POLYTECHNIC OF TURIN

Master's Degree in Aerospace Engineering

Master's Degree Thesis

Development and validation of a Vortex Particle code to evaluate Rotor and Propeller Performance

Supervisors

Prof. Domenic D'AMBROSIO

Ph.D. Manuel CARRENO RUIZ

Candidate

Francesco BELLELLI

2021/2022

Abstract

The efficiency of a lifting surface depends, among various things, on the downstream wake. In addition, the overall efficiency of an aircraft is strongly influenced by the wake of the lifting surfaces itself. This effect is evident if we consider the topic of propeller-based vehicles, which is nowadays widely gaining momentum in the community. Due to this increasing interest in the electric distributed propulsion for urban air mobility and UASs, it is relevant to find a fast yet reliable tool, to evaluate the performance of such propellers. Nevertheless, analyzing of the flow field downstream of the propeller is a complex and time-demanding task, especially in the early stages of the design process, where an incorrect evaluation of these effects may have repercussions on many aspects of the project.

This thesis proposes a relatively fast yet accurate algorithm to evaluate the flow field downstream of the propeller, based on the Vortex Particle Method, which became popular in the early 90s. The calculation performed by this method, the N body problem, involves a computational cost of $\mathcal{O}(N_p^2)$, where N_p is the number of particles. In order to reduce the computational cost to $\mathcal{O}(N_p)$, I implemented a Fast Multipole Method algorithm (FMM). FMM represents a vital tool to achieve acceptable computation times and possibly represents the key feature that makes this code a great compromise between the accuracy and the computational time.

I have studied the integration of the FMM algorithm, written in C++, and the VPM code, written in MATLAB, to ensure that the communication between both codes does not represent a problem. In addition, I performed several simulations to check both the temporal advantage in the usage of the FMM and the method's accuracy.

Moreover, throughout the entire work, I did a thorough validation and verification to ensure the best accuracy of the models used and the correct behavior of the code under all the possible configurations, both for the VPM code itself and the FMM integration.

I am currently working on coding the tool into a stand-alone APP with a simple and understandable graphic user interface (GUI) to easily allow any user to use this code, even without a MATLAB license. This will help the diffusion of the code in the scientific community.

To my family,
To my friends,
To my girlfriend.

Acknowledgements

My journey in Polytechnic of Turin's bachelor's and master's degrees has been surely a formative experience, and I owe a great deal of my success to the family and friends who have supported me along the way. Even people who have spent only a small part of their time with me have contributed to the development of my person, both academically and personally.

I want firstly to thank my family, which has been always comprehensive but also has always motivated me to work hard to the best of my possibilities. I strongly believe that the fragile and often naive soul of a growing student cannot survive easily without a life mentor.

Secondly, I want to thank the entire circle of my friends, which is luckily too wide to be explained in these few rows. In these years I met a huge amount of beautiful people. I want, especially, to thank my Polytechnic friends Jessica, Sibilla, and Giulia, whom I met in the early years of my bachelor's degree and that have soon revealed their nature as extremely beautiful people.

Then, I want to thank my Calabrian friends, who know me much before the starting of my new journey on the extreme opposite side of Italy, and who have been widely supportive, despite the physical distance which has separated us for the majority of time in these five years.

I also want to thank my girlfriend Martina who, even though met me at the end of this journey, has been extremely supportive and fundamental to my personal development.

Later, I want to thank my teammates of ICARUS Polito: an incredibly stimulating and formative, in addition to funny, experience. I met there a lot of fantastic people, which I will never forget.

Moreover, I want to thank my cousin Lorenzo for all the precious advice in the

computer field (and also in the life field), certainly arising from his profound passion for this topic.

A special mention goes to my little, but not so little, Instagram and YouTube community, which has been made a part of my entire Master's Degree journey, and with whom I have often had the opportunity to dialogue and exchange both knowledge and life pieces of advice.

Last, but not least, I want to thank my supervisor, prof. Domenic D'Ambrosio and Ph.D. Manuel Carreno Ruiz for the constant support in the development of this thesis.

Table of Contents

List of Tables

List of Figures

Nomenclature

Miscellaneous

ν	Kinematic viscosity
ρ	Density
$\vec{\omega}$	Vorticity field
$\vec{u}$	Velocity field
p	Pressure

Vortex Particle Method

α	Local incidence
$\Delta\alpha$	Angular step
Γ	Circulation
$\mathcal{H}$	Helicity (singular)
$\mathcal{H}_\sigma$	Helicity (regularized)
Ψ	Streamfunction
Ψ_σ	Regularized streamfunction
ρ	Dimensionless radial distance between particles (with respect σ)
σ	Smoothing radius
ε_σ	Enstrophy (regularized)
ε_σ^f	Enstrophy (regularized and divergence-free)
$\vec{\alpha}$	Vortex particle strength

$\vec{\Omega}$	Total vorticity
$\vec{\omega}^N$	Divergence-free singular vortiicty field
$\vec{\omega}_\sigma^N$	Divergence-free regularized vortiicty field
$\vec{\omega}_\sigma$	Regularized vorticity field
$\vec{A}$	Angular impulse
$\vec{I}$	Linear impulse
$\vec{K}$	Biot-Savar singular kernel
$\vec{u}_\sigma$	Regularized velocity field
$\vec{x}$	Vortex particle position
ζ	Regularization function
c_l	Lift coefficient
D	Drag
E	Kinetic energy (singular)
E_σ	Kinetic energy (regularized)
E_σ^f	Kinetic energy (regularized and divergence-free)
f	Frequency factor in Pedrizzetti's relaxation of the particle divergence
G	Green's function
h	Spacing between the particles
L	Lift
N_p	Number of vortex particles
q	Regularization function
T	Torque
Th	Thrust
U_R	Centroid velocity of a thin vortex ring
vol	Vortex particle volume

Fast Multipole Method

Φ	Scalar potential
I_n^m	Inner translation function
L	BBFMM3D computational domain length
$level$	BBFMM3D number of expansion levels
M_n^m	Multipole Expansion coefficients
n_{crit}	exaFMM maximum number of particles per leaf
O_n^m	Outer translation function
P	exaFMM expansion order
P_n	Legendre Polynomials
Y_n^m	Spherical Harmonics

Acronyms

CFD	Computational Fluid Dynamics
FMM	Fast Multipole Method
HOA	High Order Algebraic
LLT	Lifting Line Theory
VLM	Vortex Lattice Method
VPM	Vortex Particle Method

Chapter 1

Introduction

The purpose of the following dissertation is to explain the work done behind the development of a viscous Vortex Particle code, to evaluate the downstream wake of a lifting surface (propellers, rotors, wings, etc.), and the performances of the latter. The code is a Vortex Particle Method version of a pre-existing one, which has been developed by Alì[1] and which uses the Lifting Line Theory instead of VPM to evaluate the downstream wake.
The main goal of the thesis has therefore been the development of a suitable alternative routine that can be implemented in the code of Alì. The reader is anyway suggested to refer to Alì's thesis to have a complete overview of the LLT and his version of the code.

The main reason for using the Vortex Particle Method instead of others relies upon the following advantages:

- It is a direct and meshless method.
- It is immune to numerical diffusion.
- The number of particles is easily adapted to the flow complexity.
- It is suitable for both internal and external flows.
- It has a considerable calculation speed (if FMM is implemented).
- It is GPU and CPU parallelizable and scalable.

It has a few major drawbacks, such as the divergence-free condition of the flowfield which is not automatically satisfied, but in the following chapters, there will be presented adequate methods to avoid or mitigate those drawbacks.

As previously stated, a good calculation speed is achieved through the usage of a particular acceleration algorithm, named Fast Multiple Method. The FMM is considered to be one of the top 10 algorithms of the 20th century by the Society for Industrial and Applied Mathematics (SIAM). Therefore, an FMM code has been implemented, to reduce the computational complexity to $\mathcal{O}(N_p^2)$ to $\mathcal{O}(N_p)$, where N_p is the number of particles whose interaction due to all of them has to be computed.

The VPM code is written in MATLAB, the FMM code implemented is the exaFMM[2] (although it has been also investigated the possibility to use another one, called "BBFMM3D"[3]), so an ad hoc and suitable interaction between VPM and FMM have been developed, to ensure both the correct behavior of the calculation (with a relative error wrt the exact calculation of a few millesimal point) and that the link does not add so much time to the global computation.

1.1 Theoretical background

The first key topic which needs to be discussed is, to correctly understand the functioning and the potentiality of a VPM code, the theoretical background behind the governing equations that will be derived and discussed in the chapter 3.

1.1.1 Navier Stokes equations for an incompressible flow

The governing equations in a three-dimensional incompressible flow are the Navier - Stokes equations: the continuity scalar equation 1.1 and the momentum vectorial equation 1.2.

$$\nabla \cdot \vec{u} = 0 \tag{1.1}$$

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \vec{u} \tag{1.2}$$

In those equations, the fluid velocity $\vec{u}$ is the unknown quantity, where p is the pressure, ρ the density, and ν the kinematic viscosity.

However, as will be explained in the following chapters, the VPM describes the motion of the fluid through a Lagrangian approach, so it is necessary to write equation 1.2 in a Lagrangian form.
Remarking that the so-called lagrangian derivative is

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \vec{u} \cdot \nabla$$

Where, the eulerian derivative is $\frac{\partial}{\partial t}$, we can write equation 1.2 as

$$\frac{D\vec{u}}{Dt} = -\frac{\nabla p}{\rho} + \nu\nabla^2\vec{u} \tag{1.3}$$

Calling vorticity the curl of velocity vector $\vec{\omega} = \nabla \times \vec{u}$ and using the identity of

$$\vec{u} \times \vec{\omega} = \frac{\nabla\vec{u}^2}{2} - \vec{u} \cdot \nabla\vec{u}$$

It can be written that, from 1.2

$$\frac{\partial\vec{u}}{\partial t} + \frac{\nabla\vec{u}^2}{2} - \vec{u} \times \vec{\omega} = -\frac{\nabla p}{\rho} + \nu\nabla^2\vec{u} \tag{1.4}$$

After some trivial algebraic manipulations, one can obtain the following equation.

$$\frac{\partial\vec{u}}{\partial t} = \vec{u} \times \vec{\omega} - \frac{1}{\rho}\nabla\left(p + \frac{1}{2}\rho\vec{u}^2\right) + \nu\nabla^2\vec{u} \tag{1.5}$$

By taking the curl and using the identities $\nabla \times (\nabla\phi) = 0$ (where ϕ is a scalar) and $\nabla \times (\nabla^2\vec{A}) = \nabla^2(\nabla \times \vec{A})$, we are left with the following equation.

$$\frac{\partial\vec{\omega}}{\partial t} = \nabla \times (\vec{u} \times \vec{\omega}) + \nu\nabla^2\vec{\omega} \tag{1.6}$$

Lastly, applying the identity $\nabla \times (\vec{u} \times \vec{\omega}) = \vec{\omega} \cdot \nabla\vec{u} - \vec{u} \cdot \nabla\vec{\omega} + \vec{u}(\nabla \cdot \vec{\omega}) - \omega(\nabla \cdot \vec{u})$ and recalling that the last two terms goes to zero due to the incompressibility assumption (which implies that velocity and vorticity fields are divergence-free), it can be obtained the vorticity transport equation.

$$\frac{D\vec{\omega}}{Dt} = \vec{\omega} \cdot \nabla\vec{u} + \nu\nabla^2\vec{\omega} \tag{1.7}$$

This is the Navier-Stokes momentum equation in the so-called vorticity-velocity formulation, and it can be noticed that the pressure term does not explicitly appear in that form. It could have been derived also taking the curl of equation 1.3, using the aforementioned identities to get equation 1.7.

Equation 1.7 describes the "lagrangian" evolution of vorticity field, when appropriately coupled with the vorticity-velocity relationship $\vec{\omega} = \nabla \times \vec{u}$ and the flow position-velocity relationship $\vec{u} = \frac{\partial\vec{x}}{\partial t}$.

1.1.2 Brief discussion of the vorticity-velocity equation terms

Let's take a closer look to equation 1.7. It says that the vorticity evolution in time (in a Lagrangian way) is governed by the two terms on RHS:

- The first term, $\vec{\omega} \cdot \nabla \vec{u}$, is the so-called vortex stretching term. The vortex stretching is a phenomenon that is peculiar to three-dimensional flows, as it goes mathematically to zero in the two-dimensional flow representation.

  Stretching is also often seen to be at the basis of the turbulent energy cascade whereby energy is transferred to the smaller scales, and it thus forms an important modeling aspect of the 3D VPM, as stated by Berdowski[4].

  The vortex stretching is unfortunately a source of numerical instabilities for the numerical VPM scheme and therefore its handling has to be widely discussed, during this thesis work.

- The second term, $\nu \nabla^2 \vec{\omega}$, is instead the so-called viscous diffusion term. It becomes important in turbulent flows and, since VPM is considered particularly suited for describing such flows, it appears natural to account for this term in the VPM numerical formulation.

  As will be seen, intense vortex stretching will form prohibitive small scales of complexity within the flow, and viscous diffusion provides the only mechanism for dissipating these small scales energies[5].

1.1.3 Next steps

The Vortex Particle methods for modeling fluid dynamics are based on solving the discretized version of equation 1.7, so it will be shown in chapter 3 the complete derivation of the numerical governing equation of the VPM code, taking into account several issues that can arise from the correct linking of the numerical formulation with the physical correct behavior of the flowfield.

The most important feature that should be noticed right away is that, due to the Lagrangian description of the flowfield, which implies that the motion of a (finite) number of particles is followed through both space and time, the Vortex Particle method does not require a grid generation.

It is indeed one of the major strengths of the VPM since grid generation is certainly one of the most complex and time-demanding tasks of a CFD simulation.

1.2 Document roadmap

In this chapter, a brief overview of the dissertation goals and fluid dynamics background has been presented, to get the reader familiar with the topic and to set the path for this journey along with the study of a propeller wake and its performance.

Chapter 2 deals with a brief literature survey of the Vortex Particle Method and its applications in the past few decades since the VPM has been developed. It also gives an overview on the general topic of other methods (both Vortex Methods and others, such as LLT or BEM), their applications, and the main differences from VPM.

Chapter 3 gives a detailed description of the VPM from a theoretical point of view and also gives a brief explanation of the VPM code that has been developed through this work, with particular attention to how the VPM formulation has been written, to correctly and conveniently match the FMM input requirements.

Chapter 4 gives instead a detailed description of the FMM from a theoretical point of view and through that is briefly explained how the acceleration method has been implemented in the VPM code, as also previously stated in the previous chapter.

Chapter 5 present a strong and detailed validation of the VPM and FMM codes, using the most common and useful tools which have been largely discussed by various authors ([6],[7] et al.).

Chapter 6 deals with the overall results of the code implementation on the rotor, after a short validation on a simple wing geometry.
Conclusions, remarks, and possible future works are finally discussed in the final chapter 7.

Chapter 2

Literature survey of vortex methods and their applications

The following chapter deals with a bibliographical survey on the vortex methods, especially the VPM, and their applications in the flowfield simulations. Firstly, a complete and brief overview of vortex methods is presented.

When treating the problem of completely evaluating the flowfield downstream of the wake, several methods can be used, with several orders of accuracy, depending on their complexity and assumptions.

- Blade Element Momentum (BEM) methods are based on blade discretization and the momentum balance over the flow. However, with BEM theory is assumed a two-dimensional description of the rotor and the wake is therefore not explicitly modeled. To include transient aerodynamics and 3D flow effects, it is necessary to divert to empirical models and corrections.

- Computational Fluid Dynamics (CFD) was born at the same time as the rise of computational power. It is based on the discretization of the flow on a regular grid, in whose cells the Navier-Stokes equations (in an Eulerian form) are solved numerically.

  CFD is suitable in the case of wall-boundary conditions, but the Eulerian description poses problems over the whole flow domain as a large range of flow scales persist over the wake. Therefore a very fine mesh is required at locations of small vortical structures at the blade, and a mesh that is decreasing in coarseness in the wake, where larger scales of eddies dominate.

- The wake can be described more physically by applying a Lagrangian vortex method. The Lagrangian description makes the vortex method auto-adaptive, which means that the range of flow scales is automatically accounted for and that the flow is only described there where information is present. The aforementioned family of methods has, as previously stated, the major of being meshless, not requiring the highly time-demanding and complex task of grid generation.

If on the one hand, the CFD is more suitable for wall-bounded flow, on the other hand, the vortex methods are both more accurate and faster in turbulent wakes. Since the topic of this thesis concerns the performance of a propeller, that is largely influenced by the downstream (turbulent) wake, it appears natural to focus on this last family of methods in the following section.

2.1 Overview of vortex methods

Vortex methods represent the bridge between Blade Element Method (BEM) and Computational Fluid Dynamics (CFD).
They describe the flowfield in terms of vorticity, since the presence of a body in the field induces a vorticity variation, due to the boundary condition; the body is, therefore, a vorticity source within the domain.
Vorticity can be subsequently concentrated in vortex lines and surfaces.

2.1.1 Body modeling

Based on the location of the vortex structure, several vortex methods can be distinguished:

- Panel methods: vortex structures fit the body geometry and they are distributed on its surface. This allows us to take into account also the effects of blade thickness on the flowfield.
- Lifting surface methods: airfoil's camber line is split into several segments each of which corresponds to a vortex element. This does not take into account the thickness effects, as the panel method instead does.
- Lifting line methods: the entire body is replaced by a single vortex line, located at the quarter of the chord.

2.1.2 Wake modeling

It can also be done another differentiation, based on the treatment of the wake. The so-called prescribed vortex wake methods assign an a priori position to the

vortex elements. On the other hand, the so-called free vortex methods allow a free wake evolution.

The last family of the aforementioned vortex methods can be further split into free vortex wake and free vortex particle methods.

- In the free vortex wake methods the wake is modeled through vortex elements, such as filaments or rings. The first step to do so is to compute the vortex elements' strength Γ, which remains constant, as stated by Helmholtz's theorem.

  Then, the wake is identified through several grid nodes (in which local velocity is computed as the sum of free-stream velocity and induced velocity from the other vortex elements) and it is therefore convected, so every temporal step the wake is updated with the new vortex elements from the body.

- In the free vortex particle methods the wake is modeled through lagrangian vortex particles. This particular family of vortex methods has been, as previously stated, implemented in the following dissertation, so the reader is referred to all the following chapters for a detailed description of them.

On one hand, free vortex wake methods implicitly ensure the connection between vortex elements, resulting in fewer terms to be computed through the algorithm. One of the major drawbacks that can be encountered during the modeling of a problem such as the one discussed in this dissertation is that, even though the wake structure can be deformed (due to local velocity variations caused by induced velocity on grid nodes), it cannot mix with other wakes.
Therefore, this family of vortex methods can still be used to simulate the interaction between two wakes, but it is not so suitable to simulate its mixing.

On the other hand, projecting the vorticity field through vortex particles (also called vortons[6]) does not pose any limit on the wake topology, resulting in larger modeling freedom (several phenomena such as turbulence, viscous diffusion, and blade-wake interactions are more naturally incorporated in the model).
However, this lack of connectivity can represent an important obstacle, since that can be developed numerical instabilities, which takes the form of a not divergence-free flowfield, as it instead must be.

2.1.3 Conclusions

Remarking on the potentially turbulent and "mixed-wake" nature of the flowfield that will be computed through this thesis work, it appears natural that the most

suitable vortex methods family to correctly and conveniently calculate the wake downstream of the propeller/propellers is the Vortex Particle Method.
They offer computational and accuracy advantages over both CFD and free vortex wake methods, due to the reasons explained in the previous paragraph.

However, the VPM schemes imply the resolution of a so-called N-body problem, that has a computational cost proportional to the square of the number of particles in the simulation.
Since this number grows every time step (blade rotation of a fixed angle), the computational time of the entire simulation is proportional to the cube of N_p.
To reduce the computational cost of the VPM from $\mathcal{O}(N_p^2)$ to $\mathcal{O}(N_p)$, the application of FMM (Fast Multipole Method) should be foreseen.
As it will be more detailedly explained through chapter 4, the FMM approximately calculates the far-field interactions through clusters of particles.

2.2 Current applications of VPM

The following section provides a brief overview of the current applications of VPM in several fields, such as wind turbines, aircraft wakes, or even structural purposes.

2.2.1 VPM in the wind turbines topic

One of the main fields that have been permeated by the usage of VPM is the wake modeling of wind turbines or wind farms (clusters of wind turbines). In this sense, the works of Berdowski[4] and Seetharaman[8] from TU Delft provide good documentation on the topic.

However, a purely Lagrangian approach for the treatment of three-dimensional flows of wind turbine wakes is relatively not so used yet, since the vast majority of these works have been focused on either CFD or Eulerian-Lagrangian vortex methods. Other possibilities are represented by the application of BEM or LLT, but all of these methods result in several inaccuracies, for example in the kinetic energy recovery (BEM) or for prediction of the wind farm power, due to the dispersion of different turbulence models (CFD).

As reported by Seetharaman[8], several strategies of simulating the wake of a wind turbine through VPM can be cited.

- Using a GENUVP (GENeralized Unsteady Vortex Particle) code to study the rotor aerodynamics and aeroacoustic. It uses the Helmholtz decomposition

$$\vec{u}(\vec{x},t) = \vec{u}_{free-stream}(\vec{x},t) + \vec{u}_{body}(\vec{x},t) + \vec{u}_{wake}(\vec{x},t)$$

Where the second term is modeled through the panel method, while the third one uses the Biot-Savart law. For further details, the reader is referred to the work of Voutsinas et al[9].

- Discretizing the blade into aerodynamics segments and then calculating the airloads with the help of the angle of attack at that segment, the Mach number, and dynamic pressure. Using the Kutta-Jukowski theorem and the aforementioned loads, the blade-bound circulation is then calculated (and assumed to be constant over each blade segment). The vorticity source is finally shed into the wake after being created at the blade segments.[10]

- Doublets at the center of trailing edge segments. It is assumed to have a constant doublets distribution over the rotor surface, previously discretized into several panels. Then a linear system of N equations (where N is the number of panels) is solved for the doublet strengths μ, which is subsequently converted to its equivalent vorticity at the center of every trailing edge segment that is emitted[11].

- Panel and VPM for modeling stationary propeller wake wash. The wake of a propeller has been modeled using the VPM while the aerodynamics of the blade itself has been modeled using a panel method. As these panels are shed into the wake they are converted into vortex particles[12].

2.2.2 VPM for aeronautical applications

From the aeronautical point of view, a great work has surely been made by Calabretta[7], Alvarez[13][14] and Singh[15], who developed several VPM codes to model and treat the wake of a rotor or a propeller and its interaction with another one or with the airframe.

However, as reported by Calabretta, the Vortex Particle Method has been used for a variety of applications, such as the evaluation of structural aerodynamics of various buildings and structures or the evaluation of aircraft and helicopter rotor wakes.
Also, the work made by Singh, who focuses precisely on coaxial rotor helicopters, provides both massive documentation on the topic and represents evidence itself of a VPM application.
About this, as reported by Singh himself, and as stated before, the usage of VPM to evaluate the wake of one or more rotors of helicopters is starting to catch on, also helped by the growing interest in the topic of electric distributed propulsion.

For what is useful to this thesis work, it appears natural to further concentrate on

the usage of VPM as a complement to a traditional three-dimensional panel code, keeping in mind that the circulation on the blade will be still computed as done by Alì[1], with the Lifting Line theory.
One of the main goals of this work has been therefore the research of a suitable way to link the panel discretization of the rotor geometry and the subsequent aforementioned circulation calculation that have already been done in the work of Alì[1], to the new evaluation of the wake.

In this way, the work of Singh presents a clear procedure to effectively convert the panels into particles, as they are shed into the wake. Also, the work of Calabretta represents a good reference, regarding the complex and delicate topic of correctly evaluating the mutual interaction between panels and particles.

2.2.3 Conclusions

As stated by Calabretta himself, VPM has several advantages over traditional panel wakes, such as the lack of connectivity (which is both a good opportunity and an important drawback of the method), allowing a more physical and accurate description of the flowfield. Additionally, treating the wake with VPM avoids the high user interactivity required in selecting wake position and wake-body intersection issues. An example of the aforementioned idea can be found in the FASTAERO work, done by Willis et al[16], also giving evidence of the fact that is possible for a satisfactory integration of panel method and VPM.

Moreover, it can be seen that one could (pseudo) independently choose how to model the induced velocity component related to the presence of the body in the Helmholtz decomposition, depending on the particular application and its resulting necessities.
What is clear is that VPM could represent a powerful and novel tool to correctly, more accurately, and faster accomplish several steps in the initial phase of the aeronautical project, especially when it involves topics such as wake mixing, turbulent flows, etc.

All the aforementioned works implicitly or explicitly remark the necessity of using an acceleration algorithm such as the FMM, which may represent a further source of complexity, but that is needed to make the overall code fast and suitable, while not losing accuracy.

2.3 Literature review

The following section provides a detailed list of the works that have been consulted for the development of this dissertation.
The correct development of both theoretical basis and code implementation skills comes from massive documentation through master thesis and scientific papers.

A detailed mathematical description of the VPM can be found in the works of Wincklemans[6] and Cottet[5]. Also, the work of Calabretta[7] represents a strong linchpin, especially in the code validation and the mutual interaction between panels and particles topics.

For the development of the VPM code, instead, the works of Seetharaman[8], Berdowski[4] and Alvarez[17], in addition to the aforementioned others, have been consulted. More specifically, to correctly ensure the link between the panel discretization from Alì's code[1] and the VPM, the works of Singh[15] and Martin[12] have been consulted too.

During the development of the VPM code, to correctly implement the low-storage third-order Runge-Kutta scheme, the works of Niegemann[18], the NASA Technical Memorandum 109111[19] and the NASA Report 99-22[20] have been consulted.

Meanwhile, during the strong and rigorous validation phase of the code and in addition to the aforementioned works, scientific papers from Alvarez[13][14], Knio & Ghoniem[21], Sullivan[22] and Konstantinov[23] have been reviewed. Regarding the specific topic of particle relaxation, the works of Beale[24] and Pedrizzetti[25] have been consulted.

The development of the FMM algorithm, both through theoretical basis and actual application has been possible thanks to the works of Cheng[26], Greengard[27], Sheel[28], Fong[3], Chen[29], and Wang et al.[2].

Chapter 3

The Vortex Particle Method

In the following chapter, it will be explained the theory above the implementation of a VPM code, involving both fundamentals and other topics that are more numerically related to the latter.

3.1 Fundamental theory

To begin with, one can see a vortex particle p (also called vorton or vortex stick) as an element which has a position, denoted with $\vec{x}^p(t)$, a strength, denoted with $\vec{\alpha}^p(t)$ and a volume, denoted with vol^p. Strength is defined as the product between the vorticity and the volume of the vorton.

$$\vec{\alpha}^p(t) = \vec{\omega}^p(t) vol^p \tag{3.1}$$

The vorticity field can be further seen as the resulting one due to the influence of N_p vortex particles.
The vorticity field in a point $\vec{x}$ due to the presence of a vortex particle located at $\vec{x}^p(t)$ is defined as the product between the vorton strength and the Dirac distribution at $\vec{x} - \vec{x}^p(t)$.
Therefore, the vorticity field in $\vec{x}$ due to the presence of all the N_p vortons with which it has been discretized itself is a linear combination of all the fields generated by each vorton.

$$\vec{\omega}(\vec{x}, t) = \sum_{p=1}^{N_p} \vec{\alpha}^p(t) \delta(\vec{x} - \vec{x}^p(t)) \tag{3.2}$$

Recalling that the vorticity field in an incompressible three-dimensional flow satisfies the equation below

$$\vec{\omega}(\vec{x}, t) = -\nabla^2 \Psi(\vec{x}, t) \tag{3.3}$$

Where $\Psi(\vec{x},t)$ is the so-called streamfunction and recalling that the velocity field can be therefore obtained as the curl of the aforementioned streamfunction, then the problem can be traced back to finding $\Psi(\vec{x},t)$.

According to Wincklemans[6], the Green's function $G(\vec{x})$ for $-\nabla^2$ in an unbounded three-dimensional domain is

$$G(\vec{x}) = \frac{1}{4\pi|\vec{x}|} \tag{3.4}$$

Therefore, the streamfunction solution of equation 3.3 can be evaluated as the convolution product between the Green's function and the vorticity field.

$$\Psi(\vec{x},t) = G(\vec{x}) * \vec{\omega}(\vec{x},t) \tag{3.5}$$

Substituting both Green's function and vorticity field as previously explained, it can be obtained that

$$\Psi(\vec{x},t) = \sum_{p=1}^{N_p} G(\vec{x}-\vec{x}^p(t))\vec{\alpha}^p(t) = \frac{1}{4\pi}\sum_{p=1}^{N_p}\frac{\vec{\alpha}^p(t)}{|\vec{x}-\vec{x}^p(t)|} \tag{3.6}$$

Now we have obtained the streamfunction Ψ of the flowfield due to the influence of all the vortons with which the latter itself has been discretized.

To obtain the velocity field it is sufficient to compute the curl of the stremfunction which has been previously evaluated.

$$\vec{u}(\vec{x},t) = \nabla \times \Psi(\vec{x},t) \tag{3.7}$$

Replacing Ψ with the one obtained in equation 3.6 it can be found that

$$\vec{u}(\vec{x},t) = \sum_{p=1}^{N_p} \nabla[G(\vec{x}-\vec{x}^p(t))] \times \vec{\alpha}^p(t) \tag{3.8}$$

Finally replacing the Green's function with the one prescribed by Wincklemans, one can obtain the velocity field due to the influence of all the vortons.

$$\vec{u}(\vec{x},t) = -\frac{1}{4\pi}\sum_{p=1}^{N_p}\frac{1}{|\vec{x}-\vec{x}^p(t)|^3}(\vec{x}-\vec{x}^p(t)) \times \vec{\alpha}^p(t) \tag{3.9}$$

It can be written more simply defining the Biot-Savart kernel $\vec{K}$ as

$$\vec{K} = \frac{1}{|\vec{x}-\vec{x}^p(t)|^3}(\vec{x}-\vec{x}^p(t)) \tag{3.10}$$

So equation 3.9 becomes

$$\vec{u}(\vec{x},t) = \sum_{p=1}^{N_p} \vec{K} \times \vec{\alpha}^p(t) \tag{3.11}$$

The previous equation is one of the two evolution equations of the VPM, since $\vec{u}(\vec{x},t)$ can be seen as the velocity in a point $\vec{x}$ at time t due to the influence of all the N_p vortons, so the velocity of each vorton q can be written as

$$\vec{u}^p(\vec{x}^p(t),t) = -\frac{1}{4\pi}\sum_{q=1}^{N_p} \frac{1}{|\vec{x}^p(t)-\vec{x}^q(t)|^3}(\vec{x}^p(t)-\vec{x}^q(t)) \times \vec{\alpha}^q(t) \tag{3.12}$$

Thus the position of each vorton p can be updated solving

$$\frac{d\vec{x}^p(t)}{dt} = \vec{u}^p(\vec{x}^p(t),t) \tag{3.13}$$

The last evolution equation involves the strength update and it can be derived by recalling the vorticity-velocity form of the governing equation, the discretization of vorticity and velocity fields, and multiplying all the terms by vol^p.

$$\frac{D\vec{\alpha}^p}{Dt} = \vec{\alpha}^p \cdot \nabla \vec{u}^p + \nu \nabla^2 \vec{\alpha}^p \tag{3.14}$$

However, one of the major drawbacks of this method is that the flowfield isn't guaranteed to be divergence-free, since

$$\nabla \cdot \Psi(\vec{x},t) = -\frac{1}{4\pi}\sum_{p=1}^{N_p} \frac{\vec{x}-\vec{x}^p(t)}{|\vec{x}-\vec{x}^p(t)|^3} \cdot \vec{\alpha}^p(t) \tag{3.15}$$

Therefore, the vorticity field isn't guaranteed to be divergence-free, while the velocity field is, because it's the curl of a streamfunction.
However, as stated by Wincklemans, it can be written a divergence-free vorticity field adding to the one in equation 3.2 a term which is needed to close the vortex lines.

$$\vec{\omega}^N(\vec{x},t) = \sum_{p=1}^{N_p}\left[\vec{\alpha}^p(t)\delta(\vec{x}-\vec{x}^p(t)) + \nabla\left(\vec{\alpha}^p(t)\cdot\nabla\left(\frac{1}{4\pi|\vec{x}-\vec{x}^p(t)|}\right)\right)\right] \tag{3.16}$$

After calculating the gradient terms, Wincklemans rewrites this as

$$\vec{\omega}^N(\vec{x},t) = \sum_{p=1}^{N_p}\left[\left(\delta(\vec{x}-\vec{x}^p(t)) - \frac{1}{4\pi|\vec{x}-\vec{x}^p(t)|^3}\right)\vec{\alpha}^p(t) + 3\frac{(\vec{x}-\vec{x}^p(t))\cdot\vec{\alpha}^p(t)}{4\pi|\vec{x}-\vec{x}^p(t)|^5}(\vec{x}-\vec{x}^p(t))\right] \tag{3.17}$$

3.2 Particle regularization

Another issue can be encountered considering that the Biot-Savart kernel 3.10 is singular when $\vec{x} \to \vec{x}^p(t)$. Such a singularity is not physical and can lead to numerical instabilities, so it is necessary to regularize the Biot-Savart kernel introducing the regularization function

$$\zeta_\sigma(\vec{x}) = \frac{1}{\sigma^3}\zeta\left(\frac{|\vec{x}|}{\sigma}\right) \tag{3.18}$$

Where σ is the smoothing radius and ζ is a smoothing function.

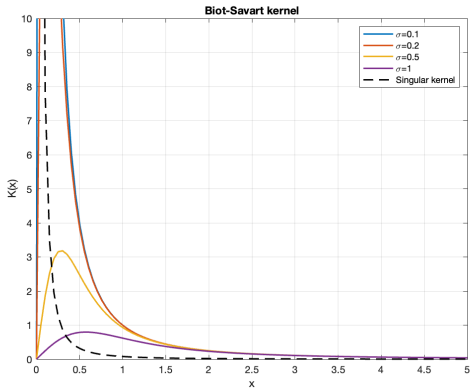


Figure 3.1: Biot-Savart kernel (singular and regularized)

Therefore, the regularized vorticity field becomes

$$\vec{\omega}_\sigma(\vec{x},t) = \zeta_\sigma(\vec{x}) * \vec{\omega}(\vec{x} - \vec{x}^p(t)) = \sum_{p=1}^{N_p} \zeta_\sigma(\vec{x} - \vec{x}^p(t))\vec{\alpha}^p(t) \tag{3.19}$$

It can be found the regularized streamfuncion $\Psi_\sigma(\vec{x},t)$ as the one which satisfies the equation below.

$$\nabla^2\Psi_\sigma(\vec{x},t) = -\vec{\omega}_\sigma(\vec{x},t) \tag{3.20}$$

Therefore

$$\Psi_\sigma(\vec{x},t) = G_\sigma(\vec{x}) * \vec{\omega}_\sigma(\vec{x},t) = \sum_{p=1}^{N_p} G_\sigma(\vec{x} - \vec{x}^p(t))\vec{\alpha}^p(t) \tag{3.21}$$

Where, according to Wincklemans[6]

$$G_\sigma(\vec{x}) = \frac{1}{\sigma}G\left(\frac{|\vec{x}|}{\sigma}\right) \tag{3.22}$$

The regularized velocity field is then obtained the same way as the singular velocity field was.

$$\vec{u}_\sigma(\vec{x},t) = \nabla \times \Psi_\sigma(\vec{x},t) = \sum_{p=1}^{N_p} \nabla[G_\sigma(\vec{x}-\vec{x}^p(t))] \times \vec{\alpha}^p(t) \tag{3.23}$$

Evaluating the gradient term, it can be written that

$$\vec{u}_\sigma(\vec{x},t) = -\sum_{p=1}^{N_p} \frac{q_\sigma(\vec{x}-\vec{x}^p(t))}{|\vec{x}-\vec{x}^p(t)|^3}(\vec{x}-\vec{x}^p(t)) \times \vec{\alpha}^p(t) \tag{3.24}$$

Where

$$q_\sigma(\vec{x}) = \int_0^{\vec{x}} \zeta(t)t^2 dt \tag{3.25}$$

Therefore, it can be seen that the new regularized Biot-Savart kernel is

$$\vec{K}_\sigma = -\frac{q_\sigma(\vec{x}-\vec{x}^p(t))}{|\vec{x}-\vec{x}^p(t)|^3}(\vec{x}-\vec{x}^p(t)) \tag{3.26}$$

So that

$$\vec{u}_\sigma(\vec{x},t) = \sum_{p=1}^{N_p} \vec{K}_\sigma \times \vec{\alpha}^p(t) \tag{3.27}$$

The regularized flowfield suffer from the same lack of divergence-free as the singular one, so the regularized and divergence-free vorticity field can thus be written as

$$\vec{\omega}_\sigma^N(\vec{x},t) = \sum_{p=1}^{N_p} [\vec{\alpha}^p(t)\zeta_\sigma(\vec{x}-\vec{x}^p(t)) + \nabla[\vec{\alpha}^p(t) \cdot \nabla(G_\sigma(\vec{x}-\vec{x}^p(t)))]] \tag{3.28}$$

Wincklemans rewrites this as

$$\begin{aligned} \vec{\omega}_\sigma^N(\vec{x},t) = \sum_{p=1}^{N_p} \Bigg[\left(\zeta_\sigma(\vec{x}-\vec{x}^p(t)) - \frac{q_\sigma(\vec{x}-\vec{x}^p(t))}{|\vec{x}-\vec{x}^p(t)|^3}\right)\vec{\alpha}^p(t) + \\ + \left(3\frac{q_\sigma(\vec{x}-\vec{x}^p(t))}{|\vec{x}-\vec{x}^p(t)|^3} - \zeta_\sigma(\vec{x}-\vec{x}^p(t))\right)\frac{(\vec{x}-\vec{x}^p(t)) \cdot \vec{\alpha}^p(t)}{|\vec{x}-\vec{x}^p(t)|^2}(\vec{x}-\vec{x}^p(t))\Bigg] \end{aligned} \tag{3.29}$$

3.3 Particle strength modeling

The particle strength evolution equation, as previously stated, can be written as

$$\frac{D\vec{\alpha}^p}{Dt} = \vec{\alpha}^p \cdot \nabla \vec{u}^p + \nu\nabla^2\vec{\alpha}^p$$

The following section will be discussed the methodologies to discretize both vortex stretching and viscous diffusion terms.

3.3.1 Vortex stretching term

As reported by Wincklemans[6], there are three formulations for the vortex stretching term in the above equation.

- The classical formulation, where $\vec{\alpha}^p \cdot \nabla \vec{u}^p = (\vec{\alpha}^p \cdot \nabla)\vec{u}^p_\sigma$.
- The transpose formulation, where $\vec{\alpha}^p \cdot \nabla \vec{u}^p = (\vec{\alpha}^p \cdot \nabla^T)\vec{u}^p_\sigma$.
- The mixed formulation, where $\vec{\alpha}^p \cdot \nabla \vec{u}^p = \frac{1}{2}[\alpha^p \cdot (\nabla + \nabla^T)]\vec{u}^p_\sigma$.

It has been shown that the regularized particle method converges to the solution of the respective formulation of the momentum equation selected[5].
However, as stated by Wincklemans[6], the transpose scheme is the only one that conserves vorticity, so it's the one implemented in this thesis work.

Substituting into the transpose scheme the regularized velocity from equation 3.24 and computing the gradients it can be found that

$$\begin{aligned}\vec{\alpha}^p \cdot \nabla \vec{u}^p = \sum_{q=1}^{N_p} \Bigg[\frac{1}{\sigma^3}\frac{q(\rho)}{\rho^3}\vec{\alpha}^p(t) \times \vec{\alpha}^q(t) + \\ + \frac{1}{\sigma^5 \rho}\frac{d}{d\rho}\left(\frac{q(\rho)}{\rho^3}\right)(\vec{\alpha}^p(t) \cdot (\vec{x}^p(t) - \vec{x}^q(t)) \times \vec{\alpha}^q(t))(\vec{x}^p(t) - \vec{x}^q(t)) \Bigg]\end{aligned} \tag{3.30}$$

Where $\rho = \frac{|\vec{x}|}{\sigma}$.
Applying the Wincklemans' high order algebraic regularization functions

$$\begin{cases} \zeta(\rho) = \frac{15}{8\pi}\frac{1}{(\rho^2+1)^{\frac{7}{2}}} \\ q(\rho) = \frac{1}{4\pi}\frac{\rho^3\left(\rho^2+\frac{5}{2}\right)}{(\rho^2+1)^{\frac{5}{2}}} \end{cases} \tag{3.31}$$

One can obtain that

$$\begin{aligned}\vec{\alpha}^p \cdot \nabla \vec{u}^p = \frac{1}{4\pi}\sum_{q=1}^{N_p} \Bigg[\frac{|\vec{x}^p(t) - \vec{x}^q(t)|^2 + \frac{5}{2}\sigma^2}{(|\vec{x}^p(t) - \vec{x}^q(t)|^2 + \sigma^2)^{\frac{5}{2}}}\vec{\alpha}^p(t) \times \vec{\alpha}^q(t) + \\ + 3\frac{|\vec{x}^p(t) - \vec{x}^q(t)|^2 + \frac{7}{2}\sigma^2}{(|\vec{x}^p(t) - \vec{x}^q(t)|^2 + \sigma^2)^{\frac{7}{2}}}(\vec{\alpha}^p(t) \cdot (\vec{x}^p(t) - \vec{x}^q(t)) \times \vec{\alpha}^q(t))(\vec{x}^p(t) - \vec{x}^q(t)) \Bigg]\end{aligned} \tag{3.32}$$

It has been chosen as the high order algebraic regularization because, as stated by Calabretta[7], it has similar convergence properties to the more commonly used Gaussian regularization, but with much simpler G and q functions, resulting also in a gradient that is easier to analytically evaluate.
Additionally, it is the only known regularization for which analytical equations for total vorticity, linear impulse, angular impulse, kinetic energy, and enstrophy can be derived, as will be seen in the section 3.7.

3.3.2 Viscous diffusion term

One of the major strengths of VPM is its possibility to easily account for viscous diffusion, which also has been demonstrated to help maintain a nearly divergence-free vorticity field, that is necessary to obtain a valid physical solution.
Thus, it appears peremptory to correctly model the viscous diffusion term in equation 3.14.

There are conventionally three possible schemes to numerically account for viscous diffusion.

- Random walk method. The particles in the domain are made to undergo a Brownian movement to simulate diffusion. To simulate the Brownian movement the particles are made to experience random position updates every time step Δt, adding to position at the previous time step x_p^n a random generated number, which follows a Gaussian distribution.

$$x_p^{n+1} = x_p^n + \xi_p^n \tag{3.33}$$

$$\xi_p^n = \frac{1}{\sqrt{(2\pi\varepsilon)^d}} e^{-\frac{x^2}{2\varepsilon}} \tag{3.34}$$

  Where d is the dimension and $\varepsilon = 2\nu\Delta t$ is the variance.

  This method, however, does not seems to be a valid option to simulate viscous diffusion, since it did not account for any changes in the magnitude of the vorticity due to diffusion.

- Core spreading method. The core radius is changed on account of stretching and diffusion separately and, after, a new blob replaces the vorton.

  Unfortunately, this method does not converge to the Navier-Stokes equations, since the core can grow beyond the characteristic length scales.

  Rossi[30] provided a correction that ensures the convergence of the method, allowing the particles to split and reposition in the cross-sectional plane, as soon as the initial particle grows beyond a specific threshold core size.

- Particle Strength Exchange (PSE). Unlike the previous two schemes, here the particles' position and the core size are not updated, but it is the circulation strength $\vec{\alpha}^p(t)$ itself, which is exchanged with neighboring particles to simulate viscous diffusion.

  Thereby, particles are so allowed to represent numerical volumes of vorticity, instead of cores. This, on the other hand, requires the ability to diffuse vorticity for neighboring particles.

As reported in the work of Berdowski[4], the implementation of the PSE scheme always gives the most satisfactory results in the end, particularly regarding the stability of the simulation.
It mitigates instabilities as stresses resulting from intense stretching, sharp curvatures, and vortex roll-up; moreover, entanglement is effectively dealt with.
Due to the aforementioned reasons, it will be implemented the PSE scheme during this thesis work, as will hereafter be detailed explained.

The idea behind this method relies upon the assumption that is possible to approximate the diffusion operator (the Laplacian) with an integral operator, that is further discretized using the particle representation. As reported by Wincklemans[6], this is valid in $\mathbb{R}^n$, but it will be shown only in $\mathbb{R}^3$ during this work.

The Laplacian of some function $f(\vec{x})$ can be thereby discretized as

$$\nabla^2 f(\vec{x}) \simeq 2 \int [f(\vec{y}) - f(\vec{x})] \eta_\sigma(\vec{x} - \vec{y}) d\vec{y} \tag{3.35}$$

Where η_σ is essentially, as stated by Wincklemans, an approximation to the kernel for heat equation. It is defined as

$$\eta_\sigma = \frac{1}{\sigma^5} \eta \left(\frac{|\vec{x}|}{\sigma} \right) \tag{3.36}$$

With $\eta(\rho)$ defined as

$$\eta(\rho) = -\frac{\zeta'(\rho)}{\rho} \tag{3.37}$$

Here ζ is a radially symmetric regularization function that satisfies the equation below ($\zeta' = \frac{d\zeta}{d\rho}$).

$$\int_0^\infty |\zeta'(\rho)| \rho^{3+r} d\rho < \infty \tag{3.38}$$

The approximation done in equation 3.35 is a good one since, as reported by Wincklemans, the difference between $\nabla^2 f(\vec{x})$ and the 3.35 is, in the appropriate norm, $\mathcal{O}(\sigma^r)$.
Still, Wincklemans showed that the lower algebraic smoothing does not satisfy the constraint given by equation 3.38, therefore it is a poor choice to model diffusion with.

Let's consider now the following convection-diffusion equation.

$$\frac{\partial f}{\partial t} + \nabla \cdot (f \vec{u}) = \nu \nabla^2 f \tag{3.39}$$

As stated before, it can be approximated as

$$\frac{\partial f}{\partial t} + \nabla \cdot (f\vec{u}) = 2\nu \int [f(\vec{y}) - f(\vec{x})]\eta_\sigma(\vec{x} - \vec{y})d\vec{y} \tag{3.40}$$

Consider now a particle approximation to $f(\vec{x}, t)$ as

$$f_\sigma(\vec{x}, t) = \sum_{p=1}^{N_p} f^p(t) vol^p(t) \zeta_\sigma(\vec{x} - \vec{x}^p(t)) \tag{3.41}$$

This allows for a particle solution to the convection-diffusion equation 3.39, leading to

$$\begin{aligned} \frac{d\vec{x}^p}{dt} &= \vec{u}^p(\vec{x}^p, t) \\ \frac{d}{dt} vol^p &= vol^p(t) \nabla \cdot \vec{u}^p(\vec{x}^p, t) \\ \frac{d}{dt}(f^p vol^p) &= 2\nu vol^p(t) \sum_{q=1}^{N_p} vol^q(t)[f^q(t) - f^p(t)]\eta_\sigma(\vec{x}^p(t) - \vec{x}^q(t)) \end{aligned} \tag{3.42}$$

Then, replacing $f^p(t)$ with $\vec{\omega}^p(t)$ leads to the final discretization of viscous diffusion term.

$$\left| \frac{d\vec{\alpha}^p(t)}{dt} \right|_{visc} = 2\nu \sum_{q=1}^{N_p} (vol^p(t)\vec{\alpha}^q(t) - vol^q(t)\vec{\alpha}^p(t))\eta_\sigma(\vec{x}^p(t) - \vec{x}^q(t)) \tag{3.43}$$

According to Wincklemans, the double approximation done to discretize the diffusion operator results in an error of $\mathcal{O}(\nu(\sigma^r + \frac{h^m}{\sigma^{m+1}}))$, where h is the spacing between the particles.
Only for small values of ν this error is lower than the one due to the convective term discretization (i.e vortex stretching), which is $\mathcal{O}(\sigma^r + \frac{h^m}{\sigma^m})$.
Anyway, these errors are valid in the assumption that $\eta(\rho) > 0 \quad \forall\rho$, which further requires regularization functions of the second order, such as the HOA functions.
Although, Wincklemans states that $\eta(\rho)$ not positive for all values of ρ still implies an error lower than the one due to the discretization of the convective term. This error is $\mathcal{O}(\nu(\sigma^{r-2} + \frac{h^m}{\sigma^{m+1}}))$.

An important remark is that $\zeta(\rho)$ not positive for all values of ρ implies that $\eta(\rho)$ is not positive for all values of ρ, but $\zeta(\rho) > 0 \quad \forall\rho$ does not guarantee that $\eta(\rho) > 0 \quad \forall\rho$.
Hence, it is a further reason to choose the HOA regularization, since it leaves the freedom to have any value of viscosity.

Although not investigated in this work, it can be possible to generalize the formulation to not constant values of ν.

Lastly, it has to be noticed that the particle discretization of the integral approximation of the Laplacian is conservative, i.e, for the viscous part

$$\frac{d}{dt}\sum_{p=1}^{N_p}\vec{\alpha}^p(t) = 0 \tag{3.44}$$

This implies that only the treatment of the convective term in the vorticity-velocity equation can affect the total vorticity $\sum_{p=1}^{N_p}\vec{\alpha}^p(t)$.
As previously stated and as it will be further explained in section 3.7, the only scheme that conserves total vorticity is the transpose one.

3.4 Complete evolution equations

Substituting in the previously derived final equations the high order regularization functions 3.31, it can be derived the final system of evolution equations which has to be solved for particles' position and strength.

$$\frac{d\vec{x}^p}{dt} = -\frac{1}{4\pi}\sum_{q=1}^{N_p}\frac{|\vec{x}^p(t)-\vec{x}^q(t)|^2+\frac{5}{2}\sigma^2}{(|\vec{x}^p(t)-\vec{x}^q(t)|^2+\sigma^2)^{\frac{5}{2}}}(\vec{x}^p(t)-\vec{x}^q(t))\times\vec{\alpha}^q(t) \tag{3.45}$$

$$\begin{aligned}\frac{d\vec{\alpha}^p}{dt} = \frac{1}{4\pi}\sum_{q=1}^{N_p}\Bigg[&\frac{|\vec{x}^p(t)-\vec{x}^q(t)|^2+\frac{5}{2}\sigma^2}{(|\vec{x}^p(t)-\vec{x}^q(t)|^2+\sigma^2)^{\frac{5}{2}}}\vec{\alpha}^p(t)\times\vec{\alpha}^q(t)+\\ +3&\frac{|\vec{x}^p(t)-\vec{x}^q(t)|^2+\frac{7}{2}\sigma^2}{(|\vec{x}^p(t)-\vec{x}^q(t)|^2+\sigma^2)^{\frac{7}{2}}}(\vec{\alpha}^p(t)\cdot(\vec{x}^p(t)-\vec{x}^q(t))\times\vec{\alpha}^q(t))(\vec{x}^p(t)-\vec{x}^q(t))+\\ &+105\nu\frac{\sigma^4}{(|\vec{x}^p(t)-\vec{x}^q(t)|^2+\sigma^2)^{\frac{9}{2}}}(vol^p\vec{\alpha}^q(t)-vol^q\vec{\alpha}^p(t))\Bigg]\end{aligned} \tag{3.46}$$

Solving the system made by equations 3.45 and 3.46 every time-step it can be traced the position of every vorton which composes the discretized flowfield, so it has to be evaluated $\vec{x}^p(t)$ and $\vec{\alpha}^p(t)$ for all $p = 1, \ldots, N_p$.

3.5 Particle relaxation

As previously stated, the particle representation leads to a vorticity field that is generally not divergence-free, even if the regularization is applied. This appears to

be one of the major drawbacks of VPM, so in the following section the possibility of making the flowfield divergence-free is discussed.

Cottet[5] noted that the classic scheme in the vortex stretching modeling manages to keep the flow divergence-free reasonably well by construction. As long as the particles maintain sufficient overlap, the initial disturbance at time zero of the divergence is not significantly amplified.
Additionally, viscous diffusion seems to have a very positive effect on the treatment of divergence.
However, the here implemented transpose scheme does not seems to have the same properties, so the handling of divergence has to be considered.

In the following paragraph are further discussed two possible methods to help the flowfield have a divergence-free property.

3.5.1 Relaxation of the particle divergence by Beale

This method relies upon the assumption that each particle's circulation strength is, at initial simulation time $t = 0$, given by

$$\alpha^p(0)_{old} = \vec{\omega}^p(\vec{x}^p,0)vol^p \tag{3.47}$$

The vorticity field $\vec{\omega}_\sigma$ is not a good representation of the exact vorticity field $\vec{\omega}$ so, in order to obtain the latter, it has to be imposed that $\vec{\omega}_\sigma = \vec{\omega}$, writing that

$$\vec{\omega}^p_\sigma(\vec{x}^p, t) = \sum_{q=1}^{N_p} \vec{\alpha}^p(0)_{new}\zeta_\sigma(\vec{x}^p(0) - \vec{x}^q(0)) = \frac{\vec{\alpha}^p(0)_{old}}{vol^p} \tag{3.48}$$

This is essentially equivalent to a linear system in the form of

$$[A]\vec{\alpha}^p(0)_{new} = \vec{\alpha}^p(0)_{old} \tag{3.49}$$

Since $\vec{\alpha}^p(0)_{new}$ is really close to $\vec{\alpha}^p(0)_{old}$, then the coefficient matrix $[A]$ is almost equal to the identity matrix $[I]$, so the system quickly becomes impossible to be solved simply inverting matrix $[A]$, as the number of particles grows.

Taking advantage of the fact that $\vec{\alpha}^p(0)_{old}$ are good guesses for $\vec{\alpha}^p(0)_{new}$, it can be easily implemented an iterative scheme, as proposed by Beale[24], that recasts equation 3.49 as

$$([A] - [I])\vec{\alpha}^p(0)_{new} + \vec{\alpha}^p(0)_{new} = \vec{\alpha}^p(0)_{old} \tag{3.50}$$

And then provides the following iterative scheme

$$\vec{\alpha}^{p,n+1}(0)_{new} = \vec{\alpha}^{p}(0)_{old} + \vec{\alpha}^{p,n}(0)_{new} - \sum_{q=1}^{N_p} \vec{\alpha}^{p,n}(0)_{new} vol^q \zeta_\sigma(\vec{x}^p(0) - \vec{x}^q(0)) \quad (3.51)$$

Since $A_{pq} = \zeta_\sigma(\vec{x}^p(0) - \vec{x}^q(0))$.

It has been shown as this method converges with a number n of iteration of the order between 5 (Wincklemans[6]) and 15 (Calabretta[7]).

Although this has been shown for $t = 0$, it is possible to adjust the vorticity field to make it divergence-free for all values of t.

3.5.2 Relaxation of the particle divergence by Pedrizzetti

A more practical approach, that does not require the iterative solution of a linear system of equations, can be found in the work of Pedrizzetti[25], named the Divergence Filtering Method (DFM).

Each time-step Δt the particles' circulation strength is updated accorting to

$$\vec{\alpha}^{p}(t)_{new} = (1 - f\Delta t)\vec{\alpha}^{p}(t)_{old} + f\Delta t\, \vec{\omega}^{p}_{\sigma}(\vec{x}^p, t) \left| \frac{\vec{\alpha}^{p}(t)_{old}}{\vec{\omega}^{p}_{\sigma}(\vec{x}^p, t)} \right| \quad (3.52)$$

Where f is a time-step dependent frequency factor.
As stated by Pedrizzetti itself, this method does not explain other than a mathematical relaxation (so it does not justify it physically), but in practice, the results obtained seem to be satisfactory.
The frequency factor f has to be manually tuned along the time step size used, but it turns out that a very small value (such as $f \approx 0.2$) often already provides desirable results.

Modification by Alvarez

As pointed out by Alvarez in his work[17], the divergence relaxation by Pedrizzetti unintentionally tends to decrease the circulation strength magnitude.
This effect is more evident as $\vec{\alpha}$ and $\vec{\omega}$ are prone to be orthogonal to each other. Moreover, this could be considered a key part of the numerical stability that this relaxation is capable to achieve.
However, it might be necessary to minimize all the effects of a numerical diffusion, so Alvarez himself provided a new version of the Pedrizzetti's relaxation, which a

more detailed explanation can be found in [17], which calculates the new circulation strength as

$$\vec{\alpha}^p(t)_{new} = |\vec{\alpha}^p(t)_{old}| \frac{(1 - f\Delta t)\frac{\vec{\alpha}^p(t)_{old}}{|\vec{\alpha}^p(t)_{old}|} + f\Delta t \frac{\vec{\omega}_\sigma^p(\vec{x}^p,t)_{old}}{|\vec{\omega}_\sigma^p(\vec{x}^p,t)_{old}|}}{\sqrt{1 - 2(1 - f\Delta t) f\Delta t \left(1 - \frac{\vec{\alpha}^p(t)_{old}}{|\vec{\alpha}^p(t)_{old}|} \cdot \frac{\vec{\omega}_\sigma^p(\vec{x}^p,t)_{old}}{|\vec{\omega}_\sigma^p(\vec{x}^p,t)_{old}|}\right)}} \tag{3.53}$$

However, as stated by the author[17], this new relaxation is not able to damp the instabilities that arise when the vorticity of the lifting surface is evaluated with the lifting line theory (which is the case of this thesis work) or with surface models, while the original Pedizzetti's formulation instead it is.

For this reason, the present work makes usage of the original Pedrizzetti's relaxation.

3.6 Particle remeshing

In the regularized vortex particle description of the flow-field, the error norms for the vorticity $\vec{\omega}_\sigma$ and velocity $\vec{v}_sigma$ fields go to zero as the number of particles increases, and the smoothing radius decreases, subject to the so called overlapping condition[6]

$$\frac{h_{res}}{\sigma} < 1 \tag{3.54}$$

To satisfy this requirement, particle splitting and particle merging are applied.

Since the VPM is completely mesh-free, it is interesting to implement a remeshing method that is mesh-free too.
To address this, it has been implemented the remeshing method presented by Wincklemans and Leonard[31].

The particle splitting is necessary to ensure that the vorticity field is well discretized at every time-step. When the circulation strength module of the particle is twice the one at initial time, then the particle is split in two particles along the direction of the original vorticity.

$$\vec{x}^{1,2}_{split}(t) = \vec{x}^p(t) \pm \frac{1}{4}\sigma \frac{\vec{\alpha}^p(t)}{|\vec{\alpha}^p(t)|} \tag{3.55}$$

The circulation strength of the 2 new particles is half the one of the splitted particle.

$$\vec{\alpha}^{1,2}_{split}(t) = \frac{1}{2}\vec{\alpha}^p(t) \tag{3.56}$$

The particle merging instead is used both to maintain a uniform vorticity distribution and to speed-up the calculation, since it reduces the number of particles in the field.
The merging is applied when two particles are closer than a prescribed distance and when their circulation strengths are nearly aligned.

$$\begin{cases} |\vec{x}^p(t) - \vec{x}^q(t)| < \frac{1}{4}\sigma \\ 1 - \frac{\vec{\alpha}^p(t)\cdot\vec{\alpha}^q(t)}{|\vec{\alpha}^p(t)||\vec{\alpha}^q(t)|} \geq 10^{-4} \end{cases} \tag{3.57}$$

When these two conditions are simultaneously matched, then the two particles merge in a single one with position

$$\vec{x}_{merge}(t) = \frac{|\vec{\alpha}^p(t)|\vec{x}^p(t) + |\vec{\alpha}^q(t)|\vec{x}^q(t)}{|\vec{\alpha}^p(t)| + |\vec{\alpha}^q(t)|} \tag{3.58}$$

And circulation strength

$$\vec{\alpha}_{merge}(t) = \vec{\alpha}^p(t) + \vec{\alpha}^q(t) \tag{3.59}$$

3.7 Performance parameters

One of the major components of verifying that the VPM scheme is correctly implemented is defining a measurement tool to use for comparison. This can be assessed through the so-called invariants of the flow, which fall into two categories: linear and quadratic.
Linear ones are the total vorticity $\vec{\Omega}$, the linear impulse $\vec{I}$ and the angular impulse $\vec{A}$.
Quadratic ones instead are enstrophy ε, kinetic energy E and helicity $\mathcal{H}$.

Invariants, as the name indicates, are conserved for a real fluid, but for a discretized field they can divert from the real solution. As it will be further seen in the validation chapter (chapter 5), the more these terms are close to the real solution, the more is accurate the VPM scheme.
However, to distinguish the real and the discretized cases, the invariants in the discretized case are called diagnostics.

Moreover, it is important to correctly account for diagnostics (both linear and quadratic) because they represent a valuable tool to assess the level of numerical error in the implemented scheme, since the simple confront between the actual behavior of diagnostics and the expected one from the real solution can be a good representation of the scheme's accuracy.

In addition, the performance of each diagnostic is dependent on specific attributes of the quality of the simulation, so it is important to have a clear picture of the entire diagnostics set, to quickly ensure the maximum possible fidelity of the VPM scheme concerning the physical reality.

The diagnostics explicit relations are dependent on the regularization function chosen, so this dissertation will be examined the ones relative to the high order algebraic regularization function.

The linear diagnostics should all be conserved while the discretized vorticity field remains a good approximation of the true one.
The transpose scheme in particular should conserve the total vorticity, and is indeed the scheme utilized in this work.
Moreover, the linear impulse should be conserved as long as the discretized vorticity field approximately remains divergence-free, and so tracking its evolution over time is a good way to monitor the divergence-free condition of the discretized field, as it should correctly be.
The quadratic diagnostics have two formulations, one of which is for the theoretical divergence-free field, and the other is instead for the regularized particle approximation.

3.7.1 Linear diagnostics

The linear diagnostics should be conserved for both inviscid and viscous flow conditions but, it is not guaranteed since the discretized flowfield can be not divergence-free.
However, it has been shown that the (nearly) divergence-free condition implies that linear diagnostics should be (nearly) conserved, so one can retain the latter to quickly observe if the implemented scheme is a good approximation of the real divergence-free flowfield or not.

Total vorticity

Total vorticity is simply defined as the summation of all the particles' circulation strengths.

$$\vec{\Omega}(t) = \sum_{p=1}^{N_p} \vec{\alpha}^p(t) \tag{3.60}$$

It turns out that only the transpose scheme conserves the total vorticity over time by construction, but only in the singular case. This implies that the latter scheme is preferred over the other two in the singular case when there isn't still a well-defined best practice in the regularized case.

However, it is still applied the transpose scheme, also to remain coherent with the other works that have been reviewed during the realization of this thesis.

As can be easily seen, the total vorticity definition is the same both for singular and regularized particle discretization.

Linear impulse

Linear impulse is defined as

$$\vec{I}(t) = \frac{1}{2}\sum_{p=1}^{N_p} \vec{x}^p(t) \times \vec{\alpha}^p(t) \tag{3.61}$$

As in the previous diagnostic term, also linear impulse is independent of the discretization technique (singular or regularized).

Angular impulse

Angular impulse is defined as

$$\vec{A}(t) = \frac{1}{2}\sum_{p=1}^{N_p} \vec{x}^p(t) \times (\vec{x}^p(t) \times \vec{\alpha}^p(t)) - \frac{1}{3}C\sigma^2\vec{\Omega}(t) \tag{3.62}$$

Where

$$C = 4\pi \int_0^\infty \zeta(\rho)\rho^4 d\rho \tag{3.63}$$

Differently from the previous two, this diagnostic term depends on the regularization, as in C appears the regularization function $\zeta(\rho)$.
In case the high order algebraic regularization is chosen, then C can be easily computed as $C = \frac{3}{2}$; this is not straightforward in other cases.

However, on the assumption that the flowfield maintains the divergence-free condition, the term that multiplies C vanishes, as $\vec{\Omega}$ goes to zero (total vorticity is conserved and at time $t = 0$ it has to be nullus), so in that case also angular momentum can be considered independent from the eventual regularization.
If the aforementioned hypothesis stands for, then the angular impulse is simply given as

$$\vec{A}(t) = \frac{1}{2}\sum_{p=1}^{N_p} \vec{x}^p(t) \times (\vec{x}^p(t) \times \vec{\alpha}^p(t)) \tag{3.64}$$

Another formulation of angular impulse can be found in several works, and is further written to completeness, although the one which will be used in this work is the first one (equation 3.62).

$$\vec{A}(t) = \frac{1}{3}\sum_{p=1}^{N_p} \vec{x}^p(t) \times (\vec{x}^p(t) \times \vec{\alpha}^p(t)) - \frac{2}{9} C\sigma^2 \vec{\Omega}(t) \tag{3.65}$$

3.7.2 Quadratic diagnostics

The quadratic diagnostics represent a more challenging problem to be solved since they are defined as the integral of a product between vorticity and velocity fields, which are difficult to analytically integrate with the case of regularized particles.

As stated by Wincklemans[6], it's possible to make an approximation, evaluating the integrals considering that only one of the two terms is relative to the regularized case, while the second one remains to be singular.
However, this assumption has several levels of validity, depending on the diagnostic term to be evaluated; in particular, the hypothesis is good for kinetic energy, moderately appropriate for helicity, and poor for enstrophy.

Moreover, Wicklemans believes that the HOA regularization function is the only one that allows an analytical integration, always on the assumption that only one term of the product is regularized.

In the following section, it will be presented, both for kinetic energy and enstrophy, two forms. The first one accounts for the eventual not divergence-free condition of the flowfield, while the second one is technically valid only if the flowfield satisfies the divergence-free condition instead.
Moreover, the comparison between these two terms can be a useful tool to quickly see how well the divergence-free condition is preserved over time. This check further indicates the eventual presence of numerical error and can be carried out through the validation phase, as it will be seen in chapter 5.

This assumption cannot be done considering the helicity, so it turns out to be less useful due to this impossibility itself. For this reason, helicity is not accounted for in the validation step.

Helicity

It is defined as

$$\mathcal{H} = \int \vec{\omega} \cdot \vec{u} d\, vol \tag{3.66}$$

Substituting the vorticity and velocity fields, in the singular case, one obtains

$$\mathcal{H} = \frac{1}{4\pi} \sum_{p,q=1}^{N_p} \frac{1}{|\vec{x}^p(t) - \vec{x}^q(t)|^3} [(\vec{x}^p(t) - \vec{x}^q(t)) \cdot (\vec{\alpha}^p(t) \times \vec{\alpha}^q(t))] \tag{3.67}$$

While in the so-called semi-regularized case it can be written as

$$\mathcal{H}_\sigma = \frac{1}{4\pi} \sum_{p,q=1}^{N_p} \frac{|\vec{x}^p(t) - \vec{x}^q(t)|^2 + \frac{5}{2}\sigma^2}{(|\vec{x}^p(t) - \vec{x}^q(t)|^2 + \sigma^2)^{\frac{5}{2}}} [(\vec{x}^p(t) - \vec{x}^q(t)) \cdot (\vec{\alpha}^p(t) \times \vec{\alpha}^q(t))] \tag{3.68}$$

Helicity is a measure of net linkage of vortex lines; since vortex lines can reconnect in a viscous VPM flow, helicity is then not conserved, whereas it is in an inviscid flow.

Kinetic energy

It is defined as

$$E = \frac{1}{2} \int \vec{u} \cdot \vec{u} d\, vol \tag{3.69}$$

In the singular case it can be obtained that

$$\begin{aligned} E = \frac{1}{16\pi} \sum_{p,q=1, p\neq q}^{N_p} \frac{1}{|\vec{x}^p(t) - \vec{x}^q(t)|^3} [|\vec{x}^p(t) - \vec{x}^q(t)|^2 \vec{\alpha}^p(t) \cdot \vec{\alpha}^q(t) + \\ + (\vec{x}^p(t) - \vec{x}^q(t)) \cdot \vec{\alpha}^p(t) (\vec{x}^p(t) - \vec{x}^q(t)) \cdot \vec{\alpha}^q(t)] \end{aligned} \tag{3.70}$$

The summation is done over $p \neq q$ since the velocity field is singular.
In the semi-regularized case instead, one can obtain (recalling that $\rho = \frac{|\vec{x}^p(t) - \vec{x}^q(t)|}{\sigma}$)

$$\begin{aligned} E_\sigma = \frac{1}{16\pi} \sum_{p,q=1}^{N_q} \Bigg[2 \frac{\rho}{\sqrt{\rho^2+1}} \vec{\alpha}^p(t) \cdot \vec{\alpha}^q(t) + \\ + \frac{\rho^3}{(\rho^2+1)^{\frac{3}{2}}} \left(\frac{(\vec{x}^p(t) - \vec{x}^q(t)) \cdot \vec{\alpha}^p(t)}{|\vec{x}^p(t) - \vec{x}^q(t)|} \frac{(\vec{x}^p(t) - \vec{x}^q(t)) \cdot \vec{\alpha}^q(t)}{|\vec{x}^p(t) - \vec{x}^q(t)|} - \vec{\alpha}^p(t) \cdot \vec{\alpha}^q(t) \right) \Bigg] \end{aligned} \tag{3.71}$$

On the potentially incorrect assumption that the vorticity field is divergence-free, the regularized kinetic energy can be written as

$$E^f_\sigma = \frac{1}{8\pi} \sum_{p,q=1}^{N_p} \frac{|\vec{x}^p(t) - \vec{x}^q(t)|^2 + \frac{3}{2}\sigma^2}{(|\vec{x}^p(t) - \vec{x}^q(t)|^2 + \sigma^2)^{\frac{3}{2}}} \vec{\alpha}^p(t) \cdot \vec{\alpha}^q(t) \tag{3.72}$$

The kinetic energy is conserved in inviscid flow, while in viscous flows it decays as

$$\frac{dE}{dt} = -\nu\varepsilon \tag{3.73}$$

This makes the kinetic energy a good indicator for viscous flow simulations.

Enstrophy

It is defined as

$$\varepsilon = \int \vec{\omega} \cdot \vec{\omega} d\, vol \tag{3.74}$$

This integral cannot be analytically evaluated for singular particles, since the integral of the square of the delta distribution is undefined[15].
However, the semi-regularized enstrophy is

$$\begin{aligned} \varepsilon_\sigma = \frac{1}{8\pi} \sum_{p,q=1}^{N_p} \frac{\sigma^3}{(|\vec{x}^p(t) - \vec{x}^q(t)|^2 + \sigma^2)^{\frac{9}{2}}} \\ \Big[(|\vec{x}^p(t) - \vec{x}^q(t)|^2 + \sigma^2) \\ (2|\vec{x}^p(t) - \vec{x}^q(t)|^4 + 7\sigma^2 |\vec{x}^p(t) - \vec{x}^q(t)|^2 + 20\sigma^4) \vec{\alpha}^p(t) \cdot \vec{\alpha}^q(t) + \\ -3(4|\vec{x}^p(t) - \vec{x}^q(t)|^4 + 18\sigma^2 |\vec{x}^p(t) - \vec{x}^q(t)|^2 + 7\sigma^4) \\ (\vec{x}^p(t) - \vec{x}^q(t)) \cdot \vec{\alpha}^p(t) (\vec{x}^p(t) - \vec{x}^q(t)) \cdot \vec{\alpha}^q(t) \Big] \end{aligned} \tag{3.75}$$

Since in the original work of Wincklemans[6] the enstrophy relationship seems to be incorrect, in this work it has been reported the correct one, that can be found in the work of Singh[15] and, as the author states, it has been obtained through personal communication with Wincklemans.

Moreover, incorrectly assuming that the vorticity field is divergence-free, one can obtain

$$\varepsilon_\sigma^f = \frac{1}{4\pi} \sum_{p,q=1}^{N_p} \frac{15}{2} \frac{\sigma^4}{(\vec{x}^p(t) - \vec{x}^q(t)|^2 + \sigma^2)^{\frac{7}{2}}} \vec{\alpha}^p(t) \cdot \vec{\alpha}^q(t) \tag{3.76}$$

Enstrophy is a measure of stretching and is therefore not conserved whenever stretching is manifested. Since stretching is present both in viscous and inviscid cases, enstrophy is not conserved.

3.8 Code implementation

Now that they have been explained the theoretical basis of the Vortex Particle Method, it is necessary to implement the evolution equations in a Matlab code, to finally evaluate the performances of rotors.

As previously mentioned, the developed code is a substantial modification of the code made by Alì[1] in his work.

Specifically, it has been modified the wake evaluation, since the VPM treats the wake as a set of particles that evolves following the VPM evolution equations. Therefore, every particle's position is updated according to evolution equations, also considering the induced velocity by the panels in which every blade is discretized and the eventual inflow velocity.

The code is a stand-alone tool where the user can insert several data as numerical input and can choose along a few settings to best match its necessities in terms both of accuracy and computational cost.
Both direct and FMM computations are parallelized and vectorized, to achieve the best computational speed possible.
The output are produced as plots of the desired performances as selected by the user and it is possible also to plot the geometry of both blades and wake every time-step, with the eventual aim of producing a video animation of the wake topology.

The user has to insert also the blade geometry and the polar function, so the code can discretize the first one and can use the second one to calculate the circulation distribution every time step.

3.8.1 Code flowchart

As indicated in the flowchart in figure 3.2, after the data acquisition and the following discretization and initialization of the problem, a temporal loop is performed, in which the code calculates the circulation on the blade, generates the particles, performs a divergence relaxation, performs the remeshing, updates the position and the circulation strength of the particles and, finally, rotates the blade of the corresponding angular step.

The evolution equations are solved with a direct calculation if the number of particles is less than a specified threshold (the verification study in chapter 5 provides a correlation between the number of particles and the elapsed time for both direct and FMM calculations), otherwise, it uses the FMM algorithm.

3.8.2 MATLAB functions

Every task in the temporal loop is addressed by a specific Matlab function.
The following paragraphs deal with the main ones.

Circulation calculation

The circulation is evaluated with an iterative calculation, as explained by Alì[1] in his work.

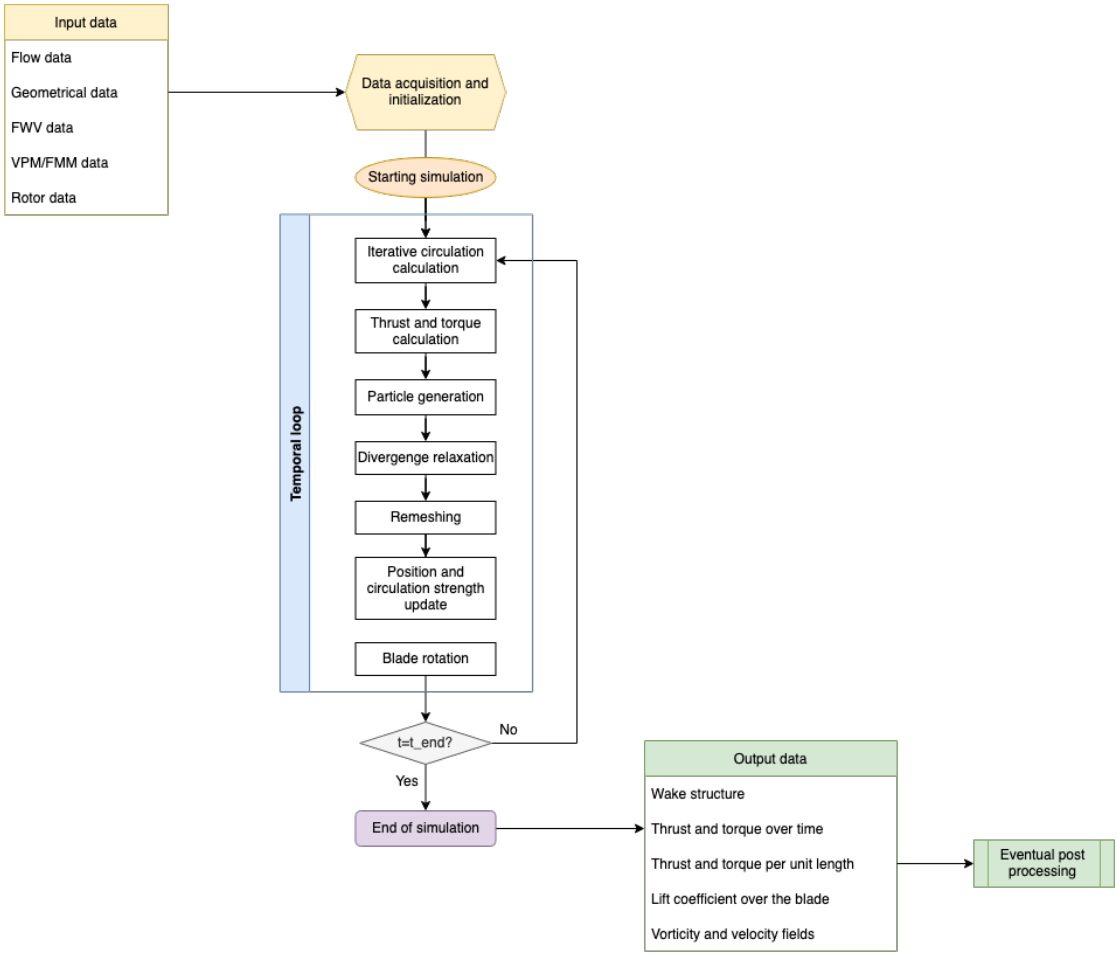


Figure 3.2: Code flowchart

This is addressed calculating the circulation according to the following equation

$$\Gamma = \frac{c_l(\alpha)\frac{1}{2}\left[\left(\vec{V}\cdot\vec{a}_1\right)^2+\left(\vec{V}\cdot\vec{a}_3\right)^2\right]dA}{\sqrt{\left[\left(\vec{V}\times d\vec{l}\right)\cdot\vec{a}_1\right]^2+\left[\left(\vec{V}\times d\vec{l}\right)\cdot\vec{a}_3\right]^2}} \tag{3.77}$$

where the velocity $\vec{V}$ is the velocity calculated on the control points, $\vec{a}_j$ is the j-th versor of the panel, according to van Garrel[32] notation, dA is the area of the panel and $d\vec{l}$ is the filament length. The panel geometry is calculated according to van Garrel work[32].
The velocity is seen as the sum of three contributions:

- Wind velocity

- Induced velocity by:
  - Panels
  - Particles
- Rotational velocity

So it can be evaluated as

$$\vec{V} = \vec{V}_{wind} + \vec{V}_{ind,pan} + \vec{V}_{ind,part} - \vec{\Omega} \times \vec{r} \tag{3.78}$$

where $\vec{\Omega}$ is the angular velocity of the blade and $\vec{r}$ is the vector radius of the panel control point from the axis origin.
The local incidence α is calculated as

$$\alpha = \arctan\left(\frac{\vec{V} \cdot \vec{a}_3}{\vec{V} \cdot \vec{a}_1}\right) \tag{3.79}$$

So it is the angle that the velocity vector $\vec{V}$ forms with the airfoil chord in the plane identified by the vectors $\vec{a}_1$ and $\vec{a}_3$.
The lift coefficient is calculated by interpolating data from an aerodynamics database or XFoil/CFD results.

Equation 3.77 is obtained by equaling two expressions for the lift on a blade panel; the first one, following the definition of lift force is

$$dL = c_l(\alpha)\frac{1}{2}\rho\left[\left(\vec{V} \cdot \vec{a}_1\right)^2 + \left(\vec{V} \cdot \vec{a}_3\right)^2\right] dA \tag{3.80}$$

while the second, following the Kutta-Jukowski theorem is

$$dL = \rho\Gamma\sqrt{\left[\left(\vec{V} \times d\vec{l}\right) \cdot \vec{a}_1\right]^2 + \left[\left(\vec{V} \times d\vec{l}\right) \cdot \vec{a}_3\right]^2} \tag{3.81}$$

The circulation is calculated every iteration and, if the guessed value differs from its more than a specified tolerance, then the previously guessed value is updated as

$$\Gamma^{k+1} = \Gamma^k + RF(\Gamma - \Gamma^k) \tag{3.82}$$

Where RF is a relaxation factor, which is necessary to avoid instability due to oscillations of Γ^k.
The convergence criterion is

$$\frac{|\max(\Gamma - \Gamma^k)|}{|\max\Gamma| + 1} \leq toll \tag{3.83}$$

The guessed value is 1 if the circulation is calculated for the first time in the temporal loop, otherwise, it is represented by the distribution at the previous time step.

Calculation of the induced velocity by the panels

To evaluate the induced velocity by the panels on both control points and particles coordinates it has been implemented the Biot-Savart law's regularized calculation presented by van Garrel[32] in his work, so, referring to his notation, the velocity is

$$\vec{V}_{ind}(P_c) = \frac{\Gamma}{4\pi} \frac{(|\vec{r}_1| + |\vec{r}_2|)(\vec{r}_1 \times \vec{r}_2)}{|\vec{r}_1||\vec{r}_2|(|\vec{r}_1||\vec{r}_2| + \vec{r}_1 \cdot \vec{r}_2) + (\delta l_0)^4} \tag{3.84}$$

Where l_0 is the filament length and δ is a percentage damping factor that is arbitrary.

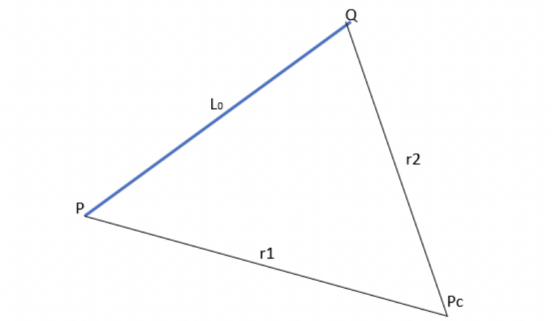


Figure 3.3: Biot-Savart law for the panel induced velocity calculation[32]

Thrust and torque calculation

In the same function as the previous, they are also calculated thrust and torque as the sum of every panel contribution.

The thrust on each panel is calculated as the sum of the vertical component of both lift and drag; the drag is calculated, as well as the lift, as

$$dD = c_d(\alpha)\frac{1}{2}\rho\left[\left(\vec{V}\cdot\vec{a}_1\right)^2 + \left(\vec{V}\cdot\vec{a}_3\right)^2\right]dA \tag{3.85}$$

Then, considering that the angle between the lift (perpendicular to the velocity in the plane individuated by vectors $\vec{a}_1$ and $\vec{a}_3$) and the vertical axis is

$$\beta - \alpha$$

where β is the angle between $\vec{a}_1$ and the horizontal axis, it can be calculated that

$$dTh = dL\cos(\beta - \alpha) - dD\sin(\beta - \alpha) \tag{3.86}$$

since the drag is perpendicular to the lift.

The torque is calculated instead as the one due to the horizontal component of both lift and drag with respect to the vector radius of the control point $\vec{r}$ (of components r_x, r_y and r_z in the cartesian xyz reference frame).

$$dT = \sqrt{r_x^2 + r_y^2}[dL\sin(\beta - \alpha) + dD\cos(\beta - \alpha)] \tag{3.87}$$

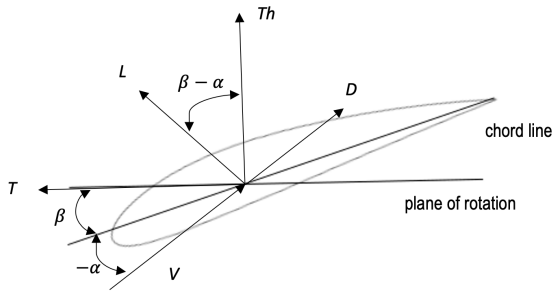


Figure 3.4: Scheme for thrust and torque calculation

Particles generation

The particle generation is done according to the formulation presented by Singh[15] in his work.
The first set of particles is shed in the wake, to account for the spatial variation of circulation on the blade, then a second set is shed, to account for temporal variation of circulation instead.
The first set is shed chordwise, while the second one is shed spanwise. Both of them are shed from the trailing edge position, which is obtained by translating the control point positions to the corresponding trailing edge coordinates.

To ensure the overlapping condition, the spatial resolution of the particles is calculated as

$$h_{res} = \frac{\sigma}{1.5}$$

Since, as stated before, it is necessary, due to numerical stability issues, that the particles' smoothing radius is greater than the spatial resolution.

A more detailed explanation on how the particles are generated can be found in appendix D.

Particle position and circulation strength update

The position of every particle is computed by integrating its velocity with the low storage four stages third-order Runge-Kutta method discussed in appendix C. The velocity of every particle is evaluated as the sum of several contributions:

- The velocity induced by all the particles in the field (equation 3.45).
- The velocity induced by all the panels in which the blades are discretized (equation 3.84).
- The eventual inflow velocity (given as input).

Instead, the circulation strength is updated according to equation 3.46.

Here, only the first contribution can be evaluated with FMM, since the second one will result in a useless overhead using the FMM algorithm (the number of panels - sources - is much less than the number of particles - targets). The third contribution is instead constant for every particle, so is added lastly.

3.9 Conclusions

Through this massive chapter, they have first presented the regularized VPM evolution equations, according to the HOA regularization, which has also dealt with the correct modeling of the vortex stretching term (with the transpose formulation, as it is the only one that conserves vorticity) and the viscous diffusion term.

Then, it has been addressed the difficult task of providing the code to be stable and also providing the velocity field to be as much as possible divergence-free; has been possible thanks to the introduction of particle remeshing and divergence relaxation.

Lastly, they have been presented with the so-called diagnostics (both linear and quadratic) tools to check and ensure the correct behavior of the code, and it has been shown a detailed code flowchart with further details on issues that have not been treated explicitly in the previous or next chapters.

Chapter 4

The Fast Multiple Method

As stated before, the Vortex Particle Method has a computational cost of $\mathcal{O}(N_p^2)$, so it is mandatory to scale it to a better one, especially intending to produce simulations that involve several revolutions (i.e a considerably high number of particles).
In particular, it is necessary to speed up the calculation of the influence, in terms of both induced velocity and circulation strength update, that every particle has on the others. This can be achieved, in general, with a so-called fast summation method, which the Fast Multipole Method is part of.

The Fast Multipole Method makes it possible to reduce the computational cost of the summation from $\mathcal{O}(N_p^2)$ to $\mathcal{O}(N_p \log N_p)$ or even to $\mathcal{O}(N_p)$.
In this work have been investigated two routines that have a computational cost of $\mathcal{O}(N_p)$ and are discussed in the sections below.

4.1 Fundamental theory

The basic idea behind the Fast Multipole Method is that the influence of all particles at a point is approximated by the influence of several clusters (at their center) of particles at the same point, in the form of multipole moments.
They are divided into clusters of only particles which are well separated from the point where it is necessary to calculate the influence of all particles, otherwise, the method will not converge.
The discrimination between well-separated and not well-separated zones (far-field and near field) is achieved by dividing the computational domain into cells (for a tridimensional problem, it is used an octree data structure). Then, the direct calculation is performed only for particles that are on the so-called nearest neighbor cells, while the FMM calculation is performed for the others.

For a more detailed explanation of this topic, the reader is referred to [27] and [26]. This section makes use of figures and notations that are the same as the ones used by Berdowski[4] in his work.

4.1.1 Summary of the steps

This method requires deep bookkeeping of the computational domain into a complex data structure, so it is mandatory to split the algorithm into several subsequent steps.

1. **Construction of the tree:** The computational domain is divided into refinement layers, with an increasing number of boxes, depending locally on the particle density.
2. **Local Multipole expansions:** They are calculated at the centers of the finest level's boxes.
3. **Translation of Local Multipole expansions:** The Local Multipole expansions are subsequently translated to the centers of coarser boxes. Now the field induced by the sources is completely described by multipole expansions.
4. **Computing far-field potential:** It is, therefore, possible to evaluate the far-field potential at each target point.
5. **Direct calculation of near field potential:** As previously mentioned, the near field is computed directly.

In the next lines, they will be discussed more in detail these steps. Figure 4.1 depicts what was just briefly explained. Although the next figures will show a 2D case of the Fast Multipole Method, it will be described as a 3D version instead.

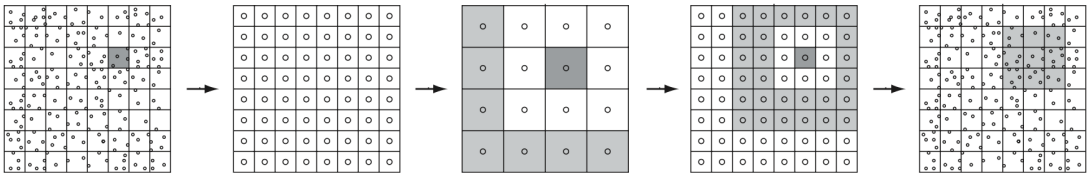


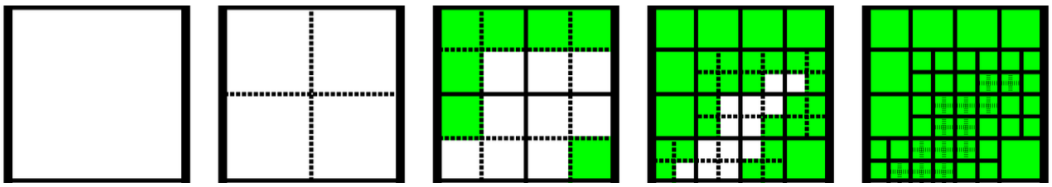
Figure 4.1: Flow of the FMM calculation

Construction of the tree

Firstly, the computational domain is subdivided into an octree; this means that, for each level, the domain is cut in a half, resulting in equally sized boxes. The

boxes obtained at a finer level from the same box at the coarser one are called children. Each box, therefore, holds 4 children in 2D and 8 children in 3D.

Until a certain condition is not satisfied (usually it is based on the maximum amount of particles in a box or a prescribed level of refinement), the subdivision continues. This step goes on until every box has reached the prescribed conditions. The child box at his finest division is called a leaf box and is shown in green in figure 4.2.



(a) Level 1 (b) Level 2 (c) Level 3 (d) Level 4 (e) Level 5

Figure 4.2: Generation of a 2D quadtree structure (octatree in 3D)

Local Multipole expansion

Let's now concentrate on the finest level of the octree structure. There, the multipole expansions of the potential of all the particles in each box are calculated about the box center. This is shown in figure 4.3a.

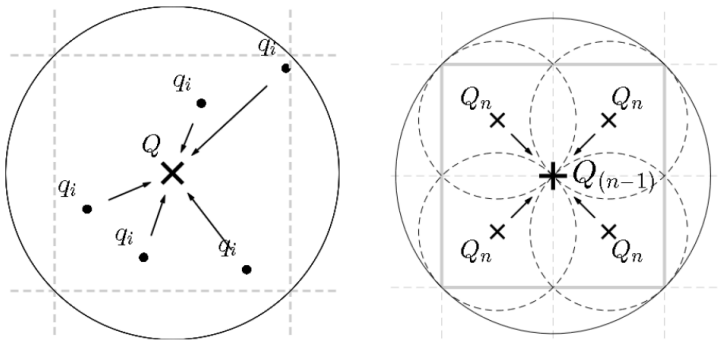
Then, they are shifted and translated to the centers of the parent boxes, so that each box at the coarser level holds a combination of 8 multipole expansions from its children. This is shown in figure 4.3b.

Far field computation

In this step, is firstly required to discriminate the far-field from the near field.
To achieve that it can be shown how the ratio θ between the diameter d of the circle in which is inscribed the box for that we want to evaluate the influence on the target and the distance x between the target and the box center has to be greater than 1. This is shown in figure 4.4

4.1.2 Calculation of the potential

In this paragraph is afforded the calculation of the laplacian potential $\Phi = q\frac{1}{||\vec{x}||}$ induced by a source of charge q in $\vec{x} = \vec{x}^p$ on a target point in $\vec{x} = \vec{x}^q$, in terms of



(a) Calculation of the potential about the box center

(b) Translation of the expansions to parent box

Figure 4.3: Local Multipole expansion

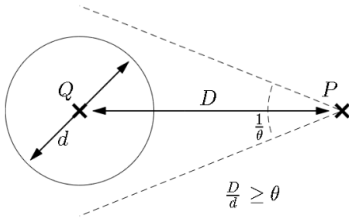


Figure 4.4: Far field discrimination

the Fast Multipole Method.

Let us firstly consider the coordinates of source and target points in a spherical reference frame, so that

$$\vec{x} = \{\rho,\, \theta,\, \phi\} \tag{4.1}$$

Then, it is recalled that the solution of the Laplace equation for the potential can also be expressed in the form

$$\Phi(\vec{x}^p, \vec{x}^q) = \sum_{n=0}^{\infty} q \frac{\rho_q^n}{\rho_p^{n+1}} P_n(u) \tag{4.2}$$

Where the P_n terms are the Legendre polynomials.
Equation 4.2 is indeed the far field potential on target $\vec{x}^p$ due to a unit-strength charge in $\vec{x}^q$.

The Legendre polynomial can be further expressed in terms of spherical harmonics

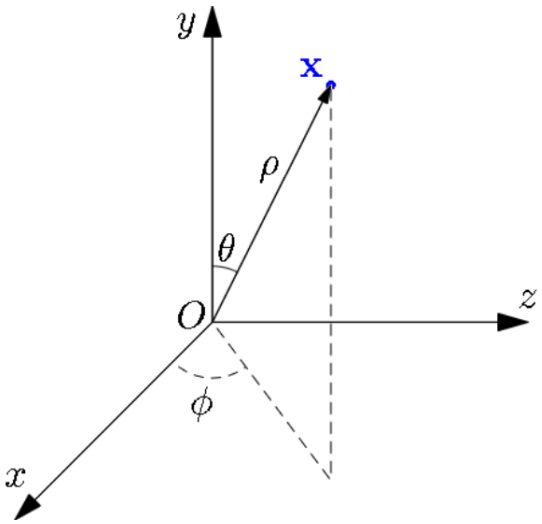


Figure 4.5: Spherical coordinates with respect the Cartesian reference frame

Y_n^m as

$$P_n(\cos\gamma_{p\to q}) = \sum_{m=-n}^{n} Y_n^{-m}(\theta_q,\phi_q)Y_n^m(\theta_p,\phi_p) \tag{4.3}$$

where $\gamma_{p\to q}$ is the angle between the points p and q.

The spherical harmonics, in their turn, can be expressed as

$$Y_n^m(\theta,\phi) = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}}P_n^{|m|}\cos\theta e^{im\theta} \tag{4.4}$$

It must be highlighted that the superscripts here does not represent a power but they indicate the order, both for Legendre polynomials and spherical harmonics.

Another important remark that needs to be done is that the P_n^m terms in equation 4.4 are independent from the P_n terms in equation 4.3, since they can be expressed as

$$P_n^m(u) = \frac{(2n-1)uP_{n-1}^m(u) - (n+m-1)P_{n-2}^m(u)}{n-m} \tag{4.5}$$

With the boundary conditions

$$P_{m+1}^m = (2m+1)uP_m^m(u) \tag{4.6}$$

$$P_m^m(u) = (-1)^m(2m-1)!!(1-u^2)^{\frac{m}{2}} \tag{4.7}$$

Here the "!!" operator denotes the double factorial operation, that is the product of odd integers only.

By writing $P_n(u)$ in equation 4.2 as indicated in equation 4.3 and generalizing all to a set of N_{part} particles, it is possible to describe the far field potential at the point p due to the N_{part} particles centered around q as

$$\Phi(\vec{x}^p, \vec{x}^q) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \sum_{i=1}^{N_{part}} q_i \frac{\rho_i^n}{\rho_p^{n+1}} Y_n^{-m}(\theta_i, \phi_i) Y_n^m(\theta_p, \phi_p) \tag{4.8}$$

The terms

$$M_n^m = \sum_{i=1}^{N_{part}} q_i \rho_i^n Y_n^{-m}(\theta_i, \phi_i) \tag{4.9}$$

are the multipole expansion coefficients and they account for the set of particles centered around p.
It can be therefore written that

$$\Phi(\vec{x}^p, \vec{x}^q) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{M_n^m}{\rho_p^{n+1}} Y_n^m(\theta_p, \phi_p) \tag{4.10}$$

The latter equation allows to describe the solution of the potential in terms of each multipole expansion of each source in a leaf box with respect the center of the box itself.

The last step that needs to be carried out is the translation of each particle expansion to the center of its box and from a certain level to the subsequent level (parent to children or vice versa).
In this way, the far field potential at $\vec{x}^p$ due to a set of particles translated to the center of their box and subsequently from that box to another box can be expressed as

$$\Phi(\vec{x}^p, \vec{x}^{Q_1}) = \sum_{l=0}^{\infty} \sum_{j=-l}^{l} D_l^j(\vec{x}^Q) O_n^m(\vec{x}^p - \vec{x}^Q) \tag{4.11}$$

Where

$$D_l^j(\vec{x}^{Q_1}, \vec{x}^{Q_0}) = \sum_{n=0}^{l} \sum_{\max(j+n-l,-n)}^{\min(j+n-l,n)} C_n^m(\vec{x}^{Q_0}, \vec{x}^{q_i}) I_{l-n}^{j-m}(\vec{x}^{Q_1} - \vec{x}^{Q_0}) \tag{4.12}$$

$$O_n^m(\vec{x}^p - \vec{x}^{Q_1}) = \frac{(-1)^j i^{|j|} Y_n^m(\theta_{Q_1 \to p}, \phi_{Q_1 \to p})}{A_l^j \rho_{Q_1 \to p}^{j+1}} \tag{4.13}$$

With

$$C_n^m(\vec{x}^{Q_0}, \vec{x}^q) = \sum_{i=1}^{N_{part}} q_i \rho_i^n Y_n^m(\theta_i, \phi_i) (-1)^{-n} i^{-|m|} A_n^m \tag{4.14}$$

$$I_{l-n}^{j-m}(\vec{x}^{Q_1} - \vec{x}^{Q_0}) = (-1)^{l-n} A_{l-n}^{j-m} \rho_{Q_0 \to Q_1}^{l-n} Y_{l-n}^{j-m}(\theta_{Q_0 \to Q_1}, \phi_{Q_0 \to Q_1}) \tag{4.15}$$

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!(n+m)!}} \tag{4.16}$$

The I and O functions are the so called inner and outer translation functions.
To have a clear overview on the notation used in this work the reader is referred to figure 4.6.

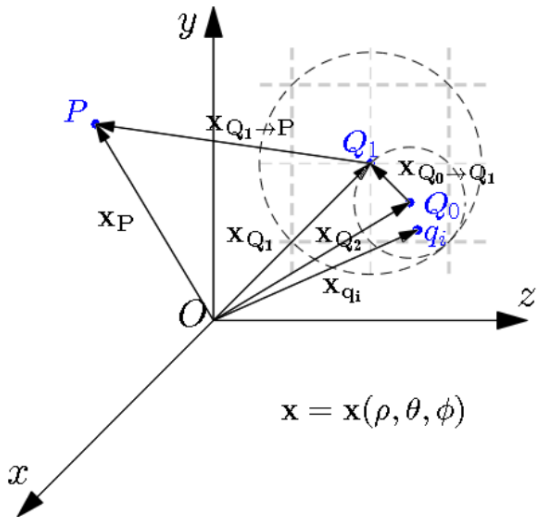


Figure 4.6: Explanation of the notation used in this paragraph

What just discussed is valid for a scalar potential, i.e a potential due to a scalar charge. The VPM evolution equations are, on the other hand, vectorial equations, so they could be all extended to a vectorial charge $\vec{q}$, to subsequently obtain a vectorial potential, which is, in our case, the streamfunction $\vec{\Psi}$ such that $\vec{u} = \nabla \times \vec{\Psi}$, as explained in chapter 3.
Unfortunately, the FMM does not provide any derivatives of the computed potential, so it would be necessary to evaluate them as a post-processing operation, adding computational effort to the calculation. This is further worsened considering that it would be also necessary to describe the aforementioned derivatives in spherical coordinates and then compute the same in cartesian ones and also considering that it would be required to regularize the laplacian kernel, as done in the evolution equations in chapter 3.

Due to this additional computational effort, in this work, it has been decided to recast the evolution equations to obtain them in terms of scalar potentials, which are directly computed with the Fast Multipole Method. Their complete derivation is collected in appendix B.
This procedure has been carried out also by Seetharaman in his work[8] and, how it will be highlighted in the next sections, this has produced a computational effort that can be however considered more than acceptable.

Another way to treat this last issue could be using the so-called complex step derivative approximation[33] (CSDA), which has been successfully used by Alvarez in his work[13].

4.2 FMM algorithms

This section will briefly describe two investigated C++ routines to execute an FMM calculation.
Both of them can calculate the potential Φ in a set of target points due to another set of sources, with their coordinates and their charges.
In addition, they are theoretically kernel independent, so it can be afforded the calculation of mostly every potential.

Both of them are available as open-source libraries, and they come with some default routines to compute the potential with the most common kernels.
Since the kernels used in this work are not included in the default settings for any of them, they have been modified to perform such a computation.
Moreover, the default routines generate random coordinates and charges, so they have been further modified to take as input the set of coordinates and charges, and then give as output the potential, which would not otherwise be collected in memory.

Both the two routines perform an error computation about the direct calculation and also a measurement of the computational time for both FMM and direct calculation, highlighting the convenience of the first one from a certain number of points.

The validation, the further modification of these algorithms, and the comparison between them are afforded in chapter 5.

4.2.1 The BBFMM3D algorithm

This algorithm is based on the work of Fong et al.[3], which used a different methodology to develop an FMM routine, concerning what was explained in the previous

section.
This algorithm can be considered kernel independent since the elements of the kernel matrix (the matrix where are stored the kernel values for all sources and targets values) are obtained through Chebyshev nodes as interpolation basis .

The BBFMM3D, as stated by the authors, requires a small pre-computation time and makes use of the minimum number of coefficients to represent the far-field potential. This results in a suitable routine for complicated kernels, such as the ones that come from the VPM evolution equations.

For a more detailed overview of this method, the reader is referred to the work of Fong[3].

Routine explanation

The library comes with four different routines: two for the laplacian kernel and the other two for a user-defined kernel. For each kernel, there is a routine for random input and another one for external input, from a `.bin` file. This is particularly suitable in Matlab since it is possible to generate such a file directly in the code.

The standard routine requires the definition of sources and targets coordinates, sources' charges, and a set of metadata, where are stored the length of the computational domain (assumed to be a cube), the number of Chebyshev nodes per dimension, the number of sources and target particles, the expansion level, the number of sets of charges, and the desired precision.
For a user-defined kernel, the latter is stored in a class in the main program. There, the user can modify the expression of the kernel and its properties, since the BBFMM3D algorithm can benefit from an eventual lower pre-computation cost depending on the kernel type.
Once are fulfilled these requirements it can be launched the routine according to the provided makefile and perform the computation. The typical output of a default routine is depicted in figure 4.7.

```
Reading pre-compute files ...
Direct calculation starts from: 0 to 0.
Pre-computation time: 0.003486
FMM computing time:   0.067124
FMM total time:   0.07061
Exact computing time: 0.072198
Relative Error: 0.00132761
```

Figure 4.7: Output of BBFMM3D default routine

It must be noticed that the pre-computation phase has to be performed only if the kernel or its parameters change, allowing a faster calculation for multiple potential evaluations based on the same kernel, as could happen in the application of the FMM to the VPM evolution equations.

4.2.2 The exaFMM algorithm

This algorithm is based on the kernel-independent variant of FMM which can be found in the work of Ying et al.[34].
The version that has been used in this work is a modification of the `exafmm-t-master` one, since it is designed to be standard, minimal, and highly optimized, enabling a ready-made and easier way to exploit the parallelizing capacity of modern computers.
Although exists a parallelized version of the BBFMM3D library (PBBFMM3D), it is more difficult to implement it, also considering that this library relies upon a not completely open-source library to execute part of the calculation.

For a more detailed overview of this method, the reader is referred to [34] and [2].

Routine explanation

The library comes with two different routines: one for the laplacian kernel and the other for the Helmholtz kernel (although it is available a modified Helmholtz kernel too).
Despite the BBFMM3D library, exafmm-t stores the kernel and the P2P operations into a header file, which can be eventually modified to be adapted to almost every kind of non-oscillating kernel.
In this way, the main of every different computation is mostly the same, differing only for the correct header call in the "`include`" phase of the code.

The aforementioned main program of the code is therefore structured as follows:

1. Generation or input reading of source and target coordinates, charges, and metadata.
2. Creation of an FMM instance for the desired kernel.
3. Building and balancing of the octree structure.
4. Building the lists and pre-computing invariant matrices.
5. Performing the effective FMM algorithm, through the upward and downward pass.

6. Sorting back the potential values from Hilbert indices to the original order of targets.

The default routines implements also a direct calculation to evaluate the error between FMM and the latter, and also a temporal measurement, as well as the BBFMM3D routine. The typical output of a default routine can be found on figure 4.8.

```
-------------- Time --------------
Build Tree           : 6.3593000e-02
Build Lists          : 5.8010000e-03
Precomputation       : 5.0999000e-01
P2M                  : 1.0633700e-01
M2M                  : 4.1094000e-02
P2L                  : 5.4220000e-03
M2P                  : 5.5180000e-03
P2P                  : 5.5481900e-01
M2L                  : 8.2243000e-02
L2L                  : 4.2484000e-02
L2P                  : 1.1055400e-01
Evaluation           : 9.4921700e-01
------------- Error --------------
Potential Error L2   : 4.8064477e-08
Gradient Error L2    : 2.9084991e-09
-------------- Tree --------------
Root Center x        : 1.9575886e-06
Root Center y        : -4.0656869e-06
Root Center z        : 6.4565588e-06
Root Radius R        : 1.0000075e+00
Tree Depth           : 3
Leaf Nodes           : 512
```

Figure 4.8: Output of exafmm-t default routine

The metadata that has to be set is the order of the expansion P and the maximum number of particles per leaf n_{crit}.
The first one is analogous to the order of a Taylor series expansion and is related to the computational cost of the algorithm as $\mathcal{O}(P^4)$[17].

4.3 Code implementation

Once the theoretical basis and the investigated routines have been discussed, it can be afforded the actual implementation of the two routines on the VPM code.
Firstly, it has been rewritten the evolution equations in a form that is suitable to be solved with a Fast Multipole Method algorithm; their complete derivation is presented in appendix B. Then, it has been implemented its calculation on the

investigated routines, as will be explained in the next lines.
Since the Fast Multipole Method relies upon several operations, as discussed in the first section of this chapter, it starts to be more convenient than the direct calculation for some thousands of particles.
To achieve the best computational time, it has been therefore implemented in the code a flag that, depending on the threshold selected by the user, performs the FMM calculations only for a certain number of particles, that is $N_p > N_{p,threshold}$.
However, the Vortex Particle code is written in Matlab, while the FMM routines rely on C++, so it has been necessary to find a practical but efficient solution to link these two languages.
The idea behind this communication is that the Matlab code stores the required variables by the FMM in text files, then the FMM routine is run directly from Matlab (through a function that allows giving commands to the terminal), and it takes them as input, and gives as output another text file, which is then read by Matlab again, as indicated in the flowchart in figure 4.9.
It has been therefore modified the default routine to read the coordinates of sources and targets, the charges values, and the metadata, and also to give as output the result of the calculation.
It has also been modified the default routine to perform the calculation with the evolution equations kernel discussed in appendix B, as stated before.

Although the file reading can represent a potential bottleneck, it can be seen in figure 4.10 how actually the FMM starts to be faster than direct calculations for about $N_p = 4000$, which represents a satisfactory value for the code, although it can be improved in a future work. This may be due to the usage of the `ifstream` and `ofstream` functions in the C++ routines, which is demonstrated to be the fastest way to perform a file I/O in C++.

The aforementioned scheme to link the Matlab VPM code with the C++ FMM routine has been generalized, allowing a simple and versatile usage of both BBFMM3D and exaFMM, with a few modifications to the functions.
They have been therefore coded as two different, but analogous, Matlab functions that execute the C++ routines; they are subsequently called by the main functions which compute the induced velocity and the circulation strength update according to the evolution equations formulation for FMM discussed in appendix B.

4.4 Conclusions

This chapter has presented a theoretical overview of the Fast Multipole Method, which allows reducing the computational cost of a direct summation from $\mathcal{O}(N_p^2)$

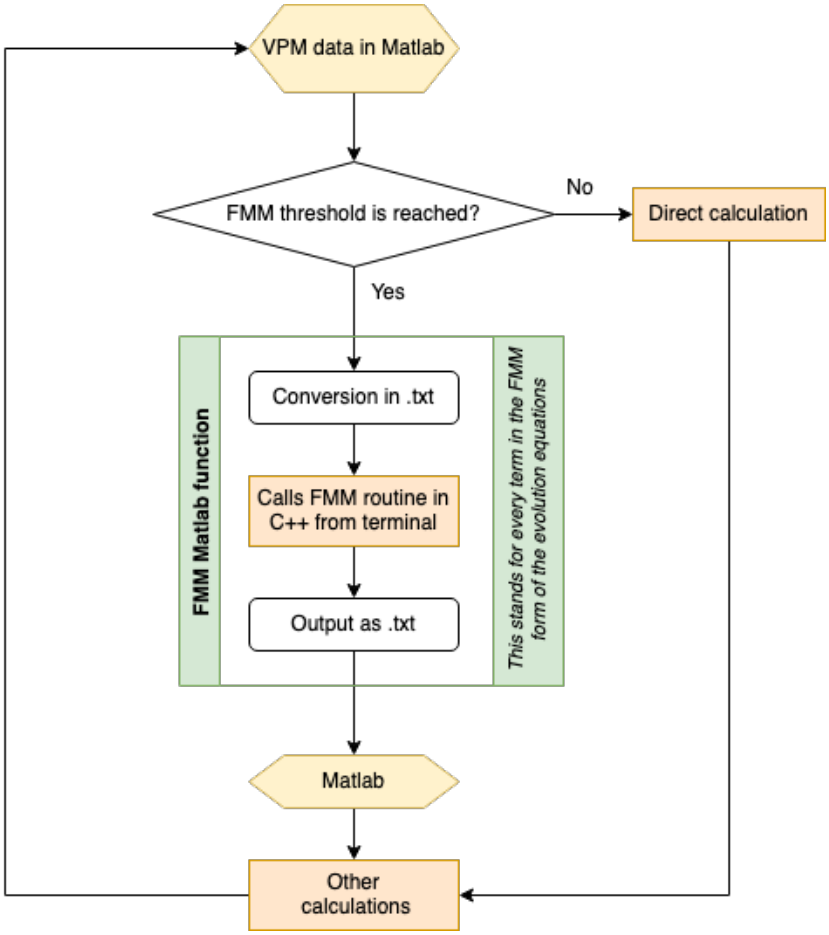


Figure 4.9: Flowchart for FMM integration with Matlab

to $\mathcal{O}(N_p \log N_p)$ or to $\mathcal{O}(N_p)$.
Then, they have been investigated two practical C++ routines that perform a $\mathcal{O}(N_p)$ computation, and how they have been implemented on the Vortex Particle code to compute the induced velocity and the circulation strength update of every particle due to all the particles in the field.

In the next chapter they will be also validated, both alone and into the VPM code, and it will also be performed a comparison study to choose the best one in terms of accuracy and computational speed.

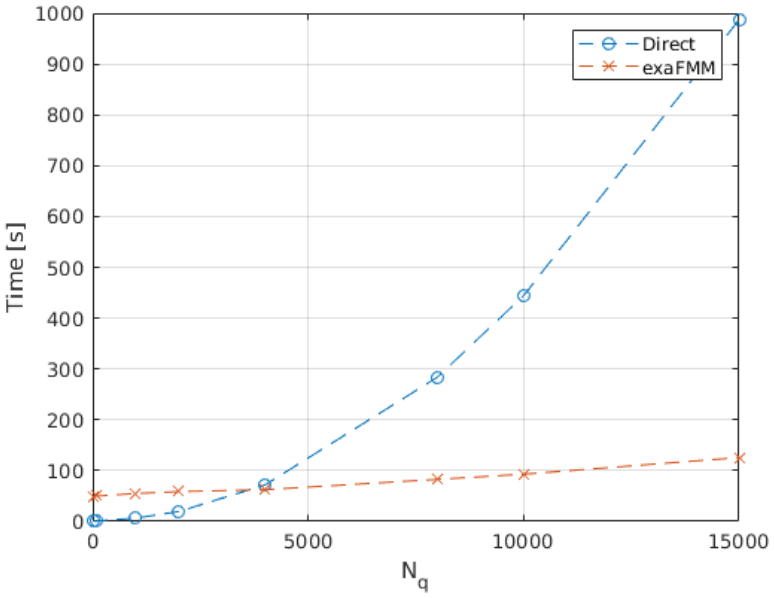


Figure 4.10: Computation time for the VPM routine (Direct vs exaFMM).

Chapter 5

Validation of the code

To ensure the correct behavior of the code, it has been necessary to perform a detailed verification study based on numerical results obtained by several authors[6],[15],[8],[12][4].
They have recreated the simulations performed by these authors and the results obtained in this study have been compared to them.

They have also been performed the verifications and validation studies of both exaFMM[2] and BBFMM3D[3].
In addition, since they have been considered both the FMM algorithms, it has been performed a comparison study between the two routines has been therefore showing the convenience of one instead of the other.

5.1 Validation of the VPM code

The first step has been the validation of the VPM evolution equations discussed in chapter 3.

The first one involves the simulation of a single vortex ring, discretized with one vortex particle as its core, and can be used as a useful tool to investigate the self-induced velocity over the circumference since analytical approximations are available.

The second one involves the simulation of a single vortex ring, discretized with 4 layers of particles as its core, and represents the most used verification tool for a VPM code.
Wincklemans[6] provided a wide choice of numerical results to validate every side of the evolution equations.

Here is also used the diagnostics, discussed in chapter 3, as a further verification tool.

Lastly, the third simulation involves the so-called "leapfrogging" problem, which is useful to verify the correct behavior of vortex stretching.
In addition, this phenomenon can be encountered in the transition phase of the wake to the turbulent breakdown, so it can be useful to further investigate it.

5.1.1 Single vortex ring with one particle on the core

Problem statement

It has been defined a vortex ring placing N vortex particles of core size σ in the $x-y$ plane on a circumference of radius R as shown in figure 5.1.
The initial vorticity $\vec{\omega}$ of every vortex particle has been defined according to Wincklemans[6] as

$$\vec{\omega}(\vec{x},0) = \frac{\Gamma}{2\pi\sigma^2}\vec{e_\theta} \tag{5.1}$$

Γ is the circulation of the vortex ring. The circulation strength $\vec{\alpha}$ can be obtained multiplying $\vec{\omega}$ by the corresponding volume of fluid associated to the particle, which is defined as the volume of a curvilinear cylinder of radius r_l which connects two subsequent vortex particles.

$$vol = \pi R \Delta\theta r_l \tag{5.2}$$

Where r_l is the radius of the circumference of fluid associated with the vortex particle on $x-z$ plane; it is defined as $r_l = \sqrt{2}\sigma$, according to Wincklemans[6], which highlight the correlation with the Hill's spherical vortex.
The angle $\Delta\theta$ is defined as in figure 5.1 and can be computed easily as

$$\Delta\theta = \frac{2\pi}{N} \tag{5.3}$$

Simulation parameters

A simulation with the input parameters shown in table 5.1 has been performed.
The validation has been done over the centroid position at every timestep. An analytic expression for centroid velocity is available for this case (HOA regularization function)[6][22].

$$U_R = \frac{\Gamma}{4\pi R}\left(\log\frac{8R}{\sigma} - \frac{1}{2}\right) \tag{5.4}$$

It has to be noticed that the cutoff radius σ in this expression, as well as the one which has to be used for the regularization function, must be corrected as

$$\sigma = \beta\sigma \tag{5.5}$$

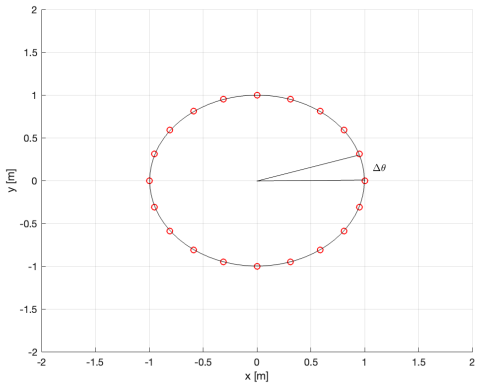


Figure 5.1: Single vortex ring with one particle on the core

Where β is a parameter that is related to the algebraic regularized kernel. Unfortunately, Wincklemans provided a value only for the low-order algebraic kernel, which is $\beta = e^{-0.75}$, so the results might differ because of this difference.
This correction is necessary because, as reported by Wincklemans[6], computing the velocity of a vortex ring with the centerline velocity yields the wrong result for most vorticity distributions. However, if the asymptotically correct velocity of a thin vortex ring of given core size σ_1 is desired, it can be obtained by taking the velocity of the ring as the Biot-Savart velocity applied on the centerline with a core size $\sigma_1 = \beta\sigma_2$.
This occurs because otherwise, it will not be consistent with the expression obtained by Leonard[35] for the thin ring.

N	R	Γ	ν	σ	Δt
200	$1\,m$	$1\,\frac{m^2}{s}$	$2.5 \cdot 10^{-3}\,\frac{m^2}{s}$	$0.1R$	$0.005s$

Table 5.1: Input parameters for the single ring with one particle on the core verification case

Results

It has been performed as a first simulation, where the centroid position over time has been calculated every timestep. The results in figure 5.2 show a good agreement between analytical ones, considering the aforementioned lack in Wincklemans' work about the β value for the HOA regularization function.

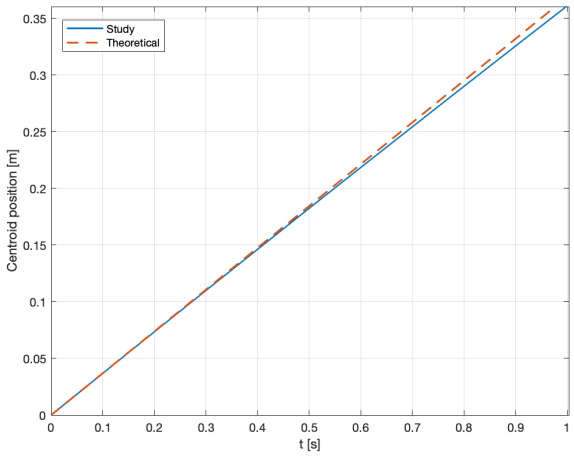


Figure 5.2: Centroid position over time

Then, it has been performed a second simulation, which evaluates the centroid velocity for several values of core size σ, from $0.1R$ to $1.4R$, with a step of $0.2R$. The results can be compared with the analytical expression 5.4 and it can be noticed that they sufficiently agree, as shown in figure 5.3.

It can be concluded that the aspects involved in this test case have been successfully validated.
As previously stated, the differences between theoretical and numerical results may rely upon the cutoff radius correction and also on the definition of r_l, which is not effectively clear from the work of Wincklemans[6].
However, the error involved in these cases is about 1.5%, which can be considered sufficiently small to be addressed to minor issues, also considering that verification studies are not so detailed and clear and do not provide precise instructions on how to recreate the simulation.

5.1.2 Toroidal single ring with multiple particles on the core

Problem statement

Here the ring's core has been discretized with multiple layers, according to Wincklemans work[6]. The ring is therefore composed of N_θ disks, each of them composed of N_ϕ vortex particles, placed in n_c layers.
Notice that in this work the notation regarding angles ϕ and θ is the opposite concerning the Wincklemans' one.

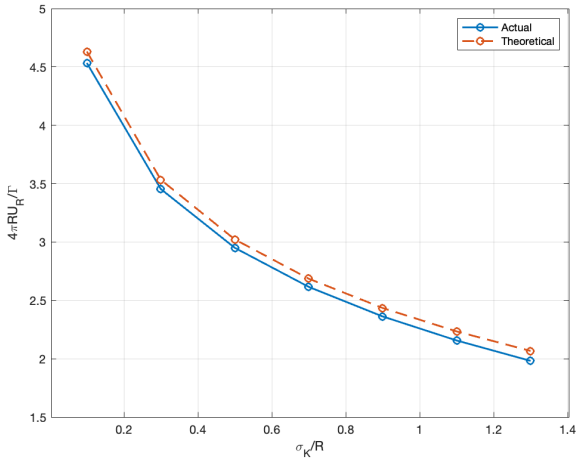


Figure 5.3: Centroid velocity for several values of $\frac{\sigma}{R}$

The generation of a single disk is now discussed, since the entire torus can be further obtained by simply applying a rotation matrix with respect z axis for every value of the angle θ, from $\Delta\theta$ to $N_\theta\Delta\theta$.
According to Wincklemans formulation[6], each particle is placed at the center of a cell. The first particle is referred to a circular cell of radius r_l, while the others are referred to semi-circular crowns of angular width $\Delta\phi = \frac{2\pi}{n}$, where n is the number of particles on the layer, and minimum and maximum radius of, respectively, $r_1 = (2n-1)r_l$, and $r_2 = (2n+1)r_l$, so the maximum radius reached through the discretization is $r_{max} = (2n_c+1)r_l$.
It can be observed that the n-th layer is composed by $8n$ particles, since every layer has 8 more particles than the previous one.

The radial position of every particle can be calculated as

$$r = \frac{1+12n^2}{6n} r_l \tag{5.6}$$

The number of particles in the disk can be therefore computed as $N_\phi = 1+4n_c(n_c+1)$, so the total number of particles in the vortex ring is $N_\theta N_\phi$.

Each cell has an area that can be proven to be

$$A = \pi r_l^2 \tag{5.7}$$

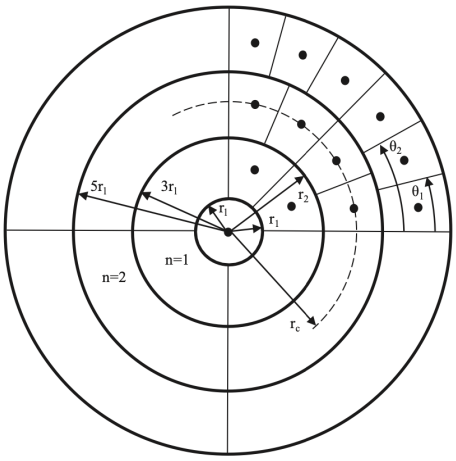


Figure 5.4: Discretization of layers of particles into a 2D disk[6]. The angle indicated as θ is here indicated with ϕ.

So the volume of fluid associated to a cell, i.e a vortex particle, can be computed as

$$vol = \Delta\theta(r_2 - r_1)\left[(\phi_2 - \phi_1)R\left(\frac{r_1 + r_2}{2}\right) + (\sin\phi_2 - \sin\phi_2)\left(\frac{r_1^2 + r_1 r_2 + r_2^2}{3}\right)\right] \tag{5.8}$$

The proof of this expression can be found in the works of Wincklemans[6] and Calabretta[7].

Once the initial disk has been generated, as shown in figure 5.5a, a revolution along z axis is done, as shown in figure 5.5b.

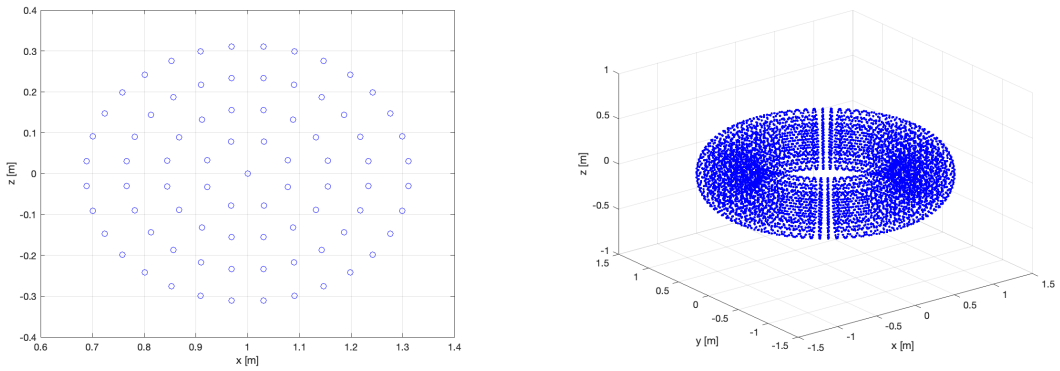
Then, it has been necessary to assign an initial strength to the particles; it can be done, according to Wincklemans' formulation[6], as

$$\vec{\omega}(\vec{x},0) = \frac{\Gamma}{2\pi\sigma^2}\left(1 + \frac{r}{R}\cos\phi\right) e^{-\frac{r^2}{2\sigma^2}}\,\vec{e_\theta} \tag{5.9}$$

It can be noticed that this formulation is a generalization of the previous validation case, as it can be obtained simply by placing $n_c = 0$, further obtaining $\phi = 2\pi$ and $r = 0$.

However, the circulation strength $\vec{\alpha}$ can be obtained simply multiplying $\vec{\omega}$ by the volume associated to every vortex particle.

$$\vec{\alpha}^p(0) = \vec{\omega}(\vec{x}^p,0)vol^p \tag{5.10}$$



(a) Initial disk **(b)** Complete vortex ring

Figure 5.5

With the above initial conditions, the initial enstrophy can be exactly evaluated, as reported in the work of Wincklemans[6], integrating the square of $\vec{\omega}(\vec{x},0)$.

$$\varepsilon(0) = \int \vec{\omega}(\vec{x},0) \cdot \vec{\omega}(\vec{x},0)\, d\vec{x} = \frac{\Gamma^2 R}{2\sigma^2}\left(1 + \frac{3\sigma^2}{2R^2}\right) \tag{5.11}$$

This value can be used to verify the correct kinetic energy initial decay, since it can be calculated as

$$\frac{dE(0)}{dt} = -\nu\varepsilon(0) \tag{5.12}$$

Therefore, it can be performed a simulation of three time-steps to calculate $\frac{dE}{dt}$ with a second order finite difference scheme.

$$\frac{dE(0)}{dt} = \frac{-E(2\Delta t) + 4E(\Delta t) - 3E(0)}{2\Delta t} \tag{5.13}$$

This can be also used as a further way to adjust the core size σ, if the values of $\frac{dE(0)}{dt}$ and $-\nu\varepsilon(0)$ does not agree.

Simulation parameters

It has been performed a simulation with the parameters shown in table 5.2. The number of layers n_c has been chosen as a compromise between the necessity of recreating one of the cases in the work of Wincklemans[6] and memory constraints. The parameters are equal to the ones used by Wincklemans in the $n_c = 4$ case.

N_θ	n_c	R	r_{max}	Γ	σ	ν	Δt
80	4	$1m$	$0.35m$	$1\frac{m^2}{s}$	$0.1R$	$2.5 \cdot 10^{-3}\frac{m^2}{s}$	$0.025s$

Table 5.2: Input parameters for the toroidal single ring case

Results

The results obtained have been reported in table 5.3, providing a comparison between values obtained in this verification case and values obtained by several authors[6][7][15][12] in their analogous validation phase of the VPM code.

Notice that the values of kinetic energy and enstrophy for the divergence-free

	Study	***Wincklemans***	***Calabretta***	***Singh***	***Martin***
$\|I(0)\|$	3.2446	3.2139	3.1326	3.1654	3.274
$E(0)$	1.0548	1.0475	1.0128	1.0167	1.0407
$E_f(0)$	1.0549	1.0476	1.0129	1.0168	1.0408
$\varepsilon(0)$	61.2560	61.346	N/A	61.411	N/A
$\varepsilon_f(0)$	60.2560	62.3842	63.3821	60.412	63.52
$\frac{dE(0)}{dt}$	-0.1455	-0.1276	-0.1276	-0.1496	N/A

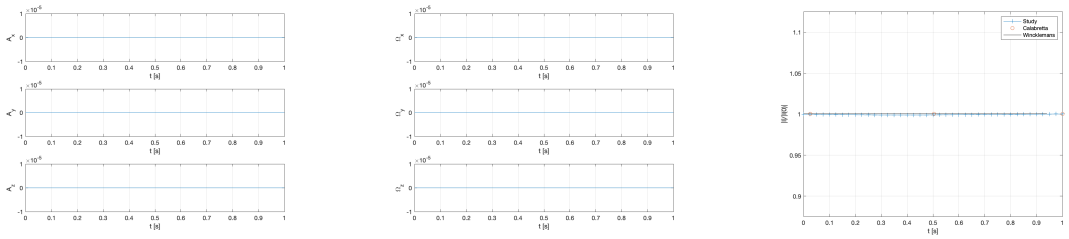
Table 5.3: Diagnostics comparison of the single toroidal vortex ring with several authors[6][7][15][12].

assumption are considerably close to the more general ones, without the divergence-free assumption. This is a clear indication that the implemented transposed scheme ensures a sufficiently acceptable divergence-free condition of the velocity field.
The initial enstrophy value is greater than the theoretical one but, as highlighted by Wincklemans, this difference becomes lower as the number of layers n_c is increased.

It can be observed that the values obtained are consistently slightly different from the ones obtained by the aforementioned authors. These differences may rely upon several issues:

- Almost every verification study refers to Wincklemans' one, but in his work, some aspects are not clearly explained, such as the possibility of a core size correction after the computation of the initial kinetic energy decay, the programming language used, and the iterations used in Beale's method for initial condition relaxation.
- Singh and Martin do not explicitly mention the usage of Beale's method for particle relaxation.

However, the error between this verification study and the others, especially Wincklemans' one, can be considered acceptable and so the validation case was successfully achieved.



(a) Angular impulse evolution

(b) Total vorticity evolution

(c) Normalized linear impulse evolution

Figure 5.6: Linear diagnostics evolution

In addition to these initial values, it has been reported in figure 5.6 the evolution of linear diagnostics, in terms of angular impulse $\vec{A}$ (figure 5.6a), total vorticity $\vec{\Omega}$ (figure 5.6b) and normalized (with respect its initial value) linear impulse $\frac{|\vec{I}|}{|\vec{I}(0)|}$ (figure 5.6c).
It can be observed that both angular impulse and total vorticity are conserved (as the transpose scheme, correctly implemented, ensure), while the linear impulse evolves as obtained by both Wincklemans and Calabretta.

In conclusion, the high amount of agreement for each investigated term, both in initial values and throughout their evolution is a clear indication that the evolution equations for velocity and stretching, as well as the time evolution scheme employed, have been correctly implemented.

5.1.3 Leapfrogging of two single rings with one particle on the core

Problem statement

They have been defined as two vortex rings that are both conceptually identical to the one discussed in the previous test case; they are placed concentrically, as shown in figure 5.7.
Since the core size has been considered constant, then the volume is not conserved, and it varies because of the radius variation every time step.

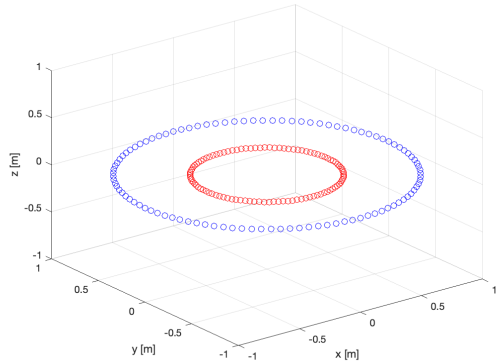


Figure 5.7: Leapfrogging rings with one particle on the core

Simulation parameters

The simulation is performed with the parameters reported in table 5.4, for an inviscid case.
Since the smaller ring has a greater induced velocity than the bigger one, it tends to rapidly expand, becoming bigger than the other.
Then, it begins to slow down, while the other ring tends now to expand and pass through the other one, repeating this until the two rings merge and diffuse.
The expansion or contraction occurs because of the vortex stretching term, so it can be expected that a correct behavior of vortex stretching causes a cyclically varying radius of the two rings.

N_1	N_2	R_1	R_2	Γ_1	Γ_2	σ_1	σ_2
100	100	$0.5\,m$	$1\,m$	$1\,\frac{m^2}{s}$	$1\,\frac{m^2}{s}$	$0.1R$	$0.1R$

Table 5.4: Input parameters for the leapfrogging verification case

Results

The results obtained with the present simulation can be compared with the available analytical expression, derived by Konstantinov[23] and also reported in the work of Berdowski[4].
As shown in figure 5.8, they agree very close so it can be concluded that the validation has been successfully done.

The vortex stretching term is not directly validated, but a correct evolution of this

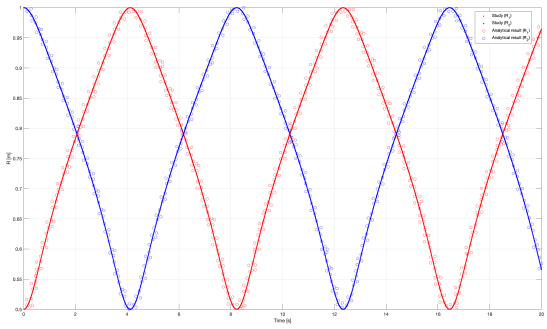


Figure 5.8: Evolution of ring radius over time for the two leapfrogging vortex rings

problem can occur only when the stretching is correctly computed, so it can be considered as implicit proof of its correctness on the code.

5.2 Validation of the BBFMM3D algorithm

After validating the direct VPM routine, it has been mandatory to validate the FMM implemented algorithm, to ensure its correct behavior on the evolution equations.

Firstly, it has been performed a study of the C++ routine with the default Laplacian kernel $K = \frac{1}{r}$ with random sources, targets, and charges values. This allowed verification of the routine itself, but also of the Matlab coupling process.
In addition, it has been possible to investigate the effective time-saving feature of this FMM algorithm and the correlation between the levels of the multipole expansion and the computational cell length, since Seetharaman[8] reported in his work such correlation, like the one that results in an evident decreasing of the root mean square (RMS) error concerning the direct calculation.

Then, the evolution equations have been rewritten in a suitable form for the algorithm, as the product of the kernel $K(x, y)$ with the source weight q. Here, x is the source and y is the target.
After that, it has been performed a verification study for the direct calculation (validated in the previous section) and the required time for both direct and accelerated calculation has been investigated.

5.2.1 Laplacian kernel

Firstly, it has been investigated the relative error trend, as several parameters are changed; has made it possible to individuate the best parameters for the subsequent simulations, to both ensure an acceptable error and a considerable time saving to the direct calculation.
For these purposes, it has been chosen to use the default laplacian kernel $\frac{1}{r}$, provided as an example in the BBFMM3D routine.

It has been slightly modified the C++ routine to give as output both sources and targets coordinates, charges values, and potential ϕ values too. Then, it has been performed the direct calculation in Matlab and then it has been computed the relative error between FMM and direct calculation for every particle; this has been done for $N_p = 10$, $N_p = 100$, $N_p = 1000$, and $N_p = 5000$. The resulting plots can be seen in figure 5.9 and it can be observed that the error decreases as the number of particles is increased.
Together with this error calculation, it has been measured the elapsed time for both FMM and direct calculations and it can be seen in figure 5.10a that for about $N_p = 2000$ the FMM algorithm starts to be faster than the direct calculation. It must be noticed that the FMM time here represented is considered also with the generation of pre-computation files by BBFMM3D. This is necessary only if the kernel changes, so if the implemented routine will have several calculations involving the same kernel, then the elapsed time will be considerably lower.
It has been performed a calculation where only the first time are generated pre-computation files and, as it can be observed in figure 5.10b, the FMM (with laplacian kernel) starts to be faster mostly immediately.

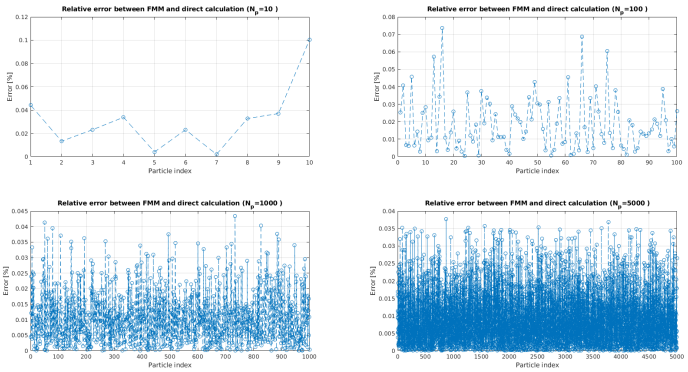
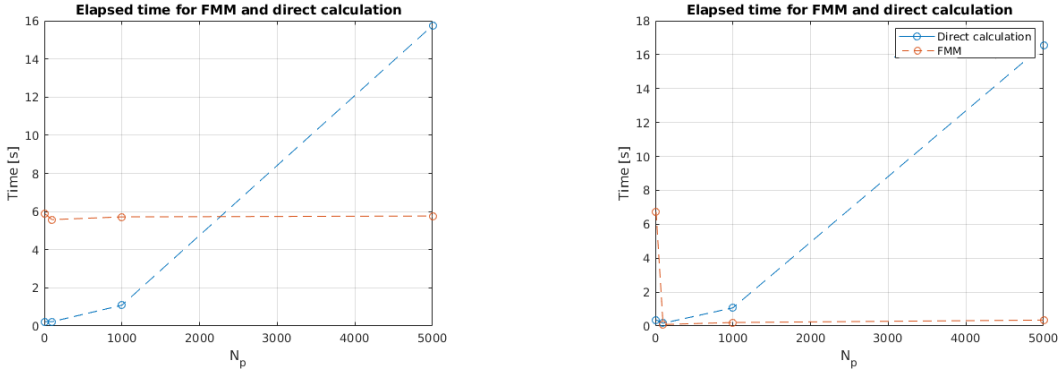


Figure 5.9: Relative error between FMM and direct calculation (in Matlab) for increasing number of particles. $L = 1$, $level = 3$

After having checked the consistency of the algorithm, it has been performed the calculation of the so-called root mean square error (RMS).

$$RMS = \sqrt{\frac{\sum(\phi_{FMM} - \phi_{dir})^2}{\sum \phi_{dir}^2}} \tag{5.14}$$

And they have been performed several simulations, where they have been variated several parameters of the algorithm, to have a complete overview of the RMS error trend with this variations.



(a) Precomputation every step **(b)** Precomputation only on first step

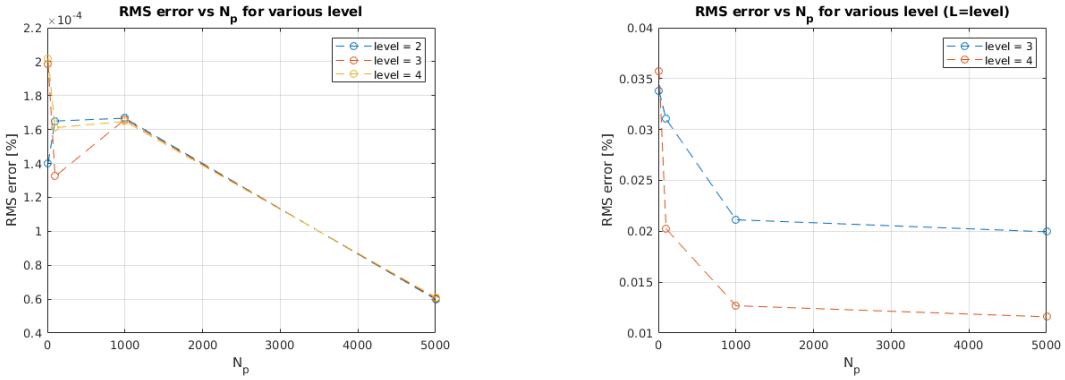
Figure 5.10: Elapsed time for FMM and direct calculation (in Matlab) for increasing number of particles. $L = 1$, $level = 3$

The parameters that can be changed in the simulation are:

- The length of the computational cell (assumed to be a cube)
- The number of levels of the multipole expansion

It can be also changed other parameters, such as the precision ϵ value, but it has to be decided to keep this value to its default one (10^{-9}).

Therefore, they have performed calculations of RMS error between FMM and direct calculation for several values of L, $level$, and, of course, N_p.
In figure 5.11a is shown the RMS error trend with N_p, for various $level$ numbers. Here L is fixed to $L = 1$. It can be observed that the algorithm, for $N_p > 1000$, tends to be not so sensible to the level number, as it is instead for a low value of N_p.
In figure 5.11b instead is shown the same plot, but where the L value has been fixed to $L = level$; it can be seen that the RMS error decreases as $level$ increases



(a) $L = 1$ **(b)** $L = level$

Figure 5.11: RMS error between FMM and direct calculation (in Matlab) for increasing number of particles and various *level* values

and, of course, as N_p increases.

Finally, in figure 5.12 can be observed the most important result: there is a

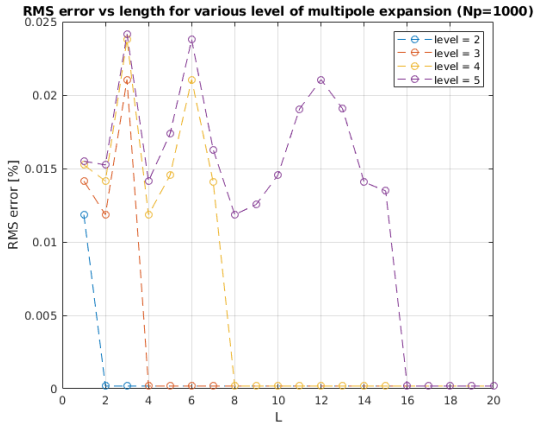


Figure 5.12: RMS error between FMM and direct calculation (in Matlab) for increasing L and various *level* values. $N_p = 1000$

correlation between *level* and the lowest L value that ensures in the lowest RMS error, and, as pointed by Seetharaman[8] in his work, it is

$$L = 2^{level-1} \tag{5.15}$$

This may allow the automatic selection of L or *level*, given the other parameter, to have the lowest RMS with the highest L or *level*, ensuring therefore both accuracy

and time saving by this FMM algorithm.

Lastly, it has been plotted in figure 5.13 the elapsed time by FMM and direct calculation for several values of L; it can be observed always a considerably time saving using FMM of about one minute.

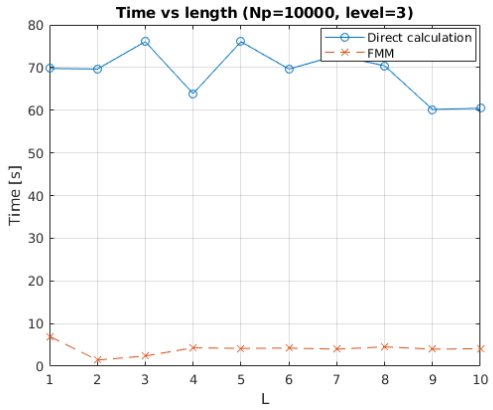


Figure 5.13: Elapsed time for FMM and direct calculation (in Matlab) for increasing L. $N_p = 10000$, $level = 3$

5.2.2 VPM evolution equations' kernels

Now that they have investigated both the validity and the performances of the BBFMM3D[3] algorithm for a standard case, it is necessary to perform a verification of the modified routine.
The already modified routine for the laplacian kernel has been further modified and repeated for all the kernels necessary for the calculation of induced velocity and circulation strength update, according to evolution equations.
Since the FMM can only evaluate the result of a computation such

$$\phi_j = \sum_{i=1}^{N} K(x, y) q_i \tag{5.16}$$

It has been necessary to rewrite the evolution equations 3.45, 3.46 in an analogous form. The complete derivation of the FMM form of the evolution equations is provided in appendix B.

After having done so, it has been performed a first check on the correctness of the kernels, evaluating the RMS error with respect the direct Matlab calculation for every kernel, with a random set of coordinates and charges; the RMS error for

all the kernels is shown in figure 5.14 and it can be appreciated its negligibility, probably mostly due to numerical precision differences between C++ and Matlab.

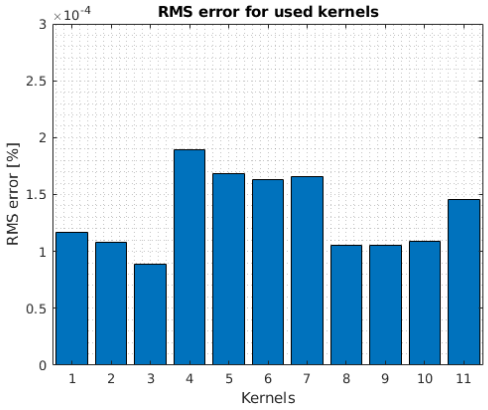
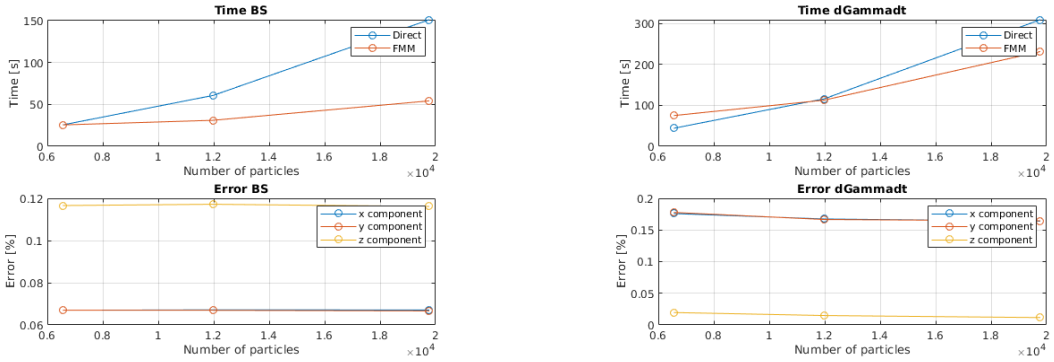


Figure 5.14: RMS error between FMM and direct calculation (in Matlab) for evolution equations' kernels. $N_p = 1000$, $L = 2^{level-1}$, $\sigma = 0.1\,m$, $\nu = 0.0025\frac{m^2}{s}$

Then, they have been implemented two Matlab functions that evaluate induced velocity and circulation strength update from the kernel formulation previously mentioned and they have been performed several test simulations for increasing N_p, where they have been computed $\vec{u}^p$ and $\frac{d\vec{\alpha}^p}{dt}$ both with FMM and direct Matlab calculation and, then, the RMS error, and the elapsed time.
In figures 5.15a and 5.15b are shown the results of these simulations.



(a) Induced velocity (Biot - Savart law) **(b)** Circulation strength update

Figure 5.15: RMS error between FMM and direct calculation (in Matlab) for evolution equations. $L = 2^{level-1}$, $\sigma = 0.1\,m$,$\nu = 0.0025\frac{m^2}{s}$.

As it can be seen, there is a consistently high time saving for both the routines, approximately for $N_p = 10000$. The induced velocity routine is faster since its equations involve a lower number of terms, while the circulation strength update one involves the computation of the dot product of a further cross product, and involves several different kernels in the computation of every component.
However, the error seems to be sufficiently low and the BBFMM3D[3] algorithm can be therefore considered successfully validated and implemented in the code.

5.3 Validation of the exaFMM algorithm

Here is presented an analogous verification study of exaFMM routine[2].
It has been firstly performed a verification based on the default Laplacian kernel $K = \frac{1}{r}$, to ensure the correctness of the routine and the linking phase in the Matlab code, then it has been performed validation of all the evolution equations' kernels.

In addition, several features have been investigated, such as the computational time and the RMS error trend.
These investigations have been performed also in comparison to the ones done with the BBFMM3D[3] routine. This has allowed the choice of one routine instead of the other, as it will be shown in the next paragraphs.

5.3.1 Laplacian kernel

An analogous verification study on this FMM algorithm has been performed, evaluating the relative error for each particle and the elapsed time, for increasing N_p, with respect to the direct Matlab calculation.
In figure 5.16 it can be observed a similar relative error trend, with respect to the

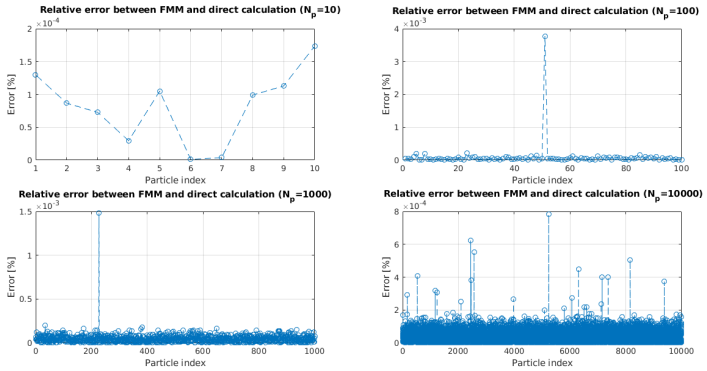
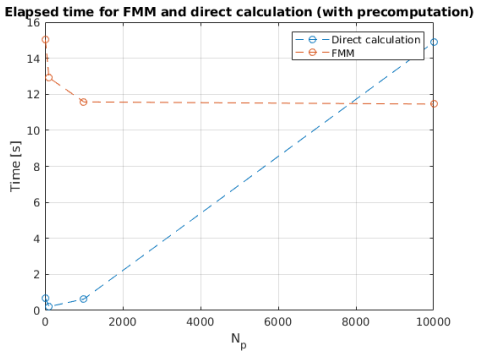
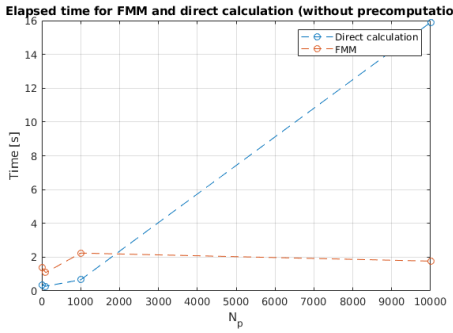


Figure 5.16: Relative error between FMM and direct calculation for increasing number of particles. $P = 8$, $n_{crit} = 400$

BBFMM3D algorithm, while in figures 5.17a and 5.17b are shown the elapsed time with and without pre-computation (for every step).
As it can be seen, the pre-computation step takes a considerable quantity of time, even more than BBFMM3D, but it has to be considered that this pre-computation is required more often in BBFMM3D than in exaFMM and, also, that the two algorithms have conceptually different kind of inputs; while the BBFMM3D requires the cut-off length from the far-field and near field, and the number of levels of the expansion, the exaFMM requires the cut-off number of particles from the far-field and near field, and the order of the expansion, so this difference may rely upon the aforementioned differences.



(a) With precomputation every step



(b) Without any precomputation

Figure 5.17: Elapsed time for FMM and direct calculation for increasing number of particles. $P = 8$, $n_{crit} = 400$

After these two main verifications, they have been performed further simulations, varying N_p, P (expansion order) and n_{crit} (cut-off number of particles).

In figure 5.18 one can observe the RMS error for increasing N_p, for various P. It can be seen how, from $P = 8$, the error does not seems to evidently change.

In figure 5.19 is depicted the RMS error for increasing n_{crit} and various P; here, again, it can be seen that from $P = 8$ the RMS error does not appear to change.

Lastly, in figure 5.20 is shown the elapsed time with the same parameters as the previous figure. It can be clearly seen how the n_{crit} represents such a cut-off number of particles, and also how the expansion order gives more accuracy but, naturally, also requires more computation time.

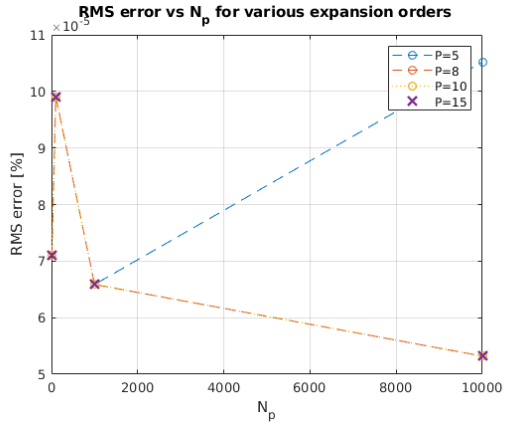


Figure 5.18: RMS error between FMM and direct calculation for increasing number of particles and various P values. $n_{crit} = 400$

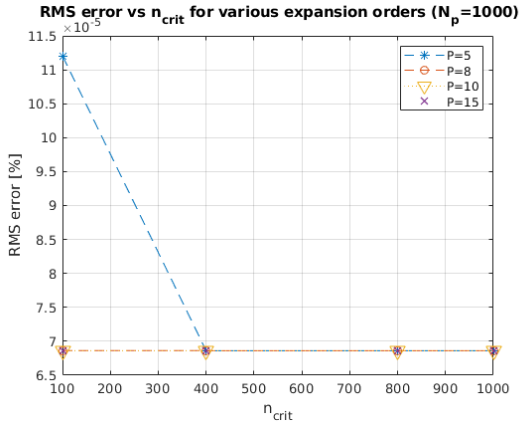


Figure 5.19: RMS error between FMM and direct calculation for increasing n_{crit} and various P values. $N_p = 1000$

5.3.2 VPM evolution equations' kernels

They have been performed the same verifications as the ones done for BBFMM3D and in figure 5.21 it can be observed the RMS error with respect the Matlab direct calculation for every evolution equations' kernel. As can be seen, the error is slightly lower than the BBFMM3D routine, but this may also be due to the conceptually different inputs that the two routines must have.

It can be evinced that the n_{crit} number has been set to a fraction of N_p (here is 25%) since it has been noticed that a lower value produces a significantly higher and unacceptable error. This may be because n_{crit} should not be so different from N_p since it is a cut-off variable from far-field to near-field (where the direct calculation is used).

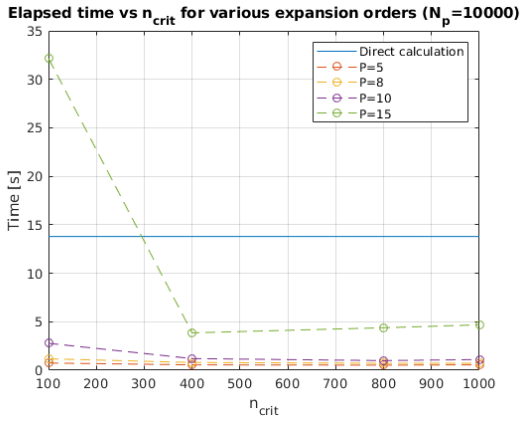


Figure 5.20: Elapsed time for FMM and direct calculation for increasing n_{crit} and various P values (no precomputation)

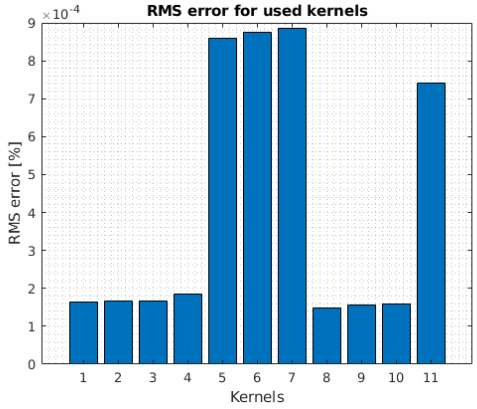
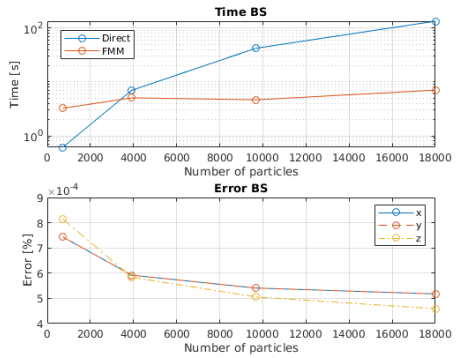
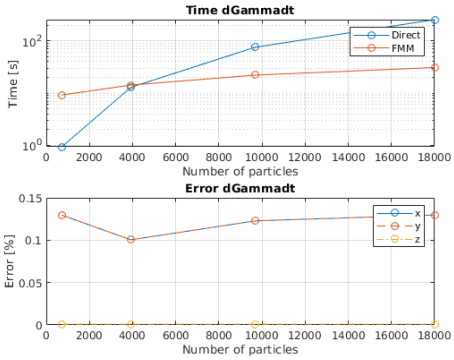


Figure 5.21: RMS error between FMM and direct calculation for evolution equations' kernels. $N_p = 1000$, $P = 8$, $n_{crit} = \frac{N_p}{4}$, $\sigma = 0.1\,m$, $\nu = 0.0025\frac{m^2}{s}$

Finally, in figures 5.22a and 5.22b are shown the RMS errors and elapsed times for Biot - Savart and circulation strength update calculations. As it can be seen, the computation time is much lower than direct calculation and BBFMM3D ones, as the error for induced velocity, while the one for circulation strength update is essentially the same concerning BBFMM3D. Further considerations regarding this verification study are the same as the ones done for BBFMM3D.



(a) Induced velocity (Biot - Savart law)

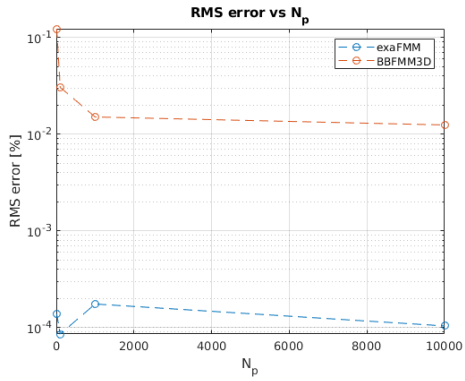


(b) Circulation strength update

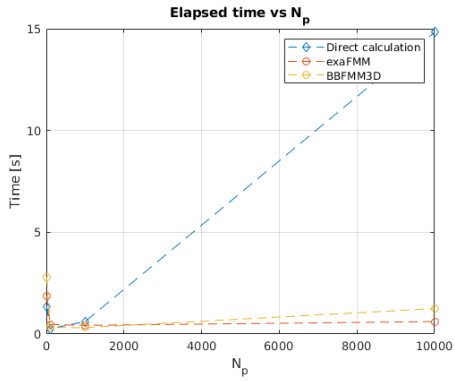
Figure 5.22: RMS error between FMM and direct calculation for evolution equations. $P = 8$, $n_{crit} = \frac{N_p}{4}$, $\sigma = 0.1\,m$,$\nu = 0.0025\frac{m^2}{s}$.

5.4 Comparison between BBFMM3D and exaFMM

Once the two routines have been validated and it has been validated also its integration into the Matlab VPM code, it can be interesting to perform a comparison between them, both in terms of RMS error and elapsed time for the Matlab direct calculation.
Figures 5.23a and 5.23b show the aforementioned results and it can be seen that exaFMM appears to be faster and even more accurate to BBFMM3D.



(a) BBFMM3D. $L = 2^{level-1}$, $level = 8$



(b) exaFMM. $n_{crit} = 400$, $P = 8$

Figure 5.23: Comparison between BBFMM3D[3] and exaFMM[2]. Laplacian kernel.

The temporal difference can be addressed to the parallelization feature that only exaFMM has, while the accuracy difference may be instead addressed to the actual difficulty to set two simulations with the same parameters, since, as previously remarked, the two algorithms have conceptually different inputs and perform slightly different kind of calculations.
However, the FMM procedures are mostly equivalent, since they can be all addressed to what is explained and discussed in chapter 4.

It is important to notice also that the pre-computation operation on the BBFMM3D routine must take place every time that the calculation involves a different kernel from the previous one, while the exaFMM routine requires a new pre-computation only if the kernel's parameters are changed concerning the previous code run, since the first algorithm has a single pre-computation file, while the second one has a pre-computation file for each kernel. This means that the exaFMM routine can guarantee a further time saving since the pre-computation can occur only on the first time step and can be skipped if a simulation is run not changing the core size σ and the kinematic viscosity ν.

5.5 Conclusions

Here, both VPM code and FMM algorithm have been validated, successfully comparing results with ones obtained in standard test cases by several authors (VPM) and with direct calculations of previously validated schemes (FMM).
All the differences between the verifications and the comparison cases result in negligible errors, that may rely upon minor differences, not however clearly discussed in the investigated works.

In addition, it has been explicitly shown the advantage, both from computational and software points of view, of using the exaFMM algorithm[2] instead of BBFMM3D one[3].
The BBFMM3D algorithm appears to be more intuitive and easily adaptable to every kind of kernel, but its two main drawbacks are that it is not parallelized and that it relies upon deprecated libraries, such as the version 2.1.5 of the FFTW library, which is not available for Windows.
In the intention of developing a fast, intuitive, and wide usable application, this has been one of the major points that influenced the choice of the exaFMM algorithm, also considering the parallelization feature of the latter, which is not a secondary point since the entire code has the aim of representing a fast but accurate numerical tool.

It has to be noticed, however, that a parallelized version of BBFMM3D actually exists, and it uses also more recent libraries, such as the FFTW3. This would considerably improve the elapsed time, but it is not guaranteed that it will be improved the accuracy.
In addition, the aforementioned routine uses an Intel library (Intel MKL) that is not completely open-source; on the intention of a completely open-source code, this has also been a discriminating factor for the choice of exaFMM.
However, as it will be indicated in the conclusion chapter 7, it can be an interesting point to perform a further comparison between exaFMM and PBBFMM3D (the parallelized version of BBFMM3D).

Chapter 6

Results

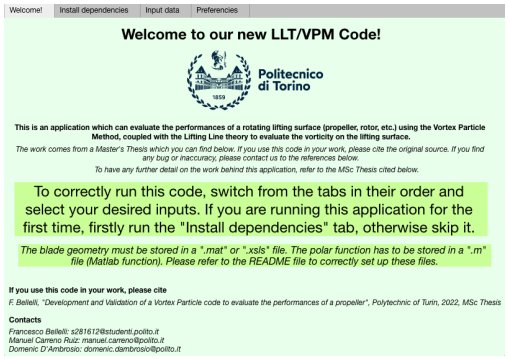
The present chapter aims to present the main results of the entire thesis work, as a high-level benchmark for future works both from the author and from the entire scientific community.
They will be presented the results of two different simulations on a T-Motor $15 \times 5''$ carbon fiber blade for which are available experimental results, CFD results, and LLT results.

The VPM code discussed in the previous chapters has been additionally translated into a stand-alone application, with a practical GUI (Graphical User Interface) which is reported in figure 6.1.
As it can be seen, the application presents several tabs, where the user can easily learn how to correctly set up the input files (the blade geometry and the polar curve), where it can be configured the eventual Windows interface for the exafmm-t code (which is written to be much more suitable for Unix systems), install the exafmm-t library itself and where they can be chosen all the parameters for the desired simulation.
The application, once run, provides a wake plot every time step and, when the simulation ends, is also able to plot the lift coefficient, the thrust per unit length, and the temporal trend of both thrust and torque.

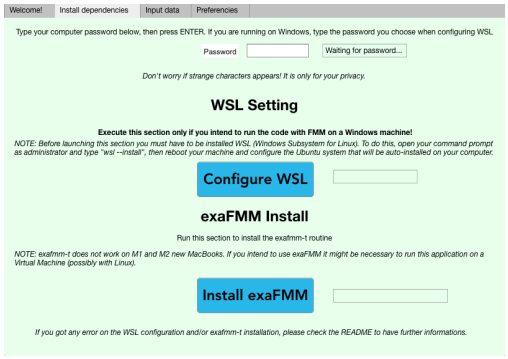
6.1 Validating framework

In this section, it will be explained more in detail the genesis of the data that will be used as a validation for the entire VPM code.

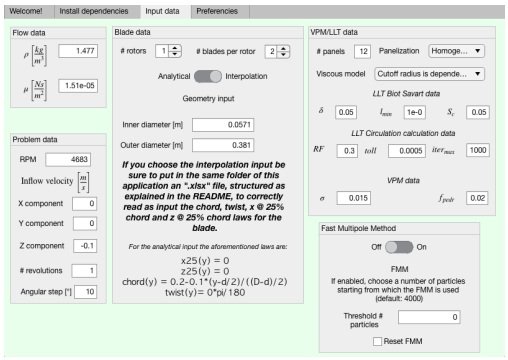
As previously mentioned, the simulations have been performed on a T-Motor $15 \times 5''$ (figure 6.2) carbon fiber blade which has been translated to a CAD model



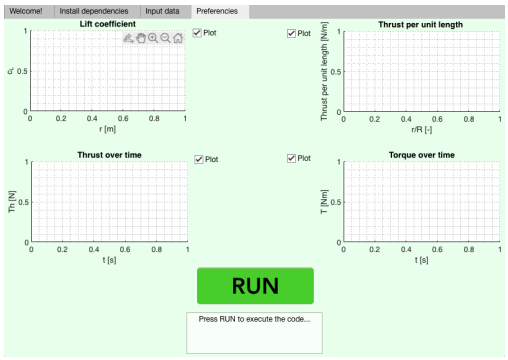
(a) Welcome tab



(b) Install dependencies tab



(c) Input data tab



(d) Run and plot results tab

Figure 6.1: Graphical User Interface of the VPM code

using an optical precision measuring machine (OPMM).
To correctly build the aerodynamic database they have performed simulations on several airfoil stations, which included a turbulent transition model above $Re = 15{,}000$ that is a combination of a $k - \omega$ one[36] with the $\gamma - Re_\theta$ transition model[37].
For further details on this work, the reader is referred to [38], [39].
The experimental results are taken from[40], while the CFD results are taken from[41].

6.2 VPM simulation parameters

In this section, it will be discussed the code setup that has been used for the performed simulations.

The two simulations have been executed with two different types of panelization: homogeneous and cosine law with tip thickening.

Figure 6.2: Top and front views comparison between original blade (left) and reconstructed CAD model (right)

The second one would have the aim of capturing more precisely the strong circulation variations on the blade tip.

Excepting for this difference on the panelization type, both the simulations have been performed with the same parameters, which are summarized in table 6.1. The

Parameter	σ	$\Delta\alpha$	δ	l_{min}	S_c	f_{pedr}	V_{inflow}
Value	$0.015\,m$	$10°$	0.05	$1 \cdot 10^{-6}$	0.05	$0.02,\ Hz$	$-0.1\,\frac{m}{s}$

Table 6.1: Parameters of the two simulations

simulations have been performed with the FMM coupling that activates starting from $N_p = 4000$.
This value has been obtained by comparing the elapsed time with direct calculation and with the exafmm-t library, as can be observed in figure 6.3.
The viscous model used for the evaluation of the induction due to the panels in which the blade has been discretized is a constant cutoff radius.

6.3 Simulations results

In this section are presented the results for the two aforementioned simulations.
The results will be shown in terms of lift coefficient, thrust per unit length, thrust and torque value, and wake plot.
As well as the experimental and CFD results, they have performed 10 revolutions of the blade.

6.3.1 Homogeneous panelization

In figure 6.4 is shown the lift coefficient distribution on the two blades, while in figure 6.5 it can be observed the thrust and torque trend over the simulation time

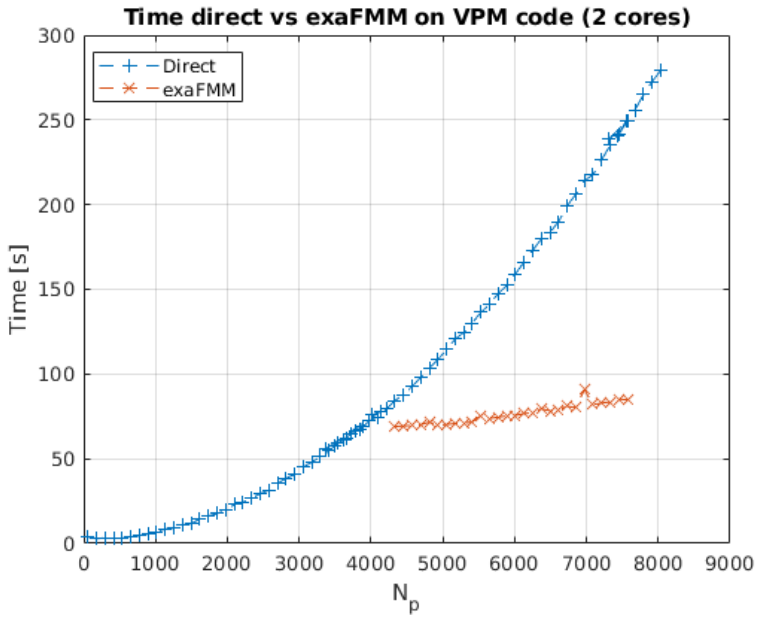


Figure 6.3: Elapsed time for direct calculation and exafmm-t for the VPM temporal loop

of 10 revolutions.
The final values of thrust and torque are $Th = 13.67\,N$ and $T = 0.259\,Nm$.
In figure 6.6 is finally depicted the thrust per unit length.
The final wake in terms of vortex particles can be seen in figure 6.7.
The simulation ended up with a total amount of 23,089 particles.

It can be performed a comparison with the CFD results, which is shown in figure 6.8.
As it can be seen, the results only slightly differ from each other, and the thrust is well resolved both for root, central, and tip stations.

6.3.2 Cosine law with tip thickening panelization

In figure 6.9 is shown the lift coefficient distribution on the two blades, while in figure 6.10 it can be observed the thrust and torque trend over the simulation time of 10 revolutions.
The final values of thrust and torque are $Th = 12.94\,N$ and $T = 0.249\,Nm$.
In figure 6.11 is finally depicted the thrust per unit length.
The final wake in terms of vortex particles can be seen in figure 6.12.
The simulation ended up with a total amount of 32,086 particles.

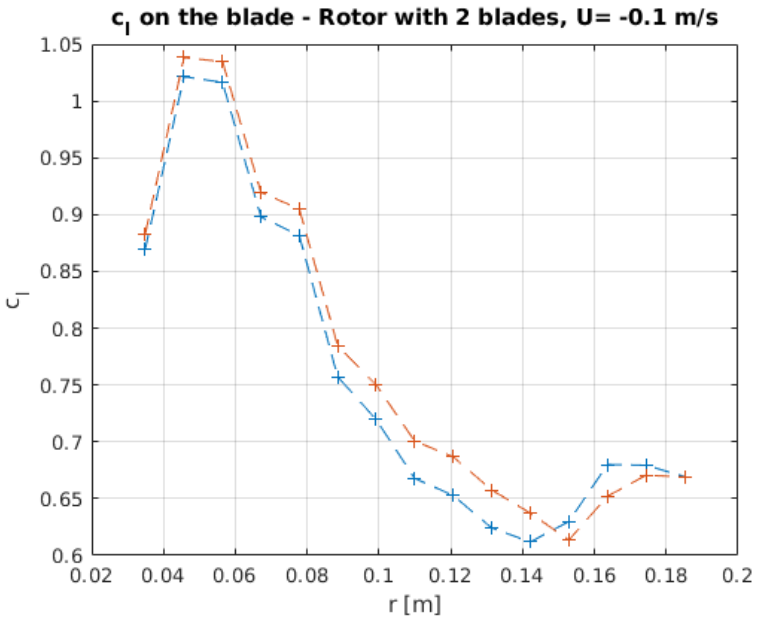


Figure 6.4: Lift coefficient for homogeneous panelization

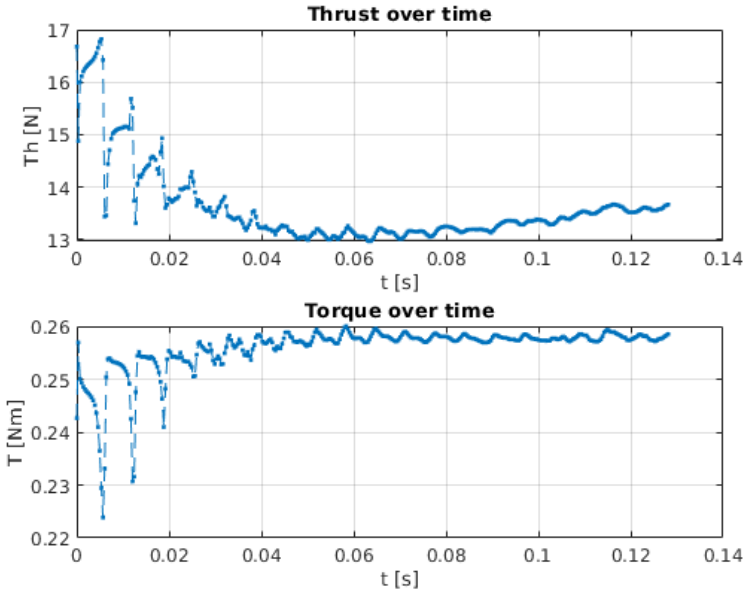


Figure 6.5: Thrust and torque temporal trend for homogeneous panelization

It can be performed a comparison with the CFD results, which is shown in figure

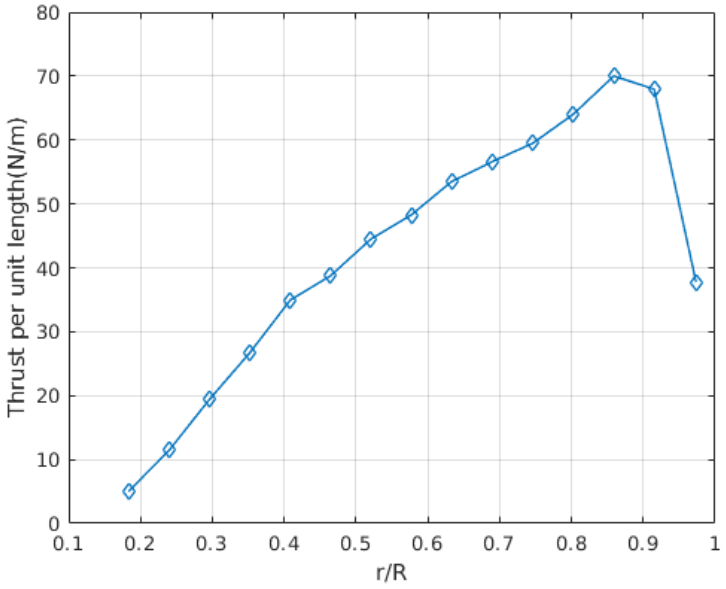
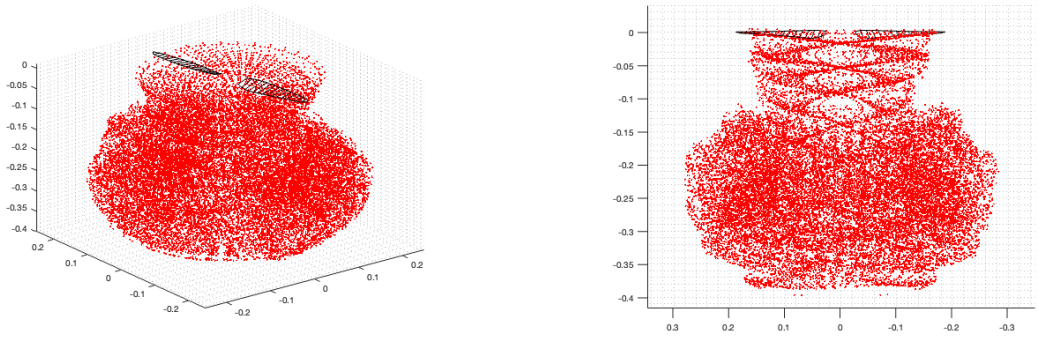


Figure 6.6: Thrust per unit length for homogeneous panelization



(a) Isometric view **(b)** Frontal view

Figure 6.7: Final wake plot for homogeneous panelization

6.13.
Here the tip seems to be not so well resolved, to the homogeneous panelization, as it can be also seen in figure 6.14.
This might be addressed to the fact that, although the cosine law allows capturing more accurately the circulation variations on the tip, it forces to generate a higher amount of particles at the tip rather than at the root/central stations.
Since the thrust relies on how much induction there is nearby, and since the more

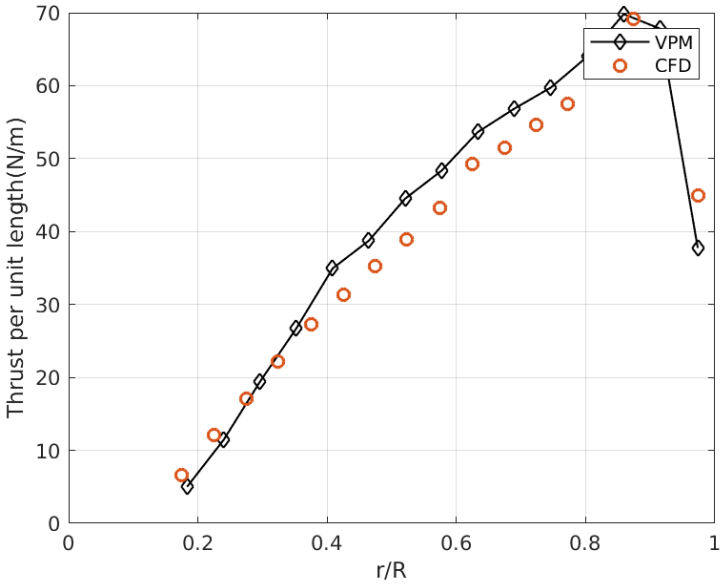


Figure 6.8: Thrust per unit length comparison vs CFD for homogeneous panelization

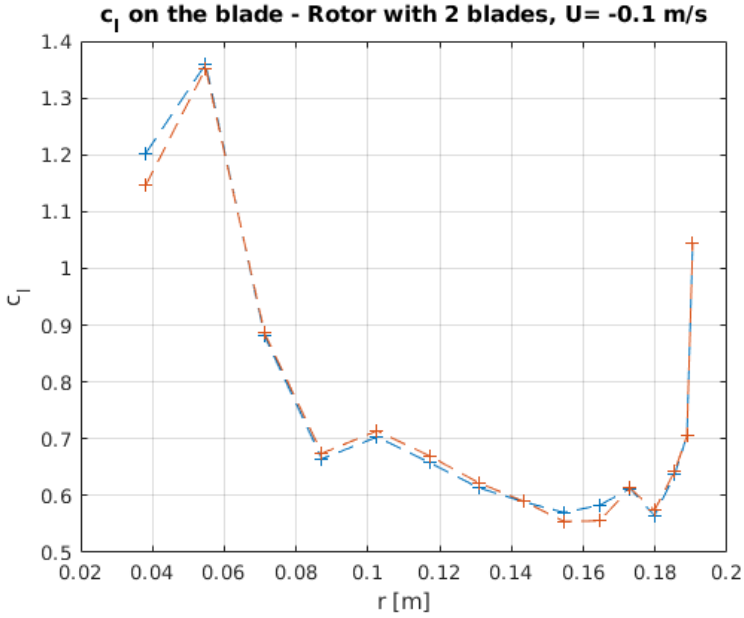


Figure 6.9: Lift coefficient for cosine law with tip thickening panelization

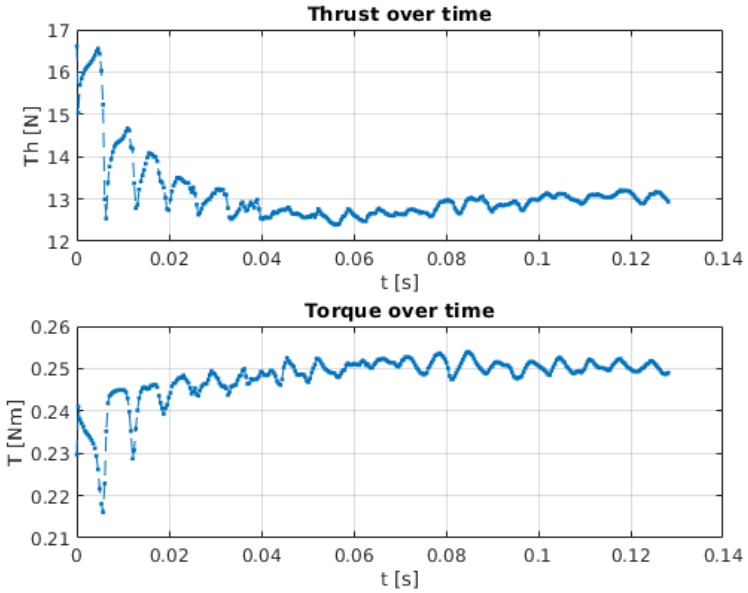


Figure 6.10: Thrust and torque temporal trend for cosine law with tip thickening panelization

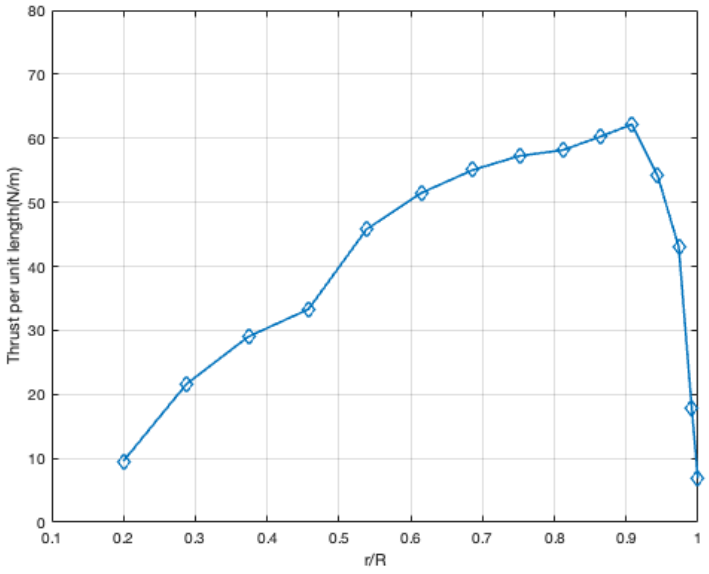
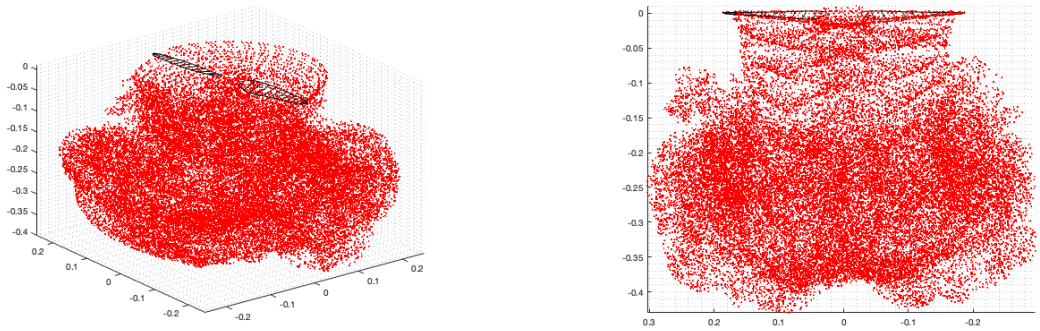


Figure 6.11: Thrust per unit length for cosine law with tip thickening panelization



(a) Isometric view **(b)** Frontal view

Figure 6.12: Final wake plot for cosine law with tip thickening panelization

induction is present, the lower will be the thrust, it can be hypothesized that a major concentration of particles on the tip results in a lower thrust there, while a minor concentration on the root stations will cause a higher thrust, as can be ascertained on the graph.

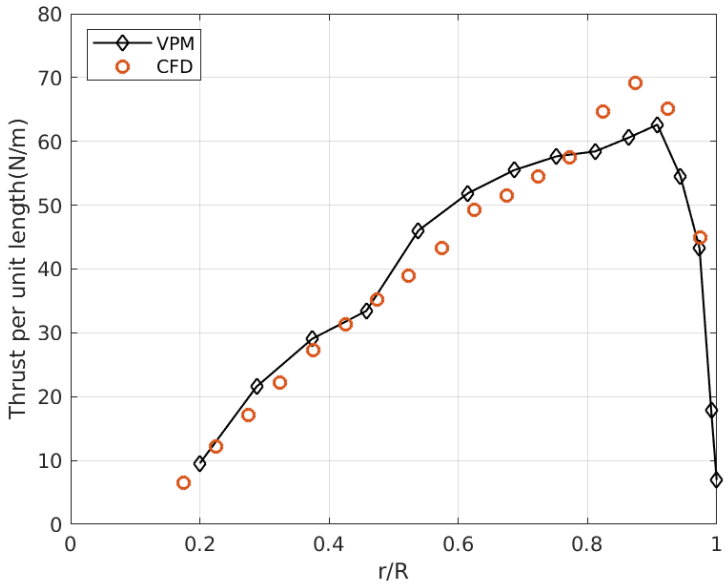


Figure 6.13: Thrust per unit length comparison vs CFD for homogeneous panelization

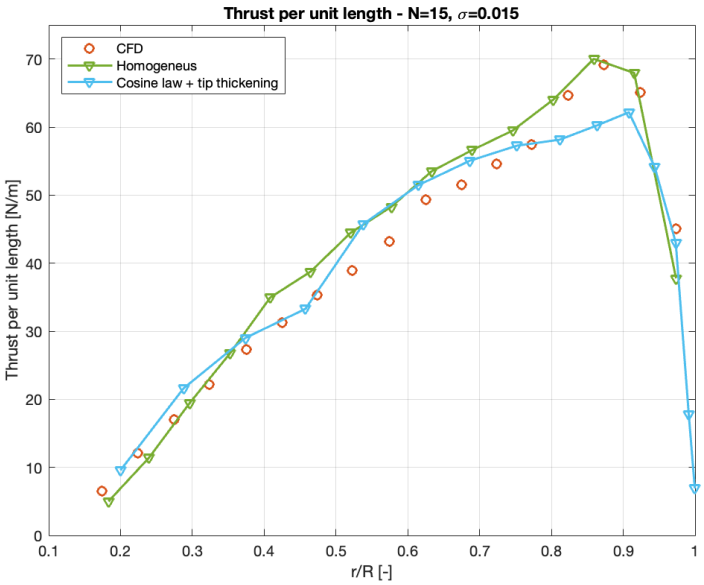


Figure 6.14: Thrust per unit length comparison vs CFD for the two types of panelization

6.4 Conclusions

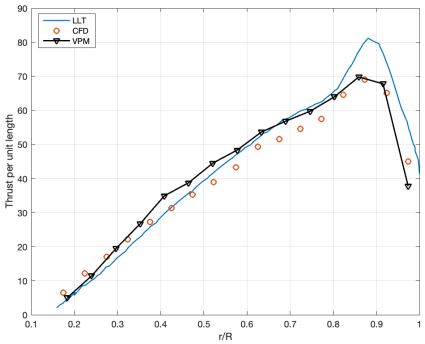
From the two discussed simulations it can be concluded firstly that the code has successfully been validated, since both the results agree with the CFD results in an error range that can be considered above $5 - 10\%$.
Then, it can be concluded that the code seems to be stable, as a high amount of particles has been shed and correctly evolved through a sufficient number of time-steps.
The thrust and the torque starts to assume an asymptotic value that differs from experimental and CFD results for an error that is above 5%.

	Experimental[40]	**CFD**[41]	**VPM (homog.)**	**VPM (cos.+tip)**	**LLT**[1]
Thrust [N]	13.41	13.16	13.67	12.94	13.88
Error wrt experimental	-	1.86%	1.94%	3.53%	3.39%
Torque [Nm]	0.253	0.246	0.259	0.249	0.257
Error wrt experimental	-	2.77%	2.2%	1.58%	1.56%
CPU Time [h]	-	2000	28		0.5

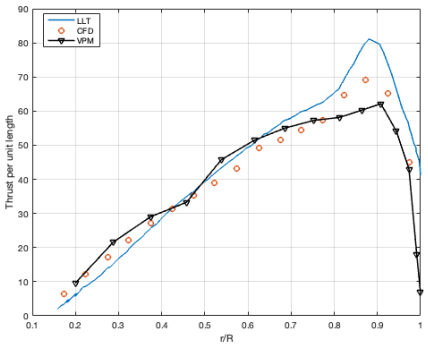
Table 6.2: Comparison of the results with several methods

In table 6.2 are reported the thrust and torque values for the experimental results[40], for the CFD results[41] for the current VPM code, and for the LLT code[1] that has been widely cited through this work.
As it can be seen, the current VPM code produces satisfactory results, that can be even compared to CFD, but with a much lower computational effort.
The two simulations have been indeed performed on a 2 core machine, resulting in a computational time of 14 hours for the first one and 20 hours for the second one, highlighting the significant timesaving, also thanks to the FMM integration on the code.
The homogeneous panelization is more accurate on the thrust prediction, while the cosine law with tip thickening is more accurate on the torque one.

In figure 6.15 is reported the thrust per unit length comparison between CFD, VPM and LLT for both the types of panelization.
It can be observed that the LLT overpredicts the thrust at the tip station, while the other ones are well resolved.
This highlight the effective conceptual difference between these two methods, especially through the fact that the VPM describes the flow field in terms of vortex particles, avoiding eventual connectivity of the vortex structures.
Additionally, the VPM code uses a third-order accurate integration scheme, while the LLT code[1] uses a first or second-order integration scheme, so eventual differences might rely also upon this.



(a) Homogeneous panelization



(b) Cosine law with tip thickening panelization

Figure 6.15: Thrust per unit length comparison with respect to CFD and LLT results for both the panelization types

Chapter 7

Conclusions and future works

This chapter includes the main conclusions on the entire thesis work, since the conclusions on every aspect that has been dealt with through the dissertation are located at the end of each of them, and includes also several possible future works that can originate from this thesis.

7.1 Conclusions

At this point, it can be stated that this thesis work achieved its major goals.
The code has been validated through both VPM evolution equations, FMM routine, and the entire coupling with the LLT.
It has also ascertained the explicit integration scheme stability.

Moreover, the developed code represents a fast but accurate design tool to predict the performances of rotors, propellers, and even wings.
The results, as seen in chapter 6, agree very well with CFD and experimental data, with the advantage of a much lower computational effort, thanks also to the implementation of the Fast Multipole Method.

However, this code has its unavoidable limitations:

- As stated by Alvarez in his work[17], the divergence relaxation by Pedrizzetti in its original form has the drawback of reducing the circulation strength magnitude.

  Since it has been clarified in the same work that a modified formulation provides instabilities to the code, it is still an open question of how to fix this

problem.

- Due to the meager current literature on the topic, it is still unclear how to correctly set up several parameters such as the particle core size, the divergence relaxation frequency, etc.

- Although it has shown that the FMM coupling is still efficient, better alternatives might exist, mainly because the link occurs with a heavy alternation between MATLAB and C++.

- To correctly predict the performances of the lifting surface, the code requires an aerodynamic database. This can be achieved by simulating several blade stations with CFD or XFoil.

  The dependence on 2D simulations makes the code highly sensible to Re variations, especially in low Re regimes.

7.2 Future works

As stated in the previous section, the developed code presents inescapable drawbacks and limitations and can be therefore improved and optimized.

- It can be coupled with a Vortex Lattice Method (VLM), instead of the here used Lifting Line theory, to compute a more accurate circulation distribution.

  This could help since the particle generation itself relies mainly upon the circulation distribution.

- It can be developed as an ad-hoc MATLAB FMM routine, to eliminate the heavy interface between the two programming languages that are currently implemented in the code.

  This will also allow the same code speed on every operating system since the Windows implementation of FMM relies on a further bridge that is represented by the Windows Subsystem for Linux.

  On the other hand, it may be thought to translate the entire VPM code into a suitable programming language (for example Python) which can already benefit from some FMM routine.

- The code could also benefit from further integration of a turbulence model, as done by Alvarez in his work[17].

  This will allow the computation of an eventual turbulent flow field.

- The entire VPM code can be coupled with a 6DOF solver, allowing the computation of critical maneuvers of multicopter, tilt-rotor, etc.

  This will be particularly suitable since the VPM is meshless and does not require any connectivity of the vortex structures, so the routine may simply compute the performances of the blades every time step and further integrate the state variables, according to governing equations, to accurately predict the kinematics and dynamics of this complicated motions.

- Lastly, the work done in this thesis can be extended to other reduced-order models (ROMs), allowing them to converge into a single stand-alone integrated tool.

  This would be a much more powerful solution to mostly every design phase, since the user can choose the method that considers most suitable for that specific phase, according mainly to the desired level of computational speed and subsequent accuracy.

Appendix A

General regularized VPM evolution equations

In this appendix are presented the regularized VPM evolution equations, along with the most common regularization functions.

A.1 Other regularization functions

In chapter 3 they have been presented the regularized evolution equations in a general form, where the regularization functions $q(\rho)$, $\zeta(\rho)$ and $\chi(\rho)$ appears.
In table A.1 are provided additional regularization functions which can be used to rewrite evolution equations.

Regularization function	$q(\rho)$	$\zeta(\rho)$	$\chi(\rho)$
Second order Gaussian	$\frac{1}{4\pi}\left(1-e^{-\rho^3}\right)$	$\frac{3}{4\pi}e^{-\rho^3}$	
Super Gaussian	$\frac{1}{4\pi}\left(erf\left(\frac{\rho}{\sqrt{2}}\right)-\sqrt{\frac{2}{\pi}}\rho\left(1-\frac{\rho^2}{2}\right)e^{-\frac{\rho^2}{2}}\right)$	$\frac{1}{(2\pi)^{\frac{3}{2}}}\left(\frac{5}{2}-\frac{\rho^2}{2}\right)e^{-\frac{\rho^2}{2}}$	$\frac{1}{4\pi}\left(\frac{1}{\rho}erf\left(\frac{\rho}{\sqrt{2}}+\frac{1}{\sqrt{2\pi}}e^{-\frac{\rho^2}{2}}\right)\right)$
Low Order Algebraic	$\frac{1}{4\pi}\frac{\rho^3}{(\rho^2+1)^{\frac{3}{2}}}$	$\frac{3}{4\pi}\frac{1}{(\rho^2+1)^{\frac{5}{2}}}$	$\frac{1}{4\pi}\frac{1}{(\rho^2+1)^{\frac{1}{2}}}$
High Order Algebraic	$\frac{1}{4\pi}\frac{\rho^3(\rho^2+\frac{5}{2})}{(\rho^2+1)^{\frac{5}{2}}}$	$\frac{18}{5\pi}\frac{1}{(\rho^2+1)^{\frac{7}{2}}}$	$\frac{1}{4\pi}\frac{\rho^2+\frac{3}{2}}{(\rho^2+1)^{\frac{3}{2}}}$

Table A.1: $q(\rho)$, $\zeta(\rho)$, and $\chi(\rho)$ for the most common regularization functions.

A.2 Classical and transpose formulation for vortex stretching term

Another issue that can be afforded regards the formulations for modeling the vortex stretching term in the circulation strength update equation (classical, transpose, mixed).

The general formulation for circulation strength update equation for the 3 aforementioned formulations are reported in equations A.1, A.2, A.3.

$$\left.\frac{d\vec{\alpha}^p}{dt}\right|_{classic} = \sum_{q=1}^{N_p} -\frac{1}{\sigma^3}\frac{q(\rho)}{\rho^3}\vec{\alpha}^p(t)\times\vec{\alpha}^q(t) + \frac{1}{\sigma^5\rho}\frac{d}{d\rho}\left(\frac{q(\rho)}{\rho^3}\right)(\vec{\alpha}^p(t)\cdot(\vec{x}^p(t)-\vec{x}^q(t))\cdot((\vec{x}^p(t)-\vec{x}^q(t)\times\vec{\alpha}^q(t)) \quad \text{(A.1)}$$

$$\left.\frac{d\vec{\alpha}^p}{dt}\right|_{transpose} = \sum_{q=1}^{N_p} \frac{1}{\sigma^3}\frac{q(\rho)}{\rho^3}\vec{\alpha}^p(t)\times\vec{\alpha}^q(t) + \frac{1}{\sigma^5\rho}\frac{d}{d\rho}\left(\frac{q(\rho)}{\rho^3}\right)(\vec{\alpha}^p(t)\cdot(\vec{x}^p(t)-\vec{x}^q(t))\times\vec{\alpha}^q(t))(\vec{x}^p(t)-\vec{x}^q(t)) \quad \text{(A.2)}$$

$$\left.\frac{d\vec{\alpha}^p}{dt}\right|_{mixed} = \sum_{q=1}^{Np} \frac{1}{2\sigma^5\rho}\frac{d}{d\rho}\left(\frac{q(\rho)}{\rho^3}\right)[\vec{\alpha}^p(t)\cdot(\vec{x}^p(t)-\vec{x}^q(t))\cdot((\vec{x}^p(t)-\vec{x}^q(t)\times\vec{\alpha}^q(t)+\vec{\alpha}^p(t)\cdot(\vec{x}^p(t)-\vec{x}^q(t))\times\vec{\alpha}^q(t))(\vec{x}^p(t)-\vec{x}^q(t)] \quad \text{(A.3)}$$

It can be shown that the classical and the transpose schemes holds the same informations since

$$[\nabla\vec{u}]\vec{\omega} = [\nabla\vec{u}]^T\vec{\omega} \quad \text{(A.4)}$$

Where

$$[\nabla\vec{u}] = \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_y}{\partial x} & \frac{\partial u_z}{\partial x} \\ \frac{\partial u_x}{\partial y} & \frac{\partial u_y}{\partial y} & \frac{\partial u_z}{\partial y} \\ \frac{\partial u_x}{\partial z} & \frac{\partial u_y}{\partial z} & \frac{\partial u_z}{\partial z} \end{bmatrix} \quad \text{(A.5)}$$

So that

$$\begin{aligned} [\nabla\vec{u}]\vec{\omega} - [\nabla\vec{u}]^T\vec{\omega} &= \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_y}{\partial x} & \frac{\partial u_z}{\partial x} \\ \frac{\partial u_x}{\partial y} & \frac{\partial u_y}{\partial y} & \frac{\partial u_z}{\partial y} \\ \frac{\partial u_x}{\partial z} & \frac{\partial u_y}{\partial z} & \frac{\partial u_z}{\partial z} \end{bmatrix}\begin{Bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{Bmatrix} - \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_x}{\partial y} & \frac{\partial u_x}{\partial z} \\ \frac{\partial u_y}{\partial x} & \frac{\partial u_y}{\partial y} & \frac{\partial u_y}{\partial z} \\ \frac{\partial u_z}{\partial x} & \frac{\partial u_z}{\partial y} & \frac{\partial u_z}{\partial z} \end{bmatrix}\begin{Bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{Bmatrix} = \\ &= \begin{Bmatrix} \frac{\partial u_x}{\partial x}\omega_x + \frac{\partial u_y}{\partial x}\omega_y + \frac{\partial u_z}{\partial x}\omega_z - \frac{\partial u_x}{\partial x}\omega_x - \frac{\partial u_x}{\partial y}\omega_y - \frac{\partial u_x}{\partial z}\omega_z \\ \frac{\partial u_x}{\partial y}\omega_x + \frac{\partial u_y}{\partial y}\omega_y + \frac{\partial u_z}{\partial y}\omega_z - \frac{\partial u_y}{\partial x}\omega_x - \frac{\partial u_y}{\partial y}\omega_y - \frac{\partial u_y}{\partial z}\omega_z \\ \frac{\partial u_x}{\partial z}\omega_x + \frac{\partial u_y}{\partial z}\omega_y + \frac{\partial u_z}{\partial z}\omega_z - \frac{\partial u_z}{\partial x}\omega_x - \frac{\partial u_z}{\partial y}\omega_y - \frac{\partial u_z}{\partial z}\omega_z \end{Bmatrix} = \\ &= \begin{Bmatrix} \omega_y\left(\frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y}\right) - \omega_z\left(\frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x}\right) \\ \omega_z\left(\frac{\partial u_z}{\partial y} - \frac{\partial u_y}{\partial z}\right) - \omega_x\left(\frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y}\right) \\ \omega_x\left(\frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x}\right) - \omega_y\left(\frac{\partial u_z}{\partial y} - \frac{\partial u_y}{\partial z}\right) \end{Bmatrix} = \vec{\omega}\times\nabla\times\vec{u} = \omega\times\omega = 0 \end{aligned} \quad \text{(A.6)}$$

However, as stated through this work, the transpose formulation is the only one that conserves vorticity.

Appendix B

Complete derivation of VPM evolution equation for the FMM

The Fast Multipole Method is able to solve a problem in the form

$$\phi_i = \sum_{j=1}^{N} K(x_i, x_j) q_j$$

It has been therefore necessary to recast the evolution equations 3.45, 3.46 to expression in the form

$$\frac{d\vec{x}^p}{dt} = \sum_{m=1}^{M} \phi_m \tag{B.1}$$

$$\frac{d\vec{\alpha}^p}{dt} = \sum_{f=1}^{F} \phi_f \tag{B.2}$$

To do this, have been expanded all the vector and dot products and then they have been considered separately the 3 components of every evolution equation.

B.1 Position update

Calling

$$d\vec{x} = \vec{x}^p(t) - \vec{x}^q(t) \tag{B.3}$$

It is possible to rewrite equation 3.45 as

$$\frac{d\vec{x}^p}{dt} = -\frac{1}{4\pi} \sum_{q=1}^{N_p} \frac{|d\vec{x}^2| + \frac{5}{2}\sigma^2}{(|d\vec{x}|^2 + \sigma^2)^{\frac{5}{2}}} d\vec{x} \times \vec{\alpha}^q(t) \tag{B.4}$$

Expanding the vector product and considering each component of $\frac{d\vec{x}^p}{dt}$ one can obtain

$$\frac{dx^p}{dt} = -\frac{1}{4\pi}\sum_{q=1}^{N_p}(K_y\alpha_z^q(t) - K_z\alpha_y^q(t)) \tag{B.5}$$

$$\frac{dy^p}{dt} = -\frac{1}{4\pi}\sum_{q=1}^{N_p}(K_z\alpha_x^q(t) - K_x\alpha_z^q(t)) \tag{B.6}$$

$$\frac{dz^p}{dt} = -\frac{1}{4\pi}\sum_{q=1}^{N_p}(K_x\alpha_y^q(t) - K_y\alpha_x^q(t)) \tag{B.7}$$

Where x^p, y^p, and z^p are the components of $\vec{x}^p(t)$ and α_x^q, α_y^q, and α_z^q are the components of $\vec{\alpha}^q(t)$.
Instead, the kernels K_{x_j} are defined as

$$K_{x_j} = K dx_j \tag{B.8}$$

Where K is the regularized Biot-Savart kernel $\frac{|d\vec{x}^2|+\frac{5}{2}\sigma^2}{(|d\vec{x}|^2+\sigma^2)^{\frac{5}{2}}}$ and dx_j is the j-th component of the vector $d\vec{x}$.

Now we have obtained a set of equations in the form of B.1 that can be solved with the FMM routine.

B.2 Circulation strength update

The circulation strength update equation can be firstly seen as

$$\frac{d\vec{\alpha}^p}{dt} = \frac{1}{4\pi}\sum_{q=1}^{N_p}[K_1(\vec{\alpha}^p(t)\times\vec{\alpha}^q(t)) + K_2(\vec{x}^p(t)-\vec{x}^q(t)) + K_3 vol^p\vec{\alpha}^q(t) - K_3\vec{\alpha}^q(t)vol^q] \tag{B.9}$$

Where

$$K_1 = \frac{|d\vec{x}^2| + \frac{5}{2}\sigma^2}{(|d\vec{x}|^2+\sigma^2)^{\frac{5}{2}}} \tag{B.10}$$

$$K_2 = 3\frac{|d\vec{x}|^2 + \frac{7}{2}\sigma^2}{(|d\vec{x}|^2+\sigma^2)^{\frac{7}{2}}}\vec{\alpha}^p(t)\cdot(d\vec{x}\times\vec{\alpha}^q(t)) \tag{B.11}$$

$$K_3 = 105\nu\frac{\sigma^4}{(|d\vec{x}|^2+\sigma^2)^{\frac{9}{2}}} \tag{B.12}$$

By expanding all the cross and dot products and considering each component of circulation strength update, one can obtain

$$\frac{d\alpha^p_{x_1}}{dt} = \frac{1}{4\pi}\sum_{q=1} N_p(K_1\alpha^p_y\alpha^q_z - K_1\alpha^p_z\alpha^q_y) \tag{B.13}$$

$$\frac{d\alpha^p_{y_1}}{dt} = \frac{1}{4\pi}\sum_{q=1} N_p(K_1\alpha^p_z\alpha^q_x - K_1\alpha^p_x\alpha^q_z) \tag{B.14}$$

$$\frac{d\alpha^p_{z_1}}{dt} = \frac{1}{4\pi}\sum_{q=1} N_p(K_1\alpha^p_x\alpha^q_y - K_1\alpha^p_y\alpha^q_x) \tag{B.15}$$

$$\frac{d\alpha^p_{x_2}}{dt} = \frac{1}{4\pi}\sum_{q=1}^{N_p}(K_{2_{xy}}\alpha^p_x\alpha^q_z - K_{2_{xz}}\alpha^p_x\alpha^q_y + K_{2_{xz}}\alpha^p_y\alpha^q_x - K_{2_{xx}}\alpha^p_y\alpha^q_z + K_{2_{xx}}\alpha^p_z\alpha^q_y - K_{2_{xy}}\alpha^p_z\alpha^q_x) \tag{B.16}$$

$$\frac{d\alpha^p_{y_2}}{dt} = \frac{1}{4\pi}\sum_{q=1}^{N_p}(K_{2_{yy}}\alpha^p_x\alpha^q_z - K_{2_{yz}}\alpha^p_x\alpha^q_y + K_{2_{yz}}\alpha^p_y\alpha^q_x - K_{2_{xy}}\alpha^p_y\alpha^q_z + K_{2_{xy}}\alpha^p_z\alpha^q_y - K_{2_{yy}}\alpha^p_z\alpha^q_x) \tag{B.17}$$

$$\frac{d\alpha^p_{z_2}}{dt} = \frac{1}{4\pi}\sum_{q=1}^{N_p}(K_{2_{yz}}\alpha^p_x\alpha^q_z - K_{2_{zz}}\alpha^p_x\alpha^q_y + K_{2_{zz}}\alpha^p_y\alpha^q_x - K_{2_{xz}}\alpha^p_y\alpha^q_z + K_{2_{xz}}\alpha^p_z\alpha^q_y - K_{2_{yz}}\alpha^p_z\alpha^q_x) \tag{B.18}$$

$$\frac{d\alpha^p_{x_3}}{dt} = \frac{1}{4\pi}\sum_{q=1}^{N_p}(K_3 vol^p\alpha^q_x - K_3\alpha^p_x vol^q) \tag{B.19}$$

$$\frac{d\alpha^p_{y_3}}{dt} = \frac{1}{4\pi}\sum_{q=1}^{N_p}(K_3 vol^p\alpha^q_y - K_3\alpha^p_y vol^q) \tag{B.20}$$

$$\frac{d\alpha^p_{z_3}}{dt} = \frac{1}{4\pi}\sum_{q=1}^{N_p}(K_3 vol^p\alpha^q_z - K_3\alpha^p_z vol^q) \tag{B.21}$$

So that

$$\frac{d\alpha^p_x}{dt} = \frac{d\alpha^p_{x_1}}{dt} + \frac{d\alpha^p_{x_2}}{dt} + \frac{d\alpha^p_{x_3}}{dt} \tag{B.22}$$

$$\frac{d\alpha^p_y}{dt} = \frac{d\alpha^p_{y_1}}{dt} + \frac{d\alpha^p_{y_2}}{dt} + \frac{d\alpha^p_{y_3}}{dt} \tag{B.23}$$

$$\frac{d\alpha^p_z}{dt} = \frac{d\alpha^p_{z_1}}{dt} + \frac{d\alpha^p_{z_2}}{dt} + \frac{d\alpha^p_{z_3}}{dt} \tag{B.24}$$

Here the kernels $K_{2_{x_ix_j}}$ are defined as

$$K_{2_{x_ix_j}} = 3\frac{|d\vec{x}|^2 + \frac{7}{2}\sigma^2}{(|d\vec{x}|^2 + \sigma^2)^{\frac{7}{2}}}dx_idx_j \tag{B.25}$$

Now we have also obtained a set of equations in the form of B.2.

Appendix C

Low-storage third order Runge-Kutta scheme

The following scheme has been used to correctly calculate the position and circulation strength update for the particles. It has been widely used in the bibliography, maybe since it is the temporal integration scheme that Winclkemans[6] used in his pioneering work.
It was firstly laid out by Williamson[42] and further investigated in several NASA documents[20],[19].
The main advantage of this scheme is that it requires a storage equivalent to a typical first-order Runge-Kutta scheme since it overwrites the solution at every stage on the same variable.

To solve the initial value problem

$$\frac{dU}{dt} = F[t, U(t)], \qquad U(t_0) = U_0 \tag{C.1}$$

Williamson provided the following scheme for each stage. Assuming a M stage scheme, it is

$$\begin{cases} dU_j = A_j dU_{j-1} + dtF(U_j) \\ U_j = U_{j-1} + B_j dU_j \end{cases}, \qquad \forall j = 1, \ldots, M \tag{C.2}$$

Here A_j and B_j are related to the traditional Butcher array variables $a_{i,j}$, b_j, and c_j as

$$\begin{cases} B_j = a_{j+1,j} & \\ B_M = b_M & \\ A_j = \frac{b_{j-1} - B_{j-1}}{b_j}, & (j \neq 1, b_j \neq 0) \\ A_j = \frac{a_{j+1,j-1} - c_j}{B_j}, & (j \neq 1, b_j = 0) \end{cases} \tag{C.3}$$

By writing down the Butcher array for a 4 stages scheme in terms of A_j and B_j and considering all the additional constraints as reported in the NASA Technical Memorandum 109111[19], it results in a one-parameter family of schemes; in particular, the free parameter is the variable c_3.
As reported in the aforementioned Technical Memorandum, the most used value of this parameter, mainly for reasons of stability and accuracy, is

$$c_3 = \frac{1 + \sqrt[3]{\frac{5}{4}}}{3}$$

Therefore, it has been chosen this value, resulting in the following values of A_j and B_j, that are reported in table C.1, where X and Y are defined as

$$X = 12c_3^3 - 24c_3^2 + 16c_3 - 3 \tag{C.4}$$

$$Y = 6c_3^2 - 6c_3 + 1 \tag{C.5}$$

j	A_j	B_j
1	0	$\frac{3c_3-2}{6c_3-3}$
2	$-\frac{36c_3^3-48c_3^2+18c_3-1}{9(2c_3-1)^3}$	$\frac{2(c_3-1)^2}{6c_3-4}$
3	$\frac{(9c_3-9)(2c_3-1^3)}{3c_3-2}$	$-\frac{c_3-1}{X}$
4	$-\frac{1}{X}$	$\frac{c_3(12c_3^2-18c_3+7)}{(6c_3-6)Y}$

Table C.1: A_j and B_j coefficients for the implemented LSRK3 scheme

To check the effective correctness, accuracy and convergence of the implemented scheme, it has been solved the test problem

$$\frac{dU}{dx} = U\cos(x), \qquad U(0) = 1$$

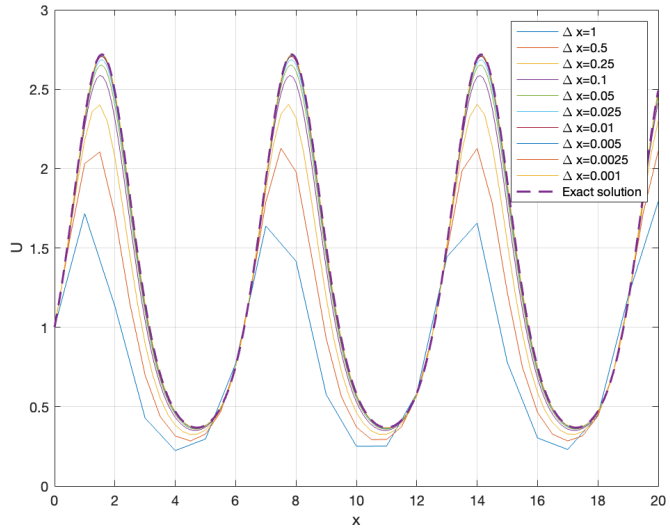
which has exact solution

$$U(x) = e^{\sin(x)}$$

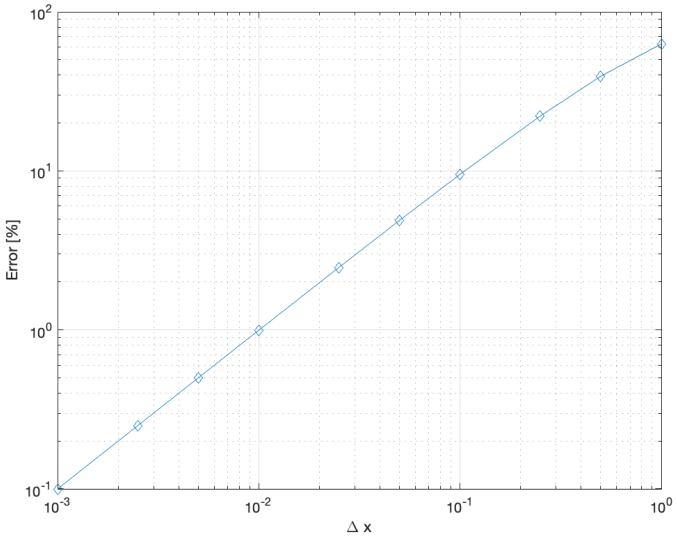
They have been performed multiple calculations for several values of the integration step Δx and the solution is reported in figure C.1a, while the error between exact and numerical solution is reported in figure C.1b.
As it can be seen, the numerical solution tends to be the exact one as the integration step is reduced and the error trend with the Δx shows a satisfactory convergence

velocity.

However, since this is an explicit method, it must be addressed with consistent attention to the integration step, to fulfill the stability condition every time step.



(a) Solution



(b) Error

Figure C.1: Test problem for LSRK3 scheme

Appendix D

Particles generation

Because the correct generation of the particles, to model the wake as best as possible, is one of the key points of a complete VPM code, it is necessary to further investigate this topic.
The particle generation is governed by the conservation of vorticity. The change in circulation over the blade generates new particles in the field such that the net vorticity is conserved over time.
Although it is possible to model both attached and separated flow, this work has implemented only the modeling of attached flow.

As explained during this work, the particles generation involves two different kinds of phenomena:

- Trailing vortex particles, due to the span-wise varying circulation over the blade.

- Shed vortex particles, due to the time-varying circulation over the blade.

Each one is then implemented in the particles generation function.

Before starting, it can be useful to establish some common issues:

1. The particle generation, in terms of position, is done at the blade trailing edge. Since the circulation is calculated only at control points, it is considered constant over the chord and then the shedding of both kinds of particles is consequently done at trailing edge coordinates, starting from control point ones, simply translating over the chord itself the latter.

2. For each kind of particle they are calculated their position, their circulation strength, and their corresponding volume of fluid associated with each particle.

3. Due to the overlapping condition, the ratio between particles' core size σ and their spatial resolution h_{res} is fixed to an arbitrary value, which has to be strictly greater than 1, as previously mentioned.

In the following section, it will be referred to the airfoil cross-section area corresponding to each control point coordinates as A_i. During the simulations it have been used a value of $A_i = c_i t_i$, where c_i is the chord length at station i, and t_i is the corresponding airfoil thickness, assumed to be the 1% of the chord at the same station.

A schematic representation of what just explained can be found in figures D.1a, D.1b, D.1c.

D.1 Trailing particles

Consider the line connecting each trailing edge point at times $t - \Delta t$ and t; this line is uniformly divided into a set of m_{t_i} particles, where their number is calculated by dividing the length of the line by the spatial resolution.

$$m_{t_i} = \frac{|\vec{TE}_i(t) - \vec{TE}_i(t - \Delta t)|}{h_{res}} \tag{D.1}$$

This has to be an integer number, so its result will be rounded to $+\infty$.
Consequently, the distance between consecutive particles along the line is

$$d_{t_i} = \frac{|\vec{TE}_i(t) - \vec{TE}_i(t - \Delta t)|}{m_{t_i}} \tag{D.2}$$

Then, they are calculated the position, the circulation strength, and the associated volume of fluid of the m_{t_i} particles as

$$\vec{x}^j_{i_t}(t) = \vec{TE}_i(t-\Delta t) + \left(j - \frac{1}{2}\right) \frac{\vec{TE}_i(t) - \vec{TE}_i(t - \Delta t)}{m_{t_i}}, \qquad \forall j = 1, \dots, m_{t_i} \tag{D.3}$$

$$\vec{\alpha}^j_{i_t}(t) = [\Gamma_{i+1}(t) - \Gamma_i(t)] \frac{\vec{TE}_i(t) - \vec{TE}_i(t - \Delta t)}{m_{t_i}}, \qquad \forall j = 1, \dots, m_{t_i} \tag{D.4}$$

$$vol^j_{i_t} = A_i d_{t_i}, \qquad \forall j = 1, \dots, m_{t_i} \tag{D.5}$$

D.2 Shed particles

Shed particles are parallel to the trailing edge of the blade.
As the previous generation, they are firstly calculated the number of particles and the distance between consecutive ones as

$$m_{s_i} = \frac{|\vec{TE}_{i+1}(t-\Delta t) - \vec{TE}_i(t-\Delta t)|}{h_{res}} \tag{D.6}$$

$$d_{s_i} = \frac{|\vec{TE}_{i+1}(t-\Delta t) - \vec{TE}_i(t-\Delta t)|}{m_{s_i}} \tag{D.7}$$

Then, they are calculated the position, the circulation strength, and the associated volume of fluid of the m_{s_i} particles.

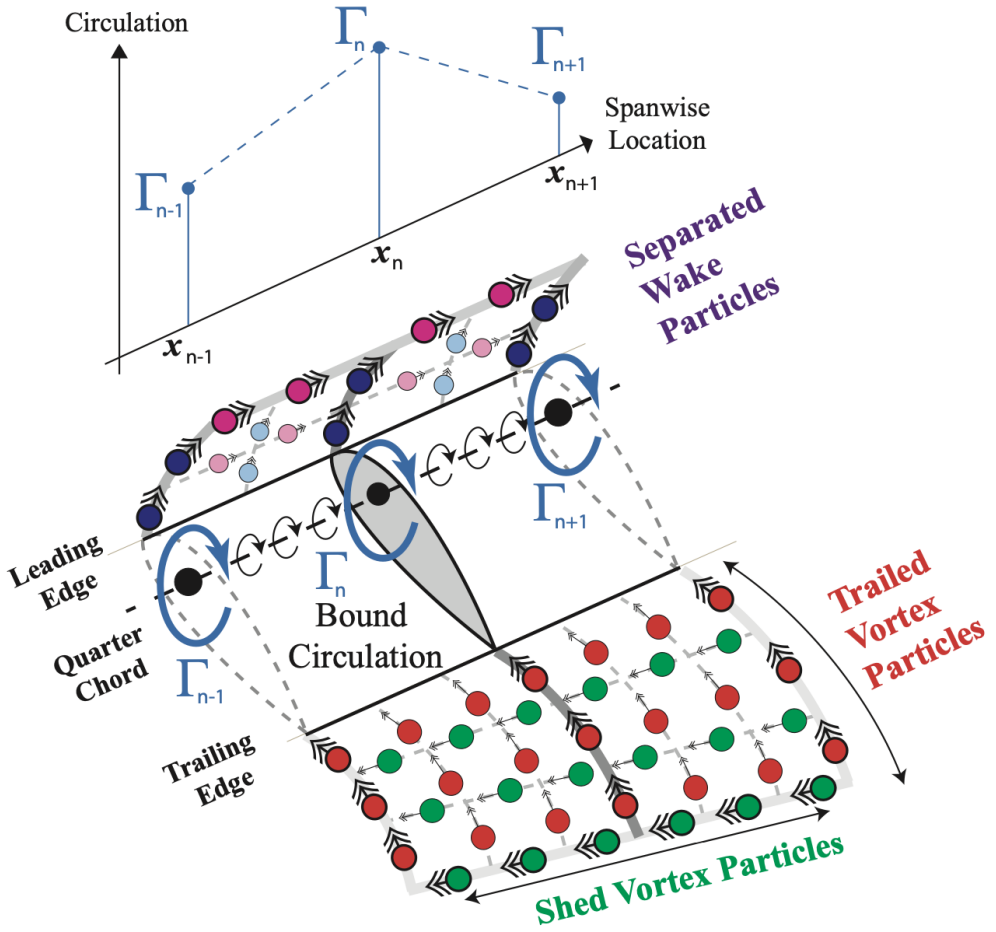
$$\vec{x}_{i_s}^j = \vec{TE}_i(t-\Delta t) + \left(j-\frac{1}{2}\right)\frac{|\vec{TE}_{i+1}(t-\Delta t) - \vec{TE}_i(t-\Delta t)|}{m_{s_i}} \tag{D.8}$$

$$\vec{\alpha}_{i_s}^j = [\Gamma_i^j(t-\Delta t) - \Gamma_i^j(t)]\frac{|\vec{TE}_{i+1}(t-\Delta t) - \vec{TE}_i(t-\Delta t)|}{m_{s_i}} \tag{D.9}$$

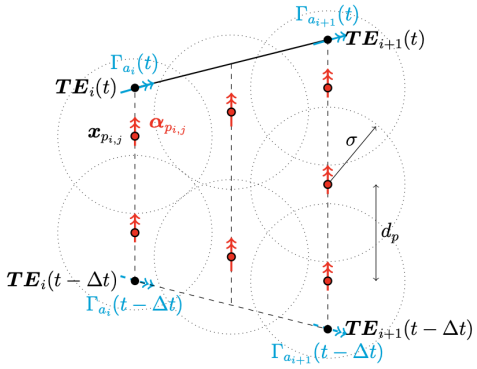
$$vol_{i_s}^j = \left[A_i + \left(j-\frac{1}{2}\right)\left(\frac{A_{i+1}-A_i}{m_{s_i}}\right)\right] d_{s_i} \tag{D.10}$$

Here, Γ_i^j is the circulation at the point j between the control point i and $i+1$ and it is calculated as

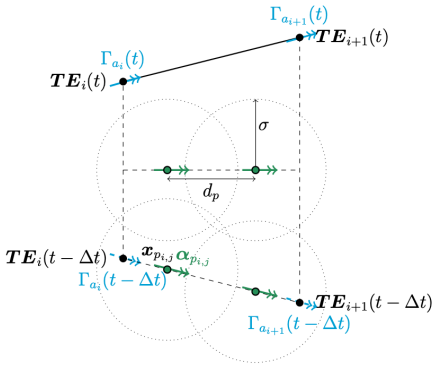
$$\Gamma_i^j(t) = \Gamma_i(t) + \left(j-\frac{1}{2}\right)\frac{\Gamma_{i+1}(t) - \Gamma_i(t)}{m_{s_i}} \tag{D.11}$$



(a) Particles generation scheme



(b) Trailing particles generation



(c) Shed particles generation

Figure D.1: Particles generation[15]

Bibliography

[1] Mario Alì. «Sviluppo e validazione di un metodo di ordine ridotto per lo studio del flusso su un rotore». MA thesis. Polytechnic of Turin, 2021 (cit. on pp. 1, 11, 12, 31, 32, 85, 86).

[2] Lorena A. Barba Tingyu Wang Rio Yokota. «ExaFMM: a high-performance fast multipole method library with C++ and Python interfaces». In: *The Journal of Open Source Software* (2021). DOI: doi:10.21105/joss.03145 (cit. on pp. 2, 12, 47, 52, 68, 72, 73).

[3] William Fong and Eric Darve. «The black-box fast multipole method». In: *Journal of Computational Physics* 228.23 (Dec. 2009), pp. 8712–8725. DOI: 10.1016/j.jcp.2009.08.031 (cit. on pp. 2, 12, 45, 46, 52, 66, 68, 72, 73).

[4] Tom J. Berdowski. «3D Lagrangian VPM-FMM for Modelling the Near-Wake of a HAWT». MA thesis. TU Delft, May 2015 (cit. on pp. 4, 9, 12, 20, 39, 52, 61).

[5] Georges-Henri Cottet and Petros D. Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, 2000. DOI: 10.1017/CBO9780511526442 (cit. on pp. 4, 12, 18, 23).

[6] Grégoire Stéphane Wincklemans. «Topics in vortex methods for the computation of three- and two-dimensional incompressible unsteady flows». MA thesis. California Institute of Technology, Feb. 1989 (cit. on pp. 5, 8, 12, 14, 16, 18, 20, 24, 25, 29, 31, 52–59, 96).

[7] Jacob Calabretta. «A three dimensional Vortex Particle - Panel code for modelling propeller - airframe interaction». MA thesis. Faculty of California Polytechnic State University, June 2010 (cit. on pp. 5, 10, 12, 18, 24, 57, 59).

[8] Venkatesan Seetharaman. «Implementation of a vortex particle scheme to analyze the wake of a wind turbine modeled using actuator lines». MA thesis. TU Delft, Mar. 2019 (cit. on pp. 9, 12, 45, 52, 62, 65).

[9] Daniel Opoku, Dimitris G. Triantos, Fred Nitzsche, and Spyros Voutsinas. «Rotorcraft aerodynamic and aeroacoustic modelling using vortex particle methods». In: 2002 (cit. on p. 10).

[10] Chengjian He and Jinggen Zhao. «Modeling Rotor Wake Dynamics with Viscous Vortex Particle Method». In: *AIAA Journal* 47.4 (Apr. 2009), pp. 902–915. DOI: 10.2514/1.36466 (cit. on p. 10).

[11] A. Zervos, S. Huberson, and A. Hemon. «Three-dimensional free wake calculation of wind turbine wakes». In: *Journal of Wind Engineering and Industrial Aerodynamics* 27.1-3 (Jan. 1988), pp. 65–76. DOI: 10.1016/0167-6105(88)90024-4 (cit. on p. 10).

[12] Evan H. Martin. «Assessment of Panel and Vortex Particle Methods for the Modelling of Stationary Propeller Wake Wash». MA thesis. Memorial University of Newfoundland, Oct. 2015 (cit. on pp. 10, 12, 52, 59).

[13] Eduardo J. Alvarez and Andrew Ning. «Development of a vortex particle code for the modeling of Wake Interaction in Distributed Propulsion». In: *2018 Applied Aerodynamics Conference* (2018). DOI: 10.2514/6.2018-3646 (cit. on pp. 10, 12, 45).

[14] Eduardo J. Alvarez and Andrew Ning. «Modeling Multirotor Aerodynamic Interactions Through the Vortex Particle Method». In: *AIAA Aviation 2019 Forum*. American Institute of Aeronautics and Astronautics, June 2019. DOI: 10.2514/6.2019-2827 (cit. on pp. 10, 12).

[15] Puneet Singh. «Aeromechanics of Coaxial Rotor Helicopters using the Viscous Vortex Particle Method». PhD thesis. University of Michigan, 2020 (cit. on pp. 10, 12, 31, 36, 52, 59, 102).

[16] David J. Willis, Jaime Peraire, and Jacob K. White. «A combined pFFT-multipole tree code, unsteady panel method with vortex particle wakes». In: *International Journal for Numerical Methods in Fluids* 53.8 (2007), pp. 1399–1422. DOI: 10.1002/fld.1240 (cit. on p. 11).

[17] Eduardo J. Alvarez. «Reformulated Vortex Particle Method and Meshless Large Eddy Simulation of Multirotor Aircraft». PhD thesis. Brigham Young University, June 2022 (cit. on pp. 12, 24, 25, 48, 87, 88).

[18] Jens Niegemann, Richard Diehl, and Kurt Busch. «Efficient low-storage Runge-Kutta schemes with optimized stability regions». In: *Journal of Computational Physics* 231.2 (Jan. 2012), pp. 364–372. DOI: 10.1016/j.jcp.2011.09.003 (cit. on p. 12).

[19] Mark H. Carpenter and Christopher A. Kennedy. *Third-order 2N-storage Runge-Kutta schemes with error control*. Tech. rep. National Aeronautics and Space Administration, June 1994 (cit. on pp. 12, 96, 97).

[20] Mark H. Carpenter, Christopher A. Kennedy, and R. Michael Lewis. *Low-storage, Explicit Runge-Kutta Schemes for the Compressible Navier-Stokes Equations*. Tech. rep. National Aeronautics and Space Administration, June 1999 (cit. on pp. 12, 96).

[21] Ahmed F. Ghoniem Omar M. Knio. «Numerical study of a three-dimensional vortex method». In: *Journal of Computational physics* 86 (1990), pp. 75–106 (cit. on p. 12).

[22] Ian S. Sullivan, Joseph J. Niemela, Robert E. Hershberger, Diogo Bolster, and Russell J. Donnelly. «Dynamics of thin vortex rings». In: *Journal of Fluid Mechanics* 609 (July 2008), pp. 319–347. DOI: 10.1017/s0022112008002292 (cit. on pp. 12, 53).

[23] M. Konstantinov. «Numerical investigation of the interaction of coaxial vortex rings». In: *International Journal of Numerical Methods for Heat & Fluid Flow* 7.2/3 (Mar. 1997), pp. 120–140. DOI: 10.1108/09615539710163220 (cit. on pp. 12, 61).

[24] J. Thomas Beale. «On the Accuracy of Vortex Methods at Large Times». In: *The IMA Volumes in Mathematics and Its Applications*. Springer New York, 1988, pp. 19–32. DOI: 10.1007/978-1-4612-3882-9_2 (cit. on pp. 12, 23).

[25] Gianni Pedrizzetti. «Insight into singular vortex flows». In: *Fluid Dynamics Research* 10.2 (Aug. 1992), pp. 101–115. DOI: 10.1016/0169-5983(92)90011-k (cit. on pp. 12, 24).

[26] H. Cheng, L. Greengard, and V. Rokhlin. «A Fast Adaptive Multipole Algorithm in Three Dimensions». In: *Journal of Computational Physics* 155.2 (Nov. 1999), pp. 468–498. DOI: 10.1006/jcph.1999.6355 (cit. on pp. 12, 39).

[27] L Greengard and V Rokhlin. «A fast algorithm for particle simulations». In: *Journal of Computational Physics* 73.2 (Dec. 1987), pp. 325–348. DOI: 10.1016/0021-9991(87)90140-9 (cit. on pp. 12, 39).

[28] Tarun Kumar Sheel. «Acceleration of vortex methods calculation using FMM and MDGRAPE-3». In: *Progress In Electromagnetics Research B* 27 (2011), pp. 327–348. DOI: 10.2528/pierb10091804 (cit. on p. 12).

[29] Long Chen. *Introduction to fast multipole methods*. https://www.math.uci.edu/~chenlong/MathPKU/FMMsimple.pdf (cit. on p. 12).

[30] Louis F. Rossi. «Resurrecting Core Spreading Vortex Methods: A New Scheme that is Both Deterministic and Convergent». In: *SIAM Journal on Scientific Computing* 17.2 (1996), pp. 370–397. DOI: 10.1137/S1064827593254397 (cit. on p. 19).

[31] G.S. Winckelmans and A. Leonard. «Contributions to Vortex Particle Methods for the Computation of Three-Dimensional Incompressible Unsteady Flows». In: *Journal of Computational Physics* 109.2 (Dec. 1993), pp. 247–273. DOI: 10.1006/jcph.1993.1216 (cit. on p. 25).

[32] A. Van Garrel. «Development of a wind turbine aerodynamics simulation module». MA thesis. Memorial University of Newfoundland, Aug. 2003 (cit. on pp. 33, 35).

[33] William Squire and George Trapp. «Using Complex Variables to Estimate Derivatives of Real Functions». In: *SIAM Review* 40.1 (Jan. 1998), pp. 110–112. DOI: 10.1137/s003614459631241x (cit. on p. 45).

[34] Lexing Ying, George Biros, and Denis Zorin. «A kernel-independent adaptive fast multipole algorithm in two and three dimensions». In: *Journal of Computational Physics* 196.2 (May 2004), pp. 591–626. DOI: 10.1016/j.jcp.2003.11.021. URL: https://doi.org/10.1016/j.jcp.2003.11.021 (cit. on p. 47).

[35] A Leonard. «Computing Three-Dimensional Incompressible Flows with Vortex Elements». In: *Annual Review of Fluid Mechanics* 17.1 (1985), pp. 523–559. DOI: 10.1146/annurev.fl.17.010185.002515 (cit. on p. 54).

[36] F. R. Menter. «Two-equation eddy-viscosity turbulence models for engineering applications». In: *AIAA Journal* 32.8 (Aug. 1994), pp. 1598–1605. DOI: 10.2514/3.12149. URL: https://doi.org/10.2514/3.12149 (cit. on p. 76).

[37] Robin B Langtry and Florian R Menter. «Correlation-based transition modeling for unstructured parallelized computational fluid dynamics codes». In: *AIAA journal* 47.12 (2009), pp. 2894–2906 (cit. on p. 76).

[38] Manuel Carreno Ruiz and Domenic D'Ambrosio. «Validation of the $\gamma - Re_\theta$ Transition Model for Airfoils Operating in the Very Low Reynolds Number Regime». In: *Flow, Turbulence and Combustion* (June 2022). DOI: 10.1007/s10494-022-00331-z (cit. on p. 76).

[39] Manuel Carreno Ruiz, Andrea Manavella, and Domenic D'Ambrosio. «Numerical and experimental validation and comparison of reduced order models for small scale rotor hovering performance prediction». In: *AIAA SCITECH 2022 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2022. DOI: 10.2514/6.2022-0154 (cit. on p. 76).

[40] M Scanavino. «Design and testing methodologies for uavs under extreme environmental conditions». PhD thesis. Ph.D. thesis, Politecnico di Torino., 2021 (cit. on pp. 76, 85, 86).

[41] Manuel Carreno Ruiz, Matteo Scanavino, Domenic D'Ambrosio, Giorgio Guglieri, and Andrea Vilardi. «Experimental and numerical analysis of multicopter rotor aerodynamics». In: *AIAA AVIATION 2021 FORUM*. American Institute of Aeronautics and Astronautics, July 2021. DOI: 10.2514/6.2021-2539 (cit. on pp. 76, 85, 86).

[42] J.H Williamson. «Low-storage Runge-Kutta schemes». In: *Journal of Computational Physics* 35.1 (Mar. 1980), pp. 48–56. DOI: 10.1016/0021-9991(80)90033-9 (cit. on p. 96).