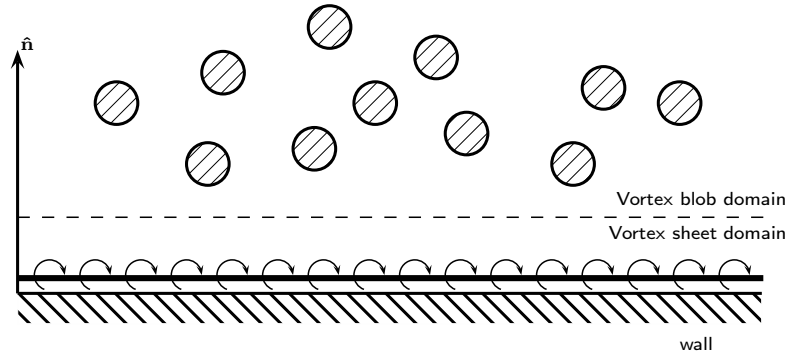


Figure 2.12: Extended vorticity field: Vortex sheet being an extension to the vorticity field (resolved by the vortex blobs) which is able to fully resolve the boundary vorticity

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,

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

The Left Hand Side (LHS) of the equation states at the point s , the velocity discontinuity is due to the vortex sheet at the point and the induced velocity of all the other vortex sheet acting on the point.

However, the equation 2.49 is not unique and accepts arbitrary solution for the vortex sheet strengths, therefore we must impose an additional constraint on the strength of the vortex sheet. The additional constraint is a constraint on the net circulation of the vortex sheet. As the vortex sheet is considered as an extension to the vorticity field, we could say that the net circulation of the vortex panels is the circulation that was not captured by the vortex blobs, figure 2.12. As vortex blobs are not available very-near the body, it is not possible for the vortex blobs to resolve the high gradient vorticity at the boundary. Therefore, all the vorticity in the vortex sheet domain must be captured by the vortex sheet.

When considering a moving boundary, we must also take into account of the circulation of the body due to its motion. The net circulation of the problem is given as

$$\Gamma = \Gamma_{\omega} + \Gamma_{\gamma} + \Gamma_b = 0, \quad (2.50)$$

where Γ_{ω} is the circulation of the vorticity in the fluid (i.e. the vortex blobs), Γ_{γ} is the wall bounded circulation (i.e. vortex sheets),

$$\Gamma_{\gamma} = \oint_S \gamma(s) ds, \quad (2.51)$$

and the Γ_b is the circulation of the moving body. The circulation inside the body can be considered also be considered as an extension to the vorticity field, where the body is filled with uniform vorticity due to the rotation of the body. Therefore the circulation of

a moving body can be calculated simply by integrating the “vorticity” inside the body

$$\Gamma_b = \iint_{body} \nabla \times \mathbf{u}_b \, dA. \quad (2.52)$$

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

$$\Gamma_\gamma = -(\Gamma_\omega + \Gamma_b). \quad (2.53)$$

Now, in a pure lagrangian, we must transfer the vorticity generated from the body to the fluid. This is typically done by diffusing the vorticity of the vortex on the fluid, however in our hybrid coupling method, we can use the eulerian domain to introduce the vorticity into the fluid. The eulerian domain acts as the near-wall solver [15], and the strengths of the particles is interpolated from the eulerian domain, section ??!!! **CiTe Hybrid section !!!**

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 equations which can be used to determine the M unknowns of the strength of the vortex panels. The method referred to as the panel method has been greatly summarized by Katz and Plotkin [26].

Katz and Plotkin have shown several types of panel distributions with various orders of accuracy, from 0th order point vortex or up to 2nd order varying panel strength curved distribution. For this project, we have used a constant-strength vortex distribution that discretizes the vortex sheet into straight segments, classified as Constant-Strength Vortex Method (CSVM).

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

$$\underbrace{\begin{pmatrix} -\frac{1}{2} & a_{12} & \cdots & a_{1M} \\ a_{21} & -\frac{1}{2} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & -\frac{1}{2} \end{pmatrix}}_{\mathbf{A}_{MM}} \underbrace{\begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_M \end{pmatrix}}_{\vec{\gamma}} = \underbrace{\begin{pmatrix} \text{RHS}_1 \\ \text{RHS}_2 \\ \vdots \\ \text{RHS}_M \end{pmatrix}}_{\vec{\text{RHS}}} \quad (2.54)$$

where \mathbf{A}_{MM} contains the weights of the influence of the vortex panels $\vec{\gamma}$ on each other and the RHS is

$$\text{RHS} = \mathbf{u}_{\text{slip}} \cdot \hat{\mathbf{s}} \quad (2.55)$$

is the boundary condition to the system of equations. In addition, we have another constraint on the net circulation of the vortex panels and in discrete form we say that,

$$\sum_i^M \gamma_i \Delta s = \Gamma_\gamma \quad (2.56)$$

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

$$\underbrace{\begin{pmatrix} -\frac{1}{2} & a_{12} & \cdots & a_{1M} \\ a_{21} & -\frac{1}{2} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & -\frac{1}{2} \\ \Delta s_1 & \Delta s_2 & \cdots & \Delta s_M \end{pmatrix}}_{\mathbf{B}_{(M+1)M}} \underbrace{\begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_M \end{pmatrix}}_{\vec{\gamma}} = \underbrace{\begin{pmatrix} \text{RHS}_1 \\ \text{RHS}_2 \\ \vdots \\ \text{RHS}_M \\ \Gamma_\gamma \end{pmatrix}}_{\vec{\text{RHS}}} \quad (2.57)$$

However, as we have an additional constraint on the net strength of the panels, we have $M + 1$ set of equations with M unknowns giving us a overdetermined problem. The approach to solve such a problem is either by using a Least-Square solution method (**LSTSQ**), or introducing a new unknown or as used by Katz, eliminating an equation. Enforcing the no-slip boundary condition at each panel location is our vital criteria and therefore, elimination of an equation is not a viable strategy and so the LSTSQ method was utilized for the project.

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 as per definition. To solve the strengths of the panel problem, we enforce the dirichlet velocity boundary conditions at the collocation point x_{cp} , which is located just below the vortex sheet, figure 2.13b. The coefficient a_{ij} of the influence matrix \mathbf{A} is defined as

$$a_{ij} = \hat{\mathbf{u}}_{ij} \cdot \hat{\mathbf{t}}_i, \quad (2.58)$$

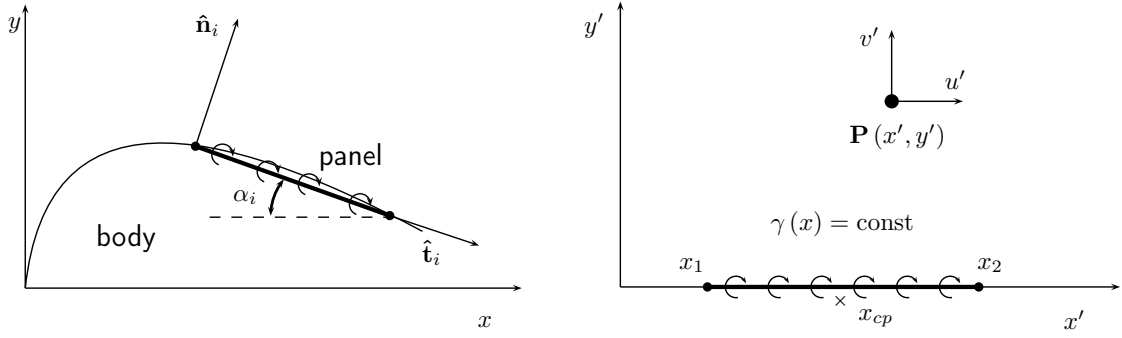
and is the tangential influence coefficient of j^{th} panel on the i^{th} panel. Therefore, induced velocity $\hat{\mathbf{u}}_{ij} = (\hat{u}, \hat{v})_{ij}$ is the unit induced velocity of the j^{th} panel on to the collocation points of i^{th} panel, where the vortex panels have unit strength (i.e $\hat{\gamma}_i = 1$). In general, the induced velocity of the vortex panels are calculated in the local panel coordinates, figure 2.13, and the transformation from the local panel coordinate (x', y') to the global coordinates systems (x, y) is given by

$$\begin{bmatrix} u_{ij} \\ v_{ij} \end{bmatrix} = \begin{bmatrix} \cos \alpha_j & \sin \alpha_j \\ -\sin \alpha_j & \cos \alpha_j \end{bmatrix} \cdot \begin{bmatrix} u'_{ij} \\ v'_{ij} \end{bmatrix} \quad (2.59)$$

The induced velocity of the vortex panel j on the collocation point i is given as

$$u'_{ij} = \frac{\gamma_j}{2\pi} \left[\tan^{-1} \frac{y'_i - y'_{j,2}}{x'_i - x'_{j,2}} - \tan^{-1} \frac{y'_i - y'_{j,1}}{x'_i - x'_{j,1}} \right], \quad (2.60a)$$

$$v'_{ij} = -\frac{\gamma_j}{4\pi} \ln \frac{(x'_i - x'_{j,1})^2 + (y'_i - y'_{j,1})^2}{(x'_i - x'_{j,2})^2 + (y'_i - y'_{j,2})^2} \quad (2.60b)$$



(a) Panel discretization of the body in the global cartesian coordinates system (x, y) and local panel coordinates rotated by α_i . **(b)** Constant strength vortex panel in the local panel coordinate system (x', y') inducing velocity on the point P .

Figure 2.13: Vortex panel's global **(a)** and local **(b)** coordinates system definition as defined by Katz and Plotkins [26].

where $(x'_1, y'_1)_j$ and $(x'_2, y'_2)_j$ are the coordinates of the panel start and end point in its local panel coordinate system. Note that the self-induction of the vortex panel (when $i = j$), the influence coefficient becomes $a_{ij} = -1/2$, and can be seen in the diagonal terms of equation 2.54. So, by taking $\hat{\gamma}_i = 1$, we are able to construct the influence coefficients of the influence matrix \mathbf{A} and can solve the strengths of the panel,

$$\mathbf{B} \cdot \gamma = \text{RHS}, \quad (2.61)$$

using the LSTSQ method.

If we are dealing with multiple geometries, figure 2.14, the panel method can be easily

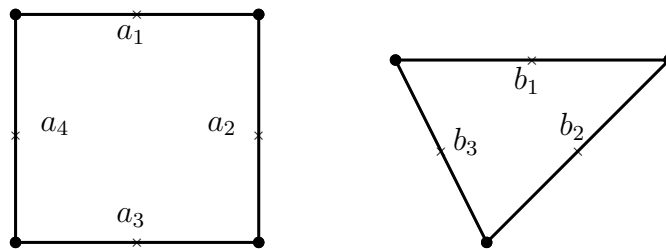


Figure 2.14: twoPanelBodies

extended by constructing a global influence matrix,

$$\underbrace{\begin{pmatrix} c_{a_1 a_1} & \cdots & c_{a_1 a_N} & c_{a_1 b_1} & \cdots & c_{a_1 b_M} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{a_N a_1} & \cdots & c_{a_N a_N} & c_{a_N b_1} & \cdots & c_{a_N b_M} \\ c_{b_1 a_1} & \cdots & c_{b_1 a_N} & c_{b_1 b_1} & \cdots & c_{b_1 b_M} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{b_M a_1} & \cdots & c_{b_M a_N} & c_{b_M b_1} & \cdots & c_{b_M b_M} \\ \Delta s_{a_1} & \cdots & \Delta s_{a_N} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \Delta s_{b_1} & \cdots & \Delta s_{b_M} \end{pmatrix}}_{\begin{pmatrix} C_{aa} & C_{ab} \\ C_{ba} & C_{bb} \\ \Delta s_a & 0 \\ 0 & \Delta s_b \end{pmatrix}} \begin{pmatrix} \gamma_{a_1} \\ \vdots \\ \gamma_{a_N} \\ \gamma_{b_1} \\ \vdots \\ \gamma_{b_M} \end{pmatrix} = \underbrace{\begin{pmatrix} \text{RHS}_{a_1} \\ \vdots \\ \text{RHS}_{a_N} \\ \text{RHS}_{b_1} \\ \vdots \\ \text{RHS}_{b_M} \\ \Gamma_{\gamma,a} \\ \Gamma_{\gamma,b} \end{pmatrix}}_{\begin{pmatrix} \text{RHS}_a \\ \text{RHS}_b \\ \Gamma_{\gamma,a} \\ \Gamma_{\gamma,b} \end{pmatrix}} \quad (2.62)$$

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

2.5.3 Error analysis of panel method

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

To test the solution of the vortex panels with the analytical solution, the problem was first run 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 norm 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; For higher resolution, this will not be there. In order to determine the accuracy of the solution, the velocity field of the panel solution was compared with

Table 2.1: Panel study parameters

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

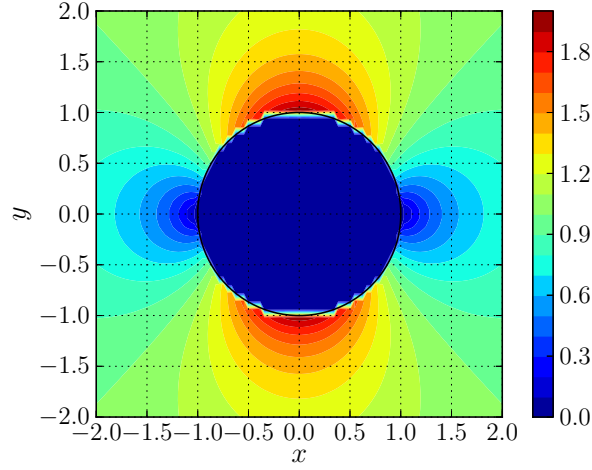


Figure 2.15: Panel method solution: Potential flow velocity field around unit cylinder. The figure depicts $\|\mathbf{u}\|$ with zero velocity inside the body.

the analytical solution. The analytical velocity field around a cylinder is given as,

$$u_r = u \left(1 - \frac{R^2}{r^2} \right) \cos \theta \quad (2.63a)$$

$$u_\theta = -u \left(1 + \frac{R^2}{r^2} \right) \cos \theta \quad (2.63b)$$

where u_r and u_θ are the radial and the angular velocity respectively. The equation 2.63 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 panel was compared with this analytical solution along the y -axis from $y = 0$ to $y = 10$, figure 2.16a. 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 is correct because the potential flow solution has

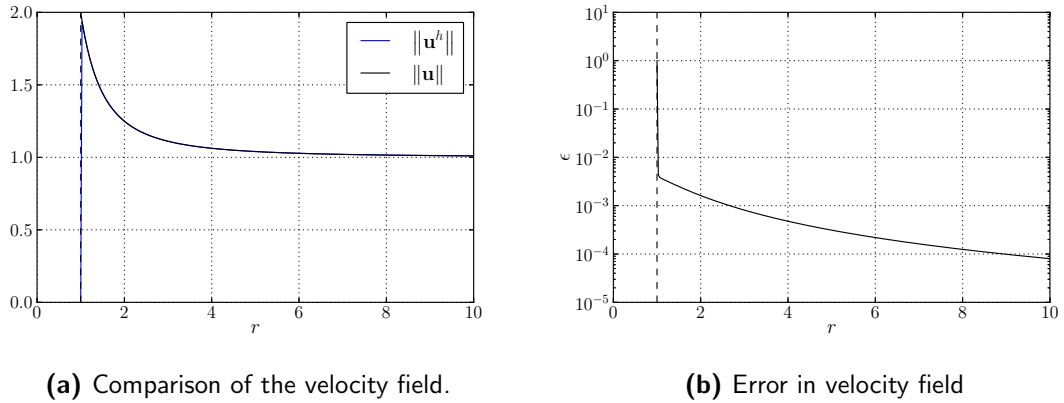


Figure 2.16: Comparison of the velocity field along the y -axis $0 \rightarrow 10$. Figure (a) shows both the solutions, the numerical $\|\mathbf{u}^h\|$ [solid blue, —] and the analytical solution [solid black, —]. Figure (b) shows the relative error ϵ between the solution, given by equation 2.64

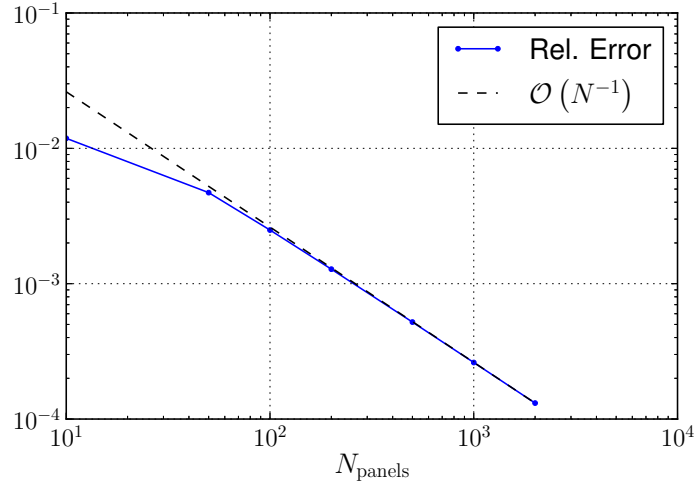


Figure 2.17: Convergence plot of the constant-strength straight vortex panels. The figure depicts the reduction in error at $\mathcal{O}(N)$.

a slip velocity (i.e. non-zero velocity) at the surface of the body, whereas the vortex panels solve for a no-slip boundary condition at the collocation points of the surface. This explains the sudden drop of the velocity from two to zero at the surface.

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

$$\epsilon = \frac{\|\mathbf{u} - \mathbf{u}^h\|}{\|\mathbf{u}\|} \quad (2.64)$$

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 $\epsilon = 5 \times 10^{-3} \rightarrow 8 \times 10^{-5}$ as we go from $r = 1 \rightarrow 10$. This behavior of the error tells us that the solution of the constant-strength vortex panels gets more accurate as we go further away from the panels; right next to the panels, we have the largest error. This is because the vortex panels discretize the body using a first-order approximation (straight panels) and also discretize the vortex sheet strength using a first-order approximation.

Convergence analysis

The convergence analysis of the vortex panels shows that the panels that we are employing are indeed a first-order method as the error converges at $\mathcal{O}(N)$, figure 2.17.

2.6 Validation of Lagrangian method

Note that during this validation study, we are unable to validate the proper handling of the Lagrangian boundary conditions. This is because, we have only defined the vorticity generation from the boundary in the Eulerian domain and we neglected the vorticity

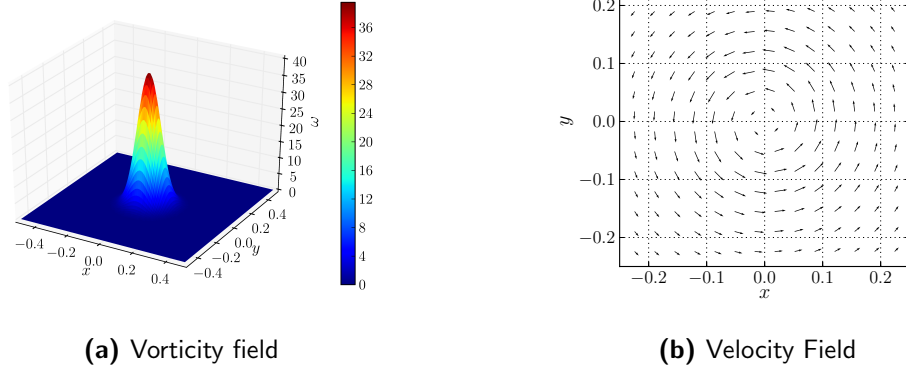


Figure 2.18: Lamb-Oseen **(a)** vorticity distribution, and **(b)** velocity distribution. $\Gamma_c = 1$, $\tau = 2 \times 10^{-3}$.

production from the vortex panels, due to the natural of the coupling method, we are unable to validate the proper coupling of vortex panels with vortex blobs. Therefore, validation study for the Lagrangian method is done separately, ensuring no-through flow on vortex panels, and ensuring proper handling of vorticity with vortex blobs.

In order to verify and validate the vortex blobs, we use the analytical solution of the Lamb-Oseen vortex evolution. The validation of the lagrangian method was done by comparing the results of the Lamb-Oseen vortex and the analyzing the convection of the Clercx-Bruneau dipole.

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 [1] and Oseen [2]. The vorticity distribution ω of the core at a given time is defined as,

$$\omega(\mathbf{x}, \tau) = \frac{\Gamma_c}{4\pi\tau} \exp\left(-\frac{r^2}{4\tau}\right), \quad (2.65)$$

and is a function of core strength Γ_c , the scaled viscous time τ , and distance from the core center r . At $\tau = 0$, the Lamb-Oseen vorticity distribution is a Dirac delta function and diffusion as time progresses. Figure 2.18a shows the vorticity distribution of the Lamb-Oseen vortex with core strength $\Gamma_c = 1$ at scaled viscous time $\tau = 2 \times 10^{-3}$. Note, the scaled viscous time is defined as $\tau \equiv \nu t$. The velocity field of this core is given as

$$u_\theta = \frac{\Gamma_c}{2\pi r} \left[1 - \exp\left(-\frac{r^2}{4\tau}\right) \right] \quad (2.66a)$$

$$u_r = 0 \quad (2.66b)$$

where u_θ is the circumferential velocity and the radial velocity u_r is zero. This can be seen on figure 2.18b.

Table 2.2: Summary of the Lamb-Oseen vortex evolution study parameters. Table shows the parameters of Tutty's diffusion method [48]

Parameters	Value	Unit	Description
Γ_c	1	$\text{m}^2 \text{s}^{-1}$	Core strength
Ω	$[-0.5, 0.5]^2$	m	Initial particle domain
\mathbf{u}_∞	$[0, 0]$	m s^{-1}	Free-stream velocity
ν	5×10^{-4}	$\text{kg s}^{-1} \text{m}^{-1}$	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
$\Delta t_c = \Delta t_d$	0.01	s	Diffusion and convection time step size
$N_{\text{t-steps}}$	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 occurrence
$f_{\text{popControl}}$	1	-	Population control occurrence
$\Gamma_{\text{threshold}}$	$(1 \times 10^{-14}, 1 \times 10^{-14})$	$\text{m}^2 \text{s}^{-1}$	Population control threshold

To determine whether the lagrangian method performs according to theory, this analytical solution will be used to verify and then validate the solution.

2.6.1 Evolution of Lamb-Oseen vortex

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 $\Gamma_c = 1$ and $\nu = 5 \times 10^{-4}$ at $\tau_0 = 2 \times 10^{-3}$. The vorticity field was discretized from (x, y) - domain $[-0.5, 0.5]^2$ using vortex blobs with $\sigma = 0.01$, and overlap = 1, giving it a fixed blob spacing $h = 0.01$. We employed the standard initialization method of vortex blobs from section 2.2.4. However, as explained before, we have to take in account of the gaussian blurring of the original vorticity field and thus posses a problem when evaluating the error with the analytical Lamb-Oseen solution.

Barba [3], had encountered the problem before when investigating with a Lamb-Oseen vortex. Her solution to the problem was to apply a “time-shift correction”, solving the problem of this very particular discretization of the diffusion equation using gaussian vortex blobs. Therefore, this is a special scenario and this approach cannot be used for other problems.

The “time-shift correction” is derived by determining the diffusion effect caused by the discretization of the diffusion equation using gaussian blobs (with $k = 2$). Barba determined that discretization of the diffusion equation reconstructs the vorticity field that has been diffused by $\sigma^2/2\nu$. So, in order to compare the results will have to shift the time by this value in order to compare with the analytical Lamb-Oseen solution. Therefore, when initializing the particles with a certain strength, we will have to reverse the time by

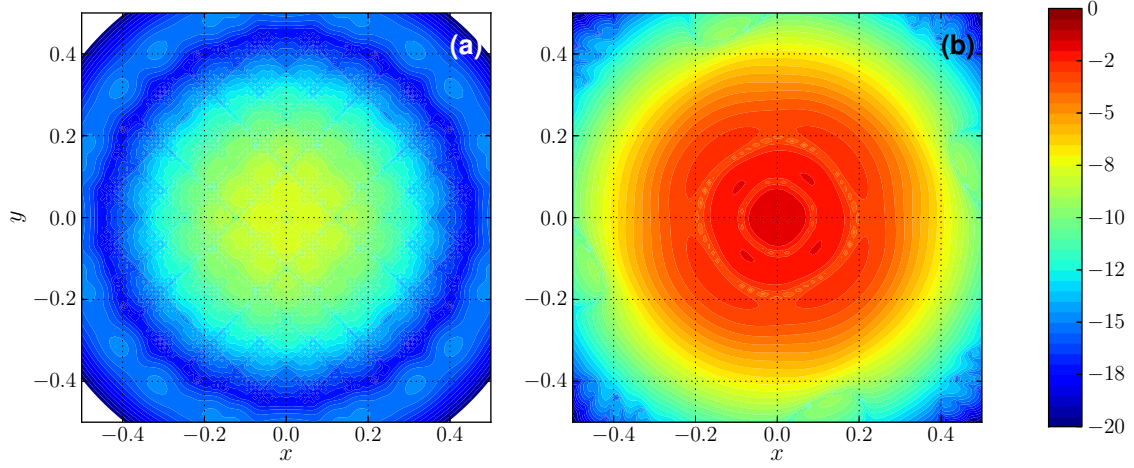


Figure 2.19: Relative error growth of Lamb-Oseen vorticity during the evolution (in logarithmic scale). The figure shows **(a)**, the initial relative error at $t_0 = 4$, and **(b)** the final relative error in vorticity at $t_f = 5$.

$\sigma^2/2\nu$, giving us the initial particles strengths of the Lamb-Oseen vorticity field as,

$$\alpha_0 = \omega_0 \cdot h^2 = \frac{\Gamma_c}{4\pi\nu(t - \sigma^2/2\nu)} \exp\left[-\frac{r^2}{4\nu(t - \sigma^2/2\nu)}\right]. \quad (2.67)$$

Using this method, we can investigate the error growth of the vortex blob method. The vortex blobs were convected and diffused according to the parameters in table 2.2. For the primary and the main investigation, we used the diffusion method proposed by Tutty [48]. The advantage of this approach is that 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), we convect and diffuse the vortex blobs at $\Delta t_c = \Delta t_d = 0.01$. The time integration of the problem is done using a 4th-order Runge-Kutta method (RK4). During the convection we will have to treat the Lagrangian distortion using the remeshing scheme discussed before. The vortex blobs are remeshed every step ($f_{\text{redist}} = 1$). Remeshing at every step is not necessary and is typically done every 10 iterations. However, as our diffusion scheme and hybrid method require a structured lattice of vortex blobs for efficient calculations, we remesh every step. The additional circulation processing we will have to do during the evolution of the vortex blobs is to remove unnecessary vortex blobs. To make the simulation as efficient as possible, it would be ideal to remove vortex elements where they are not needed. This is done by removing as many blobs as possible such that we do not violate the conservation of circulation to great extent. We place a cap on the maximum loss of total circulation, $\Gamma_{\text{threshold}} = 1 \times 10^{-14}$. This process is referred to as "population control" and is done at every iteration $f_{\text{popControl}} = 1$.

To verify where our Lagrangian scheme is function according to theory, we evaluated the error growth of the simulation with respect to the Lamb-Oseen analytical solution. Figure 2.19, shows the initial and the final relative error in vorticity. We see that initially

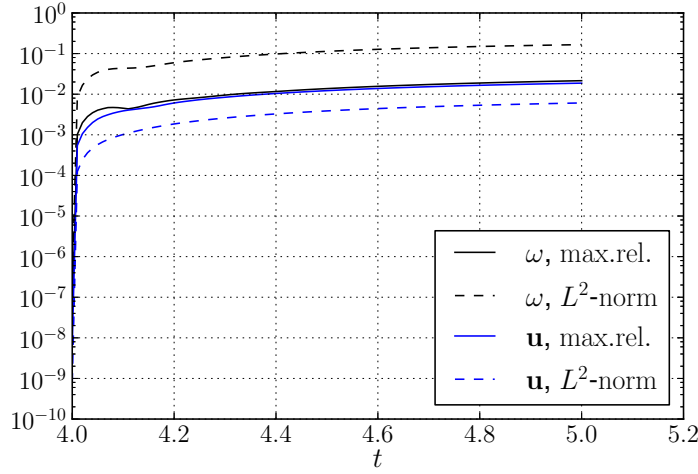


Figure 2.20: Relative error growth of Lamb-Oseen vortex during the evolution from $t_0 = 4$ to $t_f = 5$. Figure depicts the error in vorticity: maximum relative error [—, solid black], error in L^2 – norm [- -, dashed black]; and error in velocity: maximum relative error [—, solid blue], error in L^2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

we have a maximum relative error around 10^{-8} and the center of the Lamb-Oseen core. After 100 steps from $t_0 = 4$ to $t_f = 5$, we see that the maximum relative error increased 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 and due to the nature of the relationship with vorticity and velocity, the error in vorticity is an order higher than the error in velocity. The figure shows the both the maximum relative error, and the error in L^2 – norm. The maximum relative error (for example for vorticity), is determined as

$$\left\| \omega^{\text{exact}} - \omega^{\text{discrete}} \right\|_{\infty} = \frac{\max\{|\omega^{\text{exact}} - \omega^{\text{discrete}}|\}}{\max\{|\omega^{\text{exact}}|\}}. \quad (2.68)$$

The error in L^2 – norm of the vorticity is calculated as

$$\left\| \omega^{\text{exact}} - \omega^{\text{discrete}} \right\|_2 = \left(\sum_i^N |\omega^{\text{exact}} - \omega^{\text{discrete}}|^2 \cdot h^2 \right)^{\frac{1}{2}}. \quad (2.69)$$

The error in velocity is calculated in similar fashion. Investigating the plot, we see that after the first iteration, there is sudden increase in the error, however as time progresses, the error does not growth unbounded. From literature, we see that this trend has also been observed by Barba [3] and Speck [45]. For comparison, we used similar parameters and observe that the sudden jump in error has been similar to the literature.

Comparison of diffusion schemes: Tutty vs. Wee

To validate that the Tutty diffusion scheme indeed performs better than the Wee's diffusion scheme, we compared the error growth of the problem using both diffusion schemes. Figure 2.21 shows the error growth of maximum relative error in vorticity of both diffusion schemes. The solid lines represent the solution of the Wee's diffusion scheme and the dashed lines show the Tutty's diffusion scheme. The convection and the diffusion time steps were 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 Wee's diffusion scheme performs slightly better than Tutty's diffusion scheme. This is because, the diffusion of the vortex blobs is directly incorporated into the remeshing scheme, whereas the Tutty's simple redistribution scheme segregates the diffusion and the remeshing. At $h = 5 \times 10^{-3}$, the diffusion constraint of the Wee's diffusion scheme changes the allowable diffusion time step to $\Delta t_d = 0.02$, meaning that diffusion has to be performed at every second step $k_d = 2$. When investigating the growth in error, we see that the diffusion scheme has a large increase in error, whereas the simple redistribution scheme does not have this limitation and is still able to diffuse at every step and has only a slight increase in error.

We see that the simple redistribution scheme is not only more stable at performing diffusion, but it is also more versatile as it only has an upper limit for Δt_d . Therefore, if we were to use the Wee's modified interpolation kernel for diffusion, we must ensure that $k_d = 1$, by ensure $1/6 \leq c^2 \leq 1/2$. If this constraint is not possible we have to use Tutty's simple redistribution scheme.

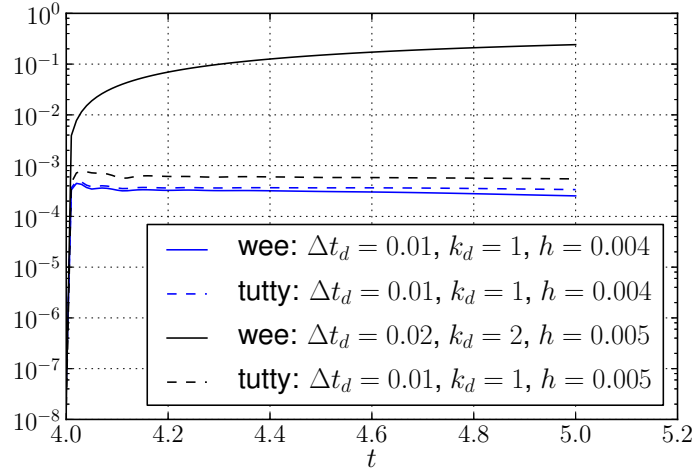


Figure 2.21: Comparison of Tutty's, simple redistribution scheme and Wee's modified interpolation method for treating diffusion. Figure depicts the growth in maximum relative error in vorticity from $t_0 = 4$ to $t_f = 5$ at $\Delta t_c = 0.01$. The Wee diffusion scheme with $\Delta t_d = \Delta t_c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

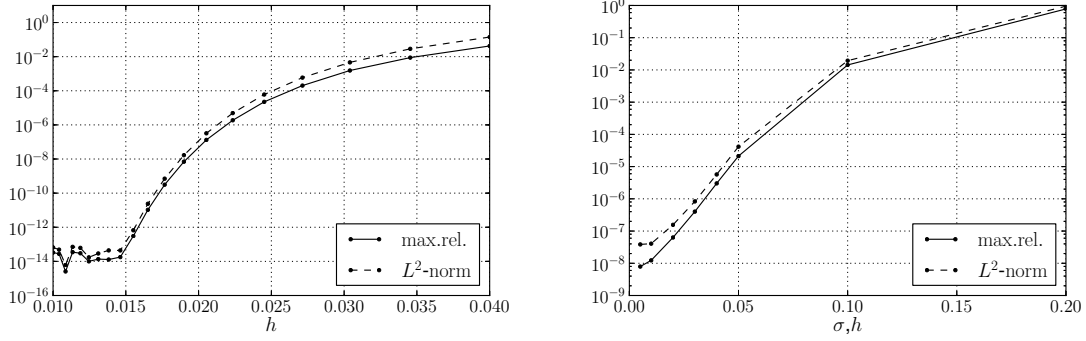
(a) Error in vorticity vs. h with $\sigma = 0.02$ (b) Error in vorticity vs. h, σ with overlap = 1.

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.

2.6.2 Convergence study of the viscous vortex method

The final validation we can perform to ensure the scheme performs as according to theory is to perform a convergence study. For a numerical scheme that is stable, the error in discretization must converge as the resolution is increased.

In the case of spatial discretization, if we were to increase the spatial resolution the error must converge. 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 according to equation 2.17. At lower blob spacing h , the error in vorticity quickly drops to near machine precision. This performs as expected and shows us that error converges as spatial resolution increased. Figure 2.22b, shows the convergence of error when overlap = 1, is maintained. This means that the blob spacing is equal to core size $h = \sigma$. So, when reducing the blobs spacing, this also reduces the core size. At this

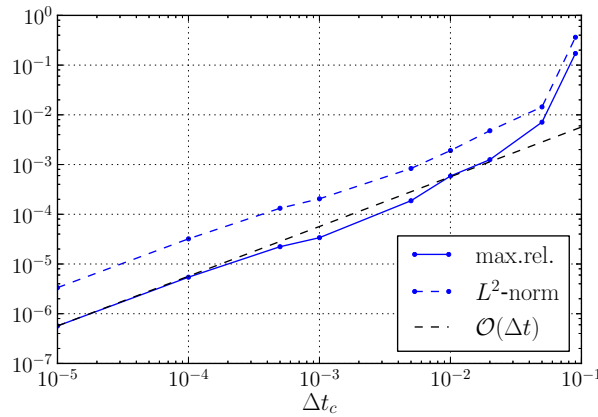


Figure 2.23: Error growth of Lamb-Oseen vorticity field after one-step.

scenario, we see that there is also a convergence is spatial discretization.

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 we are using the Tutty's diffusion scheme, so $\Delta t_d = \Delta t_c$. We see that error converges at $\mathcal{O}(\Delta t)$, meaning that the aggregate convection and diffusion scheme is of first order.

2.7 Summary of the Lagrangian method

In summary, the Lagrangian domain of the hybrid method has been described. The Lagrangian method is used to described the evolution of the wake past the geometry. As a wake predominantly involves in the convection and the diffusion of the wake, Vortex Particle Method is the ideal choice for this domain. Unlike, typical Eulerian method, VPM only required fluid elements where there is vorticity and this means the VPM is inherently auto-adaptive. Furthermore, the computation of the these elements have been accelerated using FMM and parallelized for GPU hardware. This enables the method to be high efficient and allows for future scalability.

For the VPM, the choice of fluid elements was vortex blobs. This has non-zero core size, removing the singularity when performing Biot-Savart calculations. The strengths of these particles is initialized by assigning the local circulation strength to the particle, given by equation 2.18. We see that when dealing with only a Lagrangian method this does not pose a problem. However, when we perform coupling, we see that the gaussian blurring of the original vorticity field during the initialization is a fundamental source of error during the coupling. Strategies such as Beale's iterative method, can help us recover the original vorticity field, however, it is an ill-conditioned problem and can introduce errors. Moreover, we will see that this strategy is not viable when coupling as the correction can only be performed on a non-finite domain. The only approach to minimize this error is to reduce the gaussian spreading by increasing 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 when solved can significantly improve the accuracy and the efficiency of the hybrid coupling. More on this will be discussed in section ??.

The convection of the vortex blobs is done using a 4th-order Runge-Kutta Method. However, even with this higher order convection methods, we will have to deal with lagrangian grid distortion. Lagrangian distortion occurs when particles fail to maintain the overlap ratio as particles will tend to clump together and create regions of no fluid elements due to high strains in the fluid. To deal with this, we use a M'_4 interpolation kernel that remeshing the particles that satisfies the conservation of circulation, linear and angular impulse.

Diffusion of the vortex blobs is also an import part of the VPM. Initially, we employed the modified interpolation kernel by Wee [], which integrated the diffusion process into the standard interpolation kernel. However, the method poses two problems: the diffusion time step Δt_d has a lower limit; the diffusion process introduces error when diffusion does not occur at every step, $k_d \neq 1$. To overcome this problem, we used the simple redistribution scheme by Tutty [], which enables us to perform diffusion at arbitrary

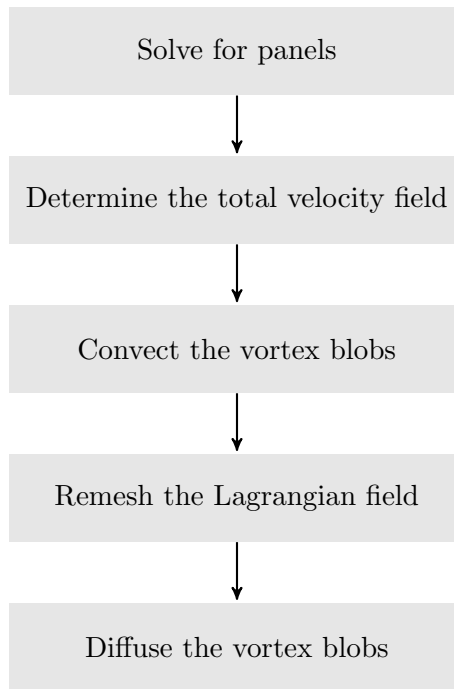


Figure 2.24: Flowchart of the Lagrangian method. The flowchart shows coupling between vortex panels and vortex blobs to evolve from t_n to t_{n+1} .

time-steps, specifically at the same frequency as convection time step. So now, we can diffuse and convect at every iteration, which make the scheme more continuous and more ideal for coupling in the hybrid method.

To deal a body in the flow (for the Lagrangian method), we used vortex panels to enforce the no-slip boundary condition. Koumoutsakos, Leonard, Pepin [], have shown that enforcing no-slip boundary condition is equivalent to no-through boundary condition and they are linked to each. Constant-Strenth Vortex panels, based of Katz [], where used to treat the body and the problem was verified and validated with the analytical solution of a potential flow around a cylinder. There is one difference between our vortex panel method and the standard boundary element method used in VPM. The panels does not solve the problem of vorticity production from the body, as this is dealt with in the Eulerian domain and transfered to the Lagrangian domain. So when validation the Lagrangian method, we cannot validate a problem containing a body.

To validate the vortex blob only Lagrangian method, we used the analytical solution of the Lamb-Oseen vortex. We verified that the Wee's diffusion method indeed miss-performed when the diffusion was not done at every step. The validation was done by investigating the error in the vorticity and the velocity field and was able to conclude that the vortex blobs performed according to literature.

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 on the collocation points of the vortex panels is enforced. We will have ensure that the total circulation of the vortex panels satisfies the conservation of circulation.
2. **Determine the total velocity field:** Determine the total velocity field \mathbf{u} , which is sum of the velocity field induced by the vortex blobs \mathbf{u}_w , the newly calculated vortex panels \mathbf{u}_γ and free-stream flow \mathbf{u}_∞ .
3. **Convect the vortex blobs:** Use the velocity field to time march the vortex blobs to the new position, from t_n to t_{n+1} , by the time step size 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 simple redistribution method for treating diffusion.

The difference here is that the vorticity is not generated from the body in this algorithm. We introduce the vorticity generation from the Eulerian method by interpolating the new vorticity field from the Eulerian domain onto the Lagrangian domain using the Hybrid coupling strategys.

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 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 typical 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 from a body. In the hybrid coupling strategy, we will interpolate this newly resolved near-wall solution on to the Lagrangian domain, where the vortex blobs will efficiently evolve the particles from the Lagrangian frame point.

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.

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

3.1 Introduction to Finite Element Method

Finite Element Method (FEM) is numerical method to solve for the solution of a given differential equation. It is solved by describing it as a variation problem, and by minimizing the error we can reach a approximate solution for the boundary value problem [5]. So, the FEM approximates the unknown variables and converts the partial differential equations to algebraic equations. It was traditionally used for solid mechanics, for the analysis of aircraft structures [40], but have since then used to solve the Navier-Stokes fluid dynamics problems [22] [25] [23].

Finite element discretization

Finite Element solves by dividing the domain of interest into small, simple regions known as “finite elements”. These “elements” are connected at the joints and are called nodes or nodal points. We use these sets of node and elements to represent the actual variation in the field (such as displacement, velocity, pressure or temperature) using simple functions, known as basis functions. Therefore, we have transformed a domain of infinite Degrees of Freedom (DOF) to a finite number of DOFs. We combine this set of equations of element equations into a global system of equations to solve for the final problem.

The discretization of a 2D domain can be observed in figure 3.1. The figures shows two connect finite elements. The cells represent the area of the element, and vertices of the cell are the node of the finite element. Therefore, we have divide the domain Ω into finite sets of cells $\mathcal{T}_h = \{T\}$ and together these cells make the mesh of the Eulerian domain. In 2D, the cells are made of simple geometrical shapes such as triangles, quadrilaterals, and tetrahedrals. There are two approaches to discretize the domain: structured and unstructured mesh. The structured mesh have cells that are of similar shape, such as rectangles and is the simplest approach in discretizing the mesh. The advantage of such discretization is that it has simple data structure and and can perform efficient computation. The downside is that the mesh quality deteriorates ws the complexity of the domain increases. The common strategy of discretizing the domain for finite element method is to use an unstructured mesh, figure 3.2. Even though this mesh data structure is complex, the advantage is that the mesh quality does not deteriorate with the domain complexity.

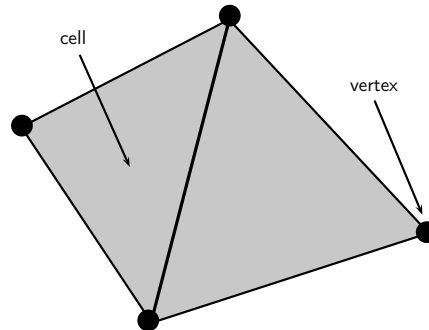


Figure 3.1: A two-dimensional Finite Element geometry. The cell represents the area of the element, and vertices are the edge of the cell.

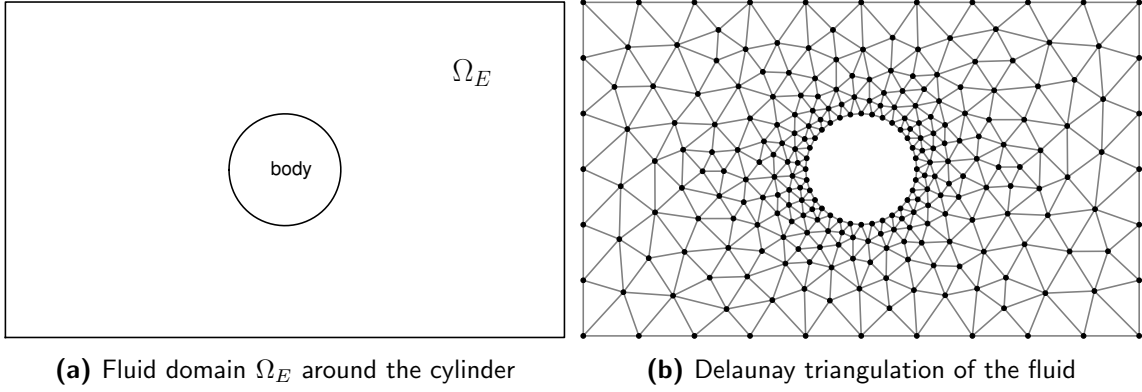


Figure 3.2: Delaunay triangulation of the fluid around a cylinder resulting in unstructured mesh with controllable cell sizes.

There are several algorithms for mesh generation. The standard approach is to employ the Delaunay triangulation method derived from the Voronoi diagram concept [6]. This divides the domain into arbitrary number of triangles, figure 3.2. This type of mesh generation allows us to connect shapes of different types in a simple manner. Furthermore, the triangulation method be controlled by predefined the boundary edges.

Finite element function and function space

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 expressed in a linear combination of basis functions $\{\phi_1, \phi_2, \dots, \phi_N\}$ of the function space V :

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

There are several numbers of Finite Element families, such as Brezzi-Douglas-Marini, Crouzeix-Raviart, Discontinuous Lagrange, Hermite, and Lagrange elements [35]. Each has its own advantage such as the Discontinuous Lagrange, also referred to as Discontinuous Galerkin (DG) element consists of functions that are totally discontinuous. Originally introduced for hyperbolic problem by Reed and Hill [41], the method ables to conserve mass at each element, and are high-order accuracy and robust in solving advection problem. However, for the current problem, we have relied on the standard Lagrange elements, also known as Continuous Galerkin (CG), which are based on the Lagrange polynomials [9].

Variational problem

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 [32]. A 1D Poisson problem is given as,

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

We can transform equation 3.2 into the variational form by multiplying with a test function v , and integrating it over the domain Ω ,

$$-\int_{\Omega} (\nabla^2 u) v \, dx = \int_{\Omega} f v \, dx. \quad (3.3)$$

The function u in variational form, equation 3.3, is known as the trial function and it is the solution that we are trying to approximate. The test function v lies in the test function space \hat{V} and our trial function u lies on the function space 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,

$$-\int_{\Omega} \nabla u \nabla v \, dx = \int_{\Omega} f v \, dx \quad \forall v \in \hat{V} \quad (3.4)$$

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} . In order to solve this continuous problem numerically, we must discretize it first into discrete variational problem,

$$-\int_{\Omega} \nabla u_h \nabla v \, dx = \int_{\Omega} f v \, dx \quad \forall v \in \hat{V}_h \subset \hat{V}, \quad (3.5)$$

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, figure 3.1, where \hat{V}_h and V_h are described by the piecewise linear functions of the triangles and the functions of this function of the test space is zero at the boundary and the function of the trial space is equal to the boundary condition u_0 .

The equation 3.5 can be summarized as follows,

$$a(u, v) = L(v), \quad (3.6)$$

where $a(u, v) = -\int_{\Omega} \nabla u \nabla v \, dx$, and $L(v) = \int_{\Omega} f v \, dx$ and is referred to as bilinear and linear form, respectively. To solve for the discrete solution we substitute,

$$u = \sum_{j=1}^N U_j \phi_j, \quad (3.7)$$

a 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} , is defined as

$$v = \sum_{i=1}^N \hat{\phi}_i. \quad (3.8)$$

The test function v is taken to be zero at the boundary and one everywhere else. Substituting equation 3.7 and 3.8 into equation 3.6 gives,

$$\sum_{j=1}^N a(\phi, \hat{\phi}_i) U_j = L(\hat{\phi}_i). \quad (3.9)$$

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

$$\mathbf{A}U = b, \quad (3.10)$$

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 3.10, we use the FEniCS Project that has implemented a comprehensive library of finite elements, and high performance linear algebra. The DOLFIN library from the FEniCS Project was used to define the Eulerian domain of the hybrid coupling scheme.

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 automated Finite Element Methods for solving the solutions of differential equation [1]. 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 [35]. 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 [36]. 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.

```

1 from dolfin import *
2
3 # Generate unit square mesh:  $24 \times 24$ 
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 #  $u_0 = \sin x \cdot \cos y$ 
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  $\partial\Omega$ 
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:  $a = -\int \nabla u \nabla v \, dx$ 
25 L = f*v*dx # RHS:  $L = \int f v \, dx$ 
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

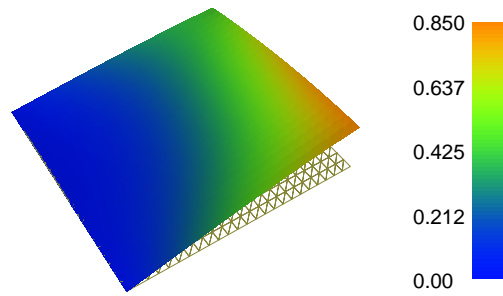


Figure 3.3: DOLFIN VTK plot of the Poisson solution, given by the problem, source code listing 3.1.

The Poisson problem to with the source code was evaluated is given by,

$$-\nabla^2 u = f, \quad (3.11)$$

where $f = 2$ and $u(x) = u_0(x) = \sin x \cdot \cos y$ on boundary $\partial\Omega$ can be automated using the `DOLFIN` library. The solution algorithm follows from section 3.1. The section 3.1 can be seen in listing 3.1.

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 a non-trivial process, as a non-ideal mesh can be computationally expensive, given un-precise data, and even cause convergence issues. There have been literatures dedicated just to improve the mesh generation, that focuses of improving the quality of the mesh thereby increasing the quality of the data and increasing the robustness of the simulation [24].

We used GMSH an open-source finite element mesh generator, which has implemented a user-friendly interface and fast algorithms, developed by Geuzaine and Remacle [19]. Its kernel uses BLAS and LAPACK linear algebra packages, and is written in C++. It allows for scriptability making it ideal to chain it with our current Python code project for 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 formulation of the Navier-Stokes to describe the evolution of the vorticity in the wake. In the Eulerian domain, where have the no-slip boundary, we have decided to use the primitive variables for formulation the fluid dynamics problem.

3.3.1 Velocity-pressure formulation

The velocity-pressure formulation is the standard formulation of the Navier-Stokes equations of the fluid dynamics problem. The 2-D incompressible Navier-Stokes equations of unit fluid density is given as,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \nabla \cdot \sigma = f, \quad (3.12a)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (3.12b)$$

where σ is the Cauchy stress tensor given as,

$$\sigma(\mathbf{u}, p) = 2\nu\epsilon(\mathbf{u}) - p\mathbf{I}. \quad (3.13)$$

The problem contains our two unknowns which are the velocity field \mathbf{u} and our pressure field p . The problem is function of the viscosity ν , external force f and the symmetric gradient,

$$\epsilon(u) = \frac{1}{2} (\nabla \mathbf{u} + \nabla \mathbf{u}^T). \quad (3.14)$$

The velocity field \mathbf{u} is vector function that lies on the vector-valued function space V and scalar pressure field p lies on the function space Q .

3.3.2 Incremental pressure correction scheme

The algorithm to solve the Navier-Stokes problem was first demonstrated by Chorin in 1968 [11], nowadays referred to as Chorin's projection method or 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 3.12. 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 Goda 1979 [20], has introduced a improvement to scheme known as the 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 determining the corrected pressure [35]. The algorithms to the IPCS scheme can be summarized as follows:

1. **Compute the tentative velocity:** The tentative velocity given as \mathbf{u}^* is determined by solving

$$\begin{aligned} \langle D_t^n \mathbf{u}^*, v \rangle + \langle \mathbf{u}^{n-1} \cdot \nabla \mathbf{u}^{n-1}, v \rangle + \langle \sigma(\mathbf{u}^{n-\frac{1}{2}}, p^{n-1}), \epsilon(v) \rangle \\ + \langle p^{n-1} \hat{\mathbf{n}}, v \rangle_{\partial\Omega} - \langle v \hat{\mathbf{n}} \cdot (\nabla \mathbf{u}^{n-\frac{1}{2}})^T, v \rangle_{\partial\Omega} = \langle f^n, v \rangle, \end{aligned} \quad (3.15)$$

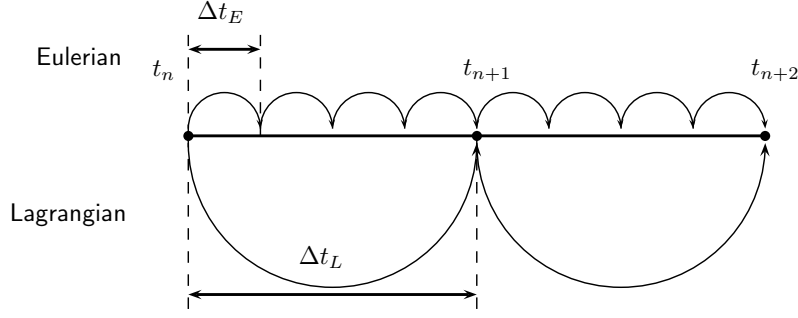


Figure 3.4: Eulerian multi-stepping to match the lagrangian Δt_L . The figures shows $\Delta t_L = 4\Delta t_E$ and required $k_E = 4$ iterations to time march from t_n to t_{n+1} .

where $\langle u, v \rangle = \int_{\Omega} u(x)v(x) \, dx$ is the inner product of the two functions. The term $u^{n-\frac{1}{2}}$ is defined as $u^{n-\frac{1}{2}} = (u^* + u^{n-1})/2$. The equation is given for all test function $v \in V$, with the valid dirichlet velocity boundary conditions at boundary $\partial\Omega$.

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

$$\langle \nabla p^n, \nabla q \rangle = \langle \nabla p^{n-1}, \nabla q \rangle - \langle \nabla \cdot u^*, q \rangle / \Delta t_n \quad (3.16)$$

with the valid pressure boundary condition, and the test function $q \in Q$. Using the corrected pressure, we can determine the final corrected velocity field.

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

$$\langle u^n, v \rangle = \langle u^*, v \rangle - \Delta t_n \langle \nabla(p^n - p^{n-1}), v \rangle, \quad (3.17)$$

for the velocity boundary condition.

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 [34].

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:

$$\text{CFL} = \Delta t_{\max} \frac{\|\mathbf{u}\|_{\max}(\nu + \Delta h_{\min} \|\mathbf{u}\|_{\max})}{\Delta h_{\min}^2} \leq 1. \quad (3.18)$$

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 3.4.

```

1 # Define the trial and test function
2 omega = TrialFunction(W)
3 v = TestFunction(W)
4
5 # Define the variation problem for vorticity
6 a = inner(omega,v)*dx # <math>\langle \omega, v \rangle</math>
7 b = inner(curl(u),v)*dx # <math>\langle \nabla \times u, v \rangle</math>
8
9 # Pre-Assemble the LHS
10 A = assemble(a)
11
12 ...
13
14 # During the time-stepping
15 omega = Function(W) # Define the function
16 B = assemble(b) # Assemble b
17 solve(A, omega.vector(), B) # Solve for vorticity

```

Listing 3.2: The PYTHON implementation of the vorticity calculation

3.3.3 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,

$$\omega = \nabla \times u, \quad (3.19)$$

and is defined as the curl of the velocity field \mathbf{u} . Using the IPCS scheme, the finite element method that we have employed solves for the velocity field and the pressure field. Therefore, to derive the vorticity field, we simply have to take the curl of our solution. However, as this evaluation have to be done at every step (i.e t_n, t_{n+1}, \dots of figure 3.4), we have to solve this problem in a efficient manner. An efficient process, that FENICS has implemented, is by pre-assemble the known of the problem, so that the only terms we have to redefine the unknowns of the problem. To do this, we must first define equation 3.19 in the variational (integral) form,

$$\int_{\Omega} \omega \cdot v \, dx = \int_{\Omega} (\nabla \times u) \cdot v \, dx, \quad (3.20)$$

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

$$a(\omega, v) = L(v) \quad (3.21)$$

where $a(\omega, v)$ contains the knowns of the problem and can be pre-calculated to optimize the problem. $L(v)$ is the unknown of the problem which has to be recalculated every time we need to solve the problem. The PYTHON implementation of the algorithm is show

```

1 ...
2
3 def epsilon(u):
4     "Returns symmetric gradient"
5     return 0.5*(grad(u) + grad(u).T)
6
7 def sigma(u,p,nu):
8     "Returns stress tensor"
9     return 2*nu*epsilon(u) - p*Identity(u.cell().d)
10
11 # Define the normal function
12 n = FacetNormal(mesh)
13
14 # Define the unit vectors
15 eX = Constant((1.0, 0.0))
16 eY = Constant((0.0, 1.0))
17
18 # Define the line integrator
19 ds = Measure("ds")[boundaryDomains]
20 noSlip = 2 # No-slip boundary identification = 2
21
22 # Determine the forces
23 # Integrate the forces over the boundaryDomain == noSlip
24 L = assemble(inner(inner(sigma(u,p,nu), n), eY)*ds[noSlip]) # Lift
25 D = assemble(inner(inner(sigma(u,p,nu), n), eX)*ds[noSlip]) # Drag

```

Listing 3.3: The PYTHON implementation of the force calculation

in listing 3.2 and we see the LHS of the problem is pre-evaluated using the `assemble` function of the DOLFIN library.

3.3.4 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

$$\sigma(\mathbf{u}, p) = 2\nu\epsilon(\mathbf{u}) - p\mathbf{I}, \quad (3.22)$$

where ϵ is the symmetric gradient, equation 3.14, 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,

$$L = \int_{\partial\Omega} [\sigma(\mathbf{u}, p) \cdot \hat{\mathbf{n}}] \cdot \hat{\mathbf{e}}_y \, ds, \quad (3.23a)$$

$$D = \int_{\partial\Omega} [\sigma(\mathbf{u}, p) \cdot \hat{\mathbf{n}}] \cdot \hat{\mathbf{e}}_x \, ds, \quad (3.23b)$$

where $\hat{\mathbf{e}}_x$ and $\hat{\mathbf{e}}_y$ are the 2D unit Cartesian vectors $[1, 0]$ and $[0, 1]$ respectively. 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-stream and the drag is tangential to it,

$$C_l = \frac{L}{\frac{1}{2} \|\mathbf{u}\|_{\infty}^2 c}, \quad (3.24a)$$

$$C_d = \frac{D}{\frac{1}{2} \|\mathbf{u}\|_{\infty}^2 c}. \quad (3.24b)$$

3.4 Validation of eulerian method

3.4.1 Lamb-Oseen Vortex

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

3.4.3 Impulsively started cylinder at $Re = 550$

3.5 Summary

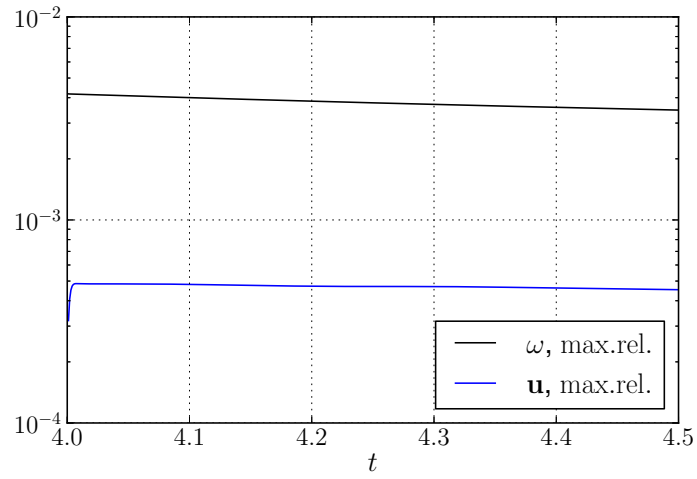
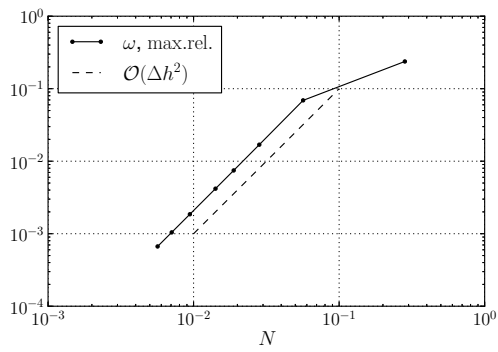
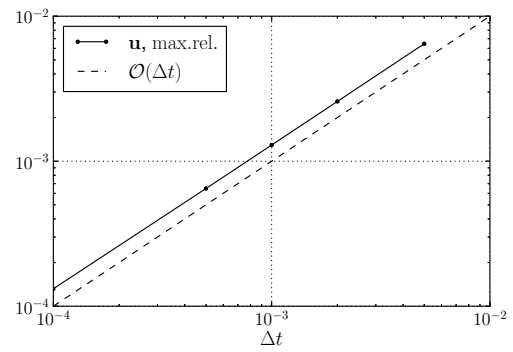


Figure 3.5: Eulerian Lamb-Oseen relative vorticity evolution



(a) Lamb-Oseen dx convergence



(b) Lamb-Oseen dt convergence

Figure 3.6: Lamb-Oseen convergence

Hybrid Eulerian-Lagrangian Vortex Particle Method

4.1 Theory of Domain Decomposition Method

4.1.1 Advantage of domain decomposition

4.1.2 Assumptions and Limitations

4.1.3 Modified coupling strategy

4.2 Eulerian-Lagrangian coupling algorithm

4.2.1 Eulerian dirichlet boundary condition

4.2.2 Vorticity interpolation algorithm

4.3 Introduction to pHyFlow: Hybrid solver

4.3.1 Program structure

4.4 Summary

Chapter 5

Verification and Validation of Hybrid Method

5.1 Error in coupling: Verification with Lamb-Ossen vortex

5.1.1 Generation of artificial vorticity

5.2 Clercx-Bruneau dipole convection at $Re = 625$

5.2.1 Comparison of vorticity contours

5.2.2 Variation in maximum vorticity

5.2.3 Variation in kinetic energy

5.2.4 Variation in enstrophy

5.3 Clercx-Bruneau dipole collision at $Re = 625$

5.3.1 Comparison of vorticity contours

5.3.2 Variation in maximum vorticity

5.3.3 Variation in kinetic energy

5.3.4 Variation in enstrophy

5.3.5 Variation in palinstrophy

5.4 Impulsively started cylinder problem at $Re = 550$

5.4.1 Evolution of the wake

5.4.2 Evolution of pressure and friction drag

5.4.3 Evolution of lift

5.5 Moving body

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] About the FEniCS Project FEniCS Project.
- [2] L.a. Barba and Louis F. Rossi. Global field interpolation for particle methods. *Journal of Computational Physics*, 229(4):1292–1310, February 2010.
- [3] Lorena A LA Barba. Vortex method for computing high-Reynolds number flows: Increased accuracy with a fully mesh-less formulation. 2004, 2004.
- [4] J.Thomas Beale. On the Accuracy of Vortex Methods at Large Times. In Bjorn Engquist, Andrew Majda, and Mitchell Luskin, editors, *Computational Fluid Dynamics and Reacting Gas Flows SE - 2*, volume 12 of *The IMA Volumes in Mathematics and Its Applications*, pages 19–32. Springer New York, 1988.
- [5] Susanne C Brenner and Ridgway Scott. *The mathematical theory of finite element methods*, volume 15. Springer, 2008.
- [6] Graham F Carey. *Computational Grids: Generations, Adaptation & Solution Strategies*. CRC Press, 1997.
- [7] Chien-Cheng Chang and Ruey-Ling Chern. A numerical study of flow around an impulsively started circular cylinder by a deterministic vortex method. *Journal of Fluid Mechanics*, 233(-1):243, April 2006.
- [8] P Chassaing. Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder. 166, 1986.
- [9] Z Chen. *The Finite Element Method: Its Fundamentals and Applications in Engineering*. World Scientific, 2011.
- [10] AJ Chorin. Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, 1973.
- [11] Alexandre Joel Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of computation*, 22(104):745–762, 1968.

- [12] H.J.H. Clercx and C.-H. Bruneau. The normal and oblique collision of a dipole with a no-slip boundary. *Computers & Fluids*, 35(3):245–279, March 2006.
- [13] Christopher D Cooper, Santa Mar, and Lorena Barba. Panel-Free Boundary Conditions for Viscous Vortex Methods. *19th AIAA Computational Fluid Dynamics*, 2009(June):1–12, June 2009.
- [14] G H Cottet and P D Koumoutsakos. *Vortex Methods: Theory and Practice*, volume 12. Cambridge University Press, 2000.
- [15] G. Daeninck. *D EVELOPMENTS IN HYBRID APPROACHES Vortex method with known separation location Vortex method with near-wall Eulerian solver RANS-LES coupling*. PhD thesis, Universit catholique de Louvain, 2006.
- [16] S. Degond, P.; Mas-Gallic, Pierre Degond, and S Mas-Gallic. The weighted particle method for convection-diffusion equations. I. The case of an isotropic viscosity. *Mathematics of Computation*, 53(188):485–507, 1989.
- [17] C J Simão Ferreira, H Bijl, G Van Bussel, and G Van Kuik. Simulating Dynamic Stall in a 2D VAWT: Modeling strategy, verification and validation with Particle Image Velocimetry data. *Journal of Physics: Conference Series*, 75:012023, July 2007.
- [18] CJ Simão Ferreira. The near wake of the VAWT: 2D and 3D views of the VAWT aerodynamics. 2009.
- [19] Christophe Geuzaine. Gmsh : a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. pages 1–24, 2009.
- [20] Katuhiko Goda. A multistep technique with implicit difference schemes for calculating two-or three-dimensional cavity flows. *Journal of Computational Physics*, 30(1):76–95, 1979.
- [21] JL Guermond and HZ Lu. A domain decomposition method for simulating advection dominated , external incompressible viscous flows. *Computers & fluids*, 29:525–546, 2000.
- [22] J.L. Guermond, P. Minev, and Jie Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44-47):6011–6045, September 2006.
- [23] J.L. Guermond and Jie Shen. A new class of truly consistent splitting schemes for incompressible flows. *Journal of Computational Physics*, 192(1):262–276, November 2003.
- [24] G A Hansen, R W Douglass, and A Zardecki. *Mesh Enhancement: Selected Elliptic Methods, Foundations and Applications*. Imperial College Press, 2005.
- [25] Hans Johnston and Jian-Guo Liu. Accurate, stable and efficient NavierStokes solvers based on explicit treatment of the pressure term. *Journal of Computational Physics*, 199(1):221–259, September 2004.

-
- [26] Joseph Katz and A Plotkin. *Low-speed aerodynamics*. Cambridge Aerospace Series. Cambridge University Press, 2001.
 - [27] P Koumoutsakos. Inviscid axisymmetrization of an elliptical vortex. *Journal of Computational Physics*, 138(2):821–857, 1997.
 - [28] P Koumoutsakos and A Leonard. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *Journal of Fluid Mechanics*, pages 1–38, 1995.
 - [29] P Koumoutsakos, A Leonard, and F Pépin. Boundary Conditions for Viscous Vortex Methods. *Journal of Computational Physics*, 113(1):52–61, July 1994.
 - [30] Petros D Koumoutsakos. Direct numerical simulations of unsteady separated flows using vortex methods, 1993.
 - [31] Horace Lamb. *Hydrodynamics*. Cambridge university press, 1993.
 - [32] HP Langtangen. A FEniCS tutorial. *Automated Solution of Differential Equations by the ...*, pages 1–94, 2012.
 - [33] Y Lecointe and J Piquet. On the use of several compact methods for the study of unsteady incompressible viscous flow round a circular cylinder. *Computers & fluids*, 12(4):255–280, 1984.
 - [34] Anders Logg. NSBench in Launchpad. <https://launchpad.net/nsbench>, 2014.
 - [35] Anders Logg, KA Kent-Andre Mardal, and Garth Wells. *Automated solution of differential equations by the finite element method: The fenics book*, volume 84. Springer, 2012.
 - [36] Anders Logg and Garth N Wells. DOLFIN : Automated Finite Element Computing. *CoRR*, abs/1103.6:1–27, 2011.
 - [37] J.J Monaghan. Extrapolating B-splines for interpolation. *Journal of Computational Physics*, 60(2):253–262, September 1985.
 - [38] MT Nair and TK Sengupta. Unsteady flow past elliptic cylinders. *Journal of fluids and structures*, pages 555–595, 1997.
 - [39] ML Ould-Salihi, GH Cottet, and M El Hamraoui. Blending finite-difference and vortex methods for incompressible flow computations. *SIAM Journal on Scientific ...*, 22(5):1655–1674, 2001.
 - [40] Singiresu S Rao. *The finite element method in engineering*. Butterworth-heinemann, 2005.
 - [41] Wm H Reed and T R Hill. TRIANGULARMESH METHODS FOR THE NEUTRON TRANSPORT EQUATION. *Los Alamos Report LA-UR-73-479*, 1973.
 - [42] S. Shankar and L.van Dommelen. A New Diffusion Procedure for Vortex Methods. *Journal of Computational Physics*, 127(1):88–109, August 1996.

- [43] Doug Shiels. *Simulation of controlled bluff body flow with a viscous vortex method*. Dissertation (ph.d.), California Institute of Technology, 1998.
- [44] Carlos Simão Ferreira, Gijs Kuik, Gerard Bussel, and Fulvio Scarano. Visualization by PIV of dynamic stall on a vertical axis wind turbine. *Experiments in Fluids*, 46(1):97–108, August 2008.
- [45] Robert Speck. *Generalized algebraic kernels and multipole expansions for massively parallel vortex particle methods*, volume 7. Forschungszentrum Jülich, 2011.
- [46] MJ Stock, A Gharakhani, and CP Stone. Modeling Rotor Wakes with a Hybrid OVERFLOW-Vortex Method on a GPU Cluster. *AIAA Applied Aerodynamics ...*, pages 1–12, 2010.
- [47] Henrik Tryggeson. *Analytical Vortex Solutions to the Navier-Stokes Equation*. PhD thesis, 2007.
- [48] O.R. Tutty. A Simple Redistribution Vortex Method (with Accurate Body Forces). *ArXiv e-prints*, September 2010.
- [49] L.J. Vermeer, J.N. Sørensen, and a. Crespo. Wind turbine wake aerodynamics. *Progress in Aerospace Sciences*, 39(6-7):467–510, August 2003.
- [50] Daehyun Wee and Ahmed F. Ghoniem. Modified interpolation kernels for treating diffusion and remeshing in vortex methods. *Journal of Computational Physics*, 213(1):239–263, March 2006.
- [51] Wikipedia. Vertical-Axis Wind Turbine, July 2013.
- [52] G S Winckelmans, J K Salmon, Universite Catholique De Louvain, M S Warren, A Leonard, Theoretical Astrophysics, Los Alamos National Laboratories, Los Alamos, B Jodoin, and Universite De Sherbrooke. Application of Fast Parallel and Sequential Tree Codes to Computing Three-Dimensional Flows with the Vortex Element and Boundary Element Methods. 1:225–240, 1996.
- [53] Grégoire Winckelmans, Roger Cocle, Louis Dufresne, and Raphaël Capart. Vortex methods and their application to trailing wake vortex simulations. *Comptes Rendus Physique*, 6(4-5):467–486, May 2005.