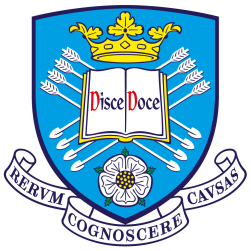


The University Of Sheffield.

Development of a GPU-accelerated flow simulation method for wind turbine applications

Ling Fang Cao

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

University of Sheffield
Faculty of Engineering
Department of Mechanical Engineering

February 7, 2022

Declaration of Authorship

I, Ling Fang CAO, declare that this thesis titled, "Development of a GPU-accelerated flow simulation method for wind turbine applications" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Success is not final, failure is not fatal: it is the courage to continue that counts. "

Winston Churchill

UNIVERSITY OF SHEFFIELD

Abstract

Faculty of Engineering
Department of Mechanical Engineering

Doctor of Philosophy

Development of a GPU-accelerated flow simulation method for wind turbine applications

by Ling Fang CAO

A new and novel GPU accelerated method has been developed for solving the Navier-Stokes equations for bodies of arbitrary geometry in both 2D and 3D. The present method utilises the vortex particles to discretize the governing equations in the Lagrangian frame. Those particles act as vorticity carriers which translate in accordance with the local velocity field. Vorticity information is thus propagated from the vorticity source to the rest of the flow domain in mimicking the advection and diffusion processes of the real flow.

In the high-fidelity method, vorticity generation can take place around the bodies. The no-slip condition produces a boundary flux which is subsequently diffused to the neighbouring particles. The new method has been successfully validated by simulating the flow field of an impulsively started cylinder. The calculated drag curve matches well with the theoretical prediction and other numerical results in the literature. To extend the applicability of the code to wind-turbine applications, a simplified re-meshing strategy is adopted which is found to produce small numerical inaccuracies.

In the engineering method, a simplified hybrid approach has been developed which decouples the advection and diffusion processes. The viscous effects are ignored on the bodies and are recovered in the wake. For this purpose, the Laplace equation that resulted from the irrotational assumption of the flow has been solved using the boundary element method. The solution produces a dipole distribution that is subsequently converted to viscous particles by employing the Hess' equivalence principle. In addition, an accurate interpolation scheme has been developed to evaluate the dipole gradient across the distorted wake geometry.

To reduce the simulation time, the fast multipole method has been implemented on the GPU in 2D and 3D. To parallelize the implementation, a novel data construction algorithm has been proposed. Furthermore, an analytical expression for the velocity strain has been derived.

The new developed methods have been applied to problems involving aerofoils and vertical axis wind turbines. Comparisons with experimental data have shown that the new techniques are accurate and can be used with confidence for a wide variety of wind turbine applications.

Acknowledgements

First of all, I would like to express my gratitude and appreciation to my supervisors: Prof. Lin Ma, Prof. Derek B Ingham and Prof. Mohamed Pourkashanian. Thanks go to Prof. Mohamed Pourkashanian for providing me with this wonderful PhD opportunity and welcoming me into the Energy2050 research group, Prof. Derek B Ingham for his tireless pastoral supports and encouragement during difficult and often emotional times and Prof. Lin Ma for his technical assistances and insightful discussions as well as providing me with encouragement at times where I most need it. Without their supports and assistance, this thesis would simply not be possible.

Secondly, I would like to extend my gratitude to all of my colleagues in the wind energy group (Jiaxin, Irving, Mohamed, Nidiana). Those people are some of the best people I have met and they are the source of inspiration for many aspects of life. They will be remembered as valuable friends.

Contents

Declaration of Authorship iii

Abstract vii

Acknowledgements ix

1 Introduction 1
- 1.1 The need for renewable energies . . . 1
- 1.2 Wind energy . . . 3
- 1.3 Wind in civil applications . . . 4
- 1.4 Wind mills and wind turbines . . . 6
- 1.5 Wind turbine parameters . . . 10
- 1.6 Recent research into VAWT wind-farms . . . 12

2 Literature review 13
- 2.1 Turbine aerodynamic performance prediction methods . . . 13
  - 2.1.1 The classical Blade-Element-Momentum theory . . . 13
  - 2.1.2 Double Multiple Streamtube . . . 14
- 2.2 Computational fluid dynamics (CFD) . . . 15
  - 2.2.1 The governing equations . . . 16
  - 2.2.2 Direct numerical simulation (DNS) . . . 17
  - 2.2.3 Turbulence modelling . . . 18
- 2.3 Vortex methods . . . 21
  - 2.3.1 The lifting line . . . 22
  - 2.3.2 The viscid-inviscid coupling algorithm . . . 23
  - 2.3.3 Double wake model for simulating separated flow past aerofoils 25
  - 2.3.4 High-fidelity vortex methods . . . 26
- 2.4 Aim and structure of the thesis . . . 26

3 Theories and methodologies 29
- 3.1 Vortex Particle Method (VPM)- the Lagrangian approach . . . 29
  - 3.1.1 Streamfunction-vorticity formulation . . . 29
  - 3.1.2 Mollification of the singular field . . . 31
  - 3.1.3 Particle-strength-exchange scheme (PSE) to model viscosity . . . 33
  - 3.1.4 Lagrangian grid distortion . . . 35
- 3.2 Boundary element method (BEM) . . . 37
  - 3.2.1 Mathematical formulation . . . 37
  - 3.2.2 Boundary conditions, panel discretization and the Kutta condition . . . 41
  - 3.2.3 Wake shedding angle and shedding length . . . 43
  - 3.2.4 Analytical evaluation of the influencing functions and the far field approximations . . . 45

3.2.5 A multipole expansion of the far-field approximation of the panels . . . 48
3.3 BEM and VPM coupling . . . 50
3.3.1 Hess' equivalent relationship . . . 50
3.3.2 Bilinear interpolation and computation of $\nabla\mu$ . . . 51
3.3.3 A robust least square fit for interpolating the dipole values . . . 54
3.4 Chapter conclusion . . . 55

4 Fast multipole methods **57**
4.1 Introduction to the fast multipole methods (FMM) . . . 57
4.2 Data structure . . . 59
4.2.1 The Morton index and the Z-order curve . . . 63
4.3 Two-dimensional FMM . . . 73
4.3.1 Multipole expansion of the complex velocity . . . 74
4.3.2 Translation operator of the 2D multipole expansion . . . 75
4.3.3 2D local expansion and the conversion operator . . . 76
4.3.4 Translation operator of the 2D local expansion . . . 78
4.4 Three-dimensional FMM . . . 79
4.4.1 Complex spherical harmonic basis (CSHB) . . . 80
4.4.2 Multipole expansion in the CSHB function space . . . 82
4.4.3 Translation operator of the 3D multipole expansion . . . 83
4.4.4 Conversion of multipole to local expansion and the translation operator of the local expansions . . . 83
4.4.5 Rotation operator for the CSHB functions . . . 84
4.4.6 Representing the multipole and local expansions using the real harmonic spherical functions . . . 86
4.4.7 Exact expression for the Biot-Savart induction and the strain field . . . 88
4.5 Initial expansion . . . 90
4.6 Upward pass . . . 91
4.7 Downward pass . . . 91
4.8 Final evaluation of the field . . . 92
4.9 Conclusion . . . 94

5 Validation of the vortex particle methods and performance measurements **97**
5.1 Dynamic evolution of an inviscid vortex ring . . . 97
5.1.1 Ring discretization . . . 97
5.1.2 A first order integration scheme in time . . . 99
5.1.3 Discussion . . . 99
5.2 Partition error of the FMM code . . . 100
5.3 Inviscid evolution of the (2:1) elliptical vortex patch (2D) . . . 102
5.3.1 Initial particle distribution . . . 102
5.3.2 Diagnostic: effective aspect ratio . . . 105
5.3.3 Discussion . . . 106
5.4 A simple validation case for the PSE scheme in 1D . . . 107
5.4.1 A choice for g_ϵ in the PSE scheme . . . 108
5.4.2 Solving Eq.(5.4.1) via finite difference . . . 108
5.4.3 Solving Eq.(5.4.1) via the PSE algorithm . . . 108
5.4.4 Discussion . . . 109
5.5 Performance measurement of the 2D FMM code . . . 110
5.5.1 Speed comparison between the direct approach and the FMM . 110

5.5.2 GPU FMM profiling . . . 111
5.6 Conclusion . . . 111

6 Two dimensional flows at moderate Reynolds number 119
6.1 A fractional time-stepping algorithm . . . 119
6.1.1 Operator splitting . . . 119
6.1.2 Modification of particle strength due to the boundary flux . . . 122
6.1.3 Enforcement of the no-slip condition . . . 125
6.1.4 The wall diffusion algorithm . . . 128
6.1.5 Numerical implementation . . . 131
6.1.6 Force calculation . . . 132
6.2 Transient flow past static cylinders . . . 133
6.2.1 Impulsively started cylinder at $Re = 550$ . . . 133
6.2.2 Impulsively started cylinder at higher Reynolds numbers . . . 138
6.3 Conclusion . . . 138

7 VAWT aerodynamic simulations 141
7.1 Aerofoil discretization and trailing edge smoothing . . . 141
7.2 Aerofoil aerodynamics . . . 145
7.2.1 Steady flow around a static NACA0012 at $\alpha = 15°$ . . . 145
7.2.2 Steady flow around a static NACA0012 at $\alpha = 30°$ . . . 150
7.2.3 Rotating NACA0012 at TSR $= 3.2$ . . . 152
7.3 Vertical axis wind turbine (VAWT) simulations . . . 155
7.3.1 BEM validation for the impulsively started NACA0012 . . . 158
7.3.2 Dynamic response of an isolated VAWT immersed in a uniform flow . . . 158
7.3.3 Wake structure of an isolated VAWT . . . 160
7.4 Performance and aerodynamic analysis of a pair of VAWTs . . . 165
7.4.1 Classification of rotor placement . . . 165
7.4.2 C1 simulations . . . 170
7.4.3 C2 simulations . . . 172
7.4.4 C3 simulation . . . 172
7.4.5 C4 simulations . . . 176
7.5 Conclusion . . . 179

8 Conclusion and future work 187
8.1 Primary conclusions . . . 187
8.2 Recommendation for future investigations . . . 189

A Compute Unified Device Architecture (CUDA) 191
A.1 Basic overview of CUDA . . . 191
A.2 The memory architecture . . . 192

B Derivation of the eigenvalue of the Sturm-Liouville problem 195

C Contour dynamic formula 197

List of Figures

1.1 Estimated renewable energy share in 2019 (REN21, 2019). 3
1.2 Schematic of the operation of a *drag device* 4
1.3 Examples of drag devices. 5
1.4 Schematic of the operation of a typical lifting-device. The turning force is produced via a combination of lift and drag forces, which are functions of the angle of attack (AoA). Typically, the lift force produced by the lifting-device is a magnitude larger than the generated drag force. 6
1.5 (A) shows the GE (General Electric) 1.5 MW HAWT. (B) illustrates the major components constituting a wind turbine. 8
1.6 Various VAWT designs in the market. 9

2.1 Schematic diagram describing how the wake is created from the variation of the bound circulation in the spatial and temporal domain. Here y is the span-wise coordinate on the wing. 23

3.1 Eq.(3.1.17) at different values of ϵ. It can be seen that ζ_ϵ approaches to the Dirac delta function asymptotically as $\epsilon \to 0$ 33
3.2 The magnitude of the mollified kernel as in Eq.(3.1.19) compared to the singular kernel, $|\nabla G|$ 34
3.3 M_4' is continuous in the first derivative. 36
3.4 Domain decomposition for solving Eq.(3.2.5). (A) is the original domain. (B) is the modified domain subjecting to two branch cuts. 38
3.5 Domain definition of the internal field ϕ^*. 39
3.6 Schematic of a low-order discretization of a 3D wing (A) and position of the collocation points used in the BEM formulation (B). 42
3.7 Nomenclature used to define the shedding geometry: α denotes the angle of the shedding line with respect to the x axis, while γ defines the length of the dipole panel during a single time-step. 44
3.8 Schematic diagram showing that the physical alignment of the shedding line is parallel to either the top or bottom surfaces; depending on the sign of the shed vorticity. 44
3.9 Iteration flow chart to determine the correct α and γ. Here, k is the iteration counter and α_k means the value of α after the k-th iteration. . 45
3.10 Near-field regions at different values of $f = 2.5, 5.0$ and 7.5 46
3.11 Normalized root-mean squared deviation of the induction velocity (u, v, w) as function of the partition parameter f. 47
3.12 (A) shows the NRMSD being plotted against the truncation number P for the induction field at $f = 5$, $N_b = 5$ and (B) shows the computational *effort* versus P 49
3.13 Nomenclature used in the bilinear interpolation of a convex quadrilateral dipole panel. The corner points are denoted by the tuple $\left(\mu_i, \vec{X}_i\right)$. 51

3.14 Example of the bilinear interpolation with some random corner values. 52
3.15 velocity field due to the far-field approximation of the dipole panel. a) x component, b) y component and c) z component. 54
3.16 velocity field due to the particle discretization with a fixed core radius $\epsilon =$ `1E-3`. a) x component, b) y component and c) z component. 54
3.17 Comparison of μ for an elliptical wing between the naive approach Eq.(3.3.18) and the robust least square fit with the fitting function given by Eq.(3.3.19) for an elliptical wing. 56

4.1 Typical components of the FMM routine. 59
4.2 The sets X and Y are well-separated if the distance between the open balls are separated by at least three times the radius. 60
4.3 Schematic diagram showing that the initial 2D domain D_0 is partitioned into the union of $D_{0,0}$, $D_{0,1}$, $D_{0,2}$ and $D_{0,3}$. Children boxes of subsequent refinements are conveniently identified by introducing the multi-dimensional index j such that $j = [0, i_1, i_2, \ldots i_n]$. 61
4.4 Illustration of how the cube is recursively divided in 3D. The resulting data structure is now an Octree. 62
4.5 An example of the `TreeConstruction` algorithm applied to a uniform set of source particles in the unit square. The source points are represented by the blue dot whilst the box centers are indicated by the red cross. Moreover, the empty childless boxes are not shown. 65
4.6 An example showing the domain division for an annular particle distribution. Again, the childless empty boxes are not shown here. 66
4.7 The z-order curve at the 2^{nd}, 4^{th} and 6^{th} iteration. This access pattern is completely determined by the interleaving operator R'. 69
4.8 From left to right, each schematic rectangle represents the memory address of the `particlePointerArray`, `permutationArray` and the particle data, respectively. Access to particle data is characterized by a series of memory mappings. However, the access pattern in the last phase is considered a random access, which may incur some speed penalties. 72
4.9 Schematic diagram showing that the effect of the map T_w is to translate the multipole coefficients in $B(r_0, w_0)$ to the ball $B(r_1, w_1)$. So for any field point z in the complement of $B(r_1, w_1)$, Eq.(4.3.12) is valid. . 76
4.10 Schematic diagram illustrating the distribution of the source particles which are constrained in the ball $B(R, w)$ with $w \neq 0$. The field point z has to satisfy the inequality $|w - z| > R$ in order for the local expansion to be convergent. 77

5.1 An example of an equal-area discretization for the 2D slice. 98
5.2 The three-dimensional ring is constructed by duplicating N_R copies of the cross-section around a central axis to form a torus (shown on the left). The vortex elements at each sectional station are assigned a constant circulation vector that is proportional to the circumferential unit vector $\vec{e}_\theta$ of that station (shown on the right). 99

5.3 The computed averaged position of the ring along the x axis. (A) A short time comparison with the theoretical result (Eq.(5.1.1)). The average velocity is calculated by fitting a linear regression to the position data. (B) Significant deviation from the theoretical result was observed for $t > 0.6$. This may be due to the fact that the core experiences a constant expansion. 101
5.4 Velocity contours in the x-component sampled at different times. Clearly, the core velocity is decreasing which is associated with the core expansion. 102
5.5 The domain partition results in the creation of large boxes to accommodate the excess particles (marked by the blue circles). In particular, convergence is severely hindered for field points inside the radius of divergence of the large boxes. In fact, the relative error curves have been computed at two reference points that are relatively close (marked by the green and magenta circle). The result is presented in Figure (5.6). 103
5.6 The normalized error curve as a function of the truncation number P at two field locations. (A) the field point is located inside the radius of divergence as outlined in Figure (5.5). (B) the field point is outside, which drastically improves the result by an order of magnitude. 104
5.7 The contour plot of the initial assignment of the particle circulations for the elliptical vortex. For the grid resolution $\delta x = 1.1 \times 10^{-3}$, a total of $N(0) = 35348$ particles are used to resolve the distribution. 105
5.8 Contour plots of the vorticity field at different time stamps of the elliptical vortex patch. The simulation demonstrates a complex filamentation process by which the ellipticity of the initial vortex is gradually transitioning to that of a circular configuration. No axisymmetrization is observed at the end of the run. 113
5.8 continued 114
5.9 Computed λ_{eff} versus those from Koumoutsakos (1997). The results agree very well for $t < 4$. But for $t > 4$, our computation shows that λ_{eff} does not converge to the circular configuration. Although the trend of the two results generally matches quite well, it is anticipated that this discrepancy could be due to the ways the particles are initialized. 115
5.10 Variation of particle size $N(t)$ as a function of time t. A sharp rise of particles numbers was observed during the filamentation process. At the end of the computation, the particle size has increased by a factor of 2.24. 116
5.11 A comparison of the vorticity profile between the PSE and finite-difference. The presented results correspond to the following simulation parameters: $T = 30$, $\nu = 10^{-2}$, $\delta x = 1.2 \times 10^{-2}$ and $\delta t = 10^{-2}$ 117

6.1 A square domain for Eq.(6.1.3). For simplification, a mixed boundary conditions were used. Only the bottom edge is subjected to the Neumann condition. 123
6.2 A comparison between the finite-difference method and Eq.(6.1.15). Figure depicts the solutions at $x = 0$ after $t = 0.064$. 124
6.3 Schematic diagram for the panel characterization (A). Once the flux is computed, the flux is parcelled to the neighbouring particles near the surface to satisfy the no-slip (B). 125

6.4 Comparison between scheme K and scheme W. The plot shows the variation of the change of circulation as a function of the vertical distance directly above the panel at a fixed x-station. The red curve corresponds to the 5-point Gauss-Legendre quadrature rule and it is to be compared to the MATLAB's built-in integration routine (diamond marker). 126
6.5 Numerical results for the static cylinder. (A) The vortex strength computed using the minimization procedure is compared to the analytical result. (B) The resulting streamline for the inviscid flow past the cylinder. 128
6.6 The computed boundary layer developed on the surface of the cylinder using a distribution of vortex particles (coloured dots). The accuracy of the boundary conditions depend on the grid spacing used. In this example, $h = 6.3 \times 10^{-4}$ was used. 129
6.7 Variation of the normal and tangential components of the velocity as a function of the normalized perpendicular distance (s/d) from the cylinder at an azimuthal station $\theta = 288^{\circ}$ at different resolutions h. The development of a surface boundary layer is clearly visible. 130
6.8 Schematic diagram depicting the main sequence of wall diffusion. Initially, the grid marks those cells that coincide with the body and are subsequently *turned-off*. Each panel is given a diffusion zone in which the wall flux is applied. Note that different diffusion zones might overlap. 131
6.9 Left shows the computed linear impulse I_x compared to Ploumhans and Winckelmans (2000) at $Re = 550$. Right shows the time-series of the total circulation in the flow domain. 135
6.10 Left shows the computed drag coefficient compared to the theoretical expression Eq.(6.2.2). Right shows the long time drag curve compared to Ploumhans and Winckelmans (2000). 135
6.11 Left shows the experimental streak-lines (Van Dyke, 1988) of the flow past the cylinder at $T = 1$ and $T = 3$, respectively. On the right, the numerical vorticity contours have been superimposed on top of the snapshots of the experimental streak-lines. The numeric result shows a fair agreement in terms of capturing the size of the recirculation zone as the wake develops. The Reynolds number for this experiment was $Re = 500$. 136
6.12 Vorticity contour plots for the impulsively started cylinder at the non-dimensionalized time shown. The Reynolds number of this flow is $Re = 550$ 137
6.13 Vorticity plots for the impulsively started cylinder at $Re = 9500$ for the case $D_0 = .5$. The wake quickly becomes asymmetric as time develops owing to the creation of two primary vortices on the surface boundary. 139
6.14 Vorticity plots for the impulsively started cylinder at $Re = 9500$ for the case $D_0 = 1.5$. The symmetry of the wake is now much better preserved. Though there is still some asymmetricity present. 140

7.1 A NACA5612 cross section profile constructed by Eq.(7.1.3). The dotted blue curve gives the mean camber line. 142

7.2 Example of the trailing edge smoothing using the quintic polynomial interpolant versus a simple circular closure. (A) The red curve corresponds to Eq.(7.1.5) whilst the blue curve shows a simple closure by a circle. (B) shows the curvature κ of the smoothing gap by the two methods. 144
7.3 Trailing edge node distribution for a NACA0012 by the cosine distribution (A) and the polynomial interpolation (B). Note that the node spacing around the gap is taken to be the averaged spacing of the last two trailing edge nodes. 145
7.4 Snapshots of the vorticity contour for the NACA0012 at the non-dimensionalized time shown. The geometric angle of attack is $\alpha = 15°$. 146
7.5 Snapshots of the non-dimensionalized u-velocity contour for the NACA0012 at the non-dimensionalized time shown. The geometric angle of attack is $\alpha = 15°$. 148
7.6 The plots show the experimental pathlines of the tracking particles conducted using the PIV by Huang et al. (2001). In increasing time, the snapshots are sequenced from top right to bottom, top left to bottom with the following timestamps: $T = 0, 0.520, 1.564, 1.825, 3.131, 3.914, 4.436, 4.958, 5.219, 6.524$ 149
7.7 C_F and C_D plots for the impuslively started NACA0012 at $\alpha = 15°$. The noisy black lines corresponds to the lift and drag coefficients obtained from Eq.(6.1.36) whilst the red line is the smoothed data using the moving box averaging technique with a 5-point averaging window. It can be observed that a large drop of lift occurs when the surface vortex has evolved to a sufficient length. 150
7.8 After the initial transient of the impulsively started NACA0012 at $\alpha = 15°$, alternating vortices are shed to create what is known as the von-Karma vortex street. 150
7.9 A sequence of streamlines captured at the non-dimensionalized times $T = 0.6521n - 0.5857, n = 1, 2, ..., 12$ at $\alpha = 30°$. Figures in the first and third column are the simulated results and the interpolated PIV streamlines (Huang et al., 2001) are given in the second and fourth column. All snapshots are taken at precisely the same time stamps as the experiment. The motion is sequenced from the top left to right and top to bottom at an equal time-interval. 153
7.10 Lift (A) and drag (B) coefficient for an impulsively started NACA0012 at $\alpha = 30°$. 154
7.11 Computed short time lift coefficient for the NACA0012 at $\alpha = 15°$ compared to Eq.(7.2.1). 155
7.12 A schematic diagram showing the typical motion cycle of the VAWT blade. The geometric angle of attack is expressed as a function of the azimuthal angle θ of the rotor. 156
7.13 Variation of α as a function of the tip-speed ratio (TSR) at a zero pitched angle $\beta = 0$. 156
7.14 The vorticity contour depicting the full sequence of the aerofoil motion in the upstream part of the VAWT cycle. θ^* is the shifted azimuthal angle. 157
7.15 Comparison of the pressure coefficient C_p for the steady solution for the NACA0012 pitched at $\alpha = 5°$ between the simulated result and XFOIL. 159
7.16 Computed streamline and pressure field for the NACA0012 at $\alpha = 5°$. 160

7.17 Evolution of the pressure coefficient (A) and the lift behaviour as a function of the non-dimensionalized time (B). 161
7.18 Computed normal and tangential force coefficients for the singled bladed rotor. The results are compared to the inviscid solution of Deglaire (2010) and the experimental result of Oler et al. (1983). 162
7.19 Computed normal and tangential force coefficients for the two-bladed rotor of Klimas (1982). 163
7.20 The idealized wake structure for the one-bladed, two-bladed and the three-bladed rotors. The idealized vortex sheet is convected by the free-stream velocity. 164
7.21 The computed streak-line of the wake (represented by blue markers) is superimposed onto the experimental streak-line for the one-bladed rotor. 165
7.22 Shed vortex circulation strength as a function of the shifted azimuthal angle for the one-bladed rotor. Dashed and dotted-dashed lines indicate the tip location at the advancing and receding side of the rotor, respectively. 166
7.23 Comparison of the streak-lines between simulation and experiment for a two-bladed rotor. 167
7.24 Wake markers for the two-bladed rotor after 20 revolutions. 168
7.25 Normalized u-velocity contour for the three rotors with increasing solidity (σ) from top to bottom. 169
7.26 Schematic diagrams showing the VAWT placements of the C1 and C2 configurations. C1 and C2 are both parametrized by the dimensionless constant $f > 1$. Here D denotes the diameter of the rotor. The free-stream is coming from the negative x-direction. 170
7.27 Schematic diagram showing the VAWT placements of the C3 and C4 configurations. 170
7.28 Computed power coefficient for the benchmark rotor. 171
7.29 (A) Shows the power coefficients between the benchmark case and C1R1 and C1R2. (B) Indicates the relative coefficients computed for each revolution. 172
7.30 Computed wake structure for the C1R1 (A) and C1R2 (B). 173
7.31 Computed wake structure for the C2R1 (A) and C2R2 (B) simulation. Symmetry is maintained in the near field , but a symmetry breaking event is observed due to numerical error in the simulations, which is amplified by the large number of time steps used. 174
7.32 (A) Shows the power coefficients between the benchmark case and C1 and C2. (B) Indicates the relative coefficients for all of the C1 and C2 simulations. 175
7.33 Normalized u-velocity contour at the symmetric plane for the C2R1 simulation. 176
7.34 Computed wake structure for the C3 simulation. 176
7.35 Computed normal and tangential coefficients for the upstream and downstream rotor for the C3R1 simulation. 177
7.36 The computed wake structure for the C4R1 (A) and C4R2 (B) simulation after the 20th revolution. 180
7.37 Computed normal and tangential coefficients for the upstream and downstream rotor for the C4R1 simulation. 181
7.38 Computed normal and tangential coefficients for the upstream and downstream rotor for the C4R2 simulation. 182

7.39 Computed power coefficients for the C4R1 and C4R2 simulations. . . . 183
7.40 Comparison of the power coefficients for the C4R1 and C4R2 simulations to the benchmark case. 184
7.41 (A) Shows the schematic interpretation of the configurations as defined by the incident wind direction. (B) Illustrates the definition of the wind incident angle θ_w. 185

A.1 A schematic illustration of the thread hierarchy implemented in NVIDIA et al. (2020). 192
A.2 A schematic illustration of the memory hierarchy implemented in NVIDIA et al. (2020). 193

List of Tables

1.1 Some examples of utility-scale HAWTs and VAWTs. 7

5.1 Parameters used in the simulation of the inviscid vortex ring. 100
5.2 shows the adverse effect of over-flow in the particle distribution, which could reduce the accuracy of the code by several orders of magnitude. 100
5.3 The vorticity moment parameters computed between Eq.(5.3.5) and Eq.(5.3.6). The absolute error is in the order of $\mathcal{O}\left(10^{-6}\right)$, which is sufficient for the current purpose. 105
5.4 Wall-clock time for the different modes of evaluation of the Biot-Savart. 110
5.5 Percentage of the total time the GPU-FMM code spent in each of the evaluation phases 111

6.1 Simulation parameters used for the impulsively started cylinder at $Re = 550$. 134

7.1 Simulation parameters used for the static NACA0012 at $Re = 1200$. Note here that no physical units are given because the units are normalized by the aerofoil chord and the free-stream speed, so that one could recover the units via dimensional analysis. 147
7.2 Darrieus rotor parameters in the experiment used by Oler et al. (1983) for the single bladed turbine 159
7.3 Rotor parameters used in the interaction of a VAWT pair. 166
7.4 Parameter matrix for the four configurations. The nomenclature can be found in Figure (7.26) and Figure (7.27). 171
7.5 Indicative percentage loss and gain for the downstream rotor due to the different wind direction characterised by the wind incident angle θ_w. 178

List of Abbreviations

AoA	**A**ngle **o**f **A**ttack
API	**A**pplication **P**rogramming **I**nterface
BEM	**B**oundary **E**lement **M**ethod
BVI	**B**lade **V**ortex **I**nteraction
CFD	**C**omputational **F**luid **D**ynamics
CFL	**C**ourant-**F**riedrichs-**L**ewy
CPU	**C**entral **P**rocessing **U**nits
CSHB	**C**omplex **S**pherical **H**armonic **B**asis
CUDA	**C**ompute **U**nified **D**evice **A**rchitecture
DOE	**D**epartment **O**f **E**nergy
DMST	**D**ouble **M**ultiple **S**tream**t**ube
DNS	**D**irect **N**umerical **S**imulation
FD	**F**inite **D**ifference
FMM	**F**ast **M**ultipole **M**ethod
GL	**G**auss **L**egendre
GPGPU	**G**eneral **P**urpose **G**raphics **P**rocessing **U**nits
GPU	**G**raphics **P**rocessing **U**nits
HAWT	**H**orizontal **A**xis **W**ind **T**urbine
L2L	**L**ocal **T**o **L**ocal
LES	**L**arge **E**ddy **S**imulation
LEV	**L**eading **E**dge **V**ortex
M2L	**M**ultipole **T**o **L**ocal
M2M	**M**ultipole **T**o **M**ultipole
MPI	**M**essage **P**assing **I**nterface
MST	**M**ultiple **S**tream**t**ube
NACA	**N**ational **A**dvisory **C**ommittee for **A**eronautics
NRMSD	**N**ormalized **R**oot-**M**ean **S**quared **D**eviation
NS	**N**avier-**S**tokes
ODE	**O**rdinary **D**ifferential **E**quations
OOP	**O**bject **O**riented **P**rogramming
P2M	**P**articles **T**o **M**ultipole
PDE	**P**artial **D**ifferential **E**quations
PIP	**P**oint-**I**n-**P**olygon
PIV	**P**article **I**mage **V**elocimetry
PSE	**P**article **S**trength **E**xchange
RANS	**R**eynolds **A**veraged **N**avier-**S**tokes
RHS	**R**ight **H**and **S**ide
RSM	**R**eynolds **S**tress equation **M**odel
SNL	**S**andia **N**ational **L**aboratories
SM	**S**treaming **M**ultiprocessors
TEV	**T**railing **E**dge **V**ortex
TSR	**T**ip-**S**peed **R**atio

VAWT	**V**ertical **A**xis **W**ind **T**urbine
VPM	**V**ortex **P**article **M**ethod

Physical Constants

Speed of Sound in air	$c_0 = 343\,\mathrm{m\,s^{-1}}$(at 20 °C)
Density of Water	$\rho_{\mathrm{water}} = 997\,\mathrm{kg\,m^{-3}}$(at 20 °C)
Density of Air	$\rho_{\mathrm{air}} = 1.225\,\mathrm{kg\,m^{-3}}$(at 20 °C)
Kinematic Viscosity of Water	$\nu_{\mathrm{water}} = 1.003 \times 10^{-6}\,\mathrm{m^2\,s^{-1}}$(at 20 °C)
Dynamic Viscosity of Water	$\mu_{\mathrm{water}} = 1.002 \times 10^{-3}\,\mathrm{kg\,m^{-1}\,s^{-1}}$(at 20 °C)
Kinematic Viscosity of Air	$\nu_{\mathrm{air}} = 1.516 \times 10^{-5}\,\mathrm{m^2\,s^{-1}}$(at 20 °C)
Dynamic Viscosity of Air	$\mu_{\mathrm{air}} = 1.825 \times 10^{-5}\,\mathrm{kg\,m^{-1}\,s^{-1}}$(at 20 °C)

List of Symbols

a	axial induction factor	
a'	circumferential induction factor	
$a_\perp$	cross-stream interference factor	
A	projected surface area	m^2
$B(R, x)$	the open ball centered at x with radius R	
c	aerofoil chord	m
$\mathbb{C}$	the set of complex numbers	
C_d	dissipation coefficient	
C_D	drag coefficient	
$C_{D,\text{theo}}$	theoretical drag coefficient	
C_E	coefficient of energy	
C_p	pressure coefficient	
C_P	power coefficient	
$\bar{C}_P$	averaged power coefficient	
C_f	friction coefficient	
C_L	lift coefficient	
$C_{L,s}$	lift coefficient in steady flow	
$C_{L,\text{theo}}$	theoretical lift coefficient	
D	rotor diameter	m
$\vec{e}_r$	radial basis vector in the cylindrical coordinate system	
$\vec{e}_\theta$	polar basis vector in the cylindrical coordinate system	
$\vec{e}_x$	Cartesian basis vector in the x-direction	
$\vec{e}_y$	Cartesian basis vector in the y-direction	
$\vec{e}_z$	Cartesian/Cylindrical basis vector in the z-direction	
$\mathbb{F}$	the unit open interval $(0, 1)$	
F_x	axial force	$\mathrm{N}(\mathrm{kg\,m\,s^{-2}})$
F_y	lateral force	$\mathrm{N}(\mathrm{kg\,m\,s^{-2}})$
F_n	normal force	$\mathrm{N}(\mathrm{kg\,m\,s^{-2}})$
F_t	tangential force	$\mathrm{N}(\mathrm{kg\,m\,s^{-2}})$
F_n^*	non-dimensionalized normal force	
F_t^*	non-dimensionalized tangential force	
G	Green's function	
h	grid size	m
H	shape factor	
k	kinematic energy of turbulent eddies	$\mathrm{J}(\mathrm{kg\,m^2\,s^{-2}})$
$\mathcal{L}$	Laplacian operator (∇^2)	
$\mathcal{L}_{\text{ST}}$	Sturm-Liouville operator	
$\mathbb{N}$	the set of natural numbers	
$\mathbb{N}_+$	the set of positive natural numbers	
p	fluid pressure	$\mathrm{Pa}(\mathrm{kg\,m^{-1}\,s^2})$
P	rotor shaft power	$\mathrm{W}(\mathrm{kg\,m^2\,s^{-1}})$
P_O	system output power	W

P_n	Legendre polynomial of degree n	
P_n^m	associated Legendre polynomial of degree n and order m	
P_w	wind power density	$\mathrm{W\,m^{-2}}$
r	relative distance	
R	rotor radius	m
$\mathbb{R}$	the set of real numbers	
Re	Reynolds number	
s	blade span	m
S_{ij}	(i,j)-component of the averaged strain tensor	$\mathrm{s^{-1}}$
St	Strouhal number	
t	dimensional time	s
T	non-dimensionalized time	
$\vec{u}$	fluid velocity	$\mathrm{m\,s^{-1}}$
u_e	inviscid edge speed	$\mathrm{m\,s^{-1}}$
u_i	i-th component of the velocity vector	$\mathrm{m\,s^{-1}}$
$\vec{u}_{\text{inviscid}}$	invicid flow velocity	$\mathrm{m\,s^{-1}}$
$\vec{u}_{\text{avg}}$	time-averaged velocity	$\mathrm{m\,s^{-1}}$
$\vec{u}'$	fluctuation velocity	$\mathrm{m\,s^{-1}}$
$\vec{u}_\infty$	free-stream velocity	$\mathrm{m\,s^{-1}}$
U	flow speed	$\mathrm{m\,s^{-1}}$
U_∞	free-stream speed	$\mathrm{m\,s^{-1}}$
Y_n^m	complex spherical harmonic basis function of degree n and order m	
α	geometric angle of attack	rad
α_0	static stall angle	rad
β	initial pitch angle	rad
γ	vortex strength	$\mathrm{m\,s^{-1}}$
Γ	Circulation	$\mathrm{m^2\,s^{-1}}$
δt	time step size	s
δ^*	displacement thickness	
δ_{ij}	Kronecker delta	
ϵ	fluid dissipation rate of turbulent eddies	$\mathrm{kg^2\,m^2\,s^{-3}}$
ϵ_{ijk}	Levi-Civita symbol	
θ	azimuthal polar angle	rad
θ^*	shifted azimuthal polar angle $(\theta - \pi/2)$	rad
θ_w	wind incident angle	
κ	body curvature	$\mathrm{m^{-1}}$
μ	fluid dynamic viscosity	$\mathrm{kg\,m^{-1}\,s^{-1}}$
ν	fluid kinematic viscosity	$\mathrm{m^2\,s^{-1}}$
ν_t	turbulent kinematic viscosity	$\mathrm{m^2\,s^{-1}}$
ρ	fluid density	$\mathrm{kg\,m^{-3}}$
σ	rotor solidity	
τ	stress tensor	$\mathrm{kg\,m^{-1}\,s^{-2}}$
ϕ	potential function	$\mathrm{m^2\,s^{-1}}$
ψ	stream function	$\mathrm{m^2\,s^{-1}}$
$\vec{\psi}$	vector stream function	$\mathrm{m^2\,s^{-1}}$
ω	vorticity	$\mathrm{s^{-1}}$
Ω	angular velocity	$\mathrm{rad\,s^{-1}}$
∇	gradient operator	$\mathrm{m^{-1}}$
ϖ	complex velocity in 2D	$\mathrm{m\,s^{-1}}$

This thesis is dedicated to my parents and my beloved wife without whose unwavering supports during the COVID-19 crisis this thesis would not be possible to come to fruition.

Chapter 1

Introduction

1.1 The need for renewable energies

Energy is the most important ingredient in nature for life to flourish on Earth. Since the creation of biology billions of years ago, organisms have thrived on it. The concept of *evolution*, proposed by Darwin, effectively underpins how life depends on the effective use of energy as a way of survival. This ideology persists till this day. Since the early human settlers, the way how energy was harvested has changed considerably. It is known since the discovery of fire, that human had sourced most of their energy demand through solid fuel (dried up vegetation/coal), which provided them with lighting, cooking and other tribal applications and kept them warm in the coldest winters. The progressive development in technologies have allowed more sophisticated approaches for harnessing energy from nature. The constant need to innovate has always been the paramount trend in modern scientific researches; driven by the very same principle that Darwin had advocated in his most celebrated theory -the need for a specie to survive.

Throughout the century, breakthroughs were seen at several fronts in energy research-the invention of atomic bombs gave rise to the possibility of generating energy via the nuclear fission process, although it is perceived by many that nuclear power is the coming trend in energy production. However, doubts on nuclear waste storage, cost overruns and safety assessments were among the primary reasons that prevent such production technique from reaching mainstream. It is wildly believed by the majority that unless there is a breakthrough in science which allows for a controlled nuclear fusion, energy productions through nuclear fission would remain controversial, not lest due to its extreme safety risk, but rather whether it can provide the sustainable character that our society seeks.

The harvesting of chemical fuel from deceased prehistoric organisms (flora and fauna) had proved revolutionary. The term *fossil fuel* was coined to refer to the complex hydrocarbon chemicals of living creatures formed under the intense and persistent heating and pressurised condition deep in the Earth's crust for thousands of millennia, which encompasses *petroleum, natural gas* and *coal*. The inception of fossil fuel has since transformed much of the way of how society evolved in the past decades. 'Fuelled' by their high energy capacity per unit weight, this form of chemical energy remains the unchallengeable source for energy production across the whole spectrum of human activities. In its period of dominance, the oil industry [1] had saw a substantial gain in position in both the political and economic

[1] The use of the term *oil industry* refers to industries that produce fossil fuel.

realm. Notably in the 18th century, which marked the beginning of a new era characterised by high energy consumption and mechanical automation, an *industrial revolution* was born in Britain. Unprecedented, society witnessed the enormous convenience brought about by this new form of energy source; brewing at the same time the capacity for economic and political dominance for countries that had monopoly on these energy sources. But decades of reliance on burning fossil fuel come at a cost insofar its position has come increasingly threatened. One of the by-product of combusting carbon-based chemical is carbon dioxide (CO_2), which is the result of a chemical reaction between the chemicals and oxygen molecules (O_2). It is believed by the majority that this chemical compound is the primary culprit responsible for degrading the efficiency of heat dissipation in the atmosphere, serving effectively as a catalyst that breaks the radiative balance of Earth. According to many established observational data collated and accumulated over the years and across the world, a consistent correlation between CO_2 concentration level and average temperature difference relative to the pre-industrial age was observed. Using extraction techniques such as the ice-core drilling on ice glaciers around the world, a detailed reconstruction of CO_2 mixing ratios, predating back to as far as 50,000 years ago, was possible and provided important clues to the anthropogenic changes of greenhouse gasses in the atmosphere over the course of the millennium. Unsurprisingly, as reported by Etheridge et al. (1996), as much as 25% increase in CO_2 concentration was observed in the post-industrial era. These studies all point to a convergent theme; a need to reduce the dependence on fossil fuel not primarily because of their limited supply (this is certainly not the case since the extraction of shale gas at utility scale began operation in 1821 in the United State), but because of the overwhelming evidence that these anthropogenic changes in CO_2 level and other by-product chemicals have had a detrimental and irreversible effect on the global ecology.

At the turn of the 20th century, the unsustainable character of fossil fuel was gradually being realised. Together with severe pollution events attributed to poorly regulated emission of hazardous particulate matters from factories in the chemical and steel sectors as well as from vehicle exhausts, the development and adaptation of alternative forms of energy production were overwhelmingly in favour in the public opinions. The term *renewable energy* is derived from the desire for an energy source that is virtually limitless in supply (being able to self-replenish in the human scale of time) and is able to avoid the costly environmental impacts that have come to associate with burning fossil fuels. Many forms of renewable conversions have developed and implemented at an accelerated rate around the globe. The most common forms of renewable energy are: solar, wind, tidal, wave, biofuel, hydroelectric and geothermal. These constitute a family of energy supply chains whose position in the global energy supply market is being increasingly dominant in past years. The REN21 (2019) report highlighted this trend (see Figure (1.1)). With a rising installation base and strong initiatives in government policies to pursue *greener* economy, it is seen as a matter of time before renewable could displace fossil fuel as the main source of supply in the coming future.

The ultimate aim of this thesis is to contribute the development of this trend in significant way by concentrating on one particular aspect of the renewable wind energy.

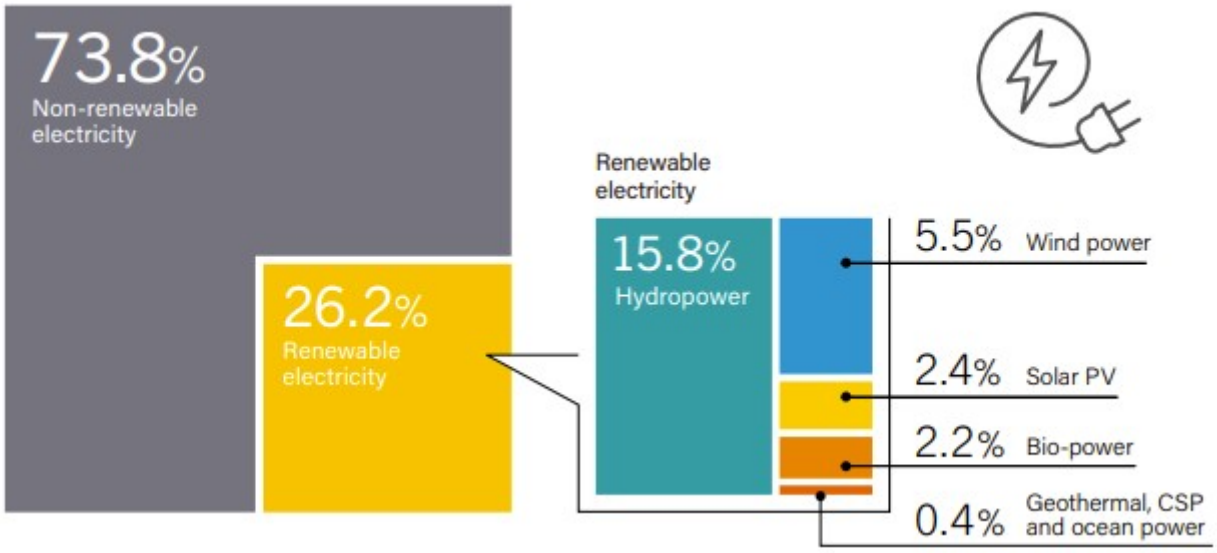


FIGURE 1.1: Estimated renewable energy share in 2019 (REN21, 2019).

1.2 Wind energy

Wind energy refers to the potential energy conversion that uses the movement of air in the atmosphere (kinetic) to convert to other useful forms of energy such as electrical or mechanical. When different parts of the Earth are heated non-uniformly, some regions of air expand and rise, separating distinct zones in space characterised by regions of high pressure and low pressure. Air is transported by virtue of molecular diffusion and pressure difference (high to low). It is this indirect correspondence between solar heating and air movement that merits the name "indirect solar energy". Despite its abundance in nature, it is only recent that wind energy is recognised as the main competitor in large-scale energy production. However the history of wind in civil applications goes back as early as 5000 B.C, where people had learnt to harness the power of wind to help propel boats and build "windmills" for pumping water, grinding wheats and improving food production. At that stage, the use of wind energy was still quite limited in the sense that they had been used in relatively small tasks. It was only in the late 19^{th} century, where people used windmill for generating electricity. But the scope of this ingenuity still confined to localised and often isolated communities.

At the turn of the 20th century, utility-scale wind farms began to popularise in the United State. Being able to generate electricity at the same magnitude as that of the conventional coal-powered station, wind energy was heralded as a serious contender. But in 1940s, due to a combination of low oil prices and the lack of political far-sight, most of the research and development (R& D) in wind turbines were marginalised. This trend continued into the 1970s. Stimulated by the oil crisis in 1974, a renewed focus on renewable energy was imminent, which marked the official re-entry into the energy sector by renewable energies. Although in recent years investments in the wind energy sector have been relatively dwindled in the United State, but elsewhere in Europe and Central Asia, especially in countries such as Germany and China, interests in turbine technologies and wind-farm optimization remain relatively high as is evident in the REN21 report.

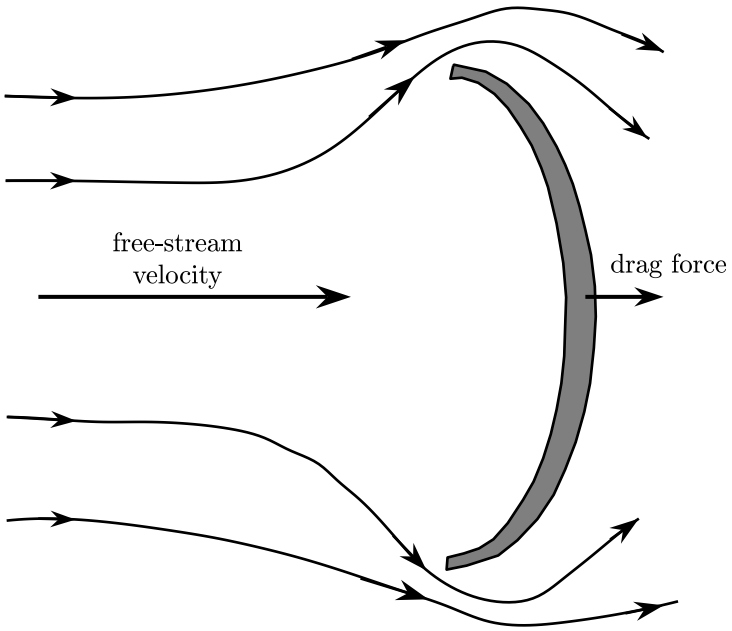


FIGURE 1.2: Schematic of the operation of a *drag device*

1.3 Wind in civil applications

The earliest use of wind is in the propagation of ancient sailing boats. This is the earliest example of what is known as the *drag-device*, which operates on the principle that the device extracts/dissipates power from the wind by the velocity difference in the windward/leeward direction. A typical drag-device is shown in Figure (1.2). To facilitate the discussion, it is necessary to give the definition of the term *drag*. In this thesis, the *drag* is referring to a force that is aligned parallel to the local flow velocity as is seen from the frame of reference of the device. Therefore the term *drag device* simply refers to a mechanism whereby power is generated/dissipated by this force.

The magnitude of power that can be extracted is dependent on the device's projected surface area and the relative speed. In fact, if P is the extracted power, U_∞ is the wind speed in some direction and U is the speed of the device translating in the same direction as the wind, then a simple relationship exists between the state variables:

$$P = \frac{1}{2}\rho \left(U_\infty - U\right)^2 A C_D U, \tag{1.3.1}$$

where ρ, A, C_D are the fluid density, projected surface area and the coefficient of drag, respectively. For this reason, early drag-devices tend to increase its projected area in order to increase the power of extraction. A typical example for this design philosophy is the *Chinese Junk*, which is an ancient Chinese sailing ship that has a wide span of rectangular sail, which is designed to capture as much power as possible (see Figure (1.3a)). Whilst this design principle had existed for many years, but drag-devices for power generations are generally considered inefficient, especially when it comes to wind turbine technology. This is because in typical engineering applications the value of the drag coefficient C_D can be quite small (especially in high speed flow around streamlined body), whereby the device has to either increase its

(A) Chinese junk ship

(B) A Typhoon deploying parachute

FIGURE 1.3: Examples of drag devices.

projected area or increase its translational speed in order to compensate for it. But adjusting to those changes would sometimes result in expensive engineering cost and challenges. Notwithstanding the fact that U can only be increased to an optimum value before maximum is reached. The maximum power for such device is then limited to 14.8% for a given drag coefficient.

Despite their low efficiency in generating power, dissipative drag devices are commonly deployed in applications such as sailing, parachuting and aerodynamic braking as seen in high speed landing of aircrafts due to their simplicity in projecting a large surface area in a short span of time (Figure (1.3b)) and their favourable dependence on speed. Using Eq.(1.3.1), for a dissipative drag-device translating at speed U in a medium of stationary fluid with density ρ the rate of losing kinetic energy is then:

$$P_{\text{loss}} = -2\rho U^3 A C_D. \tag{1.3.2}$$

Clearly, for a simple dissipative drag-device, the power loss varies to the cube of the speed, which amounts to an efficient way of reducing excess speed in conditions where this reduction is required in short span of time as in the case of a high-speed fighter aircraft needing short distance landing and recovery. Whilst this is good for this type of applications, but for other applications where power is to be generated (rather than dissipated), this mechanism is not sufficient.

In contrast, however, a *lifting* device uses the aerodynamic property of fluid to generate a lift force, which, by definition, describes a force that is aligned in the direction perpendicular to the local flow direction (see Figure (1.4)). Indeed, studies into lifting-driven devices were the main contributions that gave rise to the aeronautical industry (including fixed-wing and rotor-type aircraft) and many modern wind turbines were indeed designed to be propelled by those lifting devices. A comparison study by Wilson et al. (1976) suggests the efficiency of a lift device in power generation can be as great as 300 times of that of a drag device for a given square meter of area. Whence it is more common to find, for example, turbo-machineries and utility-scaled wind turbines be mainly based on this kind of principle for power extraction. However, realistic engineering systems seldom produce a single type of force, but a combination of lift and drag.

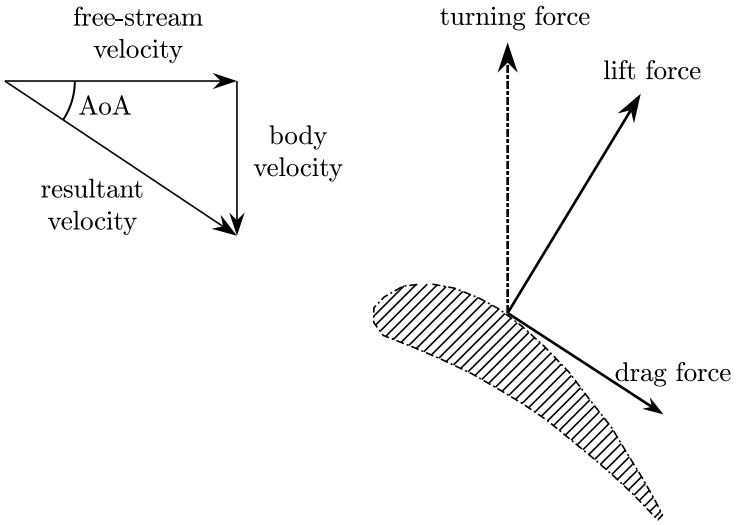


FIGURE 1.4: Schematic of the operation of a typical lifting-device. The turning force is produced via a combination of lift and drag forces, which are functions of the angle of attack (AoA). Typically, the lift force produced by the lifting-device is a magnitude larger than the generated drag force.

1.4 Wind mills and wind turbines

In order to provide clarity to the concept of *wind turbine*, it is necessary to distinguish the difference between a *wind mill* and a *wind turbine*. The central difference lies in the applications of which these machines are intended for. Traditionally, wind mills are placed in the agricultural context in which they are to be used for grinding food crops, pumping water and perhaps small-scale electricity generation. Wind turbine, on the other hand, is exclusively used for electricity generation. Whilst sometimes these two terms are used interchangeably, but it is important to bear in mind that such difference exists.

All wind turbines operate on two separate principles from which kinetic energies of air molecules are extracted. They can be broadly classified as *lift*-based, or *drag*-based. A very common wind turbine topology is what is known as the *horizontal axis wind turbine* (hereafter abbreviate as *HAWT*). These consist of finite number of blades attached to a rotor, which is mounted to a tower. The rotor shaft is connected to either a gearbox or the generator. The rotor shaft is placed in a horizontal plane relative to the blades, thus explaining the rationale of the name. HAWTs extract energy by converting the mechanical torque impacted by the air particles on the blades to drive the shaft, which turns the generator to produce an electrical current, which can then be stored in batteries or distributed to the national grid. The induced mechanical torque is mainly derived from the lift force generated by the fluid flow (see Figure (1.4)). In order to capture a stronger wind, taken account of the natural atmospheric boundary layer of wind speed variation, the central unit encompassing the rotor and the generator housed in the nacelle (also known as the *hub*) is normally placed at a height which scales in accordance to the rated power output, so for example, the *DOE/NASA Mod-5B* prototype built in 1988 has a rotor span 97.5 m at hub height 61 m which operated at a power rating 3.2 MW. Clearly, higher wind speed

could be achieved by placing the hub at a higher elevation, but this would entail additional engineering costs. Thus, most utility-scale HAWTs seem to compromise at a certain height (see Table 1.1 for the different HAWTs in commercial use) - optimising the cost function that balances the cost of operation/installation capitals versus the power generation.

Name of turbine	hub height (m)	rotor span (m)	rated power (kW)
DOE/NASA MOD-5B (HAWT)	61	97.5	3200
GE 1.5s (HAWT)	64.7	70.5	1500
DOE/SNL (VAWT)	34	17.4[a]	500

TABLE 1.1: Some examples of utility-scale HAWTs and VAWTs.

[a]The rotor span of the VAWT is calculated by the formula $\sqrt{S/\pi}$, where S is the sweep area.

HAWTs have been the subject of intense interest in the engineering community. Indeed, since the inception of wind mills, this form of turbine topology has existed long before other wind turbine designs had emerged, as both types of machines essentially share the same principle. Whilst this topology has found commercial success, but it is hard to ignore some of its most severe limitations. First of all, it is well known that the biggest disadvantage of such a machine is the requirement of land and its inability to scale. Specifically, in order to operate at a nominal level and to provide enough power, the rotor radius of a HAWT ranges in a typical length scale comparable to the hub height. During a single passage of the blades, the swept area of the rotor can reach more than 11309.7 m^2. Additionally, in compensating for wake losses, downstream turbines are typically placed between 7 to 10 rotor diameters apart, which signals a significant amount of land requirement. Any effort to scale down HAWTs will prove futile as doing so could quickly lead to a significant loss in power efficiency and not to mention the logistical and environmental difficulties (blockage of birds' migration route, noise, electromagnetic interferences in communication stations) in installing one in an urban setting as well as the complex engineering challenges to optimise the turbine configuration for adapting to the complex free-stream profile such as the inclusion of yawn control, variable-pitch mechanism, etc. As a result, it is necessary to construct *wind farm* away from urban areas, where the demand for land is less restrictive and environmental problems can be relatively easy to overcome. However, it should be noted that this might not always be the optimal solution as it is rather expensive to reconnect new grid lines if the location of the *wind farm* happens to be outside of the vicinity of the national grid network.

Little is known in the public domain that there is another type of wind turbine, which goes by the acronym VAWT (Vertical Axis Wind Turbine). Similar to HAWT, these machines use the same physics to extract energy but differ in the geometric orientation of the rotor shaft. Instead of placing in a horizontal plane relative to the blade, the shaft of a VAWT is fixed permanently in a vertical position. Thus, all electricity-generating components are located at ground level; offering the unparalleled advantage of accessing the service units more easily compared to HAWTs whereby service-men would have to conduct maintenance work inside the nacelle. Despite the fact that the concept of a vertical-axis wind mill, from which VAWTs are derived, have been in existence equally as long as the horizontal ones, interests in utility-scale VAWT have only been recent compared to the long history of Horizontal wind turbines. Partly, this is because these machines are perceived by many that their efficiencies are far inferior to their horizontal counterpart. But as Paraschivoiu

(A) GE 1.5 MW 1.5s HAWT

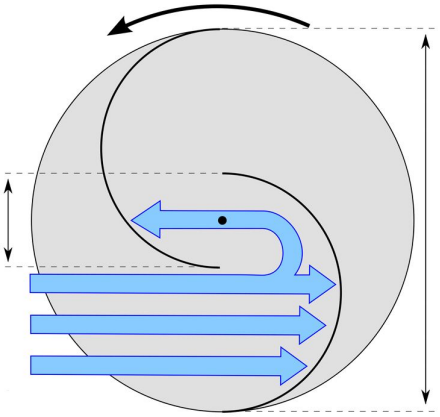
(B) Main components of a wind turbine

FIGURE 1.5: (A) shows the GE (General Electric) 1.5 MW HAWT. (B) illustrates the major components constituting a wind turbine.

(2002) has noted, this is not quite the argument as he continues to argue that the reason that propelled HAWT to where it is now is because of the immerse volume of researches that have dedicated for the subject. The same can not be said regarding VAWTs. However, with the emergence of new technologies and state of the art researches on their aerodynamics, VAWTs have shown that they too can compare and even exceed some aspects of HAWTs in certain areas, which make them especially attractive.

VAWTs come in various forms. Unlike their horizontal brethren, VAWTs can be propelled either by a drag-device or a lift-device. For cases in which utility-scale is sought, most VAWTs were lift-based. Perhaps the earliest and most studied VAWT is of the *Savonius* type (see Figure (1.6a) and Figure (1.6b)). Due to their simplicity in the manufacturing process, which can be made by cutting a barrel in half and inverting the other half and welding them together, this design is popular in areas where access to electricity is difficult. Unfortunately, being predominantly a drag-based turbine, its efficiency is severely inhibited - rendering it rather limited in terms of power output. Other VAWTs designs tried to mitigate this shortcoming by proposing a lift-based mechanism. These designs made uses of the fully developed theory of lift-based aerofoils whose performance characteristics had been fully understood both theoretically and experimentally. Perhaps, the most interesting example of a lift-based turbine is what is known as the curved-blade Darrieus turbine (Figure (1.6c)). The blade of these machines is necessarily curved and arranged in configuration known as the ideal *Troposkien* curve, which is a curve formed naturally by

(A) Savonius rotor



(B) Schematic of the Savonius rotor

(C) Sandia 17m Darrieus rotor

(D) 4Navitas 5-bladed H-type rotor

FIGURE 1.6: Various VAWT designs in the market.

an inextensible string rotating at constant angular velocity. Thus at any given instance of time, the shape of the blade is found by balancing the gravitational force, centrifugal force and the tension in the string in the *normal* and its perpendicular direction. Paraschivoiu (2002) gave an explicit derivation of the equation for the blade shape. Initially, this curve was chosen for its structural properties (minimum structural stress) but because it bears close resemblance to the parabola or the *catenary*[2], it is found more practical, in terms of the manufacturing process and aerodynamic analysis to approximate the ideal Troposkien curve by means of a parabola, catenary or a combination of straight/circular arc.

As the concept of VAWT becomes more mainstream, many researchers have

[2] a *catenary* is a curve formed naturally by a piece of flexible string whose ends are fixed at two points on the same horizontal plane. It can be found by solving the force balance between the gravitational force and the intrinsic tension

come up with ingenious designs which hope to improve the general performance of a vertically axis wind turbines. But from a business point of view, those designs are yet to prove successful as they tend to be overly complicated to be implemented at a commercial level for various reasons. Thus a more simplified solution is to be sought. A general trend is observed in recent years which seems to congregate focus to straight-bladed Darrieus rotors known by various name such as *the Giromill* and *H-type*. Those carry the concept of the original curved-bladed Darrieus rotor, but the blades are now being replaced by straight vertical sections that consist of a single type of aerofoil supported by spreaders. One such example is the *4Navitas*' new 55-kW straight-bladed Darrieus with a range of configurations suited for various wind speed conditions (Figure (1.6d)). The biggest advantage of this design which holds over its curved-bladed variant is its simplicity in production and aerodynamic simulations as it avoids the complicated structure of having tapered section of the blade with different aerofoil cross sectionals. However, it should be noted that since this design deviates from the ideal Troposkien curve, there is a tendency for a concentrated centrifugal stress to be built up in the blade as it rotates. This problem is partially resolved by using new material and structural design solution.

1.5 Wind turbine parameters

Studies into turbine designs have revealed several important parameters that describe the general characteristics of their performance. Depending on the specific role that the turbine is designed for, the trend is to optimize certain cost function which chiefly scales on those primary variables. Spera (2009) has shown that the most important cost function to maximize is the *annual energy output*, which is determined from the energy of the wind and the efficiency of its working components. Thus, by defining the *coefficient of energy* C_E as:

$$C_E = \frac{\int_{\text{year}} \mathrm{P}_O \, dt}{\int_A \int_{\text{year}} \mathrm{P}_W \, dt \, dA}, \tag{1.5.1}$$

where

$$\begin{aligned} C_E &= \text{coefficient of energy,} \\ \mathrm{P}_O &= \text{system output power (W),} \\ \mathrm{P}_W &= \text{wind power density (W/m}^2\text{),} \\ A &= \text{projected swept area of turbine rotor in the vertical plane (m}^2\text{),} \\ t &= \text{time (h).} \end{aligned}$$

Efforts should be made to optimize the quantities P_O and P_W which intrinsically relate to turbine design specification and site selection, respectively. For a comprehensive review of the importance of these variables, the reader is referred to the book by Spera (2009).

Typically, the site selection determines the various parameters existing in the term P_W. The term P_O, however, is determined from the turbine itself, which relates to key primary variables such as the type of aerofoil employed in the cross section of the wings, the nominal tip-speed ratio, the blade solidity, swept area of the turbine, Reynolds number cetera. The definitions of some of these terms will be discussed in this section as well as their important roles in the determination of the annual energy production.

Perhaps, the most important parameter in determining the performance of a particular turbine is its *power coefficient*, C_P, which specifies the amount of power that can be extracted from wind source. Since the power coefficient is a function of the tip-speed ratio (TSR), the performance gauge of a particular turbine often expresses in terms of the power-versus-TSR curve or simply as *power curve*.

For a H-type Darrieus VAWT operating at a constant angular velocity, the tip-speed-ratio TSR is found by:

$$\text{TSR} = \frac{R\Omega}{|\vec{u}_\infty|} \tag{1.5.2}$$

where $R, \Omega, \vec{u}_\infty$ are the rotor radius, angular velocity and free-stream velocity respectively. In addition, the solidity of the rotor also affects the performance characteristic of the turbine, and it is defined in this thesis as

$$\sigma = \frac{Bc}{R} \tag{1.5.3}$$

where B, c, R are the number of blades, chord length and rotor radius respectively.

Many experimental and theoretical investigations have been done in the past to examine the effects of varying the said parameters. Although there is no universal consensus that the turbine is to be built in one optimal configuration, there is a large variation in the number of design parameters that could influence their specific choice. Paraschivoiu (2002) outlined a common dilemma faced by turbine designers. At a given σ, the power coefficient obtains a maximum as the tip-speed-ratio varies, which implies the turbine should ideally be configured to work within this range of tip-speed-ratios. However, in reality, due to various reasons such as in low wind or structural limitation, it is not always the case that this operating condition can be achieved. In those circumstances it is a matter of urgency to inspect how the power curve behaves in regions that are outside of the local maximum. It can be shown that a compromise has to be made whether manufacturers favour a σ that produces the best maximum power coefficient but quickly decreases as TSR deviates from the local maximum or should they adopt a σ that maintains a relatively constant value throughout a relatively large range of TSR; thereby extending the range of optimal operating conditions. It is no simple matter to answer this sort of questions as it also depends on the choice of site on which the turbine is to be built and the meteorological profile of the location as well as other environmental and engineering considerations.

The Reynolds number Re, though not shown explicitly, is critical in aerodynamic prediction models. It is important for two reasons. First, it governs the physics foundation on which the prediction models are based on. It can be argued that any assumptions made in the model need to be consistent with the physics, otherwise it will quickly lose its predictive value. Secondly, the Reynolds number plays a role in marking the critical point at which physical processes might undergo transition which is very common in aerodynamic. The transition from laminar and turbulence has deep implication to the validity of certain flow prediction methods. Early simple prediction methods completely avoided this transition, resulted in a limited scope of applicability (i.e. based on steady assumption and avoided spatial turbulence modelling). However, as will be shown later, large-scale unsteady turbulence structures are important in resolving a detailed aerodynamic picture of the flow field, which also determine the aerodynamic performance of not just the turbine unit but also the wind-farm performance in realistic operational conditions.

1.6 Recent research into VAWT wind-farms

It has been recently identified that VAWT wind-farms could potentially offer a substantial improvement to the power density (per unit of land surface area) for a counter-rotating turbine array (Dabiri, 2011) compared to HAWT wind farm. This result motivates the needs to study the aerodynamic properties of the mutual interaction between turbines as a function of the separation distance. Several computational studies have been done in the past on the effect of pair-wise placement of the VAWT rotors (Feng et al., 2014; Parneix et al., 2016; Zanforlin and Nishino, 2016). The collective efforts of the past works have indicated the positive effect of placing the VAWT rotors close together and thus taking advantages of the increased flow speed in the blockage region. A range of approaches with varying degrees of resolution have been developed for this purpose. For example, Feng et al. (2014) used a combination of free-vortex model and the Jensen wake model to simulate the effect of multiple rotors in a farm. The Jensen wake model is an empirical simplification to the turbine wake that models the effect of interacting turbines via a simple superposition law based on momentum conservation (Jensen, 1983). The expansion of the wake is empirically modelled by a linear rate, which results in the limited applicability of the method to study the detailed wake structure. On the other hand, Zanforlin and Nishino (2016) applied a computational fluid dynamic (CFD) approach to study the detailed interaction for a range of wind incident angles for a pair of closely packed VAWTs. Their results show, at certain incident angles, enhancement to the power outputs is achieved owing to the wake suppression of the upstream rotor. For a lateral placement of the rotors, the study of Parneix et al. (2016) points to a similar trend.

Although the CFD is the most accurate approach to date, but the expensive nature of the method prevents it from being used as a fast design tool due to the complicated requirement of setting up the problems and the large amount of computing resources to obtain a converged solution (typically requiring a high performance computing (HPC) platform). Similarly, other engineering methods in the literature are mostly limited to small scale problems in which the quadratic scaling is still tractable. However, when extending to large problems, those methods proved to be too inefficient to be a viable alternative. In this context, one resolution is to develop a more efficient algorithm that takes advantage of the recent computing trend. The aim of this thesis is to develop an engineering approach coupled with the use of general purpose graphics processing units (GPGPU) to accelerate the calculation so as to achieve a fast solution time obtainable via a local workstation. In this way, the technique developed can be used in optimization problems in a wide range of wind turbine applications. Specifically, the use of a fast technique is an absolute necessity if one is to optimize the wind-farm configuration by exploring the large parametric space. The developed codes in this thesis aims to fulfil these requirements.

Chapter 2

Literature review

2.1 Turbine aerodynamic performance prediction methods

2.1.1 The classical Blade-Element-Momentum theory

In aerodynamic load analysis of wind turbines, one popular strategy is the Blade Element Momentum theory developed by Glauert (1947). The approach approximates the rotor of a HAWT as a semi-permeable surface whose role is to induce an aerodynamic force to the fluid flow passing the rotor plane. As a result, the axial velocity of the free-stream is slowed down in the near wake. Power is thus extracted by virtue of momentum conservation. A key question is then to address how much the fluid has to slow down near the rotor plane in order to account for the correct physics?

The induction factor (a) is defined as the normalized axial velocity difference between the upstream and downstream part of the rotor. Assuming a is fixed, it is found that the power coefficient is related to the induction factor as follows (Hansen, 2008):

$$C_P = \frac{P}{\frac{1}{2}\rho|\vec{U}_\infty|^3 A} = 4a\left(1-a\right)^2. \tag{2.1.1}$$

By differentiating with respect to the axial induction factor a, it is easy to see that this occurs at $a = 1/3$, with which the maximum value of C_P is obtained at 16/27 - corresponding to the classical Betz limit. Classical blade-element momentum theory draws on the assumption that the rotor is approximated by a rotor disk which contains the assertion that it is consisting of infinite number of blades, whilst this assumption is valid in the range of value of a for which $a < 0.4$, but it is far from what is observed in experiments. The discrepancies can be attributed to two causes: firstly, since the blade is modelled as a form of an actuator disk, a correction has to be applied in order to account for finite number of blades. Prandtl's tip-loss factor correctly accounts for this deficiency. Secondly, because the integrated force on the porous rotor surface scales with the axial induction factor and that the axial wake lost increases as a increases (corresponding to heavily loaded blades), this creates a distinct shear layer on the boundary of the stream-tube for which fluid outside the tube is inevitably transported to the wake by physical processes such as turbulent eddy mixing and momentum diffusion, which are strong unsteady effects. Extensive data fitting with experimental results allows an empirical correction to be made in regards to this unsteady effect. While the unsteady correction is empirical in nature, but for many practical applications and especially in turbine design space, it is deemed sufficient to yield a reasonable solution. Typically, the blade element momentum approach precedes an iterative scheme. Discussed by Hansen (2008), the axial and circumferential induction factors (a & a' respectively) are first estimated,

from which the angle of attack is determined. Proceeded by the normal and tangent force coefficients, the corrected axial and circumferential induction factors are obtained by the momentum conservation equations. i.e.

$$a = \frac{1}{\frac{4F\sin^2\phi}{\sigma C_n} + 1}, \tag{2.1.2}$$

and

$$a' = \frac{1}{\frac{4F\sin\phi\cos\phi}{\sigma C_t} - 1} \tag{2.1.3}$$

where F, ϕ, C_n and C_t are the Prandtl's tip-loss factor, the angle of the relative velocity with respect to the axis of translation, coefficient of the normal force and the coefficient of the tangent force, respectively.

The blade-element momentum method has been similarly developed for the VAWT. The implementation of the approach for any degree of practical use, however, is a lot more difficult thanks in part to the increased complexity of the flow field past such rotor. As a first approximation, Glauert's element theory can be applied to a single stream-tube past the entirety of the rotor plane. The induced velocity is thus assumed constant and emerges as part of the solution. This method subjects the same limitation as in the case of the classical blade element applied to HAWT. Therefore, it is not unusual to find that the theoretical result from the early *single-streamtube* approach to deviate quite substantially from test data. However, such method is not to be discredited as it has shown time over time the capacity to correctly predict the maximum power coefficient for a given σ. Strickland (1975) developed the idea further by splitting the rotor plane into a collection of streamtubes; each with a different induction velocities. This results a stream-wise velocity distribution with respect to the two spatial coordinates. The approach, coined *multiple streamtube*, is similar in nature to the blade-element momentum theory in that the axial induction factor is calculated iteratively for each streamtube. The `DART` computer program, developed from Strickland's multiple streamtube approach, significantly improves the predication performance over the single stream tube approach. However, care should be exercised in applications where detailed flow field is sought. The method relies on the time-average formulation of the momentum equation and therefore the instantaneous information of the flow field are lost during the force computations as Strickland noted himself, namely:

> While this approach is somewhat elegant in its simplicity and predicts overall performance rather well for lightly loaded blades, it is incapable of adequately predicting information which requires a more precise knowledge of wind velocity variations across the rotor (Strickland, 1975).

2.1.2 Double Multiple Streamtube

The close match between experimental data and the streamtube model, together with a substantial reduction in execution time compared to CFD, has made the approach quite favourable. A unique challenge that is present to all aerodynamic modelling of VAWT is the complex interaction between the wake and its blades during the downstream passage, early multiple streamtube models were unable to take into account of this effect as they tend to assume a constancy of the induced velocity throughout the interior of the rotor. Although experimental data show good agreement for low tip-speed ratio, those methods generally break down for a substantial range of tip-speed ratios.

A natural progression from the multiple-streamtube is to model the VAWT as two actuator disks in tandem. Coined *double multiple streamtube* (DMST), this type of approach, developed by Paraschivoiu (2002), has shown improvements in performance predication. Moreover, the code `CARDAAV` based on the double streamtube model has acquired the capacity to predict the load in different flow conditions including, shear flow, dynamic stalling, tower shadowing, etc. The original `CARDAAV` has since evolved to many forms such as `CARDAAX` and `CARDAAS-1D/-3D` (Paraschivoiu et al., 2009).

Although `CARDAAV` is computationally inexpensive, but it suffers the same drawback like many of the streamtube methods discussed earlier. Much of the criticism has been focused in the non-expansiveness of the streamtube geometry. Although the conventional DMST agrees well with experimental data as a whole, but agreement is limited to certain part of the azimuthal. A detailed description, drawing from the insights gained from experiment and other more accurate measures, reveals that the wake does expand as fluid flowing past the rotor plane due to the diffusion effect. Further, Madsen (1982) has warned of the inherent limitation in modeling VAWT as actuator disks. This modelling assumption of which most streamtube method are based on does not allow the wake to deform. As such, Ferreira (2009) has attributed the deficiency of such assumption to the lack of expansion of the streamtube. A more refined streamtube method is then to model an interference factor that is aligned in the perpendicular direction of the flow, this is relevant particularly when significant misalignments are observed between the free-stream and the wake velocity in the near wake region. Madsen (1982) formulated the expansion in the form of the an actuator cylinder, in which the geometry of the rotor is modelled as a porous surface formed in the shape of a circle. The purpose of the porous cylinder is to exert a radial force to the fluid. The method however requires the implementation of CFD simulation, therefore the method outlined by Madsen does not strictly fit in with the description of the streamtube class. Ferreira (2009) proposed the cross stream interference factor be formulated in terms of the doublet source whose role is to induce a velocity potential given by:

$$\phi_{\text{doublet}} = -\frac{\vec{\mu} \cdot \vec{r}}{4\pi r^3} \tag{2.1.4}$$

where $\vec{\mu}$ is the doublet strength vector and $r = |\vec{r}|$ is the radial length between the source and the point of interest. The cross-stream interference factor $a_\perp$ is determined from the induction velocity of the doublet-source panels (Katz and Plotkin, 2001). With this coupling between the conventional DMST and panel approach, Ferreira has seen a substantial gain in accuracy in terms of the azimuthal variation of the normal and tangential force distributions. However, this empirical correction, whilst accurate in predicting the variations of the turbine parameters, suffers the same deficiency as other streamtubes. Therefore, it is incapable of predicting the wake characteristics.

2.2 Computational fluid dynamics (CFD)

As the advent of more computational prowess promised by the Moores' law, high fidelity simulations now seem within grasp using mainstream workstations with moderate computing set-up. One of the key convenience brought about by this exponential increase in computing power is the possibility of simulating the fluid

dynamic directly from the governing equations. This section reviews some of the methodologies that are currently being employed in most commercial CFD codes.

2.2.1 The governing equations

The governing equations of fluid dynamics exist in various forms. Depending on the significance of the various physical processes that take place in the actual problem, the governing equations of fluid dynamics change accordingly. In low speed dynamic, a concept that will be made clear in subsequent discussion, the most prevailing form of the governing equations is based on the infinitesimal balance between inertial and internal stresses, which could be interpreted as a particular case of Newton's 2nd law. Since it is never the intention of this thesis to provide a comprehensive detail on the mathematics, the reader is referred to the book by Batchelor (1967) for detail. The full set of equations are described relative to a Cartesian frame as follows:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + f_x + \frac{2}{\rho}\vec{e}_x \cdot \nabla \cdot \tau, \tag{2.2.1a}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + f_y + \frac{2}{\rho}\vec{e}_y \cdot \nabla \cdot \tau, \tag{2.2.1b}$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} + f_x + \frac{2}{\rho}\vec{e}_z \cdot \nabla \cdot \tau, \tag{2.2.1c}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0. \tag{2.2.1d}$$

Here, Eq.(2.2.1a), Eq.(2.2.1b), Eq.(2.2.1c) are referred to as the momentum equations and Eq.(2.2.1d) is the continuity equation from which the variables u, v, w, p completely characterize the solution of the flow problem, and $f_i, \vec{e}_i,\ i = x, y, z$ are the volume force in the 3 directions and the unit vectors in each of the direction respectively. ∇, τ are the typical Laplacian operator and the tangent stress tensor. Here, the assumption of *low speed* is particularly important for the fact that the density ρ is assumed constant and that any thermodynamic processes are essentially in equilibrium. This ensures the microscopic mean path of molecules is much larger compared to the length-scale of thermal diffusivity, hence heat exchange between layers of fluid molecules can be essentially ignored. It should be noted that at high-speed applications (Mach number of the flow approaches unity or above) or situations in which heat-exchanges are dominant, the assumption of constant density and the equilibrium nature of thermal exchange breaks down. This is evident in the shock-wave produced on an aircraft undergoing supersonic transition as the shock-wave serves to alter the density across the shock line quite dramatically and induce a significant heat production or the simple observation of the induced fluid current generated in a boiler.

When the fluid is Newtonian, a simple law exists between the stress and the strain. By assuming that the stress is a linear function of the strain, τ then becomes:

$$\tau = \frac{\mu}{2}\left(\nabla\vec{u} + \nabla^T\vec{u}\right), \tag{2.2.2}$$

where μ is the dynamic viscosity. Substituting Eq.(2.2.2) to Eq.(2.2.1) and utilizing the continuity equation Eq.(2.2.1d), the *Navier-Stokes* equations are recovered,

namely

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\frac{1}{\rho}\nabla p + \vec{f} + \nu \nabla^2 \vec{u}. \tag{2.2.3}$$

Eq.(2.2.3) and Eq.(2.2.1d) form the basis for many fluid problems. Furthermore, by changing each physical variable to its non-dimensionalized equivalence, the Navier-Stokes equations can be reformulated as:

$$\left(\frac{L}{TU}\right)\frac{\partial \vec{u}^*}{\partial t^*} + \vec{u}^* \cdot \nabla^* \vec{u}^* = -\left(\frac{p_0}{\rho U^2}\right)\nabla^* p^* + \left(\frac{L f_0}{U^2}\right)\vec{f}^* + \left(\frac{\nu}{UL}\right)\nabla^{*2}\vec{u}^*. \tag{2.2.4}$$

where L, T, U, p_0, f_0 are the characteristic values of the flow variables $\vec{x}, t, \vec{u}, p, \vec{f}$, respectively. The importance of this formulation is its ability to inform modelers the relative magnitude of the terms in the equation. For example

$$St = \frac{\omega L}{U}, \tag{2.2.5}$$

with $\omega = 1/T$, is the *Strouhal* number, which is used to indicate the strength of time dependent phenomena. Perhaps, the most important quantity that merits special attention is the *Reynolds* number defined by:

$$Re = \frac{UL}{\nu}, \tag{2.2.6}$$

which quantifies the importance of viscous effect on the fluid. In engineering applications where advection is typically the dominant process, it is possible to neglect the viscous diffusion term altogether - resulting in the Euler's equations. Although viscosity tends to be unimportant in the majority of the fluid domain, but it does impact on the accuracy of the solution in regions close to solid walls. Part of the difficulty is that by neglecting the viscous term, the 'no-slip' boundary conditions cannot be satisfied by the Euler solution. One way to resolve this matter is to employ an analytical technique, which matches the solutions asymptotically at each regions (Wolles, 1968).

2.2.2 Direct numerical simulation (DNS)

As a first attempt, one might incline to solve Eq.(2.2.1) by brute force. Together with the 'no-slip' boundary conditions and initial condition, a numerical solution, subjecting to those constraints, is theoretically possible to obtain. Direct Numerical Simulation (DNS) is the most accurate method to date, which attempts to resolve all of the length-scales and time-scale existing in the fluid-flow by producing an adequate mesh fit for those requirements. The *Kolmogorov* scale is the smallest relevant spatial scale with which the mesh needs to resolve. Defined as

$$\eta = \left(\frac{\nu^3}{\epsilon}\right)^{1/4}, \tag{2.2.7}$$

this serves as the spatial range in which the dissipation of turbulence kinetic energy (representing by the term ϵ) is balanced by the viscous diffusion. If the mesh resolution is to be characterized by a spatial step-length h, then $h \leq \eta$ in order to simulate this dissipation phenomenon. Furthermore, if the number of nodal points N along a

mesh dimension scales with $1/h$, it can be shown that N scales with

$$N \propto Re^{3/4}. \tag{2.2.8}$$

For a 3-dimensional DNS simulation, a minimum number of grid nodes then proportionates with $Re^{9/4}$ for each time-step (Pope, 2000). Although it requires no empirical model for turbulence modelling, but the expensive dependence on the Reynolds number of the grid nodes hinders it from any practical use in the wind industry. A direct DNS approach is yet to be carried out by researchers working in the wind energy and it will probably remain to be untouchable in the near future for its expensive cost.

2.2.3 Turbulence modelling

The unfortunate case of a DNS simulation necessitates the means to reduce the costly dependence on the Reynolds number while at the same time produce solutions with reasonable accuracy. Through experimental observations, one of the principal characteristic associated with turbulent flow is the inherent unpredictability of the flow field both in the temporal and spatial domain, which seemed to produce small scale stochastic fluctuations that quickly average to zero. It is a question of the modellers whether those temporal or spatial fluctuations are of particular importance. In most engineering applications, these small-scale fluctuations tend to play a smaller but still significant role. Therefore efforts were expended in the past to pursue an understanding of how such fluctuations affect the solution. This pursuit has given birth to several well-known turbulence models, all of which are empirical in nature. They can be broadly classified into two groups - RANS and LES.

The acronym RANS is derived from the Reynolds averaging process of the Navier Stokes equations. This is formulated by decoupling the flow velocity $\vec{u}$ into an averaged component $\vec{u}_{\text{avg}}$ and a fluctuating component $\vec{u}'$, i.e.

$$\vec{u} = \vec{u}_{\text{avg}} + \vec{u}'. \tag{2.2.9}$$

The *Reynolds* averaging has the property that $\langle \vec{u}' \rangle = 0$, where $\langle \cdot \rangle$ denotes the Reynolds averaging operation. Upon applying the averaging operation to Eq.(2.2.1), the RANS equations are derived:

$$\frac{\partial \vec{u}_{\text{avg}}}{\partial t} + \vec{u}_{\text{avg}} \cdot \nabla \vec{u}_{\text{avg}} + \left\langle \vec{u}' \cdot \nabla \vec{u}' \right\rangle = -\frac{1}{\rho} \nabla \langle p \rangle + \left\langle \vec{f} \right\rangle + \nabla^2 \vec{u}_{\text{avg}}, \tag{2.2.10}$$

where the linearity and commutativity of the averaging operator have been implicitly assumed. It should be stressed that the fluctuation of the flow field is regarded as random and chaotic, therefore it is not necessary to derive information about $\vec{u}'$, but to analyse the effects of fluctuation have on the mean flow field $\vec{u}_{\text{avg}}$. This is captured by the terms $\langle \vec{u}' \cdot \nabla \vec{u}' \rangle$. The essence of turbulence modelling using RANS is to effectively model such terms. Moreover, by applying the vector identity:

$$A_j \partial_j A_i = \partial_j \left(A_j A_i \right) - A_i \partial_j \left(A_j \right),$$

with $A = \vec{u}'$, together with the incompressibility of Eq.(2.2.1d), it is possible to express

$$\left\langle \vec{u}' \cdot \vec{u}' \right\rangle = \nabla \cdot R,$$

where R is a 2^{nd} order tensor, which is known as the Reynold's stress tensor whose component is given by $R_{ij} = \left\langle \vec{u}_{\text{fluc},i} \vec{u}_{\text{fluc},j} \right\rangle$. The subsequent subsections set out to explore the different physical rationales in modelling the Reynold stress tensor R.

Turbulence models with the implementation of the RANS assumption are subdivided by the number of additional closure equations in the modelling process. In addition, the nature of the turbulence is unknown a priori and therefore must be assumed in the problem. Drawing from the similarity between the stress and the strain rate tensor, a common practice in turbulence modelling is to employ the *Boussinesq* hypothesis which assumes an explicit dependence of the deviatoric part of the Reynolds stress on the mean-strain of the flow field $\vec{u}_{\text{avg}}$ (Pope, 2000), namely

$$R - \frac{2}{3}kI = -\nu_t \left(\nabla \vec{u}_{\text{avg}} + \nabla^T \vec{u}_{\text{avg}} \right) \tag{2.2.11}$$

where ν_t is termed the *turbulent eddy viscosity*. It is crucial to state that the Boussinesq assumption is valid if the statistical character of the turbulence is invariant under the action of rotation - a condition referred to as *isotropic*. Not all turbulent flows are isotropic. The experimental investigation of an isotropic turbulence flow undergoing a contraction showed significant anisotropic characteristics. For this reason, the implementation of the RANS models, which are predominately based on the crucial specification of the Boussinesq hypothesis, need to be scrutinized case-by-case. But having said that, it can be shown that the assumption has had found great success in a large number of applications in the industry.

Together with Eq.(2.2.11), a solution for the mean-flow field $\vec{u}_{\text{avg}}$ is only possible if ν_t can be prescribed. The specification of ν_t constitutes what is known as the closure problem. The simplest approach, the so called 0-equation closure, uses the Prandtl mixing-length methods. These models assume that there is some length scale, called the mixing length, where the fluid parcel undergoes transfer of momentum by turbulent diffusion and maintains its "identity" before being absorbed/mixed into the surrounding fluid. On dimensional grounds, the eddy-viscosity has the unit of $l^2 t^{-1}$, where l is some small turbulent eddy length and t is some time-scale associated with these small eddies. As such it is customary to define the characteristic turbulent speed, say u_T, and an eddy length l_m such that $\nu_t = u_T l_m$ where l_m is known as the mixing length. In his original postulation, ν_t has dimensional dependencies expressible in terms of the mean strain rate as follows

$$\nu_t = l_m^2 \left| \frac{\partial u_{\text{avg}}}{\partial y} \right|$$

This model is relevant for flows where the dominant velocity gradient is $\partial_y u_{\text{avg}}$, such as in mixing layers, jets, wake axisymmetric jets, boundary layers and pipes and channels (Versteeg and Malalasekera, 1995). Since turbulence is a function of the flow, the mixing length, in general, cannot be assumed to be constant but as a function of space. Additionally, the mixing length theory implies zero eddy-viscosity whenever the strain rate varnishes, which is not physically possible and tenable. Several alterations to the original formulation had been proposed (Doshi and Gill, 1970).

Although the 0-equation closure scheme was the earliest attempt to solve the eddy-viscosity problem posed by Boussinesq. It is now known to possess several weaknesses that limit its applicability in the wider general-purpose CFD community. Typically, it limits the types of flows that the postulation is valid and it has not

the ability to model separation or recirculating flows (Versteeg and Malalasekera, 1995). However, it does lend itself useful in near wall boundary treatment where the velocity in the boundary is characterised by large strains in the normal direction. Thus this is used in more sophisticated CFD codes where wall-treatment is required (ANSYS, 2013).

A more well-known class of modelling approach is based on the $k-\epsilon$ two equation closure (Launder and Spalding, 1974). This approach examines the dynamic behaviour of turbulence and attempts to bridge the limitation of the algebraic approaches such as the mixing-length models. It consists of solving two additional partial differential equations (PDE) relating to the production and destruction of turbulent kinetic energy and energy dissipation. The total kinetic energy of a fluid parcel under the Reynolds decomposition law contains contributions from the mean flow and its fluctuation. If k is defined as the kinetic energy per unit mass of the fluid due to the fluctuating field, one could derive the exact equation for k from the NS equations, i.e.

$$\frac{\partial \rho k}{\partial t} + \partial_i \left(\rho k u_{\text{avg},i}\right) = \partial_i \left(- \left\langle p' u_i' \right\rangle + 2\rho\nu \left\langle u_j' S_{ij}' \right\rangle - \frac{1}{2}\rho \left\langle u_j' u_j' u_i' \right\rangle \right) - 2\rho\nu \left\langle S_{ij}' S_{ij}' \right\rangle - \rho \left\langle u_i' u_j' S_{ij} \right\rangle .$$

where the primed variables correspond to the fluctuating variables and S_{ij} is the component of the averaged strain-rate tensor. The other modelling quantity, the so called energy dissipation per unit volume of fluid ϵ, is defined by the term $2\nu < S_{ij}' S_{ij}' >$. The exact equation satisfied by ϵ is too cumbersome to list here but may be found in Pope (2000). Because of the primed variables, they give rise to terms that, in general, cannot be measured or calculated. Specifically, Launder and Spalding (1974) assumed that all third order terms, e.g. $\partial_i < u_i'(-p' + u_j' u_j'/2) >$, are likely to contribute to the transport of turbulent kinetic energy in a manner that depends on the gradient of the transport quantity. With this assumption, the empirical equations are then given by:

$$\partial_t k + \partial_i \left(k u_{\text{avg},i}\right) = \partial_i \left(\frac{\nu_t}{\sigma_k} \partial_i k \right) + 2\nu_t S_{ij} S_{ij} - \epsilon, \tag{2.2.12}$$

$$\partial_t \epsilon + \partial_i \left(\epsilon u_{\text{avg},i}\right) = \partial_i \left(\frac{\nu_t}{\sigma_\epsilon} \partial_i \epsilon \right) + 2C_{1\epsilon} \frac{\epsilon}{k} \nu_t S_{ij} S_{ij} - C_{2\epsilon} \frac{\epsilon^2}{k}, \tag{2.2.13}$$

$$\nu_t = C_\mu \frac{k^2}{\epsilon}. \tag{2.2.14}$$

Here the $k-\epsilon$ model is controlled by 5 adjustable constants: $\sigma_k, \sigma_\epsilon, C_{1\epsilon}, C_{2\epsilon}$ and C_μ (Launder and Spalding, 1974; Versteeg and Malalasekera, 1995). RANS with Eq.(2.2.12)-Eq.(2.2.14) is referred to as the standard $k-\epsilon$ model and it has been the subject of extensive validations over the course of its inception. In particular, strong agreement with experimental investigations in industrial flows reinforces the degree of confidence in using the $k-\epsilon$ as a viable technique despite the hand-waving approach in modelling the Reynolds stresses. In spite of its success, it is worth mentioning that several iterations of the methods have emerged which attempt to address some critical flaws in the model. For example, the RNG iteration attempts to improve the accuracy of the solution for rapidly strained flow where the standard $k-\epsilon$ is known to perform poorly (ANSYS, 2013). More recently, the *realizable* $k-\epsilon$ model modifies the definition of the eddy-viscosity to better reflect the physicality

of real turbulence.

Due to the gradient diffusion term appearing in the $k - \epsilon$ equations, their behaviours in general are of elliptic in nature. This has important consequences in specifying the type of additional boundary conditions (along with the usual boundary conditions for the NS equations) for the solver, i.e.

- Inlet: initial distribution of k and ϵ.
- Outlet: $\partial_n k = 0$ and $\partial_n \epsilon = 0$ where $\vec{n}$ is the unit outboard normal to the symmetry axis
- Free-stream: $k = 0$ and $\epsilon = 0$
- Solid walls: wall functions at high Reynolds number and wall damping treatment at low Reynolds number to ensure that viscous stresses take over from turbulent Reynolds stresses

see Versteeg and Malalasekera (1995) and the references within for the full discussion.

Of course, the complexity in the field of turbulence modelling has only meant that there are multiple approaches; each with certain advantages to a particular aspect. What we have just discussed here is a small subset of the developed techniques. Over the course of the CFD history, more complicated models have been introduced, such as the Reynolds stress equation model (RSM) which attempts to resolve the Reynolds stresses more accurately as opposed to using the heuristic Boussinesq assumption. Clearly, the more extensive of the modelling technique, the higher the computational cost. For example, the RSM method coupled with the RANS results in an addition of 7 PDE (6 transport equations of $< u_i' u_j' >$ and one ϵ equation) compared to the simple algebraic modelling of the eddy-viscosity such as the Prandtl mixing length model. Thus it is not conceptually difficult to see that the cost associated with the RSM is considerably more expensive. Moreover, the assumptions made in those models might be applicable to certain types of flow. For other flows, however, extensive adjustments to the model parameters need to be constructed in and ad-hoc basis. This then poses a significant problem when dealing with different scenario or flow types since the adjustments often need to be handcrafted case by case.

2.3 Vortex methods

An alternative approach to the grid-based methods is to solve the NS equations in a Lagrangian frame of view. This view tracks a finite number of vorticity-carrying particles which can be used to construct the global flow field by means of the Biot Savart law.

In this physical representation, the discretization is done on the vorticity rather than the velocity. This is particularly useful if vorticity occupies narrow regions of space. In such cases, a particle or filament representation of the vorticity field can be assumed. Together with a vorticity generating mechanism, this technique can offer a robust alternative to the grid-based approach without the need for an ad-hoc turbulence modelling. Furthermore, this representation sets out to resolve two of the most important weakness in the grid-based methods. One is the introduction of numerical diffusion when a grid-based discretization is applied to the diffusion term. This inherently modifies the physical diffusion which can be problematic in

low Reynolds number flow where diffusion is often the dominant effect (Cottet and Koumoutsakos, 2008). The second weakness is the large simulation time typically required by the grid in order to at least obtain a well-resolved simulation (Section 2.2.2). Commonly, a large sparse matrix equation has to be solved iteratively for convergence, the inter-dependence of the equations means the grid-based solver is difficulty to be parallelized on the GPU where data need to be compact for a fully parallel implementation.

The vortex approach is a flexible technique that can be used as a high-fidelity simulation tool or as a mid-fidelity engineering approach. The difference lies in the modelling of the viscosity in the wake and the treatment of solid objects in the computational domain. In this section, various techniques (from low to high) are examined in the context of aerofoil/wing aerodynamic.

2.3.1 The lifting line

The lifting line is a simplified approach in which the detailed representation of a three-dimensional wing is ignored. The surface of the wing is given as a continuous distribution of circulation strength along the span-wise direction (also known as the bound vortex line). By circulation strength we mean the limit of the vorticity (tube) multiplied by its cross-sectional area as the tube diameter varnishes, which becomes a vortex line whose strength is the circulation. This approximation relies on the observation that the effect of the wing is to induce flow curvature on the incoming fluid, which is aerodynamically similar to what a vortex line would do. With the correct strength distribution, the flow curvature can be approximated. In accordance with the Helmholtz's second theorem, who stated that the vortex line cannot end in a fluid; it must extend to the boundary of the fluid or form a closed path, any changes in the lift experienced by the wing is counteracted by the creation of vortex lines into the wake. This then gives rise to the vortex-filament structure which consists of the bound circulation, trailing circulation and the shed circulation. The trailing edge vortex is a response to the span-wise variation of the lift whereas the shed vortex is a response to the temporal lift variation. In this approach, a vortex filament structure can be constructed as in Figure (2.1). The distribution of the bound circulation Γ, can be determined by enforcing the kinematic condition (i.e. the no-through condition at a representative position) and the lift is calculated by means of the Kutta-Joukowski theorem (Katz and Plotkin, 2001; Moran, 1984). The wake of such systems consists of horseshoe vortices which induce a downwash on the wing; when enforcing the kinematic condition, this downwash needs to be taken into account. In addition, the pressure continuity at the tip implies that the bound circulation must varnish at the end points. This results in the Prandtl's integro-differential equation whose solution can be constructed by a Fourier-type series. Physically, the equation represents a statement of computing the effective angle of attack. Commonly, the lifting line is coupled with the free wake model, in which the lattice points of the vortex system are allowed to advect freely. The advection velocity is determined from both the bound vortex line, free-stream velocity and the other vortex filaments in the system. This is considered a transient solution. By allowing the lattice point to move freely, the vorticity stretching is automatically accounted for, however, diffusion effect remains difficult to model. Most investigators (Dixon, 2008; Sebastian and Lackner, 2012; Tescione et al., 2016) adopted the core-spreading approach, in which the regularization core of the vortex filament is to spread outwardly as a function of time - in mimicking the idea that the role of diffusion is to allow a concentrated quantity to spread uniformly in space and time, in this case, the vorticity. Unfortunately, such

simplification does not converge to the NS equations, as reported in the proof of C. Greengard (1985). Therefore, one cannot expect that the solution associated with advecting the filaments follow by spreading the regularization core to satisfy the NS equations in any meaningful way.

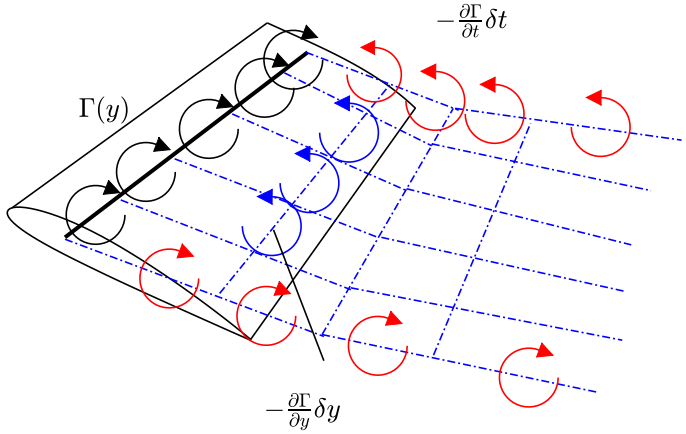


FIGURE 2.1: Schematic diagram describing how the wake is created from the variation of the bound circulation in the spatial and temporal domain. Here y is the span-wise coordinate on the wing.

Despite the flaws, the lifting line approach remains a popular choice for many investigators of rotor aerodynamic (Katz, 1981; Leishman et al., 2002; Sebastian and Lackner, 2012). Since the complicated interaction between the wing and the air flow is simplified as a series of horseshoe vortices whose effect can be calculated by means of the Biot-Savart formula. However, the lifting line model assumes the wing to be sufficiently flat and thin so that camber and thickness effects can be ignored altogether. In cases that such effects are important, the lifting line approach might not be applicable.

One generalization of the approach, taking account of the camber, is to model the wing surface as a two-dimensional surface about the camber. One can then discretize this surface as a system of vortex filaments with a circulation distribution that varies in the local x and y directions (Landhal and Stark, 1977). The same kinematic condition still applies at each representative point on the camber surface. The vorticity wake is generated in the same way as the lifting line. Continuing from this school of thought, the most complete application is to extend the panel to fit around the actual surface of the wing. This way, both camber and thickness are modelled appropriately (Wie et al., 2009). Thus, in this representation, the unknowns in the problem are the *strengths* of various sorts. In Chapter 3, we present the formal mathematical treatment and introduce the source and dipole as the unknown strengths for a fully two/three-dimensional representation with proper diffusion modelling.

2.3.2 The viscid-inviscid coupling algorithm

The limitation of the lifting line method and all of its derivatives is clear, in that there is an implicit assumption that the boundary layer developed on the surface of the wing is thin and will be remained thin in the duration of the flow problem. If the

latter proved to be false, then the effect of boundary layer has to be incorporated into the calculation. In the spirit of an engineering approach, one such scheme is the viscid and inviscid coupling algorithm. In this approach the boundary layer equations are solved and coupled with the outer inviscid solution iteratively until the solutions in each region match. In solving the boundary layer equations, two broad approaches generally adopted in the literature. A finite difference solver may be employed with the inviscid solution acting as the boundary conditions on the interaction interface (Cebeci et al., 2005). Due to the parabolic nature of the boundary layer equations, a simple space marching scheme is possible. So that the solution can start at the stagnation point and progressively marches towards the trailing edge. The solution requires the specification of the boundary layer thickness, which must be guessed initially and iteratively refined until convergence is achieved. The other important class of approach is to introduce the boundary layer variables in the form of integrals of the boundary layer velocity. Specifically, assuming incompressible flow those are the momentum thickness θ , displacement thickness δ^* and kinetic energy thickness θ^*, which are defined as follows (Drela and Giles, 1987):

$$\begin{aligned}
\theta &= \int_0^\infty \frac{u}{u_e}\left(1-\frac{u}{u_e}\right) d\eta, \\
\theta^* &= \int_0^\infty \frac{u}{u_e}\left(1-\left(\frac{u}{u_e}\right)^2\right) d\eta \\
\delta^* &= \int_0^\infty \left(1-\frac{u}{u_e}\right) d\eta,
\end{aligned}$$

where u is the boundary layer u-velocity (projected in the body fitted coordinate frame) and u_e is the outer edge inviscid u-velocity. Here, ϵ and η denote the body-fitted coordinates with ϵ parallel to the surface and η along the normal direction. Integrating the boundary layer equations results in the integral form of the equations:

$$\frac{\epsilon}{\theta}\frac{d\theta}{d\epsilon} = \frac{\epsilon}{\theta}\frac{C_f}{2} - (H+2)\frac{\epsilon}{u_e}\frac{du_e}{d\epsilon} \tag{2.3.1}$$

$$\frac{\epsilon}{H^*}\frac{dH^*}{d\epsilon} = \frac{\epsilon}{\theta}\frac{2C_d}{H^*} - \frac{\epsilon}{\theta}\frac{C_f}{2} - (1-H)\frac{\epsilon}{u_e}\frac{du_e}{d\epsilon}. \tag{2.3.2}$$

where $H^* = \theta^*/\theta$ and $H = \delta^*/\theta$, which are known as the shape factors, and the variables C_f and C_d are called the friction and dissipation coefficient, respectively. Although the integration process produces a set of two ordinary differential equations as opposed to the PDE of the finite difference approach, but those equations contain terms such as the C_d, C_f, H^* that need to be modelled in much the same way as turbulence modelling in Section 2.2.3, which results in a closure problem. By extensive data fitting with experiments and assuming a known velocity distribution inside the boundary layer for both laminar and turbulent flow, Eq.(2.3.1) and Eq.(2.3.2) can be solved to a reasonable accuracy (Drela, 1985). Another issue associated with it is the prediction of laminar bubble and the transition region in which the laminar flow near the stagnation point destabilises and subsequently transitions into turbulence. This is a crucial estimate as it directly predicts the point of separation occurring on the aerofoil surface, which is important for flow separation modelling (Riziotis and Voutsinas, 2008; Zanon et al., 2013). Drela (1985) used the amplification factor approach, in which the amplitude of the perturbation waves emanating from

the stagnation point is determined. Transition is thought to occur when the exponent of this perturbation amplitude has exceeded some critical value. Numerically, this means that there is an additional equation for the amplification factor. Beyond the transition point, the empirical terms are readjusted for turbulence and the same equations are solved for the remaining section. Once the boundary layer variables have been obtained, the coupling equation is then given by:

$$\vec{u}_{\text{inviscd}} \cdot \vec{n}|_{\eta=0} = \frac{d}{d\epsilon}\left(u_e \delta^*\right) \tag{2.3.3}$$

where $\vec{u}_{\text{inviscid}}$ is the outer inviscid flow. Eq.(2.3.3) modifies the boundary condition for the inviscid solver, thus the effect of the boundary layer can be regarded as applying a *blowing* effect to the inviscid flow field.

In an naive approach, a typical coupling loop sees u_e to be prescribed follow by solving Eq.(2.3.1) and Eq.(2.3.2) alongside with the amplification factor equation. The inviscid flow is then modified by satisfying the new boundary equations as in Eq.(2.3.3), which results in a change of u_e and the boundary layer equations are solved again. This process continues until there is no appreciable change in u_e. However, it was discovered that this direct approach cannot proceed beyond some critical point where reversed flow might occur (Wolles, 1968). This difficulty is known as the Goldstein singularity. It was later discovered that such difficulty is not a physical singularity but rather a numerical one. If instead, the displacement thickness was prescribed and the boundary layer is solved for the outer inviscid field u_e, the iteration becomes stable and is able to integrate beyond the critical point.

Although the coupling algorithm provides a convenient way to include viscous effect in an otherwise inviscid flow, it is important to recognise that the empirical relations appearing in the formulation is based on steady flow, which means it is extremely difficult to generalize this approach to a transient problem. The best one can do is to assume a quasi-steady approach, in which one freezes the boundary layer in time and solves for the viscous effect (Ramos-García et al., 2014; Zanon et al., 2013). However, the accuracy of this approach depends on the quasi-steady assumption, which clearly not applicable in highly unsteady problems.

2.3.3 Double wake model for simulating separated flow past aerofoils

Separated flow arises when there is an adverse pressure gradient on the aerofoil that causes the flow near the surface to reverse its direction. As a consequence, the flow might separate from the surface resulting in a distinct shear layer. Traditionally, this shear layer is modelled by a vortex shedding mechanism (Katz, 1981; Vezza and Galbraith, 1985a). The strength of this vortex sheet is calculated from the velocity difference across the shear layer. Vezza and Galbraith (1985b) implemented a panel approach in which the velocity immediately behind the layer is zero and the velocity in front can be calculated from the induction velocity of the panels. Once the strength is determined, a particle discretization is then applied to the layer. However, an inherent weakness in this method is the determination of the separation point which needs to be known a priori for the model to work. Previously, this vital information is either obtained experimentally or via intuition. Recent development on the double wake model is to incorporate the boundary layer equations. This serves two purposes. One is to incorporate viscous effect to the inviscid flow as discussed in Section 2.3.2 and the other is to deduce the separation point based on the boundary layer variable. Specifically, most of those methods rely on the value of the shape factor H. Flow separation is assumed to occur when H reaches a value between 1.8 and

2.4 (Cebeci et al., 1972). The point at which this occurs is taken to be the separation point and the shear layer is placed there. The shear layer is also parametrized by an angle and an length, which have to be obtained iteratively by considering the local velocity direction and magnitude. Due to the unsteadiness in the separated wake region, the boundary layer equations are only solved in the attached region of the aerofoil.

Riziotis and Voutsinas (2008) implemented the double wake along with the viscid-inviscid coupling algorithm for pitching aerofoils. Their results show good agreement with experimental data in terms of the pressure distribution. However, the resulting wake structure is not correctly represented. Indeed, it is a question whether such a low-order method is capable to reproduce the rich vortical structures observed in more sophisticated CFD codes. Based on the formulation of Riziotis and Voutsinas (2008), Zanon et al. (2013) applied the method to a VAWT. The reattachment process is empirically modelled based on the shape factor near the laminar separation region.

2.3.4 High-fidelity vortex methods

The preceded discussion has been concentrated on engineering simplifications to what would be a highly complicated flow problem, i.e. separating flow. Those low order methods can only provide a grossly under-represented picture of the local flow structure. This is certainly inadequate if one tries to develop a complete picture. Commonly, the CFD is more suited for this task. However, due to the introduction of numerical dissipation in the grid and the various turbulence models, the results are often not consistent. High-fidelity vortex method provides a means to overcome this limitation. The central difference that separates a high order scheme and a low order scheme is the treatment of the vorticity field on the surface and in the wake region. Recall that the double-wake model uses the panel approach to generate an inviscid flow field and vorticity is shed along the trailing edge and the separating shear layer. The idea behind a high-order scheme is to allow the vorticity generation to take place naturally on the surface; thereby avoiding solving the boundary layer equations. High order methods have been developed in the past, but only for flow past bluff objects (Cottet and Koumoutsakos, 2008; Eldredge, 2007; Ploumhans and Winckelmans, 2000; Ramachandran et al., 2007). One of the reason that limits its appeal is the complexity of coding such a method in a numerically efficient way. Since vorticity are allowed to diffuse around the body, a large number of vortices are created in the wake at each time-step as opposed to just two in the double wake model. Thus it necessitates an acceleration routine for computing the Biot-Savart law. One popular strategy is the fast-multipole method (L. Greengard and Rokhlin, 1987), which reduces the quadratic complexity of the problem into a linear complexity. Recent trend in this development is to incorporate the power of GPU to solve for the advection velocity of the vortex particles (Goude and Engblom, 2013).

2.4 Aim and structure of the thesis

There are many unsatisfactory strategies in the modelling approaches adopted in the literature. In theory, the lifting line provides a straightforward simplification to the problem, but the difficulty in the diffusion modelling makes it rather hard to recommend. In particular, since the induction is calculated from the filaments, it is difficult to develop an acceleration technique to speed up this part of the computation; thus

preventing it from design applications. The high-order vortex methods are viable for examining flow structures with high fidelity, but those too require significant computational resources to make the methods viable.

Taking advantages of the recent trends in using graphical processing units (GPU) in simulations, the aim of this thesis is to utilise the GPU to make substantial improvements in the existing methodologies that can be used as a tool for wind turbine applications. In this context, this thesis sets out to fulfil two main adjectives relating to the aerodynamic analysis of wind-farms and these are as follows:

- The loss of the wind-farm efficiency is primarily due to the less energetic flow in the farm. Optimization procedures require a detailed study into the aerodynamic behaviour of the flow field around the wind turbines. However, high fidelity tools are too computational demanding to be used as a design tool. Thus it necessitates a fast solution method that accurately approximates the correct physics. However, naive implementation of the vortex particle method results in a quadratic scaling of the problem, which quickly becomes untenable for large scale problems. In this context, this thesis sets out to explore an acceleration technique that could resolve the inherit weakness of the vortex particle method; thus, the new developed tool can be used as a basis for future wind-farm optimization endeavours.

- Aerofoil aerodynamic is crucial to study the macroscopic properties of the air flow, which inherently relates to the large scale flow structure observed in a real turbine. Optimization procedures need the collective efforts of both the large scale and the macroscopic scale study in order to deliver the most optimal output. For this reason, a high fidelity approach needs to be developed to complement the inherent weakness of the engineering approach in order to validate and identify the shortcomings of the large scale studies and to provide a qualitative assessment of the methodologies used. For this reason, a new highly efficient solution procedure has been developed whose resolution can be compared to DNS; this is mainly due, in a large part, to the efficient implementation of the Fast Multipole Method (FMM) on the GPU.

The structure of the thesis is divided as follows. In Chapter 3, the basic formulation of the vortex particle method (VPM) is reviewed and the general mathematical framework of the boundary element method (BEM) is presented in 3D (The 2D case follows naturally from the 3D formulation). These two approaches form the basis for the engineering method that is to be used to solve large scale problems. The novelty in this chapter is the development of the coupling algorithm based on an accurate interpolation scheme using a bilinear function. Moreover, a multipole expansion technique based on Complex Spherical Harmonic Basis (CSHB) function has been developed to accelerate the calculation of the panel influence at far field.

Chapter 4 provides the mathematical formulation of the fast multipole method (FMM) and the implementation detail on a shared memory device is given. The use of the shared memory requires a novel construction of the data tree, which plays a critical role in facilitating the parallelization construct of the new technique. The novelty in this chapter is the development of an analytical expression for computing the strain tensor, which is used to update the circulation vectors and the development of a fast data construction algorithm.

Chapter 5 provides a series of VPM validation case studies and the performance of the GPU is compared to a pure CPU implementation of the FMM. The aim of Chapter 5 is to show that the VPM gives a natural representation of the fluid flow as opposed to the Eulerian representation of the flow field.

In Chapter 6, an algorithm based on the operator splitting has been developed on the GPU for solving the NS equations for flow past arbitrary geometries. The novelty in this chapter is the efficient implementation of the approach on the GPU which relies on several fast routines to compute the body diffusion and the new remeshing technique. Timing results have shown that the new code is a magnitude faster than what is available in the literature.

In Chapter 7, the developed toolbox is applied to aerofoil/turbine aerodynamic in a series of validation case studies relating to turbine applications. Both qualitative and quantitative measures have been obtained and compared to experimental works. Moreover, several large scales problems have been solved using the engineering technique, which concerns with the interaction of a pair of VAWT rotors.

Finally, the general conclusion and the recommendation for future work of the thesis is found in Chapter 8. A basic overview of the GPU architecture used in this work (often referred without giving too much detail) is given in Appendix A.

Chapter 3

Theories and methodologies

In this chapter, the vortex particle method is formulated in terms of a series of Lagrangian markers that carry vorticity information from the vorticity source to the rest of the flow domain (vortex particles). Several aspects of the method are discussed, such as the treatment of the singular behaviour when two vortex particles become close to each other, and the diffusion modelling based on a deterministic scheme that replaces the Laplacian operator with an integral operator. It is noted that the approach requires a constant overlap of the particles to avoid creation of non-physical vortex structures. In this work, a location processing technique is implemented on the GPU which re-meshes the particles at regular intervals.

The second part of the chapter examines the boundary element method (BEM) for solving unsteady problems in attached condition. The developed method relies on solving the Laplace equations by introducing a set of singular distribution whose magnitude is solved by satisfying the no-through conditions. In addition, the current work implements a hybrid approach in which the nascent wake dipole sheets are converted to a set of vortex particles via the Hess' equivalence principle.

To accelerate the panel influence at far field, a multipole expansion technique has been developed which makes use of the Complex Spherical Harmonic Basis function (CSHB) to represent the far field influence. This acceleration technique is invaluable in high fidelity simulations where fine-grained body discretization is required.

3.1 Vortex Particle Method (VPM)- the Lagrangian approach

By simultaneously tracking the trajectories of a collection of fluid elements, one may determine the kinematic properties of the underlying flow field. In this purely Lagrangian setting, the transport phenomenon is resolved exactly. Consequently, this transforms the partial differential equations of the Navier-Stokes flow to a system of coupled ordinary differential equations (ODE). From a numerical point of view, this is a favourable proposition, since it allows an ODE solver to be implemented instead. In this section, we explore the mathematical framework of a purely Lagrangian vortex particle method.

3.1.1 Streamfunction-vorticity formulation

The Navier-Stokes equations describe the dynamic evolution of an incompressible flow. If $(\vec{u}, p)$ describes the state variables at the spatial coordinate $\vec{x}$ and time t, then Eq.(3.1.1) describes the evolution of the state variables in the absence of external forces:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \vec{u} \tag{3.1.1}$$

where $\vec{u}, p, \rho, \mu$ denote the flow velocity, pressure, density and kinematic viscosity, respectively. Additionally,

$$\nabla \cdot \vec{u} = 0 \tag{3.1.2}$$

is imposed to satisfy the incompressibility condition.

By introducing the vorticity field $\vec{\omega} = \nabla \times \vec{u}$, it is possible to transform Eq.(3.1.1) to the vorticity-equations:

$$\frac{\partial \vec{\omega}}{\partial t} + \vec{u} \cdot \nabla \vec{\omega} = \vec{\omega} \cdot \nabla \vec{u} + \nu \nabla^2 \vec{\omega}. \tag{3.1.3}$$

It is worth noting that the first term on the right hand side of Eq.(3.1.1) is known as the *stretching* term as it encapsulates the physics of intensification of the vorticity which correlate to the local strain field (note this term is zero for two-dimensional flow); meanwhile, the second term represents the molecular *diffusion*.

To satisfy the continuity equation (Eq.(3.1.2)), we introduce the *stream-function* $\vec{\psi}$ such that

$$\vec{u} = \nabla \times \vec{\psi}. \tag{3.1.4}$$

The gauge transformation ensures that the solenoidal property is imposed on $\vec{\psi}$. Using the vector identity:

$$\nabla \times \nabla \times \vec{\psi} = \nabla \left(\nabla \cdot \vec{\psi} \right) - \nabla^2 \vec{\psi} \tag{3.1.5}$$

together with the solenoidal assumption, it is readily seen that $\vec{\psi}$ satisfies the *Poisson's equations* in each of its component:

$$\nabla^2 \vec{\psi} = -\vec{\omega}. \tag{3.1.6}$$

Now, by accounting for the far-field requirement ($\left\| \vec{\psi} \right\| \to 0$ as $\|x\| \to 0$), Eq.(3.1.6) is solved by introducing the fundemental solution $G(\vec{x})$ in the 3D Laplace's equation, so that

$$\vec{\psi}(\vec{x}, t) = -\left(G \star \vec{\omega}\right)(\vec{x}, t) := -\int_{\mathbb{R}^3} G(\vec{x} - \vec{y})\, \vec{\omega}(\vec{y}, t)\ d\vec{y}, \tag{3.1.7}$$

where $\star$ denotes the convolution operator and G is given by

$$G(\vec{x}) = -\frac{1}{4\pi \|\vec{x}\|}. \tag{3.1.8}$$

Thus, Eq.(3.1.7) can be explicitly written as:

$$\vec{\psi}(\vec{x}, t) = \frac{1}{4\pi} \int_{\mathbb{R}^3} \frac{1}{\|\vec{x} - \vec{y}\|} \vec{\omega}(\vec{y}, t)\ d\vec{y}. \tag{3.1.9}$$

A particle-approximation is applied to the vorticity field $\omega(\vec{x}, t)$ with compact support in $V := \bigcup_{j=1}^{N} V_j$, where V_j is the compact support of the j-th particle. Thus $\vec{\omega}$ may be approximated by the following expression:

$$\vec{\omega}(\vec{x}, t) = \sum_{j=1}^{N} \vec{\alpha}_j(t)\, \delta\left(\vec{x} - \vec{x}_j(t)\right). \tag{3.1.10}$$

Here, $\vec{\alpha}_j$ is often referred to as the *particle circulation* and is defined by the integral of the vorticity over the support V_j, i.e.

$$\vec{\alpha}_j(t) = \int_{V_j(t)} \vec{\omega}(\vec{y}, t)\, d\vec{y}. \tag{3.1.11}$$

$\vec{x}_j(t)$ is the computed trajectory of the particle which is everywhere tangent to the local velocity field. Upon substituting Eq.(3.1.10) into Eq.(3.1.7) and taking the curl of the resulting expression, the induced velocity due to a discretized vorticity field is computed as follows:

$$\vec{u}_{\text{BiotSavart}}(\vec{x}, t) = -\sum_{j=1}^{N} \nabla G\left(\vec{x} - \vec{x}_j(t)\right) \times \vec{\alpha}_j(t) \tag{3.1.12}$$

Eq.(3.1.12) is known as the *Biot-Savart* formula and it plays many pivotal roles across different disciplines of science. Following the trajectory of the j-th vortex particle (so $\partial/\partial t + \vec{u} \cdot \nabla = d/dt$) and integrating Eq.(3.1.3) over the support V_j, the vorticity equation is equivalent to the following system of ODEs:

$$\frac{d\vec{x}_j}{dt} = \vec{u}\left(\vec{x}_j, t\right), \tag{3.1.13a}$$

$$\frac{d\vec{\alpha}_j}{dt} = \vec{\alpha}_j \cdot \nabla \vec{u} + \nu \int_{V_j} \nabla^2 \vec{\omega}\, dV. \tag{3.1.13b}$$

The velocity vector in the above typically consists of the induced Biot-Savart, a velocity field that results from a scalar potential (see Section 3.2) and a constant free-stream. As we shall discuss later, the diffusion term on the right hand side of Eq.(3.1.13b) is replaced by the particle-exchange scheme (PSE), in which the Laplacian appearing in the integrand is replaced by an integral operator.

Unfortunately, the free-space Green's function appearing in the solution of the stream-function (Eq.(3.1.6)) exhibits a strong singularity at the particles' location (or a logarithmic singularity in 2D). At the same time, the discretized-vorticity is not defined at those points because of the *Dirac* delta function. Consequently, if this singularity is not avoided, it may give rise to unbounded velocity whenever two particles are in close proximity of each other, which, on physical grounds, is not entirely realistic as viscosity is likely to regularize the field when that occurs. For this reason, as well as for stability argument, one must apply some form of regularization to avoid this singular behaviour. A formal approach is to accept the fact that the vortex particles are not strictly singular in space but they contain a core region in which diffusion can take place. This process of removing the singularity is formalized by replacing ∇G with a mollified expression parametrized by a core parameter (Cottet and Koumoutsakos (2008, p. 20)), see Section 3.1.2 for more detail.

3.1.2 Mollification of the singular field

Let ϵ denote a positive scalar and ζ be a smooth function (cut-off) chosen in such a way that it shares the same moment as the Dirac delta function as in Eq.(3.1.10) up to the order $R-1$, where R is known as the order of the cut-off. More precisely, ζ is

said to have order R if the following properties are satisfied:

$$\int_{\mathbb{R}^3} \zeta(\vec{x})\, d\vec{x} = 1, \tag{3.1.14a}$$

$$\int_{\mathbb{R}^3} \vec{x}^{\vec{i}} \zeta(\vec{x})\, d\vec{x} = 0, \quad |\vec{i}| \leq R-1 \tag{3.1.14b}$$

$$\int_{\mathbb{R}^3} |\vec{x}|^R \zeta(\vec{x})\, d\vec{x} < \infty. \tag{3.1.14c}$$

where the multi-index $\vec{i} = (i_1, i_2, i_3) \in \mathbb{N}_+^3$ has the following meanings: $|\vec{i}| = i_1 + i_2 + i_3$ and $\vec{x}^{\vec{i}} = x_1^{i_1} x_2^{i_2} x_3^{i_3}$. It should be noted that there is a natural generalization to an arbitrary space dimension.

Once a smooth function ζ is chosen, the mollification of ζ is derived by setting:

$$\zeta_\epsilon(\vec{x}) = \epsilon^{-3} \zeta\left(\frac{\vec{x}}{\epsilon}\right). \tag{3.1.15}$$

In practice, the choice of ϵ is informed on the basis of simulation resolution (which in turns depends on the Reynolds number) as well as on the stability consideration imposed by the particle-strength exchange scheme discussed in Section 5.4. For the sake of argument, it is taken for now that ϵ is a small but non-zero constant.

By replacing the Dirac delta function with ζ_ϵ, one arrives at the mollified vorticity field:

$$\vec{\omega}_\epsilon(\vec{x}, t) = \sum_{j=1}^{N} \vec{\alpha}_j(t) \zeta_\epsilon\left(\vec{x} - \vec{x}_j(t)\right) \tag{3.1.16}$$

As an example, one popular choice for a second-order cut-off function ζ is the *Gaussian* function of the form:

$$\zeta(\vec{x}) = \frac{1}{(2\pi)^{3/2}} \exp\left(-\frac{|\vec{x}|^2}{2}\right). \tag{3.1.17}$$

For this cut-off, Figure (3.1) shows the profile of the mollified function at three values of ϵ. The result shows that the mollified function does indeed converge asymptotically to the Dirac delta function as $\epsilon \to 0$. The effect of mollification can be thought of as to spread the vorticity to its surrounding region. This way, the scalar ϵ is naturally interpreted as the core radius of the vortex *blob*.

Based on the Gaussian smoothing, one can show that the mollified velocity is given by (He and Zhao, 2009):

$$\vec{u}_\epsilon(\vec{x}, t) = -\sum_{j=1}^{N} \vec{K}_\epsilon\left(\vec{x} - \vec{x}_j\right) \times \vec{\alpha}_j \tag{3.1.18}$$

where

$$\vec{K}_\epsilon(\vec{x}) := \epsilon^{-3} f\left(\frac{\|\vec{x}\|}{\epsilon}\right) \vec{x}, \tag{3.1.19}$$

and

$$f(s) = \frac{1}{s^2}\left(\frac{1}{4\pi s}\operatorname{erf}\left(\frac{s}{\sqrt{2}}\right) - \zeta(s)\right). \tag{3.1.20}$$

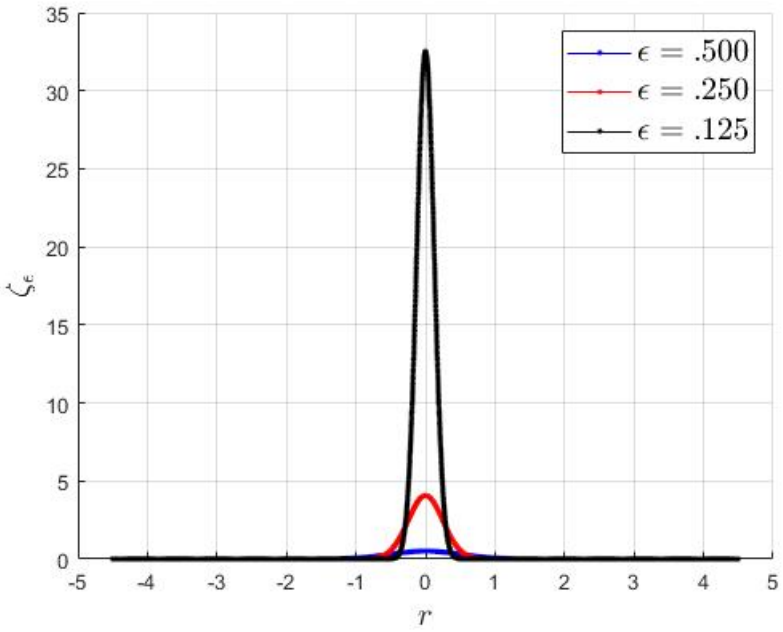


FIGURE 3.1: Eq.(3.1.17) at different values of ϵ. It can be seen that ζ_ϵ approaches to the Dirac delta function asymptotically as $\epsilon \to 0$.

Here, erf (s) is the error function defined as

$$\text{erf}\,(s) = \frac{2}{\sqrt{\pi}} \int_0^s \exp\left(-v^2\right)\, dv. \tag{3.1.21}$$

As Figure (3.2) demonstrates, an important observation is that the mollified kernel is regular at the origin as long as the core radius is non-zero there. Of course, one is not restricted to the second-order cut-off Gaussian smoothing and higher-order cut-offs are possible. However, it is often the case that higher-order cut-offs often violate the positivity property, which means that a locally positive circulation may be negative in the mollified field.

3.1.3 Particle-strength-exchange scheme (PSE) to model viscosity

The effect of viscosity can be simulated in various ways in a particle setting. The earliest offering was proposed by Chorin (Chorin, 1973). In his approach, a probabilistic strategy was adopted to mimic the large frequency spectral that is typically associated with turbulence motion. Diffusion is thus interpreted as fluid elements undergoing *Brownian* motion. In such view, a random component, which is associated with a Gaussian probability distribution, is attached to the position of the vortex blobs at each time step. While the approach demonstrates the advantage of being simple to implement, but it is also noted by various researchers in the past (see Berdowski (2015) and Cottet and Koumoutsakos (2008)) for its poor accuracy at a given computational cost. The problem is that, being a probabilistic formulation, the accuracy is only guaranteed for a relatively large sampling size, meaning a large vortex population is generally required, which is untenable in practice because of the limiting computer resource available.

A deterministic approach, however, has the advantage of representing the correct diffusion physics whilst maintaining the computational efficiency. One interesting

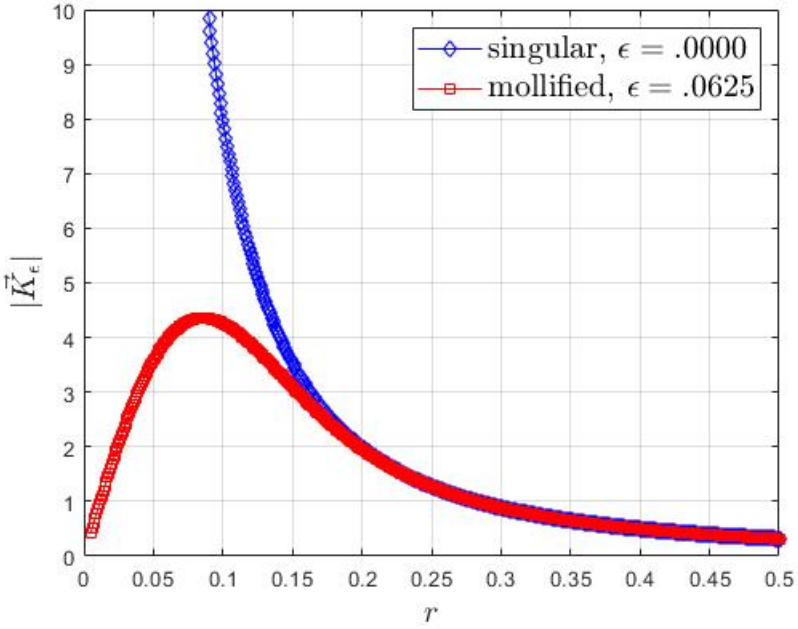


FIGURE 3.2: The magnitude of the mollified kernel as in Eq.(3.1.19) compared to the singular kernel, $|\nabla G|$

solution is to apply the particle-exchange scheme, or PSE, proposed by Degond and Mas-Gallic, 1989 to the diffusion term in Eq.(3.1.13b). Specifically, the Laplacian operator is to be replaced by the integral operator, so that:

$$\nabla^2 \vec{\omega} = \epsilon^{-2} \int_{\mathbb{R}^3} g_\epsilon \left(\vec{x} - \vec{y}\right) \left(\vec{\omega}\left(\vec{y}, t\right) - \vec{\omega}\left(\vec{x}, t\right)\right) \, d\vec{y} \tag{3.1.22}$$

where g_ϵ is the mollification of a smooth scalar function that satisfies certain moment conditions. For reason to be discussed later, we require that g_ϵ to be even in its arguments. Discretizing the integral in Eq.(3.1.22) with a quadrature rule whereby the weights and the nodes are chosen to be *volumes* and particle positions yields the discrete form:

$$\int_{V_j} \nabla^2 \vec{\omega} \, d\vec{y} \approx \sum_{j=1}^{N} \frac{1}{\epsilon^2} g_\epsilon \left(\vec{x}_i\left(t\right) - \vec{x}_j\left(t\right)\right) \left(|V_i| \vec{\alpha}_j - |V_j| \vec{\alpha}_i\right) \tag{3.1.23}$$

where $|V_j|$ is the volume of the vortex blob. In practice, each vortex blob is given a core region of variable size. It is known from Kelvin's theorem that the total circulation in the flow is a conserved quantity, so it is preferable to use a numerical scheme that respects this property. Formally, this is equivalent to the statement:

$$\frac{d}{dt} \left(\sum_{j=1}^{N} \vec{\alpha}_j\left(t\right) \right) = 0. \tag{3.1.24}$$

The matter is now turned to choosing a suitable ϵ so that Eq.(3.1.24) is fulfilled. For this purpose, the simplest symmetric core is one that averages the two core parameters. Specifically, if ϵ_i and ϵ_j denote the core parameters of the i-th and the j-th

particles, one simply replaces ϵ in Eq.(3.1.23) with ϵ_{ij}, where

$$\epsilon_{ij} = \sqrt{\frac{\epsilon_i^2 + \epsilon_j^2}{2}}. \tag{3.1.25}$$

Consequently, the conservative nature of the PSE scheme is notably visible from its skew property. To see this, consider the change of the circulation of the i-th particle due to the interaction with particle j. Ignoring the stretching effect, this change is given by $\delta\vec{\alpha}_i$, where

$$\delta\vec{\alpha}_i = \frac{\nu\delta t}{\epsilon_{ij}^2} g_\epsilon \left(\vec{x}_i - \vec{x}_j\right) \left(|V_i|\vec{\alpha}_j - |V_j|\vec{\alpha}_i\right). \tag{3.1.26}$$

Conversely, one can show that the circulation change of particle j is the negative of $\delta\vec{\alpha}_i$, so that the net exchange between particle i and particle j is effectively zero. i.e.

$$\frac{\delta\left(\vec{\alpha}_i + \vec{\alpha}_j\right)}{\delta t} = 0. \tag{3.1.27}$$

Extending the above argument to include all pair-wise interactions and taking the limit as $\delta t \to 0$ yields Eq.(3.1.24) as one would expect. However, one important corollary resulting from the above observation is that particles cannot diffuse their vorticity if they are not *overlapped*. This is because the kernel in the PSE is typically a rapidly decreasing function, so if an interactive particle is sufficiently far from the target particle, the change as given in Eq.(3.1.26) rapidly decreases. From Figure (3.1), the diffusion length-scale for the vorticity field (assuming the Gaussian function was used) is comparable to the core radius of the particles. This brings an important limitation regarding the PSE approach - particle distortion.

3.1.4 Lagrangian grid distortion

The limitation of the PSE approach outlined in Section 3.1.3 is a serious issue when particles are separated at a distance that is a magnitude greater than the core radius as it means there is no way for them to communicate so they cannot diffuse vorticity. Typically, this occurs in under-resolved region in which the local strain field spreads the particles apart. At the same time, particles may also be clustered in another direction, which results in an over-resolved region. Under such condition, the local Reynolds number diminishes, which introduces non-physical vortex structures at length-scales smaller than the allowable inter-particle distance. Consequently, if the distortion of the vortex particle distribution is not corrected under a severe local strain, the following detrimental effects can occur:

1. diffusion is severely inhibited because of the inter-particle spacing between particles exceeds the diffusion scale, which is on the order of the core radius.

2. simulation may be destabilized by the introduction of small non-physical vortex structures in regions of particle cluttering. Consequently, this instability may result in the unbounded growth of the circulation vectors

The solution to this issue are typically approached in two ways. First, one could process the circulation strength so as to account for the distortion of the Lagrangian grid. Secondly, one can restart the particle distribution at new locations at which

the overlapping condition is maintained. In this work, the location processing approach is chosen to avoid solving an ill-conditioned system of linear equations that is typically associated with the circulation processing techniques.

In the location processing method, the set of vortex particles is to be replaced by a new set whose locations coincide with the nodes of a Eulerian grid. The circulation of the new particles are derived via a high order interpolation formula which respect the moment conditions of the old particles. Principally, the new particles must conserve the circulation, linear impulse and angular impulse of the field. Specifically, if $\left(\vec{\alpha}_j^{\text{old}}, \vec{x}_j^{\text{old}}\right)$ characterizes the j-th particle, one wishes to determine the new particle $\left(\vec{\alpha}_{j'}^{\text{new}}, \vec{x}_{j'}^{\text{new}}\right)$ in such a way:

$$\vec{\alpha}_{j'}^{\text{new}} = \sum_{j=1}^{N} W\left(H^{-1}\left(\vec{x}_j^{\text{old}} - \vec{x}_{j'}^{\text{new}}\right)\right)\vec{\alpha}_j^{\text{old}} \tag{3.1.28}$$

where W denotes the interpolation matrix and $H := \text{diag}\left(h_x, h_y, h_z\right)$, where h_x, h_y, h_z denote the grid-spacing in the direction x, y and z, respectively. In practice, Eq.(3.1.28) is applied by introducing a scalar interpolation kernel in each direction, which renders W to be diagonal with diagonal elements given by

$$W_{ii}\left(\vec{x}\right) = M\left(\vec{e}_i \cdot \vec{x}\right). \tag{3.1.29}$$

Here, M is the one-dimensional interpolation kernel and $\vec{e}_i$ denotes the Cartesian basis vector.

In this thesis, a fourth-order, also commonly denoted by M_4', scheme was used (Figure (3.3)) and is defined as follows:

$$M_4'\left(x\right) = \begin{cases} 0 & \text{if } |x| > 2, \\ \frac{1}{2}\left(2 - |x|\right)^2\left(1 - |x|\right) & \text{if } 1 \leq |x| \leq 2, \\ 1 - 2.5|x|^2 + 1.5|x|^3 & \text{if } |x| \leq 1, \end{cases} \tag{3.1.30}$$

Eq.(3.1.30) was noted for its accuracy and efficiency (see Monaghan, 1985 and Cottet and Koumoutsakos, 2008 and the references within).

3.2 Boundary element method (BEM)

High Reynold number applications often give rise to situations in which real viscous flow is confined to a thin layer of strong vortical flow. Outside of this layer, however, the flow field is approximately that of inviscid and irrotational nature. In such applications, the Euler equations may be treated as a first approximation to the Navier-Stokes flow. There are many advantages of utilising the Euler equations. Chiefly, because the diffusion term in the Navier-Stoke's equations is notably missing; the computational effort in resolving the diffusion scale is altogether avoided, which means the solver only has to resolve for the dominant flow scales, which can be done in a numerically efficient way. Furthermore, with suitable modelling of the viscous flows in the wake region (e.g. vortex particle modelling), the error of the inviscid approximation is limited only to the boundary layer on the body. In attached flow scenario, this is an acceptable practice. In this section, we present the boundary element method, which solves the Euler's equations for a given solid body with arbitrary motion.

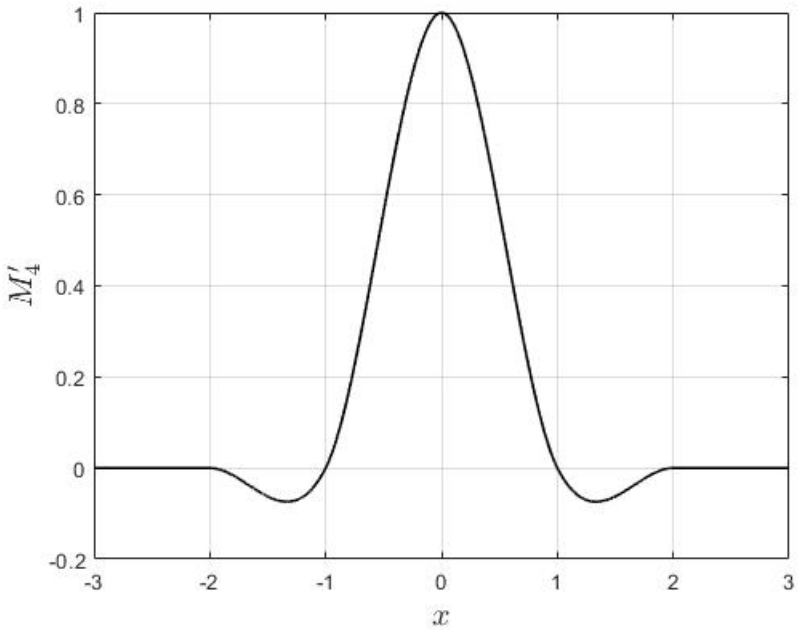


FIGURE 3.3: M_4' is continuous in the first derivative.

3.2.1 Mathematical formulation

We begin by considering the Euler's equation, which is given by

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\frac{1}{\rho} \nabla p \tag{3.2.1}$$

with mass conservation

$$\nabla \cdot \vec{u} = 0. \tag{3.2.2}$$

where $\vec{u}$ is the flow velocity, p is the pressure field and ρ is the constant density. By virtue of the irrotational assumption, which purposes the vorticity be zero ($\nabla \times \vec{u} = 0$), the velocity field can be characterized by a single scalar function, say ϕ, which is related to the velocity as

$$\vec{u} = \nabla \phi. \tag{3.2.3}$$

Indeed, one could verify that the above relation does imply the irrotational assumption. Taking the curl of Eq.(3.2.3), one has that

$$(\nabla \times \nabla \phi)_i = \epsilon_{ijk} \partial_j \partial_k \phi = -\epsilon_{ijk} \partial_k \partial_j \phi. \tag{3.2.4}$$

where we have used the anti-symmetry properties of the Levi-Civita symbol and the commutative property of the differential operators. Noting that the ith component of $(\nabla \times \nabla \phi)_i$ in Eq.(3.2.4) is equal to the negative of itself, one concludes that the sum is zero. One should note that, contrary to other grid-based methods, the pressure variable is not directly used to determined the velocity. It is, however, required to determine the force via the application of the unsteady Bernoulli's equation (see Section 7.3.1 for the exact implementation).

By mass conservation, ϕ yields the solution to the Laplace's equation

$$\nabla^2 \phi = 0. \tag{3.2.5}$$

Consider a streamlined solid body with surface boundary S_B moving through an infinite domain of inviscid flow whose boundary is denoted by S_∞. Suppose further that the body leaves a thin trailing wake region, which is denoted by S_W. Let $\vec{x}$ be a fixed point inside the domain bounded by $V := S_B \cup S_W \cup S_\infty$. A sphere, centred at $\vec{x}$, with radius ϵ is excluded in V. By introducing the domain cut, as it shown in Figure (3.4b), one can verify that the modified domain V is simply connected. Let $\phi_g = \phi_g(\vec{y}, t; \vec{x})$ be any twice differentiable function that satisfies the Laplace's equation Eq.(3.2.5) in the $\vec{y}$ variable. Suppose further that $\phi = \phi(\vec{y}, t)$. In a simply-connected region V, the divergence theorem states that for a vector-valued function $\vec{F}$, the following holds:

$$\int_V \nabla \cdot \vec{F} dV = -\int_{\partial V} \vec{n} \cdot \vec{F} dS, \tag{3.2.6}$$

where $\vec{n}$ denotes the "inward" pointing normal (hence the minus sign on the RHS). With the identity $\phi_g \nabla^2 \phi \equiv \nabla \cdot (\phi_g \nabla \phi) - \nabla \phi_g \cdot \nabla \phi$, one can show that the following holds:

$$\phi_g \nabla^2 \phi - \phi \nabla^2 \phi_g = \nabla \cdot (\phi_g \nabla \phi - \phi \nabla \phi_g). \tag{3.2.7}$$

Since $\nabla^2 \phi = 0 = \nabla^2 \phi_g$ in a simply connected region and together with Eq.(3.2.6), it follows that the following surface integral varnishes, i.e.

$$-\int_{\partial V} \vec{n} \cdot (\phi_g \nabla \phi - \phi \nabla \phi_g) \, dS(\vec{y}) = 0 \tag{3.2.8}$$

Moreover, in the limit when the length of the domain cuts varnishes, the combined contribution to the surface integral must be zero, so that

$$\int_{S_\epsilon} I(\vec{y}, t; \vec{x}) \, dS(\vec{y}) + \int_{S_B \cup S_W \cup S_\infty} I(\vec{y}, t; \vec{x}) \, dS(\vec{y}) = 0, \tag{3.2.9}$$

where

$$I := \phi_g \vec{n} \cdot \nabla \phi - \phi \vec{n} \cdot \nabla \phi_g. \tag{3.2.10}$$

Suppose $\phi_g = 1/\epsilon$ as $\|\vec{y} - \vec{x}\| = \epsilon$, so $\|\nabla \phi_g\| = 1/\epsilon^2$. This is easily verified by setting

$$\phi_g(\vec{y}, t; \vec{x}) = \frac{1}{\|\vec{y} - \vec{x}\|}. \tag{3.2.11}$$

As $\epsilon \to 0$ and assuming regularization of the first derivative of ϕ, one has

$$\int_{S_\epsilon} I(\vec{y}, t; \vec{x}) \, dS(\vec{y}) \approx 4\pi\epsilon^2 \times \left(\epsilon^{-1} n_k \partial_k \phi + \phi \epsilon^{-2}\right) = 4\pi\phi(\vec{x}, t). \tag{3.2.12}$$

Thus we have

$$\phi(\vec{x}, t) = -\frac{1}{4\pi} \int_{S_B \cup S_W \cup S_\infty} I(\vec{y}, t; \vec{x}) \, dS(\vec{y}). \tag{3.2.13}$$

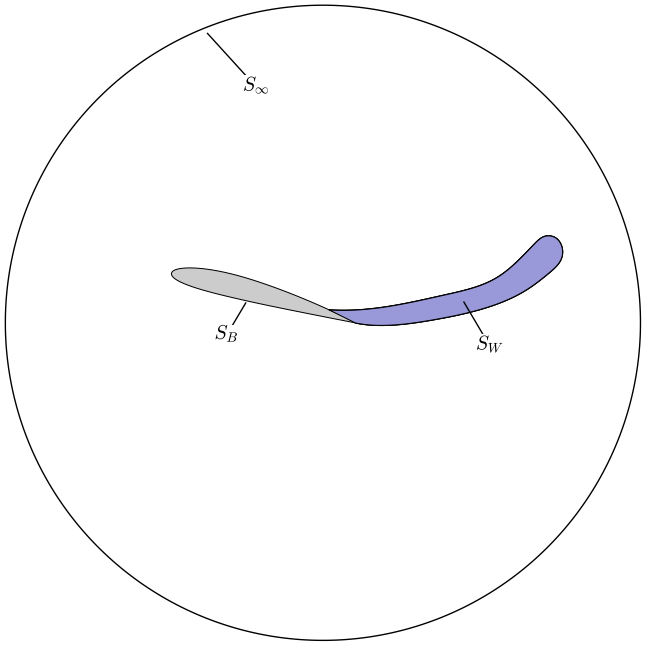
Interestingly, if one supposes that $S_B \cup S_W$ bounds a volume of inviscid flow (see Figure (3.5) for the domain definition) then it is possible to define an internal potential $\phi^* = \phi^*(\vec{y}, t)$, which, if $\vec{x} \in V$, satisfies the following equality:

$$-\int_{S_B \cup S_W} I^*(\vec{y}, t; \vec{x}) \, dS(\vec{y}) \equiv 0. \tag{3.2.14}$$

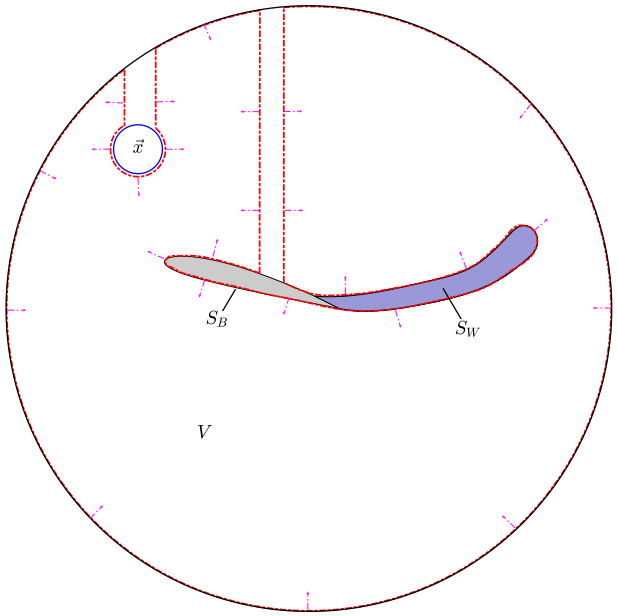
where

$$I^* := \phi_g \vec{n} \cdot \nabla \phi^* - \phi^* \vec{n} \cdot \nabla \phi_g. \tag{3.2.15}$$

Note that the minus sign in Eq.(3.2.14) reflects the fact that the normal vector now



(A) The computational domain



(B) Modified domain

FIGURE 3.4: Domain decomposition for solving Eq.(3.2.5). (A) is the original domain. (B) is the modified domain subjecting to two branch cuts.

points into V'. Hence for any function ϕ^* fulfilling Eq.(3.2.5) and Eq.(3.2.14), one can

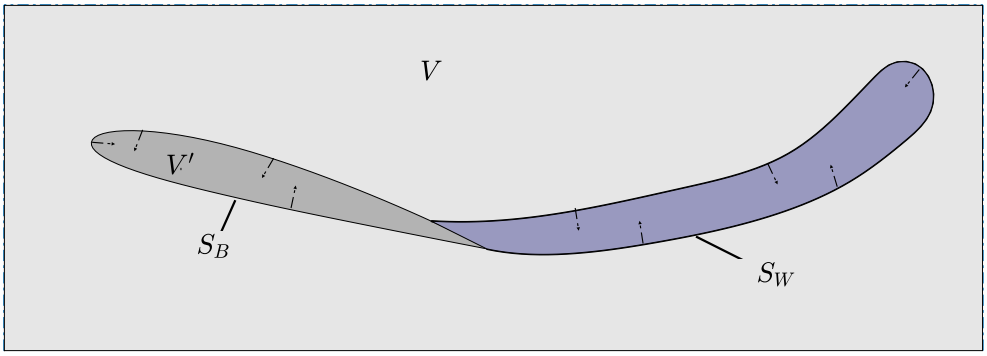


FIGURE 3.5: Domain definition of the internal field ϕ^*.

construct an arbitary solution ϕ to the Laplace's equation as

$$\phi(\vec{x},t) = \phi_\infty(\vec{x},t) - \frac{1}{4\pi}\int_{S_B\cup S_W}\left[\frac{1}{\|\vec{y}-\vec{x}\|}n_k\partial_k(\phi-\phi^*) - (\phi-\phi^*)\,n_k\partial_k\left(\frac{1}{\|\vec{y}-\vec{x}\|}\right)\right]dS(\vec{y}) \quad (3.2.16)$$

and

$$\phi_\infty(\vec{x},t) = -\frac{1}{4\pi}\int_{S_\infty} I(\vec{y},t;\vec{x})\,dS(\vec{y}). \quad (3.2.17)$$

Typically, if V is subjected to an ambient free-stream $\vec{u}_\infty(t)$, one can determine the exact form of ϕ_∞, i.e.

$$\phi_\infty(\vec{x},t) = \vec{x}\cdot\vec{u}_\infty(t). \quad (3.2.18)$$

It is convenient to introduce the singular surface distributions σ and μ (referred to as *source* and *dipole*, respectively) as

$$\sigma := n_k\partial_k(\phi-\phi^*) \quad (3.2.19)$$

$$\mu := \phi-\phi^*. \quad (3.2.20)$$

It should be noted that Eq.(3.2.19) and Eq.(3.2.20) differ from the definitions found in Katz and Plotkin (2001) by a reverse of sign due to the way that the normals were defined.

For completeness, one could also check for the consistency of the solution. That is to say, if $\vec{x}$ were located inside V', one then expects the continuity of the field to hold, i.e.

$$\phi(\vec{x},t) = \phi^*(\vec{x},t). \quad (3.2.21)$$

Indeed, by applying the proceeding analysis, ϕ^* may be expressed as

$$\phi^*(\vec{x},t) = \frac{1}{4\pi}\int_{S_B\cup S_W} I^*(\vec{y},t;\vec{x})\,dS(\vec{y}), \quad (3.2.22)$$

and for a potential field ϕ outside of V, we have

$$\int_{S_B\cup S_W\cup S_\infty} I(\vec{y},t;\vec{x})\,dS(\vec{y}) = 0, \quad \vec{x}\in V'. \quad (3.2.23)$$

By combining Eq.(3.2.22) and Eq.(3.2.23), Eq.(3.2.21) follows as expected.

In conclusion, the general solution to the Laplace's equation is controlled by three parameters. The surface values of ϕ and its normal derivatives (σ and μ) and an arbitrary internal field ϕ^*. In order to determine the field uniquely, we require further information about the problem.

3.2.2 Boundary conditions, panel discretization and the Kutta condition

Let $\vec{u}_B$ denote the velocity of S_B and $\vec{v}_p$ be any perturbation velocity in the flow, we seek values of the singular distributions (σ and μ) so that

$$\left(\nabla\phi + \vec{v}_p - \vec{u}_B\right) \cdot \vec{n}_B = 0. \tag{3.2.24}$$

This physical condition (also known as the impermeable condition) can be realised via two methods. The *Neumann* condition enforces Eq.(3.2.24) exactly, but at the expense of evaluating the gradient of ϕ to obtain the vector velocity. The other method, termed the *Dirichlet* formulation, indirectly satisfies the boundary condition by prescribing the potential field everywhere on S_B. The Dirichlet approach clearly leads to a more efficient algorithm but the issue is to enforce the correct values of the potential.

In the Dirichlet approach, the first step is to determine the source values, which has to be known a priori, which, by examining Eq.(3.2.19) and Eq.(3.2.20), can be done as follows: First, we set the internal potential to be $\phi^*\left(\vec{x},t\right) = \phi_\infty\left(\vec{x},t\right)$ and note that ϕ^* satisfies Eq.(3.2.5) and Eq.(3.2.14), so it is true from Eq.(3.2.19) that

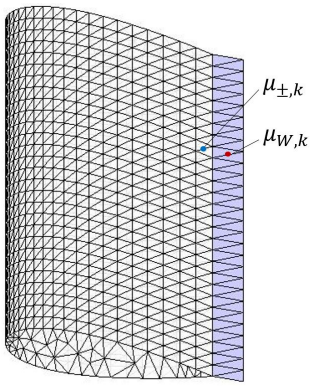
$$\sigma = \vec{n}_B \cdot \nabla\left(\phi - \phi_\infty\right) = \vec{n}_B \cdot \left(-\vec{u}_\infty - \vec{v}_p + \vec{u}_B\right). \tag{3.2.25}$$

With the source strength fixed, $\vec{x} \in V'$ and with the use of the consistency relation Eq.(3.2.21), upon cancelling ϕ_∞ on both sides in Eq.(3.2.16), we arrive at the equation:

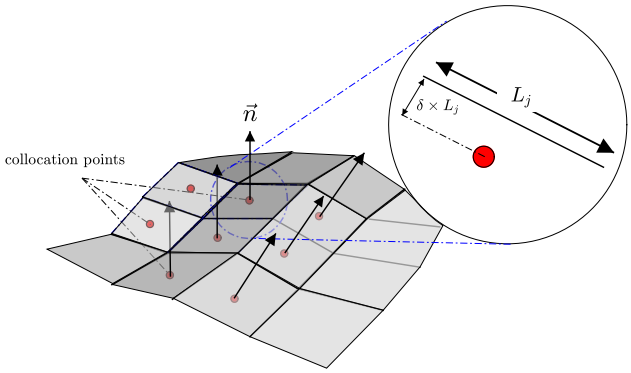
$$\frac{1}{4\pi}\int_{S_B} \sigma \frac{1}{\|\vec{y}-\vec{x}\|} dS\left(\vec{y}\right) - \frac{1}{4\pi}\int_{S_B \cup S_W} \mu \partial_n \left(\frac{1}{\|\vec{y}-\vec{x}\|}\right) dS\left(\vec{y}\right) = 0, \quad \vec{x} \in V'. \tag{3.2.26}$$

The above forms the general methodology of the BEM in a Dirichlet setting. With a suitable discretization scheme, Eq.(3.2.26) typically results in a set of linear algebraic equations for the singular elements on the body.

Low order methods, such as those found in the literature (Katz and Plotkin, 2001, Dixon, 2008), assume a piece-wise constant distribution for the dipole and the body is usually discretized by flat quadrilateral/triangular panels Figure (3.6a). This approach is conceptually simple and easy to code. However, one of the biggest drawback is the lack of accuracy. Higher order methods mitigate this deficiency by introducing a high-order interpolation function for the singular distributions while modelling the body geometry as curved panels (Willis, 2006). Clearly, higher order methods are superior in the sense of obtaining a comparable accuracy at a much coarse level, but due to the complexity of the scheme, those methods generally require a substantial amount of coding effort to be efficient. Driven by the needs to reduce the simulation time, we have adopted a low order approach whereby the body discretization is done using a combination of flat quadrilateral and triangular panels and the singular distributions are assumed piece-wise constant.



(A)



(B)

FIGURE 3.6: Schematic of a low-order discretization of a 3D wing (A) and position of the collocation points used in the BEM formulation (B).

Let M and K denote the number of body and wake panels, respectively. One discretizes the surface S_B and S_W as follows:

$$S_B = \sum_{j=1}^{M} S_j, \tag{3.2.27}$$

$$S_W = \sum_{k=1}^{K} S_{W,k}. \tag{3.2.28}$$

Further, let us introduce the *influencing* functions, which are defined by the following integrals:

$$A_j(\vec{x}) := -\frac{1}{4\pi}\int_{S_j} \vec{n}\cdot\nabla\left(\frac{1}{\|\vec{y}-\vec{x}\|}\right) dS(\vec{y}), \tag{3.2.29}$$

$$B_j(\vec{x}) := \frac{1}{4\pi}\int_{S_j} \frac{-1}{\|\vec{y}-\vec{x}\|} dS(\vec{y}), \tag{3.2.30}$$

$$C_k(\vec{x}) := -\frac{1}{4\pi}\int_{S_{W,k}} \vec{n}\cdot\nabla\left(\frac{1}{\|\vec{y}-\vec{x}\|}\right) dS(\vec{y}). \tag{3.2.31}$$

By averaging each singular distributions over S_j, the following representative values are obtained:

$$\sigma_j(t) = \frac{1}{|S_j|}\int_{S_j}\sigma(\vec{y},t)\,dS(\vec{y}), \tag{3.2.32}$$

$$\mu_j(t) = \frac{1}{|S_j|}\int_{S_j}\mu(\vec{y},t)\,dS(\vec{y}), \tag{3.2.33}$$

$$\mu_{W,k}(t) = \frac{1}{|S_{W,k}|}\int_{S_{W,k}}\mu(\vec{y},t)\,dS(\vec{y}). \tag{3.2.34}$$

Substituting the influencing functions and the averaged values to the BEM equation, one arrives at the discretized form:

$$\sum_{j=1}^{M}\mu_j(t)\,A_j(\vec{x}) + \sum_{k=1}^{K}\mu_{W,k}(t)\,C_k(\vec{x}) = \sum_{j=1}^{M}\sigma_j(t)\,B_j(\vec{x}). \tag{3.2.35}$$

Enforcing the Dirichlet conditions at the collocation points results in M algebraic equations. Accounting for the unknown values of the wake strength $\mu_{W,k}$, the number of unknowns is $M+K$, which means the problem cannot admit a unique solution without additional physical consideration on the flow pattern near the trailing edge.

Typically, a resolution of this nature is provided by the use of the *Kutta* condition, which purposes the flow field to be finite at the trailing edge. On physical grounds, this is equivalent to the following conditions:

- continuity of the pressure field across the trailing edge region;
- the total circulation is necessarily zero.

Depending on the numerical schemes, either conditions fix the amount of the wake circulation generated by the body. Mathematically, the latter item in the above list is expressed as

$$\mu_{W,k} = \mu_{+,k} - \mu_{-,k}, \tag{3.2.36}$$

where $\mu_{\pm,k}$ denote the trailing dipole strengths whose panels share a common edge with $S_{W,k}$ (see Figure (3.6a)). This, together with Eq.(3.2.35) now constitutes the correct number of constrains and unknowns.

Numerically, the enforcement of Eq.(3.2.26) occurs at the collocation points, which are defined inside S_B. Currently, the points are chosen in a manner similar to Dixon (2008), in which off-centre position (controlled by the parameter δ) are used. Typically, δ is chosen to be 1.5% of the characteristic panel length (Figure (3.6b)).

3.2.3 Wake shedding angle and shedding length

Unfortunately, the BEM formulation does not allow the immediate wake geometry to be determined. Generally, this problem has to be approached either by prescribing a known wake shape in the understanding that such shape approximates the real one in some physical sense or applying an iterative scheme in which the shape is allowed to evolve according to the local flow. More precisely, we introduce two parameters that define the generating wake in the immediate vicinity of the trailing edge - the *shedding angle*, α and the *shedding length*, γ. For the sake of argument, those are defined on a two-dimensional cross-section station (Figure (3.7)).

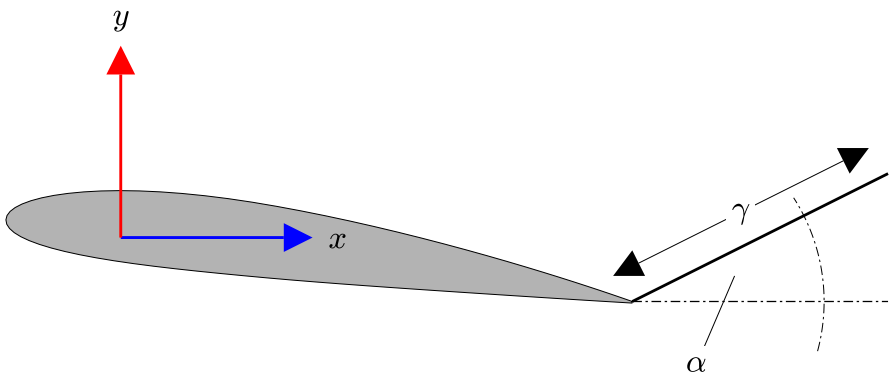


FIGURE 3.7: Nomenclature used to define the shedding geometry: α denotes the angle of the shedding line with respect to the x axis, while γ defines the length of the dipole panel during a single time-step.

In the iterative method, it is assumed that the local flow vector $\vec{u}$ is a function of the variables α and γ, which in turns relate to $\vec{u}$ via:

$$\alpha = \tan^{-1}\left(\frac{v}{u}\right), \quad \gamma = \sqrt{u^2 + v^2}, \tag{3.2.37}$$

where u and v are the projected velocity components in the local two-dimensional cross-section plane (note that $\vec{u}$ is a 3-dimensional vector in general). During the simulation, the correct shedding geometry is computed by first guessing the initial values for α and γ, followed by solving the BEM equations. The total velocity is then computed and the shedding geometry is updated according to Eq.(3.2.37). This process is said to have converged if the change between the updated values and the values from preceded iteration is less than a certain tolerance (the flow chart of this process is illustrated in Figure (3.9)). Since the elements of the influencing matrix implicitly depend on these two parameters, each iteration in the routine will: a) update the influencing matrix of the system, and b) solve the dense matrix equation. For a sufficiently resolved body geometry, the computational overhead is simply too large to ignore. For this reason alone, a different approach was sought in the code.

Basu and Hancock (1978) noted that in an unsteady 2D aerofoil simulation, the shedding line of the nascent wake aligns with the upper or the lower surface tangent depending on the sign of the shed vorticity; in general, when the sign is positive (clock-wise rotation), the shedding line is parallel to the upper surface, whereas if the sign is negative (anti-wise rotation), the shedding line is parallel to the lower surface (see Figure (3.8)). However, since the sign is not known a priori, the correct implementation of such scheme still employs the use of an iterative routine.

One simple approach, which avoids applying the iteration procedure to correct the shedding geometry, is to freeze the parameters α and γ. It is derived from the fact that in the limit as the rate change of the circulation of the blade approaches zero, the flow leaves the trailing edge at the bisector of the two surfaces. This approach has been adopted by various researchers to simplify the calculation of the shedding geometry (see Katz and Plotkin (2001)), and is implemented here as well. The value of γ is appropriately modelled by inferring from accuracy analysis and is currently controlled by a dimensionless input. In the work of Katz and Plotkin, 2001, this is set to 40% of the transition length from one time step to the next.

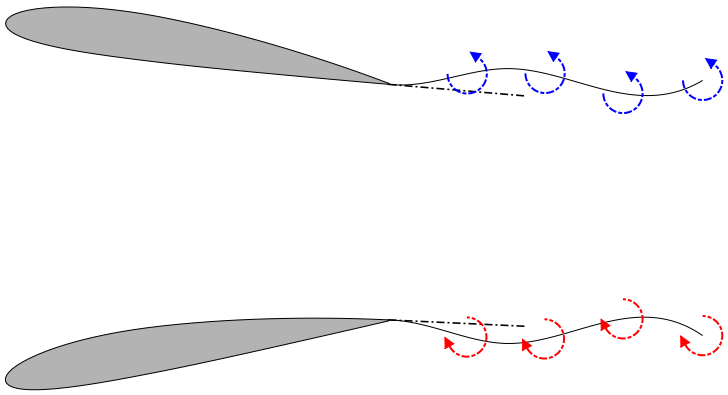


FIGURE 3.8: Schematic diagram showing that the physical alignment of the shedding line is parallel to either the top or bottom surfaces; depending on the sign of the shed vorticity.

3.2.4 Analytical evaluation of the influencing functions and the far field approximations

The task of evaluating the surface integrals as in Eq.(3.2.29)-Eq.(3.2.31) is a highly non-trivial task. The exact derivation can be found in the literature (Hess & Smith, 1967). Sufficed to say, those expressions involve a total of 4 logarithmic and inverse tangent operations, which resolve for the detailed quadrilateral/triangular geometric influence. On account of the large number of field points during simulations, this portion of the computation remains a significant bottleneck.

When $\vec{x}$ is sufficiently far from the panel, it is anticipated that the exact panel geometric detail is less important, thus one could expand the denominator in Eq.(3.2.29) -Eq.(3.2.31) as a Laurent series. The leading order term in the series give rise to the far-field approximations, which take the rather simple forms

$$A_j(\vec{x}) \sim \frac{|S_j|}{4\pi} \frac{\vec{n} \cdot (\vec{y}_j - \vec{x})}{\|\vec{y}_j - \vec{x}\|^3}, \tag{3.2.38}$$

$$B_j(\vec{x}) \sim -\frac{|S_j|}{4\pi} \frac{1}{\|\vec{y}_j - \vec{x}\|}. \tag{3.2.39}$$

where $|S_j|$ denotes the surface area of S_j and $\vec{y}_j$ is the collocation point. Algorithmically, the domain relative to the body is first partitioned into the *near-field* and *far-field* regions. The influence in the near-field region is calculated exactly while the far-field is approximated by Eq.(3.2.38) - Eq.(3.2.39).

The overall accuracy of the computation therefore depends on the partitioned domain. If the partition occurred relatively far from the body, for instance, then it is likely that the reduction in the computational effort will be limited as a large portion of the particles may still remain in the near-field region. Likewise, if the near-field region is too small, then calculation of the neighbouring far field points will generally induce a relative large error. To understand how the accuracy is impacted by the partition, one needs to investigate the error estimate as function of the size of the near-field region. The precise definition of the near-far region is summarized as: given a body composed of a collection of panels, one defines the *minimum box*

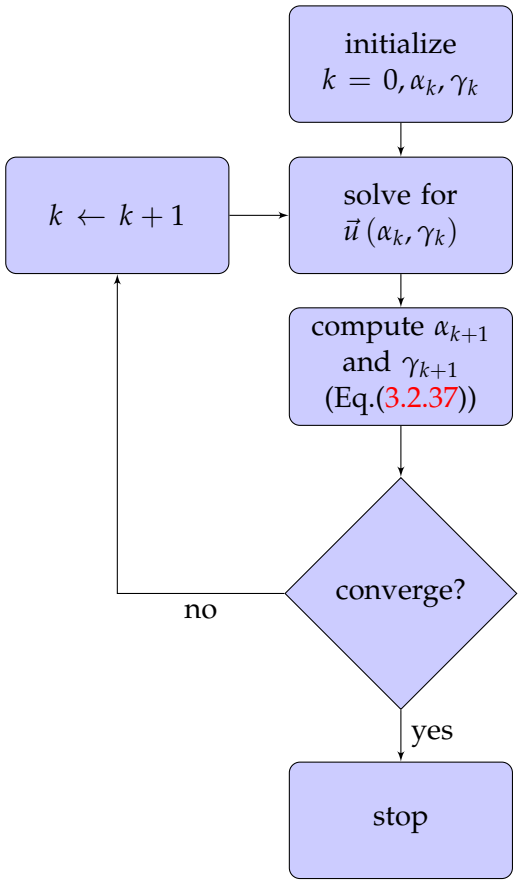


FIGURE 3.9: Iteration flow chart to determine the correct α and γ. Here, k is the iteration counter and α_k means the value of α after the k-th iteration.

as the minimum three-dimensional space that encapsulates the geometry without intercepting with its boundary. Let L be the largest characteristic panel length and f some dimensionless number, the near-field region (or box) is the extension of the minimum box whose 8 corners are outwardly translated by $f \times L$ in each direction. The size of the near-field region is thus controlled and parametrized by varying the value of f (hereafter f is called the *partition parameter*, see also Figure (3.10) for an example). To survey the variation of the error of the far-field approximation, a mesh on one of the faces is generated. The *normalized root-mean squared deviation* (NRMSD) is used to measure the expected deviation from the true values. Specifically, for two distributions g and h, the NRMSD is defined as

$$\mathrm{NRMSD}(g,h) = \sqrt{\frac{\sum_{j=1}^{N}(g_j - h_j)^2}{\sum_{j=1}^{N} h_j^2}}. \tag{3.2.40}$$

The (u, v, w) component of the induction were surveyed. Since the error term in the Laurent series behaves like $\mathcal{O}(r_0^{-5})$ where r_0 is the length between the field point and the near-field centre, it should be expected that the NRMSD should exhibit a similar asymptotic behaviour in f. Indeed, Figure (3.11) depicts a monotonic decreasing trend. Although the code provides the option to set the partition parameter, one should be conscious about the behaviour of the approximation error and how the partition parameter affects the accuracy of the simulation. For the simulation

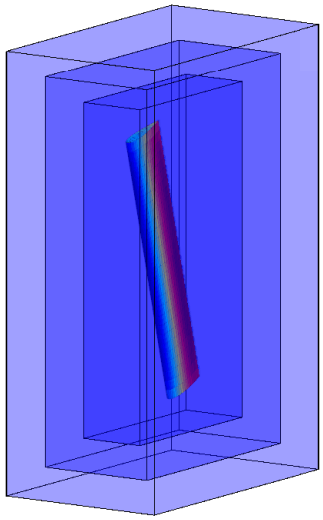


FIGURE 3.10: Near-field regions at different values of $f = 2.5, 5.0$ and 7.5

cases considered in this thesis, $f = 5$ is a reasonable compromise between speed and accuracy.

3.2.5 A multipole expansion of the far-field approximation of the panels

A second level of approximation is explored, which is motivated by the need to accelerate the far-field calculation in cases when fine-grained discretization on the body is performed. The aim is to reduce the original far-field approximation further without sacrificing accuracy too much. For this purpose, a multipole expansion based on the spherical harmonic functions is developed. First, one observes that Eq.(3.2.38) is related to Eq.(3.2.39) via the expression:

$$A_j = -\vec{n}_j \cdot \nabla B_j = \nabla \cdot \left(\frac{|S_j| \, \vec{n}_j}{4\pi} \frac{1}{\|\vec{x} - \vec{y}_j\|} \right) \tag{3.2.41}$$

At a far-field location, the total source and dipole potential are simply the sum of the product between the singular surface values and their kernels:

$$A := \nabla \cdot \vec{\Psi} \tag{3.2.42}$$

$$B := \sum_{j=1}^{N} \sigma_j B_j \tag{3.2.43}$$

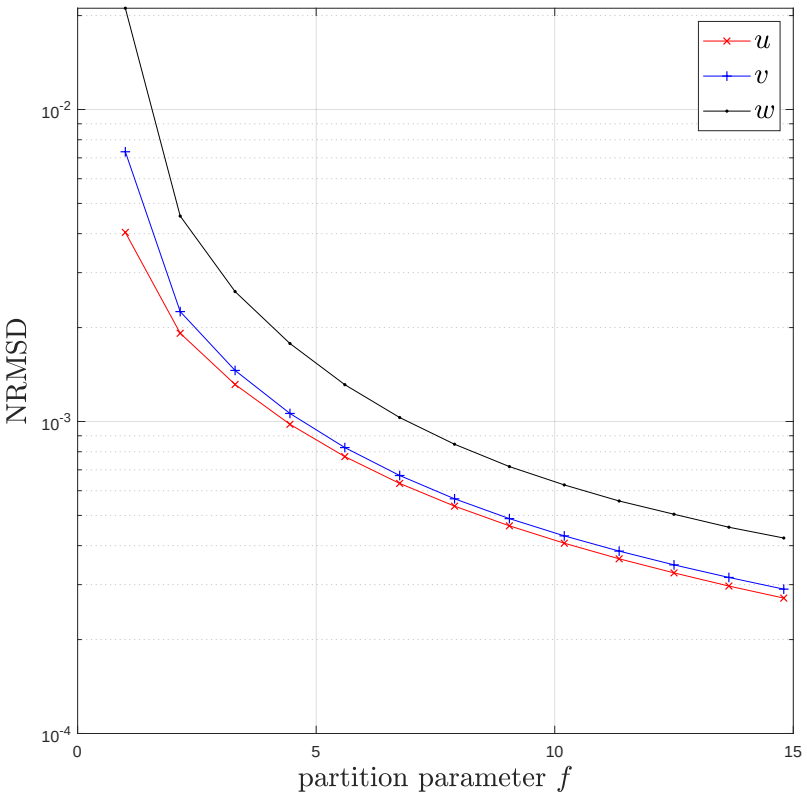


FIGURE 3.11: Normalized root-mean squared deviation of the induction velocity (u, v, w) as function of the partition parameter f.

where

$$\vec{\Psi} = \sum_{j=1}^{N} \frac{\mu_j \left|S_j\right| \vec{n}_j}{4\pi} \frac{1}{\left\|\vec{x} - \vec{y}_j\right\|}. \tag{3.2.44}$$

The idea is to convert the potential as in Eq.(3.2.44) and Eq.(3.2.43) into an expansion of the form:

$$\vec{\Psi}(r, \theta, \phi) = \sum_{n=0}^{\infty} \sum_{m=0}^{n} J_n^m(r, \theta) \vec{S}_n^m(\phi) \tag{3.2.45}$$

$$B(r, \theta, \phi) = \sum_{n=0}^{\infty} \sum_{m=0}^{n} J_n^m(r, \theta) Q_n^m(\phi). \tag{3.2.46}$$

where (r, θ, ϕ) defines the spherical coordinates relative to an expansion center (see Section 4.4.2 for detail). At degree n and order m, the induced velocity due to the source elements is given by

$$\vec{u}_{s,n}^m = \frac{\partial J_n^m}{\partial r} Q_n^m \nabla r + \frac{\partial J_n^m}{\partial \theta} Q_n^m \nabla \theta + J_n^m \frac{\partial Q_n^m}{\partial \phi} \nabla \phi, \tag{3.2.47}$$

Meanwhile, the induced velocity due to the dipole elements requires significantly more effort to derive:

$$\vec{u}^m_{d,n} = \nabla\left(\frac{\partial J^m_n}{\partial r}\right) f_1 + \frac{\partial J^m_n}{\partial r}\nabla f_1 + \nabla\left(\frac{\partial J^m_n}{\partial \theta}\right) f_2 + (\nabla J^m_n)\, f_3 + J^m_n \nabla f_3 \tag{3.2.48}$$

$$f_1 = \vec{S}^m_n \cdot \nabla r, \tag{3.2.49}$$

$$f_2 = \vec{S}^m_n \cdot \nabla \theta, \tag{3.2.50}$$

$$f_3 = \frac{\partial \vec{S}^m_n}{\partial \phi} \cdot \nabla \phi, \tag{3.2.51}$$

$$\nabla f_1 = \left(\nabla r \cdot \frac{\partial \vec{S}^m_n}{\partial \phi}\right) \nabla \phi + r^{-1}\left(\vec{S}^m_n - f_1 \nabla r\right), \tag{3.2.52}$$

$$\nabla f_2 = \frac{1}{r^2 \sin\theta}\left(f_1 \vec{k} + \left(\vec{k}\cdot\vec{S}^m_n\right)\nabla r + \left(\vec{k}\cdot\nabla r\right)\left(\vec{S}^m_n - 3 f_1 \nabla r\right)\right) - \cot\theta f_2 \nabla\theta + \left(\nabla\theta \cdot \frac{\partial \vec{S}^m_n}{\partial \phi}\right)\nabla\phi \tag{3.2.53}$$

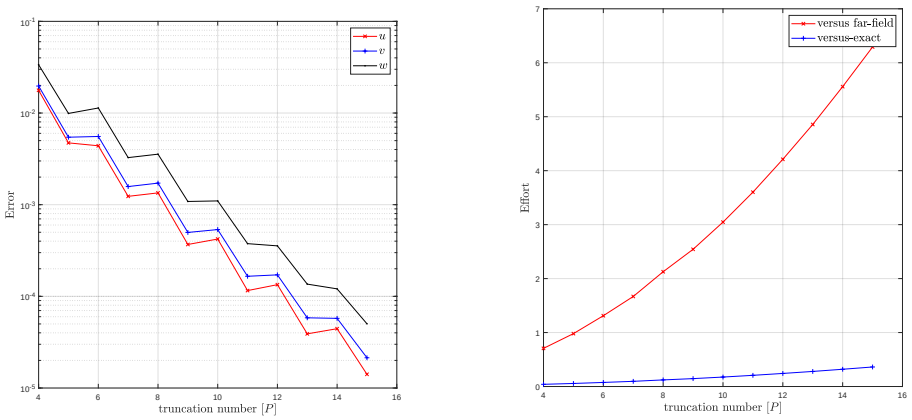
$$\nabla f_3 = -m^2\left(\nabla\phi \cdot \vec{S}^m_n\right)\nabla\phi + \|\nabla\phi\|^2\left(\vec{k} \times \frac{\partial \vec{S}^m_n}{\partial \phi}\right) - 2\left(\vec{k}\cdot\frac{\partial \vec{S}^m_n}{\partial \phi} \times \nabla\phi\right)\nabla\phi. \tag{3.2.54}$$

The strain tensor requires the derivative of the velocity field and thus requires further differentiation of the relevant quantities. Presently, no attempts have been made to derive the exact expression for the strain. Therefore, a central difference scheme is applied in each of the directions.

The error of the multipole expansions is controlled by two parameters in addition of the partition parameter f. They are:

- the truncation number in each expansions, P
- the number of expansions/body, N_b

For the latter item, the code generates N_b number of boxes to cover the body. For each box in the collection, the influences of the containing panels are transformed into the multipole expansions relative to the boxes' centre according to Eq.(3.2.45) and Eq.(3.2.46). Again, the error behaviour with respect to the truncation number P is reported in Figure (3.12a) in which the NRMSD was computed against the far field approximation in Section 3.2.4 at a partition size $f = 5$ and $N_b = 5$. Indeed, the trend in Figure (3.12a) suggest the accuracy is improved exponentially at a larger value of P, but this comes at the expense of increasing the computational *effort* (Figure (3.12b)), which is defined by the ratio between the wall-clock time of the evaluation of the multipole expansions and the direct evaluation of the far-field approximation. Thus if the computational effort is larger than 1, there is no real speed advantage of using the multipole approach. Interestingly, there is no notable gain in accuracy if an odd value of the truncation number was used. In fact, there appears to be a slight deterioration of the result. It is not known whether this is a feature of the spherical harmonic basis functions.



(A) (B)

FIGURE 3.12: (A) shows the NRMSD being plotted against the truncation number P for the induction field at $f = 5$, $N_b = 5$ and (B) shows the computational *effort* versus P

3.3 BEM and VPM coupling

In attached flow, vorticity is shed to the fluid domain at the trailing edges of the solid body. In the BEM calculation, this physical phenomenon is manifested in the introduction of nascent wake dipole panels. To properly model the vorticity shedding mechanism, a forward conversion procedure between the dipole sheet and the vortex particles is required. Before this conversion procedure can take place, however, one recalls that in the low-order modelling, the panels are assumed piece-wise constant, which, if untreated, makes the induction field close to the wake to exhibit notable numerical discontinuity. One way to reduce this apparent numerical artefact is to apply interpolation for a discrete set of dipole values. Indeed, the current code makes use of a simple *bilinear interpolation* technique which applies a linear function to approximate the continuity between two consecutive panels. Once the interpolation is done, the forward conversion is performed using the Hess' equivalent principle (Hess and Smith, 1967), which relates the induction field due to a dipole panel with an arbitrary dipole distribution to a *surface* vortex distribution plus a boundary term.

3.3.1 Hess' equivalent relationship

Let $\vec{u}_d$ denote the induction velocity due to a dipole panel with a surface dipole distribution $\mu(\vec{y})$ and outward normal $\vec{n}(\vec{y})$, where $\vec{y}$ denotes the surface point on S. $\vec{u}_d$ is obtained by taking the grad of a general dipole potential. i.e.

$$\vec{u}_d = \nabla \left(\nabla \cdot \int_S \frac{\mu(\vec{y})\,\vec{n}(\vec{y})}{4\pi} \frac{1}{\|\vec{x}-\vec{y}\|} d\vec{y} \right). \tag{3.3.1}$$

By using the Stoke's theorem, one could show that the following holds:

$$\vec{u}_d = \frac{1}{4\pi} \int_S (\vec{n} \times \nabla\mu) \times \frac{\vec{x}-\vec{y}}{\|\vec{x}-\vec{y}\|^3} d\vec{y} + \frac{1}{4\pi} \int_{\partial S} \mu \frac{d\vec{y} \times (\vec{x}-\vec{y})}{\|\vec{x}-\vec{y}\|^3}. \tag{3.3.2}$$

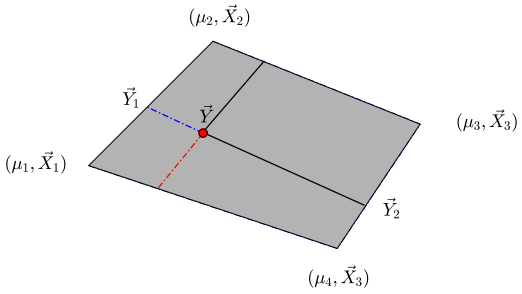


FIGURE 3.13: Nomenclature used in the bilinear interpolation of a convex quadrilateral dipole panel. The corner points are denoted by the tuple $\left(\mu_i, \vec{X}_i\right)$.

The first term on the right corresponds to the surface vortex distribution with pointwise vorticity vector assigned by $\vec{n} \times \nabla\mu$, while the boundary term is discretized as line vortices around the surface boundary ∂S. The particle conversion thus relies on applying Eq.(3.3.2) to a collection of piece-wise constant dipole panels, which has to be interpolated across the inter-panel boundaries in order to achieve a smooth discretization of the particle field on the dipole sheet.

3.3.2 Bilinear interpolation and computation of $\nabla\mu$

Fast and reliable computation of $\nabla\mu$ is essential to the accuracy of the VPM as it directly controls the amount of vorticity entering to the flow. Despite the fact that numerous investigators have included this process in their works (Wang et al., 2018), they have not included the implementation detail. For this reason, we have developed an accurate scheme for the evaluation of $\nabla\mu$. The accuracy is limited only to the bilinear interpolation. To demonstrate the technique, we consider a quadrilateral panel whose corner points are assumed coplanar. Furthermore, the dipole values are known at each of the corner point. For reason to be discussed later, we imposed convexity of the points. Hence, a convex quadrilateral dipole panel is uniquely defined by specifying the corner data, which shall be denoted by $\left(\mu_i, \vec{X}_i\right)$ for $i = 1, 2, 3, 4$.

Without loss of generality, let us suppose that the points are distributed in a clock-wise orientation. For this particular panel configuration, the bilinear interpolation requires a compact way to parametrize the surface, which can be done as follows: First, we pick a point $\vec{Y}_1$ on the line segment between $\vec{X}_1$ and $\vec{X}_2$, and we note this point splits the line by a ratio s where $s = \left\|\vec{Y}_1 - \vec{X}_1\right\| / \left\|\vec{X}_2 - \vec{X}_1\right\|$. For this value of s, one may compute the point $\vec{Y}_2$ on the line segment between $\vec{X}_4$ and $\vec{X}_3$. By the convexity assumption, points on the line segment between $\vec{Y}_1$ and $\vec{Y}_2$ are completely confined in the panel. In particular, if Y is one of such points, then one could define a ratio t such that $t = \left\|\vec{Y} - \vec{Y}_1\right\| / \left\|\vec{Y}_2 - \vec{Y}_1\right\|$. Hence, there is a one-to-one correspondence between an interior point $\vec{Y}$ and the pair (s, t) (see Figure (3.13)).

Knowing this, it is possible to construct the bilinear interpolation formula for an arbitrary point $\vec{Y}$ by applying two linear interpolations sequentially to the s and t variables. The end result is a quadratic function that takes the particularly simple

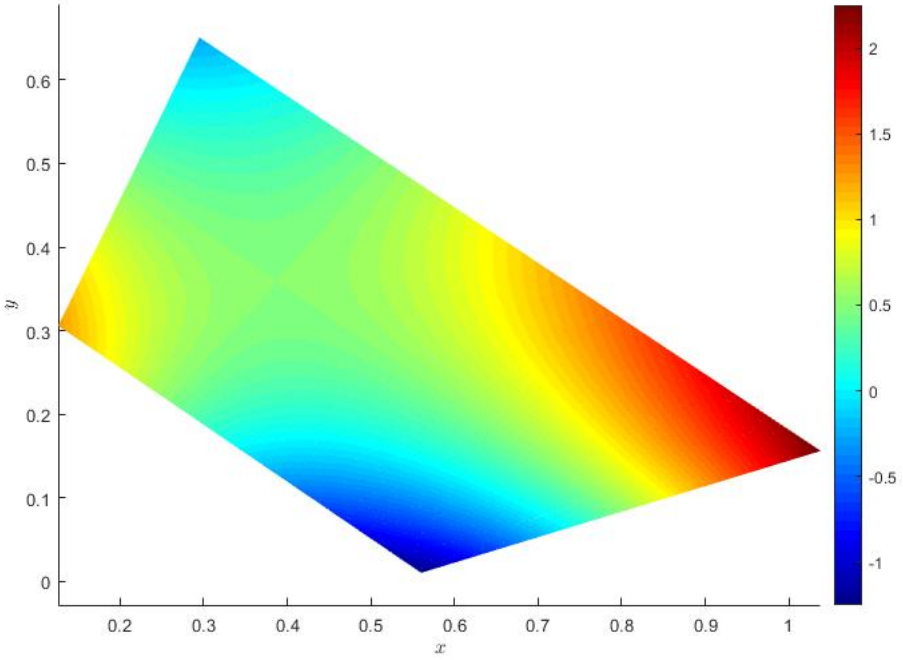


FIGURE 3.14: Example of the bilinear interpolation with some random corner values.

expression

$$\vec{Y} = \left((1-s)\,\vec{X}_1 + s\vec{X}_2\right)(1-t) + \left((1-s)\,\vec{X}_4 + s\vec{X}_3\right)t. \tag{3.3.3}$$

where $s \in [0,1]$ and $t \in [0,1]$. By the same token, the dipole strength at $\vec{Y}(s,t)$ is interpolated as

$$\mu(s,t) = ((1-s)\,\mu_1 + s\mu_2)(1-t) + ((1-s)\,\mu_4 + s\mu_3)\,t. \tag{3.3.4}$$

This particular simple interpolating scheme is capable to handle panel distortion in the projected two-dimensional plane (see Figure (3.14)), as long as the deviation in the normal direction is small compared to the other two length-scales. Having defined the parametrization of the panel, our next task is to compute $\nabla_S\mu$, which is related to $\nabla\mu$ as follows:

$$\nabla_S\mu = \nabla\mu - \vec{n}\vec{n}\cdot\nabla\mu. \tag{3.3.5}$$

where $\vec{n}$ is the normal of the panel.

Let $\vec{p}_1$ and $\vec{p}_2$ be two vectors such that $(\vec{n}, \vec{p}_1, \vec{p}_2)$ forms an orthonormal basis, so one may express the explicit form of $\nabla_S\mu$ as

$$\nabla_S\mu = \frac{\partial\mu}{\partial p_1}\vec{p}_1 + \frac{\partial\mu}{\partial p_2}\vec{p}_2. \tag{3.3.6}$$

By definition,

$$\frac{\partial \mu}{\partial p_i} = \lim_{\delta \to 0} \left(\frac{\mu\left(\vec{X} + \delta \vec{p}_i\right) - \mu\left(\vec{X}\right)}{\delta} \right), \tag{3.3.7}$$

where $\vec{X} = (x, y, z)$ is an interior point. Generally, the explicit form of $\mu\left(\vec{X}\right)$ is not available, therefore it is not possible to evaluate $\partial\mu/\partial p_i$ via Eq.(3.3.7). In the proximity of the point $\vec{X}$, however, we may expand s and t along the path $L : \vec{X} + \delta\vec{p}_i$, so as $\delta \to 0$, we have

$$t(\delta) \sim t^* + \frac{\partial t}{\partial \delta}\delta, \tag{3.3.8}$$

$$s(\delta) \sim s^* + \frac{\partial s}{\partial \delta}\delta, \tag{3.3.9}$$

where (s^*, t^*) satisfy the equation

$$\vec{X} = \vec{Y}(s^*, t^*). \tag{3.3.10}$$

Setting

$$\begin{aligned} \vec{X} + \delta\vec{p}_i &= \vec{Y}\left(s^* + \frac{\partial s}{\partial \delta}\delta, t^* + \frac{\partial t}{\partial \delta}\delta\right), \\ &= \vec{Y}(s^*, t^*) + \left(\frac{\partial \vec{Y}}{\partial s}\frac{\partial s}{\partial \delta} + \frac{\partial \vec{Y}}{\partial t}\frac{\partial t}{\partial \delta}\right)\delta + \mathcal{O}\left(\delta^2\right), \end{aligned}$$

and upon comparing the coefficients, it follows that $\partial t/\partial\delta$ and $\partial s/\partial\delta$ satisfy the equation

$$\vec{p}_i = \frac{\partial \vec{Y}}{\partial s}\frac{\partial s}{\partial \delta} + \frac{\partial \vec{Y}}{\partial t}\frac{\partial t}{\partial \delta}. \tag{3.3.11}$$

In the limit $\delta \to 0$, Eq.(3.3.7) may be replaced by

$$\frac{\partial \mu}{\partial p_i} = \lim_{\delta \to 0}\left(\frac{\mu(s(\delta), t(\delta)) - \mu(s^*, t^*)}{\delta}\right) = \frac{\partial \mu}{\partial s}\frac{\partial s}{\partial \delta} + \frac{\partial \mu}{\partial t}\frac{\partial t}{\partial \delta}.$$

Finally, solving Eq.(3.3.11) yields the following expressions for $\partial s/\partial\delta$ and $\partial t/\partial\delta$:

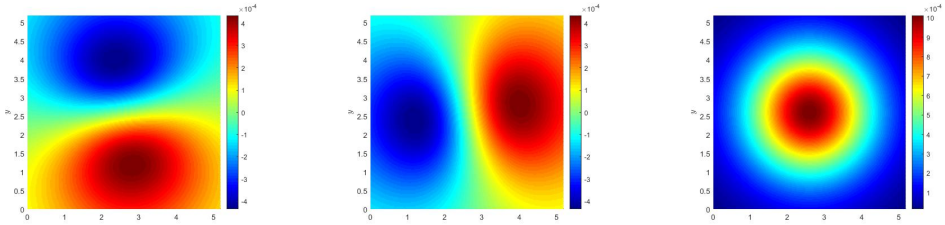
$$\frac{\partial s}{\partial \delta} = \frac{1}{\lambda}\left[\left\|\frac{\partial \vec{Y}}{\partial t}\right\|^2\left(\vec{p}_i \cdot \frac{\partial \vec{Y}}{\partial s}\right) - \left(\frac{\partial \vec{Y}}{\partial s}\cdot\frac{\partial \vec{Y}}{\partial t}\right)\left(\vec{p}_i \cdot \frac{\partial \vec{Y}}{\partial t}\right)\right], \tag{3.3.12}$$

$$\frac{\partial t}{\partial \delta} = \frac{1}{\lambda}\left[\left\|\frac{\partial \vec{Y}}{\partial s}\right\|^2\left(\vec{p}_i \cdot \frac{\partial \vec{Y}}{\partial t}\right) - \left(\frac{\partial \vec{Y}}{\partial s}\cdot\frac{\partial \vec{Y}}{\partial t}\right)\left(\vec{p}_i \cdot \frac{\partial \vec{Y}}{\partial s}\right)\right], \tag{3.3.13}$$

$$\lambda := \left\|\frac{\partial \vec{Y}}{\partial t} \times \frac{\partial \vec{Y}}{\partial s}\right\|^2. \tag{3.3.14}$$

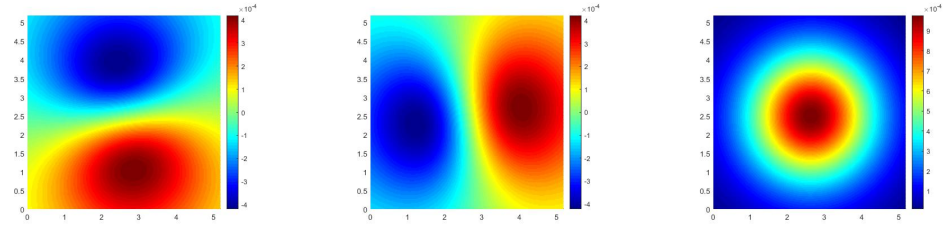
It is worth noting that the solution only exists if $\lambda > 0$ with equality only in situation where two adjacent panel edges are aligned parallel with each other. This extreme case is not handled in the code and thus one should exercise with caution when generating the body and wake geometry data.

For the purpose of validating the conversion procedure, it is useful to compare



(A) (B) (C)

FIGURE 3.15: velocity field due to the far-field approximation of the dipole panel. a) x component, b) y component and c) z component.



(A) (B) (C)

FIGURE 3.16: velocity field due to the particle discretization with a fixed core radius $\epsilon =$ `1E-3`. a) x component, b) y component and c) z component.

the induction field between the Biot-Savart induction and the far field approximation in Section 3.2.4. Recall that for a dipole panel, the far field velocity is obtained by taking the grad of Eq.(3.2.38), i.e.

$$\vec{u}_{\text{far}}\left(\vec{x}\right) = \nabla_{\vec{x}}\left(\frac{\bar{\mu}A}{4\pi}\frac{\vec{n}\cdot\left(\vec{y}-\vec{x}\right)}{\left\|\vec{y}-\vec{x}\right\|^3}\right), \tag{3.3.15}$$

where A is the area of the panel, $\vec{y}$ is the collocation point and $\bar{\mu}$ is given by

$$\bar{\mu} = \frac{1}{A}\int_0^1\int_0^1 \mu\left(s,t\right)\left\|\frac{\partial\vec{Y}}{\partial s}\times\frac{\partial\vec{Y}}{\partial t}\right\| dsdt. \tag{3.3.16}$$

Figure (3.15) and Figure (3.16) illustrate the velocity induction field at a survey plane located above the panel (along the normal direction). The plane is located at a distance $5.5L$ from the collocation point where L is the characteristic panel length. Despite some discrepancies in terms of the obtained maximum values, the overall agreement is satisfactory. Also, it is worth mentioning that although Eq.(3.3.6) is defined through the prescription of the direction vectors, one should expect that the end result should be independent on the choice of the vectors; as long as they form an orthonormal basis for the panel.

Once $\nabla_S\mu$ is determined, the panel is discretized by a set of surface vortices as well as edge vorticies according to the Hess' equivalent principle. For the surface vortices, the vorticity value is assigned to the vortex particles according to the formula

$$\vec{\omega} = \vec{n}\times\nabla_S\mu. \tag{3.3.17}$$

Meanwhile, the initial core assigned to the vortices remains an arbitrary input at this point. But principally, it should depend on the boundary thickness of the solid bodies that generate them.

3.3.3 A robust least square fit for interpolating the dipole values

In the particle conversion algorithm, a set of dipole panels are transformed to a cluster of particles representing the effect of the thin boundary layer in the wake. To do that, the dipole strengths, which are assumed constant on each panel, are first interpolated to the nodes followed by fitting a bilinear interpolation approach to reconstruct the linear variation across the panel surfaces. However, it was found that naive implementation produces highly oscillatory results. Specifically, suppose $\mu_i, i = 0, 1, \ldots, N-1$ denote the constant dipole values associated with the panels with index i. Assuming a linear variation between the nodes, it can then be shown that a simple technique can be constructed as follows:

$$F(i+1) = 2\mu_i - F(i), \tag{3.3.18}$$

where F is the node value with $F(0) = 0$. In general, the span-wise distribution of μ is characterised by sharp changes towards the tips of the blade, thus it is anticipated that the values might be erroneous in those regions even if a fine discretization is performed. Inferring from the theoretical result for the span-wise distribution of μ for the elliptical wing (Katz and Plotkin, 2001), it is clear that the gradient is singular at the tips. This could be the reason that the dipole values are prone to error in those regions. Indeed, if one is to perform Eq.(3.3.18) on such data points, the result is plagued by high frequency noise. An example of such behaviour is given in Figure (3.17) where the calculated dipole distribution failed to converge to zero even for a refined discretization towards the tips of the elliptical wing.

To avoid such cases, a non-linear regression model is developed with the fitting curve $f(x, \vec{a})$ given by

$$f(x, \vec{a}) = \sqrt{x(s-x)}\, Q_n(x, \vec{a}), \tag{3.3.19}$$

where s is the blade span, Q_n is a polynomial with degree n and $\vec{a}$ is the coefficient vector. The choice for the square root function is essential to mimic the steep gradient near the tip regions - keeping inline with the theoretical prediction. A robust variant of the least-squares was employed to circumvent the limitation of the traditional least squares regression, i.e. sensitive to outliers. For this purpose, an iteratively reweighted least squares method was used that minimizes the weighted loss function of the form:

$$E = \sum_{i=0}^{N-1} w_i(\vec{a}) \left| f(x_i, \vec{a}) - \mu_i \right|^2. \tag{3.3.20}$$

Here, the weight is defined as $w_i := 1/\max\left[\delta, \left|f(x_i, \vec{a}) - \mu_i\right|\right]$ and δ is a small non-zero positive constant introduced to prevent division by zero. Since the weight is a function of the coefficient vector, an iterative approach is used to solve the minimization problem. Figure (3.17) shows that the robust fit seems to produce excellent result especially near the tip region where the data are erroneous.

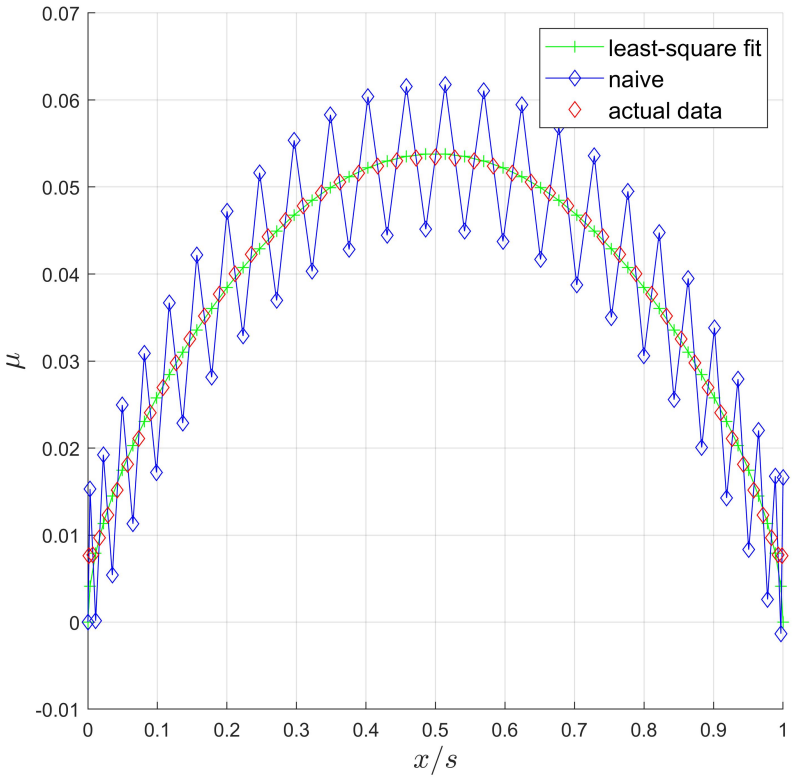


FIGURE 3.17: Comparison of μ for an elliptical wing between the naive approach Eq.(3.3.18) and the robust least square fit with the fitting function given by Eq.(3.3.19) for an elliptical wing.

3.4 Chapter conclusion

Under the assumption of high Reynolds number flow condition, the Euler's equations are solved by the introduction of the BEM approach. The new proposed method involves constructing a velocity scalar potential function, which satisfies the appropriate boundary and far-field conditions. It was shown that the potential is harmonic and a general solution is constructed by prescribing the surface singular distributions (σ and μ) as well as an internal potential for closed geometry. A new low-order BEM was developed, which uses the Dirichlet formulation for enforcing the boundary condition at the collocation.

In addition, we have presented the mathematical theory for viscous wake flow, in which the vorticity field is discretized by a set of vorticity-carrying particles. This allows a fully Lagrangian approach to solve the non-linear advection term in the Navier-Stokes equations. To avoid the singular nature of the Green's function, the field is mollified by treating the particles as blobs with a finite core region. This allows the diffusion to be modelled by a deterministic scheme (PSE).

Chapter 4

Fast multipole methods

The essence of the current work is the development of a new efficient calculation procedure to compute the Biot Savart law for a large set of vortex particles in the flow domain. This chapter explores the Fast Multipole Method (FMM) in an attempt to reduce the quadratic scaling of the N-body problem to a linear scaling. It is noted that such a reduction requires an efficient data structure for storing the interaction information. For this purpose, a novel data construction algorithm has been proposed and developed which proceeds to sort the particle data in a backward sweep manner. In the 3D implementation, an analytical expression for the strain tensor has been derived in terms of the real spherical harmonic basis functions. The use of the real basis functions is crucial for minimizing the GPU overhead in the code. For this purpose, several mathematical expressions have been derived to facilitate the complex to real conversion.

4.1 Introduction to the fast multipole methods (FMM)

The needs to apply a fast summation algorithm arise from an important class of problems in physics - the N-body problems. Applications of such problems can be found in a variety of scientific and engineering principles, for example, the Colombic interaction of celestial objects in astrophysics, the Lagrangian evolution of the vortex particles in fluid dynamics, the molecular dynamics in chemical systems and so forth. Typically, in these types of problems the solutions require the repeated evaluations of a large sum of the form:

$$\Psi\left(\vec{x}_j\right) = \sum_{\substack{i=0 \\ i\neq j}}^{N} \phi\left(\vec{x}_j - \vec{x}_i\right), \qquad j = 1, 2, ..., N \tag{4.1.1}$$

where ϕ is some scalar potential function. Direct evaluation of Eq.(4.1.1) for each position/particle requires $\mathcal{O}\left(N^2\right)$ operations and N is generally very large. Although such a scheme is numerically simple but the quadratic dependency on N often demands unrealistic load on computing power. Fast summation techniques are an important class of techniques that mitigate the quadratic asymptotic and thus they find applications across many disciplines of science and applied mathematics. The original attempt of reducing the quadratic dependency was due to Barnes and Hut (1986) in their dynamic simulation of the evolution of stars. The novelty in their work includes the approach to efficiently construct a hierarchically divided domain in which the computational elements are sorted. By the introduction of a judiciously designed interaction list, which contains the information necessary to resolve communication between the field particles and their far-field, they were able to represent the far field influence as a multipole expansion series. Interaction between a field particle

and those result from the far field is computed, not on a pair-wise basis but, by simply evaluating the resulting expansions. Accounting for the cost of constructing the data structure, the overall complexity of the scheme is $\mathcal{O}(N \log N)$. Fast summation techniques that follow this school of thought are sometimes referred to as *tree-code*.

The success of the Barnes-Hut algorithm led to development of several fast summation techniques that are still in use for many physics simulation codes. In particular, the fast multipole method, or FMM, introduces an extra layer of complexity - the local representation. Essentially, in addition of the original *tree code*, the multipole expansions of the far-field regions are successively converted and translated to a *local* expansion. This allows the expansions from multiple far-field regions to be congregated at a local level to create an effective local potential. More specifically, all aspects of the FMM routine can be categorically grouped into 5 main components (Figure (4.1)), these are summarized as follows:

- *Data structure*: The source particles are recursively sorted by dividing the computational domain into a *quadtree* (or *Octree* in 3D).
- *Initial expansion*: In each sub-domain of certain kind, the particle cluster is converted to a multipole expansion.
- *Upward pass*: Recursively, the multipole expansion of each sub-domain is translated and combined to its parent's expansion. The resulting expansion represents the influence of the source particles of its parent.
- *Downward pass*: For each sub-domain, the multipole expansion of its *far-field* region is converted to a local expansion. Then, recursively, the local expansion of the sub-domain is translated and combined to its children's local expansion.
- *Final evaluation*: For each field point, one computes the influence of its neighbour directly while the far-field influence is evaluated on the local-expansion of the sub-domain that contains the field point.

Notably, one distinguishes the tree code, such as those from the Barnes-Hut algorithm, and the FMM algorithm by the inclusion of the *Downward pass* step. Crucially, it is precisely because of this step that allows the operation count of the FMM to be reduced from $\mathcal{O}(N \log N)$ to $\mathcal{O}(N)$. In the above, we have introduced several concepts which are yet to be defined, such as the notion of a *parent*, *children* and *neighbour*. These will be subsequently discussed in Section 4.2.

Currently, both the 2-D and the 3-D FMM algorithms have been developed on graphics processing units (GPU). Although much of the theories are based on the early literature (for example, the 2D implementation is based on the work of Carrier et al. (1988) and L. Greengard and Rokhlin (1987), while the three-dimensional approach is primarily based on L. Greengard and Rokhlin (1997)), the actual implementation draws many similarities from more recent works such as those from Gumerov and Duraiswami (2008), Hu et al. (2013), and Yokota et al. (2009). The implementation of the FMM code on the GPU brings extra complexities to the already complicated FMM structure. One such concern is the parallelization of what would be a highly serialized structure in the traditional FMM formulation. To overcome this issue, we have employed a space-filling curve to construct the FMM tree in a numerically efficient way. Moreover, one notable distinction between our codes and that of the literature is that we have adapted the FMM methodology specifically for wind turbine applications under a unified framework with the modelling approach outlined in Chapter 3. For the sake of completeness, this chapter reviews and

presents the underlying mathematical theories and computer algorithms needed to construct an efficient FMM routine.

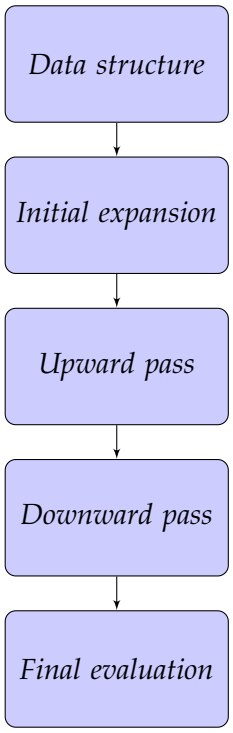


FIGURE 4.1: Typical components of the FMM routine.

4.2 Data structure

Consider a set of source particles with index set J. Each particle in the set is identified by the index $j \in J$ and parametrized by the tuple $(\vec{\alpha}_j, \vec{x}_j)$, where $\vec{\alpha}_j$ is the vector strength and $\vec{x}_j$ is the position of the particle (in 2D, $\vec{\alpha}_j$ is taken to be a scalar). Without loss of generality, we assume that the positions of the particles are scaled to fit within a unit-square (2D) or a unit-cube (3D) located at the origin. So each component in the position vector is greater than zero and less than unity. If the particles are not already normalized, one simply applies the normalization operation:

$$\vec{x}_j \leftarrow \frac{\vec{x}_j - \vec{x}_{\min}}{D}. \tag{4.2.1}$$

where $\vec{x}_{\min} := \min_{j \in J} (\vec{x}_j)$ (note that the minimization is conducted on a component-wise basis) and D is a scalar chosen to ensure that the norm of all normalized positions are within the square (cube). Since the multipole (or local) expansions rely on the truncated Taylor series, one has to make sure that the far-field region for a given source point is within the radius of convergence of such series. To make this concise, we introduce the notion of a *well-separated set*. Let $X = \{\vec{x}_j, j = 1, 2, ..., N\}$ and $Y = \{\vec{y}_j, j = 1, 2, ..., M\}$ denote two sets of Cartesian points such that

$$|\vec{x}_j - \vec{x}_0| < r \qquad \text{for } i = 1, 2, ..., N \tag{4.2.2a}$$
$$|\vec{y}_j - \vec{y}_0| < r \qquad \text{for } j = 1, 2, ..., M \tag{4.2.2b}$$
$$|\vec{x}_0 - \vec{y}_0| > 3r, \tag{4.2.2c}$$

where $\vec{x}_0$ and $\vec{y}_0$ denote the centre of the balls that contain the set X and Y, respectively. Here, r is the common radius of the balls (see Figure (4.2)).

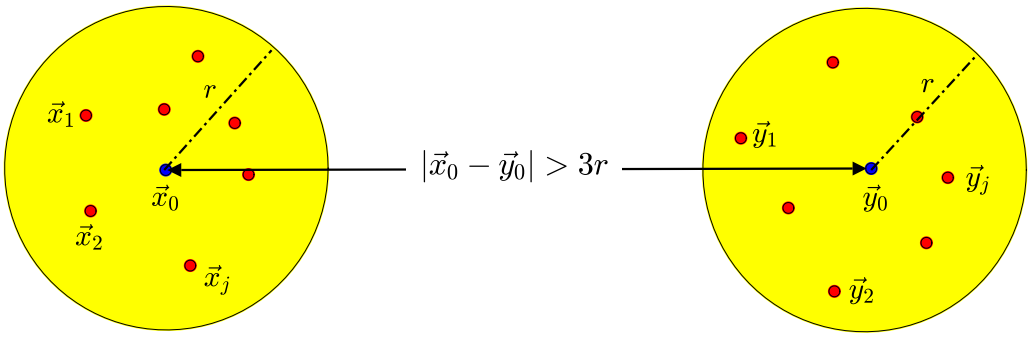


FIGURE 4.2: The sets X and Y are well-separated if the distance between the open balls are separated by at least three times the radius.

Now, if $B(X,r)$ and $B(Y,r)$ denote the ball for the set X and Y, respectively, then for any $\vec{x} \in B(X,r)$, we require that the multipole expansion due to the influence of the sources in Y to be convergent. Likewise, for any $\vec{x} \in B(Y,r)$, convergence needs to be assured for the expansion due to the source influence in X. For the forms of the expansion considered in this thesis, this property is always assumed. Therefore, in consideration of the convergence condition, it is absolutely imperative to consider the type of data structure that respects the convergence property.

For the mapping given by Eq.(4.2.1), it is readily seen that a natural choice for the data structure is a *quadtree*, in which the domain (box) is recursively divided into 4 smaller sub-domains of equal size (sub-boxes). Specifically, we denote D_0 as the initial domain at refinement level 0. After the first refinement, D_0 is partitioned into

$$D_0 = \bigcup_{j=0}^{3} D_{0,j}. \tag{4.2.3}$$

where $D_{0,j}$ denotes the j-th sub-box that results from D_0.

The $D_{0,j}$'s are referred to as the *children* of D_0 (note that in 3D, the original domain would have given birth to 8 children boxes), and D_0 is called the *parent* node of $D_{0,j}$. Proceeding from this line of thought, one can see that at the end of the second refinement level, each of the 4 children would result in a division of 4 further children boxes; resulting in a total of 16 boxes at this level. To label those, we denote the j-th children of the i-th parent at refinement level 1 as $D_{0,i,j}$. Generalizing this argument for the k-th refinement, the children boxes at refinement level k admit a unique identification in the form $D_{0,i_1,i_2,\ldots,i_k}$, where each subscript index $i_p \in \{0,1,3\}$. More generally, if d is the dimension of the space, then 2^d-tree can be constructed whose children boxes at refinement k can be labelled by $D_{0,i_1,i_2,\ldots,i_k}$, with each i_p takes values from the set $\{0,1,\ldots,2^d-1\}$.

It should be noted that the number of boxes N_{total} at the end of refinement k is given by

$$N_{\text{total}} = 1 + 4 + 4^2 + \cdots + 4^k = \frac{1}{3}\left(4^{k+1} - 1\right). \tag{4.2.4}$$

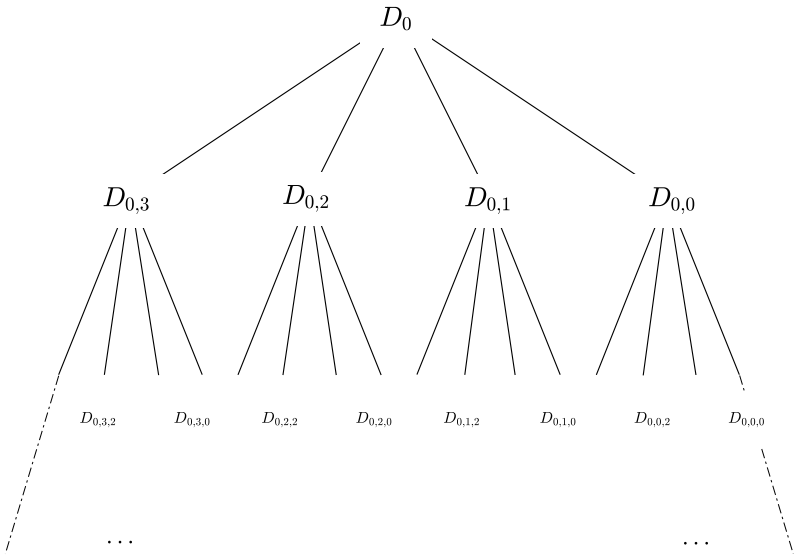


FIGURE 4.3: Schematic diagram showing that the initial 2D domain D_0 is partitioned into the union of $D_{0,0}$, $D_{0,1}$, $D_{0,2}$ and $D_{0,3}$. Children boxes of subsequent refinements are conveniently identified by introducing the multi-dimensional index j such that $j = [0, i_1, i_2, \ldots i_n]$.

The corresponding result for the 3D case is $N_{\text{total}} = \left(8^{k+1} - 1\right)/7$. One notes that N_{total} has an unfavourable exponential dependence on k, which means that if the refinement level increases without bound, the cost of constructing the tree increases exponentially, which in most cases is simply not acceptable. In practical terms, it is seldom the case that the particles would occupy the space uniformly (some boxes may not even contain any source particles), so depending on the particle distribution, one can employ an adaptive control strategy to refine only those boxes that meet certain division criteria. It is customary to introduce the parameter s, which is the maximum number of source particles in any given *childless* box. Here the box is called childless if it is non-empty and contains fewer than s source particles, otherwise it is referred to as a *parent* box. For a given multi-dimensional index $j = [0, i_1, i_2, ...i_n]$, which returns a reference to a box at refinement level n, Algorithm 4.1 checks whether the box is empty (note that a box is empty if it does not contain any source particles) and proceeds to divide the box into 2^d sub-boxes if it is not. The output is a set of box references that will be appended to the tree's division list. The idea is that by actively maintaining the division list, one can exhaust every boxes in the domain (see Algorithm 4.2 for the full construction). At the end of Algorithm 4.2, the tree will: (a) produce a Boolean array to mark which boxes are empty and (b) resolve all parental connections among non-empty boxes and spatially sort the particle data. The next phase of the processing involves defining what we mean by far-field in the FMM setting.

For demonstration purposes, let us apply the tree-construction algorithm to two distributions. Figure (4.5) illustrates the result of the initial domain after the division process for a uniform distribution of the source particles. Here, the number of source

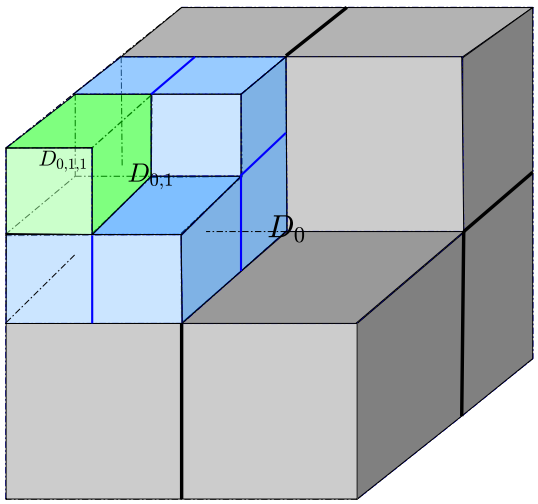


FIGURE 4.4: Illustration of how the cube is recursively divided in 3D. The resulting data structure is now an Octree.

particles was set to 500 and the division parameter was set to $s = 5$. Similarly, Figure (4.6) demonstrates the outcome of the algorithm applied to a non-uniform particle distribution in which the particles are constrained in an annular region. In the first instance, the code requires 5 refinement levels to resolve the distribution while in the latter case an additional level was performed. Notice, however, that the numbers of childless boxes for both distributions are more or less equal (being 200 and 204, respectively). But as we shall see later that in general the FMM always performs better for highly non-uniform distributions.

Once the data structure has established, one needs to introduce a hierarchical structure that defines, in a systematic sense, the notion of near and far-fields. We follow the idea introduced by L. Greengard and Rokhlin (1987) to characterize this notion formally. For each box in the tree, we maintain a set of 4 interaction lists. These are labelled as *List 1*, *List 2*, *List 3* and *List 4*. In addition, we introduce the idea of a *colleague* and an *associate*. Specifically, let b denote the reference of a box at refinement level k, then:

- A *colleague* of b is a non-empty box that results from the same refinement level as b and is also shared a common boundary with b; excluding b itself. Namely, b cannot be it's own colleague.

- An *associate* of b is a non-empty box that shares a common boundary with b and whose refinement level is smaller or equal to b. In other words, the associate box must be at least as small as b. Furthermore, we allow b to be its own associate.

- *List 1* of b is a set of box references that consist of b itself and all other childless non-empty boxes that share a common boundary with b. This list, however, is empty if b is a parent box. The *neighbours* of b are thus defined as the distinct boxes in the List 1 minus b.

- *List 2* of b is a set of box references formed by the non-empty children of the colleagues of b's parent that are well-separated from b (i.e. Eq.(4.2.2)).

Algorithm 4.1: Divide Box

input : reference of the tree object (tree), a multi-dimensional box index (j) and the division parameter (s)
output: reference to the modified tree object, tree

```
// Obtain the box reference from tree
B = tree.getBoxReferenceFromIndex(j)
if B's parent is not empty then
    B.getParticleDataFromParent()
// exit early if B is empty
if B is empty then
    return

if number of source particles in B exceeds s then
    // The splitBox() function divides the box B and performs a preliminary
       test to ascertain whether each children is empty
    B.splitBox()
    for each children box C in B do
        if C is non-empty then
            tree.appendToDivisionList(C)
            // here, the reference of box C is added to the tree's division list
```

- *List 3* of b consists of all non-empty descendants of b's colleagues whose parents share a common boundary with b but they themselves do not. This list is empty if b is a parent box.
- *List 4* of b consists of all non-empty box references whose *List 3* contains b.

By examining the definition carefully, it is a simple exercise to deduce the relative size of the boxes in the Lists. For instance, all boxes in List 2 must be the same size as b, whereas boxes in List 3 are strictly smaller since they are derived from higher refinement levels. This way, one can clarify succinctly the definition of the near- and far-field regions. For instance, the near-field of a childless box is defined to be the union of all boxes in its List 1 set. Likewise, the far-field regions relative to the box are hierarchically divided among its List 2, List 3 and List 4.

As well as providing the clarification we need, the well-separateness property can also be showed to be respected. The next task is to populate the Lists for all boxes in the domain. This is accomplished by a two-step approach. First, one has to resolve the associate list. During this stage, the tree is traversed sequentially from the canopy to the root and the associate list of each box is determined by searching the associate list from their parent's (Algorithm 4.3), which would have previously generated from the preceded sweep. In addition, the list 2 partition is also resolved at this stage. The second step involves sweeping the associate list of each box and determining their spatial position. In general, an iterative approach has to be applied (see Algorithm 4.4 for detail). At the end of this step, all of the List partitions would have resolved and the next stage in the FMM will commence.

4.2.1 The Morton index and the Z-order curve

The original sorting algorithm, as outlined in Algorithm 4.2, was inefficient and is difficult to parallelize on a *shared-memory* device due to the fact that the particle data

Algorithm 4.2: Tree-Construction

```
input : reference of the tree object (tree), a set of particle data (pData) and
        the division parameter (s)
output: reference to the modified tree object (tree)
// check if the current number is too small to initialize the construction
if the number of source particles of the initial domain is less than s then
    return
if tree is not initialized then
    tree.initialize(pData)
// perform the initial division, recall that the index representing the
   genesis domain is denoted by zero
B = tree.getBoxReferenceFromIndex(0)
list = tree.getDivisionList()
while list is non-empty do
    // copy the content of list to a temporary variable tmplist and reset the
       list
    tree.copyListTo(tmplist)
    tree.resetList()
    for each box B in tmplist do
        // applies Algorithm 4.1 sequentially to each box in the list
        Divide-Box(tree, B.getIndex(), s)
    // The output is a newly created list which contains the references of the
       non-empty boxes to be checked at the next iteration
    list = tree.getDivisionList()
```

have to be sequentially checked and compared to the boxes' locations. To overcome this difficulty and avoid serializing the checks, we adopted a more efficient way of spatially sorting the particles based on their coordinate values. To begin our presentation, one has to understand the concept of *space-filling curves*, which are commonly encountered in the field of computer science.

Essentially, the space-filling curve is a way to transform multi-dimensional data to one-dimension while preserving their local geometric properties (see Orenstein and Merrett (1984)). To achieve this, the points in a d-dimensional space is presented as a vector consisting of positive integers. The curve thus corresponds to the mapping

$$R : \mathbb{N}_{+}^{d} \to \mathbb{N}. \tag{4.2.5}$$

Depending on the map R, different types of space-filling curves can be obtained. Quite commonly, the *z-order* curve also known as the *Morton* curve is used for its consistency with the indexing convention adopted in Section 4.2 as a way to identify the boxes in the FMM tree. As previously mentioned, each component of a d-dimensional vector is represented as a positive integer, and thus one can determine its binary representation. The *bits* of the components are interleaved to produce a new positive integer of certain length (usually a 64-bit integer). The resultant integer is known as the z-value of the operation (hence the name). So for example, the

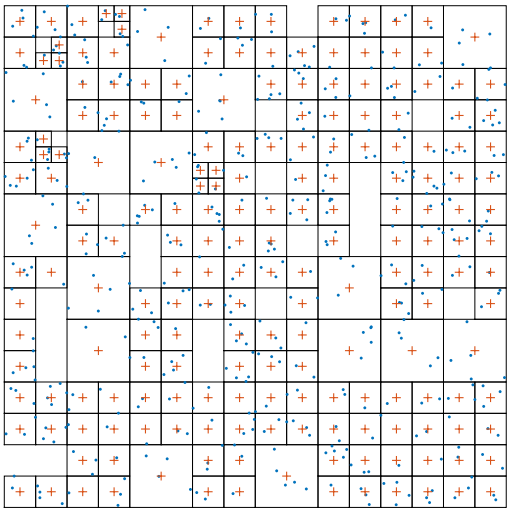


FIGURE 4.5: An example of the `TreeConstruction` algorithm applied to a uniform set of source particles in the unit square. The source points are represented by the blue dot whilst the box centers are indicated by the red cross. Moreover, the empty childless boxes are not shown.

z-value of two 8-bit unsigned integers 55 and 62 can be obtained as:

$$55 = 00110111_2 \tag{4.2.6a}$$

$$62 = 00111110_2 \tag{4.2.6b}$$

$$z = 0000111101111110_2 \tag{4.2.6c}$$

Here the subscript 2 indicates that the expression is in binary form. The result is a 16-bit integer equal to 3966 in decimal. So how could one apply this technique to the physical coordinates since they are most likely stored as floating-point number instead? A straight forward extension is to interleave the bits of the fractional part of the floating-point numbers (assumed non-negative), which is equivalent to modifying the map as in Eq.(4.2.5) to the following

$$R' : \mathbb{F}^d \to \mathbb{F}, \tag{4.2.7}$$

where $\mathbb{F}$ is the open unit interval. Indeed, this approach has been used by various authors in the literature (see Gumerov et al. (2003) for more detail) and is also adopted here. To see this, consider $x_1 = 0.25$ and $x_2 = 0.33$, the z-value as the result

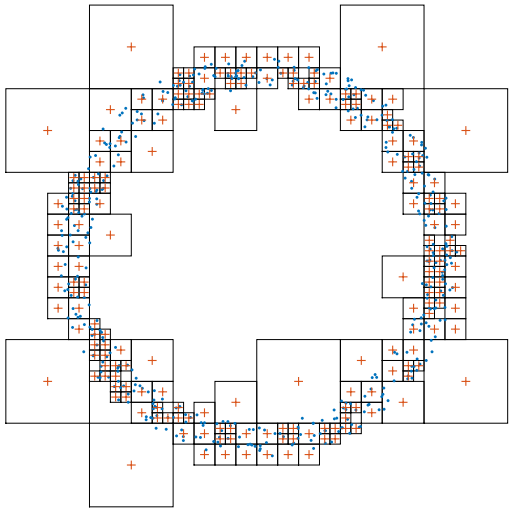


FIGURE 4.6: An example showing the domain division for an annular particle distribution. Again, the childless empty boxes are not shown here.

of the map R' is given as follows

$$0.25 = 0.01000000\ldots_2 \tag{4.2.8a}$$
$$0.33 = 0.01010100\ldots_2 \tag{4.2.8b}$$
$$z = 0.0011000100010000\ldots_2\,, \tag{4.2.8c}$$

which in decimal, the value of $R'(0.25, 0.33)$ is 0.191650390625. In terms of the FMM application, one can now demonstrate how the bit-interleaving technique can help us to spatially sort the particle in an numerically efficient and parallelizable way.

Consider a 2D source particle with coordinates given by $\vec{x} = (x_1, x_2)$, first one computes the z-value of the map R', i.e.

$$z = R'(x_1, x_2)\,, \tag{4.2.9}$$

and for a given refinement level k we label and attach all boxes at this refinement level with a unique number from the set $\{0, 1, 2, \ldots, 4^k - 1\}$. This number will be known as the *Morton index* of the box. Next, we compute the product between the z-value with the integer 4^k and we take only the integer part of the product. The result is an integer $I_{\text{morton}} \in \{0, 1, 2, \ldots, 4^k - 1\}$, which corresponds to the Morton index of the box that contains the source particle. i.e.

$$I_{\text{morton}} = \left\lfloor z \times 4^k \right\rfloor \tag{4.2.10}$$

Algorithm 4.3: Associate Partition

```
input  : a reference to the tree object (tree)
output: a modified tree object (tree)
// Initialize the search by appending the genesis box to its own associate
   list
B = tree.getBoxReferenceFromIndex(0)
B.appendAssociate(B)
for refinement level k from 1 to n do
    for each non-empty box B at refinement level k do
        // get the associate list from B's parent and denote it to the variable
           plist
        plist = B.getParentReference().getAssociateList()
        if plist is empty then
            continue
        for each box C in plist do
            if C is empty then
                continue
            if C is next to B and is childless then
                B.appendAssociate(C)
            else if C is a parent then
                // check the children boxes of C
                for each non-empty children box D of C do
                    if D is next to B then
                        B.appendAssociate(D)
                    else
                        B.appendToList2(D)
```

Additionally, one nice property of the map R' is that the centre position of the box with Morton index I_{morton} can be conveniently obtained by applying the de-interleaving operation to the index. Formally, the *de-interleaving* operation is expressed as a map Q such that

$$Q : \mathbb{N}_+ \to \mathbb{N}_+^d. \tag{4.2.11}$$

Let I_{morton} be represented in binary form (assume a 64-bit unsigned integer)

$$I_{\text{morton}} = [a_0 a_1 a_2 \dots a_n]_2 \tag{4.2.12}$$

where each coefficient a_i is either 0 or 1 and n is the significant bit position (in other words, the largest value n between 0 and 63 at which a_n is non-zero). One notes that the square bracket in Eq.(4.2.12) is defined to be the binary expansion having coefficients from left to right, i.e.

$$[a_0 a_1 a_2 \dots a_n]_2 = a_0 + a_1 2 + a_2 2^2 + \dots + a_n 2^n. \tag{4.2.13}$$

The result of Q applied to I_{morton} is a vector $\vec{n} = (n_1, n_2)$ whose elements are computed as follows:

$$n_1 = [a_1 a_3 \dots a_{2p+1}]_2, \quad n_2 = [a_0 a_2 \dots a_{2p}]_2, \tag{4.2.14}$$

Algorithm 4.4: List-Partition

```
input  : a reference to the tree object (tree)
output: a modified tree object (tree)
for each non-empty box B in tree do
    if B is a parent then
        return
    // copy the content of B's associate to the variable tmplist1 and reset
       tmplist2
    tmplist1 = B.getAssociateList()
    tmplist2.reset()
    done = false
    while done is false do
        for each box C in tmplist1 do
            if C is a parent then
                for each non-empty children D of C do
                    if D is next to B then
                        tmplist2.append(D)
                    else
                        B.appendToList3(D)
            else
                if C is next to B then
                    B.appendToList1(C)
                else
                    B.appendToList4(C)
        if tmplist2 is empty then
            done = true
        else
            tmplist1 = tmplist2
// apply a simple correction for parent boxes to avoid branching
for each non-empty parent box B in tree do
    for each non-empty associate box C of B do
        if C is childless and C is not next to B then
            B.appendToList4(C)
```

where $p = \lfloor n/2 \rfloor$(note that n_1 and n_2 are in general 64-bit integers and so all of the bits in position greater than $2p+1$ and $2p$ are suitably taken to be zero, respectively). The centre of the box, say $\vec{x}_c$, with the Morton index I_{morton} is thus given by

$$\vec{x}_c = 2^{-k}\,(\vec{n} + 0.5)\,. \tag{4.2.15}$$

The distribution of the Morton indices $\{0, 1, 2, \ldots, 4^k - 1\}$ produces a distinct access pattern. In fact, the curve defined by the box centres can be shown to be space-filling, which means as $k \to \infty$, the curve will visit every point in the square domain. Indeed, Figure (4.7) shows the z-order curve at the end of the 2^{nd}, 4^{th} and the 6^{th} iterations. Previously, one recalls that the box in the tree is labelled with the multi-dimensional index j, where $j = [0, i_1, i_2, \ldots i_n]$ at refinement level n. Using the

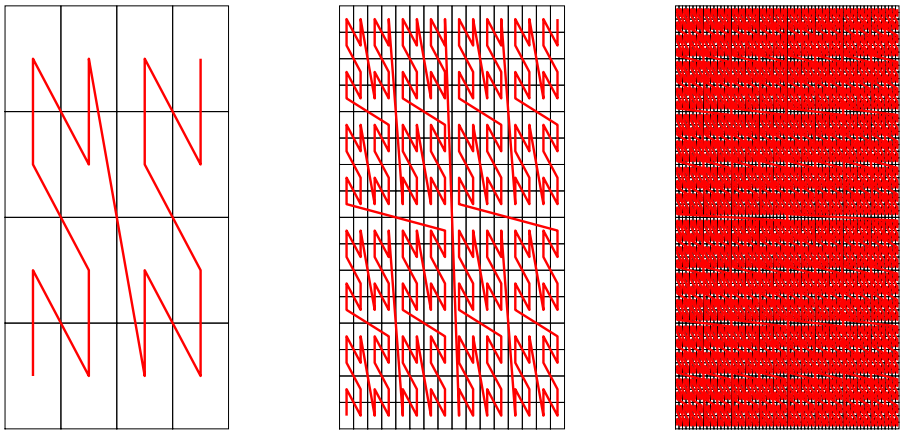


FIGURE 4.7: The z-order curve at the 2^{nd}, 4^{th} and 6^{th} iteration. This access pattern is completely determined by the interleaving operator R'.

Morton indexing, one can convert the multi-dimensional index j to its equivalent Morton index via the following transformation:

$$I_{\text{morton}} = [i_n i_{n-1} \dots i_1]_4 \tag{4.2.16}$$

where we make use of the notation in Eq.(4.2.13) to mean the number expansion with respect to the base 4 (also, there is a straightforward generalization to d-dimensional space, namely Eq.(4.2.16) may be replaced by $I_{\text{morton}} = [i_n i_{n-1} \dots I_1]_{2^d}$). One important feature of the multi-dimensional index is its ability to recover the parental index, which is simply obtained by truncating the last digit i_n. So for instance, by defining the *parent* function, we have

$$parent\,(j) = [0, i_1, i_2, \dots, i_{n-1}]. \tag{4.2.17}$$

Thus, one can further derive the Morton index of the parent index by applying Eq.(4.2.16) to its multi-dimensional index. It is desirable to derive a similar function that operates on the Morton indices instead. However, by converting to the Morton convention the refinement information is lost, so in practice, for any operation involving the boxes expressed in the Morton convention, one has to maintain an additional array which serves to keep track of the refinement level from which the boxes are derived. The *parent* function now takes the explicit form:

$$parent\,(I_{\text{morton}}) = \frac{I_{\text{morton}} - \text{mod}\,(I_{\text{morton}}, 4)}{4}. \tag{4.2.18}$$

where mod is the modulo function. Although much of the discussion have been centered on the 2D case where $d = 2$, there is a straightforward extension to 3D and therefore we will not repeat the arguments here.

Having introduced the mathematical machinery, the `TreeConstruction` algorithm (i.e.Algorithm 4.2) has to be modified to reflect the changes needed to run on a shared-memory device. Contrary to Algorithm 4.2, the algorithm we are about to describe is slightly more complicated. In general, a key characteristic of the original

`TreeConstruction` algorithm is that the leaf nodes are created progressively from the coarser levels, this can be seen as a *forward* traversal, since the sorting at finer levels is done from a set of inherit particle data belonging to the previously sorted set. The modified construction does not proceed in this forward manner. Instead, one initializes the construction by prescribing the maximum refinement level. Subsequently, the routine proceeds from the root and merges boxes belonging to the same family if the combined sum of the particles is less than the division parameter s. Precisely, it is this characteristic that we now term the routine as the *reverse* traversal approach. To this ends, several *book-keeping* arrays are now introduced, which serve to facilitate the communication between particle data and that of the leafs of the tree. These are as follows:

1. `permutationArray`. This array stores the particle indices resulted from a sorting operation based on the particles' Morton index.
2. `boxPointerArray`. This array stores the Morton index and the refinement number of the childless box in which the sorted particle resides.
3. `particleBinArray`. This array stores the number of source particles in each leaf of the tree.
4. `particlePointerArray`. This array stores the smallest sorted particle index from a set of particles that have the same Morton index and refinement number.

One can infer the lengths of those arrays. For example, the `permutationArray` and `boxPointerArray` both have a length equal to the number of particles in the computation domain, whereas `particlePointerArray` and `particleBinArray` must be maintained at a length equal to Eq.(4.2.4) to account for all particle distributions. The pseudo-code for the modified tree-construction is given by Algorithm 4.6.

Algorithm 4.5: Morton Sort

input : a particle data array (pData), maximum refinement level (n), boxPointerArray, permutationArray and particleBinArray
output: modified boxPointerArray, permutationArray and particleBinArray

```
counter = 0;
for each source particle p in pData do
    // obtain the Morton index based on p's position coordinates.  This step
       can be parallelized
    lmorton = ComputeMortonIndex(p.getCoordinates());
    boxPointerArray.idx(counter) = lmorton;
    boxPointerArray.lvl(counter) = n;
    counter ← counter + 1;
    // note here that the indexing method of particleBinArray requires two input
       arguments
    particleBinArray.idx(lmorton, n) ← particleBinArray.idx(lmorton, n) + 1;

// Sort sorts the first input argument in ascending order and returns the
   sorted array and the permutation array
Sort(boxPointerArray, permutationArray);
```

Qualitatively, the particle data are first passed to the function `MortonSort` (Algorithm 4.5), during which the Morton index of each particle is computed using

Algorithm 4.6: Modified Tree Construction

```
input : a reference to the tree object (tree), the maximum refinement level
        (n), the division parameter (s) and the particle data array (pData)
output: particleBinArray, permutationArray, boxPointerArray and
        particlePointerArray
// applies the MortonSort algorithm to spatially sort the particle array based
   on the Morton index
MortonSort(pData, n, boxPointerArray, permutationArray, particleBinArray)

// np is the effective max refinement number
np = n
// apply the merging operation
for each refinement level k from n to 1 do
    allChildlessFlag = true
    for each box b at refinement level k do
        lmorton = b.getMortonIndex()
        if particleBinArray.idx(lmorton, k) is 0 then
            b.setEmpty()
            continue
        if particleBinArray.idx(lmorton, k) is less than s then
            b.setNonEmpty()
            b.setChildless()
            b.setChildrenBoxesEmpty()
        else
            allChildlessFlag = false
            b.setParent()
        Jmorton = Parent(lmorton)
        particleBinArray.idx(Jmorton, k- 1)← particleBinArray.idx(Jmorton, k-
          1) + particleBinArray.idx(lmorton, k)
    if allChildlessFlag is true then
        np = k
for each i from 0 to boxPointerArray.length() - 1 do
    boxPointerArray.lvl(i) = np

// particlePointerArray contains the smallest sorted particle index from a set of
   neighboring particles with the same Morton index and refinement number.
for each refinement level k from np to 1 do
    for each i from 0 to boxPointerArray.length() - 1 do
        lmorton = boxPointerArray.idx(i)
        L = boxPointerArray.lvl(i)
        if tree.getBoxReferenceFromIndex(lmorton, L) is an empty box then
            lmorton ← Parent(lmorton)
            L ← L- 1
            boxPointerArray.idx(i) ← lmorton
            boxPointerArray.lvl(i) ← L
        particlePointerArray.idx(lmorton, k) ← min(
          particlePointerArray.idx(lmorton, k), i)
```

Eq.(4.2.10) and the particles are sorted in ascending order based on their computed Morton index. The output of this step is the modified `boxPointerArray`, `particleBinArray` and `permutationArray`. What follows is the merging operation whereby the *empty* and the *parental properties* are updated for each box in the tree. This step occurs from the root and is progressed successively to coarser levels. A secondary objective of this step is to identify the effective refinement number `np`, which will differ from the initial prescribed refinement number if a fewer refinement levels were able to resolve the particle distribution. The final stage of the algorithm is creating the pointer to the sorted particle data, which consists of finding and storing the smallest sorted particle index for a given group of neighbouring particles with the same Morton index and refinement number.

Using a combination of the book-keeping arrays, access to the particle data is characterized by a series of memory mappings. For instance, given a box identifier j (either a multi-dimensional index or the Morton index and the refinement number tuple) and one wishes to fetch all of the particle data contained within the box, the code passes j to the `particleBinArray` to obtain the number of source particles that reside in that box. The first sorted particle index is subsequently returned by the `particlePointerArray`, from which all of the sorted particle indices belonging to that box are deduced. The sorted particle indices are then passed to the permutation array to obtain their actual indices. This process is illustrated in Figure (4.8). Similarly, if one wishes to determine the set of neighbouring particles for a given sorted particle index, say j, then the following procedure can be applied: First, j is fed to the `boxPointerArray`, which returns the Morton index and the refinement number of the childless box that contains j. Then one repeats the preceded operation with this as the box identifier to obtain all of the neighbouring points.

It is worth noting that for a highly parallelizable code, it is important to have an independent access structure. This means threads can fetch information independent of other threads. The code structure that we just introduced fulfils precisely this role. However, one should note that there are several disadvantages to this approach. First, since the traversal occurs from the root, the adaptive control is no longer possible (meaning that the refinement level cannot proceed beyond the prescribed refinement number), which may produce a notable side effect that some boxes at the finest level violate the division criterion. A second disadvantage is that in order to establish the communication between particle data and the tree, the code maintains several large arrays. For machines with low memory capacity, this may seen problematic. Furthermore, the final access pattern, which occurred when fetching particle data, is considered a random access; the caller function has to make several fetch requests to the main memory to service the threads. Depending on the hardware, this operation can induce a substantial latency as accessing the main memory is typically the most expensive access in the GPU's memory hierarchy.

4.3 Two-dimensional FMM

In this section, the mathematical formulation of the 2D FMM implementation is presented. We follow the method introduced by Carrier et al. (1988). The main objective of the 2D-FMM is to compute the induced velocity due to a large collection of vortex elements. In the far-field limit, the induced velocity is approximated by point vortices for which the induction field behaves like $1/r$, where r is the radius between a field point and the element's position. Under such a simplification, the induced velocity can be represented using complex formulation. Let Φ define the complex

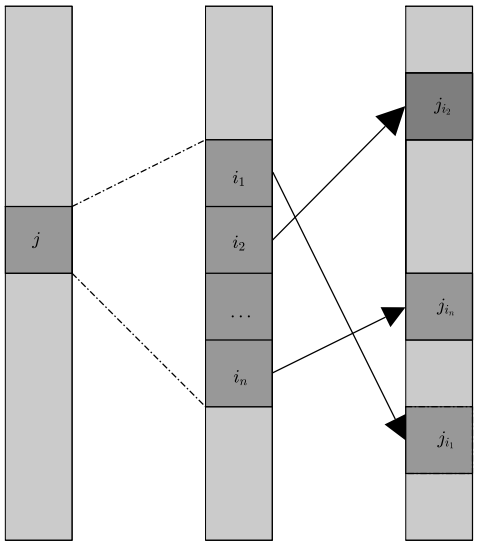


FIGURE 4.8: From left to right, each schematic rectangle represents the memory address of the `particlePointerArray`, `permutationArray` and the particle data, respectively. Access to particle data is characterized by a series of memory mappings. However, the access pattern in the last phase is considered a random access, which may incur some speed penalties.

potential in the complex plane $\mathbb{C}$. In formulae, Φ is computed as

$$\Phi = \psi + i\phi, \tag{4.3.1}$$

where ψ is the stream-function, ϕ is the velocity potential that gives rise to the induced velocity and i is the imaginary unit. Due to the analyticity of the potential field, it is expected that there exists a close form expression for Φ which depends only on the complex variable z, defined as $z = x + iy$, where (x, y) defines the usual Cartesian coordinate in the problem. Thus, one should expect that the velocity potential is linked to the stream-function by means of the Cauchy-Riemann equation, i.e.

$$\frac{\partial \phi}{\partial x} = -\frac{\partial \psi}{\partial y}, \tag{4.3.2a}$$

$$\frac{\partial \phi}{\partial y} = \frac{\partial \psi}{\partial x}. \tag{4.3.2b}$$

Indeed, Eq.(4.3.2) implies that both the stream-function and the velocity potential satisfy the Laplace's equation (i.e. Eq.(3.2.5)). For a point vortex, it is easy to see that the solution to the Laplace's equation is given by:

$$\phi = \frac{\Gamma}{2\pi} \tan^{-1}\left(\frac{y}{x}\right). \tag{4.3.3}$$

where Γ is the circulation of the point vortex. By the Cauchy-Riemann's condition, the stream-function is derived as:

$$\psi = -\frac{\Gamma}{2\pi}\log\left(\sqrt{x^2+y^2}\right). \tag{4.3.4}$$

Thus the complex potential Φ may be expressed in terms of the complex variable z as

$$\Phi(z) = -\frac{\Gamma}{2\pi}\log z + C. \tag{4.3.5}$$

where C is the constant of the integration. Since the constant term in Eq.(4.3.5) does not contribute to the induced velocity, it is preferable to neglect it in the calculation. For a collection of discreet vortices, parametrized by the tuple $(\alpha_j, \vec{x}_j)$ for $j = 0, 1, \ldots, N-1$, the total complex potential is evaluated as the sum of all of the individual vortex elements, i.e.

$$\Phi(z) = -\sum_{j=0}^{N-1}\frac{\alpha_j}{2\pi}\log(z - z_j). \tag{4.3.6}$$

where z_j is the complex position formed from the position vector $\vec{x}_j$. Since we require the velocity, it is convenient to work with the derivative of Φ, which is related to the complex velocity $\varpi := v + iu$ as:

$$\varpi(z) = \frac{d\Phi}{dz}. \tag{4.3.7}$$

Moreover, by differentiating the individual terms in Eq.(4.3.6), ϖ may also be expressed as

$$\varpi(z) = -\sum_{j=0}^{N-1}\frac{\alpha_j}{2\pi}\frac{1}{z - z_j}. \tag{4.3.8}$$

The form of the complex kernel in Eq.(4.3.8) suggests that the kernel can be written as a power series in terms of the variable z or z^{-1}. Indeed, the basic premise of the FMM is to be able to apply the translation and conversion operators to the multipole expansion to account successively the hierarchical far-field influence. The following sections describe the mathematical machinery to accomplish such a goal.

4.3.1 Multipole expansion of the complex velocity

We assume the source particles $((\alpha_j, z_j), j = 0, 1, \ldots, N-1)$ are distributed inside a closed circular region with radius r and centred at the origin. We denote this region by $B(r, 0) \subset \mathbb{C}$. Let $z \in \mathbb{C}$ denote a field point. Then for $z \notin B(r, 0)$, the following holds:

$$\left|\frac{z_j}{z}\right| < 1. \tag{4.3.9}$$

From Eq.(4.3.8), we have

$$\varpi(z) = -\sum_{j=0}^{N-1}\frac{\alpha_j}{2\pi}\frac{1}{z(1 - z_j/z)} = -\sum_{j=0}^{N-1}\frac{\alpha_j}{2\pi z}\left(1 - \frac{z_j}{z}\right)^{-1}. \tag{4.3.10}$$

Noting Eq.(4.3.9), one can show that the Taylor expansion in Eq.(4.3.10) converges and is given by

$$\left(1-\frac{z_j}{z}\right)^{-1}=\sum_{k=0}^{\infty}\left(\frac{z_j}{z}\right)^k. \tag{4.3.11}$$

By substituting Eq.(4.3.11) to Eq.(4.3.10) and interchanging the summations, one arrives at the multipole expansion for the complex velocity ϖ, which is everywhere convergent in the complement of $B(r,0)$, i.e.

$$\varpi(z)=\sum_{k=0}^{\infty}a_k z^{-k-1}, \tag{4.3.12}$$

where

$$a_k=-\sum_{j=0}^{N-1}\frac{\alpha_j z_j^k}{2\pi}, \tag{4.3.13}$$

which will be known as the multipole coefficient, and the set of multipole coefficients will hereafter denote by $[a_k]$.

4.3.2 Translation operator of the 2D multipole expansion

We formalize this process as a map T_w such that

$$T_w:\mathbb{C}^P\to\mathbb{C}^P, \tag{4.3.14}$$

which principally transforms a set of multipole coefficients in the open ball $B(r_0,w_0)$ to the set of multipole coefficients in the ball $B(r_1,w_1)$ via the complex translation vector w, where

$$B(r_0,w_0)\subseteq B(r_1,w_1). \tag{4.3.15}$$

Here, w_0 and w_1 denote the ball centres and $r_1\geq r_0$. Without loss of generality, one may suppose that $w_1=0$ (this means the origin of the coordinate system coincides with that of the centre of $B(r_1,w_1)$) so for $z\notin B(r_1,w_1)$, Eq.(4.3.12) becomes

$$\varpi|_{B(r_0,w_0)}=\sum_{k=0}^{\infty}a_k(z-w_0)^{-k-1}. \tag{4.3.16}$$

For this z, it is certainly true that

$$\left|\frac{w_0}{z}\right|<1 \tag{4.3.17}$$

therefore, one may expand the inner term in Eq.(4.3.16) as a power series of the form

$$(z-w_0)^{-k-1}=z^{-k-1}(1-w_0/z)^{-k-1}=z^{-k-1}\sum_{s=0}^{\infty}\binom{k+s}{s}\left(\frac{w_0}{z}\right)^s. \tag{4.3.18}$$

In the above, we have used the *Binomial* notation, i.e.

$$\binom{n}{k}:=\frac{n!}{k!\,(n-k)!}. \tag{4.3.19}$$

Substituting Eq.(4.3.18) to Eq.(4.3.16), one obtains the following:

$$\varpi(z)=\sum_{k=0}^{\infty}a_k z^{-k-1}\sum_{s=0}^{\infty}\binom{k+s}{s}w_0^s z^{-s}=\sum_{k=0}^{\infty}\sum_{s=0}^{\infty}a_k\binom{k+s}{s}w_0^s z^{-k-s-1}. \tag{4.3.20}$$

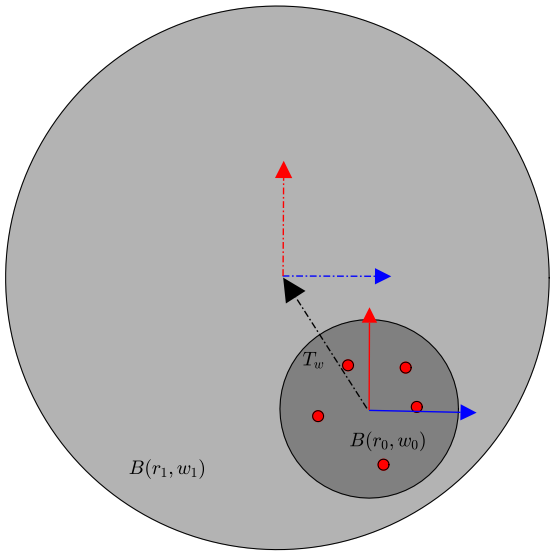


FIGURE 4.9: Schematic diagram showing that the effect of the map T_w is to translate the multipole coefficients in $B(r_0, w_0)$ to the ball $B(r_1, w_1)$. So for any field point z in the complement of $B(r_1, w_1)$, Eq.(4.3.12) is valid.

The double summation in Eq.(4.3.20) can be simplified into the following form:

$$\sum_{k=0}^{\infty}\sum_{s=0}^{\infty} a_k \binom{k+s}{s} w_0^s z^{-k-s-1} = \sum_{q=0}^{\infty} z^{-q-1} \sum_{k=0}^{q} a_k \binom{q}{q-k} w_0^{q-k}. \tag{4.3.21}$$

It follows that the multipole expansion in $B(r_1, w_1)$ is related to $B(r_0, w_0)$ via

$$\mathcal{O}|_{B(r_1,w_1)} = \sum_{k=0}^{\infty} b_k z^{-k-1}, \tag{4.3.22}$$

where

$$b_k := \sum_{s=0}^{k} a_s \binom{k}{k-s} w_0^{k-s}. \tag{4.3.23}$$

Equivalently, the translation operator T_w is explicitly constructed as

$$T_{w_0}([a_k]) = [b_k]. \tag{4.3.24}$$

In cases that $w_1 \neq 0$, then one simply applies the transformation:

$$w_0 \leftarrow w_0 - w_1, \tag{4.3.25}$$

as the effective translation vector.

4.3.3 2D local expansion and the conversion operator

Consider now that the source particles are distributed in a close ball $B(R, w)$ with $w \neq 0$. Let z be a field point close to the origin (see Figure (4.10) for the schematic

description). Assume that the following inequality is satisfied by z:

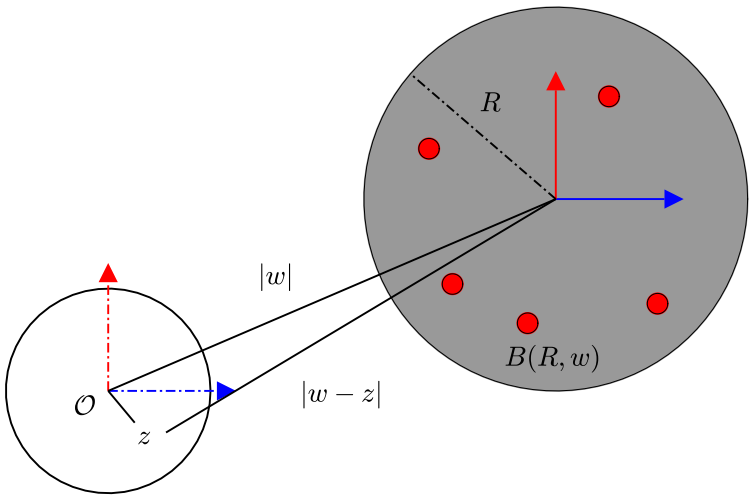


FIGURE 4.10: Schematic diagram illustrating the distribution of the source particles which are constrained in the ball $B(R, w)$ with $w \neq 0$. The field point z has to satisfy the inequality $|w - z| > R$ in order for the local expansion to be convergent.

$$|w - z| > R. \tag{4.3.26}$$

From Eq.(4.3.8) and together with the fact that $|z_j - w| < R$, then it is true that:

$$\left|\frac{z}{z_j}\right| < 1, \tag{4.3.27}$$

therefore a local expansion can be derived, i.e.

$$\varpi(z) = \sum_{k=0}^{\infty} z^k \sum_{j=0}^{N-1} \frac{\alpha_j}{2\pi} z_j^{-k-1} \tag{4.3.28}$$

with c_k defines the local coefficient of the expansion, i.e.

$$c_k := \sum_{j=0}^{N-1} \frac{\alpha_j}{2\pi} z_j^{-k-1}. \tag{4.3.29}$$

From a computational perspective, it is undesirable to compute the local expansion from first principle (i.e. Eq.(4.3.28)). It is much faster to transform the existing multipole expansion to a local expansion. To accomplish such a goal, again, we introduce this process formally by means of a map Q_w such that

$$Q_w : \mathbb{C}^P \to \mathbb{C}^{P+1}. \tag{4.3.30}$$

First we note that the multipole expansion of the ball $B(R,w)$ is expressed by a change of origin, i.e.

$$\varpi|_{B(R,w)} = \sum_{k=0}^{\infty} a_k (z-w)^{-k-1}. \tag{4.3.31}$$

Noting the inequality in Eq.(4.3.26), it is possible to show that the following inequality holds:

$$\left|\frac{z}{w}\right| < 1. \tag{4.3.32}$$

Together with the general result

$$(1-x)^{-n} = \sum_{s=0}^{\infty} \binom{n+s-1}{s} x^s, \qquad n \geq 1, \tag{4.3.33}$$

one can Tayler expand the summand in Eq.(4.3.31) in a similar way, i.e.

$$(z-w)^{-k-1} = (-w)^{-k-1}\left(1-\frac{z}{w}\right)^{-k-1} = (-w)^{-k-1}\sum_{s=0}^{\infty}\binom{k+s}{s}\left(\frac{z}{w}\right)^s \tag{4.3.34}$$

Upon substituting Eq.(4.3.34) to Eq.(4.3.31), we have

$$\begin{aligned}\varpi|_{B(R,w)} &= \sum_{k=0}^{\infty}\sum_{s=0}^{\infty} a_k (-w)^{-k-1}\binom{k+s}{s}\left(\frac{z}{w}\right)^s \\ &= \sum_{s=0}^{\infty} z^s \left(\sum_{k=0}^{\infty} a_k \binom{k+s}{s}(-1)^{-k-1} w^{-k-s-1}\right)\end{aligned} \tag{4.3.35}$$

By defining the set of transformed coefficients c_k as follows:

$$c_k := \sum_{s=0}^{\infty} a_s \binom{k+s}{k}(-1)^{-s-1} w^{-k-s-1}, \tag{4.3.36}$$

the explicit form of the conversion operator Q_w can be written as

$$Q_w([a_k]) = [c_k]. \tag{4.3.37}$$

If a change of origin is required, say W, then one applies the transformation to w

$$w \leftarrow w - W. \tag{4.3.38}$$

4.3.4 Translation operator of the 2D local expansion

The translation operator operating on the local coefficients is defined by the map V_w. Suppose that the local expansion of the open ball $B(r_1,w)$ is given by

$$\varpi|_{B(r_1,w)} = \sum_{k=0}^{\infty} c_k (z-w)^k. \tag{4.3.39}$$

Let $B(r_2,0)$ denote the open ball with radius r_2 centred at the origin and is such that

$$B(r_2,0) \subseteq B(r_1,w). \tag{4.3.40}$$

The local expansion in $B(r_2, 0)$ is obtained by expanding $(z-k)^k$ as a Binomial expansion and collecting like terms, i.e.

$$w|_{B(r_2,0)} = \sum_{k=0}^{\infty} z^k \left(\sum_{s=k}^{\infty} c_s \binom{s}{s-k} (-w)^{s-k} \right). \tag{4.3.41}$$

The effect of the map V_w is to transform the set of local coefficients $[c_k]$ to $[d_k]$

$$V_w([c_k]) = [d_k], \tag{4.3.42}$$

where

$$d_k := \sum_{s=k}^{\infty} c_s \binom{s}{s-k} (-w)^{s-k}. \tag{4.3.43}$$

4.4 Three-dimensional FMM

Having introduced the mathematical framework in Section 4.3, it is natural that one should extend the framework to the three-dimensional case. However, naive attempt to generalize this to 3D will meet with considerable difficulty. In light of Section 3.1, the vortex elements in 3D are now characterized by a vector-valued circulation. Moreover, in the limit as the core parameter ϵ varnishes, the mollified Biot Savart with the Gaussian smoothing function can be shown to approach the asymptotic expression:

$$\lim_{\epsilon \to 0} \vec{u}_\epsilon = -\sum_{j=0}^{N-1} \nabla G(\vec{x} - \vec{x}_j) \times \vec{\alpha}_j, \tag{4.4.1}$$

where $G(\vec{x})$ is the singular Green's function (see Section 3.1.1 for detail). Noting that $\vec{\alpha}_j$ is spatially independent, the far-field limit of the mollified Biot Savart induction can be alternatively expressed as:

$$\vec{u}_{\text{farfield}} = \frac{1}{4\pi} \nabla \times \left(\sum_{j=0}^{N-1} \frac{\vec{\alpha}_j}{\|\vec{x} - \vec{x}_j\|} \right) \tag{4.4.2}$$

Denote the *vector* potential $\vec{\phi}$ as

$$\vec{\phi} := \sum_{j=0}^{N-1} \frac{\vec{\alpha}_j}{\|\vec{x} - \vec{x}_j\|}, \tag{4.4.3}$$

it is not difficult to observe that the 3D FMM would require three multipole expansions along each basis direction. For this reason, the computational cost of the 3D FMM is significantly higher than its 2D counterpart. Nonetheless, we have developed a highly efficient 3D FMM engine that improves upon the current FMM codes in the literature by means of deriving an exact expression for the local strain field. The purpose of this section is to review and present the 3D FMM mathematical framework. Although the underlying mathematics detract quite significantly from that of the 2D, the underlying code, however, follows exactly the same structure. The only aspects that differ quite markedly are the translation and conversion operators (i.e. T_w, Q_w and V_w), which are a lot more complicated to describe. The subsequent presentation is a variation of the idea introduced by L. Greengard and Rokhlin (1997).

4.4.1 Complex spherical harmonic basis (CSHB)

The first step towards obtaining the FMM is to introduce a set of complete basis functions $\{\gamma_k, k = 0, \pm 1, \pm 2, \dots\}$ under which the multipole and local series have the expected convergence behaviour in this function space. In 2D, this is given by the power functions, i.e. $\{\dots, z^{-2}, z^{-1}, 1, z, z^2, \dots\}$. In 3D, one popular choice is the complete set of complex spherical harmonic functions, which we will denote by Y_n^m. The subscript $n \in \mathbb{N}_+$ is called the *degree* and the superscript $m \in \mathbb{N}$ is known as the *order*. Furthermore, it is assumed that

$$|m| \leq n. \tag{4.4.4}$$

To derive the form of Y_n^m, we note that each component of the vector potential $\vec{\phi}$ satisfies the Laplace's equation with respect to the variable $\vec{x}$. Recall that the Cartesian Laplacian is the operator $\mathcal{L}$ such that

$$\mathcal{L} = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}. \tag{4.4.5}$$

In the spherical coordinate system (r, θ, ϕ), the Cartesian coordinates (x, y, z) transform as

$$x = r \sin\theta \cos\phi, \quad y = r \sin\theta \sin\phi, \quad z = r \cos\theta. \tag{4.4.6}$$

Consequently, Eq.(4.4.5) becomes:

$$\mathcal{L} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial}{\partial r}\right) + \frac{1}{r^2 \sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta \frac{\partial}{\partial \theta}\right) + \frac{1}{r^2 \sin^2\theta}\frac{\partial^2}{\partial \phi^2}. \tag{4.4.7}$$

Let $f = f(r, \theta, \phi)$ be a solution to the spherical Laplace's equation, i.e.

$$\mathcal{L} f = 0. \tag{4.4.8}$$

One seeks separable solution of the form

$$f(r, \theta, \phi) = R(r)\, Y(\theta, \phi). \tag{4.4.9}$$

Substituting Eq.(4.4.9) to Eq.(4.4.8) yields:

$$\frac{1}{R}\frac{\partial}{\partial r}\left(r^2 \frac{\partial R}{\partial r}\right) + \frac{1}{Y \sin\theta}\left(\frac{\partial}{\partial \theta}\left(\sin\theta \frac{\partial Y}{\partial \theta}\right) + \frac{1}{\sin\theta}\frac{\partial^2 Y}{\partial \phi^2}\right) = 0. \tag{4.4.10}$$

By virtue of the separable assumption, the only way that Eq.(4.4.10) can be satisfied is when each term is equal to a constant, say λ, i.e.

$$\frac{1}{R}\frac{\partial}{\partial r}\left(r^2 \frac{\partial R}{\partial r}\right) = \lambda, \tag{4.4.11a}$$

$$\frac{1}{Y \sin\theta}\left(\frac{\partial}{\partial \theta}\left(\sin\theta \frac{\partial Y}{\partial \theta}\right) + \frac{1}{\sin\theta}\frac{\partial^2 Y}{\partial \phi^2}\right) = -\lambda. \tag{4.4.11b}$$

Using the product rule, Eq.(4.4.11a) may be solved by a power solution of the form

$$R = r^\alpha. \tag{4.4.12}$$

Substituting the trial solution to Eq.(4.4.11a), it is readily seen that α satisfies quadratic equation

$$\alpha^2 + \alpha - \lambda = 0. \tag{4.4.13}$$

Denote the two roots of Eq.(4.4.13) as $\alpha_\pm$, the general form of R (assume the roots are distinct) is given by

$$R(r) = Ar^{\alpha_+} + Br^{\alpha_-}, \tag{4.4.14}$$

for some constant A and B. A separable solution for Eq.(4.4.11b) is sought. Suppose that $Y(\theta,\phi) = \Theta(\theta)\Phi(\phi)$, then through a similar argument, it can be shown that for some complex constant m, the following holds:

$$\frac{\partial^2\Phi}{\partial\phi^2} + m^2\Phi = 0, \tag{4.4.15a}$$

$$\sin\theta\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial\Theta}{\partial\theta}\right) + \left(\lambda\sin^2\theta - m^2\right)\Theta = 0, \tag{4.4.15b}$$

The general solution of Eq.(4.4.15a) is a linear combination of the complex exponential $e^{\pm im\phi}$. Additionally, by noting the periodicity of the function $\Phi(\phi)$ at $\phi = 0, 2\pi$, one can deduce that m must take integer values. Furthermore, by dividing $\sin\theta$ and defining the operator $\mathcal{L}_{\mathrm{SL}}$ such that

$$\mathcal{L}_{\mathrm{SL}} = \frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) - \frac{m^2}{\sin\theta}, \tag{4.4.16}$$

Eq.(4.4.15b) is recast as:

$$\mathcal{L}_{\mathrm{SL}}\Theta = -\lambda\sin(\theta)\Theta, \tag{4.4.17}$$

which constitutes a *Sturm-Liouville* problem whose solution consists of finding the eigenvalues λ that satisfy the self-adjoint condition at the boundary point $\theta = 0, \pi$. Further requiring that Θ to be regular at the boundary points, one can deduce that λ must have the form

$$\lambda = n(n+1), \tag{4.4.18}$$

for some positive integer n for which $n \geq |m|$ (see Appendix B for the derivation). Substituting Eq.(4.4.18) to Eq.(4.4.13) to obtain $\alpha_+ = n$ and $\alpha_- = -n-1$. Moreover, by a change of variable $x = \cos\theta$ and $\Theta(\theta) = F(\cos\theta)$, Eq.(4.4.15b) is recast to a more familiar form:

$$\frac{d}{dx}\left(\left(1-x^2\right)\frac{dF}{dx}\right) + \left(n(n+1) - \frac{m^2}{1-x^2}\right)F = 0. \tag{4.4.19}$$

Eq.(4.4.19) is known as the *Legendre* equation and admits a set of elementary solutions called the *associated Legendre functions*. Let us denote such a set by the notation P_n^m. Suppose d_n^m is a normalization factor, then we define the complex spherical harmonic basis function (CSHB) $Y_n^m(\theta,\phi)$ at degree n and order m as

$$Y_n^m(\theta,\phi) := d_n^m P_n^{|m|}(\cos\theta)\, e^{im\phi}. \tag{4.4.20}$$

By virtue of the Sturm-Liouville theory, the set of CSHB is complete on the unit sphere in the sense that any function defined on the unit sphere can be represented by a linear combination of the CSHB functions. Indeed, the general solution to the

spherical Laplace's equation can be expressed as:

$$f(r,\theta,\phi) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \left(\alpha_n^m r^n + \beta_n^m r^{-n-1} \right) Y_n^m(\theta,\phi) \tag{4.4.21}$$

We use the same normalization factor d_n^m as Cheng et al. (1999), in that

$$d_n^m = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}}. \tag{4.4.22}$$

4.4.2 Multipole expansion in the CSHB function space

An important property of the spherical harmonic functions is that they obey what is known as the *addition law*. Formally, let γ denote the angle between two points whose spherical coordinates are (ρ,α,β) and (r,θ,ϕ). More precisely, γ is the angle between the two lines that extend from the origin to where these two points are, then it is true that

$$P_n(\cos\gamma) = \sum_{m=-n}^{n} Y_n^{-m}(\alpha,\beta)\, Y_n^m(\theta,\phi), \tag{4.4.23}$$

where P_n is the *Legendre* polynomial of degree n, which may be defined via the generating function of the form

$$\frac{1}{\sqrt{1-2xt+t^2}} = \sum_{n=0}^{\infty} P_n(x)\, t^n. \tag{4.4.24}$$

One can relate the application of Eq.(4.4.24) to the kernel $1/\left\|\vec{x}-\vec{x}_j\right\|$ as follows:

$$\begin{aligned} \frac{1}{\left\|\vec{x}-\vec{x}_j\right\|} &= \frac{1}{\sqrt{\left\|\vec{x}-\vec{x}_j\right\|^2}} = \frac{1}{\sqrt{\|\vec{x}\|^2 - 2\vec{x}\cdot\vec{x}_j + \left\|\vec{x}_j\right\|^2}} \\ &= \frac{1}{r\sqrt{1-2\cos\gamma\,(r_j/r)+(r_j/r)^2}}, \end{aligned}$$

where $r := \|\vec{x}\|$ and $r_j := \left\|\vec{x}_j\right\|$. Moreover, by matching the terms in Eq.(4.4.24), it is readily seen that

$$\frac{1}{\left\|\vec{x}-\vec{x}_j\right\|} = \frac{1}{r}\sum_{n=0}^{\infty} P_n(\cos\gamma)\left(\frac{r_j}{r}\right)^n. \tag{4.4.25}$$

Let a set of source particles $\{(\vec{\alpha}_j,\vec{x}_j), j = 0,1,\ldots,N-1\}$ occupy the open ball $B(R,0) \subset \mathbb{R}^3$, then for any $\vec{x}$ in the complement of B with spherical coordinate (r,θ,ϕ), the vector potential as in Eq.(4.4.3) may be computed by

$$\begin{aligned} \vec{\phi}|_{B(R,0)} &= \sum_{j=0}^{N-1} \vec{\alpha}_j \sum_{n=0}^{\infty} P_n(\cos\gamma_j)\, r_j^n r^{-n-1} \\ &= \sum_{j=0}^{N-1} \vec{\alpha}_j \sum_{n=0}^{\infty} \sum_{m=-n}^{n} Y_n^{-m}(\alpha_j,\beta_j)\, Y_n^m(\theta,\phi)\, r_j^n r^{-n-1} \\ &= \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \vec{M}_n^m r^{-n-1} Y_n^m(\theta,\phi). \end{aligned} \tag{4.4.26}$$

where the vector-valued multipole coefficient $\vec{M}_n^m$ is given by

$$\vec{M}_n^m = \sum_{j=0}^{N-1} r_j^n \vec{\alpha}_j Y_n^{-m}\left(\alpha_j, \beta_j\right), \tag{4.4.27}$$

and $\left(r_j, \alpha_j, \beta_j\right)$ denotes the spherical coordinate of $\vec{x}_j$. Similarly, we denote the set of expansion coefficients with a square bracket, i.e. $[\vec{M}_n^m]$.

4.4.3 Translation operator of the 3D multipole expansion

In much the same way as in Section 4.3.2, the translation operator is a map $T_{\vec{y}}$ that transforms the set of vector-valued multipole coefficients $[\vec{M}_n^m]$ in the ball $B\left(R_0, \vec{x}_0\right)$ to the ball $B\left(R_1, \vec{0}\right)$, where

$$B\left(R_0, \vec{x}_0\right) \subseteq B\left(R_1, \vec{0}\right). \tag{4.4.28}$$

Here it must be the case that $R_1 \geq R_0$, which denote the ball radii. The map $T_{\vec{y}}$ is a linear operator that transforms as follows:

$$T_{\vec{x}_0}\left([\vec{M}_n^m]\right) = [\vec{N}_n^m]. \tag{4.4.29}$$

The result is a dense matrix-vector multiplication, whose elements of the product $[\vec{N}_n^m]$ are given by the formula

$$\vec{N}_n^m = \sum_{j=0}^{n} \sum_{k=-j}^{j} \frac{\vec{M}_{n-j}^{m-k} i^{|m|-|k|-|m-k|} A_j^k A_{n-j}^{m-k} \rho^j Y_j^{-k}(\alpha, \beta)}{A_n^m}, \tag{4.4.30}$$

where

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!\,(n+m)!}}, \tag{4.4.31}$$

and (ρ, α, β) is the spherical coordinate of $\vec{x}_0$.

If the translation vector $\vec{x}_0$ is parallel to the z-axis, then a considerable saving in evaluating Eq.(4.4.30) can be made. This corresponds to the values $\alpha = 0$, or π. In such case, Y_n^m is simply

$$Y_n^m = Y_n^0 \delta_{m0}. \tag{4.4.32}$$

Therefore, one can simplify the translator in Eq.(4.4.30) as

$$\vec{N}_n^m = \sum_{j=0}^{n} \frac{\vec{M}_{n-j}^m A_j^0 A_{n-j}^m \rho^j}{A_n^m}. \tag{4.4.33}$$

Numerically, the simplification represents a reduction from the dense matrix-vector multiplication to a sparse matrix-vector multiplication. We denote Eq.(4.4.33) with the reduced map T_{reduced}.

4.4.4 Conversion of multipole to local expansion and the translation operator of the local expansions

Suppose that the source particles are located inside an open ball $B\left(R, \vec{x}_0\right)$ and $\vec{x}_0$ is identified with the spherical coordinate (ρ, α, β), let $[\vec{M}_n^m]$ denote the set of multipole

coefficients in $B(R, \vec{x}_0)$. Let $\vec{x}$ with spherical coordinate (r, θ, ϕ) satisfy the inequality

$$\|\vec{x} - \vec{x}_0\| > R, \tag{4.4.34}$$

then the local expansion at the field point $\vec{x}$ is described by the series:

$$\vec{\phi} = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \vec{L}_n^m r^n Y_n^m(\theta, \phi), \tag{4.4.35}$$

where

$$\vec{L}_n^m = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \frac{\vec{M}_j^k i^{|m-k|-|m|-|k|} A_j^k A_n^m Y_{n+j}^{k-m}(\alpha, \beta)}{(-1)^j A_{n+j}^{k-m} \rho^{n+j+1}}. \tag{4.4.36}$$

This conversion is denoted by the map $Q_{\vec{x}_0}$. Furthermore, the reduced map Q_{reduced} is obtained by setting $\alpha = 0, \pi$ and substituting Eq.(4.4.32) to Eq.(4.4.36), i.e.

$$\vec{L}_n^m = \sum_{j=0}^{\infty} \frac{\vec{M}_j^m (-1)^{m+j} A_j^m A_n^m}{A_{n+j}^0 \rho^{n+j+1}}. \tag{4.4.37}$$

The translation operator $V_{\vec{x}_0}$ operating on the set of local expansion coefficients $[\vec{L}_n^m]$ in the ball $B(r_0, \vec{x}_0)$ with (ρ, α, β) identified as the spherical coordinate of $\vec{x}_0$ is similarly defined via the transformation of $[\vec{L}_n^m]$ to the new set of local expansion coefficients $[\vec{D}_n^m]$ in the ball $B(r_1, 0)$ with $B(r_0, \vec{x}_0) \subseteq B(r_1, 0)$. The elements of the new set are the result of the product of a dense matrix-vector multiplication:

$$\vec{D}_n^m = \sum_{j=n}^{\infty} \sum_{k=-j}^{j} \frac{\vec{L}_j^k i^{|k|-|k-m|-|m|} A_{j-n}^{k-m} A_n^m Y_{j-n}^{k-m}(\alpha, \beta) \rho^{j-n}}{(-1)^{j+n} A_j^k}. \tag{4.4.38}$$

The reduced map V_{reduced} is similarly defined by setting $k = m$ in the above sum, i.e.

$$\vec{D}_n^m = \sum_{j=n}^{\infty} \frac{\vec{L}_j^m A_{j-n}^0 A_n^m \rho^{j-n}}{(-1)^{j+n} A_j^m}. \tag{4.4.39}$$

4.4.5 Rotation operator for the CSHB functions

As previously mentioned, each of the operators in the preceded discussion represents a dense matrix-vector multiplication. When considering the number of operations that needs to be carried out in the FMM tree, the computational time quickly becomes untenable. The way to resolve this issue to employ the reduced maps instead. However, this involves introducing new machinery that will enable us to transform the *dense* maps to the reduced maps such as those in Eq.(4.4.33), Eq.(4.4.37) and Eq.(4.4.39). For this purpose, we will introduce the rotation operator, denoted by D, which is a linear map that corresponds to the following operation:

$$\tilde{M}_n^m = \sum_{m'=-n}^{n} D_n^{m',m} \vec{M}_n^{m'}, \tag{4.4.40}$$

where the set $[\tilde{M}_n^m]$ is the *rotated* multipole or local expansion coefficients and $D_n^{m',m}$ are the elements of the matrix D. Apparently, there are two ways to construct $D_n^{m,m'}$. One is through a series of elementary rotations on the principle axis, i.e. through the

Euler angles. The result is the *Wigner's D matrix* (Gumerov and Duraiswami, 2015). Specifically, if α, β and γ denote the Euler angles relative to the principle axes (via the zyz convention), then one may compute the rotation operator as:

$$D_n^{m,m'} = e^{-im'\alpha} d_n^{m,m'}(\beta)\, e^{-im\gamma}, \tag{4.4.41}$$

where the explicit form of $d_n^{m,m'}$ is given by the formula:

$$d_n^{m,m'} = \sqrt{(n+m')!\,(n-m')!\,(n+m)!\,(n-m)!} \times \sum_s \left(\frac{(-1)^{m'-m+s}\, c_\beta^{2n+m-m'-2s} s_\beta^{m'-m+2s}}{(n+m-s)!s!\,(m'-m+s)!\,(n-m'-s)!} \right), \tag{4.4.42}$$

here s traverses all integer values in such a way that the factorials are non-negative and $c_\beta = \cos(\beta/2), s_\beta = \sin(\beta/2)$. The other approach is to employ a coordinate transformation, in which the coordinate system is rotated in such a way that the z-axis of the original system is aligned parallel to an input vector in the transformed system. The latter proves to be more tractable as it avoids dealing with the Euler angles, which can be quite ambiguous. Here, we note down the key results without proof. The construction is based on certain recursive relations satisfied by the associated Legendre functions (see Choi et al. (1999) for detail).

Given a normalize vector $\vec{k}$, one determines the coordinate transformation matrix R under which $\vec{k}$ becomes the new z-axis in the transformed system. It can be shown that this operation can be decomposed by two elementary rotations, namely D_y and D_z, i.e.

$$R\left(\vec{k}\right) = D_y(\beta)\, D_z(-\alpha), \tag{4.4.43}$$

where

$$D_y(\beta) = \begin{pmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{pmatrix}, \tag{4.4.44}$$

and

$$D_z(-\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.4.45}$$

The angles α and β are related to the vector $\vec{k}$ as follows:

$$\alpha = \tan^{-1}\left(\frac{\vec{e}_y \cdot \vec{k}}{\vec{e}_x \cdot \vec{k}}\right), \quad \beta = \cos^{-1}\left(\vec{e}_z \cdot \vec{k}\right). \tag{4.4.46}$$

Once R is found, the complex matrix $\mathcal{D}$ formed from the elements of R is defined as follows:

$$\mathcal{D} = \begin{pmatrix} (R_{yy}+R_{xx})/2 & R_{xz}/\sqrt{2} & (R_{yy}-R_{xx})/2 \\ R_{zx}/\sqrt{2} & R_{zz} & -R_{zx}/\sqrt{2} \\ (R_{yy}-R_{xx})/2 & -R_{xz}/\sqrt{2} & (R_{yy}+R_{xx})/2 \end{pmatrix} + i \begin{pmatrix} (R_{yx}-R_{xy})/2 & R_{yz}/\sqrt{2} & -(R_{yx}+R_{xy})/2 \\ -R_{zy}/\sqrt{2} & 0 & -R_{zy}/\sqrt{2} \\ (R_{yx}+R_{xy})/2 & R_{yz}/\sqrt{2} & (R_{xy}-R_{yx})/2 \end{pmatrix}. \tag{4.4.47}$$

Subsequently, the following recursive relations were used to determine the elements of the rotation matrix D, i.e. Eq.(4.4.40):

$$D_n^{m',m} = a_n^{m',m}\mathcal{D}_{yy}D_{n-1}^{m',m} + b_n^{m',m}\mathcal{D}_{zy}D_{n-1}^{m',m-1} + b_n^{m',-m}\mathcal{D}_{xy}D_{n-1}^{m',m+1}, \quad -n+1 \leq m' \leq n-1, \tag{4.4.48}$$

where

$$a_n^{m',m} = \sqrt{\frac{(n+m)(n-m)}{(n+m')(n-m')}}, \quad b_n^{m',m} = \sqrt{\frac{(n+m)(n+m-1)}{2(n+m')(n-m')}}. \tag{4.4.49}$$

Also for $-n \leq m' \leq n-2$, the following recursive relation is valid:

$$D_n^{m',m} = c_n^{-m',m}\mathcal{D}_{yx}D_{n-1}^{m'+1,m} + d_n^{-m',m}\mathcal{D}_{zx}D_{n-1}^{m'+1,m-1} + d_n^{-m',-m}\mathcal{D}_{xx}D_{n-1}^{m'+1,m+1}, \quad -n \leq m' \leq n-2, \tag{4.4.50}$$

where

$$c_n^{m',m} = \sqrt{\frac{2(n+m)(n-m)}{(n+m')(n+m'-1)}}, \quad d_n^{m',m} = \sqrt{\frac{(n+m)(n+m-1)}{(n+m')(n+m'-1)}}. \tag{4.4.51}$$

Finally, in the range $-n+2 \leq m' \leq n$, we employ the recursion:

$$D_n^{m',m} = c_n^{m',m}\mathcal{D}_{yz}D_{n-1}^{m'-1,m} + d_n^{m',m}\mathcal{D}_{zz}D_{n-1}^{m'-1,m-1} + d_n^{m',-m}\mathcal{D}_{xz}D_{n-1}^{m'-1,m+1}. \tag{4.4.52}$$

Using Eq.(4.4.48), Eq.(4.4.50) and Eq.(4.4.52), the dense operators are equivalent to the composition of two rotations and their reduced counterpart. For example, the multipole translation operator $T_{\vec{x}_0}$ in Eq.(4.4.29) may be expressed as

$$T_{\vec{x}_0} = D^+\left(R\left(\vec{x}_0/\left\|\vec{x}_0\right\|\right)\right) T_{\text{reduced}} D\left(R\left(\vec{x}_0/\left\|\vec{x}_0\right\|\right)\right), \tag{4.4.53}$$

where D^+ denotes the complex conjugate transpose of D. It is worth mentioning that D is a unitary operator in the sense that it satisfies the relation:

$$I = D^+ D, \tag{4.4.54}$$

so that $D^{-1} = D\left(R^{-1}\right) = D^+$.

Other dense operators are similarly generalized using Eq.(4.4.53). Through the decomposition, the dense matrix-vector product is seen as three sparse matrix-vector multiplications. Indeed, the matrix D and the reduced matrices associated with the reduced operators are both stored as sparse matrices using the compressed row format.

4.4.6 Representing the multipole and local expansions using the real harmonic spherical functions

Up to this point, we have introduced enough mathematics to describe all of the important operations on the multipole and local expansions. The final discussion concerns with reducing the memory overhead in the FMM code. The motivation to use the *real* spherical harmonic functions stems from the fact that the code dedicates two floating-point numbers to cache the real and imaginary part of each expansion

coefficient. This results in the unnecessary waste of precious memory space in the GPU, where the memory of the current hardware is quite limited. In addition, since the final evaluation of the expansion produces real-numbers, so it is anticipated that the complex parts of the operations will, at some point, vanish. Therefore, it is much more efficient to identify and discard the complex part of any operation since they will be inconsequential to the final result. For this reason, we may represent the expansions (either multipole or local) via the real spherical harmonic function basis. Specifically, recall that the vector potential $\vec{\phi}$ is real and may be represented using CSHB functions of the form

$$\vec{\phi}\left(\vec{x}\right) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \vec{M}_n^m S_n^m\left(\vec{x}\right) \tag{4.4.55}$$

where $[\vec{M}_n^m]$ and S_n^m are both complex. Indeed, let us suppose there exists a set of real coefficient $[\vec{d}_n^m]$ and kernel R_n^m such that

$$\vec{\phi}\left(\vec{x}\right) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \vec{d}_n^m R_n^m\left(\vec{x}\right). \tag{4.4.56}$$

Writing Eq.(4.4.55) as

$$\vec{\phi} = \sum_{n=0}^{\infty} \sum_{m=0}^{n} \left(\vec{d}_n^m \left(\frac{S_n^m + S_n^{-m}}{2} \right) + \vec{d}_n^{-m} \left(\frac{S_n^m - S_n^{-m}}{2i} \right) \right), \tag{4.4.57}$$

and comparing coefficients, one can infer the following ($m > 0$):

$$\vec{d}_n^0 = \vec{M}_n^0, \quad \vec{d}_n^m = \vec{M}_n^m + \vec{M}_n^{-m}, \quad \vec{d}_n^{-m} = i\left(\vec{M}_n^m - \vec{M}_n^{-m}\right). \tag{4.4.58}$$

Similarly, the real kernel R_n^m is related to S_n^m as

$$R_n^m = \begin{cases} \frac{S_n^m + S_n^{-m}}{2} & m \geq 0, \\ \frac{S_n^{|m|} - S_n^{-|m|}}{2i} & m < 0. \end{cases} \tag{4.4.59}$$

Finally, it is desirable to obtain the real rotation $R_n^{m',m}$, which operates on the set of real coefficients, i.e.

$$\tilde{d}_n^m = \sum_{m'=-n}^{n} R_n^{m',m} \vec{d}_n^{m'}. \tag{4.4.60}$$

By applying Eq.(4.4.58), it can be shown that the elements of $R_n^{m',m}$ may be deduced as follows:

$$R_n^{m',0} = \begin{cases} D_n^{0,0} & m' = 0, \\ \frac{1}{2}\left(D_n^{m',0} + D_n^{-m',0}\right) & m' > 0, \\ \frac{i}{2}\left(D_n^{-|m'|,0} - D_n^{|m'|,0}\right) & m' < 0. \end{cases} \tag{4.4.61}$$

Moreover, for $m > 0$, we have

$$R_n^{m',m} = \begin{cases} D_n^{0,m} + D_n^{0,-m} & m' = 0, \\ \frac{1}{2}\left(D_n^{m',m} + D_n^{m',-m} + D_n^{-m',m} + D_n^{-m',-m}\right) & m' > 0, \\ \frac{i}{2}\left(D_n^{-|m'|,m} + D_n^{-|m'|,-m} - D_n^{|m'|,m} - D_n^{|m'|,-m}\right) & m' < 0, \end{cases} \tag{4.4.62}$$

and

$$R_n^{m',-m} = \begin{cases} i\left(D_n^{0,m} - D_n^{0,-m}\right) & m' = 0 \\ \frac{i}{2}\left(D_n^{m',m} - D_n^{m',-m} + D_n^{-m',m} - D_n^{-m',-m}\right) & m' > 0 \\ \frac{1}{2}\left(D_n^{|m'|,m} - D_n^{|m'|,-m} - D_n^{-|m'|,m} + D_n^{-|m'|,-m}\right) & m' < 0. \end{cases} \tag{4.4.63}$$

It should be noted that by transforming to the real rotation, the unitary property of D is loss. But considering the computational saving of halving the memory cost and computational load, this is a minor compromise.

4.4.7 Exact expression for the Biot-Savart induction and the strain field

Here, we present a derivation of an analytical expression for both the far-field Biot Savart induction (Eq.(3.1.12)) and the velocity strain expressed in the transpose formulation, namely the expression $\vec{\alpha} \cdot \nabla^T \vec{u}$, where $\vec{\alpha}$ is a constant vector and $\vec{u}$ is the Biot Savart velocity. Note that the strain can be alternatively expressed as:

$$\vec{\alpha} \cdot \nabla^T \vec{u} = \nabla\left(\vec{\alpha} \cdot \vec{u}\right). \tag{4.4.64}$$

First, we note that the local expansion of the vector potential in the real spherical harmonic basis can be recast as:

$$\vec{\phi}\left(r,\theta,\phi\right) = \sum_{n=0}^{\infty}\sum_{m=0}^{n} J_n^m\left(r,\theta\right)\vec{Q}_n^m\left(\phi\right), \tag{4.4.65}$$

where

$$J_n^m\left(r,\theta\right) = r^n d_n^m P_n^m\left(\cos\theta\right) \tag{4.4.66a}$$

$$\vec{Q}_n^m\left(\phi\right) = \vec{L}_n^m \cos m\phi + \vec{L}_n^{-m} \sin m\phi. \tag{4.4.66b}$$

Here, $\vec{L}_n^m$ denotes the real local expansion coefficients derived from Eq.(4.4.36). Let $\vec{u}_n^m$ denote the velocity at degree n and order m. Noting that

$$\vec{u} = \frac{1}{4\pi}\nabla \times \vec{\phi}, \tag{4.4.67}$$

then one obtains:

$$\vec{u}_n^m = \frac{1}{4\pi}\nabla \times \left(J_n^m\left(r,\theta\right)\vec{Q}_n^m\left(\phi\right)\right). \tag{4.4.68}$$

Using a combination of the chain rule and the anti-symmetric properties of the Levi-Civita symbol, it is a simple exercise to show that Eq.(4.4.68) is given by

$$4\pi\vec{u}_n^m = \frac{\partial J_n^m}{\partial r}\left(\nabla r \times \vec{Q}_n^m\right) + \frac{\partial J_n^m}{\partial \theta}\left(\nabla\theta \times \vec{Q}_n^m\right) + J_n^m\left(\nabla\phi \times \frac{\partial \vec{Q}_n^m}{\partial \phi}\right), \tag{4.4.69}$$

where (r,θ,ϕ) is related to Cartesian coordinates (x,y,z) as follows:

$$r = \sqrt{x^2+y^2+z^2}, \quad \cos\theta = \frac{z}{r}, \quad \tan\phi = \frac{y}{x}. \tag{4.4.70}$$

The only work here is to compute the partial derivatives of the J_n^m quantity. $\partial J_n^m/\partial r$ is straightforward and can be computed as follows:

$$\frac{\partial J_n^m}{\partial r} = nr^{n-1} d_n^m P_n^m(\cos\theta) = \frac{nJ_n^m}{r}. \tag{4.4.71}$$

The latter expression in the above equation is preferred since it avoids evaluating the associated Legendre function again. For $\partial J_n^m/\partial\theta$, we make use of the backward recursive relations satisfied by the associated Legendre functions, i.e.

$$(n+m+1)(n-m) P_n^m(x) = -\frac{2(m+1)x}{\sqrt{1-x^2}} P_n^{m+1}(x) - P_n^{m+2}(x), \tag{4.4.72}$$

and

$$(x^2-1)\frac{dP_n^m}{dx}(x) = \sqrt{1-x^2} P_n^{m+1}(x) + mxP_n^m(x). \tag{4.4.73}$$

Using Eq.(4.4.72) together with Eq.(4.4.73), it is easy to show that

$$\frac{\partial J_n^m}{\partial\theta} = r^n d_n^m \frac{\partial P_n^m}{\partial\theta} = r^n d_n^m \frac{\partial P_n^m}{\partial\cos\theta}\frac{\partial\cos\theta}{\partial\theta} = -\sin\theta J_n^m \left(\frac{1}{P_n^m}\frac{\partial P_n^m}{\partial\cos\theta}\right). \tag{4.4.74}$$

Algorithmically, the code proceeds to compute Eq.(4.4.74) by first storing the values of P_n^{m+1} and P_n^{m+2}, which would have computed during the previous iteration. P_n^m is obtained by solving Eq.(4.4.72). Finally, $dP_n^m/d(\cos\theta)$ is obtained by substituting the values of P_n^m and P_n^{m+1} to Eq.(4.4.73).

In order to derive the analytical expression for the strain (Eq.(4.4.64)), we make use of the following observations. The general expression for the second order derivatives of r is given by

$$\frac{\partial^2 r}{\partial x_i \partial x_j} = \frac{1}{r}\left(\delta_{ij} - \frac{\partial r}{\partial x_i}\frac{\partial r}{\partial x_j}\right). \tag{4.4.75}$$

The corresponding results for $\partial^2\theta/\partial x_i\partial x_j$ and $\partial^2\phi/\partial x_i\partial x_j$ are as given follows:

$$\frac{\partial^2\theta}{\partial x_i\partial x_j} = \frac{1}{r^2\sin\theta}\left(\delta_{iz}\frac{\partial r}{\partial x_j} + \delta_{jz}\frac{\partial r}{\partial x_i} + \delta_{ij}\frac{\partial r}{\partial z} - 3\frac{\partial r}{\partial x_i}\frac{\partial r}{\partial x_j}\frac{\partial r}{\partial z}\right) - \cot\theta\frac{\partial\theta}{\partial x_i}\frac{\partial\theta}{\partial x_j}, \tag{4.4.76}$$

and

$$\frac{\partial^2\phi}{\partial x_i\partial x_j} = \|\nabla\phi\|^2(\delta_{jx}\delta_{iy} - \delta_{jy}\delta_{ix}) - 2\left(\frac{\partial\phi}{\partial y}\delta_{jx} - \frac{\partial\phi}{\partial x}\delta_{jy}\right)\frac{\partial\phi}{\partial x_i}. \tag{4.4.77}$$

Next, we project the velocity $\vec{u}_n^m$ onto the direction $\vec{\alpha}$ and define the variables f_1, f_2 and f_3, i.e.

$$4\pi(\vec{\alpha}\cdot\vec{u}_n^m) = \frac{\partial J_n^m}{\partial r}f_1 + \frac{\partial J_n^m}{\partial\theta}f_2 + J_n^m f_3, \tag{4.4.78}$$

where

$$f_1 = \vec{\alpha}\cdot\nabla r\times\vec{Q}_n^m, \quad f_2 = \vec{\alpha}\cdot\nabla\theta\times\vec{Q}_n^m, \quad f_3 = \vec{\alpha}\cdot\nabla\phi\times\frac{\partial\vec{Q}_n^m}{\partial\phi}. \tag{4.4.79}$$

Using Eq.(4.4.75), Eq.(4.4.76) and Eq.(4.4.77), the grad of the variables f_i are obtained:

$$\nabla f_1 = \frac{1}{r}\left(\vec{Q}_n^m\times\vec{\alpha} - f_1\nabla r\right) + \left(\vec{\alpha}\cdot\nabla r\times\frac{\partial\vec{Q}_n^m}{\partial\phi}\right)\nabla\phi, \tag{4.4.80}$$

$$\nabla f_2 = \frac{1}{r^2 \sin\theta}\left(f_1\vec{e}_z + \left(\vec{e}_z \cdot \vec{Q}_n^m \times \vec{\alpha}\right)\nabla r + (\vec{e}_z \cdot \nabla r)\left(\vec{Q}_n^m \times \vec{\alpha} - 3f_1\nabla r\right)\right) - \cot\theta f_2 \nabla\theta + \left(\vec{\alpha}\cdot\nabla\theta \times \frac{\partial \vec{Q}_n^m}{\partial\phi}\right)\nabla\phi. \quad (4.4.81)$$

$$\nabla f_3 = \left(\vec{\alpha}\cdot\nabla\phi \times \frac{\partial^2 \vec{Q}_n^m}{\partial\phi^2}\right)\nabla\phi + \|\nabla\phi\|^2\left((\vec{e}_z\cdot\vec{\alpha})\frac{\partial\vec{Q}_n^m}{\partial\phi} - \left(\vec{e}_z\cdot\frac{\partial\vec{Q}_n^m}{\partial\phi}\right)\vec{\alpha}\right) + 2\left(\nabla\phi\times\vec{e}_z\right)\cdot\left(\vec{\alpha}\times\frac{\partial\vec{Q}_n^m}{\partial\phi}\right)\nabla\phi. \quad (4.4.82)$$

Thus, by the application of the product rule, the strain is given by:

$$4\pi\nabla\left(\vec{\alpha}\cdot\vec{u}_n^m\right) = \nabla\left(\frac{\partial J_n^m}{\partial r}\right)f_1 + \frac{\partial J_n^m}{\partial r}\nabla f_1 + \nabla\left(\frac{\partial J_n^m}{\partial\theta}\right)f_2 + \frac{\partial J_n^m}{\partial\theta}\nabla f_2 + \left(\nabla J_n^m\right)f_3 + J_n^m\nabla f_3, \quad (4.4.83)$$

where

$$\nabla\left(\frac{\partial J_n^m}{\partial r}\right) = \frac{\partial^2 J_n^m}{\partial r^2}\nabla r + \frac{\partial^2 J_n^m}{\partial\theta\partial r}\nabla\theta, \quad (4.4.84a)$$

$$\nabla\left(\frac{\partial J_n^m}{\partial\theta}\right) = \frac{\partial^2 J_n^m}{\partial r\partial\theta}\nabla r + \frac{\partial^2 J_n^m}{\partial\theta^2}\nabla\theta, \quad (4.4.84b)$$

$$\nabla J_n^m = \frac{\partial J_n^m}{\partial r}\nabla r + \frac{\partial J_n^m}{\partial\theta}\nabla\theta. \quad (4.4.84c)$$

Although we have used the local series for derivation, there is a straightforward extension to handle the multipole case. Having introduced the translation and conversion operators both in 2D and 3D, we now have all of the ingredients needed to construct the FMM algorithm, which will be the subject of discussion in the subsequent sections.

4.5 Initial expansion

The *initial expansion* component, as in Figure (4.1), of the FMM algorithm is responsible to convert the source particles in each childless box into a multipole expansion using Eq.(4.3.13) or Eq.(4.4.27). Through the modified tree construction algorithm outlined in Section 4.2, this process is parallelized by assigning each thread to one childless box. The thread traverses the particle data via the access method outlined in Section 4.2.1. For each source particle the thread encounters, the thread computes the multipole expansion coefficients and accumulates them to the predefined `multipoleExpansionArray`. The form of the `multipoleExpansionArray` differs in 2D and 3D. For example, in 2D, the `multipoleExpansionArray` is simply a C-array whose real and imaginary parts of the coefficients are stored in a contiguous order, whereas for the 3D case, it consists of 3 C-arrays for the three-components. This phase is illustrated in Algorithm 4.7. In 3D, however, the complex coefficients are transformed to the real coefficients according to Eq.(4.4.58) during this step.

Algorithm 4.7: Initial Expansion (P2M)

input : A reference to the tree object (tree), the set of particle data (pData), a truncation number (P), permutationArray, boxPointerArray, particleBinArray and particlePointerArray
output: The multipole coefficient array (multipoleExpansionArray)

```
// Assume that the tree has already constructed
for each childless non-empty box b in tree do
    // obtain the Morton index and the refinement level of b
    b.getBoxReferenceIndex(Imorton, RefLvl)
    // obtain the number of source particles in b by passing the identifier to
       the particleBinArray
    numberOfSrcParticles = particleBinArray.idx(Imorton,RefLvl)
    pptr = particlePointerArray.idx(Imorton, RefLvl)
    for each i from 0 to numberOfSrcParticles- 1 do
        // Create a reference to the particle
        p = pData.getParticleReference(permutationArray.idx(pptr))
        // Compute the multipole coefficient and consolidate the coefficients
           to the existing multipoleExpansionArray
        ComputeMultipoleCoefficients(multipoleExpansionArray, P, p,
         Imorton, RefLvl)
        // increment the particle pointer
        pptr ← pptr + 1
```

4.6 Upward pass

The *upward pass* involves applying the translation operator on the multipole expansion coefficients of the children boxes to their parent boxes. The modified expansion coefficients are then accumulated at the parent boxes. The parallelization is handle by letting each thread handle a single translation for each children box. The result is stored to a temporary array. When all threads have finished processing the translations, the results are sequentially added to the parent boxes. This process is illustrated in Algorithm 4.8.

4.7 Downward pass

This phase is described in two stages. The first stage is to convert the set of multipole expansion coefficients of the List 2 and List 4 partition boxes to a set of local expansion coefficients via the conversion operator as in Eq.(4.3.37) or Eq.(4.4.36). The idea of this stage is to account for the local far-field influence of the target box. This stage is described in Algorithm 4.9. The algorithm assumes the List partitions have already been defined using Algorithm 4.4. The local expansion coefficients are stored to a dedicated array that we denote it by `localExpansionArray`. The elements are stored in a contiguous order to reduce the number of fetch operations. At the end of this stage, all non-empty boxes in the FMM tree would be allocated a local expansion, which corresponds to the local far-field influence. Subsequently, we may start the second stage of the phase in which the local expansion coefficients are recursively translated from the parent boxes to their children boxes. This is equivalent to the inclusion of the larger hierarchical far-field influences. This step is illustrated in Algorithm 4.10. The parallelization of this stage proceeds in a similar manner

Algorithm 4.8: Upward Pass (M2M)

input : A reference of the tree object (tree), the truncation number (P), the effective refinement level (np) and the multipoleExpansionArray
output: the modified multipoleExpansionArray
for *each refinement level* k *from* np *to* 1 **do**
  for *each non-empty box* b *at refinement level* k **do**

```
// Obtain the box reference from b's parent
pb = b.getParentReference()
// Obtain the translation vector
transVector = b.getBoxCoord() - pb.getBoxCoord()
// Copy the multipole expansion coefficients from box b to a temporary
   variable tmp by passing the 'start' and 'end' iterators to the
   multipoleExpansionArray
CopyElementToArray(multipoleExpansionArray, tmp, startIterator,
 endIterator)
// Apply the translation operator to tmp.  Note the caller modifies the
   array elements in tmp
TranslateMultipoleCoefficients(tmp, P, transVector)
// Modify the iterators so that they point to the begin and end
   position of pb's multipole expansion
// Combine the modified expansion to pb's expansion
AccumulateExpansion(multipoleExpansionArray, tmp, startIterator,
 endIterator)
```

as in Algorithm 4.8, in which each thread handles a children box and performs a translation from their parent's local expansion.

4.8 Final evaluation of the field

The evaluation of the Biot Savart and the velocity strain occurs at this final stage of the FMM routine. There are three types of evaluations that we need to account for: the evaluation of the near-field influence (i.e. the List 1 partition), the evaluation of the local expansion (i.e. the far-field) and the evaluation of the far-field given by the List 3 partition, which we may coin the *local influence*. Using the access method, the evaluation at the particle locations is parallelized by assigning each thread with a particle reference. The thread then proceeds to evaluate the direct influence using the mollified Biot-Savart induction, i.e. Eq.(3.1.18), follows by the evaluation of the local expansion using Eq.(4.3.28) or Eq.(4.4.68). The final outcome of the algorithm is a velocity array (`velocityArray`) and the velocity strain array (`strainArray`) computed at the particles' locations. Moreover, we note that `strainArray` is empty in 2D, which is a consequence of the fact that the stretching term varnishes in 2D. The near field evaluation is described in Algorithm 4.11. The evaluation of the local expansion and the local influence proceeds in a very similar fashion and therefore we will not be presenting them here.

Algorithm 4.9: Downward Pass Conversion (M2L)

input : a reference to the tree object (tree), the truncation number (P), the multipoleExpansionArray and the localExpansionArray
output: the modified localExpansionArray
for *each non-empty box* b *in* tree **do**
    tmplist = b.getList2();
    for *each box* c *in* tmplist **do**
        // Obtain the translation vector
        transVector = c.getBoxCoord() - b.getBoxCoord();
        // Copy the c's multipole coefficients to the variable tmpArray. Here startIterator and endIterator point to c's start and end position in the multipole array, respectively.
        CopyElementToArray(multipoleExpansionArray, tmpArray, startIterator, endIterator);
        // Apply the conversion operator.
        ConvertMultipoleToLocal(tmpArray, P, transVector);
        // Copy the result to the localExpansionArray. startIterator and endIterator should be appropriately modified so that they point to the b's local expansion in the local array
        AccumulateExpansion(localExpansionArray, tmpArray, startIterator, endIterator);
    repeat for the List 4 partition of b;

Algorithm 4.10: Downward Pass Translation (L2L)

input : a reference to the tree object (tree), the truncation number (P), the effective refinement number and the localExpansionArray
output: the modified localExpansionArray
for *each refinement level* k *from* 2 *to* np **do**
    for *each non-empty box* b *at refinement level* k **do**
        // Obtain the box reference of b's parent
        pb = b.getParentReference();
        // Obtain the translation vector
        transVector = pb.getBoxCoord() - b.getBoxCoord();
        // Copy the local expansion of pb to a temporary variable tmpArray using the iterators
        CopyElementToArray(localExpansionArray, tmpArray, startIterator, endIterator);
        // Apply the translation operator to tmpArray.
        TranslateLocalCoefficients(tmpArray, P, transVector);
        // Accumulate the local expansion to b's local expansion. Assume that startIterator and endIterator have appropriately modified
        AccumulateExpansion(localExpansionArray, tmpArray, startIterator, endIterator);

Algorithm 4.11: Near Field Evaluation

```
input : a modified tree object (tree), particle data (pData), velocityArray,
        strainArray, permutationArray, boxPointerArray, particleBinArray and the
        particlePointerArray
output: modified velocityArray and strainArray
counter = 0
for each source particle p1 in pData do
    Imorton = boxPointerArray.idx(counter)
    RefLvl = boxPointerArray.lvl(counter)
    // Obtain the box reference from tree with the identifier (Imorton, RefLvl)
    b = tree.getBoxReferenceFromIndex(Imorton, RefLvl)
    for each box c in b.getList1() do
        // Obtain the Morton index and the refinement for the box c
        c.getBoxReferenceIndex(Jmorton, Jlvl)
        // Get the first sorted particle pointer to the box c as well as the
           number of source particle
        pptr = particlePointerArray.idx(Jmorton, Jlvl)
        nsrc = particleBinArray.idx(Jmorton, Jlvl)
        for i from 0 to nsrc- 1 do
            // Obtain the reference to the source particle corresponding to pptr

            p2 = pData.getParticleReference(permutationArray.idx(pptr))
            // Evaluate the direct influence between p1 and p2 and store the
               result to the velocityArray and strainArray. The actual particle
               index is required to be ensured that the data is stored at the
               right memory location
            DirectEvaluate(velocityArray, strainArray, p1, p2,
             permutationArray.idx(counter))
            // increment pptr
            pptr ← pptr + 1
    // increment counter
    counter ← counter + 1
```

4.9 Conclusion

The FMM calculation is an important and necessary component to the current turbine simulation code as it substantially reduces the simulation time by an order of magnitude. This reduction allows the simulation to reach steady state much faster. To achieve such an accomplishment, we have developed a new parallelized data structure whereby communication between vortex elements and the underlying FMM tree is efficiently facilitated by means of maintaining several large arrays. Moreover, all parts of the FMM code are implemented natively on the GPU. In addition to the data structure, we have introduced several important mathematical concepts that play pivotal roles in allowing us to perform translation and conversion operations on the multipole coefficients. This is a crucial step in the FMM algorithm as it allows the code to achieve the $\mathcal{O}(N)$ asymptotic time. Moreover, the rotation operator in the three-dimensional calculation decomposes the dense matrix vector multiplication to three sparse matrix vector products, which further reduces the computational cost.

In this chapter, an important contribution to the literature has been made by deriving an analytical expression for the velocity strain in terms of the real spherical harmonic basis functions. This analytical derivation has not been found in the literature and therefore it will prove very useful for improving the efficiency of the *final evaluation* phase of the FMM routine. Previously, investigators have used the central difference approach to approximate the strain field. This approach is not only marred by inaccuracies, but it also proved to be inefficient as it generally requires 2 evaluations to obtain the partial derivative in each direction. The presented work in this chapter mitigates this deficiency entirely. The result is a simple and efficient evaluation routine that makes use of the stable backward recursive formula of the associated Legendre functions.

Chapter 5

Validation of the vortex particle methods and performance measurements

Several validation cases are presented in this chapter, which serve to illustrate that the new proposed vortex particle method can provide a good and accurate representation of the flow physics in a variety of flow problems. Additionally, the GPU performance is analysed by measuring the performance gain on the most intensive parts of the simulation which are denominated by the evaluation of the Biot Savart induction. The results derived from this analysis will impact on future works in terms of optimizing the GPU parameters in the code.

5.1 Dynamic evolution of an inviscid vortex ring

Due to the collective influence of the vorticity contained within the vortex ring, in the absence of external flow the vortex ring undergoes a steady translation along the axis of symmetry. The self-induced velocity field due to an inviscid vortex ring is analysed in this section. The reason for investigating its kinematic behaviour is due to the fact that under the thin core approximation the ring as a whole undergoes a constant translation whose velocity approaches asymptotically to the theoretical expression (Sullivan et al., 2008):

$$U = \frac{\Gamma}{4\pi R}\left(\log\left(\frac{8R}{\sigma}\right) - \frac{1}{2}\right), \tag{5.1.1}$$

where Γ is the circulation, R is the ring radius and σ is the core radius. The self induced velocity is thus oriented in the normal direction to the ring plane. The convenience of Eq.(5.1.1) provides a simple test case for the three-dimensional code in the absence of viscosity.

5.1.1 Ring discretization

Let $\vec{\omega}$ denote the vorticity field of the ring. Under the thin core approximation (i.e. $\sigma \ll R$), one may assume that $\vec{\omega}$ can be written as

$$\vec{\omega} = \omega_0 \vec{e}_\theta, \tag{5.1.2}$$

where $\vec{e}_\theta$ is the azimuthal vector relative to the ring (see Figure (5.2)) and ω_0 is a constant. By imposing that the circulation be constant. i.e.

$$\int_S \vec{\omega} \cdot \vec{e}_\theta dS = \Gamma, \tag{5.1.3}$$

where S represents the cross-sectional area of the ring, one may deduce that:

$$\omega_0 = \frac{\Gamma}{\pi\sigma^2}. \tag{5.1.4}$$

The ring is discretized in two stages. In stage one, the cross-section of the ring is first discretized by a set of '2D' particles. The construction of the 2D slice is subject to the constraint that the total area of the particle must add up to the area of the circle and each particle carries the same area. An example of such discretization scheme is given in Figure (5.1). Subsequently, the slice is copied around a circle with

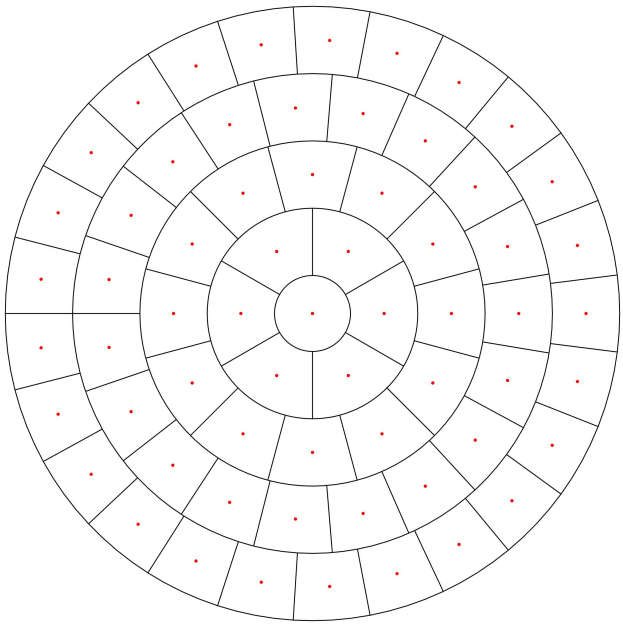


FIGURE 5.1: An example of an equal-area discretization for the 2D slice.

a prescribed normal to produce the effective particle discretization. Each particle is associated with the tuple $(\vec{\alpha}_j, \vec{x}_j)$, which characterises the vorticity field. Initially, the circulation of the j-th particle may be assigned by:

$$\vec{\alpha}_j = \frac{\Gamma \mathrm{Vol}_j}{\pi\sigma^2}\vec{e}_{\theta_j}. \tag{5.1.5}$$

where Vol_j denotes the volume of the particle.

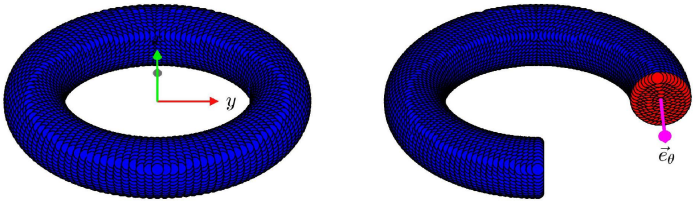


FIGURE 5.2: The three-dimensional ring is constructed by duplicating N_R copies of the cross-section around a central axis to form a torus (shown on the left). The vortex elements at each sectional station are assigned a constant circulation vector that is proportional to the circumferential unit vector $\vec{e}_\theta$ of that station (shown on the right).

5.1.2 A first order integration scheme in time

Under the collective influence of the vortex element, the ring is propelled by a non-zero induction field along the normal of the ring plane, which is taken to be $\vec{k} = (1,0,0)^T$. The position of the ring is tracked by measuring the mean position of the vortex particles in each instance of time. In order to reduce the computational cost, we have adopted a straightforward first order explicit integration scheme for the position $\vec{x}_j$ and the circulation $\vec{\alpha}_j$. Subsequently, when the Biot-Savart step is completed, the position and the circulation of the j-th particles are updated according to the following discretized formula:

$$\vec{x}_j^{n+1} = \vec{x}_j^n + \delta t\, \vec{u}\left(\vec{x}_j^n\right), \quad \vec{\alpha}_j^{n+1} = \vec{\alpha}_j^n + \delta t\, \vec{\alpha}_j \cdot \nabla^T \vec{u}, \tag{5.1.6}$$

where the superscript, n, indicates the value of the variable at the n-th time-step, and δt is the time-stepping size, which is currently set to $\delta t = 0.01$. Note here that we have employed the transpose scheme for the stretching term for its robust conservation properties (see Cottet and Koumoutsakos (2008) for an in-depth discussion).

5.1.3 Discussion

To validate the new proposed code, the averaged position of the ring is tracked at each time step. Figure (5.3a) shows the x-value of the position for a short time. Due to the variation of the particles' position, a linear regression was fitted to compute the average velocity of the ring. In the computation, the slope on the fitted line is about $\bar{U}_{\text{computed}} = 0.55177$. This is to be compared to the theoretical value of $U_{\text{theoretical}} = 0.5473$, which produces a relative error of 8×10^{-3}. However, for $t > 0.6$, a significant departure from the theoretical value was observed. On a close inspection of the particles' distribution, one may suggest that the discrepancy might be due to the core expansion experienced by the ring. Evidently, by sampling the velocity data of the ring in the $x - z$ plane at different time-intervals, one observes that the core velocity decreases monotonically over time (Figure (5.4)), thus lowering the self-induced speed. The parameter for this simulation is listed in Table 5.1. In order to maintain the stability of the simulation, particles are re-meshed every 5 steps using the M_4' scheme (Section 3.1.4). In addition, a form of population control was implemented to avoid excessive particle growth during the re-meshing step. A

vorticity cut-off value ω_0 was used so that only particles whose magnitude of the circulation vector greater than ω_0 are kept. In conclusion, the computed result using the 3D FMM code compares favourably to the theoretical result, which validates the new FMM code in Chapter 4.

param	value	description
δt	1×10^{-2}	time step size
h	1.12×10^{-2}	grid size
N_0	1.77×10^{4}	initial particle number
f_{remesh}	5	re-meshing frequency
P	7	FMM expansion order
ω_0	1×10^{-5}	vorticity cut-off

TABLE 5.1: Parameters used in the simulation of the inviscid vortex ring.

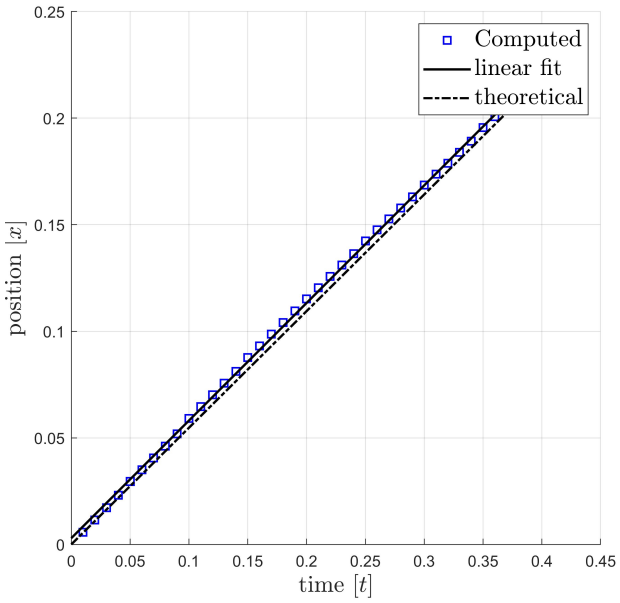
5.2 Partition error of the FMM code

One particular form of error in the current FMM code is associated with the normalization procedure outlined in Section 4.2. As previously mentioned, the physical vortex particles are transformed to fit within a unit square or cube by applying a translation and a scaling operations. Although the convergence error was known to scale with the truncation number P, it is unfortunate that this scaling depends on the type of particle distribution. In the worst case of scenario, the scaling behaviour of the error might incur a substantially large scaling factor, as is evidence in Figure (5.5), in which the particle partition produces an overflow of particles that results in the creation of excessively large boxes to accommodate the excess particle counts. The result is a slow convergent behaviour that demands a relatively large truncation number to suppress the resulting error. Indeed, if one samples the error as a function of the truncation number P, it is readily seen that the error evaluated at field points that are outside of the radius of divergence for the larger boxes improves substantially, at least by several orders of magnitudes, as compared to the points that are inside. Although this is an extreme form of the particle distribution which seldom occurs in real simulations, but it does pose an inherent weakness to the FMM code, which has not been previously documented in the literature. One resolution to this issue is to judiciously introduce an offset in the normalization procedure in such a way that over-flow should altogether be avoided. However, given the complexity of the undertaking, this weakness is presently left untreated.

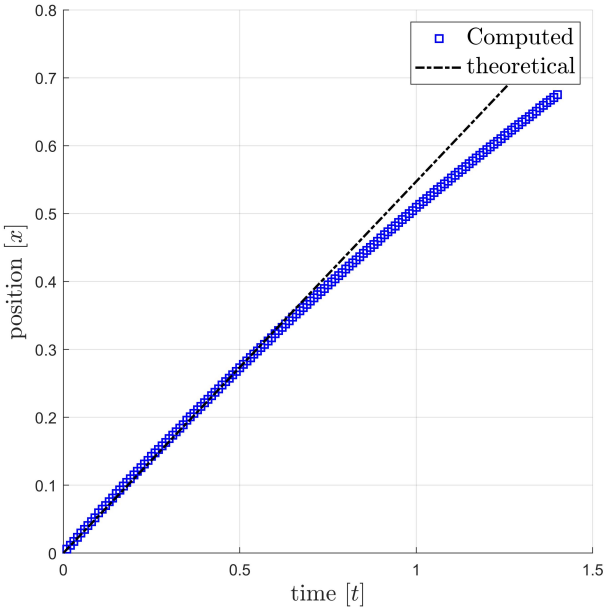
Comparatively, if over-flow is not present in the partitioned domain, convergence exhibits the expected property that is typical to the usual FMM error behaviour. Indeed, by computing the NRMSD (see Section 3.2.4), Table 5.2 displays a clear symptom of the detrimental effect of over-flow in the partition algorithm.

particle # (n)	truncation #(P)	Over-flow	u-NRMSD	v-NRMSD
2000	15	yes	1.028×10^{-3}	1.464×10^{-3}
2000	15	no	3.156×10^{-6}	4.000×10^{-6}

TABLE 5.2: shows the adverse effect of over-flow in the particle distribution, which could reduce the accuracy of the code by several orders of magnitude.

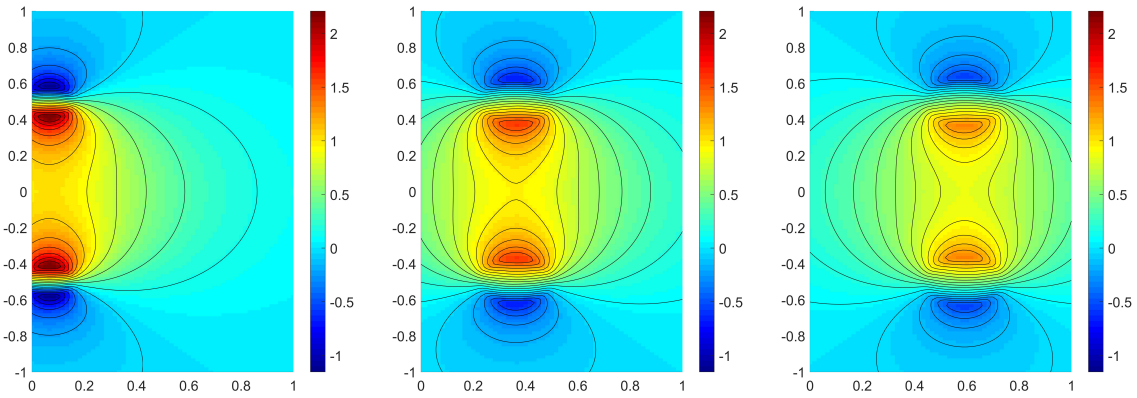


(A)



(B)

FIGURE 5.3: The computed averaged position of the ring along the x axis. (A) A short time comparison with the theoretical result (Eq.(5.1.1)). The average velocity is calculated by fitting a linear regression to the position data. (B) Significant deviation from the theoretical result was observed for $t > 0.6$. This may be due to the fact that the core experiences a constant expansion.



(A) $t = 0.11$ (B) $t = 0.68$ (C) $t = 1.19$

FIGURE 5.4: Velocity contours in the x-component sampled at different times. Clearly, the core velocity is decreasing which is associated with the core expansion.

5.3 Inviscid evolution of the (2:1) elliptical vortex patch (2D)

The two-dimensional vortex particle method is validated in this section and we present a parametric study on the effect of the various simulation parameters that affect the solution. To validate our approach, we simulated the inviscid evolution of an elliptical vortex patch with a non-uniform vorticity profile given by the following formula:

$$\omega(r,\phi) = \Lambda \begin{cases} 1 - f_q\left(r/R_0(\phi)\right), & \text{if } r/R_0 < 1, \\ 0, & \text{otherwise,} \end{cases} \tag{5.3.1}$$

where $f_q(z) := \exp\left(-(q/z)\exp\left(1/(z-1)\right)\right)$ and R_0 is a function of the azimuthal angle only. Here Λ and q are both constants.

This initial vorticity profile was investigated by Koumoutsakos (1997) and this serves as a benchmark case for the validation study. The choice for this flow is made based on the fact that the initial simple vortex structure will subsequently evolve into a complex one through the process of filamentation. This mechanism is initialized by the differential rotation of the vorticity field in the flow, which is akin to the Kelvin-Helmholtz type instability. As a result, the ellipticity of the vortex is reduced and approaches to that of a circular configuration. The aim of this section is to accurately capture this transition.

5.3.1 Initial particle distribution

Following Koumoutsakos (1997), the values $q = 2.56085$, $\Lambda = 20$ were used. The R_0 function represents the radial length from the boundary of the ellipse and is given by the following formula:

$$R_0(\phi) = r_0\sqrt{1 + (\lambda^2 - 1)\sin^2 f(\phi)}, \tag{5.3.2}$$

where $f(\phi) := \tan^{-1}\left(\lambda^{-1}\tan\phi\right)$ and λ is the aspect ratio of the vortex. Currently, the value of $r_0 = 0.8$ and $\lambda = 2$ were used.

The initial vorticity distribution is discretized by a particle quadrature rule where the quadrature nodes are the particles' positions. To initialize the discretization,

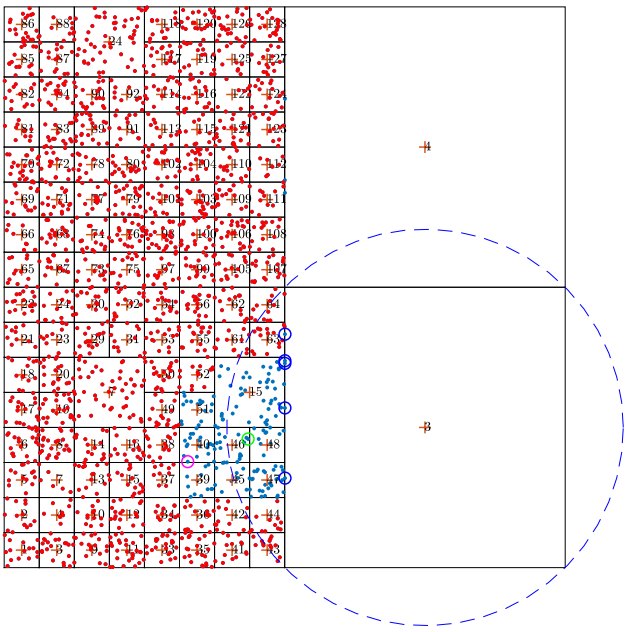
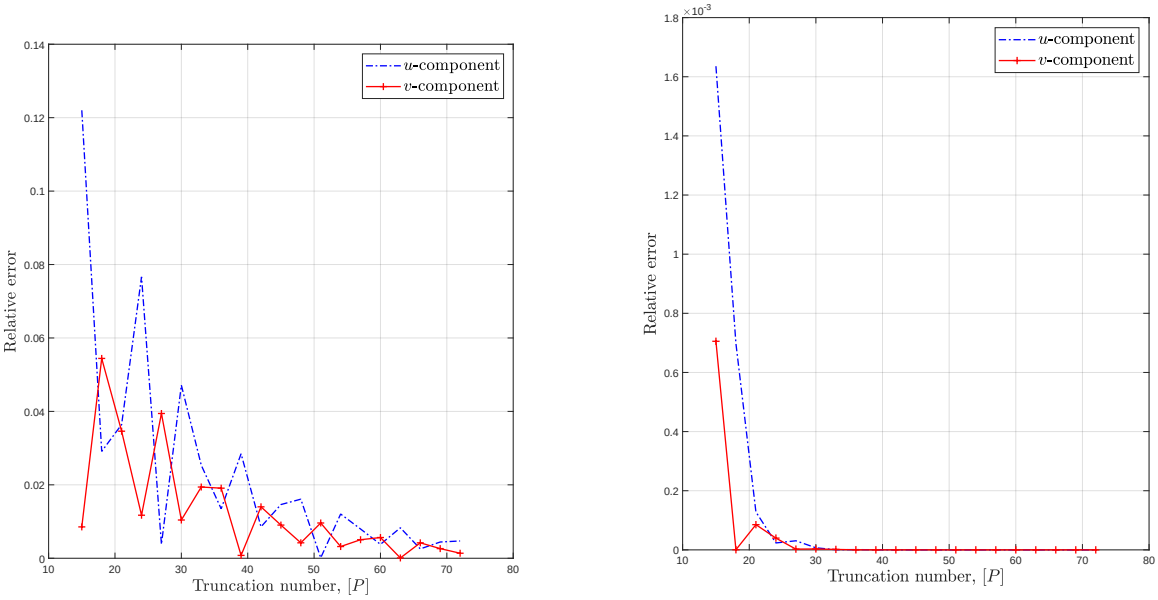


FIGURE 5.5: The domain partition results in the creation of large boxes to accommodate the excess particles (marked by the blue circles). In particular, convergence is severely hindered for field points inside the radius of divergence of the large boxes. In fact, the relative error curves have been computed at two reference points that are relatively close (marked by the green and magenta circle). The result is presented in Figure (5.6).

a uniform grid with grid spacing $(\delta x, \delta y)$ is over-laid on the support of Eq.(5.3.1). In order to reduce the initial discretization error, the initial particle circulations Γ_i were *not* directly assigned by $\Gamma_i = \omega(x_i, y_i)\,\delta x \delta y$, instead the following system of equations were solved for Γ_i:

$$\sum_{j=0}^{N-1} A_{ij}\Gamma_j = \omega(x_i, y_i) \tag{5.3.3}$$

where $A_{ij} := \xi_\epsilon(x_i - x_j, y_i - y_j)$ and ξ_ϵ is the smooth Gaussian core function discussed in Section 3.1.1. However, due to the size of the resulting matrix equation, Eq.(5.3.3) is currently solved using the successive over-relaxation (SOR) iterative



(A) (B)

FIGURE 5.6: The normalized error curve as a function of the truncation number P at two field locations. (A) the field point is located inside the radius of divergence as outlined in Figure (5.5). (B) the field point is outside, which drastically improves the result by an order of magnitude.

scheme. To validate this step of the simulation, we compute the exact vorticity moment parameter J_{nm} and compare those from the particle discretization. Precisely, the vorticity moment parameter is defined by

$$J_{nm} = \int \int_{\mathbb{R}^2} \omega(x,y)\, x^n y^m \, dx\, dy. \tag{5.3.4}$$

In order to numerically evaluate the integral in Eq.(5.3.4), it was shown that it is necessary to transform it to the (r,ϕ) coordinate following to the (u,v) coordinate where $u = r/R_0(\phi), v = \phi$, so that Eq.(5.3.4) becomes

$$J_{nm} = \int_0^{2\pi} R_0^{n+m+2}(v) \sin^m v \cos^n v \, dv \int_0^1 u^{n+m+1} F(u) \, du. \tag{5.3.5}$$

Here, $F(u) = \Lambda\left(1 - f_q(u)\right)$. A high-order numerical scheme is then employed to integrate Eq.(5.3.5).

For the particle discretization, the moment parameter may be expressed as:

$$J_{nm}^p = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \Gamma_j \xi_\epsilon \left(x_i - x_j, y_i - y_j\right) x_i^n y_i^m \tag{5.3.6}$$

In practice, however, the double sum is not evaluated directly and instead it is assumed that the non-negligible contribution from each particle is due to its immediate neighbourhood so that a fast tree-code can be employed (see Chapter 4). For this

	Eq.(5.3.5)	Eq.(5.3.6)
J_{00}	22.119502439638271	22.119502417163982
J_{11}	0.000000000000000	0.000000000000002
J_{02}	5.280561671991749	5.280561673576734
J_{20}	1.320140417997938	1.320140416374350

TABLE 5.3: The vorticity moment parameters computed between Eq.(5.3.5) and Eq.(5.3.6). The absolute error is in the order of $\mathcal{O}\left(10^{-6}\right)$, which is sufficient for the current purpose.

reason, the code permits a real-time monitoring of those variables. For comparison between the two approaches, we tabulated the values of J_{00}, J_{11}, J_{02} and J_{20} in Table 5.3. The absolute errors of these calculations are of the order of $\mathcal{O}\left(10^{-6}\right)$, which is sufficient for the current simulation.

For a grid size $\delta x = \delta y = 1.1 \times 10^{-2}$, the support of the initial vorticity field is resolved by a total number of $N(0) = 35348$ particles. Each particle is assigned a core radius given by

$$\epsilon = 0.9 \times \delta x. \tag{5.3.7}$$

Figure (5.7) shows the initial circulation of the particle field used to resolve the vorticity field.

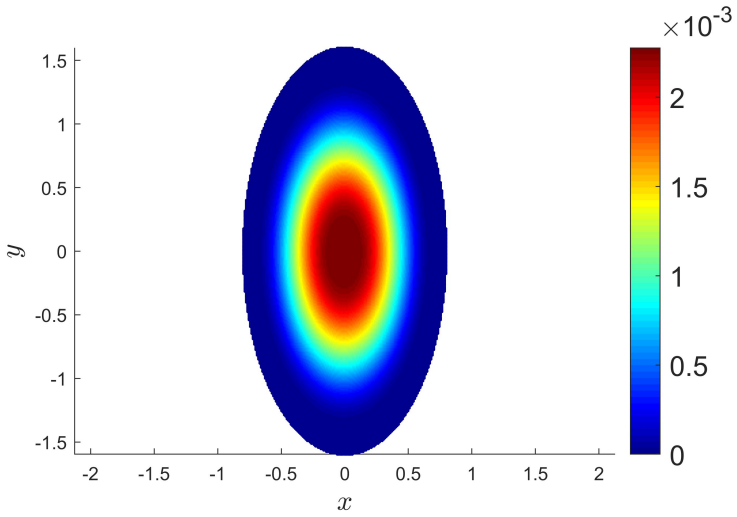


FIGURE 5.7: The contour plot of the initial assignment of the particle circulations for the elliptical vortex. For the grid resolution $\delta x = 1.1 \times 10^{-3}$, a total of $N(0) = 35348$ particles are used to resolve the distribution.

5.3.2 Diagnostic: effective aspect ratio

It order to gauge the correctness of our numerical approach, it is important to introduce a scalar that encapsulates the correct physical attributes of the problem. Following Dritschel (1989), it was observed that the elliptical vortex will undergo a process of *axisymmetrization* in which the highly elliptical core transfers to a more

stable circular configuration. This phenomenon is characterized by the effective aspect ratio λ_{eff}, which is defined by:

$$\lambda_{\text{eff}}^2 = \frac{J+R}{J-R}, \tag{5.3.8}$$

where $R^2 = D^2 + \lambda^2 J_{11}^2$, $D := J_{20} - J_{02}$ and $J := J_{20} + J_{02}$. Numerically, it can be shown that the initial value of λ_{eff} is indeed equal to the aspect ratio of the ellipse.

5.3.3 Discussion

Outside of the re-meshing step, the code uses the second order Adam-Bashforth scheme to advance the solution. The rationale for a multi-step scheme is to avoid performing the Biot-Savart evaluation multiple times to achieve the same second order accuracy; though at the expense of allocating memory to store the previous velocity data. After a re-meshing step, however, there are no previous velocity data available for the new set of particles. To circumvent this issue, the code uses a predictor-corrector scheme immediately after the re-meshing. A time step size of $\delta t = 4 \times 10^{-3}$ was used in all of the subsequent simulations and the M_4' interpolation scheme was employed to re-mesh the particles.

The time series of the evolution is presented in Figure (5.8). Initially, the strong vortical flow in the core causes a differential angular velocity amongst the vortex layers with the inner layer rotating at a faster rate. This differential speed results in the creation of two distinct vortex filaments that are become progressively thinner and interact strongly with the core (Figure (5.8a) - Figure (5.8d)). This stage of the evolution is encapsulated by a sharp reduction in the effective aspect ratio (Figure (5.9)). At a critical distance, the mutual interaction between the core and the filaments causes the core to eject vorticity, which also produces a plethora of small-scale filament structures(Figure (5.8e) - Figure (5.8h)).

At a later time, however, these small scale structures are damped by the surrounding flow. The result is an equilibrium state in which the vortex core experiences a periodic absorption and the production of vorticity from its neighbour. During the simulation, the effective aspect ratio was found to hover between 1.1 and 1.2 for $t > 12$ and no axisymmetrization was observed. In our computation, the results are closely matched with those of Koumoutsakos (1997) at least for $t < 4$. However, for $t > 4$, there appears to be an apparent shift in the effective aspect ratio. This discrepancy might be attributed to the way in which the particle is initialized; particularly in the choice of the function R_0, which was not specified explicitly in his work. Indeed, Koumoutsakos, through his systematic parametric studies, supports the observation that the final configuration of the vortex is a strong function of the initial vorticity profile. In particular, the two profiles tested in Koumoutsakos (1997), both of which utilised the spherical symmetric assumption for the initial vorticity distributions for some reason (i.e. $\omega = \omega(r)$), evolve to two complete different configurations. Although it is observed also that the long time configuration of both profiles seems to exhibit a moderate dependence on the simulation parameters such as the time-step size and the remeshing frequency (e.g. see the difference between case 1 and case 2 of the ω^{II} profile in the published work of Koumoutsakos (1997)). However, the simulated result closely follows the conclusion derived in the preliminary calculation in his PhD thesis (Koumoutsakos, 1993), in which Koumoutsakos concluded that *axisymmetrization* did not occur at the end of the long time simulation ($T > 24$). However, neither studies provided clarification on the choice of the

R_0 function. In fact, R_0 was taken to be a constant in the published work, which is puzzling given the fact that one has to *stretch* the distribution to fit the ellipse; a process that is extremely ambiguous. For the purpose of the current chapter, which serves to validate the developed method, such a pursuit to match with the literature to an excessive degree is not necessary. As we shall see in Chapter 6, there is a strong base to argue that our method is indeed correct despite the minor discrepancies.

The present simulation is conducted on the GPU with an efficient data tree that permits a higher number of particles to resolve the initial vorticity profile. The re-meshing process, in particular, is efficiently handled by using a custom-written atomic operation that allows a high level of parallelization. During the simulation, both creation and deletion of particles are allowed. New particles are constantly being introduced by the re-meshing while particles whose circulations are less than a threshold are deleted. There is a net increase of particle size $N(t)$ that experiences a sharp increase between the time interval $t = 0$ and $t = 5$. This rise might be attributed to the initial formation of the filaments. Subsequent evolution sees $N(t)$ to vary more slowly and appear to approach to a local balance (Figure (5.10)). At the end of the computation, $N(t)$ has increased by a factor of 2.24.

The code is extremely efficient, e.g. for 5300 time steps, it only takes approximately 10 minutes to finish. This is in stark contrast to other Eulerian methods where simulation times are on the order of days rather than minutes. In conclusion, we have validated the 2D vortex particle code.

5.4 A simple validation case for the PSE scheme in 1D

We present a simple validation case for the PSE scheme, in which the one-dimensional heat equation is solved using the PSE algorithm. The aim of this section is to verify our PSE code. To achieve such an aim, we present a rather contrived model problem, in which we seek the solution $\omega(x,t)$ to the following set of equations:

$$\frac{\partial \omega}{\partial t} = \nu \frac{\partial^2 \omega}{\partial x^2}, \quad (x,t) \in (-L/2, L/2) \times (0,\infty), \tag{5.4.1a}$$

$$\omega(\pm L/2, t) = 0, \quad t > 0, \tag{5.4.1b}$$

$$\omega(x,0) = f(x), \quad x \in (-L/2, L/2),. \tag{5.4.1c}$$

where L is a constant and ν is the thermal diffusivity. Physically, Eq.(5.4.1) may be interpreted as finding the transient heat distribution of a 1D rod subject to an initial heating given by the function f. The boundary condition is there for convenience of the finite-difference solver but one may also interpret the constraint as connecting the rod to a heat sink at both ends. It should be noted, however, that if the Dirichlet conditions were to be replaced by the Neumann conditions $\omega_x(\pm L/2, t) = 0$, this is analogous to the situation where the two ends are insulated (total heat is conserved in the rod). In such a case, the PSE scheme provides the direct solution to the Neumann problem instead. Since the role of the PSE is to redistribute the heat amongst the particles whilst conserves the total heat, this is equivalent to the Neumann condition that the heat flux shall varnish at the boundary. However, for the purpose of validating the new PSE code, it is not necessary to complicate the boundary discretization in the finite-difference solver.

5.4.1 A choice for g_ϵ in the PSE scheme

In one-dimension, one has to modify the heat-kernel $g_\epsilon(x) := \epsilon^{-1} g(x/\epsilon)$ appropriately so that the moment properties are satisfied (see Section 3.1.3). It is readily seen that a Gaussian kernel of the form can be used:

$$g(x) = C \exp\left(-\frac{x^2}{2}\right)., \tag{5.4.2}$$

where the constant C is determined from the moment condition:

$$\int_{-\infty}^{\infty} x^2 g(x)\, dx = 2. \tag{5.4.3}$$

Indeed, using that fact that $\int_{-\infty}^{\infty} \exp\left(-u^2\right) du = \sqrt{\pi}$, it is straightforward to show that the constant C is equal to $2/\sqrt{2\pi}$.

5.4.2 Solving Eq.(5.4.1) via finite difference

Eq.(5.4.1) can be solved in various ways. A straightforward approach is to implement a finite difference to approximate both the time and spatial derivatives. In this study, we opt for a first order time and a second order central difference in x. Specifically, if $\omega_i^n := \omega(i\delta x, n\delta t), i = 1, 2, \ldots, N$, then the discretized form of Eq.(5.4.1a) can be derived as follows:

$$\frac{\omega_i^n - \omega_i^{n-1}}{\delta t} = \nu \left(\frac{\omega_{i+1}^n - 2\omega_i^n + \omega_{i-1}^n}{\delta x^2}\right) \tag{5.4.4}$$

The boundary conditions Eq.(5.4.1b) are enforced by setting $\omega_0^n = \omega_{N+1}^n = 0$.

Equivalently, a sparse matrix can be assembled and the resulting equations of the system is

$$\begin{pmatrix} 1+2\lambda & -\lambda & 0 & \ldots & 0 & 0 \\ -\lambda & 1+2\lambda & -\lambda & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & -\lambda & 1+2\lambda \end{pmatrix} \begin{pmatrix} \omega_1^n \\ \omega_2^n \\ \vdots \\ \omega_N^n \end{pmatrix} = \begin{pmatrix} \omega_1^{n-1} \\ \omega_2^{n-1} \\ \vdots \\ \omega_N^{n-1} \end{pmatrix}, \tag{5.4.5}$$

where $\lambda := \nu\delta t/\delta x^2$.

5.4.3 Solving Eq.(5.4.1) via the PSE algorithm

As previously mentioned, the PSE replaces the Laplacian by an integral expression:

$$\frac{\partial^2 \omega}{\partial x^2}(x,t) \approx \epsilon^{-2} \int_{-\infty}^{\infty} (\omega(y,t) - \omega(x,t))\, g_\epsilon(y-x)\, dy \tag{5.4.6}$$

with g given by Eq.(5.4.2). A particle discretization sees that ω be approximated by a set of particles with the properties (α_i, x_i), where α_i denotes the particle strength given by $\alpha_i := \omega_i \delta x$ and x_i denotes the trajectory. Since only diffusion is resolved, the trajectory remains constant throughout.

Upon applying the PSE approximation, the particle strength satisfies the coupled ODE

$$\frac{d\alpha_i}{dt} = \frac{\nu\delta x}{\epsilon^2} \sum_{j=1}^{N} (a_j - a_i)\, g_\epsilon(x_j - x_i). \tag{5.4.7}$$

As discussed in Section 3.1.3, particles need to maintain an overlap to achieve the second order accuracy of the scheme, therefore an overlap ratio of 1.2 was used - so that the core parameter is $\epsilon = 1.2\delta x$. Suppose α_i^n denotes the i-th particle's strength at the time-step n, then a forward Euler scheme can be used to integrate Eq.(5.4.7), i.e.

$$\alpha_i^n = \alpha_i^{n-1} + \frac{\nu \delta x \delta t}{\epsilon^2} \sum_{j \in P_i} \left(\alpha_j^{n-1} - \alpha_i^{n-1} \right) g_\epsilon \left(x_j - x_i \right). \tag{5.4.8}$$

where the summation is applied only to those particles within a certain radius away from the i-th particle. For this purpose, a fast tree-code was used to determine the sum.

5.4.4 Discussion

The PSE algorithm does not satisfy the boundary condition exactly even though the initial particle strengths do. This can be corrected by using a sufficiently large L. In this particular simulation $L = 6$ seems to be sufficient. The initial condition is given by the function $f(x)$:

$$f(x) = \omega_{\max} \left(1 - 4 \left(\frac{x}{L} \right)^2 \right) \exp\left(- \left(x/\epsilon_0 \right)^2 \right), \tag{5.4.9}$$

where $\omega_{\max} = 0.6$ and $\epsilon_0 = 0.37796$ were used. There is no particular importance to those values. They were randomly chosen from a pseudo-random number generator.

In any case, the vorticity profile scaled by $\omega_{\max}$ is shown in Figure (5.11) after the 3000-th time step. The presented result demonstrated an excellent agreement between the two approaches. In the PSE algorithm, the total number of particles used to resolve the vorticity field was 500. It was observed that higher particle numbers did not improve the result. In fact, instability can arise if a large enough number of particles was used (leading to a blowup of the simulation). Indeed, Degond and Mas-Gallic (1989) presented a stability condition relating the diffusivity ν, the time discretization and the core parameters. In their analysis, the PSE scheme is stable if

$$\nu \delta t < \lambda \epsilon^2. \tag{5.4.10}$$

for some number λ, which depends on the type of time-marching scheme and the overlap ratio. Since ϵ is related to the grid spacing δx, which itself is derived from the particle number, a larger particle count will reduce the core parameter. Eventually, at some critical value $\epsilon = \sqrt{\nu \delta t / \lambda}$, the stability condition will cease to be valid. It should be noted that decreasing the overlapping ratio will destabilise the simulation to the same effect.

One of the biggest advantage of the PSE is its ability to generalize to arbitrary dimension. This only involves modifying the heat-kernel g so that it satisfies the appropriate moment conditions in that dimension. In 2D and 3D, g will always be taken to be a Gaussian with different constants. However, since each update for the particle strengths requires a summation from every particle, a direct approach is extremely expensive. Usually, only neighbouring particles contribute to the sum, therefore a fast neighbour search algorithm has to be used in conjunction to speed up the computation. In the current implementation, this is automatically done using the tree-structure outlined in Chapter 4. So it presents no particular difficulties in solving the coupled ODE in $\mathcal{O}(N)$ time. Admittedly, however, the PSE algorithm

on the GPU runs marginally slower than the finite-difference, this is to be expected since the particle discretization is too coarse so that the parallelization capability of the GPU is not being fully realized.

Although only the 1D case has been investigated and verified, the underlying code structures are exactly the same both in 2D and 3D with minor modifications to the heat-kernel. Therefore, one may assume that the new PSE code is correct and this concludes the aim of this section.

5.5 Performance measurement of the 2D FMM code

The most intensive part in the vortex particle method is the evaluation of the Biot Savart law. For N particles in the domain, the work scales as $\mathcal{O}\left(N^2\right)$, which quickly becomes untenable for large N if a brute-force approach is used. In Chapter 4, we have introduced the concept of the Fast-Multipole Method, which aims to reduce the scaling to $\mathcal{O}\left(N\right)$. In this section, we present the timing results between the different modes of evaluating the Biot Savart law.

5.5.1 Speed comparison between the direct approach and the FMM

The wall-clock time of the Biot Savart computation is compared between the direct approach and the FMM. The direct approach was implemented in the CPU using MATLAB's programming language. To speed up the direct calculation, a simple parallelization approach was used in which the evaluation points are split between 4 worker nodes in a local parallel cluster. Each worker node has its memory space, so data are copied to the workers' memory space before the computation can proceed. This parallelization approach requires minimum effort to code thanks to the MATLAB's parallel computing toolbox. The only real work here is to replace the for-loop with a parfor-loop and to modify the output data to a cell array for which the worker nodes can copy back the data to the host. For the CPU FMM implementation, however, no such parallelization method was used but vectorization was explored whenever was possible. The wall-clock times of the various evaluation modes are shown in Table 5.4 for the 4 particle clusters investigated. In each cluster, the particles are randomly generated in the unit square and all of the particles are of Rankine type. All times are measured in seconds.

N	CPU-FMM(s)	GPU-FMM(s)	CPU-Direct(s)	(CPU-FMM/GPU-FMM)
2×10^4	5.62	0.14	2.17	40.14
5×10^4	5.87	0.18	15.23	32.61
1×10^5	28.51	0.24	73.57	118.79
5×10^5	295.16	0.54	2361.85	546.59

TABLE 5.4: Wall-clock time for the different modes of evaluation of the Biot-Savart.

The GPU-FMM code demonstrates an excellent speed up compared to the single threaded implementation of the CPU-FMM and the parallelized CPU-Direct; reaching an impressive 546 times the speed-up compared to CPU-FMM. Perhaps what is surprising is that the CPU-FMM runs marginally slower than CPU-Direct for smaller particle clusters. Using the MATLAB's built-in profiling tool, one is not hard to observe that, at those sizes, the CPU-FMM spent almost three-quarter of its time creating the FMM tree, which is not surprising given that the original tree-construction

algorithm (Algorithm 4.2) has proven to be inefficient. This is compounded by the fact that the CPU-FMM uses a fully adaptive tree; meaning for each increase of refinement level in the tree there is an exponential increase of the children boxes that need to be traversed and sorted. In all of cases considered, the GPU-FMM code is clearly superior.

5.5.2 GPU FMM profiling

On a close inspection of the GPU-FMM code, one may identify how much time the FMM code spent in each of the evaluation phases, namely the data structure, the initial expansion (P2M), the upward pass (M2M), the downward pass (M2L + L2L) and the final evaluation (Section 4.1). Table 5.5 shows the percentage of the time in each phases. As to be expected, the multipole to local conversion step (M2L) is the most expensive step as it dominates the majority of the time. Indeed, this observation is consistent with the reported results of Gumerov and Duraiswami (2008) and Yokota et al. (2009). The reason for this is that each box in the tree needs to traverse the *List 2* partition, which contains the most elements, and convert a multipole expansion in the list to a local expansion. Consequently, a large number of fetch requests was made which drastically increases the latency of the execution as a large volume of data needs to be exchanged back and forth between the L2 cache and the main memory. As discussed before in Section 4.2, accessing the main-memory is typically the most expensive access in the GPU's memory hierarchy. This issue becomes even more pronounced in 3D where the List 2 partition contains, on average, 189 boxes. Therefore, an obvious direction for future improvement of the code is to optimize the memory transfer.

Data tree	P2M	M2M	M2L	L2L	final-evaluation
14.8%	3.3%	2.7%	63.7%	3.3%	12.1%

TABLE 5.5: Percentage of the total time the GPU-FMM code spent in each of the evaluation phases

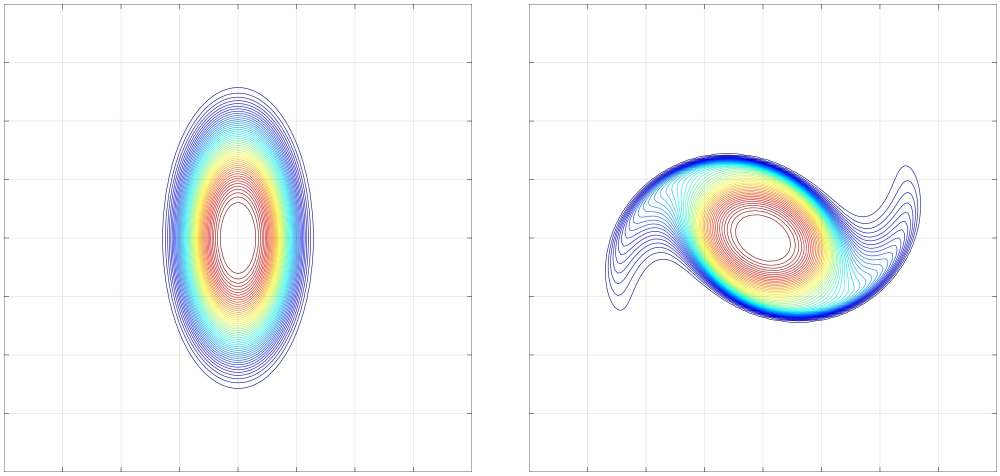
5.6 Conclusion

Several aspects of the vortex particles methods have been validated against simple flow problems. The new three-dimensional FMM code was used to simulate the evolution of the inviscid vortex ring. The results showed that the code managed to capture the self-induced velocity with a relative error of 8×10^{-3} compared to the theoretical value. For longer time, however, it was observed that the core undergoes significant expansion which results in a clear departure from the theoretical prediction.

The two-dimensional code was validated by simulating an elliptical vortex patch. The new results agree quite well with the existing literature values. However, some discrepancies are present especially in the computation of the effective aspect ratio, which we attributed the reason as due to the particle initialization process.

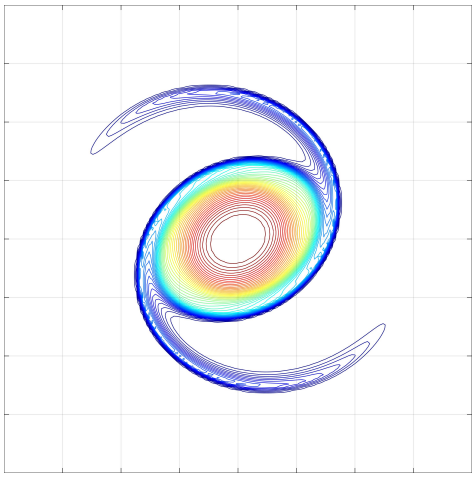
In addition, a simple 1D case was presented for solving the heat equation via the particle-strength exchange scheme (PSE) and finite difference approach. The two results agree extremely well. It was noted that the PSE requires a fast neighbour search algorithm in order to avoid the $\mathcal{O}\left(N^2\right)$ cost. This approach is efficiently handled by using the tree-structure outlined in Chapter 4.

Finally, the performance of the GPU-accelerated FMM code has been analysed and was found that, compared to a single-threaded CPU implementation of the FMM, the GPU manages to achieve an impressive 546 times the speed up. This capability allows high-fidelity simulations to be performed in both 2D and 3D flow problems in realistic times.

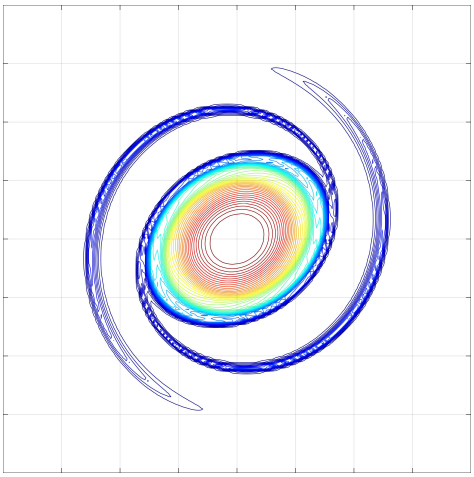


(A) $T = 0$

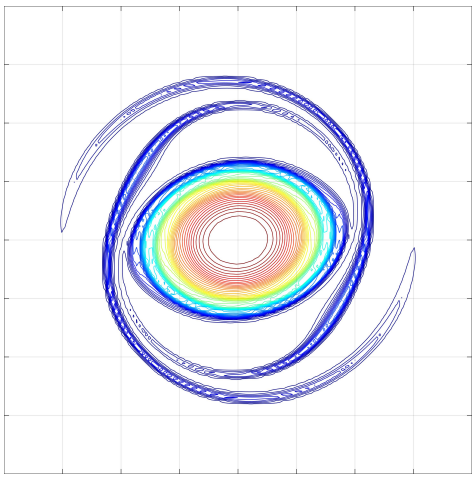
(B) $T = 0.6$



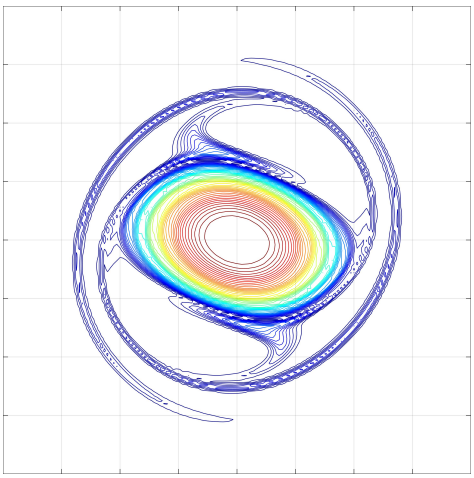
(C) $T = 1.2$



(D) $T = 2.0$

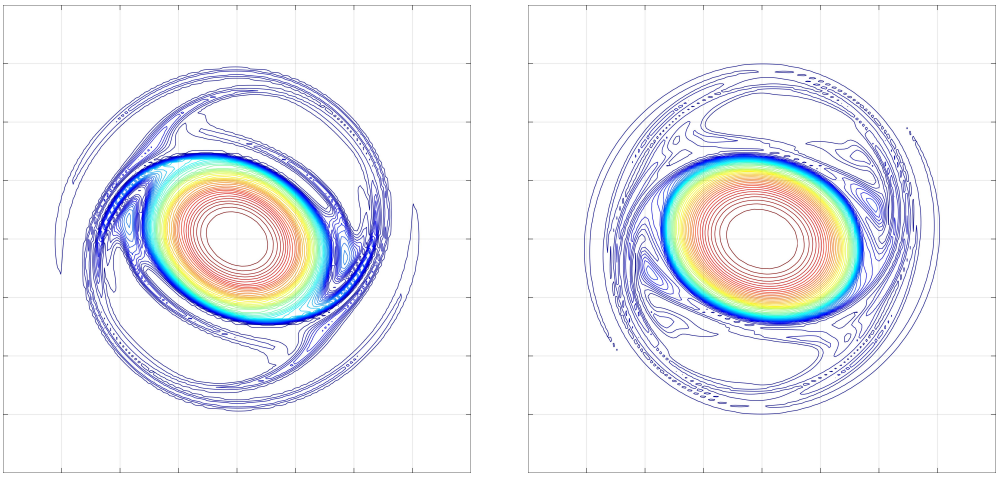


(E) $T = 3.0$

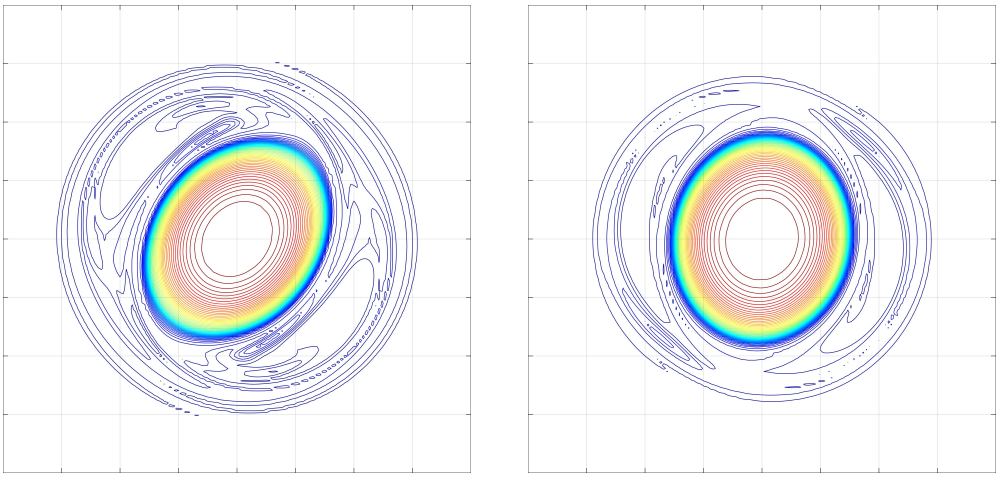


(F) $T = 4.0$

FIGURE 5.8: Contour plots of the vorticity field at different time stamps of the elliptical vortex patch. The simulation demonstrates a complex filamentation process by which the ellipticity of the initial vortex is gradually transitioning to that of a circular configuration. No axisymmetrization is observed at the end of the run.



(G) $T = 5.0$ (H) $T = 10.8$



(I) $T = 16.8$ (J) $T = 21.2$

FIGURE 5.8: continued

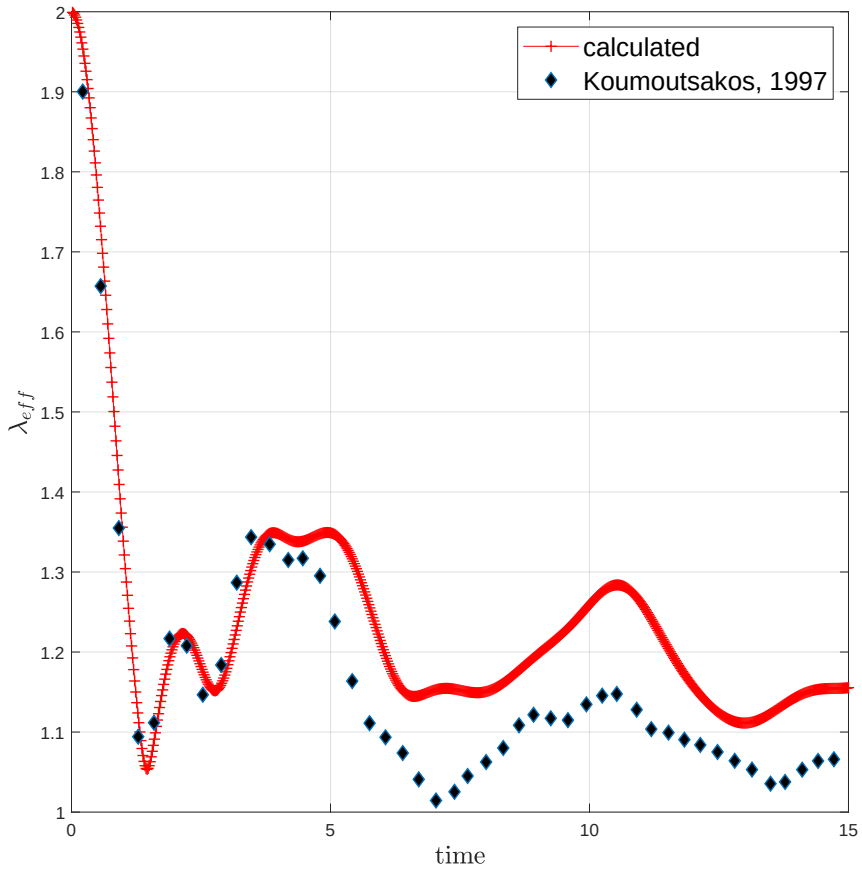


FIGURE 5.9: Computed λ_{eff} versus those from Koumoutsakos (1997). The results agree very well for $t < 4$. But for $t > 4$, our computation shows that λ_{eff} does not converge to the circular configuration. Although the trend of the two results generally matches quite well, it is anticipated that this discrepancy could be due to the ways the particles are initialized.

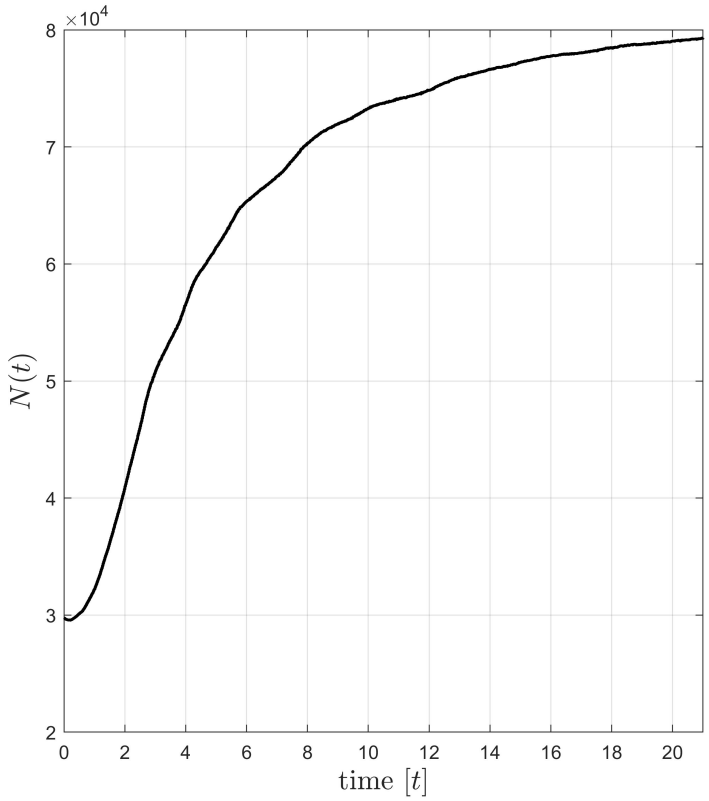
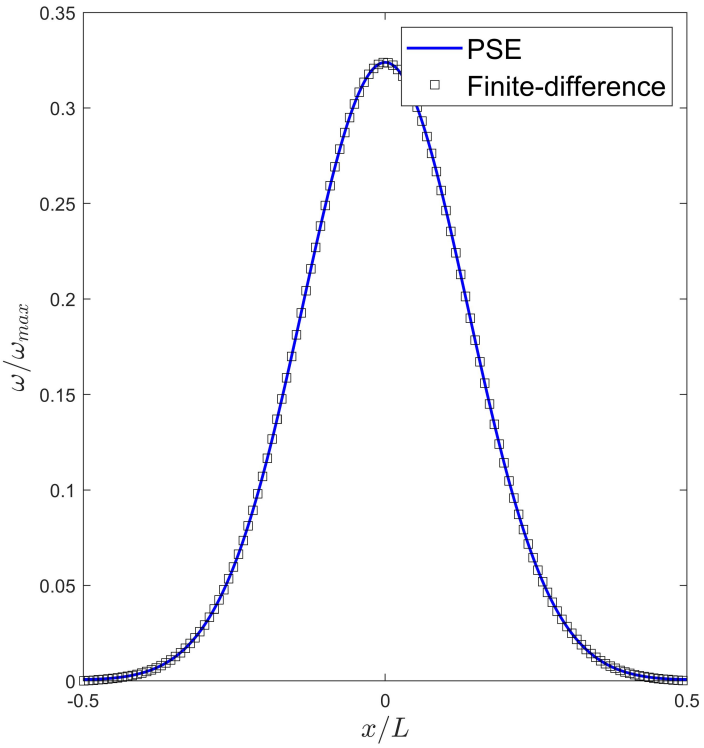


FIGURE 5.10: Variation of particle size $N(t)$ as a function of time t. A sharp rise of particles numbers was observed during the filamentation process. At the end of the computation, the particle size has increased by a factor of 2.24.



.

FIGURE 5.11: A comparison of the vorticity profile between the PSE and finite-difference. The presented results correspond to the following simulation parameters: $T = 30$, $\nu = 10^{-2}$, $\delta x = 1.2 \times 10^{-2}$ and $\delta t = 10^{-2}$

Chapter 6

Two dimensional flows at moderate Reynolds number

For moderate Reynolds number, flow separation is a common phenomenon in turbine applications. In Section 3.2.1, we outlined a potential approach to solve the inviscid Euler's equations analytically. This approach, combined with the Kutta conditions, assumes the flow remains attached on the body. Physically, however, turbines with relative large TSR may still undergo severe separation in instances where they interact with regions of concentrated vorticity. Furthermore, since separation is predominately a viscous effect, an accurate simulation thus requires one to focus on a viscous particle methods where the correct viscous boundary effects are accounted for. In three-dimensions, this places a large strain on computing resources, therefore it is not implemented in this work. However, in two-dimensions, together with the GPU accelerated FMM code in Chapter 4, high fidelity approaches are afforded. With this in mind, this section highlights the new modelling approach and presents the results for a variety of flow scenarios.

6.1 A fractional time-stepping algorithm

To correctly account for the kinematic boundary conditions, namely the 'no-slip', it is important to derive an equivalent boundary condition for the vorticity equations as it constitutes the main source of vorticity generation in the viscous flow domain. The incorporation of the vorticity generation mechanism has to be dealt with in two stages, which are: a) inertial and particle diffusion, and b) wall diffusion. Although these two processes are linked intrinsically and the particle method is capable on resolving inertial and particle diffusion, but the main difficulty arises when trying to satisfy the boundary conditions simultaneously. To alleviate these issues, a fractional time-stepping algorithm is adopted which serializes the two processes.

6.1.1 Operator splitting

During a particular time-step, say at the k-th time step, the algorithm seeks to determine the vorticity field for the $(k+1)$-th time step, which procedurally solves the advection step and the no-through boundary conditions. The latter step is particularly important in that the vorticity generation in the flow depends entirely on the correct vorticity flux which, in general, is expressed as a function of the vortex sheet introduced to enforce the no-slip.

To this effect, the splitting scheme treats these two processes in a serial manner. First, Eq.(6.1.1) are solved subjecting to the no-through conditions:

$$\frac{\partial \omega}{\partial t} + \vec{u} \cdot \nabla \omega = 0, \quad \vec{x} \in \mathbb{R}^2 \backslash D,\ t > 0, \tag{6.1.1a}$$

$$\omega\left(\vec{x}, 0\right) = g\left(\vec{x}\right), \quad \vec{x} \in \mathbb{R}^2 \backslash D, \tag{6.1.1b}$$

$$\vec{u}\left(\vec{x}, t\right) \cdot \vec{n} = \vec{u}_b \cdot \vec{n}, \quad \vec{x} \in \partial D,\ t > 0, \tag{6.1.1c}$$

where g denotes the initial condition which represents the vorticity field of previous time-step, $\vec{u}_b$ denotes the body velocity, D and ∂D denotes the union of the rigid bodies and their boundaries, respectively. It should be noted that Eq.(6.1.1c) represents the no-through condition. Consequently, the no-slip is not enforced so that there is no new vorticity generated at this stage. Also, diffusion among particles is carried out as a sub-step, which corresponds to solving the following set of equations:

$$\frac{\partial \omega}{\partial t} = \nu \nabla^2 \omega, \quad \vec{x} \in \mathbb{R}^2 \backslash D,\ t > 0, \tag{6.1.2a}$$

$$\omega\left(\vec{x}, 0\right) = g\left(\vec{x}\right), \quad \vec{x} \in \mathbb{R}^2 \backslash D, \tag{6.1.2b}$$

which can be efficiently solved in $\mathcal{O}\left(N\right)$ time (see Section 5.4) using the developed PSE scheme in Section 3.1.3. The effective vorticity field at the end of this sub-step is obtained as a superposition of the solutions of Eq.(6.1.1) and Eq.(6.1.2). With a proper velocity field that respects the no-through conditions in Eq.(6.1.1c), one should expect that the solutions at the end of step one should not introduce vorticity inside D. However, in practice, such idealization is difficult to realise due to various factors such as the accumulation of numerical errors in the simulations. To avoid this complication, the method employs a particle deletion algorithm which seeks to identify and remove particles that settle too close to ∂D whilst still respecting the conservation nature of the vorticity field in 2D. The exact implementation is discussed in Section 6.1.4.

In any case, at the end of the first step the original vorticity field is modified. This modified field then results in a spurious slip velocity at the boundary, which has to be nullified in order to satisfy the no-slip. The nullification is done by introducing a vortex sheet whose strengths are solved by the panel method discussed in Section 3.2. Subsequently, the vorticity flux $(\partial\omega/\partial n)$ is computed acting as the boundary conditions for the following system of equations:

$$\frac{\partial \omega}{\partial t} = \nu \nabla^2 \omega, \quad \vec{x} \in \mathbb{R}^2 \backslash D,\ t > 0, \tag{6.1.3a}$$

$$\omega\left(\vec{x}, 0\right) = 0, \quad \vec{x} \in \mathbb{R}^2 \backslash D, \tag{6.1.3b}$$

$$\nu \frac{\partial \omega}{\partial n} = F\left(\vec{x}, t\right), \quad \vec{x} \in \partial D,\ t > 0, \tag{6.1.3c}$$

where F denotes the vorticity flux function. In such calculation, the aim is then to determine a secondary vorticity field satisfying Eq.(6.1.3) to incorporate the effect of vorticity generation in the presence of rigid bodies in the flow domain. Once a solution is found in step two, the effective vorticity field is advanced to the next time step by superimposing the solution of Eq.(6.1.3) with those of Eq.(6.1.1) and Eq.(6.1.2).

Eq.(6.1.3) admits an exact solution of the form (see Friedman (2008)):

$$\omega\left(\vec{x}, t\right) = \int_0^t \int_{\partial D} G\left(\vec{x}, \vec{y}, t - \tau\right) \mu\left(\vec{y}, \tau\right)\ d\tau\ ds\left(\vec{y}\right), \tag{6.1.4}$$

where G is the *heat-kernel* given by equation Eq.(6.1.5):

$$G\left(\vec{x},\vec{y},t\right) := \frac{1}{4\pi\nu t}\exp\left(-\frac{\|\vec{x}-\vec{y}\|^2}{4\nu t}\right), \tag{6.1.5}$$

here μ is called the *surface density* whose value is obtained from the Neumann condition in Eq.(6.1.3c). Indeed, by applying the *BEM* approach (see Section 3.2.1), it is readily verified that μ satifies a Fredholm integral equation of the second kind:

$$-\frac{1}{2}\mu\left(\vec{x},t\right)+\int_0^t\int_{\partial D}\frac{\partial G}{\partial n}\left(\vec{x},\vec{y},t-\tau\right)\mu\left(\vec{y},\tau\right)\,ds\left(\vec{y}\right)\,d\tau = F\left(\vec{x},t\right). \tag{6.1.6}$$

For a relatively small time-step size, one may assume that F and μ do not change much during a time-step, so that F and μ can be treated as time-independent and whose values may be parametrized by the arc-length s only (i.e. $F(\vec{x}(s),t)=F(\vec{x}(s))=F(s)$, $\mu(\vec{x}(s),t)=\mu(\vec{x}(s))=\mu(s)$). Under this assumption, Koumoutsakos (1993) derived the dominant contribution in the Taylor expansion for the surface density relating the flux F and the curvature of the geometry:

$$\mu\left(s\right) = -2F\left(s\right)\left(1-\kappa\left(s\right)\sqrt{\pi\nu\delta t}\right)^{-1}. \tag{6.1.7}$$

here, δt denotes the time step size, κ is the curvature and s is the arc-length parametrization.

Assuming that the body is approximated by M straight panels and the surface density is independent of time, one may decompose the boundary as an union of linear panels, i.e. $\partial D = \bigcup_{j=0}^{M-1}\partial D_j$, where each linear ∂D_j is characterised by the centroid $\vec{x}_0=(x_0,y_0)$, the orientation angle θ and the length d. Denote ξ and η the projected distance along the tangent and normal direction of the panel, i.e.

$$\begin{aligned}\xi &= +(x-x_0)\cos\theta+(y-y_0)\sin\theta,\\ \eta &= -(x-x_0)\sin\theta+(y-y_0)\cos\theta.\end{aligned}$$

Further, let $\psi_j(\vec{x},t)$ be defined as follows:

$$\psi_j\left(\vec{x},t\right) := \int_{\partial D_j} G\left(\vec{x},\vec{y},t\right)\mu\left(\vec{y}\right)ds(\vec{y}). \tag{6.1.9}$$

If $\vec{y}\in\partial D_j$, then for some scalar s, $\vec{y}(s)=\vec{x}_0+(s-s_0)\left(\cos\theta,\sin\theta\right)^T$, where s_0 is the arc-length at the centroid location. Whence,

$$\|\vec{x}-\vec{y}\left(s\right)\|^2 = (s-s_0-\xi)^2+\eta^2. \tag{6.1.10}$$

Replacing G by its definition and substituting Eq.(6.1.10) to Eq.(6.1.9), one has that:

$$\begin{aligned}\psi_j\left(\vec{x},t\right) &= \int_{\partial D_j} G\left(\vec{x},\vec{y},t\right)\mu\left(\vec{y}\right)ds\left(\vec{y}\right)\\ &= \int_{s_0-d/2}^{s_0+d/2}\frac{1}{4\pi\nu t}\exp\left(-\frac{\|\vec{x}-\vec{y}\left(s\right)\|^2}{4\nu t}\right)\mu\left(s\right)ds,\\ &= \frac{1}{4\pi\nu t}\int_{s_0-d/2}^{s_0+d/2}\exp\left(-\frac{(s-s_0-\xi)^2+\eta^2}{4\nu t}\right)\mu\left(s\right)ds.\end{aligned}$$

Furthermore, one might impose the piece-wise constant assumption on μ so that μ over ∂D_j may be replaced by an averaged value so that Eq.(6.1.9) becomes:

$$\psi_j\left(\vec{x}\left(\xi,\eta\right),t\right)=\frac{\bar{\mu}_j}{4\pi\nu t}\exp\left(-\frac{\eta^2}{4\nu t}\right)\int_{s_0-d/2}^{s_0+d/2}\exp\left(-\frac{\left(s-s_0-\xi\right)^2}{4\nu t}\right)ds. \tag{6.1.11}$$

where $\bar{\mu}_j$ denotes the averaged value of μ of the panel. Using the substitution $v=\left(s-s_0-\xi\right)/\sqrt{4\nu t}$ in the above integral, Eq.(6.1.11) becomes

$$\psi_j\left(\vec{x}\left(\xi,\eta\right),t\right)=\frac{1}{2}\frac{\bar{\mu}_j}{\sqrt{4\pi\nu t}}\exp\left(-\frac{\eta^2}{4\nu t}\right)\left(\operatorname{erf}\left(\frac{\xi+d/2}{\sqrt{4\nu t}}\right)-\operatorname{erf}\left(\frac{\xi-d/2}{\sqrt{4\nu t}}\right)\right), \tag{6.1.12}$$

so that Eq.(6.1.4) is approximately:

$$\omega\left(\vec{x},t\right)\approx\int_0^t\sum_{j=0}^{M-1}\psi_j\left(\vec{x},\tau\right)d\tau. \tag{6.1.13}$$

There are various ways to integrate the time integral in Eq.(6.1.13). For example, Koumoutsakos (1993) used the mid-point rule over a small time-interval $t=\delta t$, e.g.

$$\omega_{\text{mid}}\left(\vec{x},\delta t\right)\approx\delta t\sum_{j=0}^{M-1}\psi_j\left(\vec{x},\delta t/2\right) \tag{6.1.14}$$

In this study, however, a P-point Gauss-Legendre quadrature rule is adopted in the same small time-interval (Hildebrand, 2013), i.e.

$$\omega_{\text{GL}}\left(\vec{x},\delta t\right)=\frac{\delta t}{2}\sum_{k=0}^{P}\sum_{j=0}^{M-1}w_k\psi_j\left(\vec{x},\frac{\delta t}{2}\left(1+r_k\right)\right). \tag{6.1.15}$$

where w_k and r_k are the quadrature weights and nodes, respectively.

To validate the implementation of Eq.(6.1.15), Eq.(6.1.3) was solved using both the finite-difference method and Eq.(6.1.15) in the square domain $D=[-L/2,L/2]\times[0,L]$. In order to simplify the problem, a mixed boundary condition was used for the finite difference approach, with the Neumann condition $\nu\partial\omega/\partial y|_{y=0}=F$ applied only at the bottom edge of the domain and the Dirichlet condition $\omega=0$ for the rest (see Figure (6.1)).

A comparison of the results is shown in Figure (6.2) for the function $F(x)=\lambda\left(1-4\left(x/L\right)^2\right)\exp\left(-\left(x/\epsilon_0\right)^2\right)$. In general, the two results agree reasonably well; albeit there appears to be some discrepancies close to the bottom wall, which is attributable to the use of the piece-wise constant assumption of the surface density μ.

6.1.2 Modification of particle strength due to the boundary flux

Anticipating the descretization of the body geometry into a set of panels, the flux in each panel contributes a change of circulation $\Delta\Gamma$ to the neighbouring particles with coordinates (x,y) and volume h^2 (see Figure (6.3b)). If one assumes that the vorticity is constant in the volume, one recovers the Koumoutsakos scheme (scheme K), in which the change of circulation at the end of the interval $[0,\delta t]$ due to the

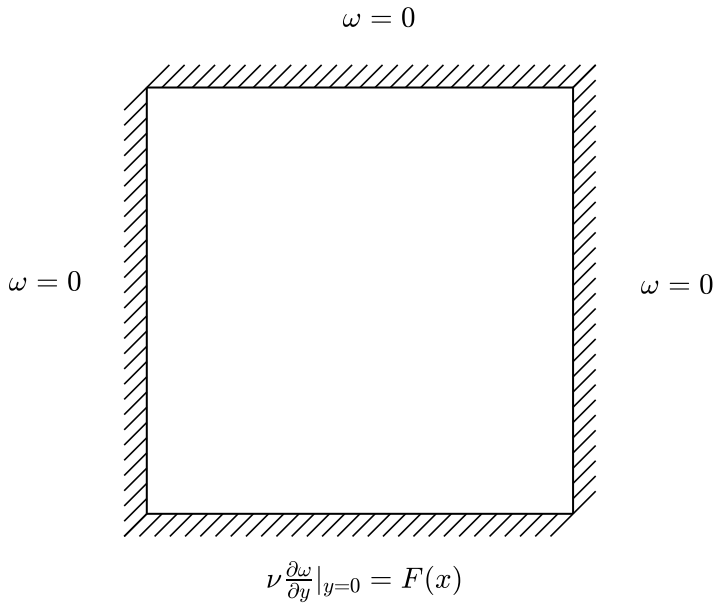


FIGURE 6.1: A square domain for Eq.(6.1.3). For simplification, a mixed boundary conditions were used. Only the bottom edge is subjected to the Neumann condition.

boundary flux is approximated by Eq.(6.1.14), i.e.

$$\Delta\Gamma\left(x, y, \delta t\right) = h^2 \omega_{\text{mid}}\left(x, y, \delta t\right). \tag{6.1.16}$$

However, it was pointed out by Ploumhans and Winckelmans (2000) that such a scheme may not be strictly conservative. Since if particles settle near the surface, some parts of their volume may cross the boundary surface in an arbitrary way. To alleviate such a limitation, it is necessary to allow the vorticity to vary in the volume, so that the $\Delta\Gamma$ is expressed as an integral of the form:

$$\Delta\Gamma\left(x, y, t\right) = \int_{x-h/2}^{x+h/2} \int_{h'}^{y+h/2} \omega\left(x', y', t\right)\, dx'\, dy', \tag{6.1.17}$$

where ω is given by Eq.(6.1.13) and $h' = 0$ if the volume of the particle intercepts the boundary, otherwise $h' = y - h/2$. To evaluate the integral in Eq.(6.1.17), a local panel transformation is performed so that $x' \to x_0 + \xi' \cos\theta - \eta' \sin\theta$ and $y \to y_0 + \xi' \sin\theta + \eta' \cos\theta$. The time rate of $\Delta\Gamma$ concerned by the j-th panel may be then expressed as follows:

$$\frac{\partial \Delta\Gamma}{\partial t}\left(\vec{x}\left(\xi, \eta\right), t\right) = \int_{\xi-h/2}^{\xi+h/2} \int_{h'}^{\eta+h/2} \psi_j\left(\vec{x}\left(\xi', \eta'\right), t\right) d\xi' d\eta', \tag{6.1.18}$$

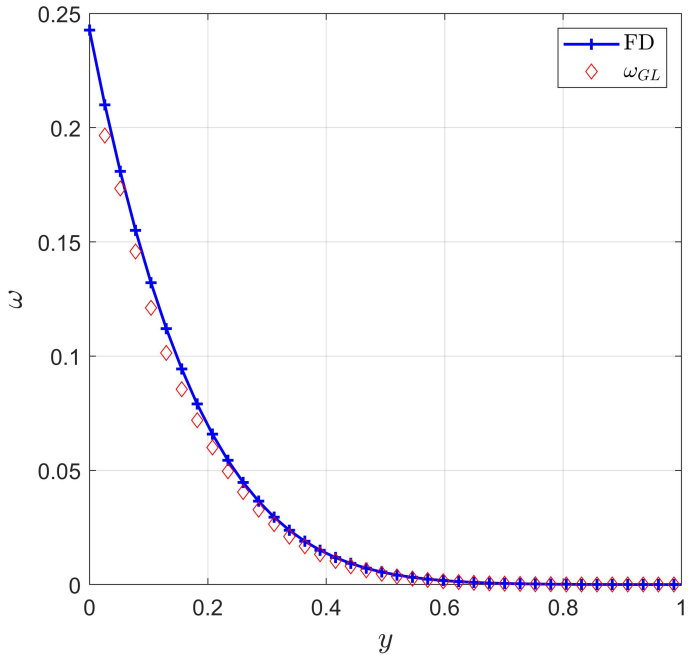


FIGURE 6.2: A comparison between the finite-difference method and Eq.(6.1.15). Figure depicts the solutions at $x = 0$ after $t = 0.064$.

where we made use of the fact that $dx'dy' = d\xi' d\eta'$ such that the Jacobian of the transformation is unity. Substituting the expression for ψ_j, so Eq.(6.1.18) becomes:

$$\frac{\partial \Delta\Gamma}{\partial t} = \frac{1}{2}\frac{\bar{\mu}_j}{\sqrt{4\pi\nu t}} \int_{h'}^{\eta+h/2} \exp\left(-\frac{\eta'^2}{4\nu t}\right) d\eta' \times \int_{\xi-h/2}^{\xi+h/2} \left(\operatorname{erf}\left(\frac{\xi'+d/2}{\sqrt{4\nu t}}\right) - \operatorname{erf}\left(\frac{\xi'-d/2}{\sqrt{4\nu t}}\right)\right) d\xi'. \quad (6.1.19)$$

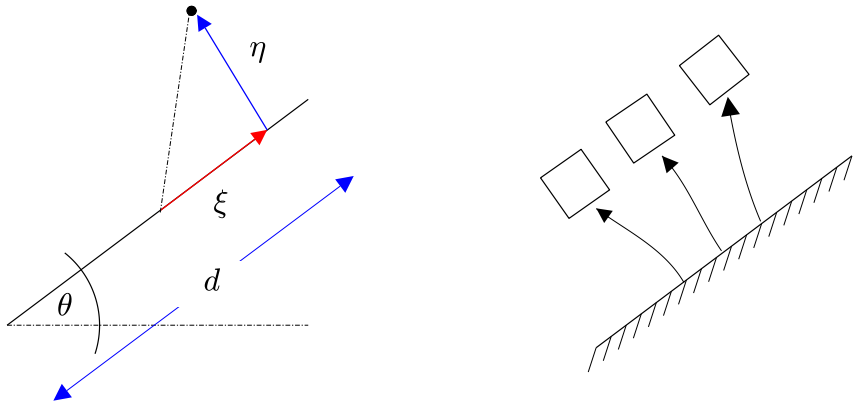
Straight forward integration by parts on the ξ' integral gives the following expression:

$$\frac{\partial \Delta\Gamma}{\partial t} = \frac{\bar{\mu}_j}{4}\sqrt{4\nu t}\left[\operatorname{erf}(u)\right]_{h'/\sqrt{4\nu t}}^{(\eta+h/2)/\sqrt{4\nu t}} \times \left(\left[\operatorname{ierf}(u)\right]_{(\xi-h/2+d/2)/\sqrt{4\nu t}}^{(\xi+h/2+d/2)/\sqrt{4\nu t}} - \left[\operatorname{ierf}(u)\right]_{(\xi-h/2-d/2)/\sqrt{4\nu t}}^{(\xi+h/2-d/2)/\sqrt{4\nu t}}\right). \quad (6.1.20)$$

where $\operatorname{ierf}(x) := x\operatorname{erf}(x) + \exp\left(-x^2\right)/\sqrt{\pi}$. Finally, $\Delta\Gamma$ is obtained by integrating the above expression using the same P-point Gaussian-Legendre quadrature rule on the time-variable, i.e.

$$\Delta\Gamma\left(\vec{x}\left(\xi,\eta\right),\delta t\right) \approx \frac{\delta t}{2}\sum_{k=0}^{P} w_k \frac{\partial \Delta\Gamma}{\partial t}\left(\vec{x}\left(\xi,\eta\right), \frac{\delta t}{2}\left(1+r_k\right)\right). \quad (6.1.21)$$

Note here that a similar expression was also proposed by Ploumhans and Winckelmans (2000), therefore Eq.(6.1.21) will be referred to as scheme W. An example



(A) (B)

FIGURE 6.3: Schematic diagram for the panel characterization (A). Once the flux is computed, the flux is parcelled to the neighbouring particles near the surface to satisfy the no-slip (B).

calculation is presented in Figure (6.4) in which the circulation is sampled at different heights for a panel fixed at $y = 0$. It should be noted for $y < h/2$, some parts of the particle's volume intercept the boundary which should be excluded from the integration. Indeed, this results in a noticeable discontinuity in the vertical derivative. It is for this reason that scheme W is more conservative than scheme K.

6.1.3 Enforcement of the no-slip condition

Once Eq.(6.1.1) and Eq.(6.1.2) have been solved using the viscous particle method, the no-slip, in general, is no longer satisfied. The aim of the second step in the fractional time-stepping algorithm is to introduce a vortex sheet around the surface to nullify this spurious slip velocity. However, care should be exercised to ensure that the no-through condition, i.e. Eq.(6.1.1c) is also properly respected. One approach is to extend the vorticity field inside the geometry, so that one treats the body motion as being induced by an interior vorticity field. Under such generalization, Spalart and Leonard (1981) showed that if the no-slip is satisfied then the no-through is automatically satisfied as well.

Thus, it is necessary to identify the velocity components that constitutes the effective field. If we denote $\vec{u}_{\text{eff}}$ as the effective field valid everywhere in the flow domain, then one can write it as:

$$\vec{u}_{\text{eff}} = \vec{u}_\gamma + \vec{u}_f + \vec{u}_\infty + \vec{u}_\Omega, \tag{6.1.22}$$

where $\vec{u}_\gamma, \vec{u}_f, \vec{u}_\infty$ and $\vec{u}_\Omega$ denote the velocity fields induced by the vortex sheet, the exterior vorticity, the free-stream and the interior vorticity, respectively. Note, however, only $\vec{u}_\gamma$ remains unknown and $\vec{u}_\Omega$ is evaluated according to the contour dynamic formula of Zabusky et al. (1979), see Appendix C for a derivation of the formula.

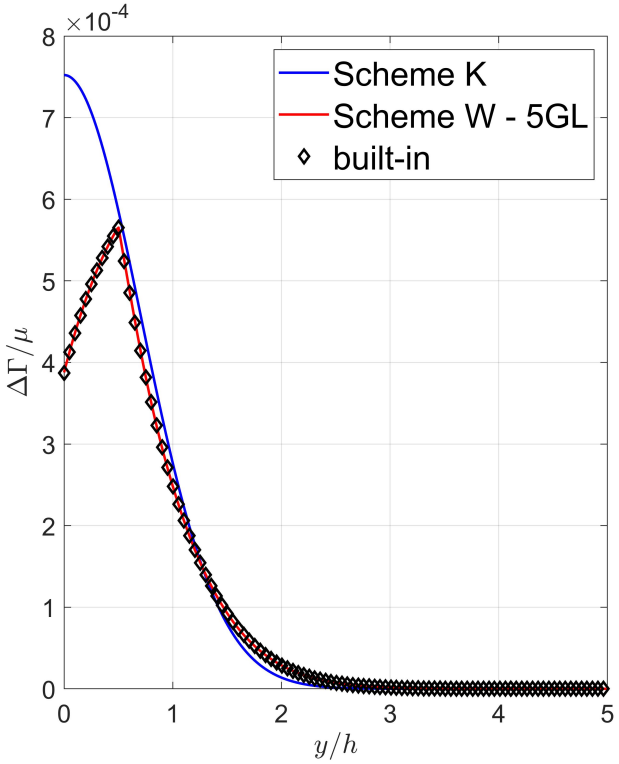


FIGURE 6.4: Comparison between scheme K and scheme W. The plot shows the variation of the change of circulation as a function of the vertical distance directly above the panel at a fixed x-station. The red curve corresponds to the 5-point Gauss-Legendre quadrature rule and it is to be compared to the MATLAB's built-in integration routine (diamond marker).

Indeed, the no-slip implies the effective velocity must satisfy Eq.(6.1.23)

$$(\vec{u}_{\text{eff}} - u_B) \cdot \vec{t} = 0, \tag{6.1.23}$$

which we might solve it via the BEM.

In light of the Kelvin's theorem, there is a related constraint that must be imposed on top of Eq.(6.1.23). If the body is rotating with angular velocity Ω, then the vortex sheet with strength γ must also satisfy the following equation:

$$\oint_s \gamma(s)\, ds = -2A\delta t \frac{d\Omega}{dt}, \tag{6.1.24}$$

where A is the area of the body.

Effectively, Eq.(6.1.23) together with Eq.(6.1.24) and the use of the boundary element method, constitute an over-determined system of linear equations whose solution may not even exist. Currently, the matrix equation is solved in an approximate sense.

Suppose that Eq.(6.1.23) and Eq.(6.1.24) is expressed in the matrix form:

$$Ax = b, \tag{6.1.25}$$

we are interested in x that minimizes the norm $\|Ax - b\|$. The underlying problem thus transforms to the following minimization problem:

$$\min_x \{\|Ax - b\|^2\} \tag{6.1.26}$$

subject to the set of linear constraints of the form:

$$\begin{aligned} c_1^T x - r_1 &= 0, \\ c_2^T x - r_2 &= 0, \\ &\vdots \\ c_k^T x - r_k &= 0. \end{aligned}$$

It should be noted that the constraints are the results of Kelvin's condition (Eq.(6.1.24)) applied separately for the k bodies in the simulation.

By introducing the Lagrangian $\mathcal{L}$ and the Lagrangian multipliers $\lambda_i, i \in \{1, 2, \ldots, k\}$

$$\mathcal{L}(x, \lambda) := \|Ax - b\|^2 - \sum_{n=1}^{k} \lambda_n \left(c_n^T x - r_n \right) \tag{6.1.28}$$

Eq.(6.1.26) is solved by requiring that $\partial \mathcal{L} / \partial x = 0$. Noting that

$$\nabla \|Ax - b\|^2 = 2A^T Ax - 2A^T b,$$

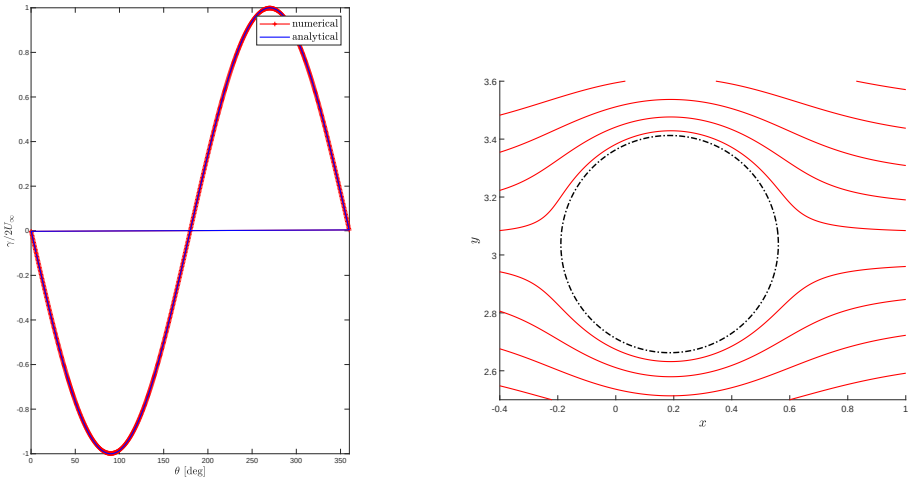
then it is possible to show that the new matrix equations can be recast in the following way:

$$\begin{pmatrix} 2A^T A & c_1 & c_2 & \ldots & c_k \\ c_1^T & 0 & 0 & \ldots & 0 \\ c_2^T & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_k^T & 0 & 0 & \ldots & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_k \end{pmatrix} = \begin{pmatrix} 2A^T b \\ r_1 \\ r_2 \\ \vdots \\ r_k \end{pmatrix} \tag{6.1.29}$$

which can be solved by a dense matrix solver.

To validate the minimization procedure, we apply the approach to a stationary cylinder immersed in an infinite stretch of incompressible fluid with an ambient velocity field $(U_\infty, 0)$. The cylinder is impulsively started, so immediately after $t > 0$, the initial flow is inviscid and there is a velocity difference on the boundary. It is well known that this difference corresponds to a vortex sheet with strength given by $\gamma(\theta) = -2U_\infty \sin\theta$, where θ is the azimuthal angle of the cylinder (Batchelor, 1967). For this flow, Eq.(6.1.29) is used to determine the correct vortex strength. Indeed, Figure (6.5a) illustrates the numerical result for the normalized vortex strength and is compared to the analytical solution. There is almost no discernible difference between the two calculations. For this particular strength, Figure (6.5b) shows the streamlines seeded at a uniform interval. The developed flow pattern shows that the no-through condition is indeed justified.

Once the vortex sheet strength is determined (i.e. $\gamma(s)$), it must flux to the flow domain in order to nullify the slip velocity. To achieve this, there must be particles surrounding the bodies to receive the flux. Initially, several layers of zero-strength particles are initialized (see Section 6.1.4) with the number of layers as a function of the diffusion scale. Subsequently, the flux function F in Eq.(6.1.3c) is computed from



(A) (B)

FIGURE 6.5: Numerical results for the static cylinder. (A) The vortex strength computed using the minimization procedure is compared to the analytical result. (B) The resulting streamline for the inviscid flow past the cylinder.

γ, i.e.

$$F\left(\vec{x}\left(s\right)\right) = -\frac{\gamma\left(s\right)}{\delta t}. \tag{6.1.30}$$

Once this is done, The flux must be passed to the surrounding particles as a circulation change $\Delta\Gamma$ (Eq.(6.1.21)). The accuracy of the no-slip condition depends on the grid resolution used to resolve the boundary layer on the solid body. For example, for the same static cylinder, the surface velocity is computed at an azimuthal position $\theta = 288^\circ$ after the vortex sheet has transferred to the particles. Evidently, the accuracy of both the no-slip and no-through conditions marginally improves as the grid spacing h is reduced (Figure (6.7)). One interesting observation is that the particle cluster serves to establish the boundary layer on the surface, which is physically consistent with the understanding that the generation of vorticity is a direct manifestation of an existence of a boundary layer. Furthermore, one can visualize how the boundary layer has developed after the cylinder has impulsively started (Figure (6.6)). These calculations confirm the validity of the current numerical approach.

6.1.4 The wall diffusion algorithm

The wall diffusion is handled by prescribing a uniform grid. At each time step, the cells in the grid are given an active state. This parameter determines if the cell is receptacle to receive a circulation from the wall flux. For cells that occupy inside of the bodies, they are set inactive. The aim of this step is to apply the requirement that the vorticity shall only be generated outside of the body; any cells that fall within a certain distance of the boundary need to be marked. The first stage of the algorithm is to associate a Boolean array to the grid and identify the inactive cells. This requires an efficient point-in-polygon (PIP) algorithm that can be easily implemented in the GPU. The winding number algorithm is particularly suited for this task (O'Rourke,

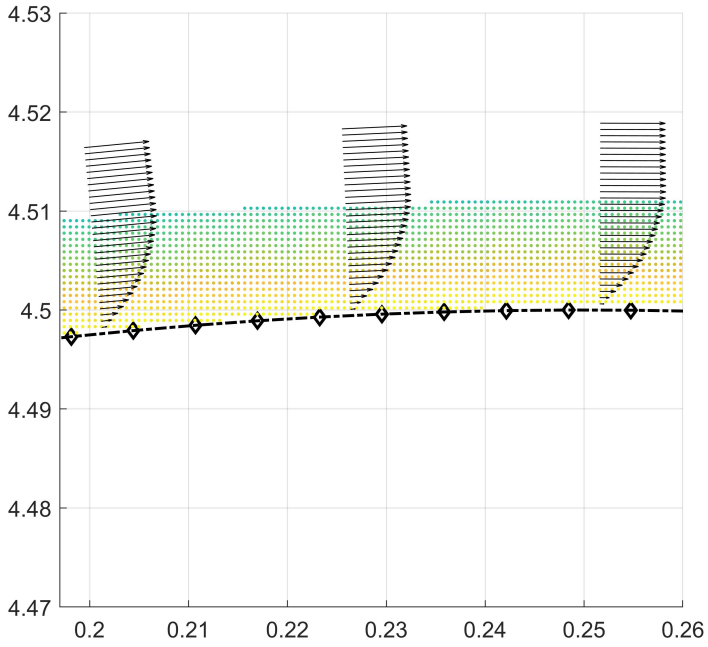
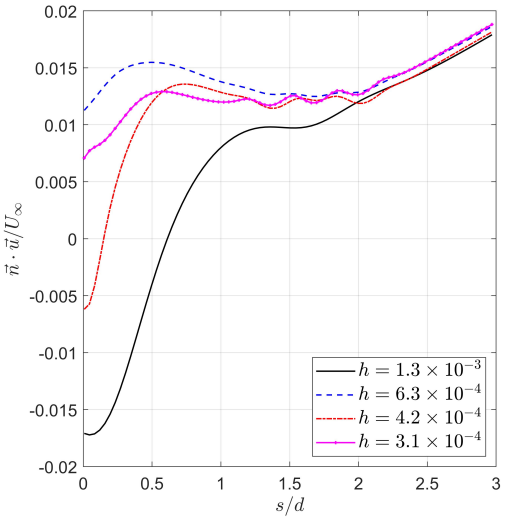


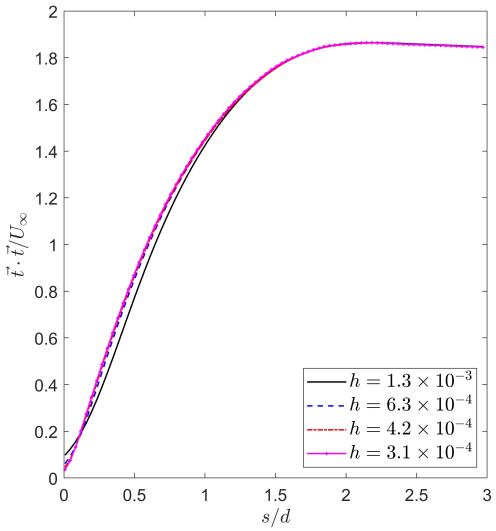
FIGURE 6.6: The computed boundary layer developed on the surface of the cylinder using a distribution of vortex particles (coloured dots). The accuracy of the boundary conditions depend on the grid spacing used. In this example, $h = 6.3 \times 10^{-4}$ was used.

1998). Once this is done, the body is assumed to produce a vorticity flux, which must create new or modify the circulation of existing particles. However, it is not necessary to diffuse the flux to the whole of the domain since the influence is a product of the exponential and error functions (Eq.(6.1.20)), which quickly diminish as ξ or η increases. Only the closest particles to the body are significant. In the code, this length is controlled by a diffusion parameter in terms of the number of cells that must extend outward to capture the necessary diffusion. The actual number of cells used for the diffusion depends on the kinematic viscosity of the flow. More precisely, it is related to the diffusive length scale $L_{\text{Diff}} := C\sqrt{\nu\delta t}$, where C is a user specified constant. The diffusive length scale may be interpreted as the average distance for which the diffusion phenomenon has propagated over the time-interval $[0, \delta t]$. This process is illustrated in Figure (6.8). Each panel would have a diffusive zone in which the wall flux is applied. If a particle already occupied some cells in the zone, then the flux is applied to that particle instead. Any unoccupied cells are subsequently turned into free vortex particles in a way that minimizes the void between particles whilst maintaining spatial homogeneity. To limit the growth of the number of particles, however, a threshold is implemented. Only those particles whose circulation greater than the threshold are considered.

Moreover, it is desirable for the code to conserve the total circulation of the wall diffusion. It was pointed out by Ploumhans and Winckelmans (2000) that, unless a body conforming grid is used, it is not always the case that the panel diffusion



(A) normal component



(B) tangential component

FIGURE 6.7: Variation of the normal and tangential components of the velocity as a function of the normalized perpendicular distance (s/d) from the cylinder at an azimuthal station $\theta = 288^\circ$ at different resolutions h. The development of a surface boundary layer is clearly visible.

would conserve the circulations. In their recommendation, $\Delta\Gamma$ should be appropriately modified so that the equality is strictly enforced:

$$d_j\gamma_j - \sum_{i\in I_j} \Delta\Gamma_{i,\text{conserved}} = 0 \tag{6.1.31}$$

where d_j is the length of panel j and $\Delta\Gamma_{i,\text{conserved}}$ is the actual circulation received by the i-th particle concerned by the panel. Those particles are labelled by the index set I_j. A Lagrangian multiplier approach was used to minimize the quantity

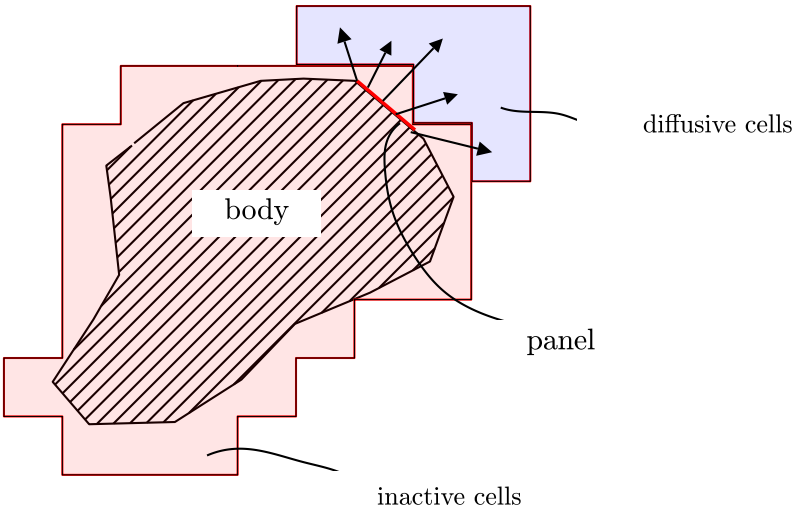


FIGURE 6.8: Schematic diagram depicting the main sequence of wall diffusion. Initially, the grid marks those cells that coincide with the body and are subsequently *turned-off*. Each panel is given a diffusion zone in which the wall flux is applied. Note that different diffusion zones might overlap.

$\sum_{i\in I_j}\left(\Delta\Gamma_i - \Delta\Gamma_{i,\text{conserved}}\right)^2/\Delta\Gamma_i^2$, which gives the correction as follows:

$$\Delta\Gamma_{i,\text{conserved}} = \Delta\Gamma_i + \frac{\Delta\Gamma_i^2}{\sum_{k\in I_j}\Delta\Gamma_k^2}\left(b_j\gamma_j - \sum_{k\in I_j}\Delta\Gamma_k\right). \tag{6.1.32}$$

The presence of solid bodies in the domain presents a particular difficulty in the re-meshing procedure. When particles settle near the surface (one or two particle width away), they do not have a complete re-meshing stencil for the re-meshing to work properly (recall that a re-mesh would require a 3×3 stencil in which the particle's strengths are interpolated to those cells). Ploumhans and Winckelmans (2000) devised a sophisticated approach in which several re-meshing schemes are combined together to minimise a global penalty function. While their approach is physically sound, but in practice, such a scheme is difficult to implement on the GPU. For this reason, a simpler approach is used instead. The current scheme uses the proper stencil for the re-meshing even for those particles within the close proximity of the wall. If the interpolated cells happen to intercept the wall boundary, the cells are deleted and their interpolated circulations are copied to the right hand side of Eq.(6.1.24) as a source term. This way, the correct vorticity is fluxed back to the domain in the subsequent time-step; therefore conserving the total circulation. Note this scheme is similar to the one used in Eldredge (2007).

6.1.5 Numerical implementation

The numerical procedure is described in this section. The work flow is split into two phases. First is the geometry generation. Users can specify the number of solid bodies present in the computational domain and their dynamic motion by creating the body objects with the specified motion handle (MATLAB's internal handle class). Secondly, the simulation parameters are specified. This involves constructing a uniform grid which is used to facilitate the wall diffusion algorithm in Section 6.1.4 as well as to handle the grid distortion discussed in Section 3.1.4.

Once the simulation parameters have been specified, the main loop of the code proceeds as follows:

1. If the number of free vortex particles is non-zero, the Biot-Savart induction is evaluated at the particles' locations and the particles are advanced following the local velocity. At the new locations, the circulation strengths of the particles are modified according to the PSE scheme.

2. The coordinates of the bodies are updated by solving the motion equations: $d\vec{x}_b/dt = \vec{u}_b$ and the uniform grid is re-established (i.e. updating the boolean array of the cells)

3. If remeshing is required, the particle cluster is restarted at the grid centroid points using the M_4' interpolation scheme.

4. If any particles fall within a certain distance of the boundary, they are removed from the domain and their circulations are added to the right hand side of Eq.(6.1.24) as the source term.

5. The influencing matrix is reassembled due to the relative motion of the bodies (If there is only one body, this step is skipped) and the particle velocity is evaluated at the control points. This step partitions the field into the near and far fields relative to the bodies. A direct sum is applied by any 'near-field' particles while a local expansion is constructed at the bodies' centroid due to the 'far-field' particles.

6. The vortex sheet strength γ is then determined by solving Eq.(6.1.29), which is subsequently turned to the boundary flux F.

7. Any particles within the diffusive length scale L_{diff} of the bodies receive a change of circulation $\Delta\Gamma$ due to this flux, i.e. Eq.(6.1.21).

8. post-process outputs and repeat step 1.

6.1.6 Force calculation

The aerodynamic force acting on a body generally consists of two components - the pressure difference and the viscous shear. To derive these two forces, we note that the Navier-Stokes equations for the velocity $\vec{u}$ and pressure p can be expressed in component form:

$$\frac{Du_i}{Dt} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j}, \tag{6.1.33}$$

where D/Dt denotes the material derivative, ρ is the density and τ_{ij} is the 'kinematic' incompressible stress-tensor defined as follows:

$$\tau_{ij} := \nu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right).$$

Integrate Eq.(6.1.33) in the fluid domain D and utilise the Divergence theorem, one has that:

$$\rho \int_D \frac{Du_i}{Dt} dA = \int_{\partial D} (p_\infty - p)\, n_i ds + \rho \int_{\partial D} \tau_{ij} n_j ds,$$

where n_i is the i-th component of the outward pointing normal on the bodies, and p_∞ is the far-field pressure. The first integral on the right gives the pressure force per unit length while the second integral gives the viscous force.

By defining the force vector $\vec{F}$, the force acting on the bodies is then expressed as follows:

$$\vec{F} = -\rho \frac{d}{dt} \int_D \vec{u}\, dA.. \tag{6.1.34}$$

For a body prescribed with the body velocity of the form $\vec{u}_b = \vec{u}_0(t) + \Omega(t)\vec{k} \times (\vec{x} - \vec{x}_{\text{cen}}(t))$, Eq.(6.1.34) can be expressed, with the use of the internal body vorticity, in terms of the particle circulation and positions (Koumoutsakos, 1993):

$$\vec{F} = -\rho \frac{d}{dt} \left(\sum_{i=0}^{N-1} \Gamma_i \vec{e}_z \times \vec{x}_i \right) + \rho |D| \frac{d\vec{u}_0}{dt} - 2\rho |D| \frac{d}{dt} \left(\Omega(t)\, \vec{e}_z \times \vec{x}_{\text{cen}}(t) \right). \tag{6.1.35}$$

Here, $|D|$ denotes the area of the body. However, care should be exercised to compute the time derivative after a re-meshing step. This is because, while the M_4' interpolation kernel conserves the linear moment but it does not necessarily conserve the time derivative. Alternatively, Eldredge (2007) derived an expression that avoids dealing with the time discretization explicitly:

$$\vec{F} = \rho\nu \oint_{\partial D} \left((\vec{y}(s) - \vec{x}_{\text{cen}}) \times \vec{k} \frac{\partial \omega}{\partial n} + \vec{t}\omega \right) ds + \rho |D| \frac{d\vec{u}_0}{dt}, \tag{6.1.36}$$

Nonetheless, once the force is computed the drag (C_D) and the lift coefficient (C_L) may be then defined by:

$$C_D = \frac{\vec{F} \cdot \vec{e}_x}{\frac{1}{2}\rho U_\infty^2 L}, \quad C_F = \frac{\vec{F} \cdot \vec{e}_y}{\frac{1}{2}\rho U_\infty^2 L},$$

where L is the characteristic length of the body, $\vec{e}_x$ is the streamwise unit vector and $\vec{e}_y$ is the cross-stream unit vector.

6.2 Transient flow past static cylinders

The application of our solution procedure can be applied to a variety of flow scenario. Perhaps the simplest problem, which is considered as the benchmark problem by many investigators, involves the calculation of the flow velocity for an impulsively started cylinder translating in unbounded incompressible flow. The choice to reproduce this flow calculation is based on the fact that this problem has been amassed a large volume of research works in the past, therefore a good base of qualitative and quantitative metrics is available to gauge the accuracy of our codes.

6.2.1 Impulsively started cylinder at $Re = 550$

In this section, the flow past an impulsively started cylinder is simulated at the Reynolds number $Re = 550$. At this number, the flow is unsteady and separation develops on the leeward side of the cylinder which results in a recirculation zone that grows in time (Van Dyke, 1988). To characterise this flow, we define the non-dimensionalized time $T = U_\infty t/D$, where U_∞ is the free-stream velocity and D is the diameter of the cylinder. To obtain a well-resolved simulation, the time-step size δt, the kinematic viscosity ν and the grid spacing h must satisfy the inequality (Eldredge, 2007)

$$D_{\min} < \frac{\nu \delta t}{h^2} < D_{\max}, \tag{6.2.1}$$

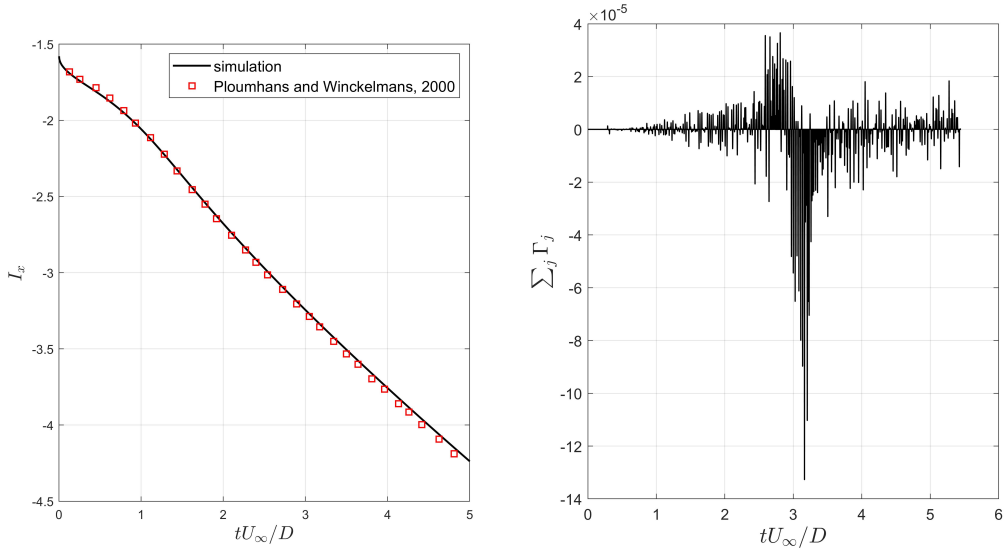
where the lower limit $D_{\min}$ is associated with the vorticity creation while the upper limit is associated with the stability condition of the PSE (Section 5.4). In effect, the time step is chosen in such a way that $\nu \delta t/h^2 = D_0$, where $D_0 = .5$. This value was used by Eldredge (2007) and Ploumhans and Winckelmans (2000) which seems to produce stable results. For this reason, the same value is adopted in this work. The parameter used for this particular simulation is shown in Table 6.1.

parameter	value	description
D_0	$1/2$	resolution parameter
δt	2.7×10^{-3}	time step size
ν	1.8×10^{-3}	kinematic viscosity
U_∞	1	free-stream velocity
δs	6.2×10^{-3}	body discretization
D	1	Cylinder diameter
ω_0	5.45×10^{-7}	vorticity cut-off
P	20	FMM truncation number

TABLE 6.1: Simulation parameters used for the impulsively started cylinder at $Re = 550$.

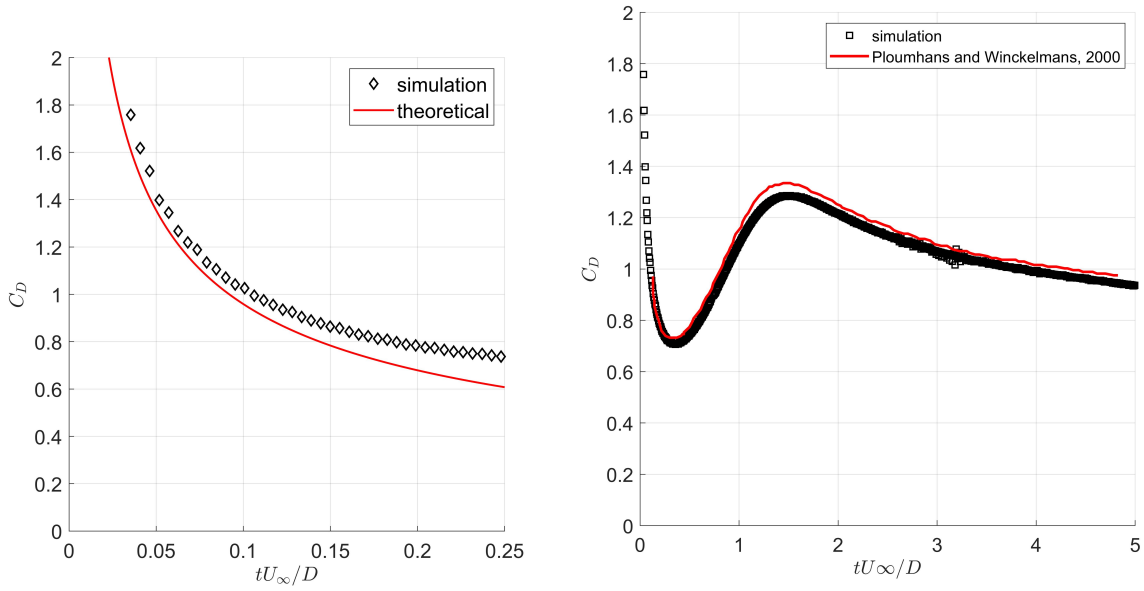
A crucial step towards validating the current numerical code is to determine whether the code is able to capture the recirculation zone properly. A series of snapshot is shown in Figure (6.12). Those plots show the development of the vorticity field at the non-dimensionalized time $T = 0.5, 1, 2, 3$ and 4. By comparing to the experimental results (Figure (6.11a)-(6.11c)), which show the streak-lines at times $T = 1$ and 2, the simulated size of the recirculation zone shows a fair agreement, especially at the latter time. Though the experimental streak-lines were for a slightly lower Reynolds number $Re = 500$. Initially, there is no vorticity in the flow. As the cylinder starts to move, an inviscid velocity field is established on the boundary of the surface. Vorticity is thus generated by the no-slip condition. Subsequently, the vorticity is shed and diffused to the surrounding fluid, which results in an increase thickness of the boundary layer. At some short time later, separation begins on the leeward side of the cylinder, which marks the start of the recirculation zone. The current implementation automatically ensures that such physical process is handled properly using layers of vortex particles to provide a means of receiving the boundary flux.

Quantitatively, the linear impulse I_x and the drag coefficient of the cylinder might be compared to the literature. Indeed, Figure (6.9a) presents a time-series for the linear momentum in the flow and this is to be compared with the results of Ploumhans and Winckelmans (2000) for time up to $T = 5$. Figure (6.10b) shows the drag curve.



(A) (B)

FIGURE 6.9: Left shows the computed linear impulse I_x compared to Ploumhans and Winckelmans (2000) at $Re = 550$. Right shows the time-series of the total circulation in the flow domain.

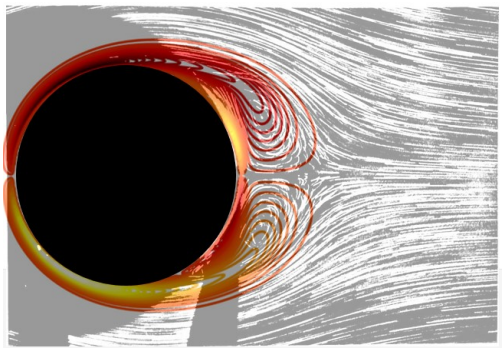


(A) (B)

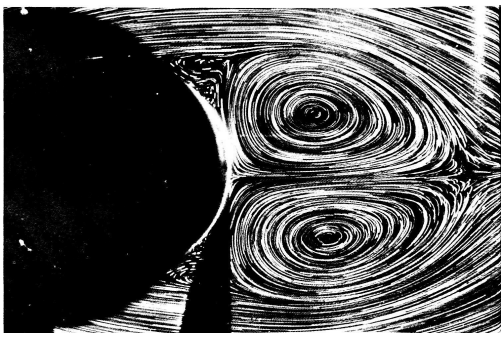
FIGURE 6.10: Left shows the computed drag coefficient compared to the theoretical expression Eq.(6.2.2). Right shows the long time drag curve compared to Ploumhans and Winckelmans (2000).

The results match quite well, except possibly at the local maxima at which our result shows a slight underestimation. The trend at least agrees well at a qualitative level. On theoretical ground, Bar-Lev and Yang (1975) derived an asymptotic expansion in terms of the small non-dimensionalized time T and large Reynolds number. The

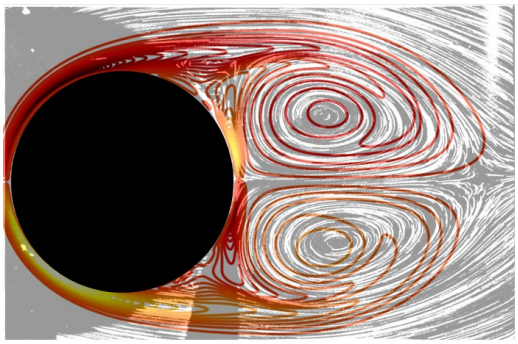
(A) $T = 1$, experiment



(B) $T = 1$, superimposed



(C) $T = 3$, experiment



(D) $T = 3$, superimposed

FIGURE 6.11: Left shows the experimental streak-lines (Van Dyke, 1988) of the flow past the cylinder at $T = 1$ and $T = 3$, respectively. On the right, the numerical vorticity contours have been superimposed on top of the snapshots of the experimental streak-lines. The numeric result shows a fair agreement in terms of capturing the size of the recirculation zone as the wake develops. The Reynolds number for this experiment was $Re = 500$.

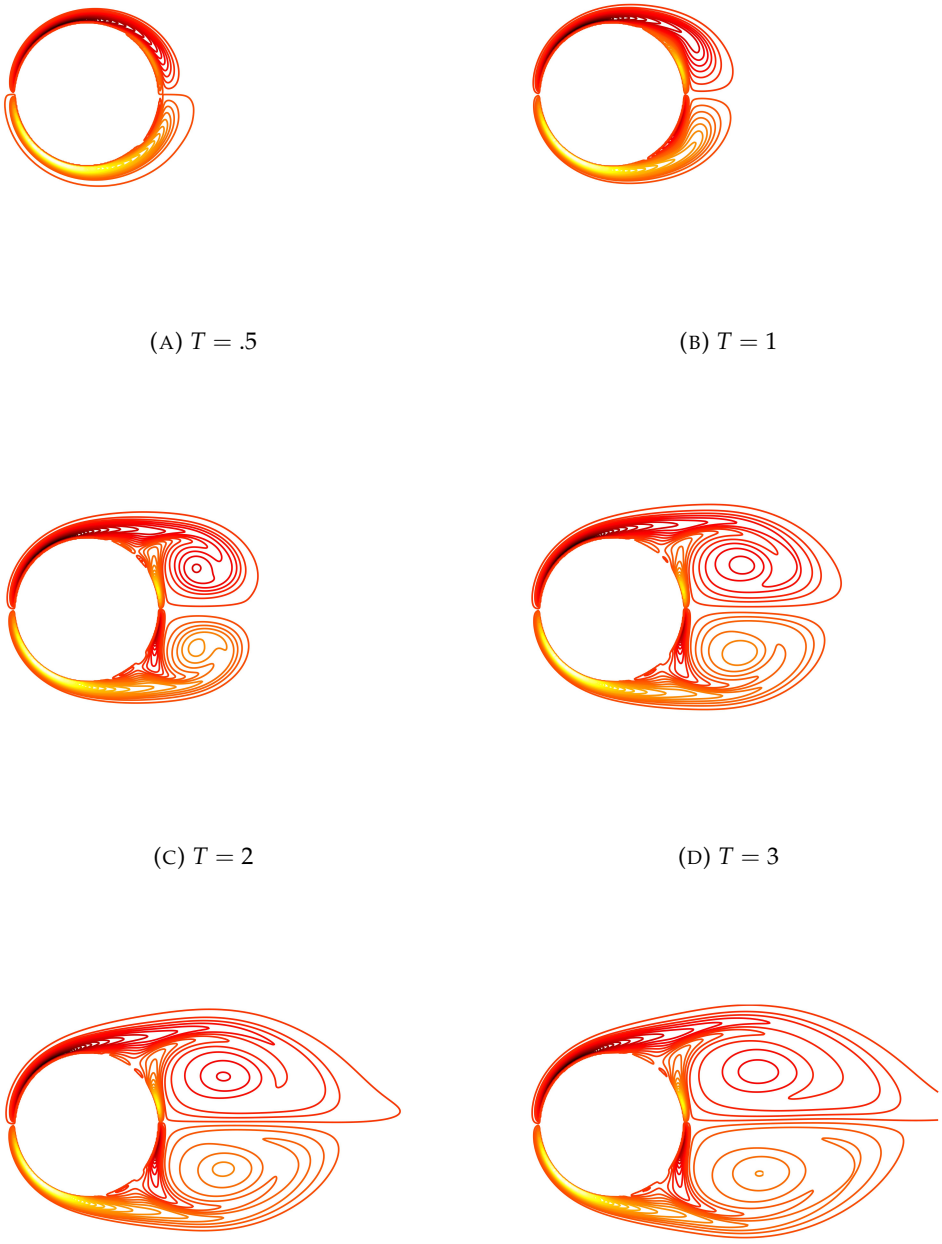
theoretical drag coefficient ($C_{D,\text{theo}}$) might be then computed as follows:

$$C_{D,\text{theo}} = 4\pi^{\frac{1}{2}} Re^{-\frac{1}{2}} T^{-\frac{1}{2}} + \pi \left(9 - \frac{15}{\pi^{1/2}}\right) Re^{-1}. \tag{6.2.2}$$

It is interesting to see how well the new results compare to the theoretical prediction. Figure (6.10a) shows the drag behaviour for small time. It is noteworthy to point out that Eq.(6.2.2) possesses a square root singularity. The computed results may be seen to tend to the asymptotic limit as $T \to 0$. For $T > 0.05$ the two curves start to diverge.

In addition, the total circulation is physically conserved in the real flow. The accuracy of the code, not just for force calculations, depends also on how well this property is respected. A time series of the circulation is presented in Figure (6.9b). Admittedly, the circulation curve exhibits a high level of fluctuations - possibly due to numerical errors in the re-distribution of the deleted circulations. But the code quickly corrects this spurious circulation by fluxing the correct amount of vorticity to satisfy Kelvin's theorem. This explains the random fluctuations. Overall, the averaged circulation is seen to be well maintained around zero, which shows that the new code is accurate.

Compared to the existing simulation codes in the literature, the new proposed



(A) $T = .5$ (B) $T = 1$

(C) $T = 2$ (D) $T = 3$

(E) $T = 4$ (F) $T = 5$

FIGURE 6.12: Vorticity contour plots for the impulsively started cylinder at the non-dimensionalized time shown. The Reynolds number of this flow is $Re = 550$

implementation is extremely efficient and fast. For example, in this simulation the total number of vortex particles went from $\sim$ 10000 to 415230 at the end of the simulation. The code manages to complete the simulation in less than 25 minutes, which

shows a large speed up compared to the simulation time in Ploumhans and Winckelmans (2000), where a low resolution run can take several hours. In the new code, most of the intensive parts of the simulation are offloaded to a GPU except for the linear solver, which uses MATLAB's built-in matrix equation solver instead.

6.2.2 Impulsively started cylinder at higher Reynolds numbers

At higher Reynolds number, the symmetric property of the recirculation zone is very sensitive to numerical errors. At $Re = 9500$, we were unable to maintain the symmetric wake as time progresses for the same resolution we used for the low Reynolds case. Specifically, $D_0 = 0.5$ is inadequate. Looking from the vortex strength distribution around the cylinder, the wake is symmetric for small time. However, the wake quickly develops asymmetries for $T > 1$. This occurs at the instances where the creation of two primary vortices are visible (see Figure (6.13)). Subsequently, the evolution of the primary vortices distorts the field further, which results in the progressive worsening of the wake symmetry. This is in stark contrast with the numerical results of Koumoutsakos (1993), where they managed to maintain symmetries past $T = 4$. However, by increasing the value D_0 to 1.5, it is readily seen that the symmetry is much better preserved than the previous case (Figure (6.14)). This suggests that the issue is not of the implementation of the method, but as a consequence of the stability condition required by the method. A possible limitation is that the current numerical code is not equipped with the adaptive grid in which the grid is conformed to the bodies. Although such grid strategy is the most accurate one can hope for, but their approach cannot be extended to bodies with arbitrary geometry. Together with the fact that incompressible flow is highly sensitive to flow disturbances at high Reynolds number, the numerical errors in the code act as artificial perturbations, which prematurely causes the wake to loss symmetry. However, one should note that the computed field contains most of the key characteristics as observed in Koumoutsakos (1993), such as the formation of primary vortices and etcetera. One could further investigate the parameter space to obtain a better match with the literature, but this is left as a topic for future work.

6.3 Conclusion

We have developed a new GPU-accelerated Navier Stokes solver for the flow field past bodies with arbitrary geometry in motion. The solver relies on the concept of operator splitting in which the advection and diffusion processes are serialized. The accuracy of the new developed code has been validated by simulating the impulsively started cylinder. All the results obtained show that the new developed code is extremely efficient and may be employed with advantage as a high fidelity CFD tool for a variety of flow problems in scientific and engineering applications.

(A) $T = 0.5$

(B) $T = 1.0$

(C) $T = 1.5$

FIGURE 6.13: Vorticity plots for the impulsively started cylinder at $Re = 9500$ for the case $D_0 = .5$. The wake quickly becomes asymmetric as time develops owing to the creation of two primary vortices on the surface boundary.

(A) $T = 0.5$

(B) $T = 1.0$

(C) $T = 1.5$

FIGURE 6.14: Vorticity plots for the impulsively started cylinder at $Re = 9500$ for the case $D_0 = 1.5$. The symmetry of the wake is now much better preserved. Though there is still some asymmetricity present.

Chapter 7

VAWT aerodynamic simulations

The motion cycle of the VAWT operates in a similar manner to that of a pitching aerofoil in that the angle of attack α, changes periodically. Thus the two engineering problems can be considered aerodynamically similar. However, one notable distinction between the two is that the VAWT may experience episodes of blade-vortex interaction (BVI), during which the shed vorticity from previous time-steps interact with the blade surfaces. This effect could be detrimental to the operation of the VAWT as a large variation of noise and vibration is introduced. Unfortunately such phenomenon is still not well studied and is an active area of research.

The aim of this chapter is to validate the new developed methodology and code in the context of the wind-turbine and aerofoil aerodynamics and to apply the new method to investigate the aerodynamics of VAWT and the aerodynamic interactions between turbines.

7.1 Aerofoil discretization and trailing edge smoothing

The geometry module forms an important part of the code. With a few exceptions, many optimised aerofoils do not have a closed analytical expression for the geometry definition. However, for the purpose of the current work, it suffices to use the popular NACA 4 digit series aerofoils whose construction can be derived by applying the concept of *camber* line and thickness distribution (Abbott and von Donenhoff, 1949). The construction begins by defining the mean *camber* line with equation $y_c(s)$, where s is the normalized parametrization variable. Specifically, for the 4 digit series aerofoils, the camber is taken to be of the form:

$$y_c(s) = \begin{cases} \frac{m}{p^2}\left(2ps - s^2\right) & 0 \leq s < p, \\ \frac{m}{(1-p)^2}\left(1 - 2p + 2ps - s^2\right) & p \leq s \leq 1. \end{cases} \tag{7.1.1}$$

where m is the maximum camber (in percentage of the chord c) and p determines the location of the maximum camber along the chord line. In fact, the first digit in the NACA designation gives the value $100m$ whilst the second digit designates the location of this camber.

In addition, a thickness distribution $t(s)$ is used in conjunction with the camber line to obtain the coordinates of the upper and lower surfaces. For the NACA 4 digit series aerofoils, t is given by:

$$t(s) = 5t_0\left(a_0\sqrt{s} + a_1 s + a_2 s^2 + a_3 s^3 + a_4 s^4\right), \tag{7.1.2}$$

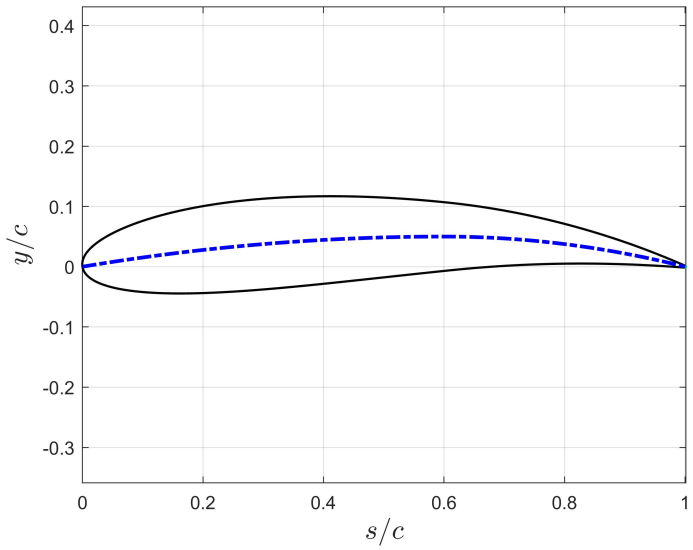


FIGURE 7.1: A NACA5612 cross section profile constructed by Eq.(7.1.3). The dotted blue curve gives the mean camber line.

with the constants taking values:

$$a_0 = +0.2969,\ a_1 = -0.1260,\ a_2 = -0.3516,\ a_3 = +0.2843,\ a_4 = -0.1015,$$

where t_0 is the maximum thickness. For example, the NACA5612 aerofoil sets the maximum camber to be 5% of the chord at the 60% chord position with a maximum thickness of 12% of the chord (see Figure (7.1)).

By defining the normal vector along the camber line as $\left(n_x(s), n_y(s)\right)$, the normalized coordinates $(\tilde{x}, \tilde{y})$ are obtained:

$$\tilde{x}(s) = s \pm t(s)\, n_x(s), \quad \tilde{y}(s) = y_c(s) \pm t(s)\, n_y(s). \tag{7.1.3}$$

For the NACA aerofoils, a small gap can be found at the trailing edge, which could give rise to stability issues in the code if left untreated. Currently, a quintic Hermite polynomial interpolation curve is fitted at the trailing edge so as to continuously vary the curvature around the gap. To this ends, let $\vec{S}(\lambda)$ denotes the two dimensional quintic polynomial, where λ denotes the local parametrization of the curve. We impose the condition that the curve shall be continuous in function value and the first and second derivatives. Specifically, let $\vec{x}_{\pm}$ denote the upper and lower trailing

positions respectively. Then, the interpolated curve must satisfy the following conditions:

$$\vec{S}(\lambda_{\pm}) = \vec{x}_{\pm}, \tag{7.1.4a}$$

$$\frac{d\vec{S}}{d\lambda}(\lambda_{\pm}) = \mp \frac{d\vec{x}_{\pm}}{ds} \tag{7.1.4b}$$

$$\frac{d^2\vec{S}}{d\lambda^2}(\lambda_{\pm}) = \mp \frac{d^2\vec{x}_{\pm}}{ds^2}. \tag{7.1.4c}$$

where $\lambda_{\pm}$ are the local parametric values that correspond to the upper and lower trailing points (in general, the two parts of the body might use different parametrizations to characterise the two regions separately). The change of sign in front of Eq.(7.1.4b) and Eq.(7.1.4c) determines the anti-clock wise orientation of the parametrization.

A special form of the Hermite polynomial may be constructed as follows:

$$\vec{S}(\lambda) = \vec{Q}_1(\lambda - \lambda_+) + (\lambda - \lambda_+)^3 \vec{Q}_2(\lambda - \lambda_-). \tag{7.1.5}$$

where $\vec{Q}_i(x) := \vec{a}_i + \vec{b}_i x + \vec{c}_i x^2, i = 1,2$ for some constant vectors $\vec{a}_i, \vec{b}_i$ and $\vec{c}_i$. The form in Eq.(7.1.5) makes substituting the conditions Eq.(7.1.4) rather trivial.

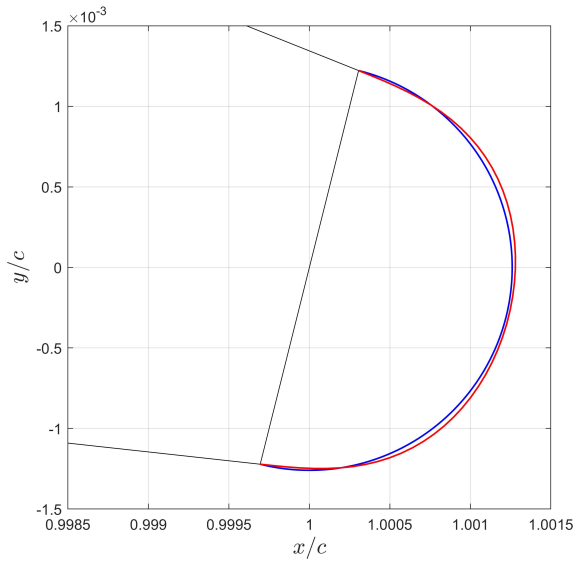
For demonstration of the interpolation technique, Figure (7.2) shows that interpolated trailing region for the NACA5612 aerofoil. Compared to a simple circular closure curve, which shows noticeable kinks at the joining points, the quintic interpolatant gives a smooth interpolation across the joining faces. Furthermore, the continuity of the curvature is seen to be properly respected as can be observed in Figure (7.2b).

If a non-normalized parametrization is used and the chord length is not unity, then the physical coordinates of the aerofoil are simply $c(\tilde{x}(s/c), \tilde{y}(s/c))$.

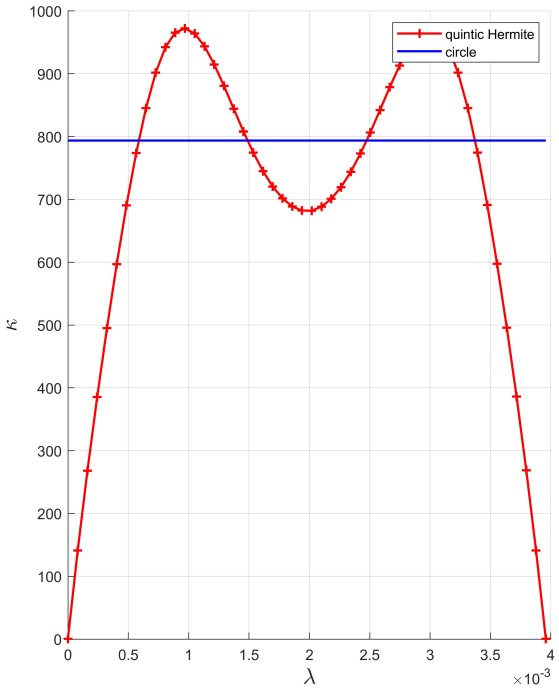
Generally, the curvature of the prescribed aerofoils is large at the leading and trailing edge. Indeed, it is often observed that the flow around those regions exhibits a large spectrum of kinematic variation. It is therefore desirable to have a nodal distribution that increases the node density in those sensitive regions. A *cosine* discretization is often adopted by many investigators (Dixon, 2008; Katz and Plotkin, 2001). Essentially, the parametric values s are computed by a cosine function of the form:

$$s = \frac{1}{2}(1 - \cos(\phi)), \tag{7.1.6}$$

where $\phi \in [0, \pi]$, which is to be discretized linearly. A simple analysis reveals that the node spacing $\Delta h \left(:= c\sqrt{(d\tilde{x}/ds)^2 + (d\tilde{y}/ds)^2}\Delta\phi ds/d\phi\right)$ approaches zero towards the trailing edge, which means the trailing edge nodes might be unrealistically packed together; resulting in the deterioration of the node uniformity (see Figure (7.3a)). One way to resolve this issue is to make sure that Δh does not tend to zero but instead approaches to a fixed controlled value. Thus, one can construct a quintic interpolation curve $s(\phi) = S(\phi - \phi_0)$ that satisfies the edge boundary conditions: $s = 1, ds/d\phi = \gamma, d^2s/d\phi^2 = 0$ at $\phi = \pi$, where S takes the 1-D form of Eq.(7.1.5) and γ is a user specified input. In practice the interpolation does not start from the leading edge but at the 75% chord. At the joining point, the interpolant matches the function value, first and second derivatives of the cosine distribution. For the same number of surface points (excluding the gap), Figure (7.3b) shows the node uniformity is much better respected.

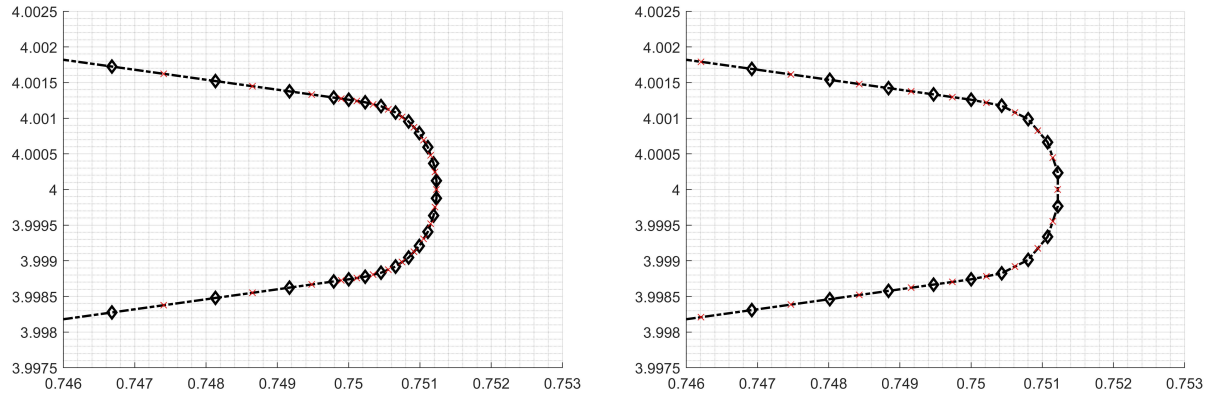


(A)



(B)

FIGURE 7.2: Example of the trailing edge smoothing using the quintic polynomial interpolant versus a simple circular closure. (A) The red curve corresponds to Eq.(7.1.5) whilst the blue curve shows a simple closure by a circle. (B) shows the curvature κ of the smoothing gap by the two methods.



(A) (B)

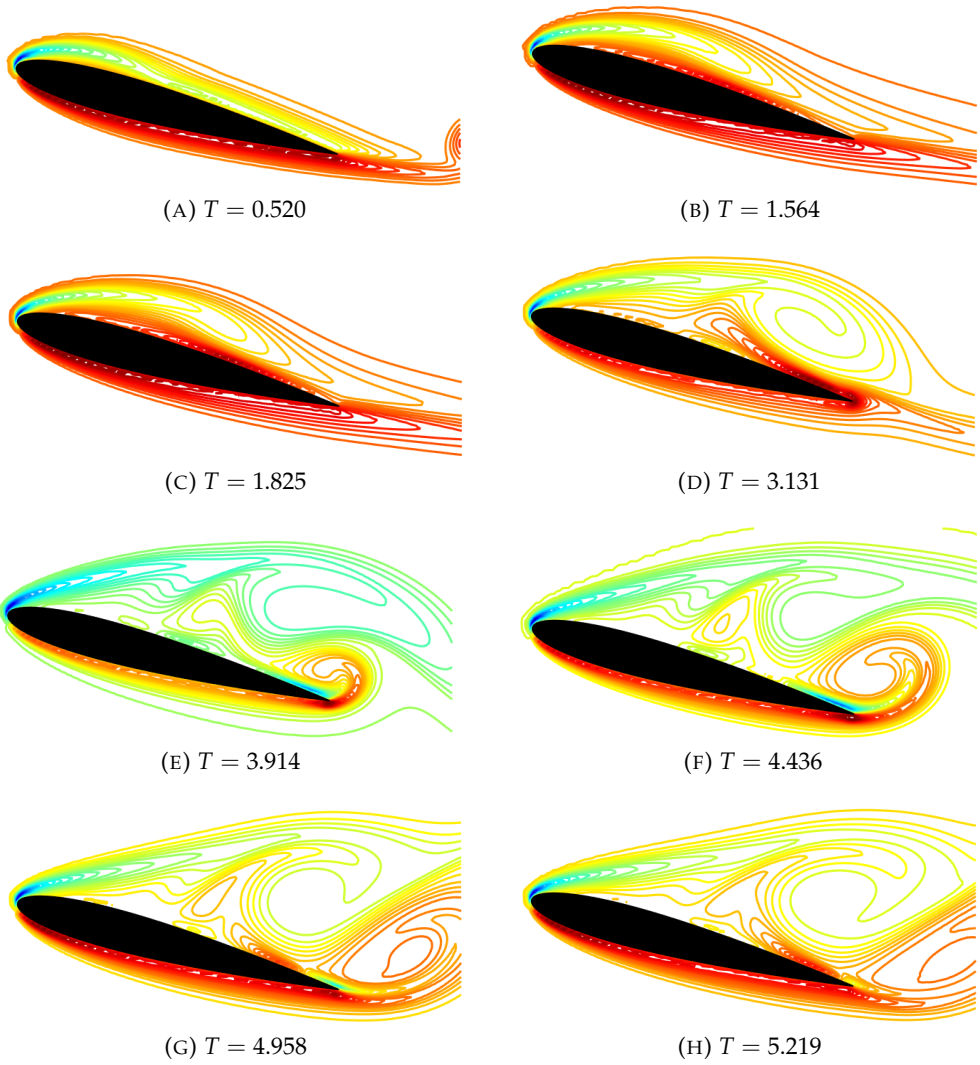
FIGURE 7.3: Trailing edge node distribution for a NACA0012 by the cosine distribution (A) and the polynomial interpolation (B). Note that the node spacing around the gap is taken to be the averaged spacing of the last two trailing edge nodes.

7.2 Aerofoil aerodynamics

As a starting point for the aerodynamic analysis of wind turbines, we investigated a range of aerofoil motions that correlate with the typical motion characteristic of a VAWT blade. The aim of this section is to provide a qualitative description and a quantitative measure of the flow behaviour during the stall event. Admittedly, the Reynolds number simulated here might not be applicable to the normal operating range (this is limited by the current hardware setup where the direct numerical simulation (DNS) approach provided by the code is still proved to be too taxing on the GPU at higher Reynolds number), but the approach may be important for applications such as start-up motion of the VAWT blade where the Reynolds number may remain low for a significant period of time.

7.2.1 Steady flow around a static NACA0012 at $\alpha = 15^\circ$

A key characteristic of the VAWT is the occurrence of dynamic stall at low tip speed ratio, during which the aerofoil experiences a lift *overshoot* followed immediately by a large reduction in lift. This is typically accompanied by the formation of a large leading edge vortex. This flow separation behaviour is investigated by simulating the flow field at a range of angle of attacks around a static NACA0012 aerofoil at low Reynolds number. The aim is to study the qualitative feature of the stall behaviour that occurs when the geometric angle of attack (α) exceeds that of the static stall angle (α_0). This behaviour is traditionally known to be difficult to predict accurately using the RANS models.



(A) $T = 0.520$ (B) $T = 1.564$

(C) $T = 1.825$ (D) $T = 3.131$

(E) $T = 3.914$ (F) $T = 4.436$

(G) $T = 4.958$ (H) $T = 5.219$

FIGURE 7.4: Snapshots of the vorticity contour for the NACA0012 at the non-dimensionalized time shown. The geometric angle of attack is $\alpha = 15°$.

The impulsively started NACA0012 at low Reynolds number was investigated experimentally by Huang et al. (2001). They used the particle tracking flow visualization method and particle image velocimetry (PIV) to visualize the instantaneous flow path of the surface flow when the aerofoil has impulsively started. The reported streamlines show that a significant unsteadiness is present even for cases when $\alpha < \alpha_0$. Huang et al. (2001) has characterised 5 distinct flow evolution modes depending on the value of α. For $\alpha < 2.5°$, the flow remains attached to the aerofoil. A trailing vortex separation occurs for $2.5° < \alpha < 6°$, a separation vortex for $6° < \alpha < 15.5°$, leading edge vortex separation for $15.5° < \alpha < 60°$ and finally bluff body flow for $\alpha > 60°$. In order to gain an understanding of how such process evolves and to correlate this understanding with dynamic stall, we perform two high fidelity simulations targeting at the separation vortex regime and the leading edge vortex regime using the developed code at the Reynolds number $Re = 1200$.

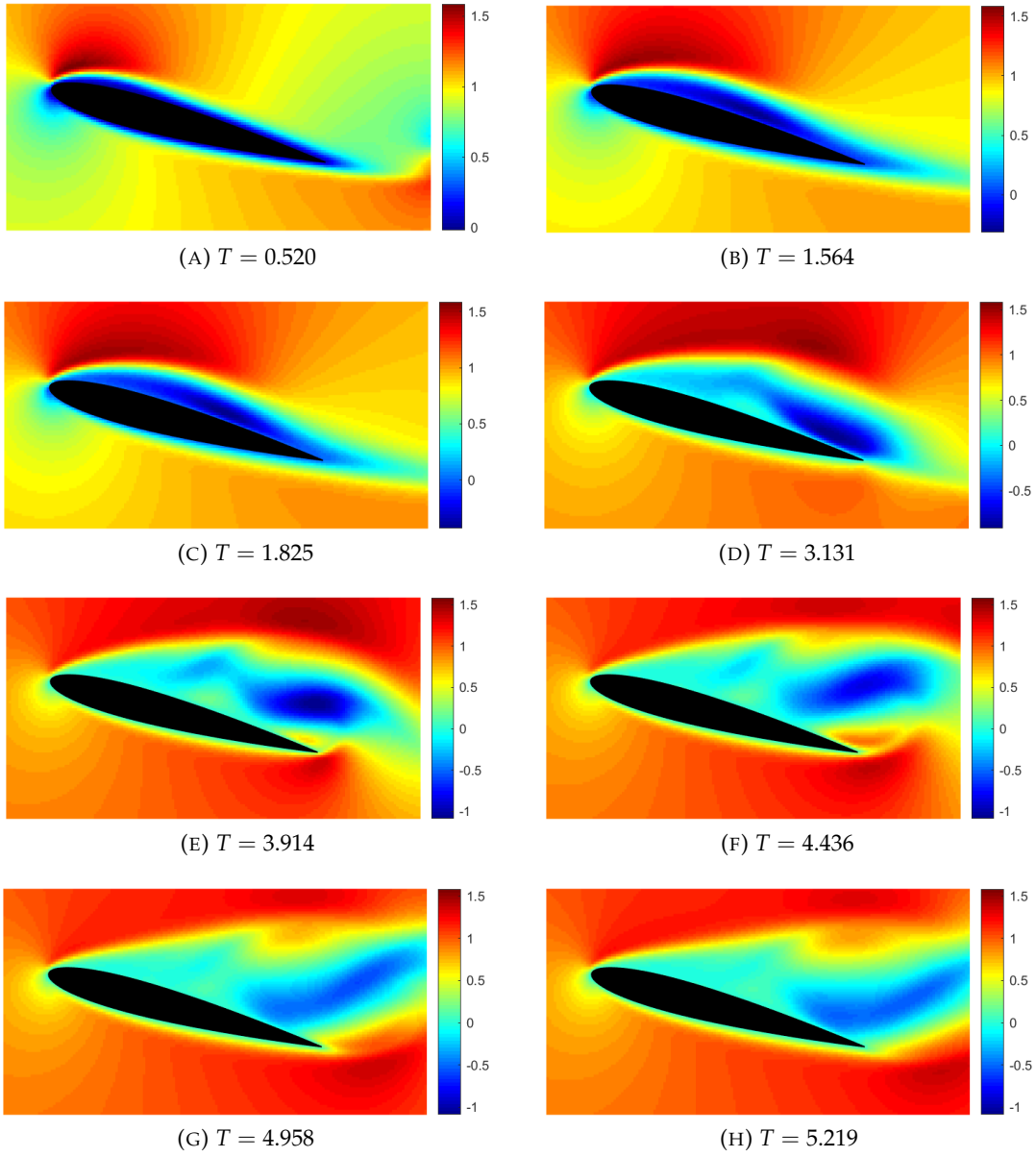
To reduce the simulation time, several vortex particle reduction schemes were employed and in particular is the implementation of a bounding box. This box removes any vortex particles when they are convected outside of the box boundary.

The deleted circulations do not redistribute back to the remaining vortices therefore one might not expect that the circulation in the domain to be conserved after a finite time. Secondly, the circulation cut-off was used to remove any *weak* vortices. This mainly affects vortices that have been convected sufficiently far from the aerofoil therefore it should not have impacted our analysis. Finally, the code uses a *mixed* order time integration scheme. By mixed order, we mean a combination of first-order forward Euler and second-order Adam-Bashforth. The first order scheme is applied to newly generated particles and for the re-meshed particles after the re-meshing operation has been performed. The second order scheme is for those remaining particles where their velocity data have been stored in the previous time-step. This is in contrast to the static cylinder simulations where we have used a second-order Runge-Kutta for the re-meshed particles. However, after several numerical tests (not documented), it is found that there is no significant difference in terms of the flow visualization and the force quantities. The parameters for the simulations are listed in Table 7.1.

parameter	value	description
c	1	aerofoil chord
$\Delta h_{\max}$	7.19×10^{-3}	largest body panel length
$\Delta h_{\min}$	2.58×10^{-4}	smallest body panel length
$\Delta h_{\max}/\Delta h_{\min}$	27.8	aspect ratio of the panels
h	3.60×10^{-3}	grid size
δt	7.77×10^{-3}	time step size
ν	8.33×10^{-4}	kinematic viscosity
D_0	1/2	resolution/stability parameter
ω_0	2.50×10^{-6}	circulation cut-off
$\lambda := \sigma/h$	1	overlap factor
P	17	FMM truncation number
U_∞	1	free stream speed
α	$15^\circ, 30^\circ$	geometric angle of attack

TABLE 7.1: Simulation parameters used for the static NACA0012 at $Re = 1200$. Note here that no physical units are given because the units are normalized by the aerofoil chord and the free-stream speed, so that one could recover the units via dimensional analysis.

In the first test case, the geometric angle of attack was set to 15°. At this angle, the static stall angle has been exceeded. After the initial transient (as evident by the convection of the starting trailing vortex), the flow over the suction side remains attached. No reverse flow was observed (Figure (7.5a)). At $T = 1.564$, reversed flow is now sighted near the trailing edge indicating the eminent formation of a surface vortex. At some short time later ($T = 1.825$), the surface vortex has grown enough which triggers the boundary layer to separate from the surface. Reverse flow becomes much more prevalent in much of the suction side. As the surface vortex develops, it is convected towards the trailing edge whilst entrains the fluid around it (Figure (7.4d)). This leads to the formation of a secondary counter clockwise rotating vortex at the trailing edge that grows in size as the primary vortex continues to leave the surface. Subsequently, when the secondary trailing vortex has grown in sufficient length, it perturbs the outer flow and creates a new clockwise rotating surface vortex near the mid chord (Figure (7.4g)). The pair then interacts and moves away from the aerofoil surface turning the trailing edge fluid circumferentially towards the suction side from the pressure side (thus forming a new counter clockwise



(A) $T = 0.520$ (B) $T = 1.564$

(C) $T = 1.825$ (D) $T = 3.131$

(E) $T = 3.914$ (F) $T = 4.436$

(G) $T = 4.958$ (H) $T = 5.219$

FIGURE 7.5: Snapshots of the non-dimensionalized u-velocity contour for the NACA0012 at the non-dimensionalized time shown. The geometric angle of attack is $\alpha = 15°$.

rotating trailing edge vortex (TEV)). Due to this, a periodic shedding of vortices is subsequently established forming the well-known von-Karma vortex street (Figure (7.8)). This sequence of events is confirmed in the flow visualization carried out by Huang et al. (2001). Indeed, it can be seen that the time stamps of this sequence of characteristic motions match quite well with the reported experiment in Figure (7.6). From a dynamic point of view, when the primary surface vortex has grown in sufficient length (around $T = 3$), this is accompanied by a large reduction in the lift coefficient as can be observed in Figure (7.7a). Indeed, when the primary surface vortex has left the foil, a period of recovery by the lift can be found. Subsequent evolution sees the lift to increase and decrease due to the alternating shedding of vortices.

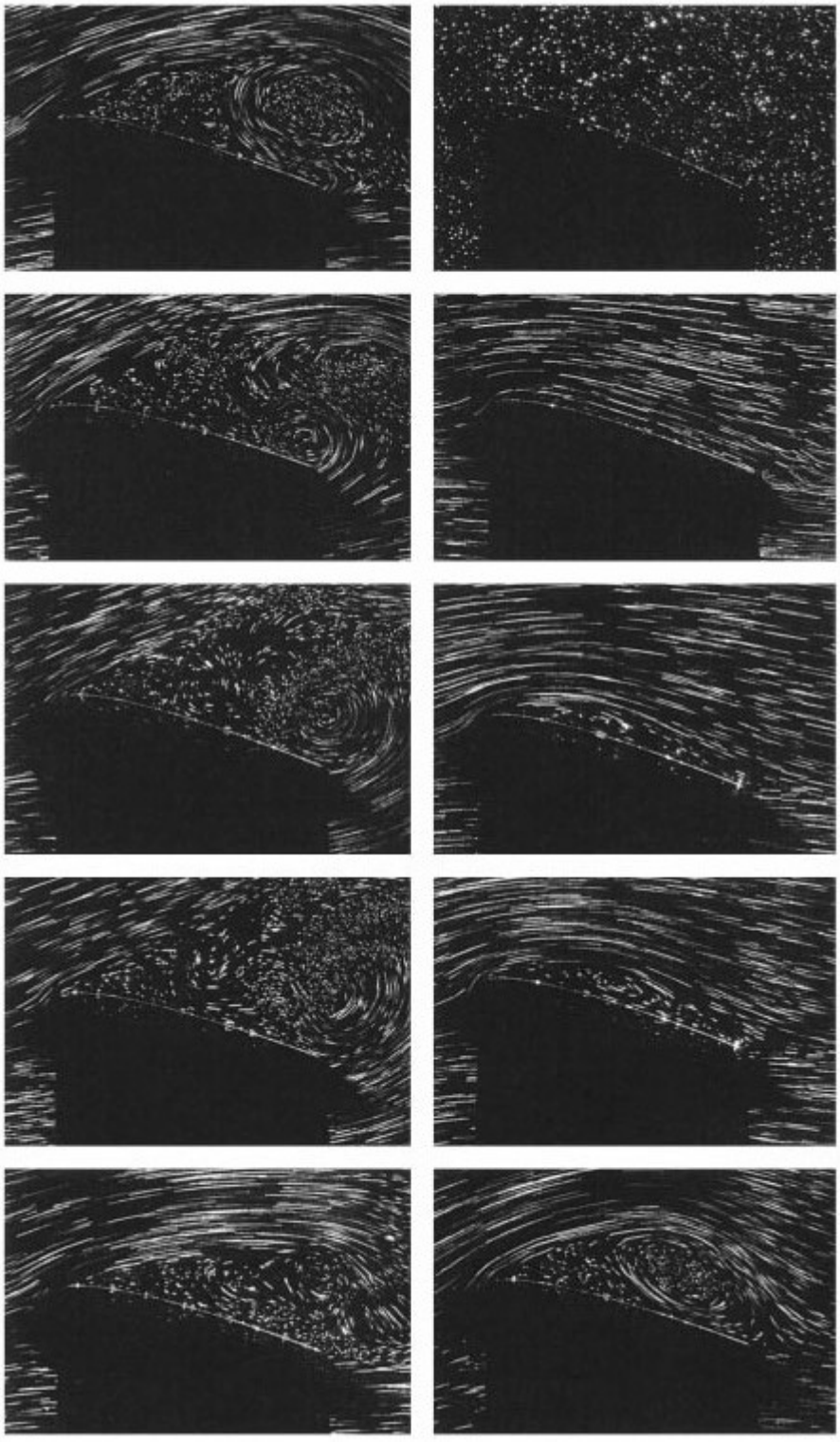
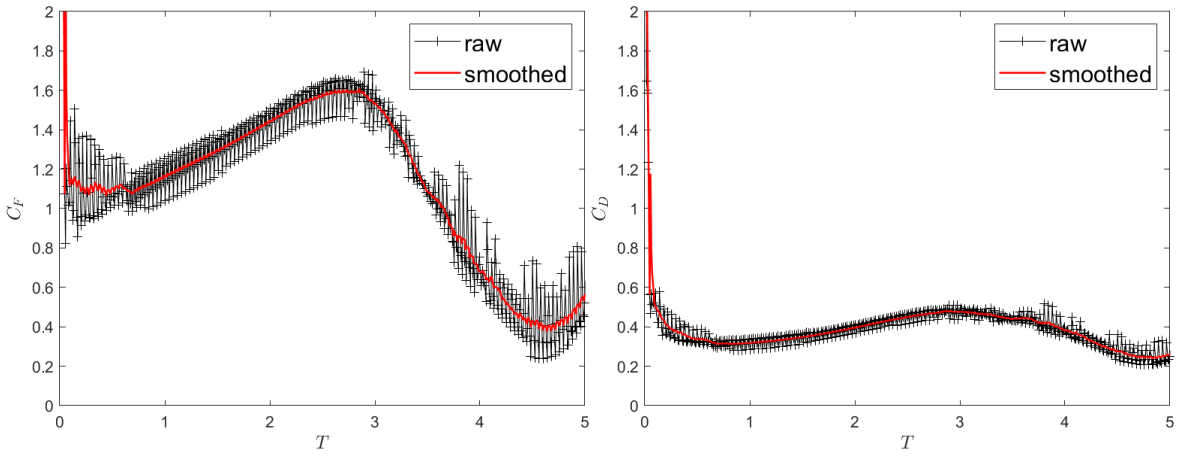


FIGURE 7.6: The plots show the experimental pathlines of the tracking particles conducted using the PIV by Huang et al. (2001). In increasing time, the snapshots are sequenced from top right to bottom, top left to bottom with the following timestamps: $T = 0, 0.520, 1.564, 1.825, 3.131, 3.914, 4.436, 4.958, 5.219, 6.524$



(A) C_F (B) C_D

FIGURE 7.7: C_F and C_D plots for the impuslively started NACA0012 at $\alpha = 15°$. The noisy black lines corresponds to the lift and drag coefficients obtained from Eq.(6.1.36) whilst the red line is the smoothed data using the moving box averaging technique with a 5-point averaging window. It can be observed that a large drop of lift occurs when the surface vortex has evolved to a sufficient length.

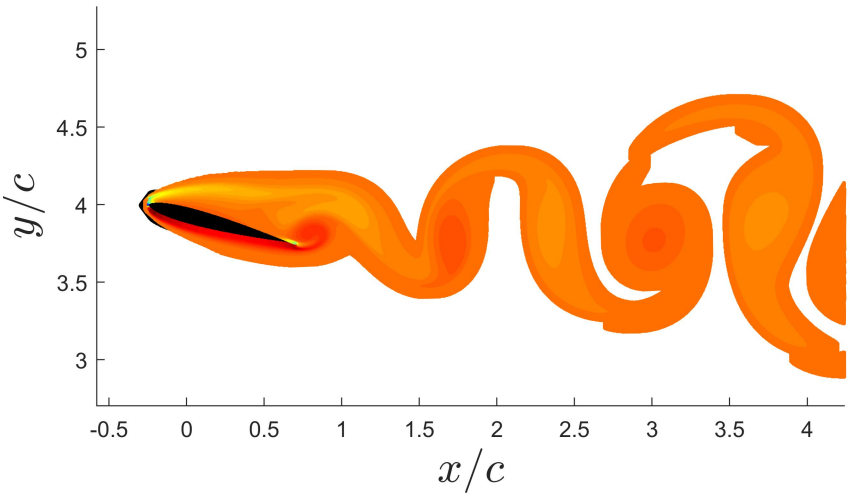


FIGURE 7.8: After the initial transient of the impulsively started NACA0012 at $\alpha = 15°$, alternating vortices are shed to create what is known as the von-Karma vortex street.

7.2.2 Steady flow around a static NACA0012 at $\alpha = 30°$

At $\alpha = 30°$, the flow moves into the leading edge vortex regime which is characterised by the formation of a strong vortex near the leading edge. Immediately after

the aerofoil is jerked into motion, adverse pressure gradient near the leading edge results in a laminar separation bubble. This bubble persists for a short time and is quickly developed into a leading edge vortex (LEV). The vortex does not immediately propagate to the wake and is instead moving along the suction surface; growing in size as it does so. Near the trailing edge, the LEV entrains the fluid around the trailing edge which results in a counter-rotating TEV that then *lifts* the LEV into the wake. In the same manner as in the previous case, a clockwise rotating vortex is then formed near the mid chord. Under the effect of the leading edge vortex, the trailing edge vortex quickly leaves the aerofoil leaving the nascent surface vortex to continue to grow. The instantaneous streamlines of this sequence is presented in Figure (7.9). The interpolated PIV streamlines of Huang et al. (2001) is also given which shows that the computed behaviour of the flow matches extremely well with the experimental results. Unsurprisingly, at such extreme angle of attack, the drag force is the dominant force at least during the initial leading edge vortex development (Figure (7.10)). However, at $2 < T < 2.66$ drop in both the lift and drag is reported. This drop can be attributed to the fact that the leading edge vortex has reached the trailing edge. Subsequent formation of the trailing edge vortex helps to recover the lift marginally ($2.66 < T < 3.66$) without inducing too much drag. Indeed, this is to be expected since the counter-clockwise rotating vortex serves to attach the separated flow on the surface whereby reducing the adverse pressure gradient. However, as the surface vortex continues to grow, substantial drop in both the lift and drag is observed.

The short time analytical solution for the lift coefficient has been derived by Graham (1983) for an impulsively started aerofoil at high angle of attack. The theoretical lift is found to be:

$$C_{L,\text{theo}} = T^{\frac{3-2k}{2k-1}}, \tag{7.2.1}$$

where $k = 2 - \theta_0/\pi$ and θ_0 is the trailing edge angle. Note that Eq.(7.2.1) is independent of the angle of attack. For the NACA0012, $\theta_0 = 0.303$ or 17.38°. Figure (7.11) shows a comparison between the computed lift coefficient for $\alpha = 15^\circ$ at short time and Eq.(7.2.1). Although the computed data fluctuate, but one can agree that the two set of data show a fair agreement. More reassuringly, at $\alpha = 30^\circ$, the computed lift coefficient also follows a similar short-time trend.

To be more explicit, the computed force is obtained using Eq.(6.1.36), in which the surface vorticity ω is evaluated using the a central finite difference scheme according to the Poisson's equations:

$$\nabla^2\psi = -\omega,$$

where ψ is the stream function due to the non-mollified vortex particles. The close analytical expression for ψ is given by Eq.(4.3.4). The reason for the observed fluctuation in the computation of the forces can be attributed to the complex dynamic between creation and destruction of vortex particles. One recalls that the method employs a destruction algorithm in which particles close to the body geometry is automatically removed from the fluid domain. Meanwhile, new particles are constantly introduced to accept the boundary flux. Therefore, in the proximity of the boundary surface, the number of vortex particles fluctuates quite irregularly. This fluctuation is especially severe at the onset of the simulation where the boundary layer has not yet established. Determining the force distribution during such a short transience (before the establishment of the boundary layer) has always been a difficult task in a vortex particle method (Noca et al., 1999). Moreover, Eq.(6.1.36) introduces additional uncertainties as it involves determining the surface vorticity, which

is not well behaved in cases that the number of vortex particles is not fixed. Therefore Figure (7.11) should serve as an indicative measure of the expected qualitative behaviour.

Whilst Eq.(6.1.35) minimises the fluctuation as is evidenced in the computation of the flow past circular cylinder in Chapter 6, unfortunately such an approach cannot be correct in cases when particles have to be removed from the domain for the sake of avoiding the particle cluster size becoming too large. One approach, which is currently not implemented in this work, is to obtain the force via a finite and arbitrarily chosen region and whose velocity fields in the region are known to be well-behaved. Indeed such an approach can be found in the work of Noca et al. (1999) where they obtained an expression relating the local velocity fields and their derivatives, therefore bypassing the limitations of Eq.(6.1.35) and Eq.(6.1.36).

In order to capture the complex aerodynamic behaviour, a large number of vortex particles was used. At the end of both simulations, over a half million particles were present in the computational domain. At this resolution, the computation can be regarded as a DNS. Accounting for the post-processing (i.e. force calculation and solution information display), the GPU takes approximately one to two seconds to update the solution at each time step. This represents a huge reduction in computing time compared to the literature (Eldredge, 2007; Ramachandran et al., 2007).

7.2.3 Rotating NACA0012 at TSR $= 3.2$

The dynamic motion of a NACA0012 undergoing a constant rotation Ω about the origin is simulated in this section. In this mode, the aerofoil has the same operating characteristic as a VAWT whose geometric angle of attack is given as a function of the azimuthal angle θ. Let $\vec{u}_b$, $\vec{u}_\infty$ and $\vec{u}_e$ denote the body velocity, wind velocity and the effective/resultant velocity respectively, then one can show that the local geometric angle of attack α is given by:

$$\tan(\alpha+\beta) = \frac{\vec{u}_e \cdot \vec{e}_r}{\vec{u}_e \cdot \vec{e}_\theta} = -\frac{\cos\theta}{\mathrm{TSR}+\sin\theta} \tag{7.2.2}$$

where TSR is the tip-speed ratio and β is the initial pitch angle. We adopted the convention that the blade pitches outward when β is positive and inwardly pitched when $\beta < 0$. Since $\theta = \Omega t$, therefore α is a function of time. Figure (7.12) shows the geometric definition for the various nomenclatures used. The tip-speed ratio directly controls the aerodynamic property of the rotor, since a higher value (meaning that the rotor rotates faster) would lower the value of α. Moreover, if α is sufficiently small (at least smaller than the static stall angle) then flow attachment is possible for the full range of the azimuthal positions. One should note that at the blade level, the blade experiences a varying Reynolds number. In fact, if one defines the Reynolds number to be $Re := \|\vec{u}_e\|\, c/\nu$, with $\vec{u}_e$ given by:

$$\vec{u}_e = -\Omega R\vec{e}_\theta + \vec{u}_\infty.$$

where R is the rotor radius. Then as θ varies, the maximum value of $\|\vec{u}_e\|$ is $U_\infty\,(\mathrm{TSR}+1)$ with $U_\infty = \|\vec{u}_\infty\|$. Therefore in this simulation we adopt the definition $Re = U_\infty\,(\mathrm{TSR}+1)\,c/\nu$ as the effective Reynolds number. This definition mainly affects how the kinematic viscosity is prescribed in the code. In this simulation, the rotor consists of a single NACA0012 with a solidity of 0.25. The aim is to examine the flow structure during the stall event in the context of the VAWT. Therefore only the upstream part of the cycle is considered in this simulation ($90^\circ < \theta < 300^\circ$). The Reynolds number is

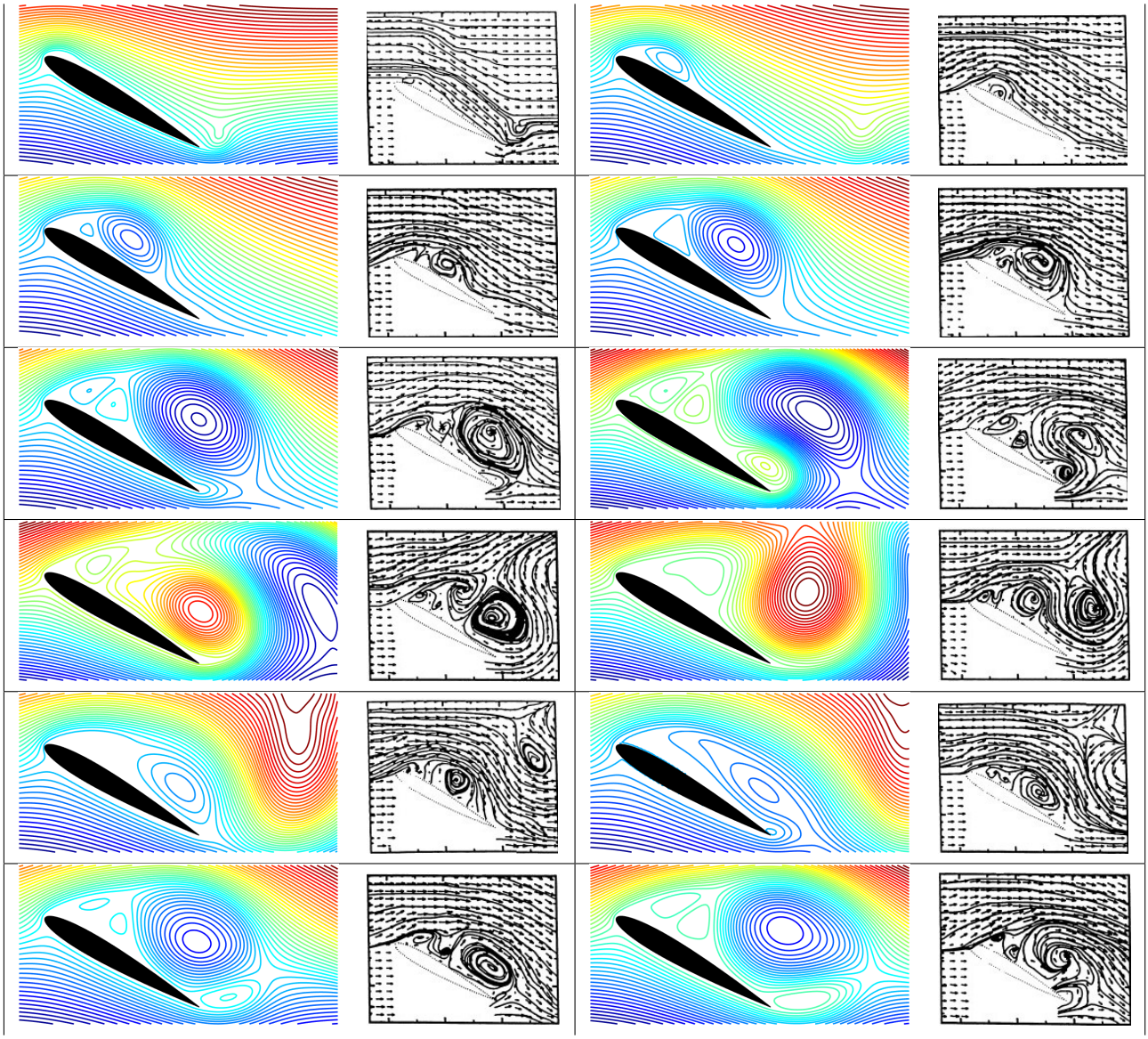
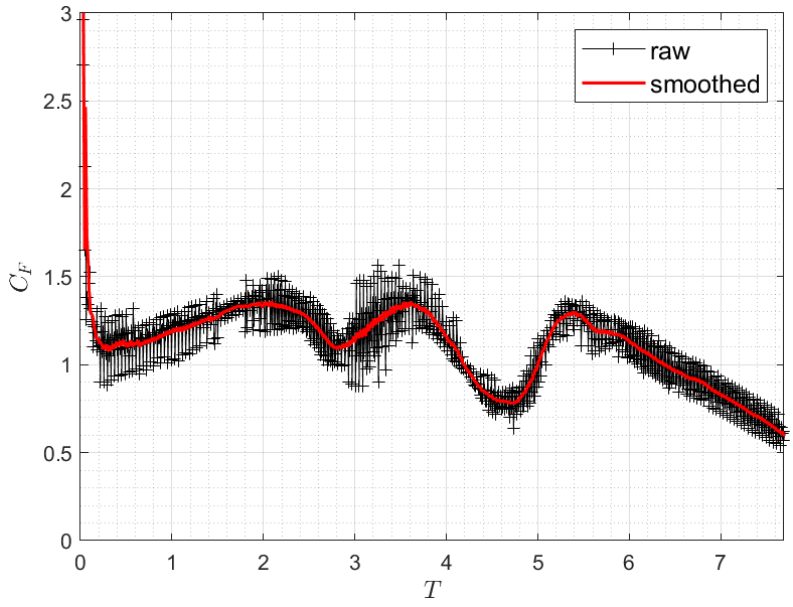


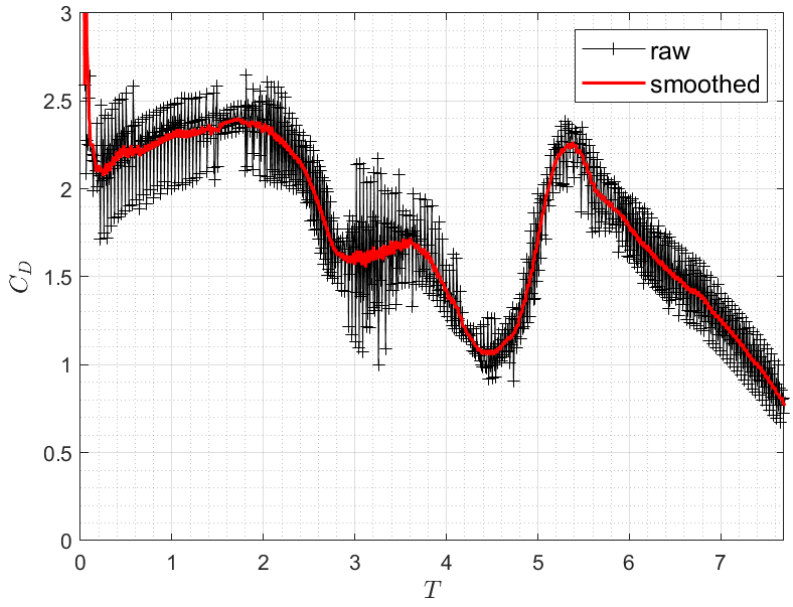
FIGURE 7.9: A sequence of streamlines captured at the non-dimensionalized times $T = 0.6521n - 0.5857, n = 1, 2, ..., 12$ at $\alpha = 30^\circ$. Figures in the first and third column are the simulated results and the interpolated PIV streamlines (Huang et al., 2001) are given in the second and fourth column. All snapshots are taken at precisely the same time stamps as the experiment. The motion is sequenced from the top left to right and top to bottom at an equal time-interval.

set to 3000. This alters the time-step size δt to 2.96×10^{-3} keeping in line with the condition that $D_0 = 0.5$ for a well-resolved simulation. A tip-speed ratio of 3.2 is used.

At this TSR, the aerofoil experiences severe stall in much of its upstream trajectory. The topological structure of the flow can be characterised by examining the vorticity field near the aerofoil surface and is found to share many of the topological similarities to the steady flow around the NACA0012 at a fixed $\alpha = 15^\circ$. Indeed, Figure (7.14) demonstrates the vorticity contours of the aerofoil in the upstream part of the cycle. To better facilitate the discussion, the shifted azimuthal angle (θ^*) is introduced, which is related to θ as follows: $\theta^* = \theta - \pi/2$. At the shifted azimuthal



(A)



(B)

FIGURE 7.10: Lift (A) and drag (B) coefficient for an impulsively started NACA0012 at $\alpha = 30°$.

$0° < \theta^* < 7°$, the flow remains attached to the aerofoil and is seen to be fairly stable. As α increases ($7° < \alpha < 11°$), in accordance with Eq.(7.2.2), instability waves start to propagate in the wake region leading the flow to separate from the surface around $\alpha = 11°$. The vortical structure at this stage is reminiscence of the static NACA0012 at $\alpha = 15°$ in which there is an alternating shedding of positive and negative vortices. Moreover, as α starts to decrease from the the maximum $\alpha(\approx 19°)$, a clear reattachment process can be seen. This reattachment process does not occur abruptly as reported by Riziotis and Voutsinas (2008) at high Reynolds number. Instead, the reattachment is gradual and it appears that the aerofoil has to convect the *sludge* flow off the surface in order for the reattachment to be finalized. At about $\theta^* \approx 200°$, the flow is fully attached.

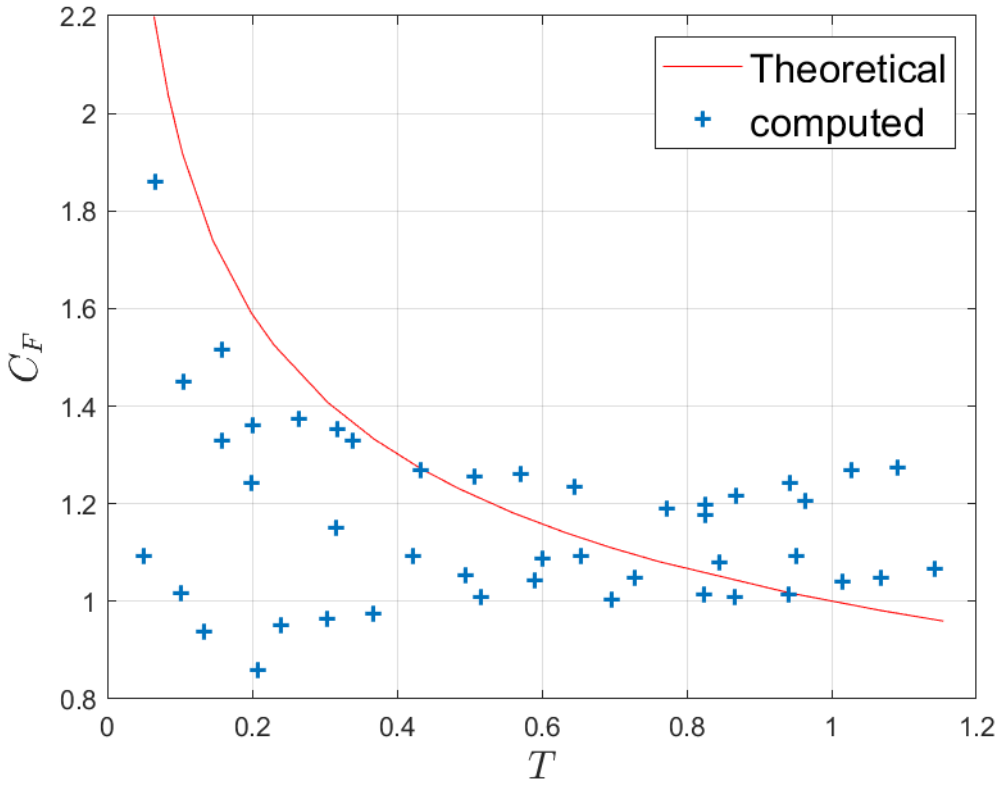


FIGURE 7.11: Computed short time lift coefficient for the NACA0012 at $\alpha = 15^\circ$ compared to Eq.(7.2.1).

7.3 Vertical axis wind turbine (VAWT) simulations

Using the developed technique in Chapter 3, namely the 2D variant of the BEM, this section explores the aerodynamic characteristic of the VAWT via a low-order method. As discussed in Section 3.3, the method is applicable in attached flow condition. However, as the blades in the VAWT experience a varying angle of attack (Eq.(7.2.2)), it is possible to ensure that flow remains attached to the blade during the full azimuthal traversal by making sure that the TSR is sufficiently large. To measure this precisely, one can show that the maximum angle of attack ($\alpha_{\max}$) experienced by the blade is obtained by differentiating Eq.(7.2.2), so that:

$$\alpha_{\max} = \tan^{-1}\left(\frac{1}{\sqrt{\mathrm{TSR}^2 - 1}}\right). \tag{7.3.1}$$

Here, the initial pitch is zero. Attached condition is assumed whenever $\alpha_{\max}$ is less than the static stall angle. Thus the minimum tip-speed ratio ($\mathrm{TSR}_{\min}$) required for attached condition for a given static stall angle α_0 is obtained:

$$\mathrm{TSR}_{\min} = \frac{1}{\sin\alpha_0}. \tag{7.3.2}$$

It should be stressed that determining the static stall angle is not a trivial task as it also depends on the surface roughness. Sufficed to say, the general consensus seems to be that α_0 varies between 10° and 12° for the NACA0012 at high Reynolds number (Abbott and von Donenhoff, 1949), which corresponds to the $\mathrm{TSR}_{\min}$ varying between 4.81 and 5.76. In practice, however, the TSR can be slightly lower without

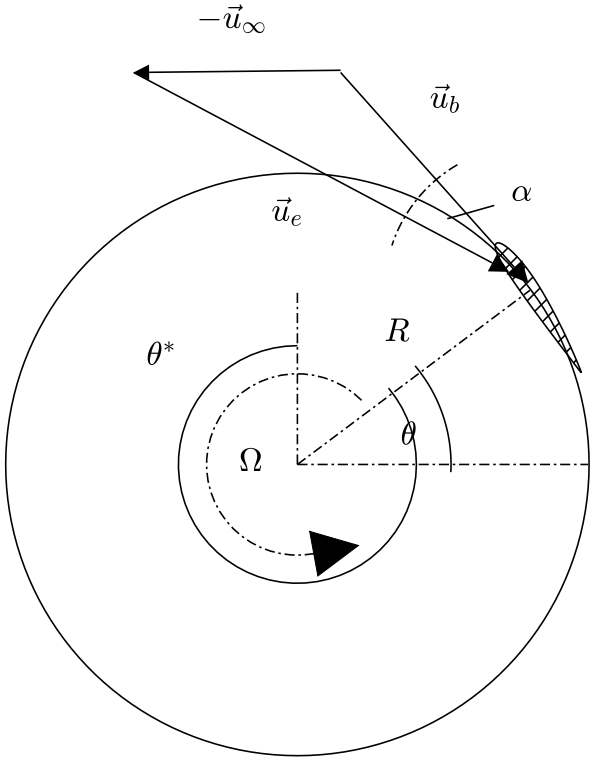


FIGURE 7.12: A schematic diagram showing the typical motion cycle of the VAWT blade. The geometric angle of attack is expressed as a function of the azimuthal angle θ of the rotor.

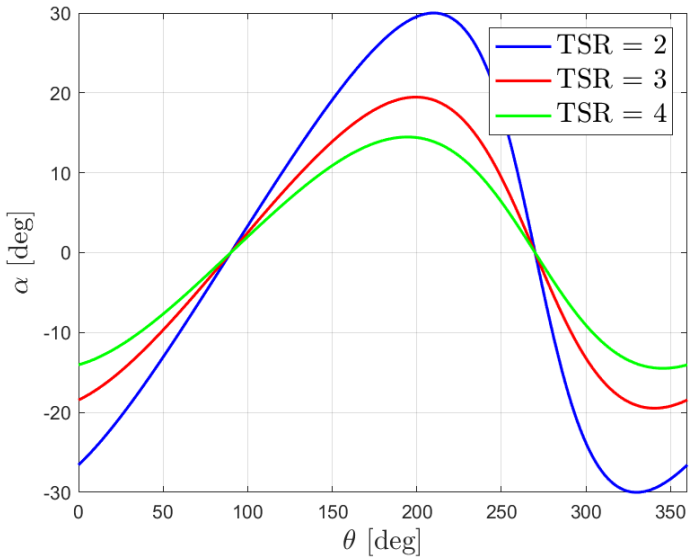


FIGURE 7.13: Variation of α as a function of the tip-speed ratio (TSR) at a zero pitched angle $\beta = 0$.

FIGURE 7.14: The vorticity contour depicting the full sequence of the aerofoil motion in the upstream part of the VAWT cycle. θ^* is the shifted azimuthal angle.

impacting on the assumption for a fully attached condition. Since the stall angle is exceeded in a relatively short azimuthal window, so that flow separation is limited

to the trailing edge region.

7.3.1 BEM validation for the impulsively started NACA0012

Before we present the result, it is important to validate the low order model in the context of aerofoil aerodynamic. This is simply done by calculating the steady solution for the NACA0012 pitched at $\alpha = 5^\circ$, which is obtained by prescribing a developed wake sheet in the formulation parallel to the free-stream. The strength of the dipole wake sheet follows from the steady Kutta condition (Eq.(3.2.36)). Figure (7.15) shows the computed pressure coefficient and is compared to XFOIL, which is a validated steady viscid- inviscid coupling solver developed by Drela (1989). The results show excellent agreement, which is not surprising given the fact that XFOIL uses the same potential formulation as in Chapter 3. The computed streamlines and the pressure field are given in Figure (7.16). For the unsteady validation, it is sufficed to apply the formulation to the unsteady version of the same problem. The only distinction is that a finite trailing wake-panel is introduced to account for the variation of the body circulation for the impulsive motion and to model the transient development of the wake sheet. Once the strength of the wake panel is solved, the panel is transformed to the vortex particles according to the algorithm outline in Section 3.3. In order to compute the pressure correctly, the unsteady potential term $(\partial\phi/\partial t)$ now appears in the Bernoulli equation, i.e.

$$\frac{p_\infty - p}{\rho} = \frac{1}{2}\|\nabla\phi\|^2 + (\vec{u}_\infty - \vec{u}_b)\cdot\nabla\phi + \frac{\partial\phi}{\partial t}. \tag{7.3.3}$$

The potential at the collocation point can be evaluated by integrating the velocity field along a path that emanates from a representative far-field point and follows the aerofoil contour. In the present work, the velocity integration uses the 5-point Gauss Legendre quadrature rule and the time derivative is approximated by an explicit Euler scheme.

Figure (7.17a) shows the pressure coefficient of the unsteady problem at 4 different non-dimensionalized times. Evidently, one can see that the pressure distribution tends asymptotically to the steady distribution. Indeed, the lift force, now time dependent, illustrates the temporal behaviour in that there appears to be a large increase of the lift follow by a more gradual trend (Figure (7.17b)). The asymptotic behaviour is well represented. The computed results match well both qualitatively and quantitatively with Vezza and Galbraith (1985a). The small discrepancy might be attributed to the fact that a frozen wake approach was used (Section 3.2.3), which likely results in the small inaccuracy of the wake shape.

7.3.2 Dynamic response of an isolated VAWT immersed in a uniform flow

To further validate the developed code, several studies have been carried out to examine the aerodynamic properties of an isolated VAWT. This is accomplished in two parts. First is the dynamic response of the blades due to a variation of the effective angle of attack. The second part examines the flow structure and is compared to smoke trailing visualization experiments. The cases considered here focused on TSR = 5, where experimental data suited for attached conditions are available. In the first test case, we considered a single-bladed Darrieus rotor whose parameters are detailed in Table 7.2. This rotor appeared in the experiment conducted by Oler et al. (1983) to measure the normal and tangential forces on the blade. The forces were obtained by placing strain gauges on the supports of the blade, which were mounted

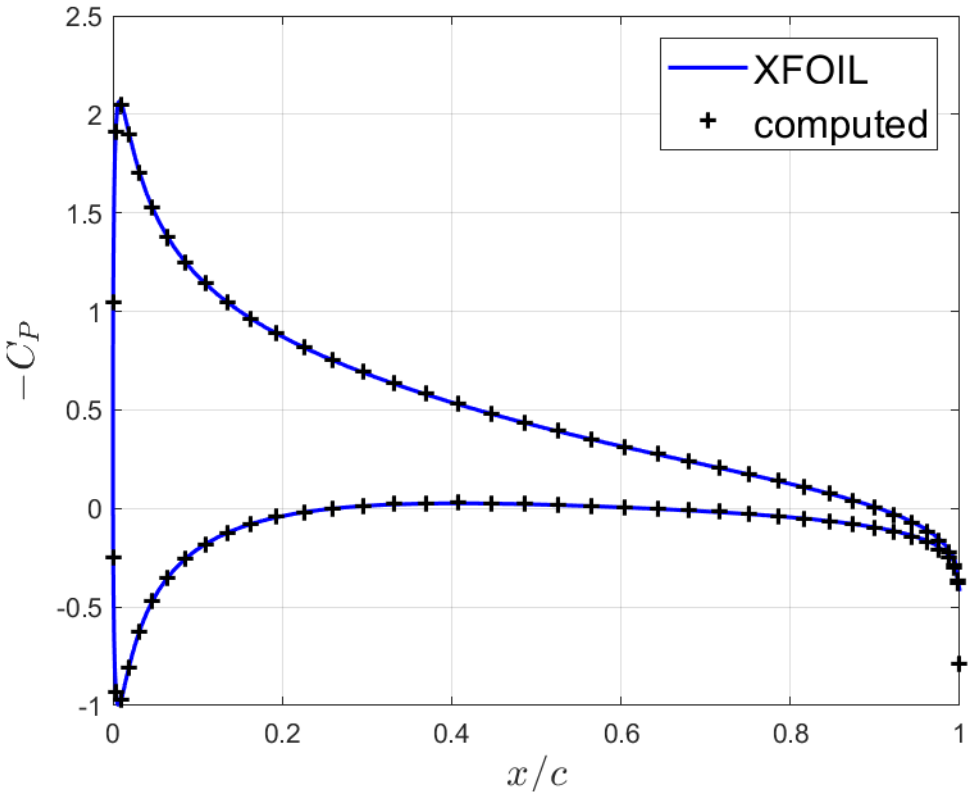


FIGURE 7.15: Comparison of the pressure coefficient C_p for the steady solution for the NACA0012 pitched at $\alpha = 5°$ between the simulated result and XFOIL.

at the mid-chord position. The strain gauges were arranged in a way that they are only sensitive to the desired forces. In the simulation, the forces per unit span are

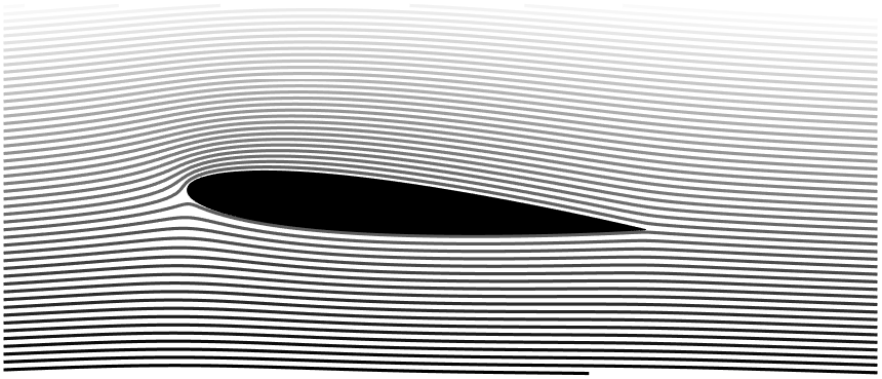
rotor parameter	value
aerofoil profile	NACA0015
chord	15.24 cm
# of blades	1
rotor radius	0.61 m
rotor height	1.1 m
TSR	5

TABLE 7.2: Darrieus rotor parameters in the experiment used by Oler et al. (1983) for the single bladed turbine

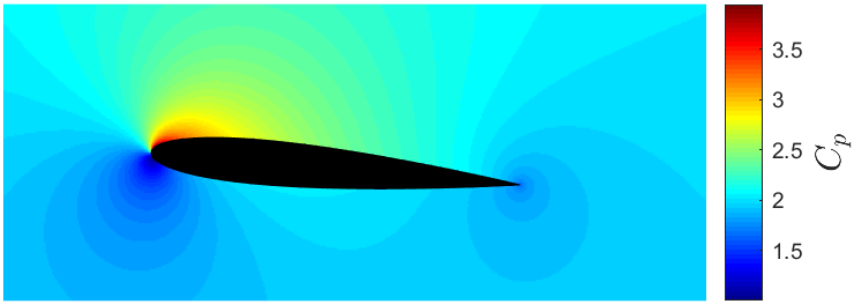
obtained by integrating the pressure distribution and projecting the x and y direction of the forces to the normal (F_n) and tangential (F_t) direction of the rotor. The forces per unit span are then normalized by the free-stream and the aerofoil chord, i.e.

$$F_n^* = \frac{2F_n}{\rho c|\vec{u}_\infty|^2}, \quad F_t^* = \frac{2F_t}{\rho c|\vec{u}_\infty|^2},$$

where the starred variables denote the normalized forces. The Reynolds number reported in the experiment was 40000. Figure (7.18) shows the computed force coefficients in the 4th cycle. It is noted that the model produces good agreement in the normal direction but the tangential force is over-predicted. This is a common feature of the inviscid model as no friction drag is explicitly incorporated into the



(A)



(B)

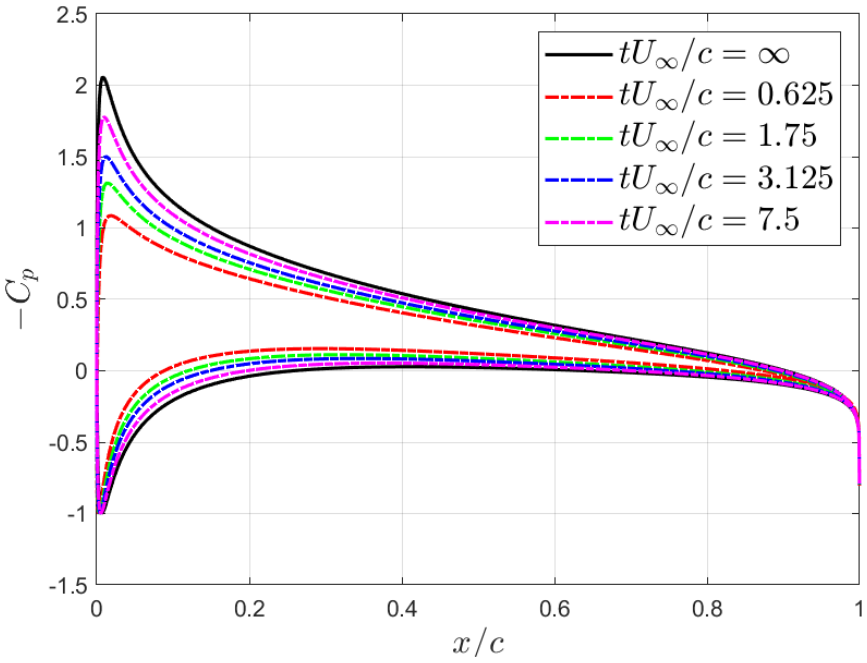
FIGURE 7.16: Computed streamline and pressure field for the NACA0012 at $\alpha = 5^{\circ}$

simulation. On a more positive note, the simulated results match quite well with the results of Deglaire (2010), who used a conformal mapping approach to map the aerofoils into circles in the complex plane and the BEM equations are solved analytically in the mapped domain. The tangential force can be corrected by solving the integral boundary layer equations (Eq.(2.3.1), Eq.(2.3.2)) and substituting the boundary layer variables to the friction drag coefficient term. It should be stressed again that the friction closure is based on data fitting with experiments in steady conditions, therefore it is questionable whether such a correction would be applicable in such a highly unsteady problem. For the purpose of this work, which is used as a preliminary design tool, such complication is deemed unnecessary.

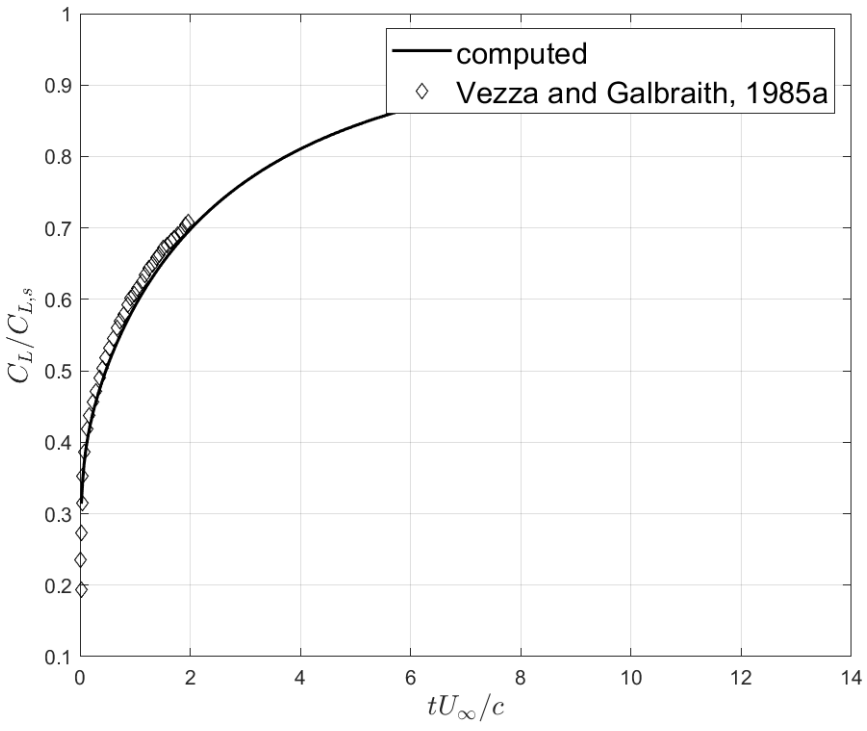
Following Deglaire (2010), a second test case for a two-bladed rotor is performed. The dynamic response from the VAWT rotor of Klimas (1982) is reported in Figure (7.19). The rotor uses the NACA0012 for the aerofoil profile and it has a solidity value of 0.30 and TSR $= 5$. The computed forces show good agreement with experimental data both in the normal and tangential directions, which is in contrast with the single-bladed case. Deglaire (2010) attributed the discrepancy observed in the single-bladed case to the relatively large chord to rotor radius ratio for which 3D effects are a significant factor. Nonetheless, the developed code manages to reproduce the results from other inviscid codes, which confirms the validity of our methodology in Chapter 3.

7.3.3 Wake structure of an isolated VAWT

The second part of this study is to visualize the development of the near-field wake structure. This is important in assessing the influence of the downstream turbines



(A)

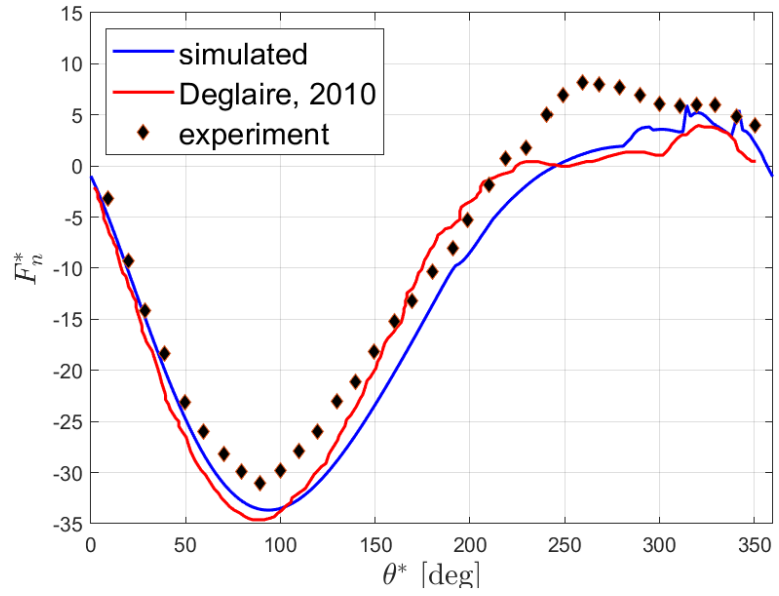


(B)

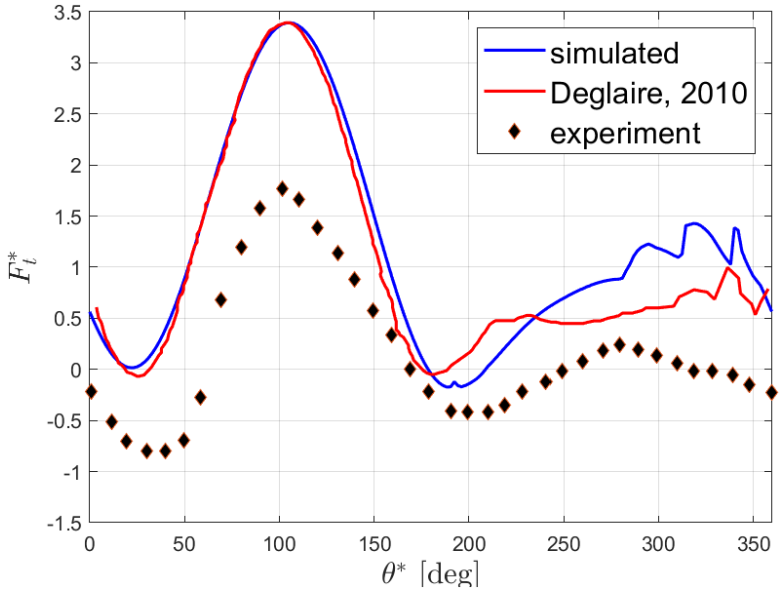
FIGURE 7.17: Evolution of the pressure coefficient (A) and the lift behaviour as a function of the non-dimensionalized time (B).

in a VAWT farm where a majority of power loss in the farm is due to the less energetic flow. Strickland et al. (1979) performed several flow visualization studies for Darrieus rotors in a one-bladed, two-bladed and three-bladed configuration. This is useful for a qualitative comparison of the near wake structure between the simulation and experiment.

Owing to the variation of the geometric angle of attack, a vortex sheet is shed



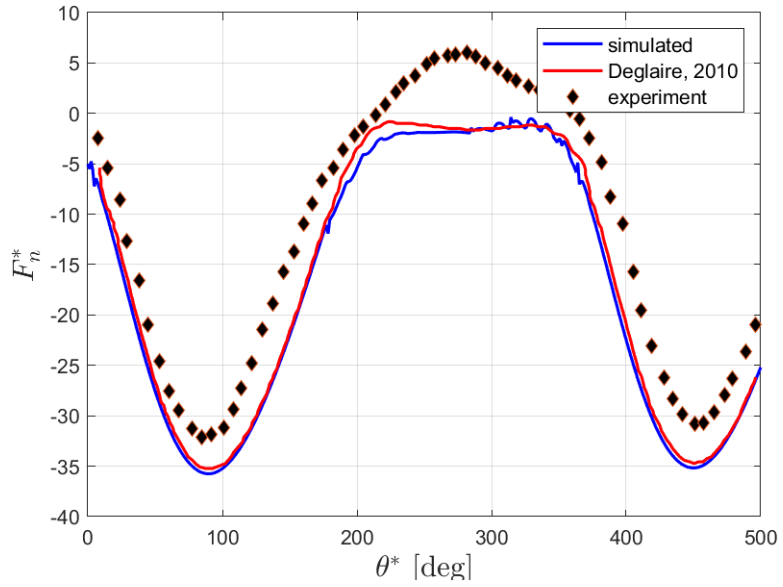
(A) F_n^*



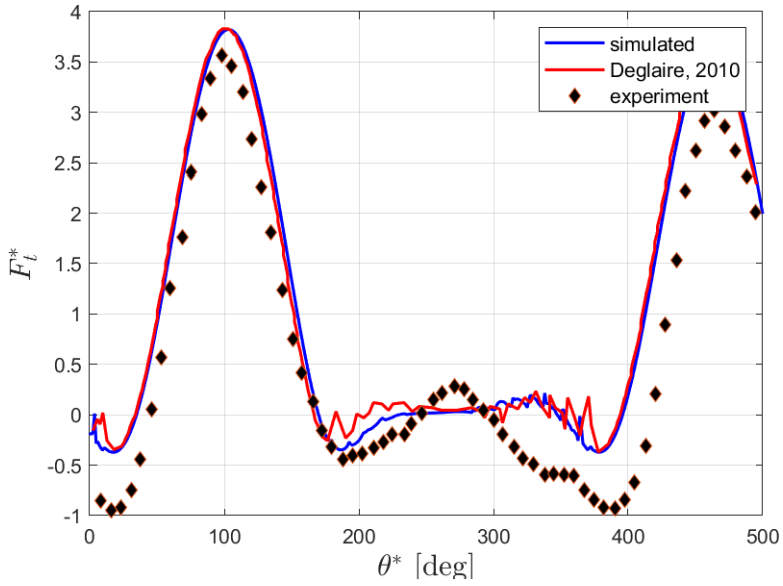
(B) F_t^*

FIGURE 7.18: Computed normal and tangential force coefficients for the singled bladed rotor. The results are compared to the inviscid solution of Deglaire (2010) and the experimental result of Oler et al. (1983).

continuously about the trailing edge. The vortex sheet convects with the local flow velocity. A general structure of the generated wake is idealised by an epicycloidal path formed by a point on the circumference of a circle that undergoes both translation and rotation (Figure (7.20)). Physically, however, due to the interference of the neighbouring vortex elements in the wake, the path becomes highly distorted and eventually gives rise to Kelvin-Helmholtz type instability, which de-regularize the vortex sheet resulting in an early transition to turbulent flow. This phenomenon is most evident in VAWTs with a large solidity value for which the added unsteadiness in the problem serves to accelerate the growth rate of the instability wave. The epicycloidal structure of the wake has two main consequences in the flow field. First



(A) F_n^*



(B) F_t^*

FIGURE 7.19: Computed normal and tangential force coefficients for the two-bladed rotor of Klimas (1982).

is the asymmetric formation of the velocity deficit about the rotor mean line in the near field, which is attributed to the non-uniform distribution of the vortex sheet in the cycle. And secondly, it is expected that blockage becomes more severe as the rotor solidity increases.

In the flow visualization experiments of Strickland et al. (1979), a NACA0012 profile was used and mounted at the quarter chord position and the chord to rotor radius ratio was kept constant (approximately 0.15) for the three configurations. The rotors were subjected to a towing speed in a water tank to produce a TSR $= 5$. The Reynolds number reported in the experiment was 40000. To illuminate the flow streak-lines, dye was injected through the nozzle at the trailing edge of *one* of the blade.

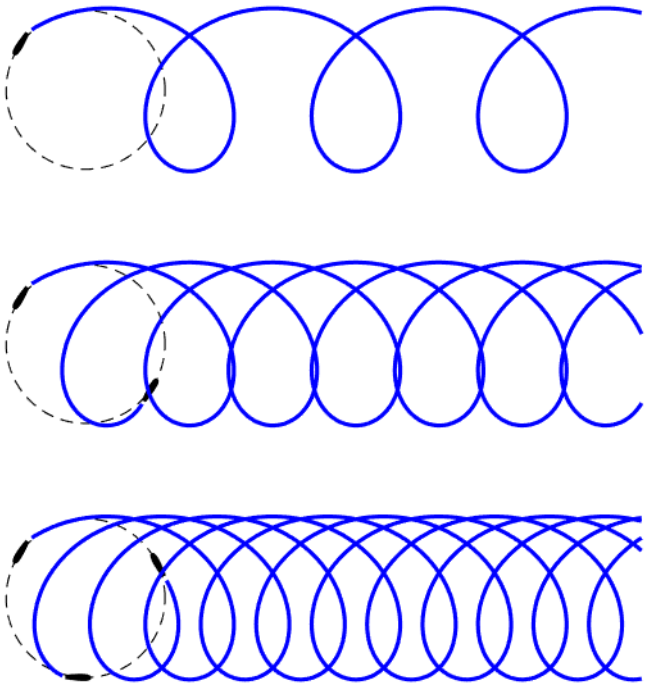


FIGURE 7.20: The idealized wake structure for the one-bladed, two-bladed and the three-bladed rotors. The idealized vortex sheet is convected by the free-stream velocity.

Figure (7.21) shows the simulated streak-lines of the wake for the one-bladed rotor at $\theta^* = 1620^\circ$. The computed result is then superimposed onto the experimental streak-lines at the same azimuthal angle. Good agreement can be observed in the near-field region. Due to the large strain in the flow, the vortex sheet near the tips of the rotor subjects to severe Kelvin-Helmholtz instabilities which result in the break-up of the vortices. Indeed, at those regions the strengths of the vortex sheet are reported to be the largest in magnitude, which is consistent with the understanding that the intensification of the vorticity is to promote instability in the flow. Figure (7.22) illustrates the variation of the vortex circulation (Γ) as a function of the shifted azimuthal angle θ^*. The large negative value at $\theta^* = 0$ corresponds to the starting vortex of the aerofoil. Tip vortices of opposite parity are subsequently shed not exactly at the tip locations as indicated by the dashed and dotted-dashed lines but at a slightly shifted forward azimuthal angle. The irregularities on the curve are the results of the blade-vortex interaction (BVI) but of much lesser magnitude.

The wake structure of a two-bladed rotor is shown in Figure (7.23). One thing to note here is that in the flow visualization, only one blade was fitted with a dye ejection mechanism whereas in fact the vortex sheet from the other blade also contributes to the formation of the wake structure as is shown in the simulated result. By matching the streak-line with one of the blade, the simulated results again give good agreement, especially in the inboard region where the convection velocity is correctly calculated.

A long time simulation has been performed for the two-bladed rotor. Figure (7.24) shows the vortex markers after 20 revolutions. At this point, the near-wake region can be regarded as already having reached steady state, however, a clear transition point is observed, beyond which the vortices organise themselves in a way that resembles the von-Karma vortex street. The length of the steady region might

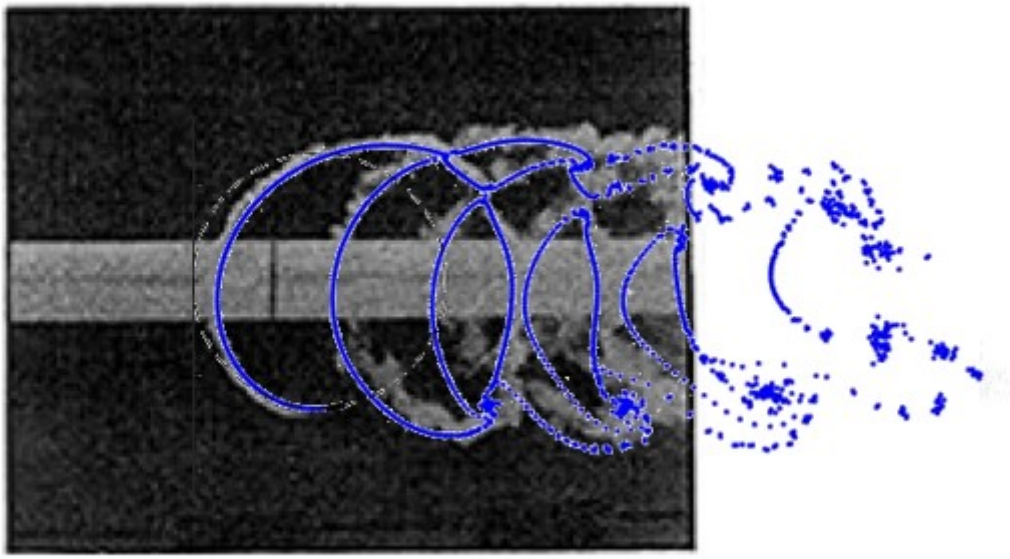


FIGURE 7.21: The computed streak-line of the wake (represented by blue markers) is superimposed onto the experimental streak-line for the one-bladed rotor.

be postulated as a function of the tip-speed ratio and the rotor solidity. Similar simulation with higher solidity value at the same tip-speed ratio (TSR $= 5$) results in a much shorter region. Moreover, the shorter steady region is typically accompanied with increasing severity of the blockage effect in the near-wake. For the three rotors examined in this section, the solidity values are 0.15, 0.30 and 0.45 respectively. Figure (7.25) shows the u-velocity contour after the 4th cycle normalized by the free-stream. A general trend is that a large extent of stagnant flow prevails much of the near-field region at high solidity whilst intensifying the shear layers that emanates from the tips. Interestingly, by taking the solidity to infinity, the flow field can be seen to approach that of a circular cylindrical flow.

7.4 Performance and aerodynamic analysis of a pair of VAWTs

Having validated the developed model for an isolated VAWT, the code is applied to simulate the effect of having multiple VAWTs arranged in some configuration. This is important in designing farm configuration for maximizing the power extraction of the wind-farm. Due to the limited experimental and simulation data, this study serves as a preliminary study into the complicated flow field inside the VAWT wind-farm.

7.4.1 Classification of rotor placement

As a starting point for the simulation of a VAWT wind-farm, studies have been carried out to examine the dynamic response and wake structure for a pair of VAWTs in four configurations. All of the rotors share the same parameters except for the rotation orientation. The normalized rotor parameters can be found in Table 7.3. It should be noted that the rotor used in these simulations is geometrically similar to the ones simulated in Section 7.3.3 except the TSR has been reduced to 4. The reason for selecting a slightly lower TSR is to reduce the unsteadiness in the problem as it

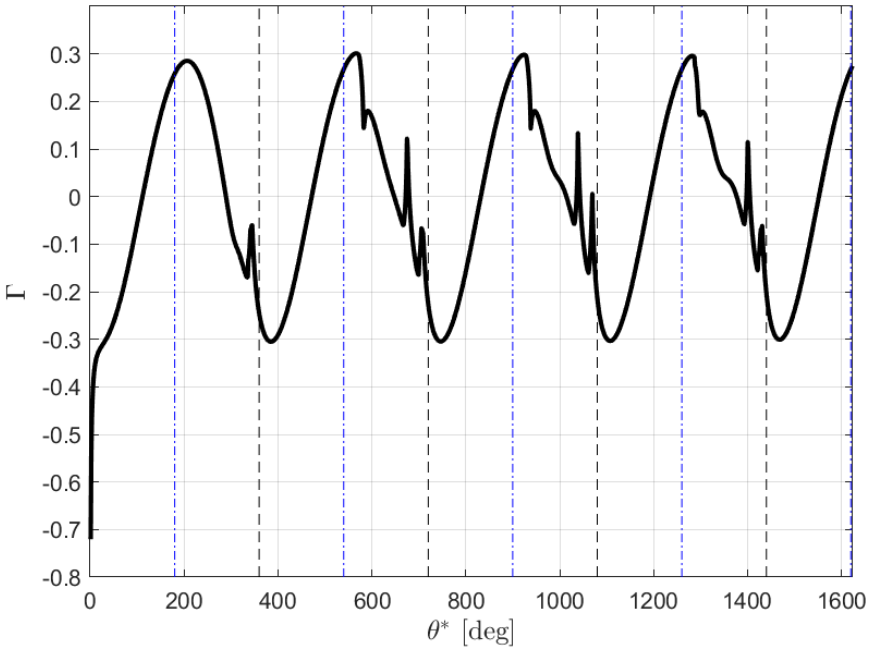


FIGURE 7.22: Shed vortex circulation strength as a function of the shifted azimuthal angle for the one-bladed rotor. Dashed and dotted-dashed lines indicate the tip location at the advancing and receding side of the rotor, respectively.

would be the case in the real environment since the added unsteadiness may amplify the unsteady load on the blade which could ultimately lead to the degradation of the structural integrity of the rotor. One should note that at this TSR, Eq.(7.3.2) indicates that at some points on the azimuthal cycle the blades may experience a higher than static stall angle of attack. It is anticipated that this would not violate the validity of the attached assumption too much given the fact that the violation often occurs in a relative small azimuthal window so that flow separation is thought to be limited to the trailing edge region.

rotor parameter	value
aerofoil profile	NACA0012
# of blades	2
chord	1
solidity	0.30
TSR	4

TABLE 7.3: Rotor parameters used in the interaction of a VAWT pair.

In addition to the 4 configurations of the VAWT pair, each configuration can also vary by a set number of parameters. It is instructive to adopt a naming convention for the various runs. The run ID is given by the designation CxRy, where x and y denote the configuration number and run number within the respective configuration, respectively. The general characteristic of the 4 configurations can be summarised as follows:

- C1: The two VAWTs are aligned in the y-direction with a co-rotating motion. This configuration is parametrized by the normalized separation distance between the rotor centres.

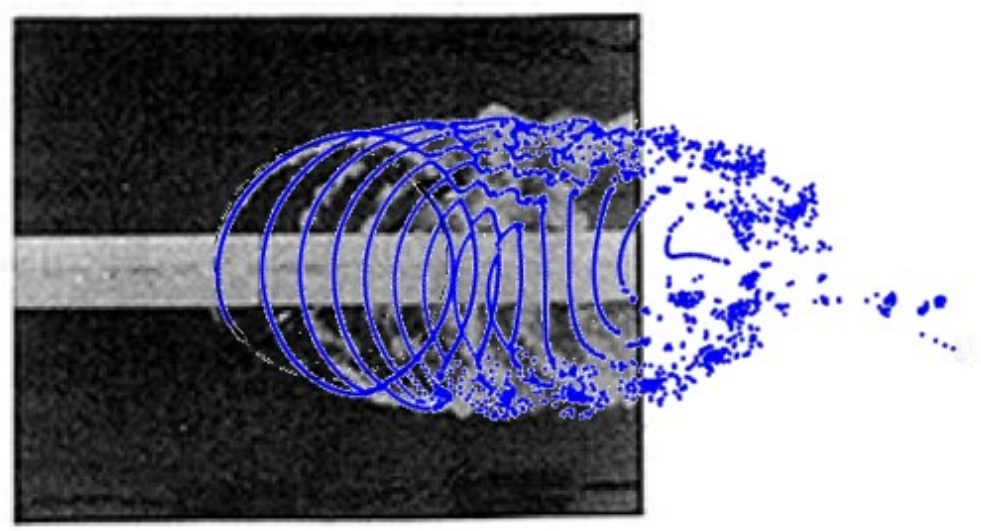


FIGURE 7.23: Comparison of the streak-lines between simulation and experiment for a two-bladed rotor.

- C2: Same as the C1 but with a counter-rotating motion.
- C3: The VAWTs are aligned in the x-direction and are parametrized by the relative rotation orientation (co-rotating or counter-rotating) and the normalized distance between the rotor centres.
- C4: The VAWTs are arranged in a staggered grid and are parametrized by the relative x and y displacements of the rotor centres as well as the relative rotation orientation.

It should be noted that $C1, C2$ and $C3$ can be regarded as special cases of $C4$. In any case, the classifying of the configurations outlined above completely characterises the pairwise turbine placement and the operation characteristic in a VAWT windfarm. The schematic arrangements of the various configurations can be found in Figure (7.26) and Figure (7.27). The aim of the C1 and C2 configurations is to examine the performance magnification owing to the increased velocity surrounding the rotors (due to blockage). While the C3 configuration is aimed at studying the power degradation of the downstream turbine as a result of operating in the wake of the primary rotor thus subjecting to a much retarded flow. And finally the C4 combines the effect of magnification and degradation by operating the downstream turbine partly in the wake of the first.

To measure the performance of the rotor, one defines the torque (τ) as

$$\tau = R \sum_{k=1}^{B} \frac{1}{2} \rho \left\| \vec{u}_\infty \right\|^2 c F_{t,k}^*. \tag{7.4.1}$$

where B denotes the number of blades and $F_{t,k}^*$ is the tangential force coefficient of the k-th blade. The power coefficient (C_P) is derived as follows (Hansen, 2008):

$$C_P = \frac{\tau \Omega}{1/2 \rho \left\| \vec{u}_\infty \right\|^3 D h_z}. \tag{7.4.2}$$

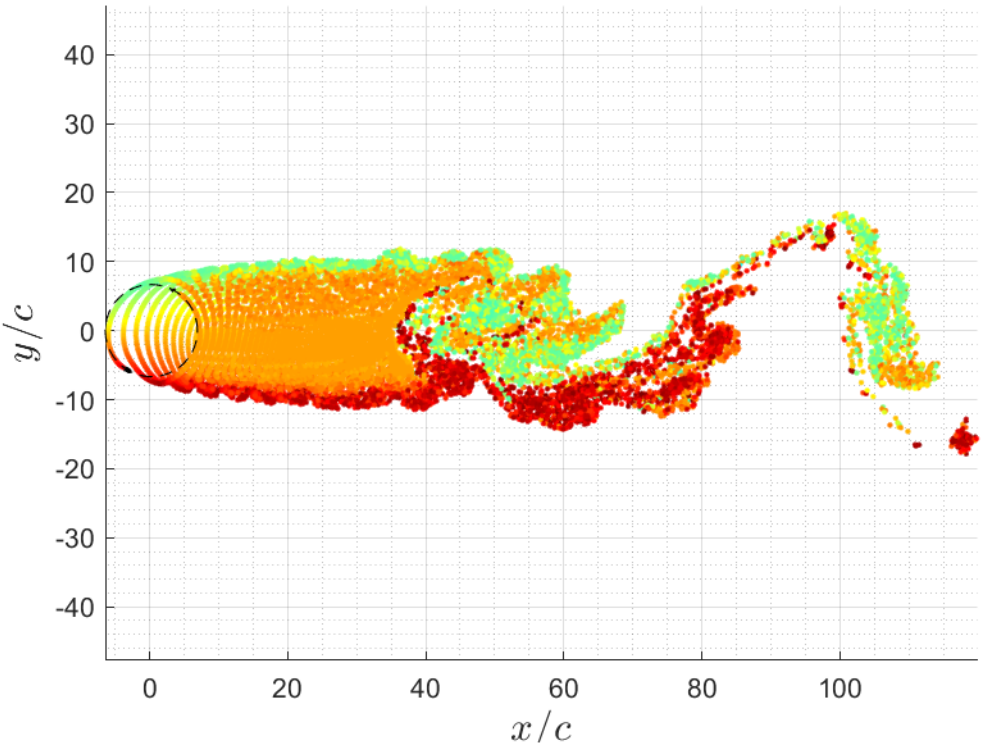


FIGURE 7.24: Wake markers for the two-bladed rotor after 20 revolutions.

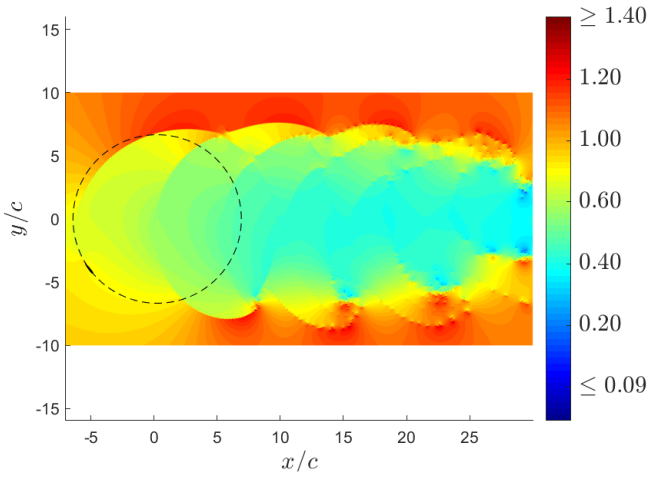
where h_z is the blade-span. As Brusca et al. (2014) noted, the performance of the VAWT rotor is sensitive to the prescription of the aspect ratio. Without loss of generality, we take h_z to be the diameter of the rotor. A benchmark case was first performed, in which a single rotor was simulated and the power coefficients were obtained as a function of the non-dimensionalized time $T = t \left\| \vec{u}_\infty \right\| / c$. The benchmark power coefficient is shown in Figure (7.28). The cycle average ($\bar{C}_P$) is also given, which is obtained by applying the trapezoidal rule to the power coefficient and averaging over time, i.e.

$$\bar{C}_P = \frac{1}{2\pi} \int_0^{2\pi} C_P d\theta.$$

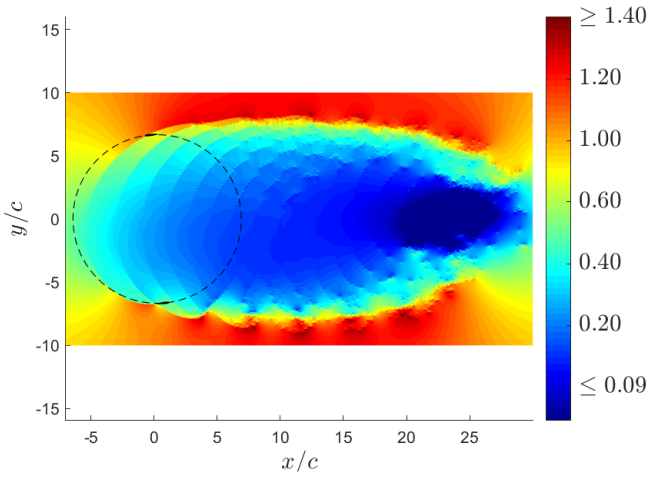
The parameters for these runs can be found in Table 7.4. The relative rotation orientation of the rotors is given by the parameter $S := \text{sign}(\Omega_1)\,\text{sign}(\Omega_2)$. Furthermore, to understand the influence of the rotors on each other, a relative power coefficient ($\bar{C}_{P,\text{relative}}$) may be defined as follows:

$$\bar{C}_{P,\text{relative}} := \frac{\bar{C}_P}{\bar{C}_{P,0}}, \tag{7.4.3}$$

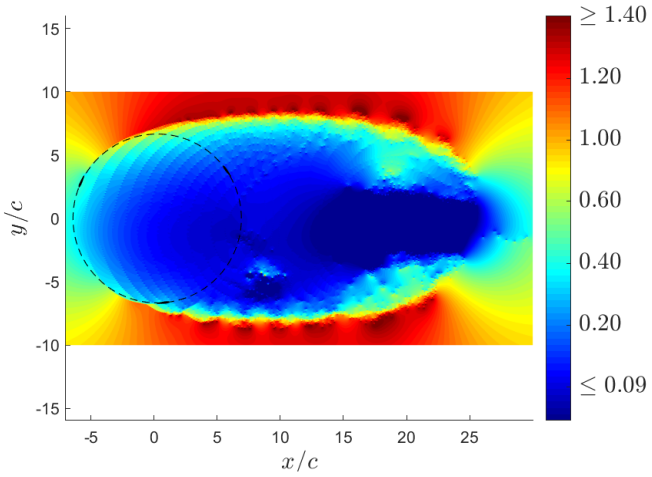
where $\bar{C}_{P,0}$ denotes the time-averaged power coefficient of the benchmark case. Each simulations were run for 20 cycles with a time-step size of $\delta t = 5.5 \times 10^{-3}$. This time-step size was used in the validation study in Section 7.3.2. The 20th cycle was chosen because the averaged power coefficient of the benchmark case seems to have converged during the initial parametric study (not documented). In addition, the term *primary* is used throughout the section, which refers to the rotor whose centre



(A) $\sigma = 0.15, B = 1$



(B) $\sigma = 0.30, B = 2$



(C) $\sigma = 0.45, B = 3$

FIGURE 7.25: Normalized u-velocity contour for the three rotors with increasing solidity (σ) from top to bottom.

of rotation coincides with the origin.

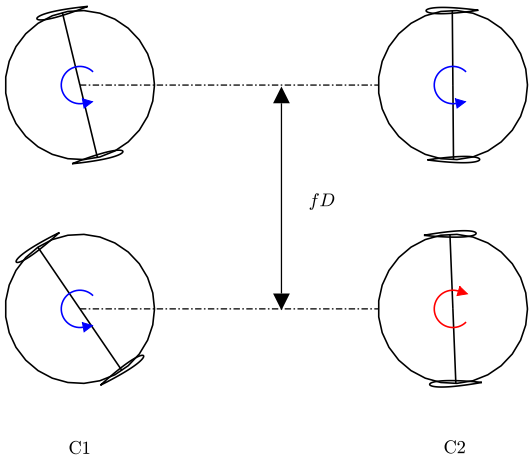


FIGURE 7.26: Schematic diagrams showing the VAWT placements of the C1 and C2 configurations. C1 and C2 are both parametrized by the dimensionless constant $f > 1$. Here D denotes the diameter of the rotor. The free-stream is coming from the negative x-direction.

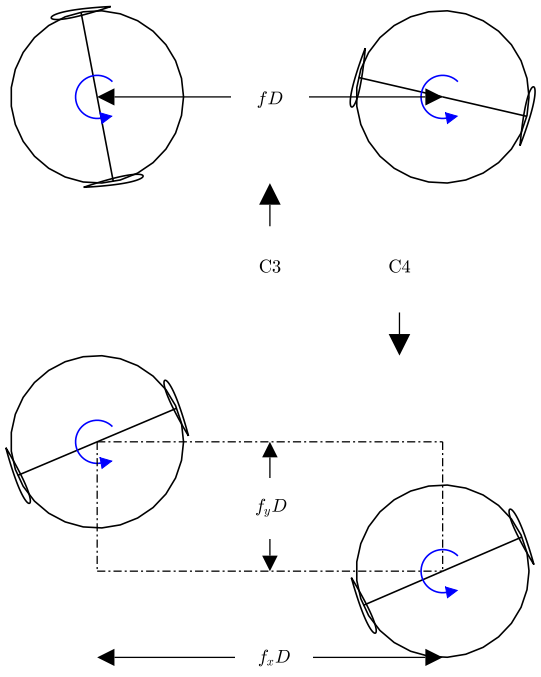


FIGURE 7.27: Schematic diagram showing the VAWT placements of the C3 and C4 configurations.

7.4.2 C1 simulations

For the C1 configuration, two simulations were performed with $f = 1.2$ and $f = 2.0$ (see also Table 7.4). The relative power coefficients are reported in Figure (7.29b). The result shows an improvement to the primary rotor as to be expected. Using the

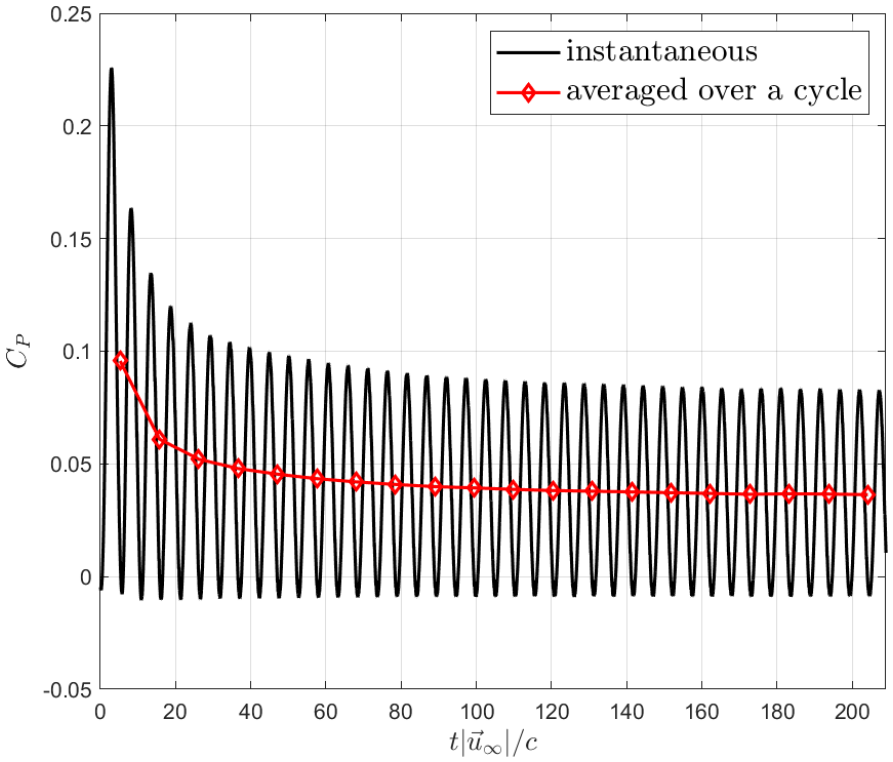


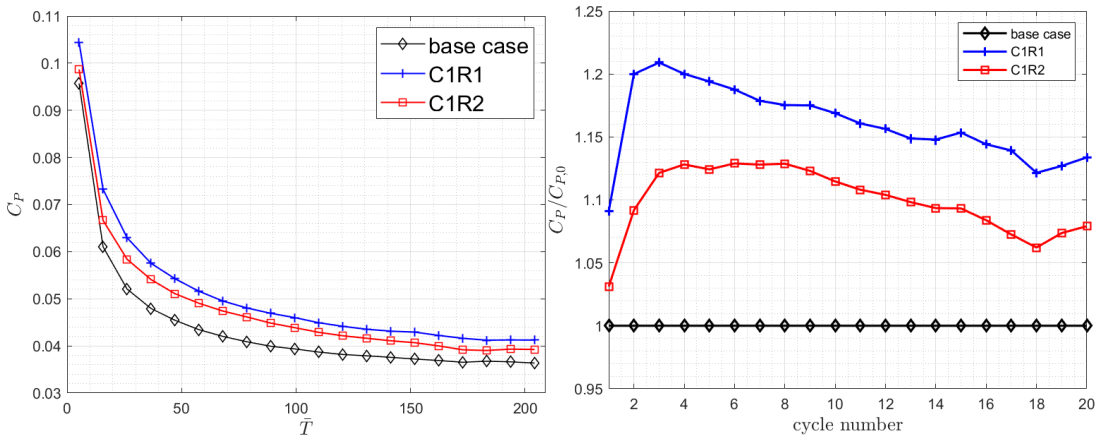
FIGURE 7.28: Computed power coefficient for the benchmark rotor.

20th revolution as an indicative measure, the primary rotor was found to achieve a 13.37% gain averaged over the 20th cycle. Moving the secondary rotor further in the lateral direction ($f = 2.0$) yields in a moderate gain of 8%, which is consistent with the understanding that $\bar{C}_{P,\text{relative}} \to 1$ as $f \to \infty$. A parametric model might be constructed to determine the performance gain in terms of the dimensionless parameters, which would be an interesting topic for future research.

run ID	f	f_x	f_y	S
C1R1	1.2	/	/	$+1$
C1R2	2.0	/	/	$+1$
C2R1	1.2	/	/	-1
C2R2	2.0	/	/	-1
C3R1	2.0	/	/	$+1$
C4R1	/	2.0	0.5	$+1$
C4R1	/	2.0	1.3	$+1$

TABLE 7.4: Parameter matrix for the four configurations. The nomenclature can be found in Figure (7.26) and Figure (7.27).

The wake structure of the co-rotating turbines is given in Figure (7.30). In the C1R1, it is surprising that the wake of the two rotors maintained their identity up to the transition point, beyond which the vortices in the wake interact strongly to form a large scale fluctuation. In the C1R2, the wake of the two rotors started out identically, but due to the mutual interaction, the wakes evolve to a considerable degree of asymmetric both in the near and far field. Beyond the transition point, a von-Karma street is clearly visible.



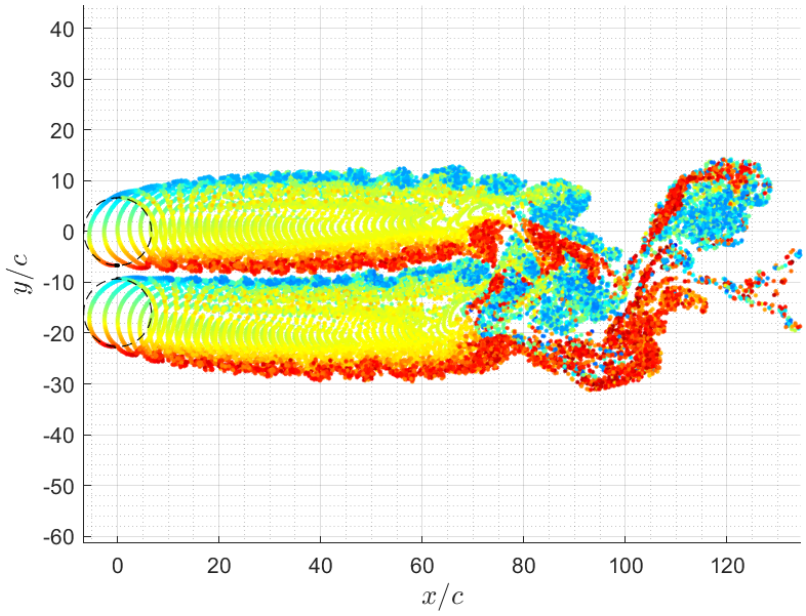
(A) (B)

FIGURE 7.29: (A) Shows the power coefficients between the benchmark case and C1R1 and C1R2. (B) Indicates the relative coefficients computed for each revolution.

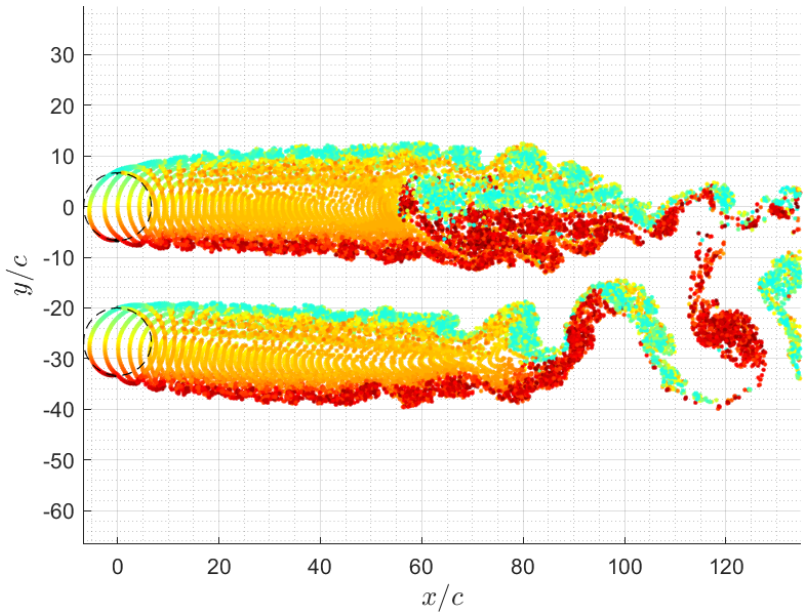
7.4.3 C2 simulations

For the C2 configuration, the same values of f were used, except this time the secondary rotor reverses its rotation direction. The resulting situation is analogous to a wall-bounded flow in which an imaginary wall can be thought to be placed at the symmetric plane of the rotor centres. Therefore one should expect some degree of symmetry in the wake. The simulated results for the C2 runs are found in Figure (7.31). A general trend here is that the symmetric property of the wake is largely respected in the steady region except near the transition region in which the numerical error in the code serves to trigger the large-scale mixing between the wake vortices. Moreover, owing to the interference of each other, the shear layer at the receding side is severely suppressed which results in an almost flat layer.

The power output is collated and compared to the benchmark case as well as to the C1 configuration. Figure (7.32) paints a rather interesting picture. All of the cases considered so far yield improvement to the primary rotor as a result of the increased velocity near the symmetric plane. However, at a small f, the co-rotating configuration generally produces high yield as compared to the counter-rotating configuration across all cycles. This trend somehow does not carry over to the case when $f = 2$. Indeed, Figure (7.32b) shows that the co-rotating configuration yields a slightly better performance up to the third cycle. But as soon as the wake starts to develop, the counter-rotating configuration is able to provide a much higher gain than the co-rotating configuration. After the 20th cycle, it is found that a gain of 10.46% and 9.4% was observed for the C2R1 and C2R2 simulation, respectively. The similarity between the two numbers is counter-intuitive as one should expect that C2R1 should perform better since the rotors are subjected to a higher blockage at the receding side. A possible explanation is that the blockage is not as large as one might imagine near the rotor boundary due to the relative small solidity. The normalized u-velocity contour is shown in Figure (7.33), which appears to support this observation.



(A) Wake markers after the 20th cycle



(B) Wake markers after the 20th cycle

FIGURE 7.30: Computed wake structure for the C1R1 (A) and C1R2 (B).

7.4.4 C3 simulation

Only one simulation is performed for the C3 configuration. The aim is to examine the detrimental effect on power production on the downstream rotor if it is placed in the steady region of the upstream rotor. Both the tangential and normal forces were examined for the blade on the downstream turbine. Visualization of the wake markers at the end of the simulation is found in Figure (7.34). Several notable features can be extracted. First is that the wake, as a collective entity, shows a greater lateral expansion possibly due to the limited power production still taking place by the advancing blades (corresponding to the positive tangential force in Figure (7.35b)).

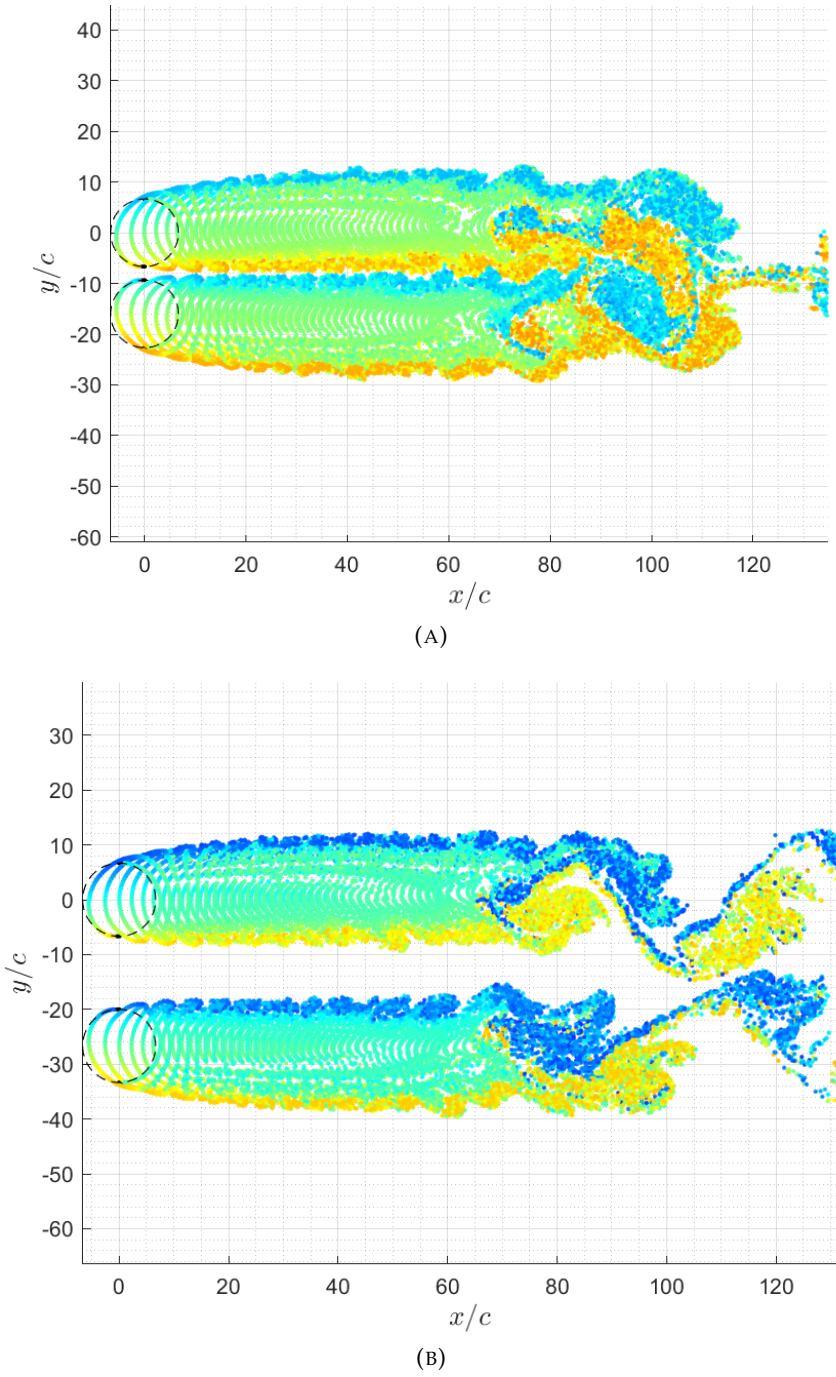
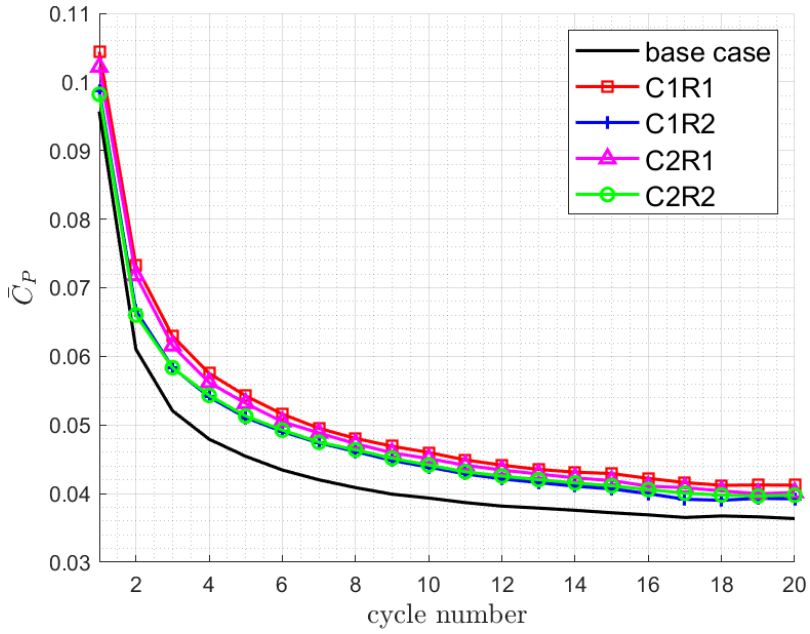


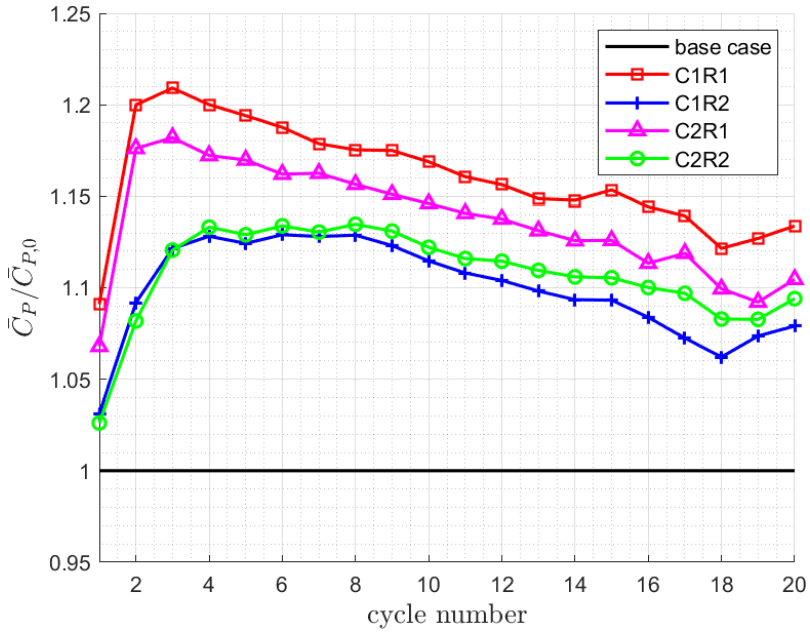
FIGURE 7.31: Computed wake structure for the C2R1 (A) and C2R2 (B) simulation. Symmetry is maintained in the near field , but a symmetry breaking event is observed due to numerical error in the simulations, which is amplified by the large number of time steps used.

Secondly, the transition region has been reduced considerably. The transition point occurs just slightly downstream of the secondary rotor. The generated torque of the advancing downstream blade on the secondary rotor shows a similar magnitude to the upstream rotor operating in the wake region.

On the downstream rotor, the force has converged after the 4th cycle. Furthermore, it is found that the downstream rotor could not extract any power since the averaged power coefficients were found to be negative. The negative contribution is



(A)



(B)

FIGURE 7.32: (A) Shows the power coefficients between the benchmark case and C1 and C2. (B) Indicates the relative coefficients for all of the C1 and C2 simulations.

primarily due to the blades operating in the downstream part of the cycle. The overall conclusion is that while the advancing blade is still able to extract some power as indicated by the positive tangential force, but this is quickly offset by the other blade. It is expected that a net positive torque might be achieved by varying the number of blades, tip-speed ratio and solidity. A consequence arises from this is that, due to its presence, there appears to be an improvement to the wake recovery, which is beneficial to the downstream rotors in a real VAWT wind-farm. In this view, the secondary rotor can be treated as a vortex generator whose role is to provide a means

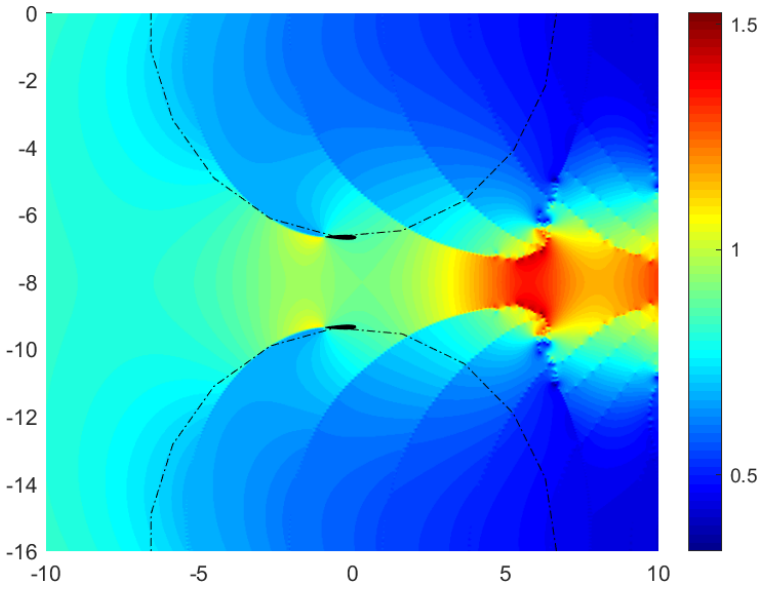


FIGURE 7.33: Normalized u-velocity contour at the symmetric plane for the C2R1 simulation.

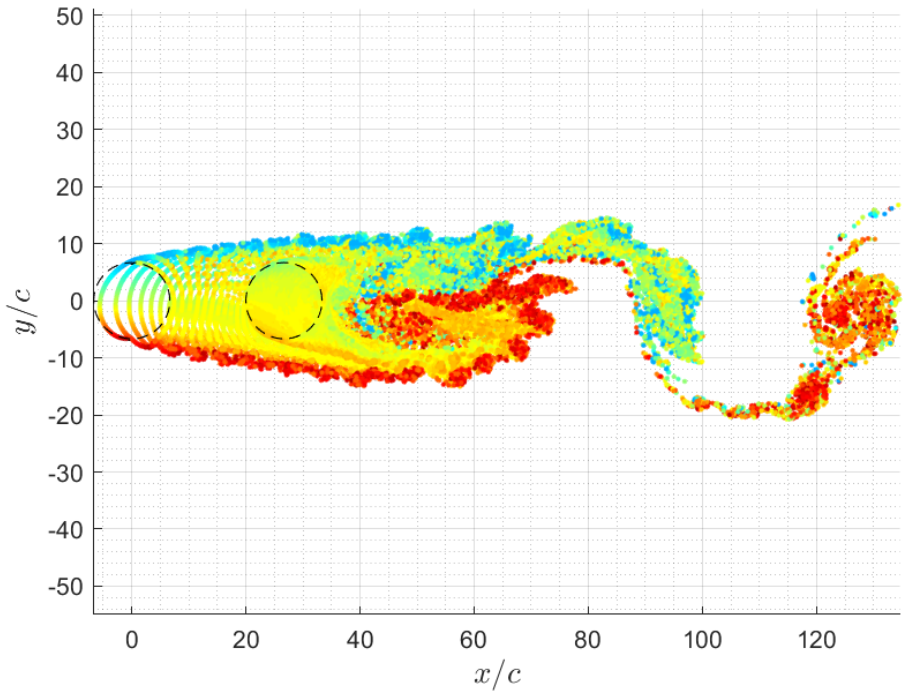
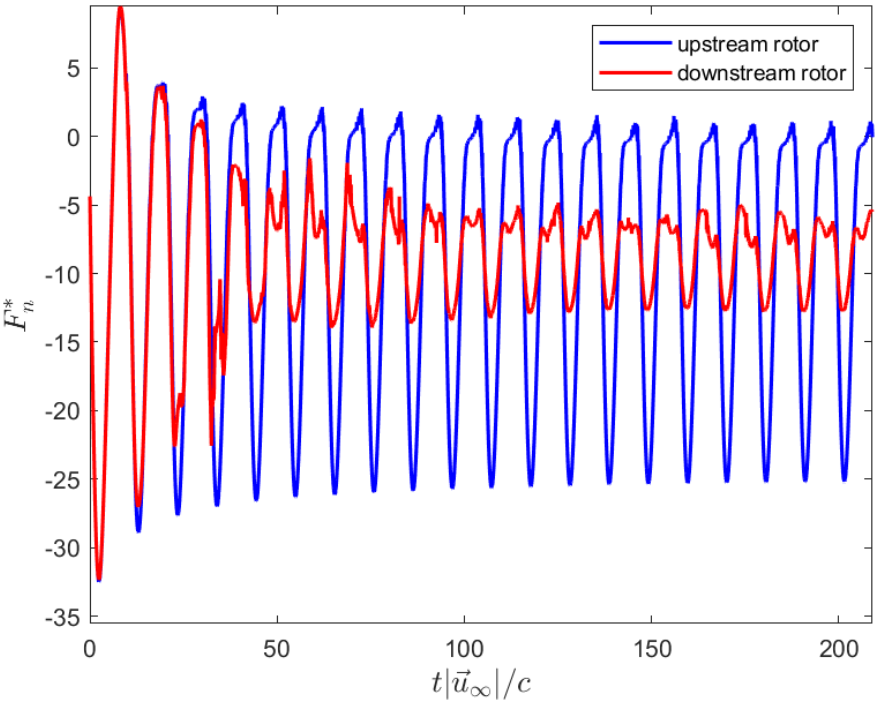
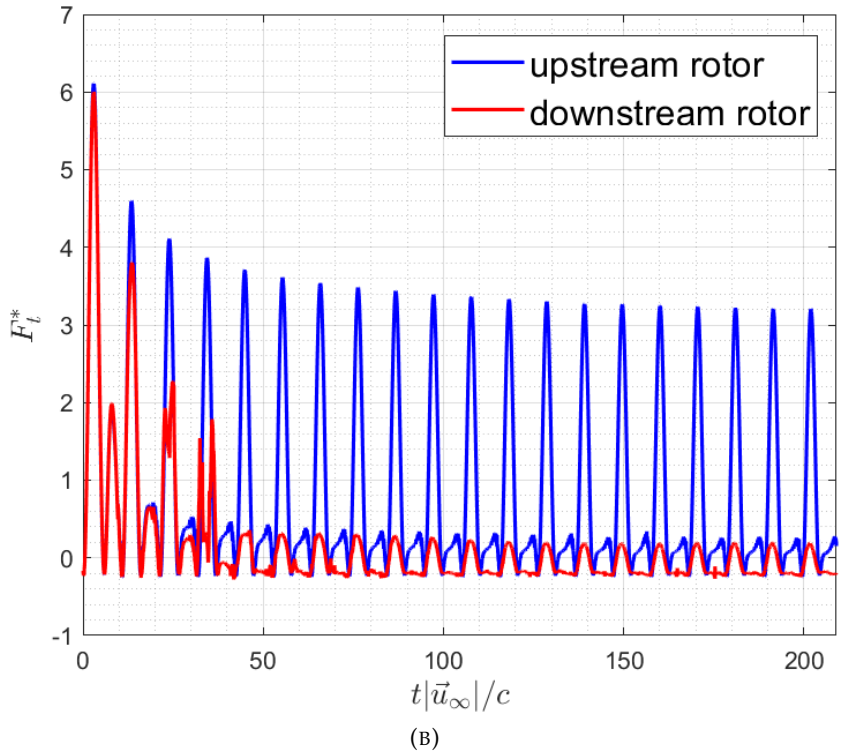


FIGURE 7.34: Computed wake structure for the C3 simulation.

for achieving a faster wake recovery so that the more efficient downstream turbines might benefit from it. While at the same time, the rotor itself would be able to produce a net positive torque. In the simulation, the said rotor clearly could not fulfil such a role. One possible research direction is to determine the rotor parameters and performance characteristics so as to optimise the operation in the wake region if such strategy were to be implemented.



(A)



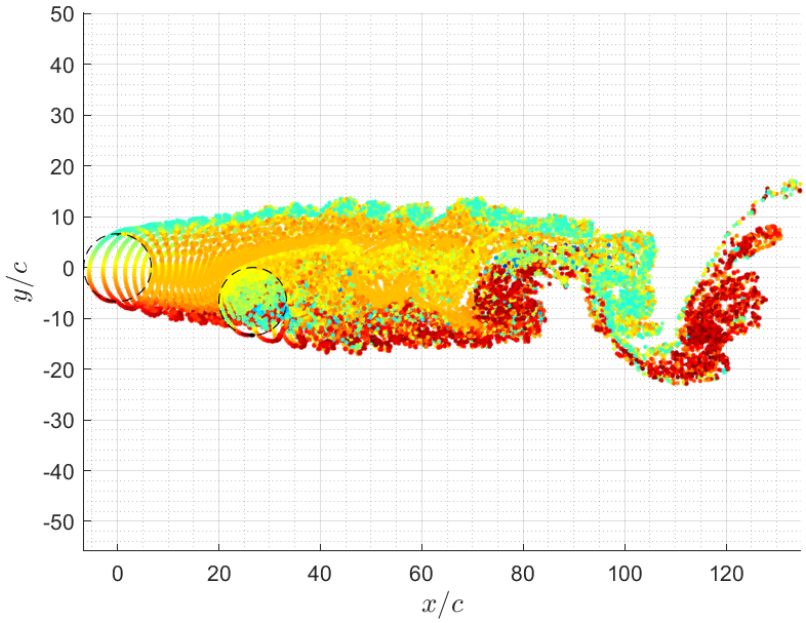
(B)

FIGURE 7.35: Computed normal and tangential coefficients for the upstream and downstream rotor for the C3R1 simulation.

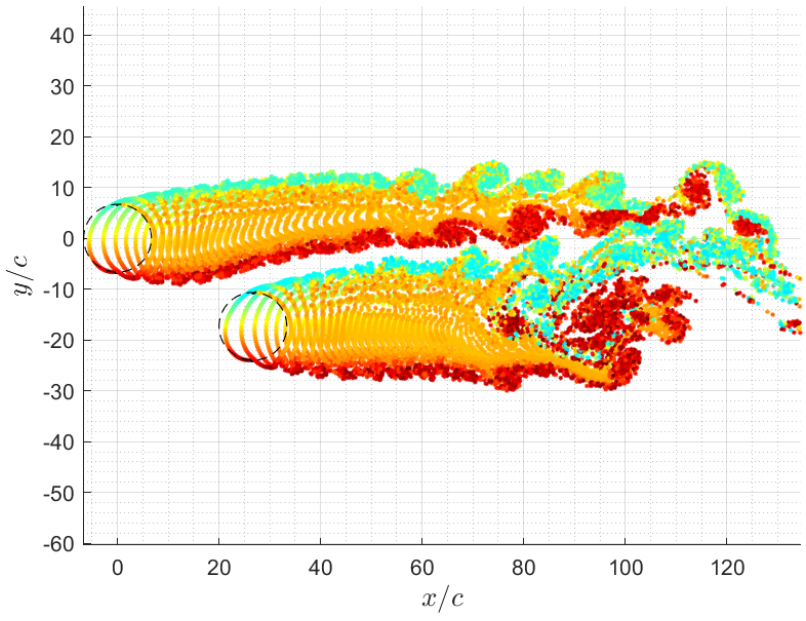
7.4.5 C4 simulations

For the final configuration (C4), two simulations were performed with separation displacement $(f_x, f_y) = (2, 0.5), (2, 1.3)$. In the former case, the first and second quadrants of the secondary rotor expose to the wake of the first (idealised by a shear

line from the tip), whereas in the latter, the secondary rotor is fully subjected to the high velocity stream-tube. For an isolated VAWT pair, the latter configuration is clearly superior. However, in a large VAWT farm, the former might be preferred since the secondary rotor might be used to promote momentum transfer between the energetic and retarded flow; leading to a more efficient wake recovery mechanism.



(A)



(B)

FIGURE 7.36: The computed wake structure for the C4R1 (A) and C4R2 (B) simulation after the 20th revolution.

The normal and the tangential force coefficients can be found in Figure (7.37). It is noted that the downstream rotor subjects to severe unsteadiness in the flow especially in the inboard region where strong vortices from the upstream shear layer mix with the retarded flow, resulting in a significant fluctuation of the force vector. No significant degradation was observed for the tangential coefficient in the upstream

cycle, however, power degradation is observed primarily due to the negative torque induced by the second blade operating in the downstream cycle. The wake structure is given in Figure (7.36a). One should mention that the large fluctuation of the force vector plays a detrimental role to the structural property of the rotor owing to the higher BVI rate and with stronger vortices.

The case $f_y = 1.3$ results in a completely different flow structure as observed in the C4R1 case. The lateral expansion of the upstream wake is significantly suppressed as a result of the mutual interaction. A clear passage is formed between the shear layers of the primary and secondary rotors. The passage maintains its identity until the wakes have sufficiently mixed further downstream. The relative power coefficients for the downstream rotor have been obtained for both cases (Figure (7.39)). The result shows that the maximum power output is almost halved in the C4R1 simulation as compared to C4R2. However, on average, the rotor is still able to generate positive power, albeit at a significantly reduced efficiency. Indeed, Figure (7.39a) and Figure (7.40b) show a reduction of 74.31% can be expected using the 20th cycle as an indicative mark. Although at this point the force has clearly not converged. On the other hand, by placing the downstream rotor slight downward in the lateral direction, a favourable gain of 20.63% in efficiency is obtained.

A different perspective one can take is that the 4 configurations examined in this section represent the inflow wind direction past a pair of static VAWTs (Figure (7.41a)). The simulated results provided clues to the indicative loss and gain in operating under such conditions. To make this concise, one can define the wind-incident angle (θ_w) to be the acute angle between the wind-direction and the rotor centre orientation (see Figure (7.41b)). The effect of the different inflow conditions can then be summarised in Table 7.5 for the cases that the separation distance between the rotors is at least twice the rotor diameter. When $\theta_w = 0$, the downstream rotor operates in the wake of the first, simulated results show that the tested rotor could not generate any torque. As the incident angle increases, the power efficiency is rapidly increasing to a theoretical maximum value beyond which the efficiency starts to decrease again until it is perpendicular to the centre orientation (C1 and C2), which might be speculated as a local minimum. Beyond which we suspect that the trend follows in a similar manner except the positions of the rotors have reversed.

θ_w [deg]	$\bar{C}_{P,\text{relative}} - 1$
0.00°	$\leq -100\%$
14.04°	-74.31%
33.02°	$+20.63\%$
90.00°	$+8.00\%$

TABLE 7.5: Indicative percentage loss and gain for the downstream rotor due to the different wind direction characterised by the wind incident angle θ_w.

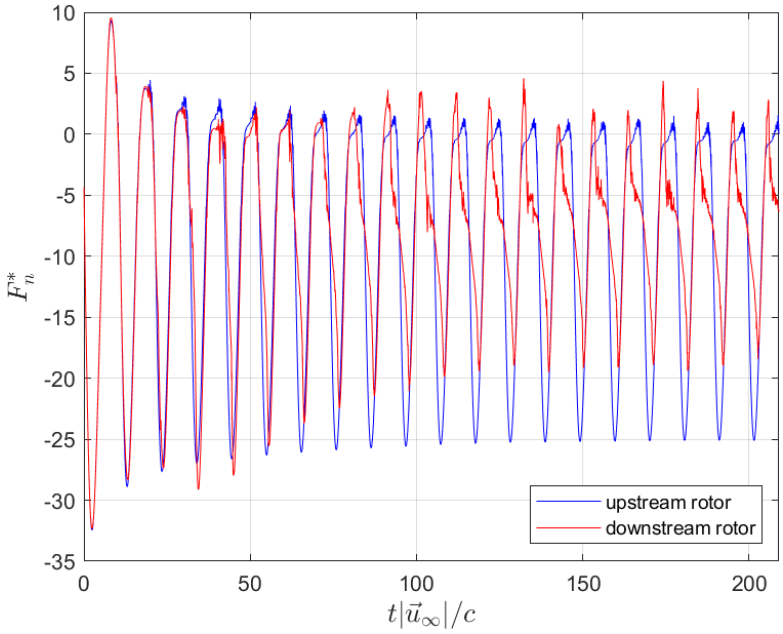
7.5 Conclusion

The new developed methodology has been applied in the simulation of aerofoil aerodynamic with high resolution. Qualitative comparisons with experimental data have been made for a static NACA0012. The new reported results show excellent agreement with PIV visualization. Both the vortex shedding frequency and the topological properties of the vortex structure are well simulated. In addition, the lift and

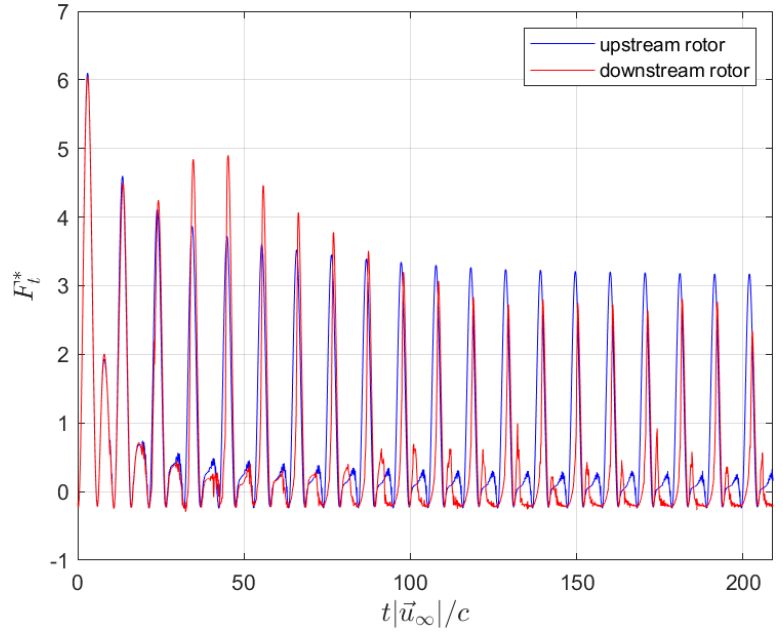
drag data have been obtained and match with the theoretical prediction at early time.

The 2D adaptation of the BEM has been successfully validated by simulating the dynamic response and the wake structures of the VAWT rotors in different configurations. The results show good agreement mostly in the normal direction. However, some discrepancies are seen for the tangential calculation. It is speculated that the discrepancy is due to the lack of viscosity modelling in the model. A possible improvement has been suggested, which is to couple the boundary layer equations to determine the friction drag coefficient. It was cautioned that care should be exercised in order to make sure that the unsteady effect is correctly represented.

The method is then applied to the simulation of the interaction between a pair of VAWTs, which serves to provide an indicative study of the complicated aerodynamic in a fully VAWT wind-farm. The rotors were derived from Strickland et al. (1979). The study points to a rich set of wake dynamic that depends on the wind-incident angle. It is found that when a downstream rotor is carefully placed, wake collision between the upstream and downstream rotors can be avoided. Furthermore, owing to the increased velocity, the downstream rotor shows a significant performance gain of more than 20%. However, at a zero incident angle, the downstream rotor was found to produce negative power but due to its presence, the transition point has been significantly reduced, which may point to a strategy that the downstream rotor might function as a mixing device whose role is to facilitate momentum transfer. However it has also been pointed out that such a strategy needs extensive optimization for the mixing rotor to produce a positive torque. At a 90° incident angle, two configurations were tested. The co-rotating configuration was found to produce a higher gain at a small separation distance ($f = 1.2$) and the counter-rotating configuration might be preferred at a larger separation. At $f = 2.0$, the difference, however, was marginal; being 8% and 9.4%, respectively.

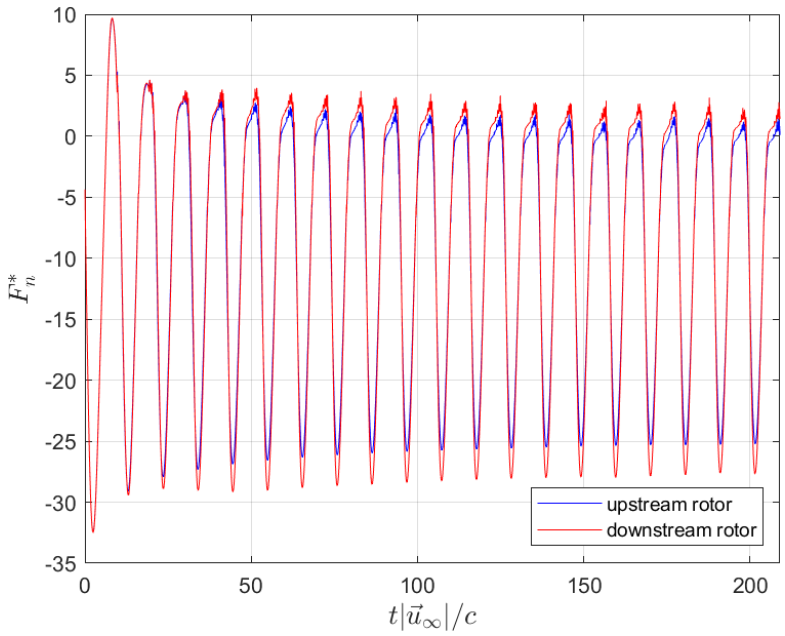


(A)

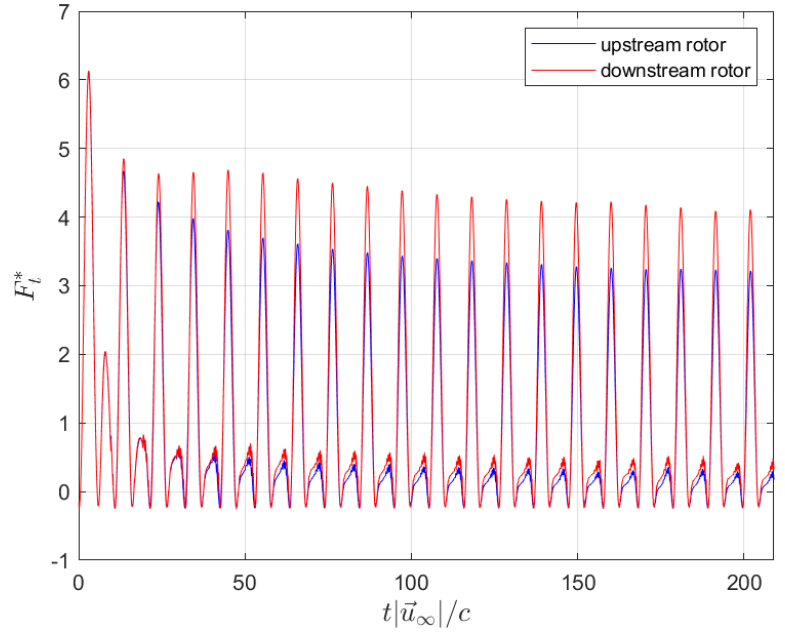


(B)

FIGURE 7.37: Computed normal and tangential coefficients for the upstream and downstream rotor for the C4R1 simulation.

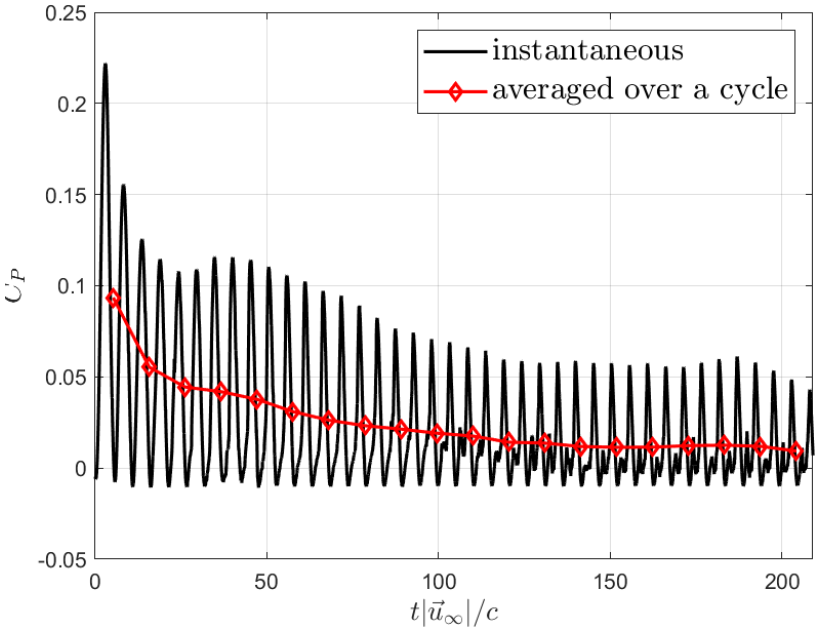


(A)

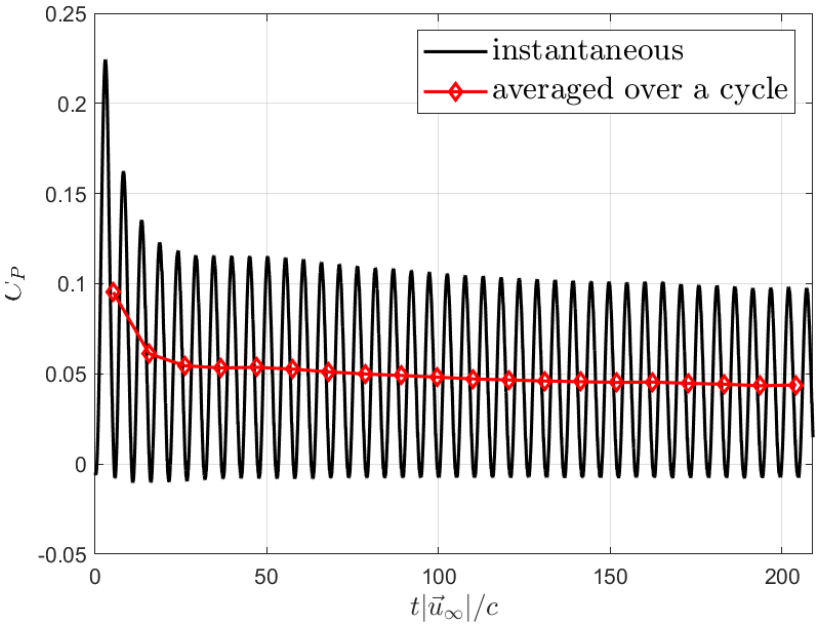


(B)

FIGURE 7.38: Computed normal and tangential coefficients for the upstream and downstream rotor for the C4R2 simulation.

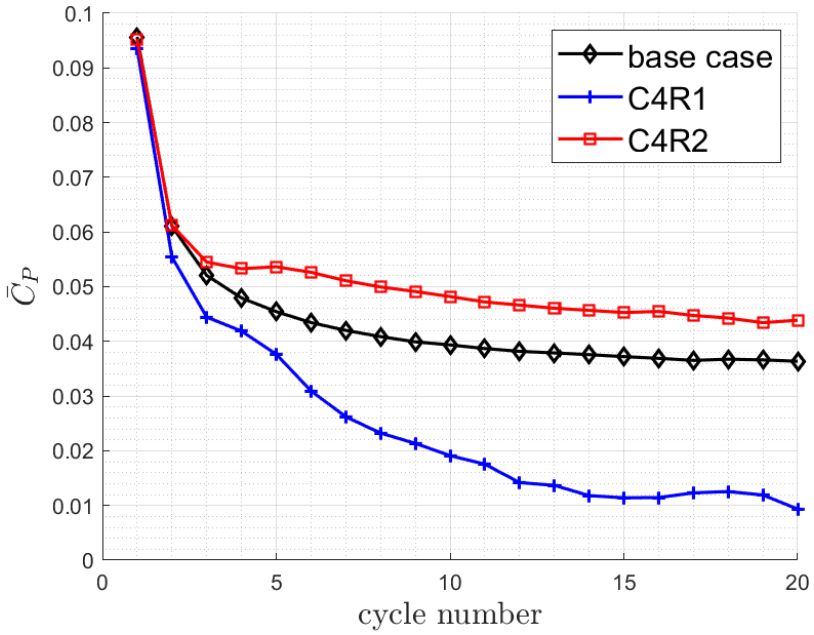


(A) $f_x = 2.0, f_y = 0.5$

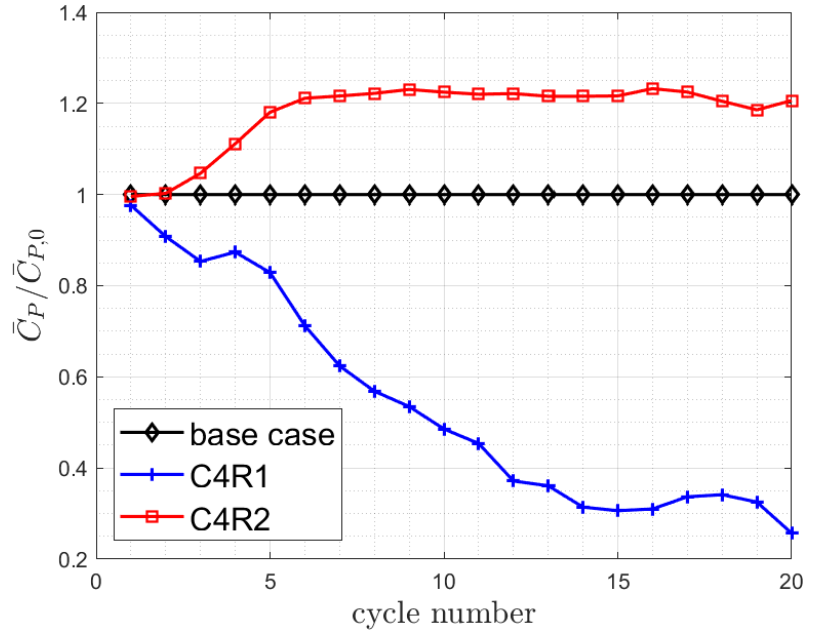


(B) $f_x = 2.0, f_y = 1.3$

FIGURE 7.39: Computed power coefficients for the C4R1 and C4R2 simulations.

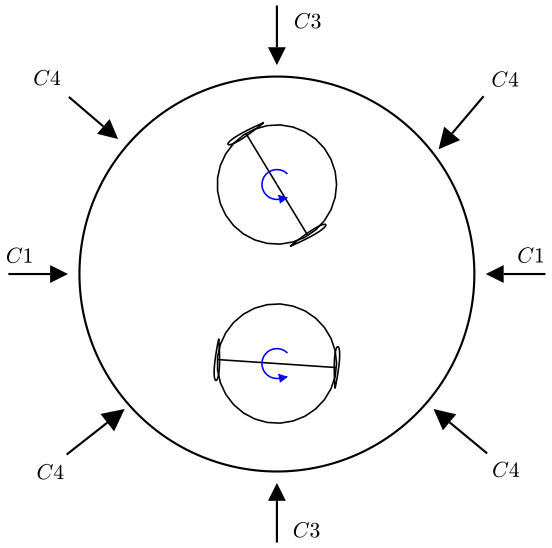


(A) $\bar{C}_P$

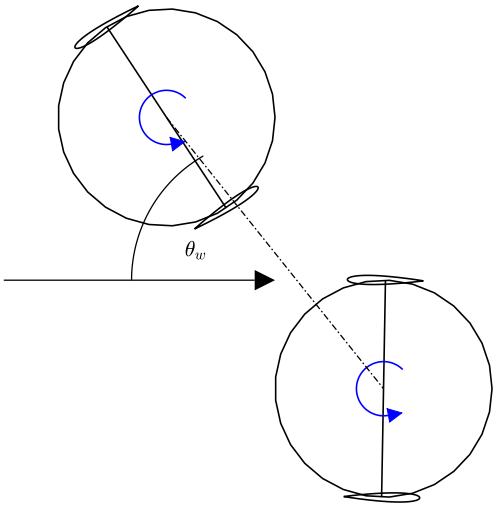


(B) $\bar{C}_{P,\text{relative}}$

FIGURE 7.40: Comparison of the power coefficients for the C4R1 and C4R2 simulations to the benchmark case.



(A)



(B)

FIGURE 7.41: (A) Shows the schematic interpretation of the configurations as defined by the incident wind direction. (B) Illustrates the definition of the wind incident angle θ_w.

Chapter 8

Conclusion and future work

The original aim of the thesis was to study the interaction of wind turbines in the context of a wind-farm. Preliminary studies have shown that the actuator disk assumption coupled with CFD does not give an accurate representation of the near-field wake dynamics. This is particularly the case when multiple VAWTs are in close proximity to each other. As a result of those studies, the idea of using CFD was quickly dropped in favour of an engineering approach in the near-field. Two objectives were borne out of this. First is to develop a low-order engineering method that is capable to represent the flow physics in an approximate sense. Secondly, we aim to complement the low-order method with a high-fidelity simulation tool so as to account for the inherent weakness in the low-order approach. This thesis has largely fulfilled these two objectives. This chapter reviews the primary conclusions of the studies conducted in this thesis.

It is at the heart of the code development cycle that a large degree of generalization should be exercised in order to apply the developed technique to a wide variety of engineering problems relating to fluid flow. With the increasing demand for wind energy, optimization of wind-farm and aerofoil sections plays an increasingly important role to deliver the most optimal solution both at the blade and rotor scale. Load prediction methods such as the Jensen wake model and the DMST cannot predict the wake structure of an isolated turbine, therefore they are excluded from a large set of problems where an accurate wake analysis is required. On the other hand, CFD methods are too computational demanding to be used as a viable design tool. Coupled with the empiricism for turbulence modelling and the excessive numerical dissipation inherent in the grid-based discretization, the methods are deemed insufficient for the applications considered in this thesis. The adaptation of the BEM in the context of rotor aerodynamic is aplenty in the literature, however, the implementation of those approaches is vastly inefficient owing to the quadratic complexity of evaluating the Biot Savart law. The novelty of this work is to provide a unified code frame in which several acceleration techniques are exploited and implemented on the GPU. This results in an order of magnitude speed-up compared to other BEM codes. Thus this work serves as an important basis for future endeavour on the aerodynamic analysis of rotor wake as well as on the optimization techniques of wind-farm.

8.1 Primary conclusions

In chronological order, the achievements in this work are as follows:

1. The mathematical formulation of the Boundary Element Method (BEM) has been reviewed and adapted for both 2D and 3D flows. To speed up the computation of the induction calculation, an acceleration routine has been developed

which uses the multipole expansion method to approximate the far-field influence. In the 3D case, the influence of the panels are represented by the Complex Spherical Harmonic Basis (CSHB) functions. Timing results show that the accelerated routine manages to achieve a speed up that is dependent on the truncation number. In practice, such a tool is invaluable when fine-grained discretization is sought and especially in the high-fidelity simulations.

2. The vortex particles method (VPM) has been reviewed and incorporated in the code as vorticity carriers. The developed tool offers an extensive modelling option such as the particle-strength-exchange (PSE) scheme, the random-walk scheme and the grid interpolation kernels. The developed methodologies have been coded in the GPU, which offers a substantial speed-up compared to a pure CPU computation.

3. A coupling strategy has been developed which couples the BEM with VPM via an accurate interpolation technique. The coupling has been successfully validated in later studies.

4. The Fast Multipole Method (FMM) has been reviewed and implemented in the GPU for both 2D and 3D. Due to the different memory architecture in the GPU, an efficient tree-construction algorithm has been developed, which sorts the particle cluster in a reversed order. In addition, an analytical expression has been derived for calculating the strain exerted by the neighbouring vortex particles. As far as we are aware such an expression is not available in the literature.

5. A series of validation studies have been successfully completed. In particular, the 3D VPM was validated by simulating the dynamic evolution of an inviscid vortex ring. The computed result agreed well with the theoretical prediction. The 2D VPM was validated by simulating the inviscid elliptical vortex patch. The effective aspect ratio was correctly calculated at some time, but discrepancy was observed. We attributed the discrepancy to the variation of the initial vorticity distribution. The particle-strength-exchange scheme was successfully validated by solving the heat-equation on a one-dimensional line.

6. The mathematical formulation for the high-fidelity approach has been reviewed and implemented in the GPU. The method relies on the concept of operator splitting in which the advection and diffusion processes are serialized. To extend the method to arbitrary geometry, the particle re-meshing algorithm near the geometric bodies has been simplified. The method was tested on flow past circular cylinders. The linear impulse and the drag calculation matched well with the literature.

7. The high-fidelity approach has been applied to a variety of flow problems involving aerofoils. The simulated results have been successfully compared to experimental flow visualization. The method is then used to study dynamic stall of a one-bladed rotor. It was found that the topological properties of the vortex shedding is similar to the static case at a similar angle of attack.

8. The low-order BEM was applied to VAWT with different solidity and blade numbers. In the validation studies, the normal and tangential forces were calculated and compared to experiments. Good agreement was obtained for both normal and tangential forces for low solidity rotors. Flow visualization revealed that the wake structure is correctly represented.

9. The BEM was used to simulate the pairwise interaction between two VAWTs. Several configurations have been proposed to study the aerodynamic influence in a co-operating state. A key-finding indicates that the downstream rotor could extract as much as 20% more power if placed near the wake region. The computed results also revealed that if the wind incident angle is zero, the effect on the downstream rotor is detrimental. However, flow visualization suggests there is an increased recovery rate as indicated by the shorter transition region. It was proposed that the downstream rotor might function as a momentum mixer.

8.2 Recommendation for future investigations

Despite the myriad of achievements in this thesis, there are still many gaps in our understanding. For example, the far-field wake in the simulation of the VAWT might not be correctly represented despite the PSE scheme being applied. In the low-order approach, the time and spatial scales associated with the vortex shedding can not be correctly resolved via the grid interpolation technique due to the large Reynolds number inherent in the assumption of the BEM. This then leads to the inaccurate prediction of the far-field where diffusion is significant. For this reason, a LES correction has to be applied to the flow to account for those small scales.

In the high-fidelity approach, a uniform grid was prescribed to facilitate the remeshing operation. The size of the grid cell is kept constant throughout the computational domain, which can incur a substantial cost penalty if the points of interest lie close to the body geometry. One way to alleviate this cost is to implement a body-fitted overset mesh. In this hybrid approach, a fine grid may be employed in the vicinity of the geometry while a coarser grid can be used together with a LES scheme to model the vorticity field outside. An effective interpolation scheme is then required which must satisfy several conservation rules, such as the total circulation, the linear momentum and the angular momentum. Moreover, an adaptive grid may also be constructed to eliminate the numerical perturbation of the developed method observed in Section 6.2.2. However, one should note that the adaptive grid may not be possible to construct for complicated body geometry.

Currently, all of the simulations were performed exclusively on a workstation with a dedicated graphics card. It is anticipated that utilising multiple graphics cards via Message Passing Interface (MPI) could achieve a much better solution time. This is essential if one wants to apply the high fidelity simulation tool for a real turbine. Thus one needs to modify the algorithms in the code to take full advantage of the High Performing Computing (HPC) facility of the university. Another possible improvement to the code is to port some of the back-end operations to a low-level language such as C++ or Fortran; utilising the highly efficient linear algebra package whilst avoiding most of the frontend code checking procedures of MATLAB.

Presently, the viscous forces exerted on the body were omitted in the engineering model. The viscous forces become important in large solidity rotors. It has been suggested that a viscid-inviscid coupling strategy may be employed by solving the integral boundary layer equations and deriving the friction coefficients due to viscous drag. However, one needs to be careful to ensure that the unsteadiness in the problem is not too large so as to invalidate the empirical terms in the boundary layer formulations.

In a real environment, the observed flow structure exhibits a strong 3D effect, especially in the tip region where strong tip vortices are shed. This is not modelled

in the 2D simulations. In addition, the validations of the 3D codes in the context of turbine aerodynamic are left for future work.

In the simulations of rotor interactions, it is found that when the wind incident angle is zero, the downstream turbine could not generate any torque. However, due to its presence, the transition region has been reduced considerably. One strategy to achieve a faster wake recovery under such an inflow condition is to utilise the downstream rotor as a momentum mixer. However, from an economic consideration, the downstream rotor needs to produce a positive torque. A possible future topic is to conduct parametric studies into the operating characteristic for a downstream rotor by varying the number of blades, the solidity, the TSR as well as the blade height in the 3D case.

In the FMM code, the most expensive operation involves computing the local expansion, which is done via a two-step approach. A significant latency was observed in the M2L step; accounting for over 60% of the total GPU time. Part of the reason is that a large amount of random fetch requests were made to the main memory. This step can be improved substantially by designing an efficient memory management system with a careful usage of shared and constant memory space.

Appendix A

Compute Unified Device Architecture (CUDA)

In this thesis, the GPU implementation uses the Compute Unified Device Architecture (CUDA) developed by Nvidia (NVIDIA et al., 2020). CUDA provides a parallel computing platform and the application programming interface (API) to implement parallel calculations. The platform is designed to work with C, C++ and Fortran programming languages. In this appendix, a basic overview of CUDA is given.

A.1 Basic overview of CUDA

In computing, a basic unit of executing a sequence of instruction is known as a *thread*. The GPU architecture is built around a scalable array of multithreaded *Streaming Multiprocessors* (SM), which is designed to execute hundreds of these threads concurrently. To manage such a large amount of threads, each multiprocessor is based on the *SIMT* (Single-Instruction, Multiple-Thread) architecture, in which groups of 32 parallel threads (called the *warps*) are created and managed. CUDA abstracts this layer by introducing a thread hierarchy in which the threads are grouped into blocks (called the thread block) and the blocks are grouped into grids. Each thread within a thread block is identified by a three-component vector $\left(t_x, t_y, t_z\right)$ (accessible by the struct `threadIdx`) and a thread block within a local grid may be identified by the three-component vector $\left(D_x, D_y, D_z\right)$ (accessible by the struct `blockIdx`). Due to the limited memory resources on each computing core, each thread block may contain up to 1024 threads. Thus, a global index in a three-dimensional array of data may be associated with a local thread and can be identified as follows:

$$id_{x,\text{global}} = t_x + D_x L_x,\ id_{y,\text{global}} = t_y + D_y L_y,\ id_{z,\text{global}} = t_z + D_z L_z,$$

where $\left(L_x, L_y, L_z\right)$ denotes the block lengths in the x, y, z direction, respectively (accessible by the struct `blockDim`). This thread hierarchy is illustrated in Figure (A.1). Functions that are executable on the GPU are called the *Kernels*. In a typical GPU calculation, device pointers (specialized pointers that operate on the GPU) are first created and memories are allocated via the API provided by CUDA. Host (CPU) data are then copied to the GPU memory via the dedicated pointers. The kernels are launched by specifying the launch parameters, such as the thread block sizes, the number of grids in each direction and the amount of shared memory to be used in the kernel. During a single time step update, kernels are launched sequentially from the host (CPU). For ease of implementation, the current work uses MATLAB's `CUDAKernel` class to manage the host side of the kernel launch.

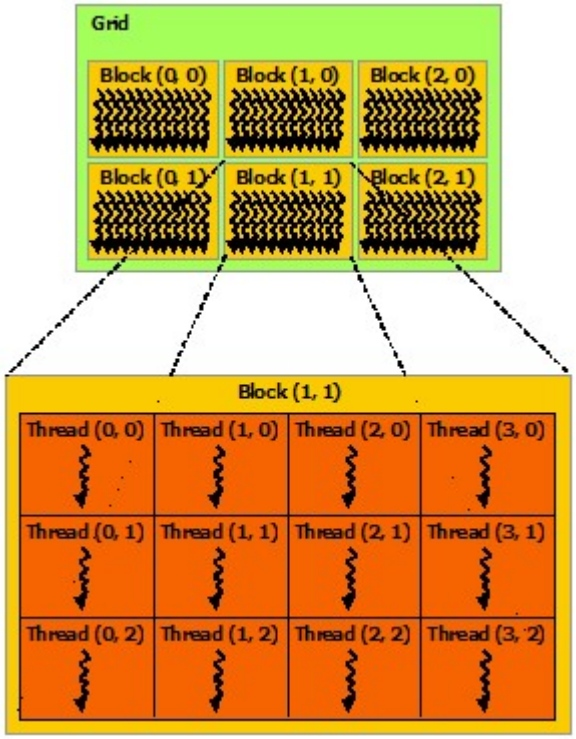


FIGURE A.1: A schematic illustration of the thread hierarchy implemented in NVIDIA et al. (2020).

A.2 The memory architecture

In contrast to the CPU's memory architecture, which is based on a distributed memory model (each computing processor has its private memory space), the Nvidia GPU is built around a *shared memory model*, in which a global memory space is visible to all threads in the kernel at all time. In addition, each CUDA thread has private local memory and each thread block has shared memory visible only to the threads within that block.

Data manipulation requires the data to be stored in different hardware memory caches. Access latency is characterised by how far the caches are located relative to the processing units. For example, the *L1* cache is built on-chip of the SM, therefore they have the highest bandwidth but at a much smaller size. Data are first write to the caches before they are being processed by the SM. An important performance aspect of GPU computing is to have as few cache transaction as possible since fetch requests to the main memory requires a large number of clock cycles. Typically, the *L1* and the *L2* are used to cache access to local or global memory.

CUDA offers a high level programming language which abstracts much of the memory optimization techniques used by the device. From a user's perspective, a simplified notion of the memory hierarchy is illustrated in Figure (A.2). Memory management plays an important part of the GPU optimization. This typically involves making clever use of the fast memory caches on the device in order to avoid memory bottleneck.

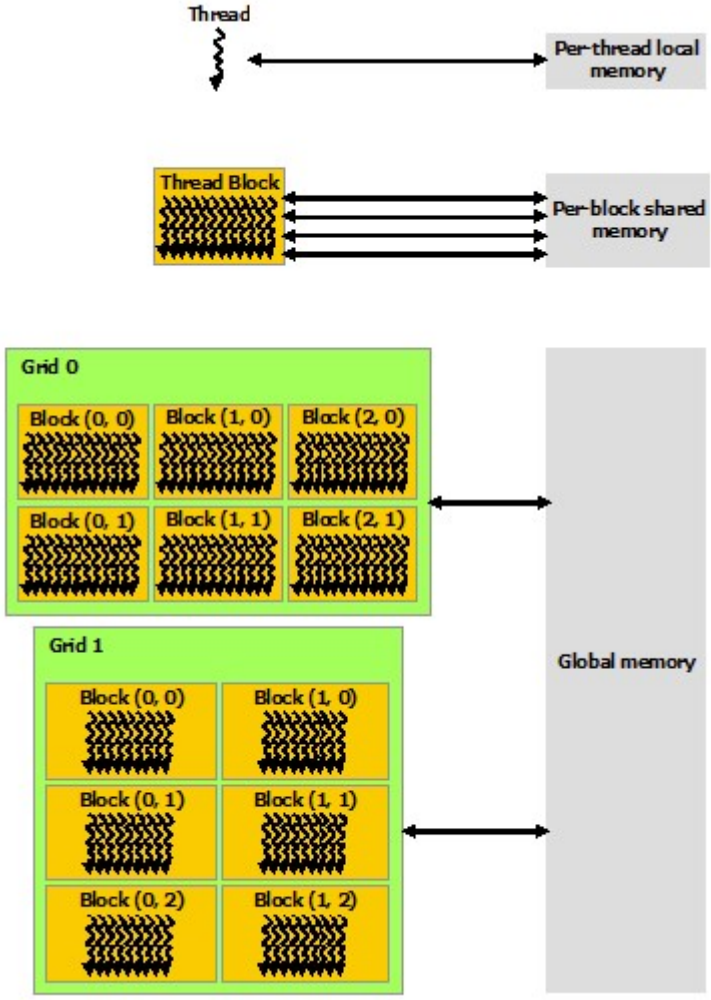


FIGURE A.2: A schematic illustration of the memory hierarchy implemented in NVIDIA et al. (2020).

Appendix B

Derivation of the eigenvalue of the Sturm-Liouville problem

Derivation of the Eigenvalue of the Sturm-Liouville is given in this appendix. Recall that the Sturm-Liouville problem sees us to determine the eigen solution to the differential operator $\mathcal{L}_{\text{SL}}$ such that, for some constant λ, the following holds:

$$\mathcal{L}_{\text{SL}}\Theta = -\lambda \sin\theta\Theta, \tag{B.0.1}$$

where the Sturm-Liouville operator is defined by:

$$\mathcal{L}_{\text{SL}} := \frac{d}{d\theta}\left(\sin\theta\frac{d}{d\theta}\right) - \frac{m^2}{\sin\theta}. \tag{B.0.2}$$

First, a change of variable is performed. Suppose $\Theta(\theta) = F(\cos\theta)$, with $x := \cos\theta$ and together with the chain rule, one can show that the θ derivative can be replaced with

$$\sin\theta\frac{d}{d\theta} = \sin\theta\frac{dx}{d\theta}\frac{d}{dx} = -\left(1-x^2\right)\frac{d}{dx}, \tag{B.0.3}$$

where we have used the fact that $1 = -\sin\theta d\theta/dx$ and $dx/d\theta = (d\theta/dx)^{-1}$. Substituting the above expression to Eq.(B.0.1), one has that:

$$\left(1-x^2\right)\frac{d}{dx}\left(\left(1-x^2\right)\frac{dF}{dx}\right) + \left(\lambda\left(1-x^2\right) - m^2\right)F = 0. \tag{B.0.4}$$

Divide Eq.(B.0.4) by the factor $\left(1-x^2\right)$ and setting $m = 0$, one obtains the particular equation of the form:

$$\frac{d}{dx}\left(\left(1-x^2\right)\frac{dF}{dx}\right) + \lambda F = 0. \tag{B.0.5}$$

We assume the solution for F is bounded at the boundary points $\theta = 0, \pi$, so one could look for a power expansion for the solution F, i.e.

$$F(x) = \sum_{n=0}^{\infty} a_n x^n. \tag{B.0.6}$$

Substituting Eq.(B.0.6) to Eq.(B.0.5) and collecting like-terms, one obtains:

$$\left(1-x^2\right)\frac{dF}{dx} = a_1 + \sum_{n=1}^{\infty}\left(\left(n+1\right)a_{n+1} - \left(n-1\right)a_{n-1}\right)x^n. \tag{B.0.7}$$

Differentiate again w.r.t x, so that Eq.(B.0.7) becomes:

$$\frac{d}{dx}\left(\left(1-x^2\right)\frac{dP}{dx}\right)=\sum_{n=0}^{\infty}\left(\left(n+1\right)\left(n+2\right)a_{n+2}-n\left(n+1\right)a_n\right)x^n. \tag{B.0.8}$$

Eq.(B.0.5) implies for each $n\in\mathbb{N}^+$, the series coefficients satisfy the recursive relation:

$$a_{n+2}=\frac{n\left(n+1\right)-\lambda}{\left(n+1\right)\left(n+2\right)}a_n. \tag{B.0.9}$$

If $\lambda\neq k\left(k+1\right)$, for some positive interger k, then one can be informed from elementary convergence test ($|a_{n+2}/a_n|\to 1$ as $n\to\infty$) that the series is diverging, which violate the assumption of F. Therefore, a_n must varnish at some finite value of n and $\lambda=k\left(k+1\right)$ for some integer k. For a given k, we denote P_k, after an appropriate normalization, to be the polynomial solution to Eq.(B.0.5). The polynomial P_k of degree k is known as the *Legendre polynomial*.

For $m\neq 0$ (recall that m must take integer values), the solution to Eq.(B.0.4) might be found by the following formula (assume that $m>0$):

$$P_k^m=\left(1-x^2\right)^{m/2}\frac{d^m}{dx^m}\left(P_k\right). \tag{B.0.10}$$

Since P_k is a polynomial of degree k, it follows that if $m>k$, then $P_k^m\equiv 0$.

Appendix C

Contour dynamic formula

In this appendix, we present the derivation of the contour dynamic formula for evaluating the velocity field due to a constant vorticity distribution with close boundary C. This is motivated by the notion of the extended vorticity field concept. To put this into perspective, suppose a body is described by the body velocity $\vec{u}_b$:

$$\vec{u}_b(\vec{x},t) = \vec{u}_0(t) + \Omega(t)\vec{k} \times (\vec{x} - \vec{x}_0(t)), \tag{C.0.1}$$

for some spatial independent functions $\vec{u}_0$, Ω and $\vec{x}_0$. Let $\vec{\omega}_b$ denote the 'body' vorticity, such that $\vec{\omega}_b = \nabla \times \vec{u}_b$, so one may express the body vorticity as

$$\vec{\omega}_b = \nabla \times \vec{u}_b = 2\Omega\vec{k} = \omega_b\vec{k}, \tag{C.0.2}$$

where ω_b is the magnitude of $\vec{\omega}_b$ and is spatially independent (here, we have slightly abused the notation, ω_b should really be the compact support of the vorticity, so that it is zero for points outside of D. But this does not invalidate the subsequent derivation). Let $\vec{u}_\Omega$ denote the velocity due to this vorticity distribution. In general, $\vec{u}_\Omega \neq \vec{u}_b$. For this velocity, we may find a stream-function ψ_Ω such that

$$\vec{u}_\Omega = \nabla \times \left(\psi_\Omega \vec{k}\right). \tag{C.0.3}$$

Taking the curl of Eq.(C.0.3), one may show that ψ_Ω satisfies the Poisson equation:

$$\omega_b = -\nabla^2 \psi_\Omega. \tag{C.0.4}$$

We solve Eq.(C.0.4) via the method of Green's function. Let $G(\vec{x},\vec{y})$ denote the free-space Green's function to Eq.(C.0.4), so it satisfies the equation:

$$\nabla_x^2 G = \delta(\vec{x} - \vec{y}). \tag{C.0.5}$$

Note here the subscript x means we are differentiating with respect to the x variable in its argument. Indeed, one can show that:

$$G(\vec{x},\vec{y}) = \frac{1}{2\pi}\log\|\vec{x} - \vec{y}\|. \tag{C.0.6}$$

So we have:

$$\psi_\Omega(\vec{x},t) = -\frac{\omega_b(t)}{2\pi}\int_C \log\|\vec{x} - \vec{y}\|\, dA(\vec{y}) \tag{C.0.7}$$

Substitute for u_Ω, one has that:

$$\vec{u}_\Omega = -\frac{\omega_b}{2\pi}\int_C \nabla_x(\log\|\vec{x} - \vec{y}\|) \times \vec{k}\, dA(\vec{y}). \tag{C.0.8}$$

Taking the dot product with the basis vector $\vec{e}_x$ and rearranging the resulting expression, we arrive at the expression:

$$\vec{u}_\Omega \cdot \vec{e}_x = -\frac{\omega_b}{2\pi} \int_C \left(\vec{e}_x \times \vec{k}\right) \cdot \nabla \log \|\vec{x} - \vec{y}\| \; dA\left(\vec{y}\right). \tag{C.0.9}$$

Noting that $\vec{e}_x \times \vec{k} = \vec{e}_y$, we can further simplify Eq.(C.0.9) to

$$\vec{u}_\Omega \cdot \vec{e}_x = -\frac{\omega_b}{2\pi} \int_C \nabla \cdot \left(\vec{e}_y \log \|\vec{x} - \vec{y}\|\right) \; dA\left(\vec{y}\right). \tag{C.0.10}$$

By virtue of Stokes' theorem, which states that for a sufficiently smooth vector field $\vec{F}$, the following holds:

$$\int_C \nabla \cdot \vec{F} dA = \oint_{\partial C} \vec{k} \times \vec{F} \cdot d\vec{s}. \tag{C.0.11}$$

So, Eq.(C.0.10) becomes:

$$\begin{aligned} \vec{u}_\Omega \cdot \vec{e}_x &= -\frac{\omega_b}{2\pi} \oint_{\partial C} \left(\vec{k} \times \vec{e}_y\right) \cdot \; d\vec{s} \log \|\vec{x} - \vec{y}\|, \\ &= -\frac{\omega_b}{2\pi} \oint_{\partial C} \left(\vec{e}_x \cdot d\vec{s}\right) \log \|\vec{x} - \vec{y}\|. \end{aligned} \tag{C.0.12a}$$

In the same manner, the expression $\vec{u}_\Omega \cdot \vec{e}_y$ is similarly derived. Finally, the velocity field induced by a constant vorticity field is given as follows:

$$\vec{u}_\Omega\left(\vec{x}, t\right) = -\frac{\omega_b\left(t\right)}{2\pi} \oint_{\partial C} \log \|\vec{x} - \vec{y}\| \; d\vec{s}\left(\vec{y}\right). \tag{C.0.13}$$

Eq.(C.0.13) is known as the contour dynamic formula.

In practice, Eq.(C.0.13) is applied for a set of linear panels which represent the body geometry. Consider the i-th panel whose end nodes are given by $\vec{x}_i$ and $\vec{x}_{i+1}$. Further, let L_i denote the length and $\vec{t}_i$ be the local tangent. The velocity due to this panel, denoted by $\vec{u}_{\Omega,i}$, is expressed as follows:

$$\vec{u}_{\Omega,i} := -\frac{\omega_b\left(t\right)}{2\pi} \vec{t}_i \int_0^{L_i} \log \|\vec{x} - \vec{y}\left(s\right)\| \; ds, \tag{C.0.14}$$

where $\vec{y} := \vec{x}_i + s\vec{t}_i$. The resulting integral can be simplified:

$$\vec{u}_{\Omega,i} = -\frac{\omega_b}{4\pi} \vec{t}_i \int_0^{L_i} \log \left|\left(s - \alpha\right)^2 + \beta^2\right| \; ds \tag{C.0.15}$$

where $\alpha := \vec{t}_i \cdot \left(\vec{x} - \vec{x}_i\right)$ and $\beta^2 = \|\vec{x} - \vec{x}_i\|^2 - \alpha^2$. An analytical expression can be found by applying integration by parts. In any case, the analytical expression is derived as follows:

$$\vec{u}_{\Omega,i} = \frac{\omega_b}{4\pi} \vec{t}_i \left[X \log\left(X^2 + \beta^2\right) - 2X + 2\beta \tan^{-1}\left(\frac{X}{\beta}\right)\right]_\alpha^{\alpha - L_i}. \tag{C.0.16}$$

One may verify that Eq.(C.0.16) gives the principal value if $\vec{x}$ is a boundary point. This corresponds to the fact that $\beta \to 0$. Specifically, if $\vec{x}$ is a boundary point then there exists a $\lambda^* \in (0, 1)$ such that $\vec{x} = \vec{x}_i + L_i \lambda^* \vec{t}_i$, so that the principal value is given

by:

$$\vec{u}_{\Omega,i} = -\frac{\omega_b}{2\pi}\vec{t}_i L_i \left(\log L_i + \lambda^* \log \lambda^* + (1-\lambda^*)\log(1-\lambda^*) - 1\right). \tag{C.0.17}$$

An important observation is that $\vec{u}_{\Omega,i}$ is properly defined even at the end points, i.e. $\lambda^* = 0, 1$.

Bibliography

Abbott, I. H., & von Donenhoff, A. E. (1949). *Theory of Wing Sections including a summary of airfoil data* (1st). Dover.

ANSYS. (2013). Fluent Theory Guide, Release 15.0.

Bar-Lev, M., & Yang, H. T. (1975). Initial flow field over an impulsively started circular cylinder. *Journal of Fluid Mechanics*, *72*(4), 625–647. https://doi.org/10.1017/S0022112075003199

Barnes, J., & Hut, P. (1986). A hierarchical O(N log N) force-calculation algorithm. *nature*, *324*, 446–449.

Basu, B. C., & Hancock, G. J. (1978). The unsteady motion of a two-dimensional aerofoil in incompressible inviscid flow. *Journal of Fluid Mechanics*, *87*(1), 159–178.

Batchelor, G. (1967). *An Introduction to Fluid Dynamics*. Cambridge University Press.

Berdowski, T. J. (2015). *3D Lagrangian VPM-FMM for Modelling the Near-Wake of a HAWT* (Master Thesis). Delft University of Technology.

Brusca, S., Lanzafame, R., & Messina, M. (2014). Design of a vertical-axis wind turbine: how the aspect ratio affects the turbine's performance. *International Journal of Energy and Environmental Engineering*, *5*(4), 333–340.

Carrier, J., Greengard, L., & Rokhlin, V. (1988). A Fast Adaptive Multipole Algorithm for particle simulations. *SIAM J. Sci. Stat. Comput.*, *9*(4), 669–686.

Cebeci, T., Platzer, M., Chen, H., Chang, K., & Shao, J. (2005). *Analysis of Low-Speed Unsteady Airfoil Flows*. Horizon Publishing Inc.

Cebeci, T., Mosinskis, G. J., & Smith, A. M. O. (1972). Calculation of separation points in incompressible turbulent flows. *Journal of Aircraft*, *9*(9), 618–624.

Cheng, H., Greengard, L., & Rokhlin, V. (1999). A Fast Adaptive Multipole Algorithm in Three Dimensions. *Journal of Computational Physics*, *155*(2), 468–498.

Choi, C. H., Ivanic, J., Gordon, M. S., & Ruedenberg, K. (1999). Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion. *Journal of Chemical Physics*, *111*(19), 8825–8831.

Chorin, A. J. (1973). Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, *57*(4), 785–796. https://doi.org/10.1017/S0022112073002016

Cottet, G.-H., & Koumoutsakos, P. D. (2008). *Vortex Methods Theory and Practice*. Cambridge University Press.

Dabiri, J. O. (2011). Potential order-of-magnitude enhancement of wind farm power density via counter-rotating vertical-axis wind turbine arrays. *Journal of Renewable and Sustainable Energy*, *3*(4).

Deglaire, P. (2010). *Analytical Aerodynamic Simulation tools for Vertical Axis Wind Turbines* (PhD thesis). Uppsala University.

Degond, P., & Mas-Gallic, S. (1989). The Weighted Particle Method for Convection-Diffusion Equations. Part 1: The Case of an Isotropic Viscosity. *Mathematics of Computation*, *53*(188), 485. https://doi.org/10.2307/2008716

Dixon, K. R. (2008). *The Near Wake Structure of a Vertical Axis Wind Turbine: Including the Development of a 3D unsteady Free-Wake Panel Method of VAWTs* (Master thesis). Delft University of Technology.

Doshi, M. R., & Gill, W. N. (1970). A note on the mixing length theory of turbulent flow. *AIChE Journal*, *16*(5), 885–888. https://doi.org/10.1002/aic.690160532

Drela, M. (1985). *Two-dimensional transonic aerodynamic design and analysis using the Buler equation* (PhD thesis). Massachusetts Institute of Technology.

Drela, M. (1989). XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils. *Conference on Low Reynolds Number Airfoil Aerodynamics*.

Drela, M., & Giles, M. B. (1987). Viscous-inviscid analysis of transonic and low Reynolds number airfoils. *AIAA Journal*, *25*(10), 1347–1355. https://doi.org/10.2514/3.9789

Dritschel, D. G. (1989). Contour dynamics and contour surgery: Numerical algorithms for extended, high-resolution modelling of vortex dynamics in two-dimensional, inviscid, incompressible flows. *Computer Physics Reports*, *10*(3), 77–146. https://doi.org/10.1016/0167-7977(89)90004-X

Eldredge, J. D. (2007). Numerical simulation of the fluid dynamics of 2D rigid body motion with the vortex particle method. *Journal of Computational Physics*, *221*(2), 626–648. https://doi.org/10.1016/j.jcp.2006.06.038

Etheridge, D., Steele, L., Langenfields, R., & Francey, R. (1996). Natural and anthropogenic changes in atmospheric CO2 over the last 1000 years from air in Antarctic ice and firn. *Geophysical Research*, *101*, 4115–4128.

Feng, G., Troyer, T. D., & Runacres, M. (2014). Optimizing land use for wind farms using vertical axis wind turbines. *Proceedings of the EWEA Annual Event 2014*.

Ferreira, C. S. (2009). *The near wake of the VAWT 2D and 3D views of the VAWT aerodynamics* (Doctoral dissertation). https://doi.org/urn:NBN:nl:ui:24-uuid:ff6eaf63-ac57-492e-a680-c7a50cf5c1cf

Friedman, A. (2008). *Partial Differential Equations of Parabolic Type*. Dover Publications Inc.

Glauert, H. (1947). *The Elements of Airfoil and Airscrew Theory* (2nd). Cambridge University Press.

Goude, A., & Engblom, S. (2013). Adaptive fast multipole methods on the GPU. *Journal of Supercomputing*, *63*(3), 897–918. https://doi.org/10.1007/s11227-012-0836-0

Graham, J. M. R. (1983). The lift on an aerofoil in starting flow. *Journal of Fluid Mechanics*, *133*, 413–425.

Greengard, C. (1985). The core spreading vortex method approximates the wrong equation. *Journal of Computational Physics*, *61*(2), 345–348. https://doi.org/10.1016/0021-9991(85)90091-9

Greengard, L., & Rokhlin, V. (1987). A fast algorithm for particle simulations. *Journal of Computational Physics*, *73*, 325–348. https://doi.org/10.1006/jcph.1997.5706

Greengard, L., & Rokhlin, V. (1997). A new version of the Fast Multipole Method for the Laplace equation in three dimensions. *Acta Numerica*, *6*, 229–269. https://doi.org/10.1017/S0962492900002725

Gumerov, N. A., & Duraiswami, R. (2008). Fast multipole methods on graphics processors. *Journal of Computational Physics*, *227*(18), 8290–8313. https://doi.org/10.1016/j.jcp.2008.05.023

Gumerov, N. A., & Duraiswami, R. (2015). Recursive Computation of Spherical Harmonic Rotation Coefficients of Large Degree. In R. Balan, M. Begue, J. Benedetto, W. Czaja, & K. Okoudjou (Eds.), *Applied and numerical harmonic analysis* (pp. 105–141). Birkauser, Cham. https://doi.org/10.1007/978-3-319-13230-3_5

Gumerov, N. A., Duraiswami, R., & Borovikov, E. A. (2003). *Data Structures, Optimal Choice of Parameters, and Complexity Results for Generalized Multilevel Fast Multipole Methods in* (tech. rep.). University of Maryland.

Hansen, M. O. L. (2008). *Aerodynamics of Wind Turbines* (2nd). Earthscan.

He, C., & Zhao, J. (2009). Modeling Rotor Wake Dynamics with Viscous Vortex Particle Method. *AIAA Journal*, *47*(4), 902–915. https://doi.org/10.2514/1.36466

Hess, J., & Smith, A. (1967). Calculation of potential flow about arbitrary bodies. *Progress in Aerospace Sciences*, *8*, 1–138. https://doi.org/10.1016/0376-0421(67)90003-6

Hildebrand, F. (2013). *Introduction to Numerical Analysis* (2nd ed.). Courier Corporation.

Hu, Q., Gumerov, N. A., & Duraiswami, R. (2013). GPU accelerated fast multipole methods for vortex particle simulation. *Computers & Fluids*, *88*, 857–865. https://doi.org/10.1016/j.compfluid.2013.08.008

Huang, R. F., Wu, J. Y., Jeng, J. H., & Chen, R. C. (2001). Surface flow and vortex shedding of an impulsively started wing. *Journal of Fluid Mechanics*, *441*, 265–292. https://doi.org/10.1017/S002211200100489X

Jensen, N. O. (1983). A note on wind generator interaction. *Risø-M-2411 Risø National Laboratory Roskilde*, 1–16. https://doi.org/Riso-M-2411

Katz, J. (1981). A discrete vortex method for the non-steady separated flow over an airfoil. *Journal of Fluid Mechanics*, *102*(1981), 315–328. https://doi.org/10.1017/S0022112081002668

Katz, J., & Plotkin, A. (2001). *Low-Speed Aerodynamics* (2nd). Cambridge University Press.

Klimas, P. (1982). Darrieus Rotor Aerodynamics. *J.Sol.Energy Eng*, *104*(2), 102–105.

Koumoutsakos, P. (1993). *Direct numerical simulations of unsteady separated flows using vortex methods* (PhD thesis). California Institute of Techology.

Koumoutsakos, P. (1997). Inviscid axisymmetrization of an elliptical vortex. *Journal of Computational Physics*, *138*(2), 821–857. https://doi.org/10.1006/jcph.1997.5749

Landhal, M., & Stark, V. (1977). Numerical lifting surface theory-problems and progress. *AIAA Journal*, *6*(11), 2049–60.

Launder, B., & Spalding, D. (1974). The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, *3*(2), 269–289. https://doi.org/10.1016/0045-7825(74)90029-2

Leishman, J. G., Bhagwat, M. J., & Bagai, A. (2002). Free-vortex filament methods for the analysis of helicopter rotor wakes. *Journal of Aircraft*, *39*(5), 759–775. https://doi.org/10.2514/2.3022

Madsen, H. A. (1982). *The actuator cylinder: A flow model for vertical axis wind turbines* (tech. rep. January 1982). Institute of Industrial Constructions and Energy Technology. https://doi.org/10.13140/RG.2.1.2512.3040

Monaghan, J. (1985). Particle methods for hydrodynamics. *Computer Physics Reports*, *3*(2), 71–124. https://doi.org/10.1016/0167-7977(85)90010-3

Moran, J. (1984). *An Introduction to Theoretical and Computational Aerodynamics* (Dover). Wiley.

Noca, F., Shiels, D., & Jeon, D. (1999). A comparison of methods for evaluating time dependent fluid dynamic forces on bodies, using only velocity fields and their derivatives. *Journal of Fluids and Structures*, *13*, 551–578.

NVIDIA, Vingelmann, P., & Fitzek, F. (2020). CUDA. https://developer.nvidia.com/cuda-toolkit

Oler, J., Strickland, J. H., Im, B., & Graham, G. (1983). *Dynamic-Stall Regulation of the Darrieus Turbine* (tech. rep.). Texas Tech University. Lubbock.

Orenstein, J. A., & Merrett, T. H. (1984). A class of data structures for associative searching. *Proceedings of the 3rd ACM SIGACT-SIGMOD symposium on Principles of database systems - PODS '84*, 181. https://doi.org/10.1145/588011.588037

O'Rourke, J. (1998). *Computational Geometry in C* (2nd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511804120

Paraschivoiu, I. (2002). *Wind Turbine Design: with Emphasis on Darrieus Concept* (S. Schettini, Ed.). Presses inter Polytechnique.

Paraschivoiu, I., Trifu, O., & Saeed, F. (2009). H-Darrieus wind turbine with blade pitch control. *International Journal of Rotating Machinery*, *2009*. https://doi.org/10.1155/2009/505343

Parneix, N., Fuchs, R., Silvert, A., & Deglaire, P. (2016). Efficiency Improvement of Vertical-axis Wind Turbines with Counter-Rotating Lay-out. *Wind Europe*.

Ploumhans, P., & Winckelmans, G. S. (2000). Vortex Methods for High-Resolution Simulations of Viscous Flow Past Bluff Bodies of General Geometry. *Journal of Computational Physics*, *165*(2), 354–406. https://doi.org/10.1006/jcph.2000.6614

Pope, S. B. (2000). *Turbulent Flows* (1st). Cambridge University Press.

Ramachandran, P., Ramakrishna, M., & Rajan, S. C. (2007). Efficient random walks in the presence of complex two-dimensional geometries. *Computers and Mathematics with Applications*, *53*(2), 329–344. https://doi.org/10.1016/j.camwa.2006.02.050

Ramos-García, N., Sørensen, J. N., & Shen, W. Z. (2014). A strong viscous-inviscid interaction model for rotating airfoils. *Wind Energy*, *17*(12), 1957–1984. https://doi.org/10.1002/we.1677

REN21. (2019). Renewables 2019: Global Status Report.

Riziotis, V. A., & Voutsinas, S. G. (2008). Dynamic stall modelling on airfoils based on strong viscous-inviscid interaction coupling. *International Journal for Numerical Methods in Fluids*, *56*(June 2007), 185–208. https://doi.org/10.1002/fld

Sebastian, T., & Lackner, M. A. (2012). Development of a free vortex wake method code for offshore floating wind turbines. *Renewable Energy*, *46*, 269–275. https://doi.org/10.1016/j.renene.2012.03.033

Spalart, P. R., & Leonard, A. (1981). Computation of separated flows by a vortex tracing algorithm. *14th Fluid & Plasma Conference*. https://doi.org/10.1007/3-540-59280-6_96

Spera, D. A. (2009). *Wind turbine technology: fundemental concepts of wind turbine engineering* (2nd). ASME Press.

Strickland, J. H. (1975). *The Darrieus Turbine, A Performance Prediction Method Using Multiple Stream Tubes* (tech. rep.). Sandia Laboratories.

Strickland, J. H., Webster, B. T., & Nguyen, T. (1979). A Vortex Model of the Darrieus Turbine: An Analytical and Experimental Study. *Journal of Fluids Engineering*, *101*(4), 500–505.

Sullivan, I. S., Niemela, J. J., Hershberger, R. E., Bolster, D., & Donnelly, R. J. (2008). Dynamics of thin vortex rings. *Journal of Fluid Mechanics*, *609*, 319–347.

Tescione, G., Simão Ferreira, C. J., & van Bussel, G. J. W. (2016). Analysis of a free vortex wake model for the study of the rotor and near wake flow of a vertical axis wind turbine. *Renewable Energy*, *87*, 552–563.

Van Dyke, M. (1988). *An Album of Fluid Motion* (4th ed.).

Versteeg, H. K., & Malalasekera, W. (1995). *An introduction to computational fluid dynamics. The finite volume method* (1st Editio). Longman Scientific & Technical.

Vezza, M., & Galbraith, R. A. M. D. (1985a). A method for predicting unsteady potential flow about an aerofoil. *International Journal for Numerical Methods in Fluids*, *5*(4), 347–356. https://doi.org/10.1002/fld.1650050405

Vezza, M., & Galbraith, R. A. M. D. (1985b). An inviscid model of unsteady aerofoil flow with fixed upper surface separation. *International Journal for Numerical Methods in Fluids*, *5*, 577–592.

Wang, Y., Abdel-Maksoud, M., & Song, B. (2018). A boundary element-vortex particle hybrid method with inviscid shedding scheme. *Computers and Fluids*, *168*, 73–86. https://doi.org/10.1016/j.compfluid.2018.03.062

Wie, S. Y., Lee, S., & Lee, D. J. (2009). Potential panel and time-marching free-wake coupling analysis for helicopter rotor. *Journal of Aircraft*, *46*(3), 1030–1041. https://doi.org/10.2514/1.40001

Willis, D. J. (2006). *An Unsteady, Accelerated, High Order Panel Method with Vortexx Particle Wakes* (PhD thesis). Massachusetts Institute of Technology.

Wilson, R. E., Lissaman, P. B., & Walker, S. N. (1976). *Aerodynamic Performance of Wind Turbines*. Oregan State University Press.

Wolles, B. A. (1968). *Computational viscid-inviscid interaction modelling of the flow about aerofoils* (PhD thesis). University of Twente.

Yokota, R., Narumi, T., Sakamaki, R., Kameoka, S., Obi, S., & Yasuoka, K. (2009). Fast multipole methods on a cluster of GPUs for the meshless simulation of turbulence. *Computer Physics Communications*, *180*(11), 2066–2078. https://doi.org/10.1016/j.cpc.2009.06.009

Zabusky, N. J., Hughes, M. H., & Roberts, K. V. (1979). Contour dynamics for the Euler equations in two dimensions. *Journal of Computational Physics*, *30*(1), 96–106. https://doi.org/10.1016/0021-9991(79)90089-5

Zanforlin, S., & Nishino, T. (2016). Fluid dynamic mechanisms of enhanced power generation by closely spaced vertical axis wind turbines. *Renewable Energy*, *99*, 1213–1226. https://doi.org/10.1016/j.renene.2016.08.015

Zanon, A., Giannattasio, P., & Ferreira, C. J. S. (2013). A vortex panel model for the simulation of the wake flow past a vertical axis wind turbine in dynamic stall. *Wind Energy*, *16*, 661–680. https://doi.org/10.1002/we