Accelerated Mesh-Based Bodies for the Boundary Data Immersion Method

Application to the TU Delft LEI V3 kite as a case study

MT54035: MSc Thesis

Rajan Hellemans

Delft University of Technology

TU Delft

Accelerated Mesh-Based Bodies for the Boundary Data Immersion Method

Application to the TU Delft LEI V3 kite as a case study

by

Rajan Hellemans

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on T.B.A.

Student number:	5227674
Project duration:	November 12, 2024 – September 5, 2025
Thesis committee:	Prof. G. D. Weymouth, TU Delft, supervisor
	Prof.dr.ir. S. Hickel, TU Delft

Cover:	Vorticity magnitude of a LEI kite simulated at RE=1e5 and $\alpha = 7.5°$; rendered using X-ray (top) and Black,Blue and White (bottom) color palette.
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Thesis for the degree of MSc in Marine Technology in the specialization of
Ship Hydromechanics

Accelerated Mesh-Based Bodies for the Boundary Data Immersion Method

By

Rajan Hellemans

Performed at

TU Delft

This thesis (**MT.25/26.004.M**) is classified as confidential in accordance with the general conditions for projects performed by the TUDelft.

15/10/2025

Thesis exam committee
Chair/Responsible Professor: Prof. G.D. Weymouth
Staff Member: Prof.dr.ir S.Hickel
Staff Member: Dr. M. Lauber

Author Details
Studynumber: 5227674

Preface

Global shipping accounts for a significant percentage of global emissions, but is an indispensable party of our modern society. On the quest to reduce these emissions wind-assisted shipping is being investigated as a viable, zero-emission propulsion method. This can be done using kites that are able to generate large amounts of force, but are aerodynamically difficult objects to study. This thesis investigates these aerodynamic properties by performing Large Eddy Simulations and in doing so implemented a fast way to study the aerodynamic behavior of complex shapes

The renewed interest in wind-assisted propulsion offers a compelling intersection between modern engineering and ancient seafaring. As global shipping faces mounting pressure to reduce greenhouse gas emissions, alternative propulsion methods have come back into focus. Among them, large-scale kites have emerged as a promising solution, capable of harnessing high-altitude wind energy without relying on fossil fuels.

As a marine engineer and sailor this object was well fit for me as it combines both these interests. I only started to my interest in Computational Fluid Dynamics during my Master's degree. Whether it are the pretty pictures or how there are seemingly infinite ways to obtain them, it is a branch of engineering that I fully enjoy and hope to keep using throughout my career. I truly enjoyed working on the MeshBody package for *WaterLily* and seeing how it works, both fast and correct.

I sincerely thank Professor G.D. Weymouth for his guidance and keeping this thesis going in the right direction. Special thanks to Dr. M. Lauber for helping me with technical issues and for patiently entertaining my many questions.

I would like to start by thanking my parents for being the wind in my ”kite”. Their gentle nudges about school have finally paid of. I would not know where I would stand without my sister beside me. Our shared drive has truly made us both better. I am grateful for the example my grandparents set for me, both in how to live and how to work. I would also like to thank my friends in Delft for making the past years more than just studying and for the much needed company during all the hours in the faculty and library.

Rajan Hellemans
Delft, September 2025

“ As the sail bellied out with the wind, the ship flew through the deep blue water, and the foam hissed against her bows as she sped onward ”

- Homer, The Odyssey, Book II
(Translated by Samuel Butler, 1900 [28])

Summary

Wind-assisted shipping is one of the oldest forms of marine transportation. In 2025, it has re-emerged as a promising method for reducing emissions. The atmospheric boundary layer provides a free, renewable source of energy that produces no greenhouse gases. Large kites are currently being tested both in the marine propulsion and energy sectors to harness this energy. These kites operate at high speeds, resulting in high Reynolds numbers and fully turbulent flow. While current methods, such as Reynolds-averaged Navier–Stokes (RANS) simulations and vortex step models, can model turbulent effects, they do not resolve them. In this thesis, we perform Large Eddy Simulations (LES) to investigate whether turbulence significantly affects the lift and drag forces acting on the kite.

To carry out these simulations, we used the *WaterLily* fluid solver. This implicit LES solver is based on the Boundary Data Immersion Method (BDIM), which allows for fully Cartesian grids and avoids body-fitted meshing. However, it requires a signed distance function (SDF) to define the submerged geometry. While SDFs are trivial for simple parametric bodies, generating them for complex meshes is computationally expensive. We implemented an efficient Bounding Volume Hierarchy (BVH) approach to reduce the number of distance calculations, significantly speeding up the submersion process.

The method led to a large reduction in computational cost and was validated for both bounding and non-bounding geometries. A new force evaluation routine was also implemented, looping over mesh elements rather than grid points—more efficient when mesh complexity is low relative to grid resolution. Using this improved framework, we simulated the TU Delft LEI V3 kite. Wall-resolved LES showed good agreement with existing data for the mean forces, confirming the validity of our results.

The flow field analysis revealed that at low angles of attack, the pressure side exhibited turbulence, while the suction side produced a strong drag wake. At higher angles, the flow partially reattached on the pressure side, reducing turbulence locally. Although absolute force fluctuations increased with angle of attack due to stronger turbulence, the relative fluctuations (compared to the mean) were actually larger at lower angles. Spectral analysis showed no dominant peaks, indicating an absence of coherent periodic forcing.

All simulations used an implicit LES model with no explicit subgrid-scale modeling. The non-dimensional wall-distance, y+, remained on the edge of the theoretical linear sublayer range. In one case, a significant deviation from validation data was traced to a lack of flow reattachment. This suggests that some simulations may be under-resolved. Incorporating an explicit subgrid-scale model could help clarify this. Future work should focus on dynamic simulations, where the Reynolds numbers are even higher. The current framework supports this through boundary condition manipulation or kite rotation, without requiring remeshing at each timestep.

Contents

Preface ii

Summary iv

Nomenclature x

1 Introduction 1

2 Literature Review 2
2.1 Emission goals for the future and kites as a solution . . . 2
2.2 Details and research on kite systems . . . 4
2.3 Large Eddy Simulations . . . 8
2.4 Boundary Data Immersion Method . . . 11
2.5 Signed distance functions . . . 11

3 Implementation of mesh based bodies in WaterLily 13
3.1 Boundary data immersion method . . . 13
3.2 Methodology for the Bounding Volume Hierarchy . . . 15
3.3 Traversing the tree . . . 18
3.3.1 Depth first, stack based traversal . . . 18
3.3.2 Stack-less, finite state machine traversal . . . 19
3.4 Constructing sdf's using the BVH . . . 22
3.5 Validation of signed distance functions . . . 23
3.6 Benchmarking of sdf generation . . . 26
3.7 Pressure probing from a mesh . . . 30
3.7.1 Implementation . . . 30
3.7.2 Validation of pressure probing from a mesh . . . 31
3.7.3 Obtaining y+ values from mesh . . . 34

4 Static simulations 36
4.1 Numerical setup . . . 36
4.1.1 Boundary conditions . . . 36
4.1.2 Kite geometry and domain size . . . 37
4.2 Results . . . 42
4.2.1 Mean drag and lift coefficients . . . 42
4.2.2 Estimated wall resolution . . . 45
4.2.3 Flow Field Analysis . . . 45
4.2.4 Fluctuating forces . . . 55
4.2.5 Pressure probes . . . 59

5 Proof of concept: Dynamic simulation 62
5.1 Simulation set-up . . . 62
5.2 Results . . . 63

6 Conclusion 65

References 66

List of Figures

2.1 Global international shipping CO_2 emissions 1970-2023, data plotted from Statista [14] 2
2.2 CO_2 emission reduction potential from individual measures, classified in 5 main categories of measures [6] . . . 3
2.3 Potential fuel savings thanks to wind-assisted ship propulsion [36] . . . 4
2.4 Fuel savings of a 50 000 DWT tanker using a propulsive kite [43] . . . 5
2.5 Parametrization of figures of eight with different pole circle sizes [13] . . . 5
2.6 Schematic drawing of SkySails steering system. Deformation is generally reduced to a minimum and profile ribs are almost not affected [18] . . . 5
2.7 Contour plots showing the axial induction factor distribution for an instant of a cycle at the kite location. The white rectangular region in the plots represents the kite [35]. . . . 6
2.8 Lift-over-drag ratio as a function of the angle of attack without side-slip [39]. . . . 7
2.9 Different types of aircraft in Ground-Gen systems. (a) LEI SLE (Leading Edge Inflatable, Supported Leading Edge) Kite; (b) LEI C-kite; (c) Foil Kite, design from Skysails;(d) Glider, design from Ampyx Power; (e) Swept rigid wing, design from Enerkite; (f) Semi-rigid wing, design from Kitegen. [9] . . . 8
2.10 Turbulent kinetic energy spectrum showing the different energy ranges [51] . . . 9
2.11 Phase-averaged spanwise vorticity over a heaving plate by Franck & Breuer [21], contour levels: -10 (blue) to 10 (red) . . . 9
2.12 Phase-averaged lift coefficient with the steady lift coefficient subtracted for better comparison. Two heaving cycles are shown to better display the trend that occurs upon the transition of one cycle to the next. [21] . . . 10

3.1 Conceptual sketch of the problem. Figure (a) is the original two-domain problem P0 with domains Ω b and Ωf and their interface σs and the respective governing equations and Figure (b) is the equivalent single-domain problem Pϵ with governing meta-equation constructed by the boundary data immersion method by convoluting and assembling the equations over a width ϵ. The transition between the equations is smooth and located within the small distance ϵ of the interface σ ; Taken from Weymouth and Yue, 2011 [66] . . . 14
3.2 Smoothing across the immersed boundary. The equations valid in each domain are convolved with a kernel of radius ϵ and added together. The gradient of gray illustrates how the contribution of b_ϵ and f_ϵ to the smoothed equation changes in the boundary region. The kernel at a point (marked by a dot) that belongs to the boundary region is represented; Taken from Maertens $\&$ Weymouth [41] . . . 14
3.3 General tree lay-out witrh 4 levels; 1-7 are nodes, 8-15 are leaves . . . 15
3.4 Box operations; in order from left to right: expanding, splitting into children, merging into parent . . . 16
3.5 16 subdivisions on the Stanford bunny from a 5-level BVH tree, expanded to show overlap 17
3.6 Leaves around the Stanford bunny for a BVH with 5 levels . . . 17
3.7 Two leaves (black) and their constructed parent node (green); part of the mesh inside the left leaf is shown in dark green . . . 17
3.8 Stack based traversal for a 3 level BVH; A green node indicates that the point is inside of the box and a red one indicates that it isn't . . . 19
3.9 FSM based traversal for a 3 level BVH; A green node that the point is inside of the box and a red one indicates that it isn't . . . 20
3.10 z component of wrongly calculated normal vectors; Both sides of the back canopy have normals pointing in the same direction . . . 23
3.11 z component of correctly calculated normal vectors; Both sides of the back canopy have normals pointing in different directions . . . 23
3.12 Slice of the analytical sdf of the two interlocked links . . . 24
3.13 Slice of the numerical sdf of the two interlocked links made with 6 levels . . . 25

3.14 Mean L1 errors in the sdf of two interlocked links plotted over the resolution 25
3.15 Mean L1 errors in the sdf of two interlocked links plotted over the resolution 26
3.16 Stanford bunny and Asian dragon used for benchmarking the BVH method and for validating the hydrostatic pressure routine 27
3.17 Total saving factor by level 27
3.18 Total saving factor by domain size 28
3.19 Construction times for different number of levels, mesh elements and domain sizes . . . 29
3.20 Measurement times for two different mesh sizes, scales and levels 29
3.21 Time saving factor for two different mesh sizes, scales and levels 29
3.22 Vectors and points used to determine the pressure on a triangle element; $\overline{cq_1}$ is defined by $n * \delta_1$ and $\overline{q_1 q_2}$ is defined by $n * \delta_2$ 30
3.23 Pressure distribution on a spherical mesh 32
3.24 Boxplot of errors between current WaterLily method and mesh based method 32
3.25 Drag coefficient over time for a sphere at increasing domain sizes: (a) 320x128x128, (b) 480x196x196 (c) 640x256x256 33
3.26 Layers on the edge of a submerged object. Green line represents the mesh, full line is the 0 iso-contour of the resulting sdf at the half thickness from the mesh, dotted lines are the edges of the transition zone. Red lines represent part of the submerged object we can not include with the mesh based method 34

4.1 Velocity magnitude at the inflow due to the Biot-Savart boundary conditions 37
4.2 Velocity magnitude at the top boundary due to the Biot-Savart boundary conditions . . . 37
4.3 Detailed geometry of the TU Delft V3 LEI kite [1] 39
4.4 Signed distance field of the kite at the symmetry plane; Grey areas represent locations that were not measured 40
4.5 Signed distance field of the wing at x/c=0.5; Grey areas represent locations that were not measured 40
4.6 0 iso-contour representing the location of the kite in the domain, the high resolution makes it almost identical to surface the mesh 41
4.7 Detail of where strut is attached to leading edge and canopy; iso-contour of $|\mu_0| = 1.5$. 41
4.8 Time traces of drag (C_d) and lift (C_l) coefficients at $Re = 100000$ for different angles of attack; **(a,c,e** & **g)** : C_D ; **(b,d,f** & **h)** : C_L 43
4.9 Mean drag force, C_D, over angle of attack; Validation data from [1] 44
4.10 Mean lift force, C_L, over angle of attack; Validation data from [1] 44
4.11 Mean lift over drag force over angle of attack; Validation data from [1] 45
4.12 Streamlines and magnitude of the velocity field in the wing symmetry plane for Re= 1e5 and $\alpha = 6^\circ$ taken from [39]; Streamlines show a recirculation zone, but reattach on the canopy 46
4.13 Velocity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 0^\circ$ 47
4.14 Velocity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 7.5^\circ$ 48
4.15 Velocity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 15^\circ$ 48
4.16 Velocity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 20^\circ$ 49
4.17 Pressure field in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 0^\circ$ 49
4.18 Pressure field in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 7.5^\circ$ 50
4.19 Pressure field in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 15^\circ$ 50
4.20 Pressure field in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 20^\circ$ 51
4.21 Vorticity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 0^\circ$ 51
4.22 3D representation of the Q-criterion iso-contours of $Q_{crit} = 0.01$ for $Re = 1 \times 10^5$ and $\alpha = 0^\circ$ 52
4.23 Vorticity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 7.5^\circ$ 52
4.24 3D representation of the Q-criterion iso-contours of $Q_{crit} = 0.01$ for $Re = 1 \times 10^5$ and $\alpha = 7.5^\circ$ 53
4.25 Vorticity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 15^\circ$ 53
4.26 3D representation of the Q-criterion iso-contours of $Q_{crit} = 0.01$ for $Re = 1 \times 10^5$ and $\alpha = 15^\circ$ 54
4.27 Vorticity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 20^\circ$ 54

4.28 3D representation of the Q-criterion iso-contours of $Q_{crit} = 0.01$ for $Re = 1 \times 10^5$ and $\alpha = 20°$ 55
4.29 Fourier analysis of the time trace of the drag coefficient for different angles of attack . . 56
4.30 Fourier analysis of the time trace of the lift coefficient for different angles of attack . . . 57
4.31 Kernel density estimate of the fluctuations in the lift coefficient, C'_L for different angles of attack 58
4.32 Kernel density estimate of the fluctuations in the lift coefficient, C'_L relative to the mean lift coefficient C_L for different angles of attack 58
4.33 Kernel density estimate of the fluctuations in drag force, C'_D for different angles of attack 59
4.34 Kernel density estimate of the fluctuations in drag force, C'_D relative to the mean lift coefficient C_D for different angles of attack 59
4.35 Points at which the pressure was recorded for 15 additional time steps 60
4.36 Time traces of the recorded pressure, labels indicate the x position of the point at which the data was recorded 60
4.37 Fourier transforms of the pressure value at different points along the kite 61
4.38 Time averaged pressure on the pressure and suction side of the kite 61

5.1 $\mu_{0,x}$ field at the centerline of the kite at different timesteps. 62
5.2 Time histories of drag and lift forces during one period of motion. 63
5.3 Pressure field snapshots during dynamic motion. 63
5.4 Streamwise velocity (u_x) field snapshots during dynamic motion. 64
5.5 Vertical velocity (u_z) field snapshots during dynamic motion. 64

List of Tables

2.1 Overview of parameters in current research 7

3.1 Overview of levels, grid cells, and mesh elements used for validation 24
3.2 Errors in comparing calculated hydrostatic pressure and volume 31

4.1 Domain parameters 38
4.2 Estimated $y+$ statistics 45

5.1 Domain parameters for dynamic simulation 63

Nomenclature

Abbreviations

Abbreviation	Definition
sdf	signed distance function
BVH	Bounding Volume hierarchy
DFS	Deapth First Search
FSM	Finite State Machine
BDIM	Boundary data immersion method
LES	Large Eddy simulation
RANS	Reynolds Averaged Navier Stokes
LEI	Leading edge inflatable (kite)
IQR	Inter Quartile Range
Std	Standard deviation

Symbols

Symbol	Definition	Unit
N	Number of grid cells	-
M	Number of mesh elements	-
p	pressure	-
d	signed distance	cells
i	Index used for BVH	-
$\overline{x}$	A point in the grid	-
$\overline{c}$	The center of a triangle	-
$\overline{n}$	The normal vector of a triangle	-
L	Length	$\#$ of cells
Re	Reynolds number	-
U_∞	Free stream velocity	-
ν	kinematic viscosity	-
C_D	Drag coefficient	-
C_L	Lift coefficient	-
C_Y	Side force coefficient	-
C'_D	Fluctuations in Drag coefficient	-
C'_L	Fluctuations in Lift coefficient	-
C'_Y	Fluctuations in Side force coefficient	-
$y+$	Dimensionless walll distance	-
A	Area	$cells^2$
Q_{crit}	Q-criterion	$cells^2$
α	Angle of attack	$^\circ$
δ_1	Offset for measuring pressure	-
δ_2	Offset for extrapolating pressure	-
ϵ	Thickness of transition region	-
μ_0	Kernel moment	-
ρ	Density	-
ν	viscosity	$\frac{cells^2}{timestep}$

1

Introduction

The global shipping industry is a key enabler of international trade, with over 80% of goods transported by sea as of 2021 [63]. Its environmental impact is a growing concern, with the sector responsible for about 3% of global carbon emissions [30]. Regulatory pressure is increasing. The International Maritime Organization (IMO) adopted a revised greenhouse gas strategy that targets at least a 40% reduction in emissions by 2030 relative to 2008 [59].

Meeting these targets requires a portfolio of measures. Bouman et al. provide a comparative overview of options across design, operations and energy carriers [6]. Among renewable auxiliaries, wind assisted systems are again viable at scale. Rigid sails, Flettner rotors, vertical turbines and kites are all being deployed. Reported savings for kite assistance exceed 30% in favorable conditions [36]. Kites have been studied in both the energy sector [15, 17, 26] and in maritime applications [29, 43].

Aerodynamic force prediction remains central. Methods based on potential flow with empirical three dimensional corrections [65, 43, 13], RANS [39], and fast vortex based solvers [8] capture mean loads well, but they do not resolve the unsteady turbulence that appears at Reynolds numbers above 10^5. These fluctuations matter for control and structural assessment. For this reason we use Large Eddy Simulation (LES) in the *WaterLily* solver [68]. *WaterLily* is based on the Boundary Data Immersion Method (BDIM) [67] and runs on a Cartesian grid. The method requires a signed distance function (SDF) to define the fluid–solid interface.

This thesis focuses on the fast and robust construction and use of SDFs for complex triangulated bodies. The main contribution is a Bounding Volume Hierarchy (BVH) accelerated pipeline that reduces the number of triangle distance evaluations, together with a traversal strategy and inside–outside test that remain stable for non watertight meshes. We couple this SDF to BDIM for body submersion in *WaterLily*, and we add sampling procedures that recover surface pressure, shear and integrated forces directly on the original mesh. The result is an end to end workflow that turns a complex mesh into a simulation ready SDF, advances the flow with implicit LES, and returns time resolved load signals at practical cost. Where relevant, we reference standard SDF construction and reinitialization tools [33, 4, 19, 57, 44, 50, 53, 71, 47] to position our choices.

Kite propulsion remains as a motivating example rather than the main goal. We use the TU Delft LEI V3 geometry [1] to illustrate the workflow and to demonstrate how the accelerated SDF, BDIM coupling and force extraction behave across angles of attack. The case shows why fast SDF generation and reliable mesh based measurements matter when moving from mean load models to time resolved predictions.

The structure of the thesis is as follows. Chapter 2 reviews aerodynamic modeling methods for wind assisted propulsion and the role of SDFs in immersed methods. Chapter 3 presents the BVH based SDF pipeline, traversal and sign assignment, and the mesh based force and pressure extraction. Chapter 4 describes the simulation setup and validates the workflow on static test cases. Chapter 5 demonstrates the full workflow on a dynamic case. Conclusions follow in Chapter **??**.

2

Literature Review

2.1. Emission goals for the future and kites as a solution

The global shipping industry is a key enabler of international trade, with over 80% of goods transported by sea as of 2021 [63]. However, the environmental impact of shipping has become a critical concern, as the sector is responsible for approximately 3% of global carbon emissions [30]. Figure 2.1 shows how for the past 50 years the emissions have only increased. This has led to increasing regulatory pressure, including the International Maritime Organization's (IMO) revised GHG reduction strategy, which aims to cut emissions by at least 40% by 2030 relative to 2008 levels [59].

These are ambitious, but necessary goals and will require effort and adaptations from the shipping industry. Many different methods are already present and have been extensively tested. E.A. Bouman [6] collected and compared the current literature to compare different CO_2 reduction methods.

Among the various strategies to achieve these reductions, wind-assisted propulsion has re-emerged as a viable solution. Specifically, kite-assisted propulsion offers significant fuel savings and CO_2 reduction potential, with some studies estimating a decrease in greenhouse gas emissions of over 30% under optimal conditions [36]. This is a very promising result compared to other reduction methods as seen in Figure 2.2. Despite these promising benefits, existing research lacks a detailed understanding of the aerodynamic forces and control mechanisms required for effective implementation.

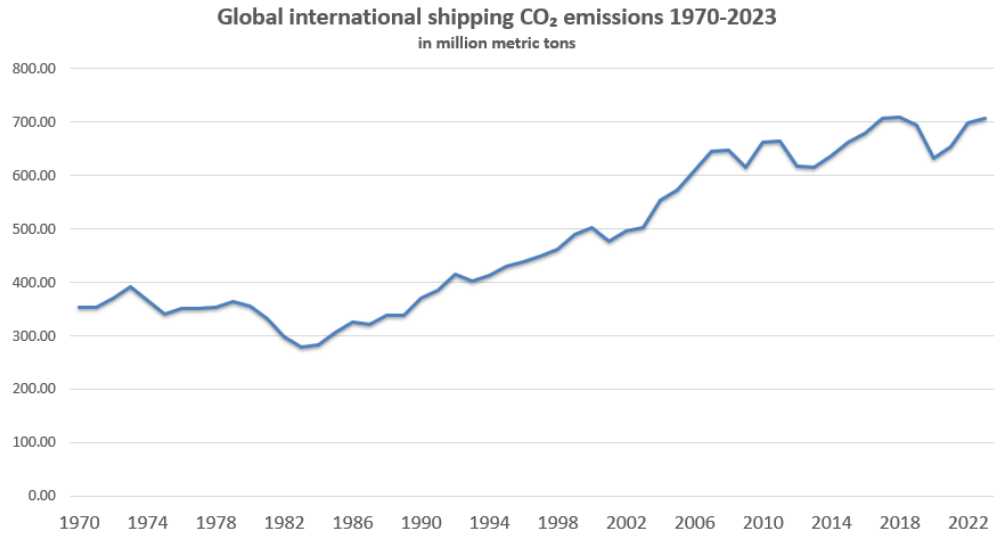


Figure 2.1: Global international shipping CO_2 emissions 1970-2023, data plotted from Statista [14]

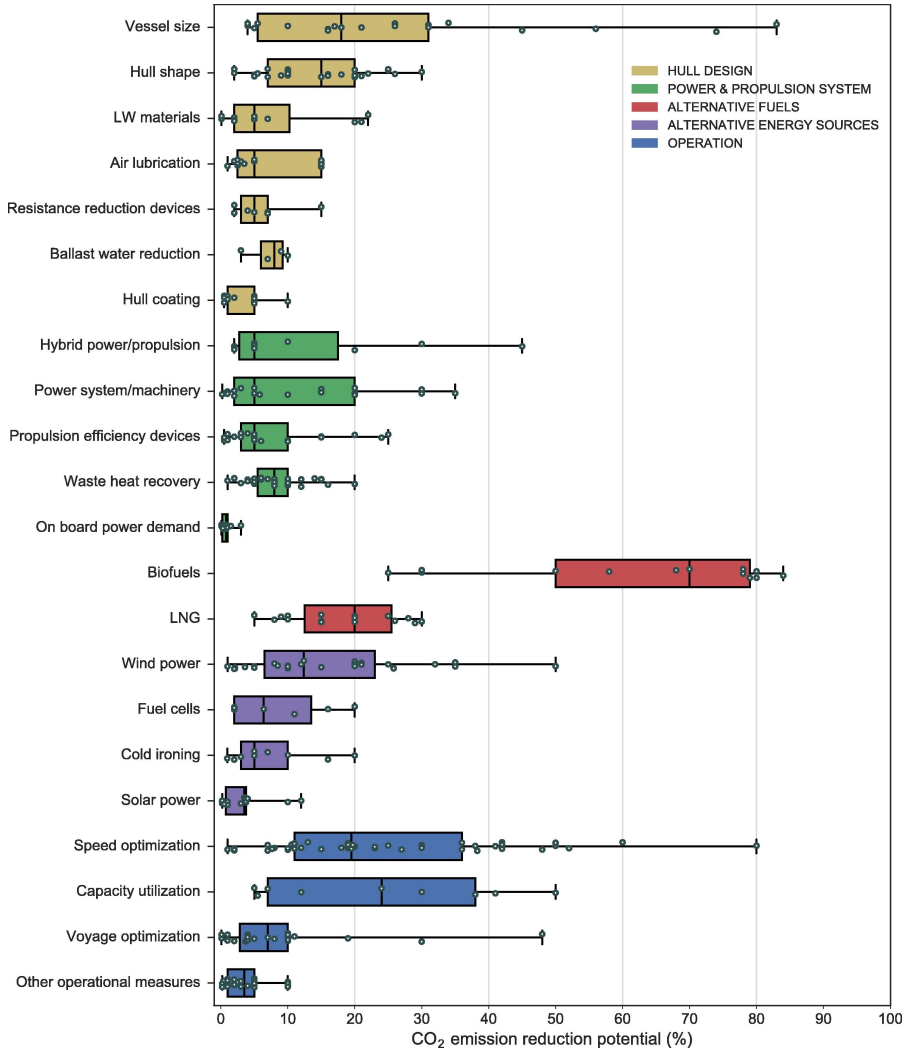


Figure 2.2: CO_2 emission reduction potential from individual measures, classified in 5 main categories of measures [6] .

This study aims to address this gap by conducting a high-resolution Large Eddy Simulation (LES) to investigate the aerodynamic behavior of kites under realistic operating conditions. The findings will contribute to improving kite-based propulsion systems, enabling better integration with commercial shipping operations.

Wind assisted shipping has been around longer than fossil fuel based transport, but it could once again become a viable option to reduce future CO_2 emissions. As the possible fuel savings may lie in excess of 30% [36], mainly due to the fuel saving during the trip. This means that wind assisted shipping might also be more economically viable depending on it's use [61]. In 2007 Naaijen assessed the performance of auxiliary wind propulsion using kites [43]. In this paper the drag and lift force of the kite are calculated from the 2D cross-section using Prandtl's lifting line theory and a 3D correction factor is used to obtain more accurate results. As expected the fuel saving is greatly dependent on the wind speed and the true wind direction which can be seen in Figure 2.4. Wind speeds are very seasonally dependent and thus the potential power saving could vary a lot. A route between the west coast of the US and Asia could expect ≈ 17 kn of wind in January, but only ≈ 12 kn on the same route in October [73]. Nevertheless, Naaijen's research shows that kites are a viable option as an auxiliary propulsion mechanism.

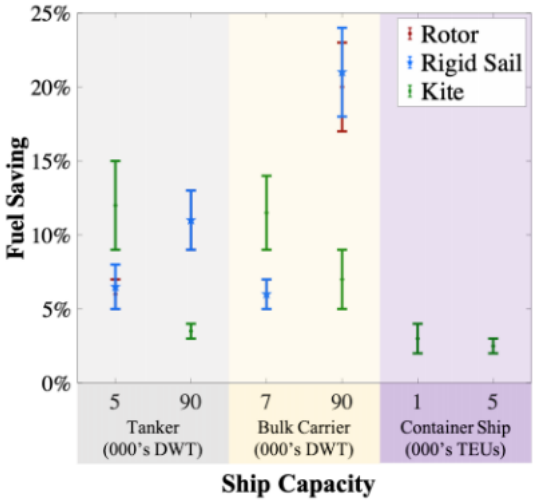


Figure 2.3: Potential fuel savings thanks to wind-assisted ship propulsion [36]

2.2. Details and research on kite systems

Naaijen's research focuses mainly on showing the potential fuel savings, but does not include details about the kite system itself. We will now look farther into the current research and past findings regarding kite systems. It is already known for a long time that a figure of eight will provide more propulsive power due to the increased kite velocity [13]. This figure of eight shown in Figure 2.5 is the basis for a lot of the current research as this dynamic flight is a lot more complex than a kite flying in a steady state. The figure of eight was initially adopted to avoid tether twisting, as the figure of eight detangles itself after every loop. A more recent paper by Eijkelhof, Rossi and Schmehl in 2024 comparing the two flight patterns found that while the circular flight path provides a higher average lift force, the figure of eight provides a lift force over time with lower peaks (a lower peak-to-average ratio) [15]. In order to make circular flight paths viable a complex system needs to be designed that prevents the twisting of the tether lines and control cables going up to the kite. Another possible flight path is referred to as the sine wave, during the flight he kite is moved up and down so that it resembles a sine wave from an earth fixed reference point, this particular flight path proves well suited for upwind sailing [13].

The figure of eight flightpath was extensively tested by Dadd, Hudsen and Shenoi [13]. By calculating the onset velocity of the kite and making use of the kites known drag and lift coefficient (previously established in this scenario) the total propulsive force is calculated. Testing different parametrized figures of eight they found that increasing the pole circle size as well as increasing the pole circle separation reduces the total towing force of the kite. This means that a tighter flying pattern is preferential, but a tighter turning radius might also mean that smaller scale fluid effect will be of more importance.

Control The figure of eight is a difficult to achieve flight path, thus a lot of research is aimed on the design of control systems. Kites are typically controlled with flying lines attached to a control box that is flying with it [12, 43, 54]. The tractive forces on the lines are often calculated based on the estimated drag and lift and velocity of the kite and used in combination with a PID controller [29]. The problem is already partially solved as there are currently several systems in place [2, 54], they are however still in the initial testing phase. SkySails, one of the systems currently being deployed, uses a control system designed by Erhard [16]. The steering deflection for a LEI kite is achieved by reeling in the steering lines on one end and providing more slack on the other side. This only deforms the kite slightly and the main effect is a roll motion around the kites local roll axis, which changes the direction of the total aerodynamic force vector as seen in Figure 2.6. However the aerodynamics used are substantially simplified. Based on experimental data they gathered the paper implements a simple turn rate law that is only dependent on the steering deflection, the onset velocity and a proportionality constant g:

$$\dot{\psi}_m = g_k v_a \delta$$

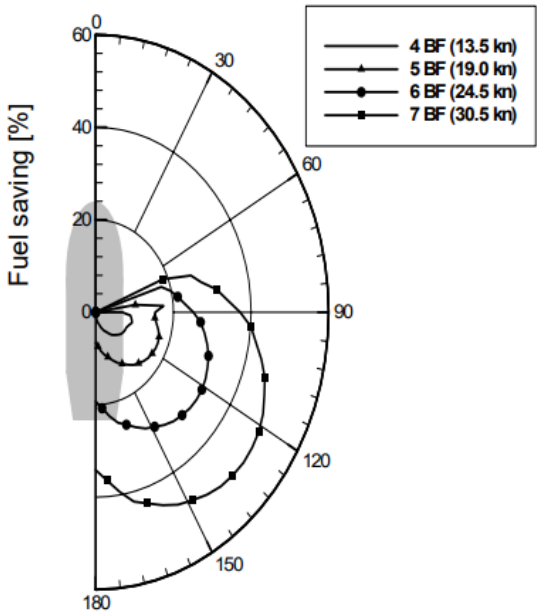


Figure 2.4: Fuel savings of a 50 000 DWT tanker using a propulsive kite [43]

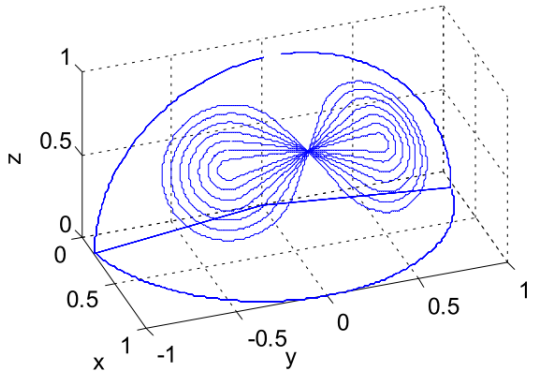


Figure 2.5: Parametrization of figures of eight with different pole circle sizes [13]

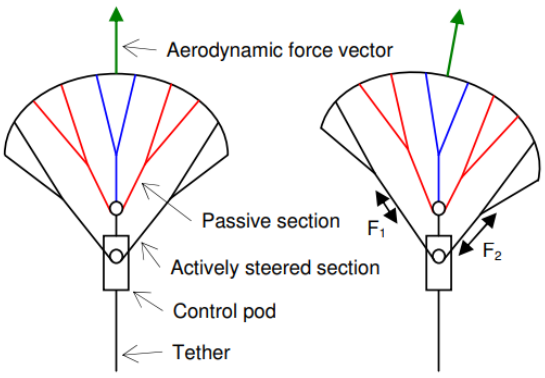


Figure 2.6: Schematic drawing of SkySails steering system. Deformation is generally reduced to a minimum and profile ribs are almost not affected [18]

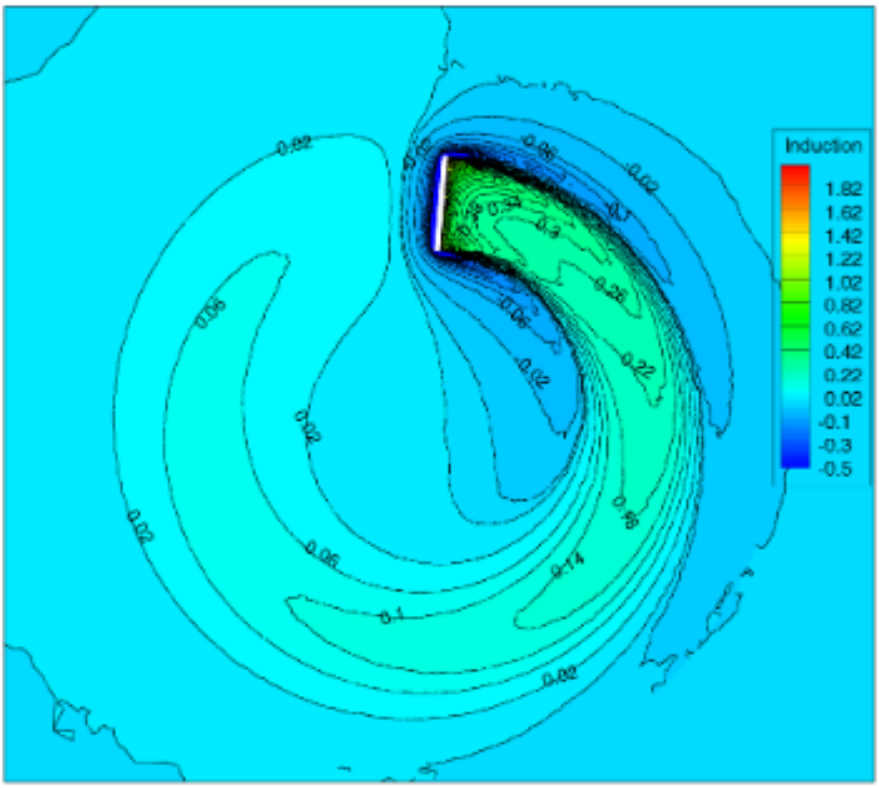


Figure 2.7: Contour plots showing the axial induction factor distribution for an instant of a cycle at the kite location. The white rectangular region in the plots represents the kite [35].

Fluid structure interactions Aside from control a lot of effort is aimed at optimizing the kites structure, since the FSI's are very complex due to the strong coupling between the deformation and the aerodynamic loading. This led to the development of a fast aero-structural model based on the vortex step method that can be used in the early design phases [8]. While the vortex step method gives better results compared to classical lifting line theory it does not calculate the full loading over the kite surface or account for small scale flow features. Another method proposed by Thedens [60] does solve for the full flow field using a viscous-inviscid interaction method. This method calculates the potential flow solution around the kite and couples it with an integral method to calculate the boundary layer. This is especially useful in higher Reynolds number flows with thin boundary layers. The authors improved the existing methods by improving the handling of separation and reattachment. While the method solves for a quasi-steady situation and not for dynamical flight it has the main advantage that it can now calculate the more complex deformations that occur near the attachment points of the lines.

Numerical methods All research presented here is highly dependent on the calculations of the forces on the kite, which highlights the need for an accurate force prediction method. A lot of research uses Prandtl's lifting line theory [45], but this is only a very rough estimations based on lifting line theory and a correction factor for the neglected three dimensional effects. Other methods are RANS or URANS simulations. A steady state RANS simulation is able to predict the average air flow around the kite in its steady state, but is by design unable to account for dynamical or unsteady effects. Using transition models increases the accuracy and makes it possible to better predict boundary layer transition region as shown by Folkersma [20].

A large scale Unsteady RANS simulation of a kite like structure with the aim of developing an analytical model for calculating the downstream wake features was carried out by Khieri in 2022 [35, 34]. For his simulation the URANS equations are closed by the k-ω model and the kite is flying a circular, prescribed path. Khieri finds that the induction factor is higher than 1 near the edges of the kite and the author notes this may indicate a flow reversal The full field can be seen in Figure 2.7. This is the most detail

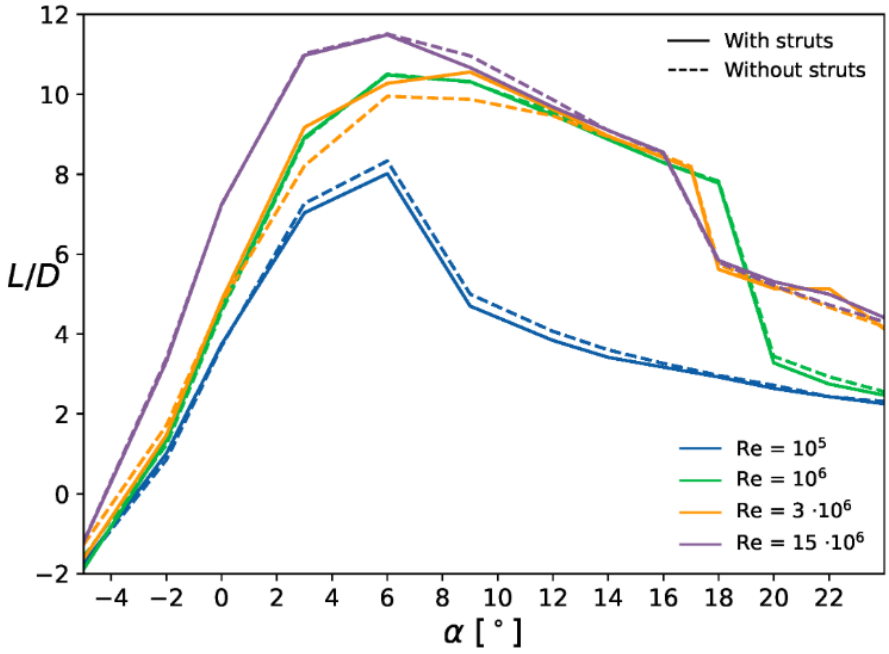


Figure 2.8: Lift-over-drag ratio as a function of the angle of attack without side-slip [39].

the URANS method is able to simulate as it is still calculating an average flow field and only models the effect of turbulence on the flow. As a result, it cannot resolve the intricate small-scale structures and flow instabilities that could contribute to the observed flow reversal.

Lebesque [39] performed several RANS simulations of the TU Delft LEI V3 kite. The paper is able to calculate the lift and drag profile of the kite and find the optimal Lift over drag ratio. The L/D results are shown in Figure 2.8. The RANS results are able to provide accurate results, but can never actually calculate any fluctuating forces acting on the kite. At the high Reynolds numbers we expect to see a lot of turbulence that will affect both the pressure distribution on the kite as well as the total force.

As a lot of research uses different kite models a summary of the used kites is presented below, methods and Reynolds numbers is presented. It is clear a large amount of different methods have been used to describe kite dynamics at a range of different Reynolds numbers. As seen in table 2.1 a range of different kite models is currently used. Cherubini [9] made an overview of kites currently used in the power generation sector. He identifies six different model and shows that that the leading edge profiles are used the most. The geometry of the kite is often based on extruded wing profiles, which is why lifting line theory is often used. Naaijen [43] used an extruded NACA profile, making it a foil kite. A more common profile is the leading edge inflatable profile. Here the profile exists of a large inflated tube at the leading edge whereas the rest of the kite is simply a canopy. The ram-air kite from Skysails [54] shown here is the same model used by their marine propulsion department.

Author	Kite type	Method	Re_{c,U_∞}
Naaijen & Koster (2007) [43]	NACA 4416	Lifting lines	5.25×10^6 - 11.85×10^6
Eijkelhof, Rosse & Schmehl (2024) [15]	MegAWES	Predefined	4.52×10^6
Dadd, Hudsen & Shenoi (2011) [13]	Flexifoil Blade III	Lifting lines	2.57×10^6 - 4.34×10^6
Cayon, Gaunaa & Schmehl (2023) [8]	LEI v3	Vortex step	–
Thedens ,de Oliveira & Schmehl (2018) [60]	Ram-air kite	Viscous-inviscid	11×10^6
Folkersma, Schmehl & Viré (2019) [20]	LEI	RANS-SST	100×10^3 - 50×10^6
Kheiri, Victor, Karakousian & Bourgault (2022) [35][34]	Clark-Y airfoil	URANS-SST	3.1×10^6
Haas, De Schutter, Diehl & Meyers (2019) [26]	AWES	LES (ASM)	10×10^6
Lebesque (2020) [39]	TU Delft LEI V3	RANS	1×10^5 - 1×10^6

Table 2.1: Overview of parameters in current research

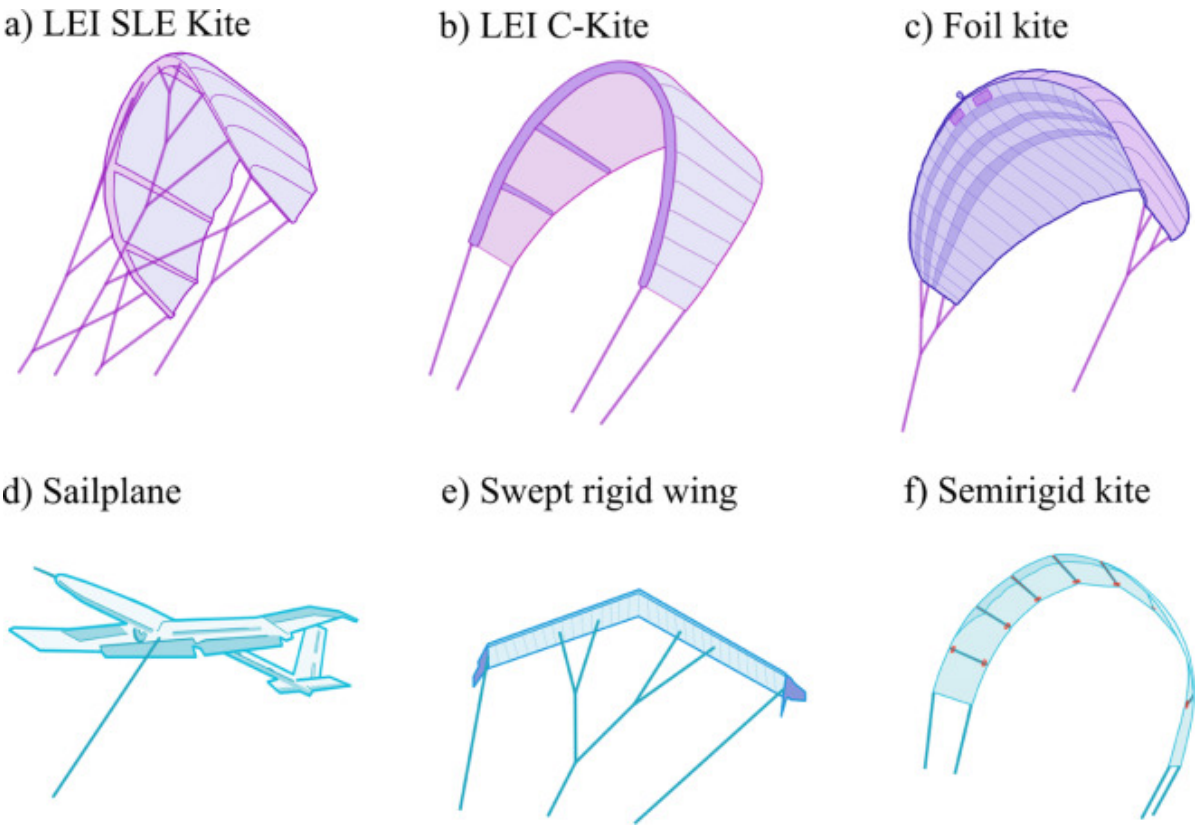


Figure 2.9: Different types of aircraft in Ground-Gen systems. (a) LEI SLE (Leading Edge Inflatable, Supported Leading Edge) Kite; (b) LEI C-kite; (c) Foil Kite, design from Skysails;(d) Glider, design from Ampyx Power; (e) Swept rigid wing, design from Enerkite; (f) Semi-rigid wing, design from Kitegen. [9]

2.3. Large Eddy Simulations

As discussed RANS and URANS methods are used to simulate average flow fields, they already provide a better result compared to the lifting line theory, but they have their own set of limitations as described by Bush [7]. These limitation have lead to the adoption of higher accuracy methods such as DNS and LES. Direct numerical simulation is a very powerful tool, but it is extremely computationally expensive making it unpractical for engineering purposes. The computational cost of the simulation is dependent on the smallest scales in the fluid, the Kolomogorov scales, the grid resolution and time steps need to be small enough to capture the these scales. Current research by Yang (2021) shows that to total computational cost for a DNS scales with $Re_{L_x}^{2.91}$ [72]. DNS is therefore used mainly for research of turbulence itself on small scales and generally small Reynolds numbers. An example of such might be turbulent flow over rough surfaces, a paper by Modesti, Sathyanarayana et al. [42], showing the capabilities of DNS to obtain valuable data on very small scales which can be used as a basis for models used on larger scales or validation purposes.

Compared to DNS large eddy simulations are cheaper and regarding accuracy lie between RANS and DNS. Yang [72] also finds that wall resolved LES and wall modeled LES scale with factors of $Re_{L_x}^{2.71}$ and $Re_{L_x}^{1.14}$ respectively. A full review of the basics of LES modeling is described by Sarkar [51]. The idea of LES stems from the turbulent kinetic energy spectrum. Turbulent kinetic energy is generated at the largest scales, transferred down to the inertial range and finally dissipated at the smallest scales. LES imposes a cutoff filter somewhere in this spectrum and models the scales below this filter. The cutoff can be explicitly imposed by using convolution filters, but in finite volume methods the grid size acts as a cutoff range as scales smaller than a grid cell can never be resolved. The energy spectrum below the cutoff is then typically modeled using SGS-models such as the Smagorinsky model [55] which was the first Eddy viscosity model for large eddy simulations. The model is numerically stable but is dependent on the Smagorinsky constant C_s for which the optimal value varies a lot depending

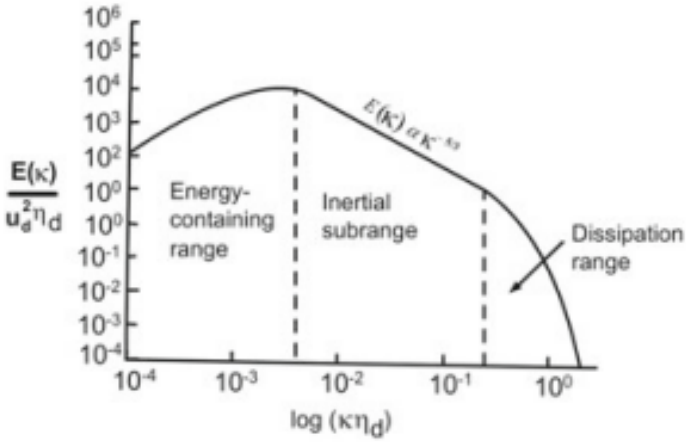


Figure 2.10: Turbulent kinetic energy spectrum showing the different energy ranges [51]

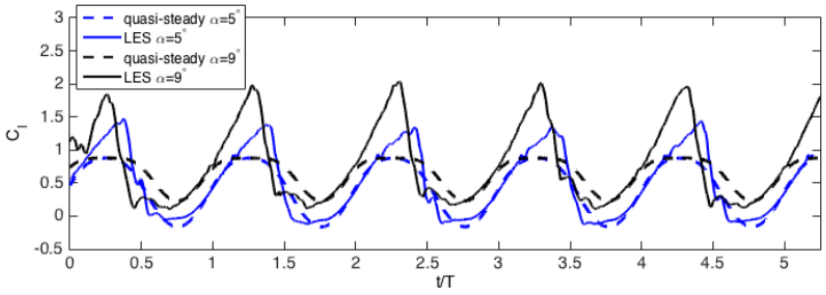


Figure 2.11: Phase-averaged spanwise vorticity over a heaving plate by Franck & Breuer [21], contour levels: -10 (blue) to 10 (red)

on flow conditions. Several other models exist such as the dynamic Smagorinsky or scale similarity models.They are widely used and tested in various test cases [31]but it is also possible to model the SGS tensor implicitly. This method, appropriately called implicit LES, models the SGS from the terms emerging from the discritization scheme. Margolin [37] showed that if the right discritization schemes are used the error resembles the divergence of the subgrid scale tensor.The method has the advantage that it is easier to implement and that it is not dependent on externally defined parameters that might invalidate results when employed incorrectly.

Constantinescu & Squires [11] provide an example of how LES and DES can be used to to analyze oscillating drag forces on a simple cylinder. Franck & Beuer used LES in 2017 to analyze the flow over a heaving flat plate showing how LES can be used to resolve small flow features and oscillating force responses. This can be clearly seen as they analyze the fluctuating part of the lift coefficient over two cycles of heaving as seen in figure 2.12.

A type of LES has already been performed on kites by Haas [26], focusing on the wake field of an array of flying kites. However the study does not calculate the flow around the kite, rather it uses the actuator set method [3]. This method, initially developed for wind turbines, is an adaptation of actuator disk models where the loads of the rotor, or here kite, are added to the momentum equation using a disk in the mesh. The ASM enhances this method by consolidating the forces not on a disk but on lines representing the rotors or kite. While this method is capable of calculating the downstream wake of a kite it can never solve the flow near the kite.

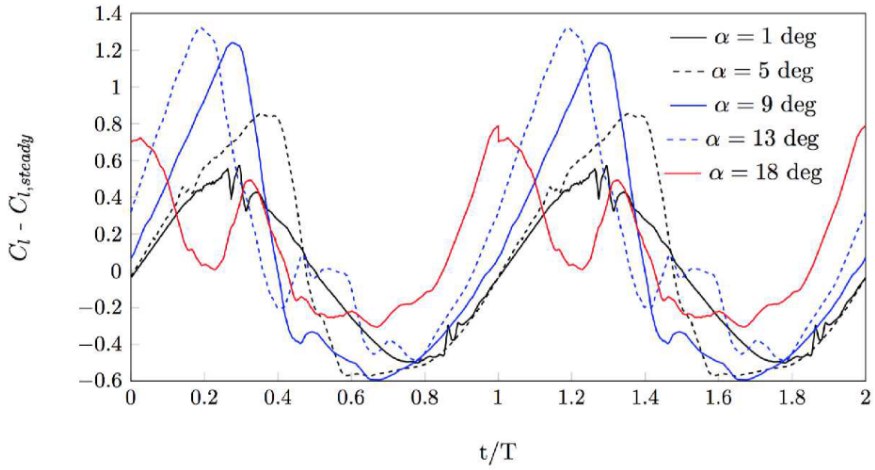


Figure 2.12: Phase-averaged lift coefficient with the steady lift coefficient subtracted for better comparison. Two heaving cycles are shown to better display the trend that occurs upon the transition of one cycle to the next. [21]

Large Eddy Simulation resolves the large, energy containing motions of turbulence and models the smaller subgrid scales that the mesh cannot capture. In a filtered Navier Stokes view, the resolved field obeys transport equations with an extra subgrid stress term. That term must be modeled to represent the net effect of unresolved motions on the resolved ones. Classical closures use an eddy viscosity idea, for example the Smagorinsky model.

Dynamic procedures estimate the model coefficient from the flow itself. Germano, Piomelli, Moin and Cabot [24] introduced an identity that compares stresses at two filter levels to infer the coefficient from the resolved field. Lilly [40] proposed a least squares modification that improves stability and performance. Together these papers established dynamic modeling as standard practice for inhomogeneous and wall bounded turbulence.

A central caveat for LES is the interaction between numerics and modeling. Ghosal [25] analyzed how dispersion and dissipation from discrete schemes can contaminate subgrid dynamics and bias energy transfer if the grid and the scheme are not chosen carefully. In practice it is better to use conservative and low dispersion discretizations, apply dealiasing where appropriate, and verify energy budgets and inertial range spectra so that the model, not truncation error, governs dissipation.

At very high Reynolds numbers, resolving the near wall region is often not feasible. Wall modeled LES (WMLES) replaces the near wall resolution with a model for wall stress or heat flux while resolving the outer layer. Piomelli and Balaras [46] reviewed equilibrium and non equilibrium wall models and their performance on canonical cases. Bose and Park [5] surveyed modern WMLES practice, including two layer zonal approaches that work on unstructured meshes and in separated or shock affected flows. These models can recover mean loads and key spectral features on coarser near wall grids.

Implicit LES (ILES) is another option. In ILES no explicit subgrid model is added. The carefully designed dissipation of a monotone or shock capturing scheme acts as the subgrid mechanism. Margolin, Rider and Grinstein [37] explained how scheme induced dissipation can mimic forward cascade behavior in high Reynolds number flows and discussed how to tune it for stability and accuracy. Garnier, Mossi, Sagaut, Deville and Comte [23] examined the use of shock capturing schemes in LES and the conditions under which their numerical dissipation is compatible with turbulence physics. When using ILES it is good practice to report grid to integral scale ratios, check spectral ranges and slopes, and compare transport budgets, so that the effective dissipation stays physically acceptable.

In summary, LES resolves the energy containing motions and models the rest. Dynamic subgrid closures and wall models extend its reach to complex, high Reynolds number flows. ILES offers a complementary path that uses numerical dissipation as the subgrid mechanism, provided that spectra and budgets are verified against theory and reference data.

2.4. Boundary Data Immersion Method

The Boundary Data Immersion Method (BDIM) embeds body boundary conditions into the governing equations by an analytic convolution across a thin kernel that spans the fluid solid interface. Weymouth and Yue [66] derived the method at the level of the continuous equations, then discretized. The result is a set of mixed equations on a single Cartesian grid that enforces the boundary data and keeps pressure and velocity behavior stable near the interface. Because the coupling is derived at the equation level, BDIM is less sensitive to body grid alignment than direct forcing strategies and shows robust convergence on canonical benchmarks.

A second order extension improves interface accuracy. Maertens and Weymouth [41] introduced a higher order correction that removes interfacial velocity gradient discontinuities and reduces spurious pressure fluctuations. They validated moving body and canonical flows up to Reynolds numbers of order 10^5. This makes BDIM attractive for unsteady near body aerohydrodynamics where full resolution of turbulence is not feasible.

BDIM is also available for compressible flow and aeroacoustics. Schlanderer, Weymouth and Sandberg [52] formulated a compressible version that keeps the same kernel based embedding of boundary conditions and enables coupled prediction of near body flow and radiated sound on Cartesian grids. This supports simulations of moving and rotating bodies that produce broadband noise, while keeping time advancement stable and the near body coupling accurate.

For high Reynolds number applications BDIM is typically used together with LES or with ILES. In the ILES setting the regularization from the numerical scheme, together with the local filtering implied by the BDIM convolution near the interface, provides the effective subgrid dissipation. The guidance from Margolin, Rider and Grinstein [37], from Garnier, Mossi, Sagaut, Deville and Comte [23], and from Ghosal [25] applies here. It is good practice to document mesh and kernel parameters, show grid convergence of forces and spectra where feasible, and check energy budgets so that the modeled or implicit dissipation stays consistent with turbulence theory.

To summarize, BDIM provides an equation level, Cartesian grid coupling between fluid and solid that enforces boundary data without ad hoc forcing. The second order extension improves accuracy near the interface, and the compressible variant extends the method to flow acoustics. These properties make BDIM a practical near body method that fits well with LES and ILES for high Reynolds number problems where fully resolved simulations are out of reach.

2.5. Signed distance functions

A signed distance function $\phi(\mathbf{x})$ gives the distance from a point to a surface and uses the sign to mark inside or outside. By definition $\phi = 0$ on the interface and $|\nabla\phi| = 1$ near it. Jones, Baerentzen and Sramek [33] give a clear overview of why SDFs are useful in simulation and geometry processing. In our context the value is practical. The zero set locates the interface, $\nabla\phi/|\nabla\phi|$ gives a normal, and point queries are cheap.

There are three common ways to build an SDF. The first starts from a triangle mesh and computes closest point distance, then assigns a sign. Baerentzen and Aanaes [4] show how the angle weighted pseudonormal yields a robust inside outside test together with the closest point. Wu, Man and Xie [70] propose a double layer strategy that builds interior and exterior distance layers and combines them, which is efficient and avoids many non watertight issues.

The second way is grid based. One marks interface cells on a Cartesian grid and runs a distance transform. Felzenszwalb and Huttenlocher [19] give linear time transforms for common metrics. The result is an unsigned field on the grid; the sign then comes from an inside outside test such as parity counting or a winding number. Jones, Baerentzen and Sramek (2006)[33] discuss these choices and their trade offs.

The third way is PDE based. Sussman, Smereka and Osher [57] introduce reinitialization that relaxes a generic level set to signed distance while keeping the interface fixed. Peng, Merriman, Osher, Zhao and Kang [44] make this local and fast near the front, and Russo and Smereka [50] improve accuracy of the distance reconstruction. When the front advances monotonically, fast marching gives an efficient

solution of the eikonal $|\nabla\phi| = 1$ away from the interface, as shown by Sethian [53].

Near walls and contact lines the details matter. Della Rocca and Blanquart (2014)[49] show that naive reinitialization near a contact line can create errors and propose a fix. Xue, Sun, Adriaenssens, Wei and Liu [71] give a finite element reinitialization that uses a shifted boundary idea to restore the signed distance property without moving the zero set.

3

Implementation of mesh based bodies in WaterLily

This section will discuss the development, validation and benchmarking of the MeshBody package. All methods heavily rely on the Boundary Data Immersion method that is used in *WaterLily* so this chapter will begin by explaining the BDIM and the other methods used with in the current framework. Next the methodology of the Bounding Volume Hierarchy will be laid out. The generation of signed distance fields will then be validated an compared to a brute force approach. We will also outline a new method for calculating the forces acting on the body. This method will work from the mesh instead of the full flow field. This method will also be validated and it's shortcomings highlighted.

3.1. Boundary data immersion method

In this section we will explain the BDIM method and how it is used to submerge objects in *WaterLily*. This is important as we want to implement a new method to submerge these objects. We will also discuss how the current forces on submerged objects are calculated. What follows is an overview of the Boundary Data Immersion method and is a summary of a collection of papers (Weymouth $\&$ Font, 2024 [68], Weymouth $\&$ Yue, 2011[66] and Maertens $\&$ Weymouth,2015 [41])

The BDIM is a method to submerge object in a fluid simulation that only relies on a Cartesian grid and does not rely on conformal meshes. Instead of explicitly enforcing boundary conditions at the fluid–solid interface, BDIM smoothly blends the governing equations of the fluid and the solid through a narrow transition band around the body surface. This is achieved by replacing the separate equations for the fluid, solid, and interface with a single meta-equation:

$$M^\epsilon(W) = d_B^\epsilon B(W) + d_F^\epsilon F(W) + d_S^\epsilon S(W) = 0$$

where:

- W is the state variable,
- $B(W)$,$F(W)$ and $S(W)$ are the governing equations for the body, the fluid, and the interface condition, respectively,
- d_B^ϵ,d_F^ϵ and d_S^ϵ are kernel-weighted moments that smoothly blend the respective equations across the interface region.

The weights are derived from the signed distance function $d(x)$, which measures the distance from each point x in the domain to the nearest point on the surface of the immersed body.The transition region is defined by a smoothing width ϵ, we will use $\epsilon = 1$ making the transition region 2 cells wide, see Figure 3.1. The kernels are calculated from the signed distance by a cosine function and is defined only at a

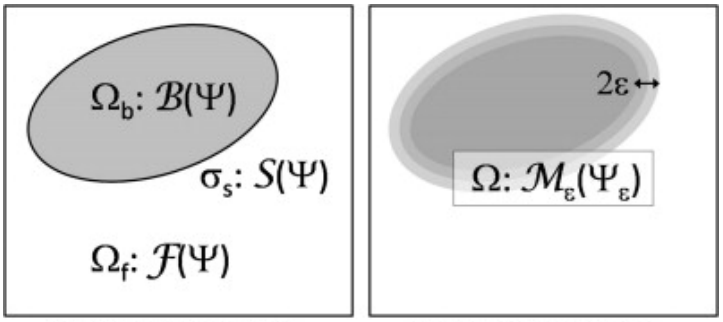


Figure 3.1: Conceptual sketch of the problem. Figure (a) is the original two-domain problem P0 with domains Ω b and Ωf and their interface σs and the respective governing equations and Figure (b) is the equivalent single-domain problem Pϵ with governing meta-equation constructed by the boundary data immersion method by convoluting and assembling the equations over a width ϵ. The transition between the equations is smooth and located within the small distance ϵ of the interface σ ; Taken from Weymouth and Yue, 2011 [66]

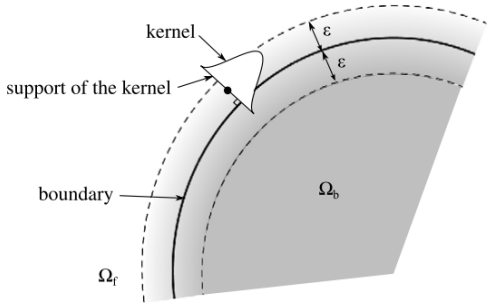


Figure 3.2: Smoothing across the immersed boundary. The equations valid in each domain are convolved with a kernel of radius ϵ and added together. The gradient of gray illustrates how the contribution of b_ϵ and f_ϵ to the smoothed equation changes in the boundary region. The kernel at a point (marked by a dot) that belongs to the boundary region is represented; Taken from Maertens & Weymouth [41]

distance $s = \frac{d}{\epsilon} \in [-1, 1]$ the kernel is defined by:

$$kern(s) = \frac{1}{2} + \frac{1}{2}\cos(\pi s)$$

From this the zeroth and first kernel moments are calculated:

$$\mu_0(d, \epsilon) = \frac{1}{2} + \frac{1}{2}\frac{d}{\epsilon} + \frac{1}{2\pi}\sin(\pi\frac{d}{\epsilon})$$

$$\mu_1(d, \epsilon) = \epsilon\left[\frac{1}{4}\left(1 - \left(\frac{d}{\epsilon}\right)^2\right) - \frac{1}{2\pi^2}\left(\frac{d}{\epsilon}\sin(\pi\frac{d}{\epsilon}) + 1 + \cos(\pi\frac{d}{\epsilon})\right)\right]$$

A visual representation of how this kernel looks is shown in Figure 3.2. These kernel moments define how the behavior inside and outside are blended together. Inside of the body, $\mu_0 \to 0$ which is used to suppress any pressure gradient. In the fluid domain where $\mu_0 \to 1$ the equations are fully ”activated”. In the transition region then simply uses the intermediate values of μ_0, removing the need for explicit boundary conditions.

While BDIM is very efficient, it requires computing the signed distance from each grid point to the immersed surface,d. For simple parametric bodies, this is trivial. However, for complex geometries such as kites, it becomes computationally expensive. In this thesis, an accelerated method is developed to evaluate signed distance fields more efficiently, which is essential for the practical use of BDIM in high-resolution simulations.

In order to obtain the forces on the object we need to integrate the pressure over the surface of the body, approximated by the kernel moments. In the BDIM, the force contribution from each grid cell is calculated by multiplying the normal of the closest point on the object by the value of the kernel function at that location an example of the smoothed z component of the normals can be seen in Figure 3.11.

$$f(x) = p(x)\overline{n}(x) \cdot kern\left(\frac{d(x)}{\epsilon}\right)$$

Only locations on the grid with in the transition region contribute to the force as the value of $f(x)$ is zero farther away. The total force on the object is then sum of $f(x$ over the entire domain Ω_ϵ:

$$F_p = \sum_{x \in \Omega_\epsilon} p(x)\overline{n}(x) \cdot kern\left(\frac{d(x)}{\epsilon}\right)$$

3.2. Methodology for the Bounding Volume Hierarchy

As we have established how the BDIM works and why the signed distance function is important we will now describe the method used to decrease the computational cost of doing so. In order to reduce the number of measurements a bounding volume hierarchy is introduced to split up the mesh and quickly determine whether or not a point is worth evaluating. This method is adapted from computer graphics where it is used for ray-tracing applications [27]. The method splits the spatial domain into boxes that are related in a parent/child structure and stores them inside of a tree as seen in figure 3.3. Starting at the top we check if a point is located inside of a box, if it does we continue by checking its children. If not, we check remaining siblings and possible children. Only if the point is located in a leaf (8 through 15 in figure 3.3) do we measure its distance to the mesh otherwise we do not need it and can set it to a large value. We can also reduce the number of triangles we need to check by only measuring to triangles located inside of the leaf that the point is located in.

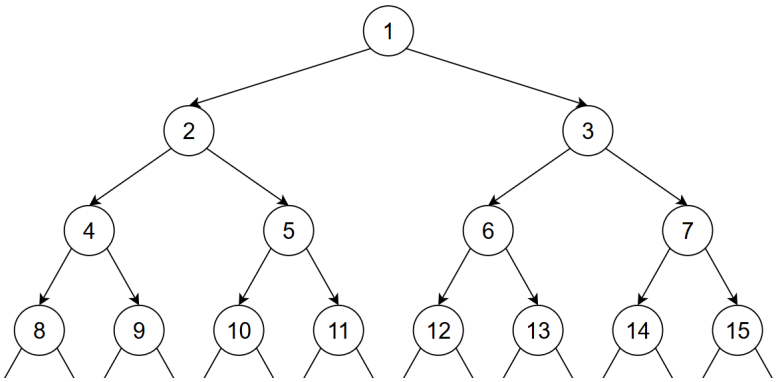


Figure 3.3: General tree lay-out witrh 4 levels; 1-7 are nodes, 8-15 are leaves

There are three main steps in building the BVH, the first involves working down and subdividing the domain. The second step splits the mesh into parts corresponding to the subdivisions and re-snaps the bounding boxes to fit closer to the mesh parts. Finally the third steps rebuilds the nodes of the tree.

BoundBox type and type-specific operations In order to easily work with the code a custom type was defined in Julia called *BoundBox*. This element consist of vectors describing two diagonal points on a box stored as keywords lo and up. This type is used for all instances of boxes and the syntax allows for easy to read logic and easy manipulation of the boxes. We define three distinct box-operations. First expanding a box only requires subtracting an adding a scalar to *BoundBox.lo* and *BoundBox.up* respectively. Splitting a box in half requires calculating the widths of a box, which is the element wise subtraction of the two vectors, halving the desired direction and calculating the new diagonal points. A new box can then be constructed using these values. Finally reconstructing a parent box out of two

boxes is simply taking the minima of the box.lo's and the maxima of the box.up's. Figure 3.4 shows a 2d representation of the box-operations.

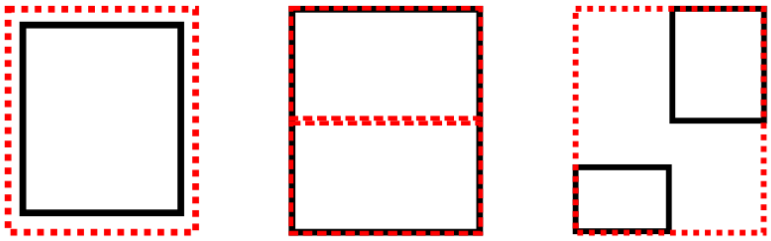


Figure 3.4: Box operations; in order from left to right: expanding, splitting into children, merging into parent

Generating Subdivisions starts by generating a bounding box around the mesh an slightly expanding it so that there is a border of a couple cells around it. This is important as we need the signed distance field on both sides of the mesh in order for the kernel function to work properly. The box is then split along its longest direction, this is done iteratively until the desired amount of leaf boxes is created. These subdivision are then slightly expanded so they overlap in order to reduce error. It might happen that the closest mesh point is located just outside of the box and having overlap between the boxes removes this problem. A benefit of generating the subdivisions from the top down is that we know for certain that two boxes that are close to each other in space will also be close to each other in the memory array. This is beneficial when rebuilding the nodes as we do not need to add any logic determining whether boxes are clos too each other or not. The resulting subdivisions for a tree with 5 levels can be seen in Figure 3.5 where we apply to process to the Stanford bunny [62].

Splitting the mesh The subdivision are then used to split the mesh in sub-meshes based on the locations of the three vertices defining a triangle. If one of the vertices is inside of the subdivision we add it to the sub-mesh. There are cases where this does not work such as a large triangle element that intersects a subdivision, but has all three of its vertices outside of the subdivision. For our use case we assume that in general triangles will be a lot smaller than the subdivision. Since the subdivision overlap so do the sub-meshes to reduce the chance of missing the closest point. Since leaves that are largely empty will result in a lot of unnecessary measurements, bounding boxes fitting the sub-meshes are generated and these are then used as the leaves. In some cases a subdivision might be completely empty and the bounding box will have an invalid lo an up. It is possible to remove these boxes and construct an incomplete tree such as by Chitalu et al. [10]. However this complicates traversal, requiring a map between the tree nodes and the memory index. Instead we leave the tree structure intact and define that a point can never be inside such a box and is thus not measured. The final leaves can be seen in figure 3.6 and an example of a sub-mesh is shown in green in figure 3.7

Constructing the nodes Finally the upper nodes are constructed from the leaves by merging two boxes that are next to each other in space to avoid overly large nodes. Because we build the tree such that leaves next to each other are next also together in the memory we can use boxes that are located next to each other in the memory array. Merging two boxes that are far apart does not induce additional error, but will result in more boxes being checked then necessary leading to a longer traversal time. This process is repeated for each level until the tree is filled. During this process all boxes are expanded to make sure there is a solution all around the mesh. Two leafs and their constructed parent node are shown in figure 3.7

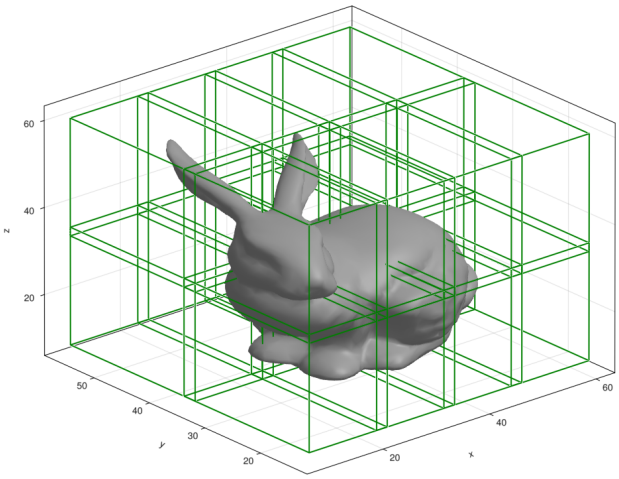


Figure 3.5: 16 subdivisions on the Stanford bunny from a 5-level BVH tree, expanded to show overlap

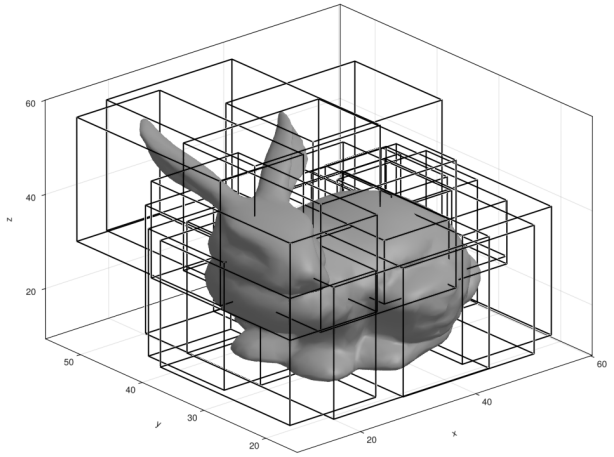


Figure 3.6: Leaves around the Stanford bunny for a BVH with 5 levels

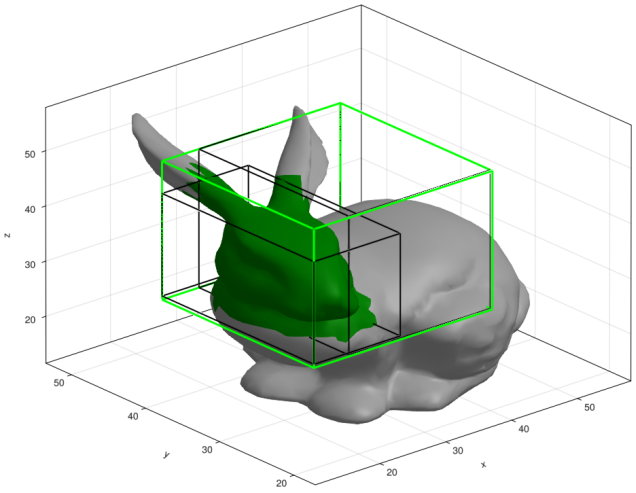


Figure 3.7: Two leaves (black) and their constructed parent node (green); part of the mesh inside the left leaf is shown in dark green

3.3. Traversing the tree

When constructing the signed distance field we will need to traverse the tree for every grid point. We will only measure a distance if the point we are measuring is located inside of a leaf. Traversing the tree consists of checking whether a coordinate, x, is located inside of a node. If it does we continue by checking the children of that node, this is repeated until we find one, more or no leaves in which the point is located. In order to traverse the tree we need to be able to jump between boxes based on the relationship between them. There are three possible relationships: *Child*, *Parent* or *Sibling*. These are relationships for points that are only one operation away from each other, but farther relationships can be achieved by using the operations multiple times.

Children The left child of a node is found by doubling the current node index. The right child is then simply its sibling. Example using figure 3.3: The children of node 7 are $2 \cdot 7 = 14$ and $2 \cdot 7 + 1 = 15$

$$Children(i) = 2i \quad \& \quad 2i + 1$$

Parent The parent of a node is defined as the greatest integer less than or equal to half of its index.Example using figure 3.3: The parent of node 7 is $\lfloor \frac{7}{2} \rfloor = 3$

$$Parent(i) = \left\lfloor \frac{i}{2} \right\rfloor$$

Sibling Finally the sibling of a node is found by adding or subtracting 1 depending on whether the current index is even or uneven. Example using figure 3.3: The sibling of node 7 is $7 + 1 = 8$.

$$Sibling(i) = \begin{cases} i + 1 & \text{if } i \text{ is even} \\ i - 1 & \text{if } i \text{ is odd} \end{cases}$$

We implemented and tested two different traversal algorithms. The first, a classical depth first traversal and the second a stack-less method that implements a finite state machine. Below both methods will be described. It is difficult to determine what method is best and it depends on the compiler. The FSM-based method is ready to run on a GPU whereas for the stack based method this is not possible as GPU's do not allow dynamic memory. Adding elements to an array is generally expensive and this would make the stack based method more expensive. However modern compilers like Julia handle dynamic memory extremely efficient and don't always have to allocate new memory when pushing new values to an array. In practice we have seen that both methods perform equally.

3.3.1. Depth first, stack based traversal

This method starts by creating a stack that only contains the root index, 1 and initializing a best solution, set to a large value. We then take the first index in the stack and check whether the point we are checking, x, is located inside of it. If so the children of the box are pushed into of the stack, if not we check the remaining nodes in the stack. Only when we find x to be inside of a leaf do we measure the distance to the corresponding sub-mesh. If that solution is better than the one previously found (or set) we update the best solution and continue working through the stack until it is empty. Since we have overlap between the leaves the point can be in multiple leaves ands we have to continue, as there is the possibility that another leaf will give a better solution. This process is repeated until the stack is empty and we can accept the best solution. If no leaves are found we just return the initial large value for the distance and the point was not measured at all. A pseudo-code representation of the algorithm is shown in the algorithm below.

Figure 3.8 shows a possible traversal for a simple 3 level BVH with arrows showing the order in which the nodes are checked. In this example the point is located in both leaf 7 and 5 so we have to measure twice. In this small scenario all boxes are checked, but in larger BVH's this is not be the case.

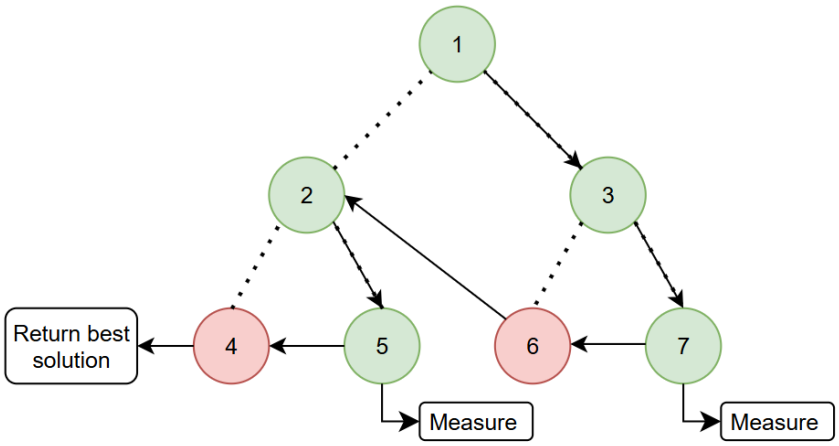


Figure 3.8: Stack based traversal for a 3 level BVH; A green node indicates that the point is inside of the box and a red one indicates that it isn't

Algorithm 1 Stack based, DFS traversal

```
function Traverse(x, BVH)
    Initialize stack with root index 1
    Set best_solution to a large initial value
    while stack is not empty do
        i ← pop(stack)
        box ← bounding box at node i
        if x is inside box then
            if i refers to a leaf node then                    ▷ Leaf node found
                d ← GetDist(x, mesh[i])
                if |d| < |best_solution| then
                    best_solution ← d
                end if
            else                                               ▷ Internal node
                Push children 2i and 2i + 1 onto stack
            end if
        end if
    end while
    return best_solution
end function
```

3.3.2. Stack-less, finite state machine traversal

The second method is adapted from [27].They have devised a finite state-machine that is able to traverse a BVH-tree. Their method is designed for ray-tracing it can be easily adapted and even simplified for our case. Algorithm 2 shows the detailed layout of the method. The method works by defining a state based on the relationship between the current node and the previous node. Three states are defined based on the previous node that was checked; `fromParent`, `fromChild` and `fromSibling`.

If the state is `fromChild` we already know that the point x is inside of the current box so we do not have to check this again. This means we either have to move to a sibling or farther up if the sibling has already been checked. Hapala [27] defines a near and far child depending on the direction of the ray. Since we are only testing points and not rays we can simplify the process just checking whether the node index is even or odd. From an even node index we move to the sibling and for an odd index we move to the parent. This distinction is clear in Figure 3.9 with nodes 2 and 3.

In the `fromSibling` case we have to test whether we are inside of the box as it is the first time at that

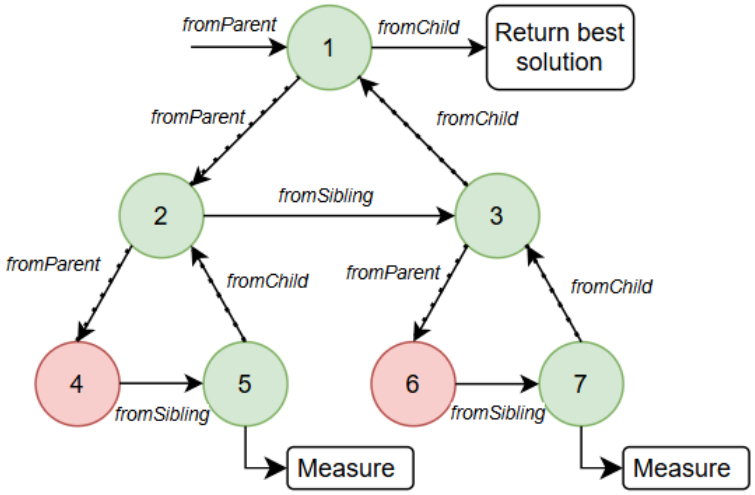


Figure 3.9: FSM based traversal for a 3 level BVH; A green node that the point is inside of the box and a red one indicates that it isn't

index. If not we leave the branch and move back up to the parent. If we are in the box we check if we are in a leaf and update the solution if necessary. If we are in a node we continue down the tree following the left branch.

The final state `fromParent` requires checking if the point is inside of the box. Here we deviate from the method by Hapala et al [27], we do want to check the first bounding box and if we are not inside of it we immediately stop the traversal. If we are not in the box, but not checking the first box we move to the sibling. If we are in the box we check if it is a leaf and if so we measure the distance and compare it to the current best to see if we accept it. If we are in an internal node we move down the tree to the left child.

This method has to check the same amount of boxes as the stack based method. While it looks like we are accessing some internal nodes multiple times, the state machine is able to tell whether a box has already been checked based on the state and does not need to check the point again. Figure 3.9 shows how node 2 is checked once when we enter it coming from node 1. When we get to node 2 again from node 5 we do not have to check it again, rather we see that it is a left child and simply move to its sibling, node 3.

Algorithm 2 Stackless BVH Traversal by Hapala et al. [27]

```
function Traverse(x, BVH)
    current = 1
    state = fromParent
    while true do
        Switch (state) do
            Case fromChild
                If current == root → Return
                Else if current == leftChild(Parent(current))
                    → current = Sibling(current)
                    → state = fromSibling
                Else
                    → current = Parent(current)
                    → state = fromChild
                Continue

            Case fromSibling
                If inBox(x,current) == FALSE
                    → current = Parent(current)
                    → state = fromChild
                Else if isLeaf(current) == TRUE
                    → MEASURE
                    → current = Parent(current)
                    → state = fromChild
                Else
                    → current = leftChild(current)
                    → state = fromParent
                Continue

            Case fromParent
                If inBox(x,current) == FALSE AND current ==1
                    Break
                Else if inBox(x,current) == FALSE
                    → current = Sibling(current)
                    → state = fromSibling
                Else if isLeaf(current) == TRUE
                    → MEASURE
                    → current = Sibling(current)
                    → state=fromSibling
                Else
                    → current = leftChild(current)
                    → state=fromParent
                Continue
    end while
end function
```

3.4. Constructing sdf's using the BVH

The final thing we need to do is measure the distance from a point to a triangle. For this two algorithms are in place. The first one can be used when the triangles of the mesh are small relative to the grid cells and simply calculates the distance from point $\overline{x}$ to the center of the triangle. This simple distance calculation is very fast and thus preferable if it proves to give correct results.

The second algorithm is more robust but is also computationally more expensive. If triangles are large compared to grid cells we need to check which part of the triangle is closest and use that distance. This can either be one of the three vertices,on one of the three edges or x can be located inside of the triangle. When the closest part of a triangle is found we use that distance. For our use case we have chosen for the fast method as we still assume that mesh elements will be small. When measuring the distance between $\overline{x}$ and a sub-mesh we simply loop over all triangles and store the best result[1].

An added complexity are non-bounding meshes, these are meshes that do not bound of a partition in space. A sphere is the best example of a bounding mesh as it partitions space into inside or outside the sphere. A simple plate or the canopy of a kite can be considered a non-bounding mesh as there is no clear in or outside. When constructing the sdf we have to slightly inflate it around the mesh by subtracting a desired half thickness from the measured distance as we need a clear in or outside on order for the solver to work. This minimum thickness is $2 + \sqrt{3}$ where two is twice the length of the transition zone ,2ϵ and $\sqrt{3}$ is the maximum length of a vector in a cell. This thickness is thus defined such that the transition zones do not overlap and the kernel vectors at the edge of the transition can never point to another transition zone. Not respecting the minimum distance leads to wholes in the geometry.

For the force calculation we also need the normal direction of the closest point on the mesh through the domain. The already implemented force routing uses this field of normal vectors to calculate the pressure force. While we will be implementing our own force routine later in section 3.7, we still want the implemented force method to work as intended. Once the closest triangle to a point has been found we take the normal vector of that triangle and store it for position $\overline{x}$. This works for bounding meshes, but for non bounding meshes it is too simple. Since we simply take the normal of the closest mesh element this means that at both sides of the mesh the same normal is used This can be seen in figure 3.10 where the z-component of the normal is shown for a low resolution kite at a high angle of attack. Both sides of the canopy have normals facing downwards which will lead to wrong force calculations. In order to solve this we need to know which side of the mesh we are on and flipping the normal vector accordingly. This is done by multiplying the normal with:

$$sgn\left(\overline{n} \cdot (\overline{x} - \overline{c})\right)$$

Implementing this leads to the z components of the normals in figure 3.11 which shows a kite at a zero angle of attack and where we can clearly see the difference between the top and bottom side of of the canopy.

[1] These methods were already implemented in the MeshBody.jl

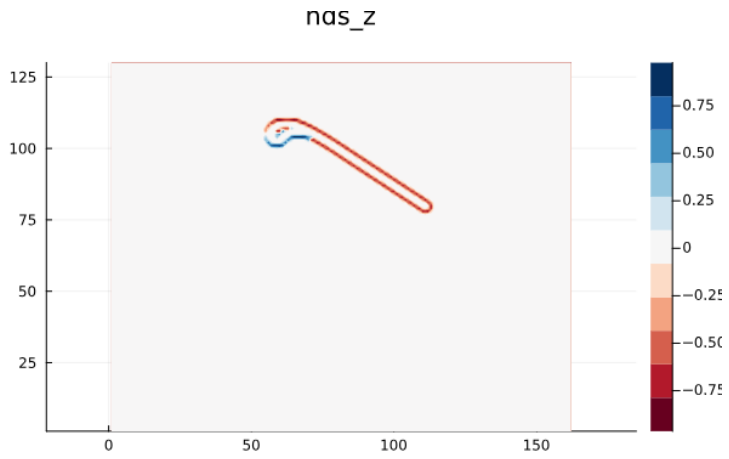


Figure 3.10: z component of wrongly calculated normal vectors; Both sides of the back canopy have normals pointing in the same direction

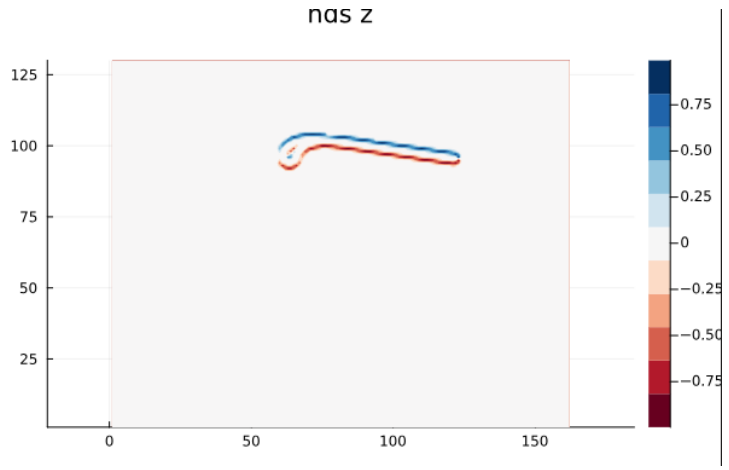


Figure 3.11: z component of correctly calculated normal vectors; Both sides of the back canopy have normals pointing in different directions

3.5. Validation of signed distance functions

In order to validate the method we need to compare it to a know signed distance field that is sufficiently complex. We use the sdf library by Quilez [48] and used the exact sdf for a link. Making use of the sdf transformations we generated a signed distance field of two interlocked links, where one is rotated $90°$. This geometry was chosen as it is also easy to replicate the shapes with a mesh. A slice of the analytical solution can be seen in figure 3.12. Four different meshes with a varying amount of triangles are generated and are scaled to accommodate three different grid sizes. Five BVH's with a different number levels are then constructed and the signed distance field is calculated for each case. The L1 error norm of the sdf is only calculated close to the mesh as we have not calculated the full field. Statistical data of the error is collected and below we report the mean error to study the effects of the resolution and the amount of levels.

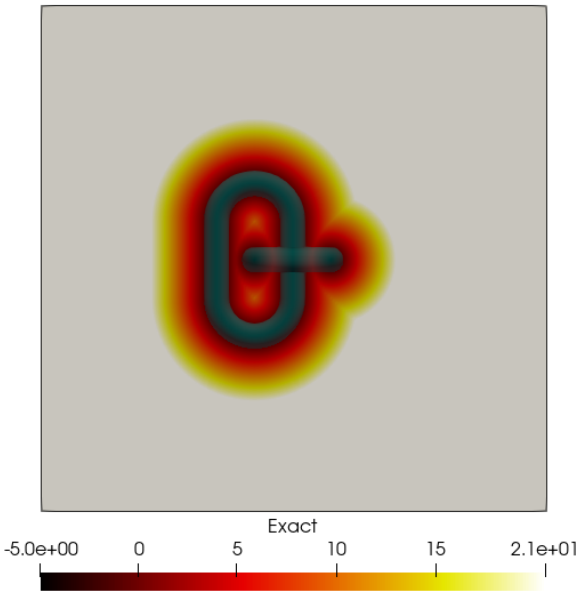


Figure 3.12: Slice of the analytical sdf of the two interlocked links

Table 3.1: Overview of levels, grid cells, and mesh elements used for validation

Levels	Grid Cells	Mesh Elements
2	50	3,156
4	200	11,220
6	800	28,871
8	–	41,404
10	–	–

First we look at the effect of the resolution of the mesh compared to the grid cells. This resolution is simple defined as the number of mesh elements over the number of grid elements. We see that the mean error quickly reduces when increasing the resolution and is generally very low. The main reason for the diminishing error is likely due to a decrease in modeling error. In other words increasing the resolution reduces the difference between the mesh itself and the exact shape of the links, thus also decreasing the error in the sdf.

We also look at the mean L1 error compared to the number of levels used. Figure 3.15 shows the distribution of the mean errors. Note that a box here is defined by all the mean errors of different combinations of the variables defined in table 3.1. We see that changing the number of levels has little to no impact on the error in the sdf. This means we can increase the number of levels to speed up calculations without an increased error in the sdf.

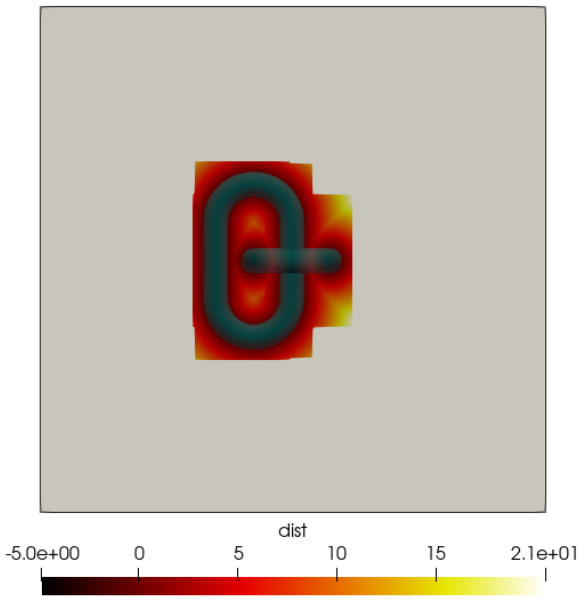


Figure 3.13: Slice of the numerical sdf of the two interlocked links made with 6 levels

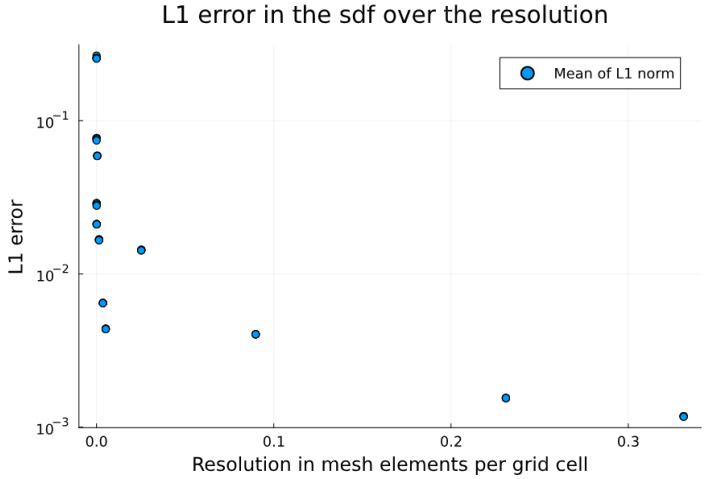


Figure 3.14: Mean L1 errors in the sdf of two interlocked links plotted over the resolution

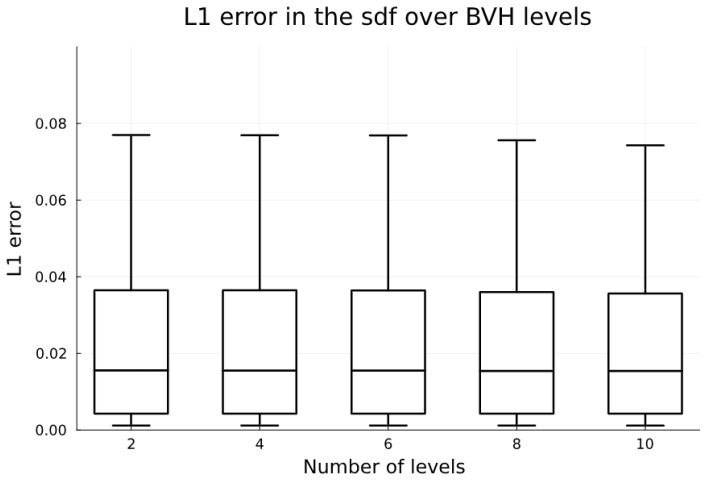
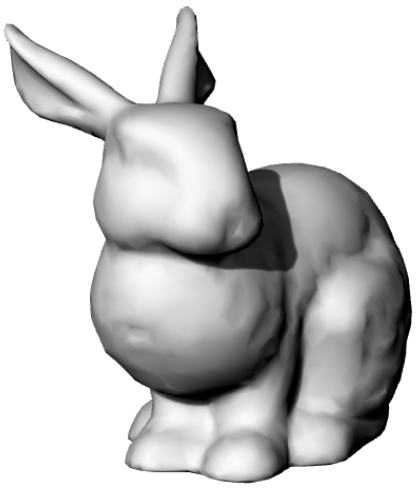


Figure 3.15: Mean L1 errors in the sdf of two interlocked links plotted over the resolution

3.6. Benchmarking of sdf generation

To analyze the effect of the method we do not make an actual speed comparison, rather we focus on the reduction in the number of operations. We define a measurement as finding the distance between a grid point and a mesh. This involves checking all triangles with in that mesh and finding the shortest distance. We will investigate three different meshes in 3 different domains, a sphere, the Stanford Bunny [62] and the Stanford Asian Dragon [56] seen in Figures 3.16a and 3.16b respectively. Each mesh will be placed in a 3 different domain sizes, a small domain of 128 by 128 by 128 cells, an intermediate domain of 256 by 256 by 256 cells and a large domain of 512 by 512 by 512 cells. The meshes will be scaled up together with the domain such that the volume ratio between the mesh and the domain remains the same. For each mesh a MeshBody using 4 different numbers of levels. The reduction factors are relatively independent on the size of the mesh inside the domain compared to other metrics. When the object becomes larger in the domain the BVH method will creat boxes that are completely with in the mesh. These boxes are inside of the mesh, but no actual elements are inside so we do not measure there leading to a ”hollow” sdf. We normalize all results here by the number of operations that would be necessary in a brute force approach and by the relative mesh size. In the brute force approach we need to check N grid points and for each grid point we need to check M mesh elements.



(a) Render of the Stanford bunny [62]

(b) Render of Stanford's Asian dragon [56]

Figure 3.16: Stanford bunny and Asian dragon used for benchmarking the BVH method and for validating the hydrostatic pressure routine

We compare the number of operations to a simple bounding box method. This can be seen as building a BVH with only one level so that a single box is used to check wheter or not we want to measure the grid point or not. We are only interested in the total number of times that the distance between a triangle and a point is calculated and we normalize the results to the number of triangles checked when using the simple method. This is simply equal to the volume of the bounding box times the number of mesh elements, $vol(BBox) \cdot M$. Figure 3.17 shows the reduction factor for a different number of levels and Figure 3.18 for different domain sizes. We see that the number of triangles reduces rapidly and for a large number of levels we can expect up to 90% fewer triangles checked. The effect does diminish due to the overlap between the leaves, once they become sufficiently small the overlap dominates their size and we are doing extra work. We see a slight added reduction when increasing the domain size, but not as significant as when increasing the number of levels. This is because when we increase the domain size the number of measurements reduces, but the effect on the number of triangles is more complex and depends a lot on the objects geometry.

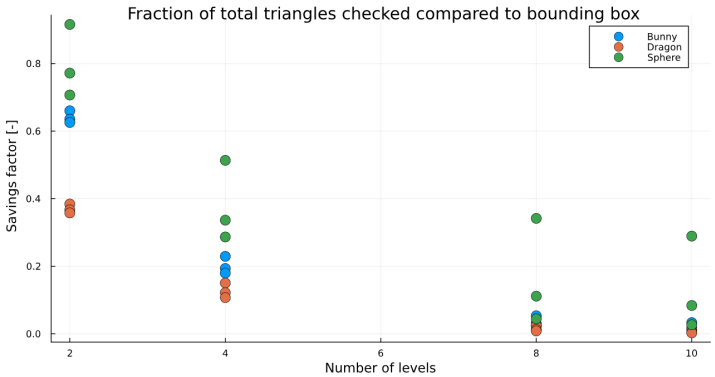


Figure 3.17: Total saving factor by level

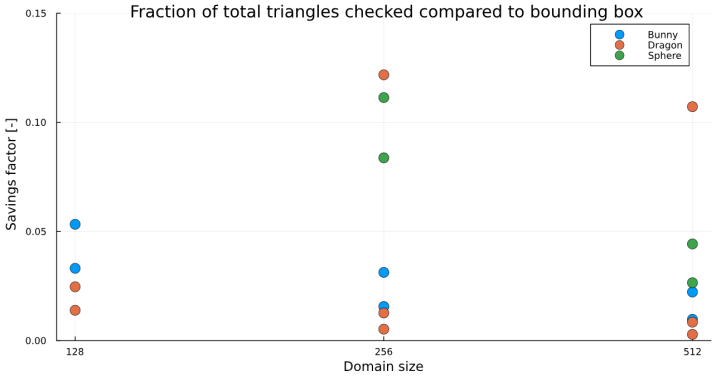


Figure 3.18: Total saving factor by domain size

Besides calculating the number of triangles that need to be checked we also time the calculation. We do this using large dragon meshes consisting of $\approx 1.1 \times 10^6$ and $\approx 2.5 \times 10^5$ faces. We time both the creation of the BVH and the time required to construct the sdf for different levels and sizes. The calculations are performed using 2 AMD Ryzen Threadripper PRO 5975WX 32-Cores so we have a total of 64 cores available. Since in the WaterLily framework the measuring is parallelized over the grid we take the same approach. Julia does not allow for multiple levels of parallelization, so there is no need to time our code using different numbers of threads as our code is not parallelized, but rather gets run in parallel. We use the same domain sizes as before and check for levels 3,5,7 and 9. For each data point 10 - 50 calculations are performed in order to obtain accurate results, however when analyzing the results the variance in the data is so small it does not show up clearly so it has been omitted here. In figure 3.19 we see the total construction time for the BVH. This is important as it is an extra step that is not required when using a simple bounding box approach. Therefor it is important that no time is lost on this step. First we see that the construction time is independent of the mesh scaling. The larger mesh (dragon_1) does take longer to process then the smaller mesh (dragon_5) this is due to the construction of the sub meshes which is an intensive process. We also see that the construction time increases with the number of levels. This can be attributed to the same reason, where now we need to construct more sub meshes. Overall the construction is extremely fast and as we will see now completely insignificant compared to the time required.

Figure 3.21 show the time spent constructing the sdf. As expected a larger mesh at a larger scale takes significantly longer. The first data point is the time needed to construct the sdf with a simple bounding box. We see that even using a small BVH of 3 levels can have a significant impact on the measuring time. Based on these numbers the construction time that is significantly less than a single second is acceptable. Besides the absolute time we are more interested in the time saved. Normalizing the time to the time required using a simple bounding box shows some interesting behavior. First we see that there is a clear optimal number of levels that depends on the mesh and domain size. The levels tested here all still result in a time saving, but it is clear that using too many levels might eventually even be worse than using the bounding box method. Secondly we also see that for both a larger mesh and a larger domain size there is a larger time savings. This is great as it means that of we encounter larger meshes and domains we can always increase the number of levels up to a certain point to save more time.

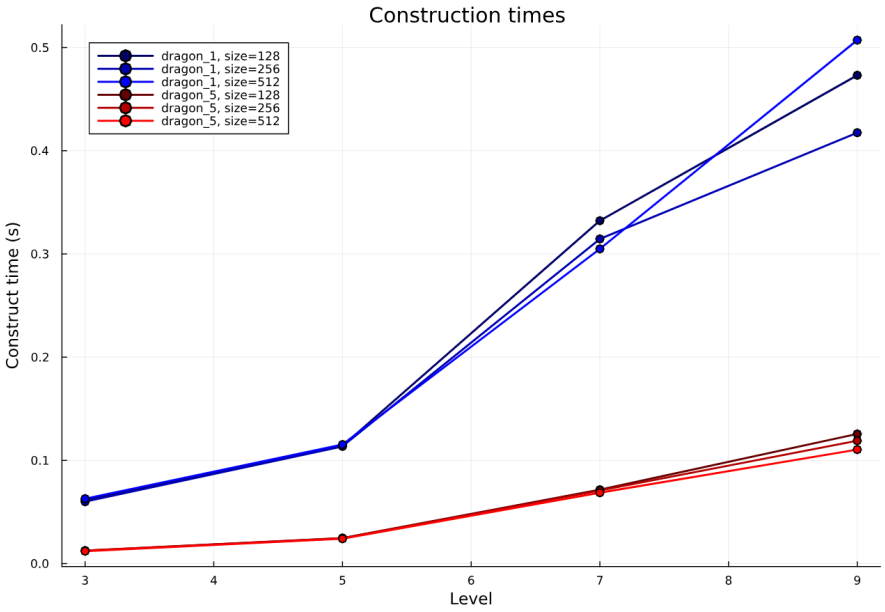
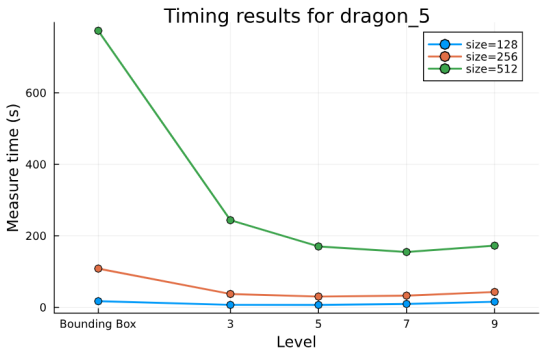
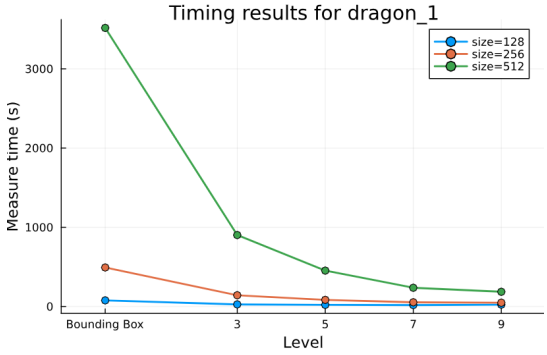


Figure 3.19: Construction times for different number of levels, mesh elements and domain sizes

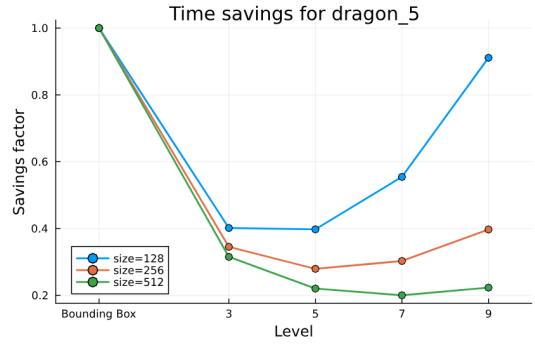


(a) Measurement times for dragon 5 ($\approx 2.5 \times 10^5$ faces)

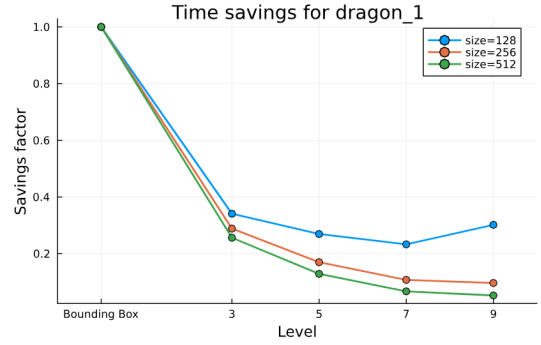


(b) Measurement times for dragon 1 ($\approx 1.1 \times 10^6$ faces)

Figure 3.20: Measurement times for two different mesh sizes, scales and levels



(a) Time saving factor for dragon 5 ($\approx 2.5 \times 10^5$ faces)



(b) Time saving factor for dragon 1 ($\approx 1.1 \times 10^6$ faces)

Figure 3.21: Time saving factor for two different mesh sizes, scales and levels

3.7. Pressure probing from a mesh

The current method to extract the pressure an friction forces on an object in *WaterLily*, outlined at the start of this chapter, requires looping over the entire computational domain and using the kernel to determine the approximate location of the submerged object. If we implement a mesh it will be more efficient to calculate the forces from this mesh as there will often be less mesh elements than grid cells. In this section we will discuss the method used to extract the pressures and forces on the mesh.

3.7.1. Implementation

In order to obtain the pressure forces on the body we need to account for the working of the boundary data immersion method. The method takes the problem consisting of two domains, inside the body and the fluid, and transforms it to just one body by applying a smoothing region of width 2ϵ on the interface. The method is described in more detail and validated by Weymouth and Yue (2011) [66] and figure 3.1 shows a general two dimensional example. The kernel function used for this generate a smooth transition between the inside and outside of the body. When we generate a body from a mesh we have a clearly defined boundary such as in figure 3.1(a) and the transition region extends away from the body. We need to take this transition region into account when we calculate the pressure acting on a mesh element.

The first step is to calculate the center point, c, and the normal, $\overline{n}$ of the triangle. Depending on the mesh that is used information of the normals might already be included, in that case these normals are used instead. We then obtain the pressure at a distance δ_1 and $(\delta_1 + \delta_2)$ from c in the direction of $\overline{n}$, resulting in two values for the pressure at points q_1 and q_2. Often q_1 and q_2 will not coincide with the grid points of the computational domain so a tri-linear interpolation of the pressure field, already implemented in *WaterLily*, is performed to find the pressures p_{q1} and p_{q2}. These values are then used to linearly extrapolate the pressure values back to c resulting in the following formula.

$$p_c = p_{q1} - \frac{p_{q2} - p_{q1}}{\delta_2} \cdot \delta_1$$

This formula gives us the extrapolated pressure at the center of a triangle which we can use to calculate the force on the triangle by multiplying it with it's area. This simplification relies on the condition that the mesh elements are smaller than the grid cells as we can then assume the simulated pressure is constant over the area of the triangle. Since the pressure in the solver is non-dimensional the exact unit of this force is $cells^2$ or simply an area. So when changing the force to a force coefficient we divide by the front facing area and are left with a non-dimensional force again.

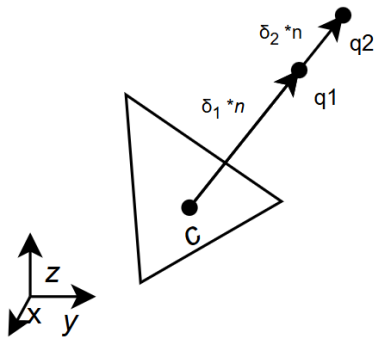


Figure 3.22: Vectors and points used to determine the pressure on a triangle element; $\overline{cq_1}$ is defined by $n * \delta_1$ and $\overline{q_1 q_2}$ is defined by $n * \delta_2$

This method only requires the determination of the parameters δ_1 and δ_2. The main consideration for δ_1 is to be outside of the transition zone, but still as close to the body as possible leading to a choice of $\frac{\epsilon}{2}$. Secondly we want δ_2 to be as small as possible because this brings the calculation closer to the derivative's limit, which assumes an infinitesimally small distance. However if δ_2 is too small, the difference between p_{q1} and p_{q2} may be too insignificant to capture any meaningful variation data, leading to a slope calculation that approaches zero and results in a value of p_{q1} at the center of the triangle. Experimentation resulted in an optimal value of 1e-8. Both points are still located in the transition zone, but the values where chosen for the best agreement with the existing method.

$$\delta_1 = \frac{\epsilon}{2} = 0.5$$
$$\delta_2 = 10^{-8}$$

3.7.2. Validation of pressure probing from a mesh

In order to validate the working of the force extraction we will start by submerging several objects in a linear pressure gradient representing hydrostatic pressure. The hydrostatic pressure force on the mesh will be calculated and compared to the volume of the mesh. The correct volume will be calculated using the volume tool in Rhinoceros 8. A domain of 64 by 64 by 64 grid cells will be created and the mesh will be placed in the middle of the domain and scaled to make make sure the object is fully inside of the domain. The applied pressure gradient follows $p(\overline{x}) = -\overline{x}_3$. Note that results are the same if the pressure gradient is applied in a different direction or if the sign is flipped. In table 3.2 we can see that the error is extremely low and near exact. While these cases give the impression the method works perfect we cannot validate the chosen values for δ_1 and δ_2 based on these results. Since a linear extrapolation of a linear function is exact (see derivation below) we end up with negligible errors and are not actually using δ_1 and δ_2 in the calculation as both cancel out in the equations.

$$\begin{aligned}
p_1 &= z_c + \delta_1 n_3 \\
p_2 &= z_c + (\delta_1 + \delta_2) n_3 \\
\Delta &= \frac{z_c + (\delta_1 + \delta_2) n_3 - z_c - \delta_1 n_3}{\delta_2} = n_3 \\
p_c &= p_1 - \Delta \delta_1 \\
p_c &= z_c + \delta_1 n_3 - n_3 \delta_1 = z_c
\end{aligned}$$

Object	**# of faces**	**Exact volume** $cells^3$	**Calculated Volume** $cells^3$	**Error %**
Sphere	79,600	17,153.4044	17,153.3665	0.000222 %
Stanford Bunny	5,002	9,165.9713	9,165.9704	0.000010 %
Stanford Asian dragon	1,125,000	2,410.2639	2,410.1892	0.003103 %

Table 3.2: Errors in comparing calculated hydrostatic pressure and volume

In order to validate the functionality of the proposed method and the corresponding values for δ_1 and δ_2 , a benchmark case is reproduced from the work of Weymouth and Lauber [69]. This recent study introduces novel boundary conditions to the WaterLily simulation framework. In order to evaluate the accuracy and robustness of our approach, we conduct simulations of a sphere in computational domains of varying sizes, and the drag force exerted on the sphere is measured over the course of the simulation. This is done using both the existing WaterLily implementation and the new mesh-based method under investigation.

The test sphere has a radius of 44 grid cells and is modeled using a high-resolution triangular mesh composed of 79,600 elements. The time evolution of the drag coefficients, as illustrated in figure 3.25, reveals that the two computational methods produce almost indistinguishable results across the entire duration of the simulation. This close agreement is a strong indicator of the mesh-based method's validity.

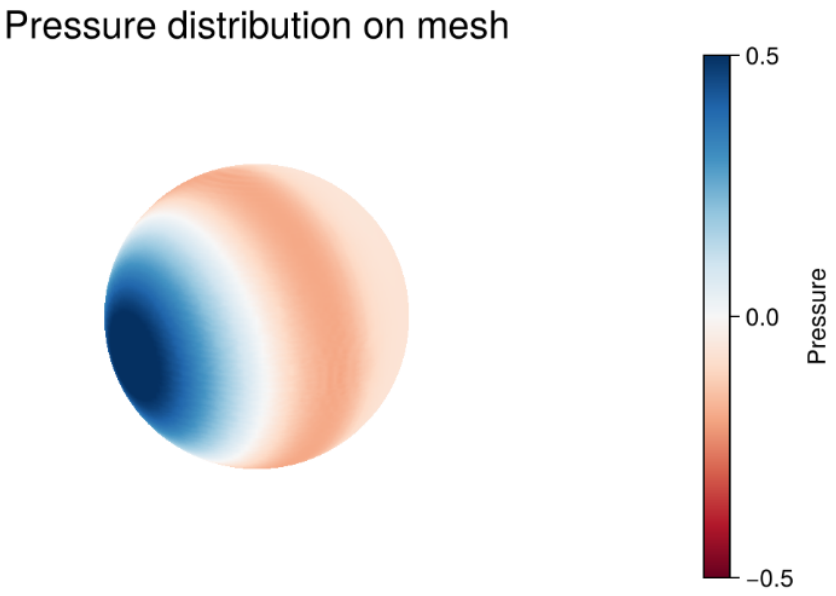


Figure 3.23: Pressure distribution on a spherical mesh

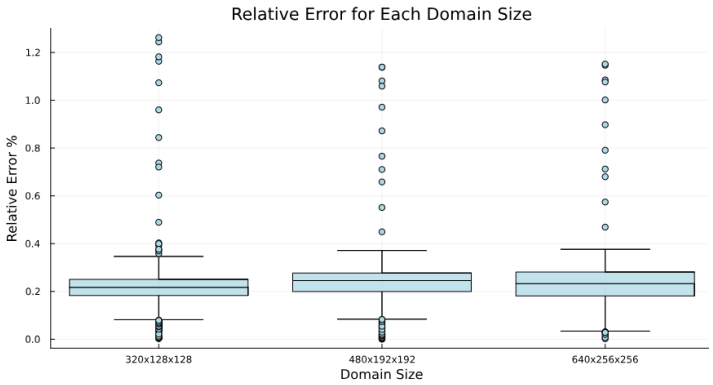
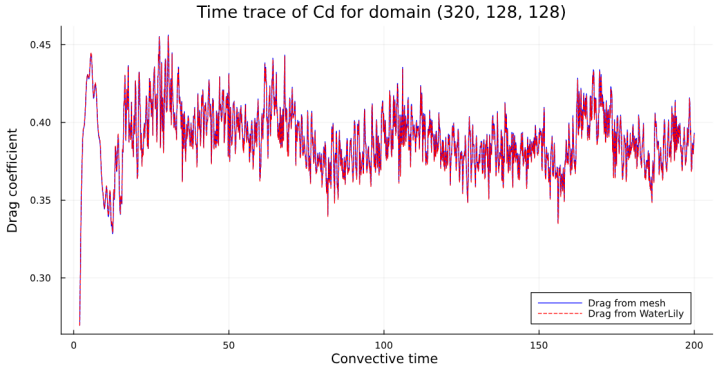


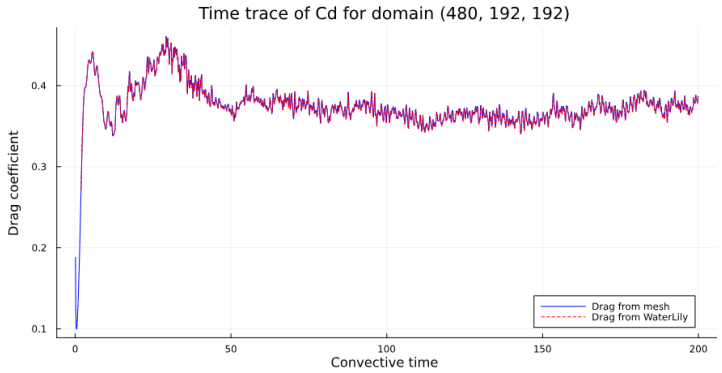
Figure 3.24: Boxplot of errors between current WaterLily method and mesh based method

To further assess accuracy, we examine the relative error with respect to the reference solution provided by the established WaterLily implementation. The results demonstrate that the median error remains extremely low for all three tested domain configurations. As depicted in figure 3.24, the boxplots of the percentage error throughout the simulation confirm this finding. While at some time steps the error is $> 1\%$, we can see that these are outliers and taking the time averaged quantity will filter these out. With a median error of approximately $\approx 0.24\%$ in all cases, we conclude with confidence that the new mesh-based approach performs exceptionally well and is in excellent agreement with the reference method.

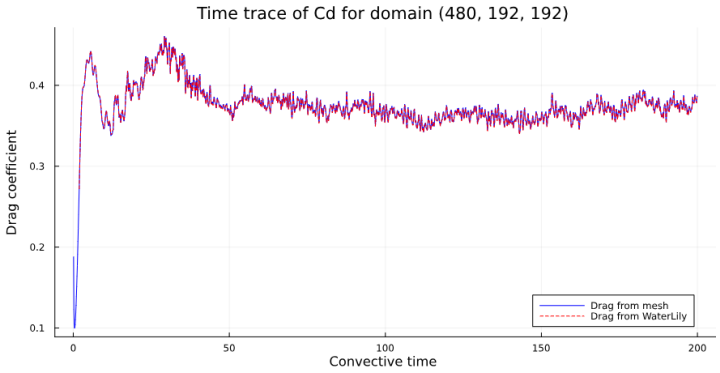
Since we now have an easy way to access parts of the domain from the mesh we can easily create pressure distributions which was previously more difficult. Figure 3.23 shows the pressure distribution, rendered in 3d, on the sphere from the smallest domain.



(a) Time trace for domain size 320 x 128 x 128 .



(b) Time trace for domain size 480 x 196 x 196 .



(c) Time trace for domain size 640 x 256 x 256 .

Figure 3.25: Drag coefficient over time for a sphere at increasing domain sizes: (a) 320x128x128, (b) 480x196x196 (c) 640x256x256 .

Finally we need to address the case of non bounding meshes. Due to the inflation of the sdf the edge of a plate deforms and gets rounded to a an arc with a radius of the half thickness. Figure 3.26 shows such an edge and the corresponding layers from the BDIM. The forces on the rounded edge do get included with the existing method, but when probing from a mesh only forces normal to the mesh elements are taken into account. The ratio of the area of the curve to the area of the mesh can be expressed as

$$\frac{S\pi\frac{thk}{2}}{A_{surface}}$$

where S is the length of the edge and $A_{surface}$ the full surface area of the actual plate. For a simple square plate with a length of 100 cells the ratio equals

$$\frac{4 \cdot 100 \cdot \pi \cdot (2+\sqrt{3})}{2 \cdot 100^2} \approx 0.23$$

which means the surface area of the 0 iso-contour of the sdf is much larger that the area of the mesh. When we iterate over all mesh elements we will neglect a significant part of the surface area around which the flow is calculated. For this reason this method is not fit for non- bounding structures. When simulating the kite which has a thin canopy structure we will be forced to use the existing method which will rely on the correct determination of the normal vector field, which we validated in Figures 3.11.

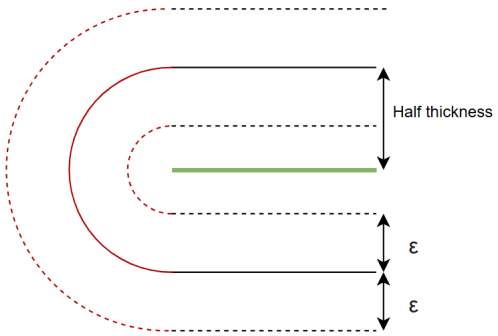


Figure 3.26: Layers on the edge of a submerged object. Green line represents the mesh, full line is the 0 iso-contour of the resulting sdf at the half thickness from the mesh, dotted lines are the edges of the transition zone. Red lines represent part of the submerged object we can not include with the mesh based method

3.7.3. Obtaining y+ values from mesh

Since we now have a mesh inside of the simulation we can calculate y+ values at the surface in order to validate the results. The y+ value will be calculated for each mesh element. We do want to note that together with the BDIM the y+ value is more abstract as the transition between the wall and fluid flow is not at a discrete location, rather it is blurred out. For the sake of obtaining a value we can say that the effective wall is located halfway the transition point which is the location of the mesh element in our case. However this still remains an estimation of y+ and results should not be taken as absolute.

For each triangle two velocity vectors will be taken similarly to how two pressure values were used in section 3.7. In order to find the shear velocity component we subtract the normal velocity component and use the magnitude of the remaining vector. Doing this for both velocity vectors allows us to calculate the shear stress using a simplified second-order finite difference estimate by assuming zero velocity at the boundary [38]. We cannot use the actual velocity at the mesh center as this will always be in the transition zone or, for non bounding meshes on the inside of the submerged object. For bounding meshes we use $\delta = 1$ and for non bounding meshes we add the half thickness making $\delta = 1 + \frac{2+\sqrt{3}}{2}$.

$$u_{sh,i} = |\overline{u_i} - (\overline{u_i} \cdot \overline{n})\overline{n}|$$
$$\tau = \nu \cdot \frac{2u_{sh1} - u_{sh2}}{2\delta}$$
$$y+ = \frac{\sqrt{|\tau|}}{\nu}$$

4

Static simulations

Since we can now efficiently use meshes within WaterLily, we will use this to simulate the flow around a kite. We have chosen to use the TU Delft V3 leading edge inflatable kite as a model. This decision was made due to the existing data that can be used as validation data for our simulations.

In this section we will start by laying out the numerical setup and explaining the details of the WaterLily solver. Results will then be compared to existing RANS data and the time trace of the forces on the kite will be investigated in order to find possible oscillating forces.

4.1. Numerical setup

We perform the simulations using the WaterLily solver and using the newly made method described in the previous chapter. This is an implicit LES solver [37], that uses the Boundary Data Immersion method [66] to submerge the object [68]. It also makes use of a geometric multi-grid method made easier by the Cartesian grid. No external wall models are used as we aim for $y+ = \mathcal{O}(1)$. Since y+ is only an estimate and thus not enough to validate the simulation it will be further validated by comparing to existing experimental and RANS data. We will simulate the kite at a Reynolds number of 1×10^5 for a range of angles of attack between 0° and 20°. Each simulation will run for 45 convective time steps which was chosen after longer initial simulations showed the simulations were converged. Forces and y+ data will be calculated every 0.25 time steps and the full pressure and velocity field will be stored every 15 time steps for visualization. *WaterLily* is an incompressible fluid solver however when calculating the Mach number of the flow using a wind velocity between 13 and 30 knots [43] and an air temperature of 284 Kelvin [58] gives a Mach number of:

$$M = \frac{U_{13-30}}{\sqrt{\gamma R T}} \approx [0.0121, 0.02780]$$

These are extremely low mach numbers and we can assume that the effect of the compression of air is negligble.

4.1.1. Boundary conditions

WaterLily only uses a Cartesian grid and while this has many benefits it also has some drawbacks. The main disadvantage is that it is not possible to locally refine the grid. For detailed geometries or flows with high Reynolds number this means that the small required cell sizes needs to be used over the entire domain. Using classical boundary conditions that need to be several body lengths away from the submerged object leads to several billion cells, which is not very feasible. In order to solve part of this issue we use Biot-Savart boundary conditions [69]. These boundary conditions allow for the body to be within half a body length of the edge of the domain. The boundary conditions work by reconstructing the velocity at the domain boundaries from the vorticity within the domain using the Biot–Savart integral.

These boundary velocities are iteratively coupled into the pressure solver making use of a fast multi-grid

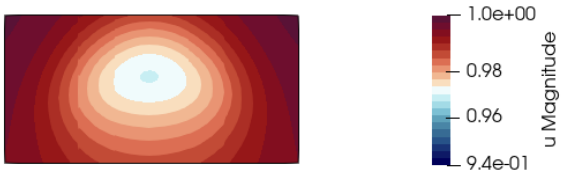


Figure 4.1: Velocity magnitude at the inflow due to the Biot-Savart boundary conditions

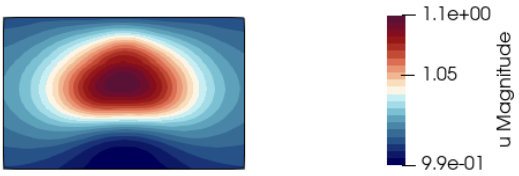


Figure 4.2: Velocity magnitude at the top boundary due to the Biot-Savart boundary conditions

method. Figure 4.1 and 4.2 show the velocity at the inflow and the top of the domain respectively for an initial simulation of the kite. Using these boundary conditions allows for a clearance of up to half a body length on all sides, but preferably more downstream [69].

4.1.2. Kite geometry and domain size

For the kite geometry we will use the TU Delft V3 LEI kite [1] which is freely available, a detailed layout of the kite can be seen in Figure 4.3. The available kite model has two separate surfaces that are close together to define the canopy which will not work in our solver as they are too close together, so we have removed the back of the bottom surface and consider the top surface as a non bounding surface that we will slightly inflate. To determine the domain size we used the aspect ratios of the kite. The maximum domain size for the NVIDIA RTX A6000 48 GB GPU was determined and based on this, the found aspect ratio's and required clearance the following domain size and maximum chord length were determined. The final domain size and clearances can be seen in table 5.1.

A mesh of the kite consisting of 158,282 triangular faces was placed inside of this domain. The kite was scaled to have a chord length of 205 cells. The sdf is calculated using the new BVH measuring method. Measuring is done on the CPU and the weights for the BDIM, μ_0 & μ_1, are copied to GPU. The back canopy is inflated to a thickness of $2 + \sqrt{3} \approx 3.73$. This does mean that we make a modeling error as the actual thickness of the kite (when scaled to a chord length of 205) would be 0.885. Accurately capturing the very thin canopy would require scaling up the kite by a factor of 4.22 which leads to a chord length of ≈ 863 cells. Scaling the domain in all directions would result in a computational domain of $\approx 25,163,017,642$ cells which, using an estimate of $\approx 6,990,506 \frac{cells}{GB}$ would require $\approx 3,600GB$ of memory. These estimates show that capturing very thin surfaces is difficult using the current method and thus we are making a necessary modeling error.

Figure 4.4 shows the signed distance field at the centerline and Figure 4.5 shows the field at $x = 270$ in domain coordinates which corresponds to halfway the chord length. Grey areas represent locations at which we did not have to measure due to the BVH based measuring. The amount of grey in the pictures is a great example of how efficient the method is. We can clearly see we are only measuring very close to the immerse object. Figure 4.4 shows how the canopy of the kite is slightly inflated to give it more thickness. Figure 4.6 shows the iso-contour at a distance of 0, the clearest representation of where the kite is actually located with in the domain. The signed distance function is then used to calculate the zero-th kernel moment μ_0, which is actually used to determine the boundary conditions. An important part about the kite geometry is how the struts are attached to the kite. Lebesque [39] showed how they need to be attached smoothly, however the inflation of the signed distance function and the calculation of the kernel moment already have a smoothing effect requiring no additional smoothing. Figure 4.7 shows a zoomed in detail of where a strut is attached with the iso-contour at $|\mu_0| = 1.5$, also visible on

height/chord	≈ 1.2
width /chord	≈ 3.2
Domain size	640 x 1024 x 512
Number of cells	335,544,320
Chord	205 cells
Clearance in z	131 cells or 0.64 chords
Clearance in y	182 cells or 0.89 chords
Clearance in x (front)	170 cells or 0.83 chords
Clearance in x (wake)	265 cells or 1.3 chords

Table 4.1: Domain parameters

this figure is an ”edge” behind the leading edge. This is the effect of cutting away the bottom canopy to achieve a non-bounding mesh. This transition may seem abrupt, but it is important to remember the working of the BDIM [67]. The submerged object is not at a discrete location, rather the boundary conditions are smoothed out over a region in space. This edge will be smoothed out and is not a discrete ”jump” in the geometry as the figure might suggest. We will see in the results section of this chapter that the jump does not affect the solution.

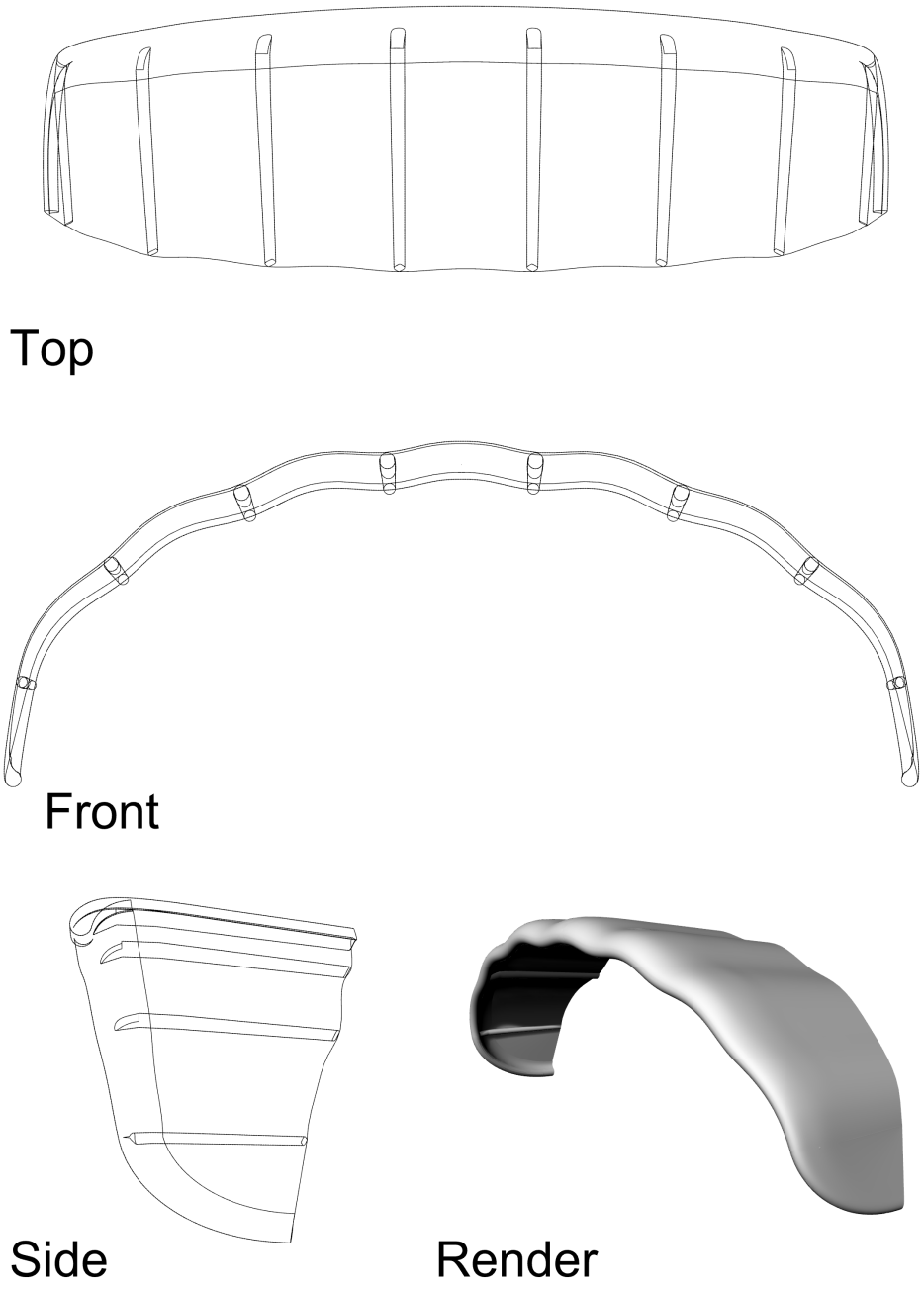


Figure 4.3: Detailed geometry of the TU Delft V3 LEI kite [1]

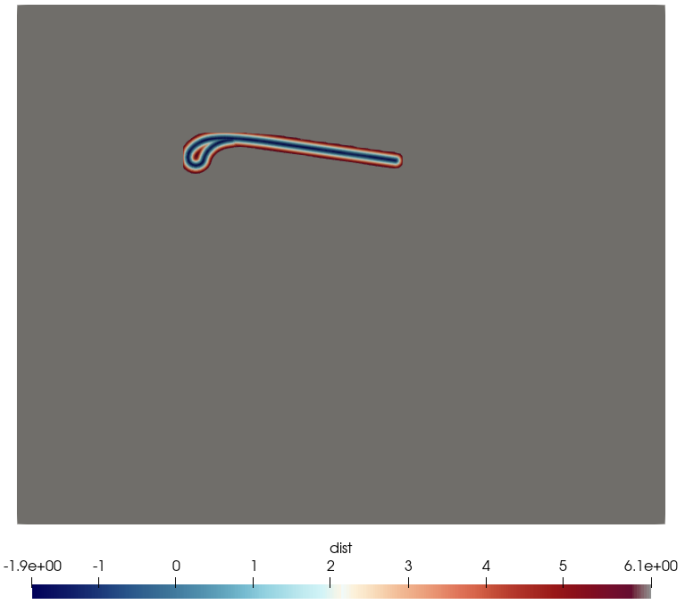


Figure 4.4: Signed distance field of the kite at the symmetry plane; Grey areas represent locations that were not measured

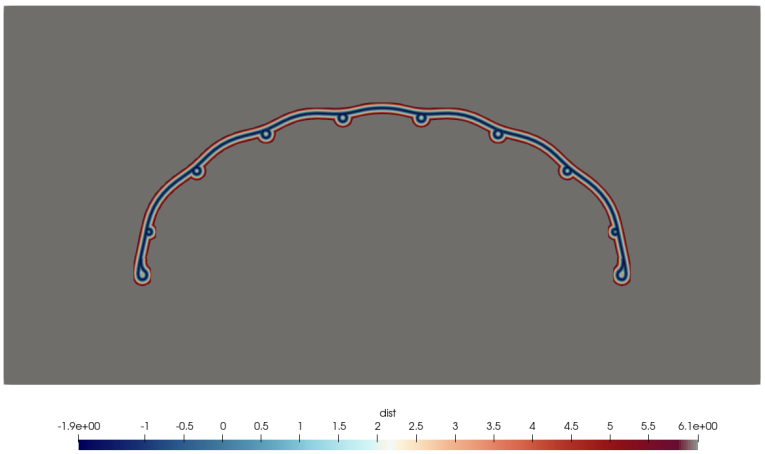


Figure 4.5: Signed distance field of the wing at x/c=0.5; Grey areas represent locations that were not measured

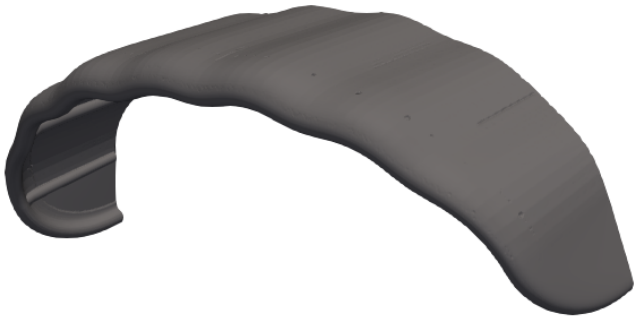


Figure 4.6: 0 iso-contour representing the location of the kite in the domain, the high resolution makes it almost identical to surface the mesh

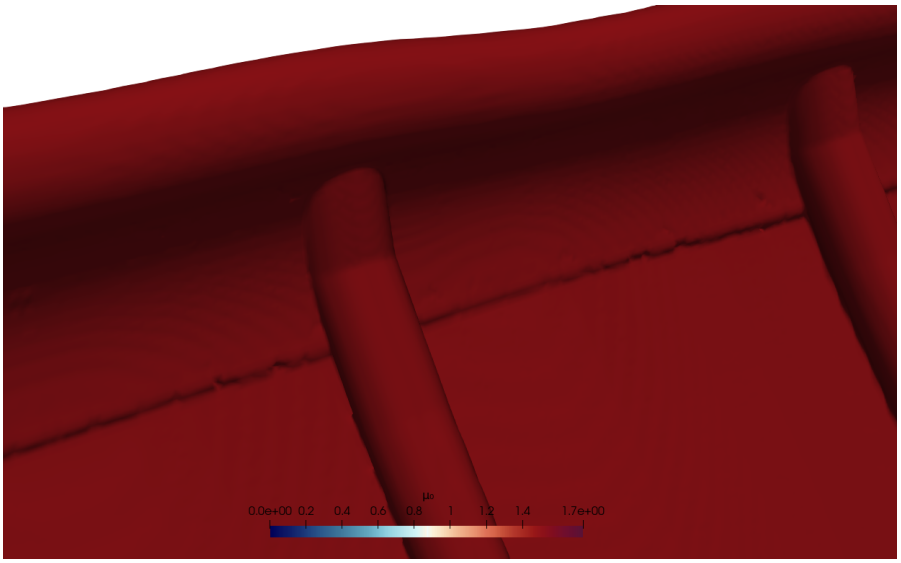


Figure 4.7: Detail of where strut is attached to leading edge and canopy; iso-contour of $|\mu_0| = 1.5$

4.2. Results

In this section the results of the static simulations will be discussed. We will validate the results by comparing mean forces to validation data and inspecting the estimated $y+$ values. An inspection of the flow fields will follow after which we will investigate the fluctuating forces and the effect of turbulence. We will also analyze the pressure along the chord at the centerline to show how the pressure fluctuates locally

4.2.1. Mean drag and lift coefficients

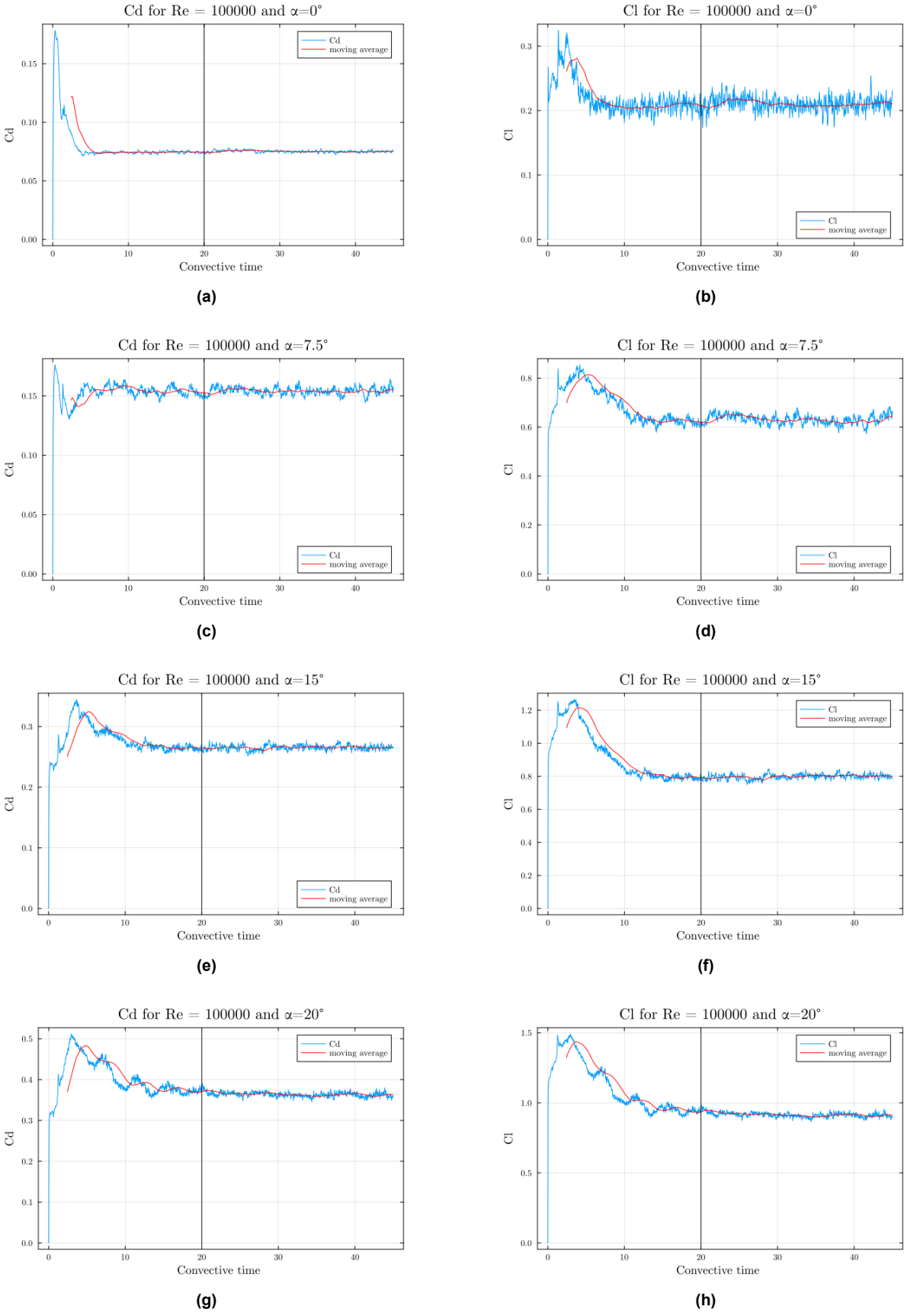
The drag an lift coefficients are calculated from the measured forces on the kite. Since both the density, ρ and the free stream velocity,U_∞ are set to one this simplifies to only dividing by half the projected area,A.

$$C_D = \frac{F_x}{0.5A}$$
$$C_L = \frac{F_z}{0.5A}$$

The mean coefficient was determined by averaging the force starting from where the solution was converged. For these simulations we started using data starting from $t = 20$, shown by a vertical line in the time traces in Figure 4.8. In all plots in Figure 4.8 we can see the recorded values in blue. These were sampled every 0.025 time steps or 40 times per time step. This is a fairly high sampling frequency,f_s, which was chosen to find high frequency force oscillations which will be discussed in subsection 4.2.4. the drawback of this high sample frequency is that it's more difficult to assess when the solution has converged. In order to filter out some of the noise we also plot a moving average with a window size of 100 samples or 2.5 time steps, shown in red. From this we can clearly see that the mean values do not change significantly after 20 time steps.

Comparing to experimental and wind tunnel data we can see that our calculations agree on the course of the drag an lift coefficients. Figure 4.9 shows the calculated drag coefficient and the reference data[1]. It appears that we overestimate C_D for almost all angle's of attack, but the difference is mainly visible for $\alpha = 7.5°$ where we obtain almost double the drag compared to RANS or experimental data. This becomes a larger discrepancy in Figure 4.11, showing the $\frac{L}{D}$ ratio. The increase in drag completely changes the profile of the curve. While for the reference data a clear optimal working point can be seen between $\approx 5°$ and $\approx 10°$, this optimum is not as defined anymore.

Looking at C_L in Figure 4.10 we see that our results fully agree with the reference data. We are closer to the wind tunnel results at $Re = 5e5$ than to the RANS results at $Re = 1 \times 10^5$. This could be explained due to an overly attached flow in the RANS simulation [32] that is unable to account for enough turbulence production [22] and changes the pressure distribution on the leading edge. We will investigate this discrepancy when analyzing the flow fields.



(a) **(b)** **(c)** **(d)** **(e)** **(f)** **(g)** **(h)**

Figure 4.8: Time traces of drag (C_d) and lift (C_l) coefficients at $Re = 100000$ for different angles of attack; **(a,c,e** & **g)** : C_D ; **(b,d,f** & **h)** : C_L

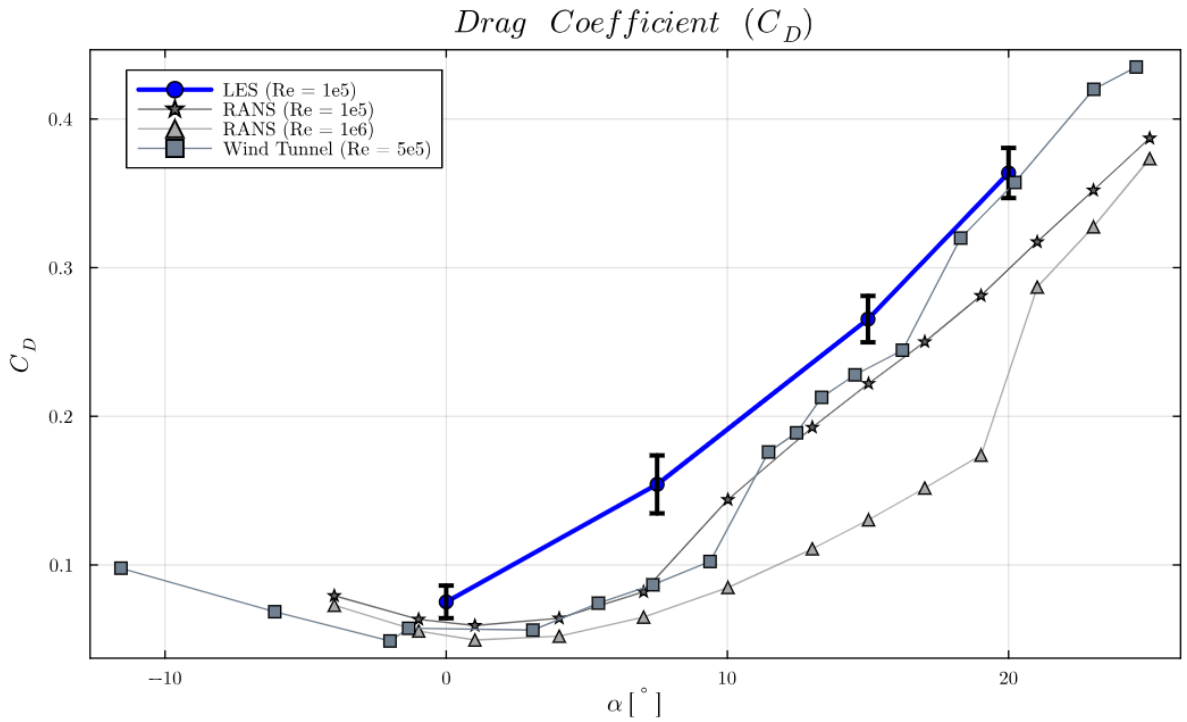


Figure 4.9: Mean drag force, C_D, over angle of attack; Validation data from [1]

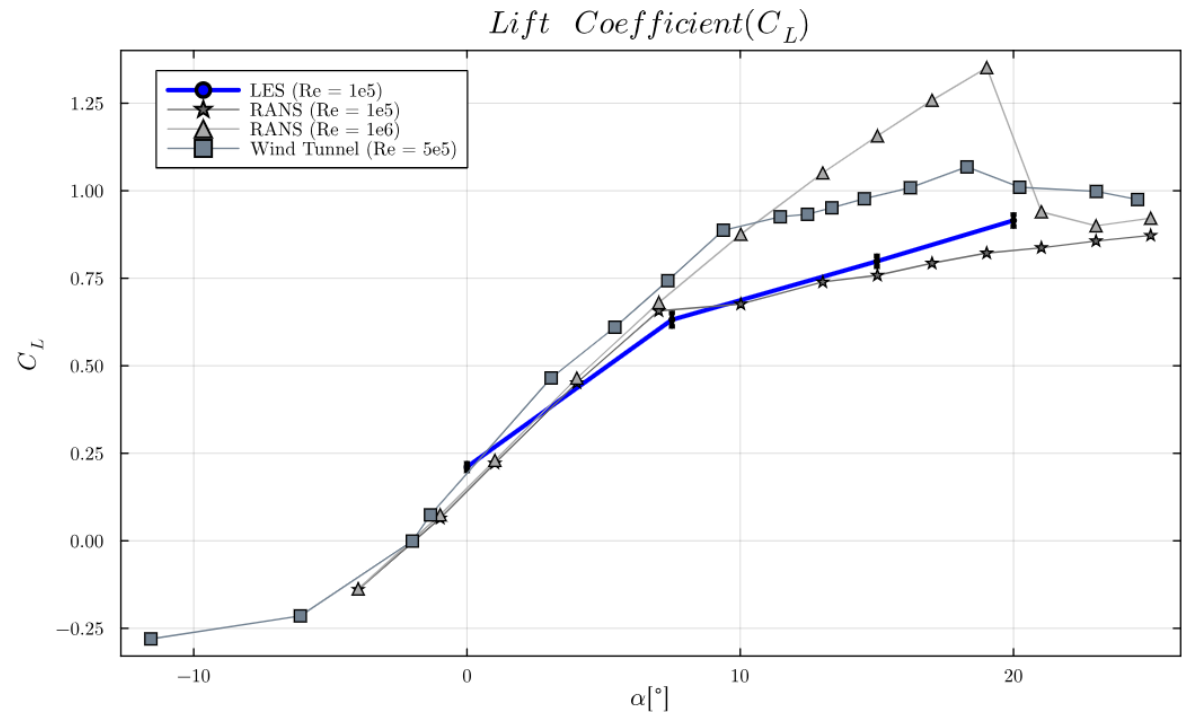


Figure 4.10: Mean lift force, C_L, over angle of attack; Validation data from [1]

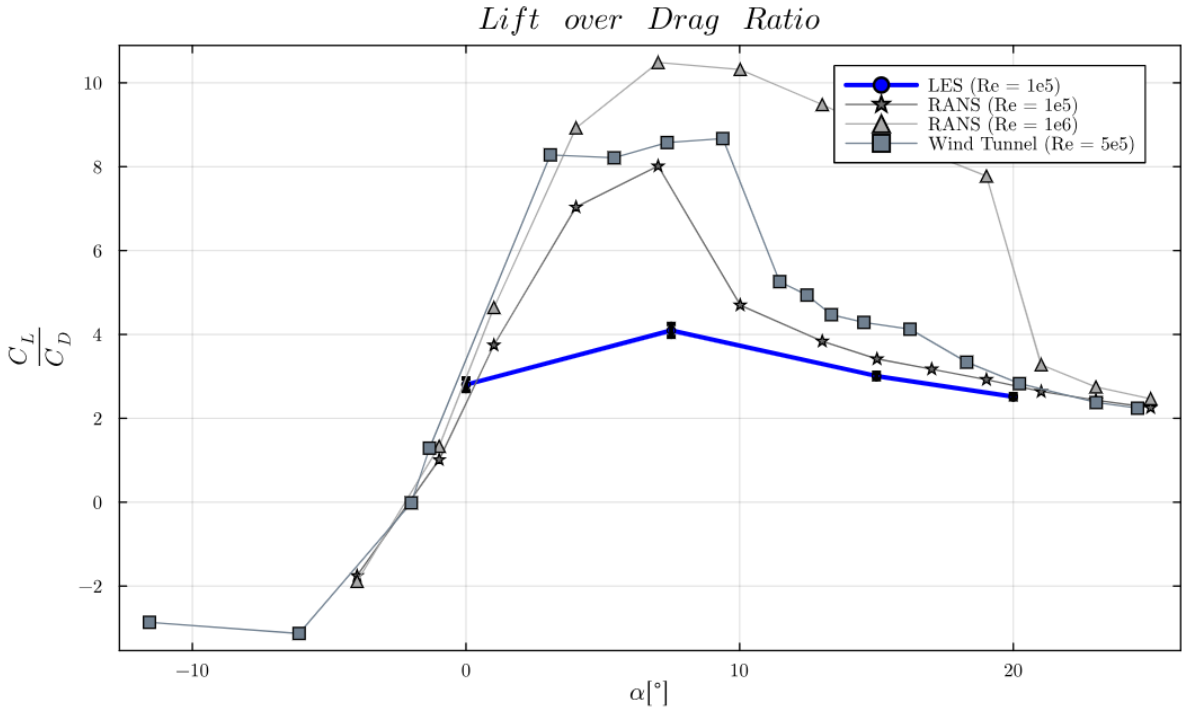


Figure 4.11: Mean lift over drag force over angle of attack; Validation data from [1]

4.2.2. Estimated wall resolution

In order to validate the results we will start by looking at the y+ values. The y+ data from the final 25 convective time steps was averaged and the data is presented here. For full time traces of the data see appendix[1]. In Table 4.2 we can see that for all simulations the median y+ is $\mathcal{O}(1)$. The standard deviation and inter quartile range both show that most of the data points are within acceptable limits. The minimum measured value rounds down to 0.0 for all simulation showing that on some parts of the kite the cells are a lot smaller than they need to be. On the other side we see that the maximum measured value is around or more than 10. This indicates that the first cell is no longer close enough to be within the viscous sublayer and we would require explicit wall models. However since the calculated y+ value can only be an estimate for the BDIM method due to the lack of a well defined wall and as the results agree with existing research we accept the outcome of the simulations.

Table 4.2: Estimated $y+$ statistics

Simulation	Median	Std	IQR	Min	Max
$Re = 1 \times 10^5$, $\alpha = 0°$	1.276	2.405	3.341	0.0	11.773
$Re = 1 \times 10^5$, $\alpha = 7.5°$	1.195	2.557	3.748	0.0	13.128
$Re = 1 \times 10^5$, $\alpha = 15°$	1.201	2.71	4.234	0.0	10.514
$Re = 1 \times 10^5$, $\alpha = 20°$	1.569	2.681	4.298	0.0	10.79

4.2.3. Flow Field Analysis

We will take a detailed look at snapshots of resulting flow fields. For simplicity we analyze the fields at a slice on plane of the centerline of the kite. We provide a 3D rendering of the Q-criterion shown by iso-contours at $Q_{crit} = 0.01$. As the location of the kite is not directly visible in the results we have added the approximate outline of the kite in blacj for clarity. For the 3D representations we have added the original mesh to the images. We can confirm the working of the Biot-Savart boundary conditions by inspecting the pressure fields. The pressure contours follow natural shapes and do not suddenly change direction when approaching the edge of the domain. Unfortunately some of the data got corrupted

[1] In Progress

during a data transfer, for $\alpha = 0°$ and $\alpha = 15°$ the final snapshot at $t = 45$ are presented. For $\alpha = 7.5°$ a snapshot at $t = 30$ is used, which is still part of the converged solution. For $\alpha = 20°$ more data is missing and we are forced to present a snapshots at $t = 15$, which represents an unconverged solution[2].

At low angle of attack $\alpha = 0°$ we see a very attached flow field. The velocity magnitude in Figure 4.13 shows the flow follows the top of the kite. The pressure field in Figure 4.17 shows little difference between the pressure and suction side of the foil and no well defined downwind low pressure area's. This leads to both a low drag and lift as we have observed in Figures 4.9 & 4.10. The Q-criterion in Figure 4.22 shows that most of the turbulence is generated on the pressure side of the kite as the flow detaches at the bottom of the inflated leading edge which is visible in the velocity field, but more apparent in the vorticity field on Figure 4.21. Overall we see a flow with very little turbulence.

Increasing the angle of attack to $7.5°$ shows a clear increase in turbulence and wake structure. The pressure field in Figure 4.18 shows a larger pressure difference between the pressure and suction side as well as a larger low pressure area on the suction side. Small vortices are also being shed on both sides of the kite. We already see a more turbulent flow that only stays attached for a short distance on the suction side, but still detaches underneath the leading edge which is seen in the vorticity field in Figure 4.23. The detached flow on the suction side does not reattach leading to high turbulence on both sides of the kite seen in the iso-contours of the Q-criterion in Figure 4.24. This is different from the RANS results by Lebesque in 2020[39] which are the basis for the validation data in Figures 4.9& 4.10. They show RANS results for $\alpha = 6°$ that show a recirculation zone that reattaches to the kite shown here in Figure 4.12. Our flow field shows no signs of reattachment at the approximately same angle of attack. The faster, reattached flow would result in a lower pressure at the pressure side, decreasing the drag on the kite. We would like to propose this as the reason for the discrepancy in drag seen in Figure 4.9.

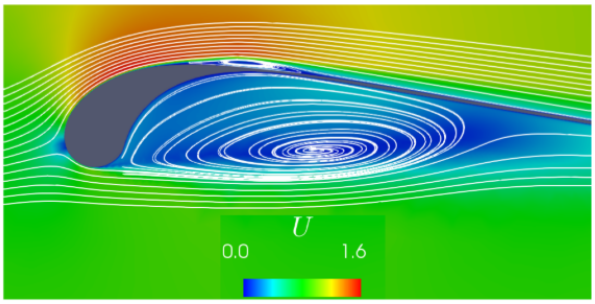


Figure 4.12: Streamlines and magnitude of the velocity field in the wing symmetry plane for Re= 1e5 and $\alpha = 6°$ taken from [39]; Streamlines show a recirculation zone, but reattach on the canopy

At an angle of $15°$ the velocity field in Figure 4.15 shows the flow has significantly slowed down behind the kite. The corresponding pressure wake in Figure 4.19 is even more pronounced now. We also start to see more pronounced vortex shedding at the suction side coming from the leading edge, as the flow is now almost completely detached leading to a highly turbulent flow field in the wake which can be seen from the vorticity magnitude in Figure 4.25. On the pressure side of the vorticity field we see that the flow detaches from the leading edge, but the produced turbulence hits the canopy farther down and appears to re-attach which is most visible on the velocity profile in Figure 4.15. The Q-criterion iso-contours in Figure 4.26 show a highly turbulent suction side, but on the pressure side we see some turbulence is already dissipated by the time it reaches the trailing edge. For this higher angle of attack we see that the flow does reattach to the canopy. We confirm this by the dissipation of the turbulence in Figure 4.26 and 4.25.

For the high angle of attach of $20°$ we would like to note again that these flow fields are from the unconverged solution at $t = 15$. We will analyze the flow field ignoring this fact for now. The velocity profile in Figure 4.16 shows an even more turbulent suction side. On the pressure side we see again

[2]Note to committee: At the time of writing not enough time remained for the simulations to be performed again, but for the final deadline all data should be available and will be presented in full

that the flow detaches on the leading edge, but is now re-attached even sooner leading to very little turbulence on the pressure side as seen in Figure 4.28. The pressure field in Figure 4.20 shows clear vortex shedding from the leading edge at the suction side and a large pressure difference between both sides. The vorticity magnitude in Figure 4.27 shows the highly turbulent wake at the suction side as well as the vortex shedding from the leading edge. The pressure side shows the turbulence coming from the leading edge and being dissipated by the canopy when the flow reattaches.

Velocity Magnitude and Pressure fields

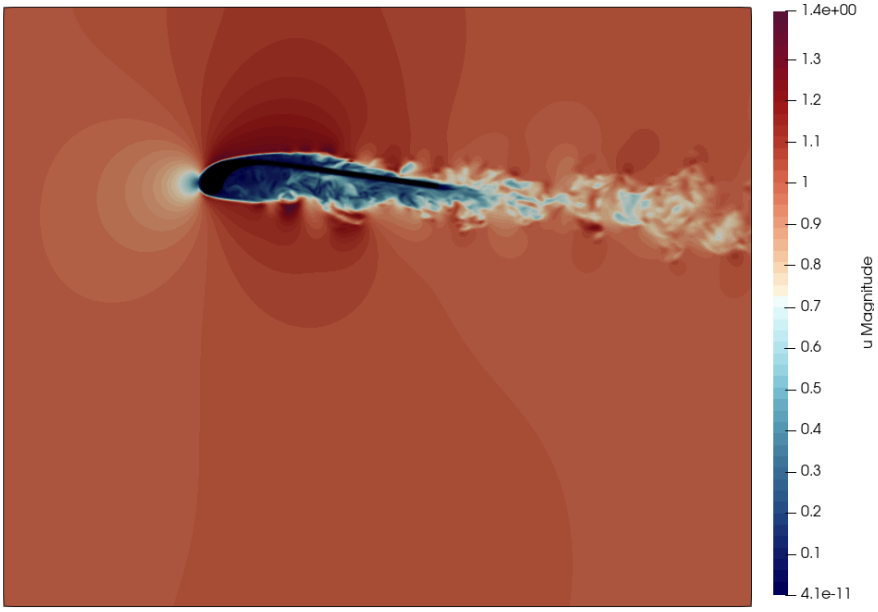


Figure 4.13: Velocity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 0°$

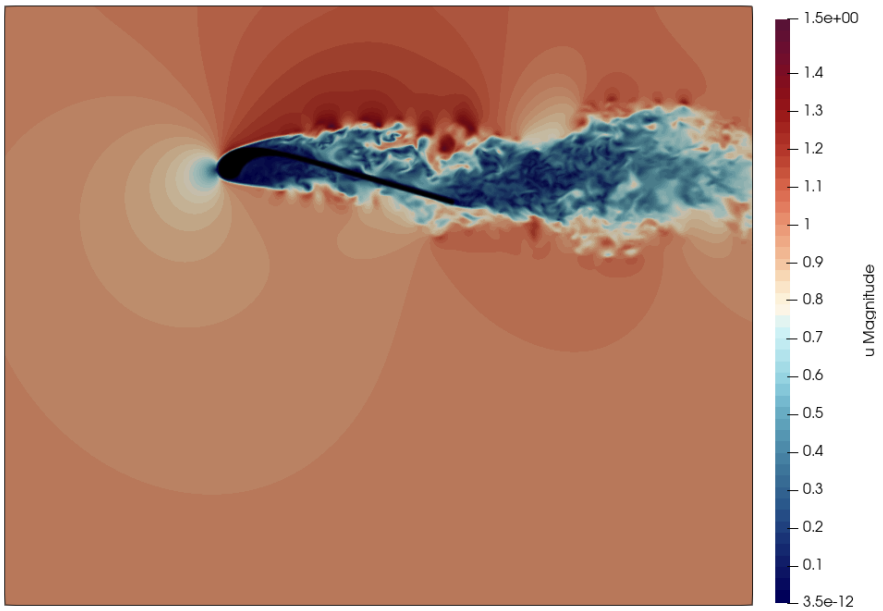


Figure 4.14: Velocity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 7.5°$

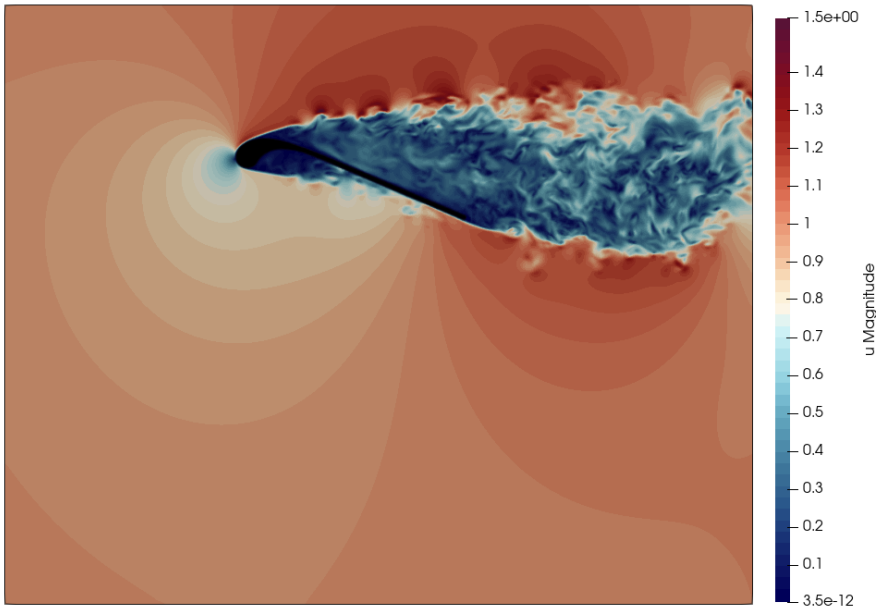


Figure 4.15: Velocity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 15°$

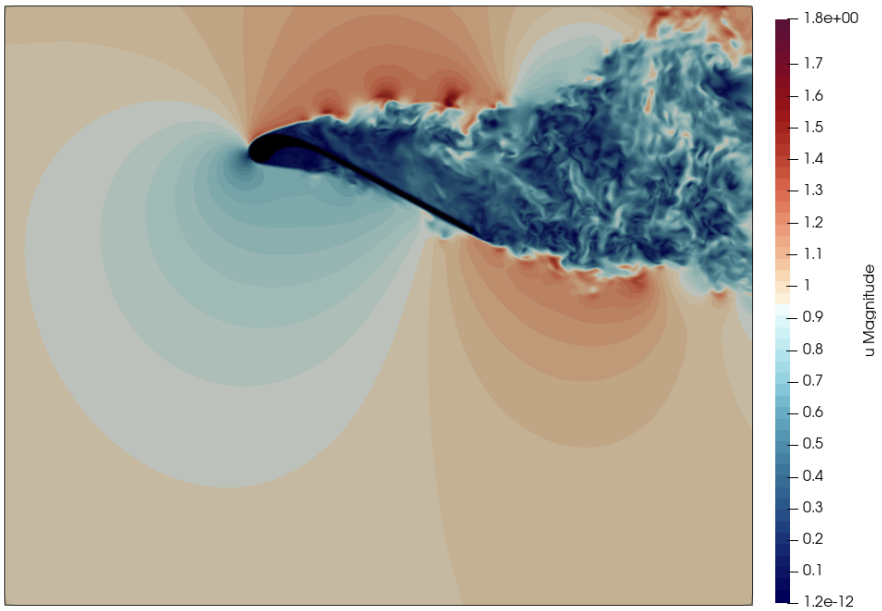


Figure 4.16: Velocity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 20°$

Pressure Fields

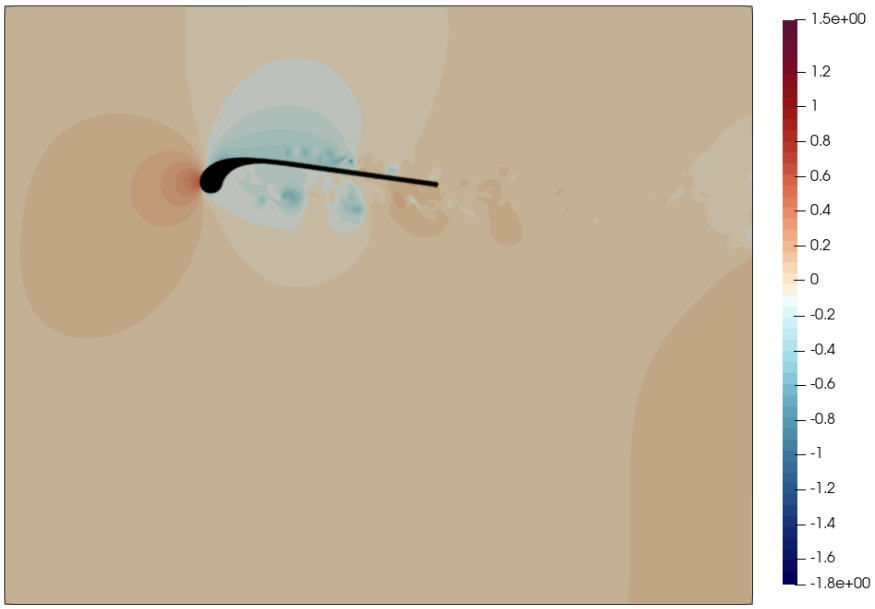


Figure 4.17: Pressure field in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 0°$

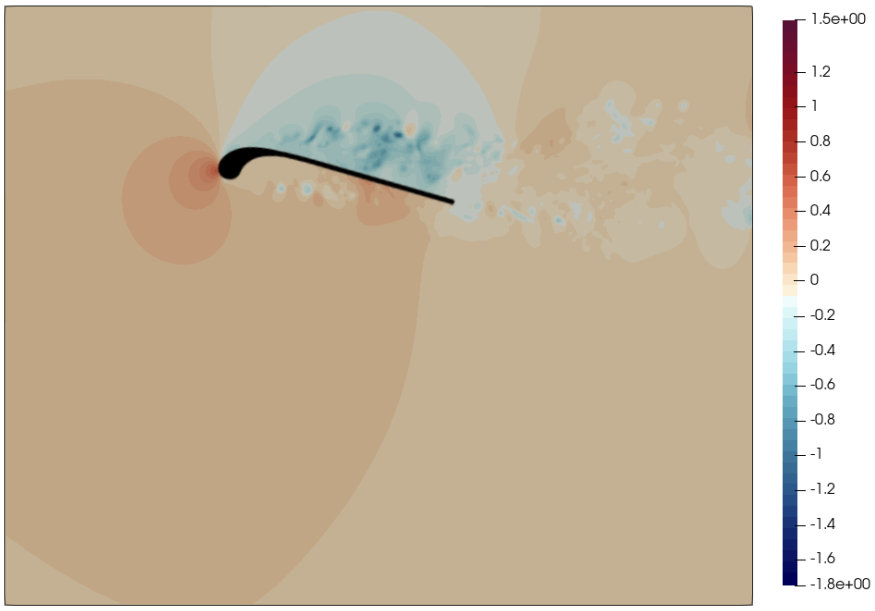


Figure 4.18: Pressure field in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 7.5°$

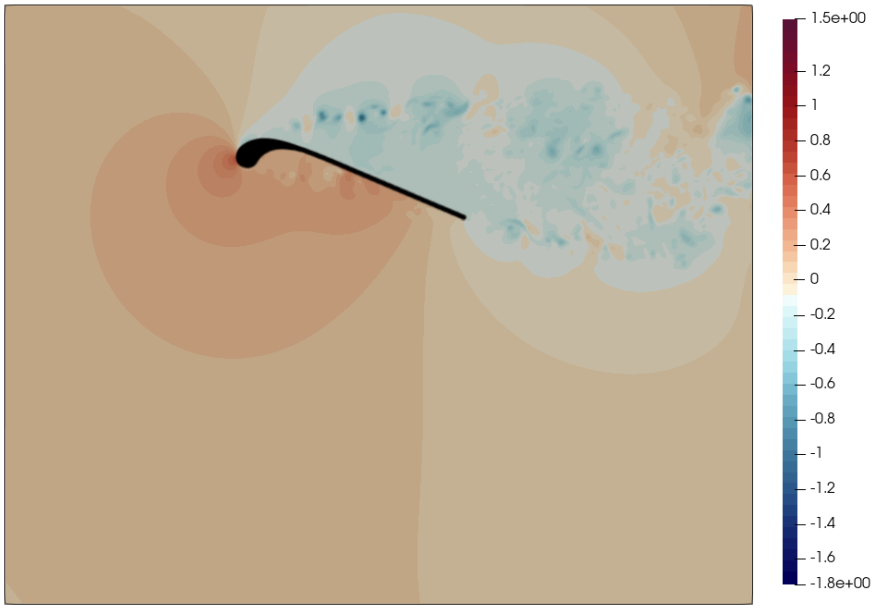


Figure 4.19: Pressure field in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 15°$

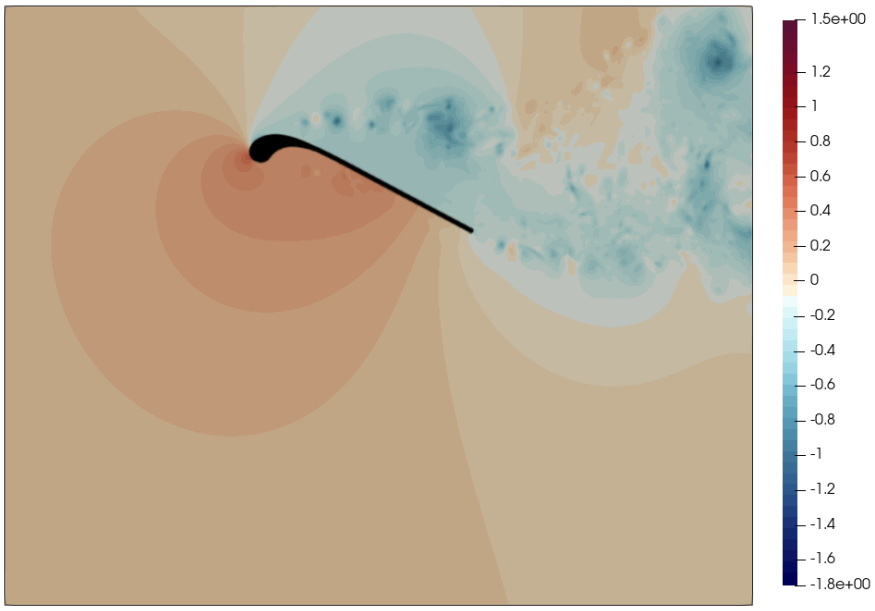


Figure 4.20: Pressure field in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 20°$

Vorticity and Q-Criterion Fields

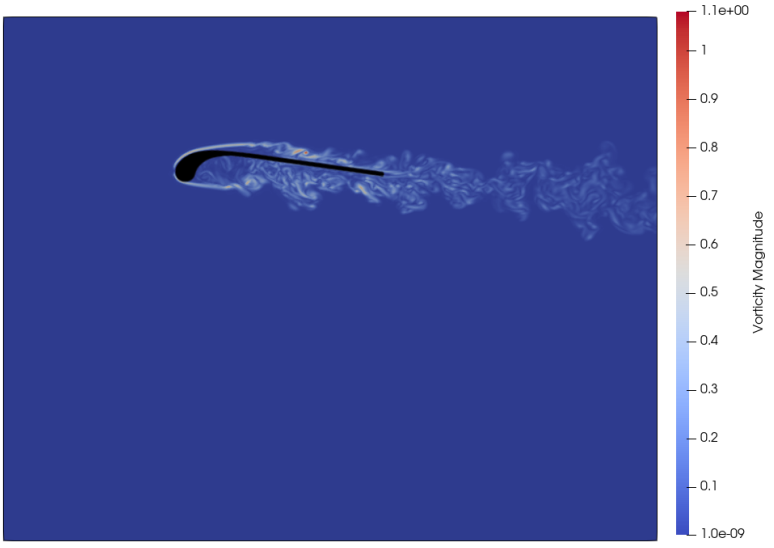
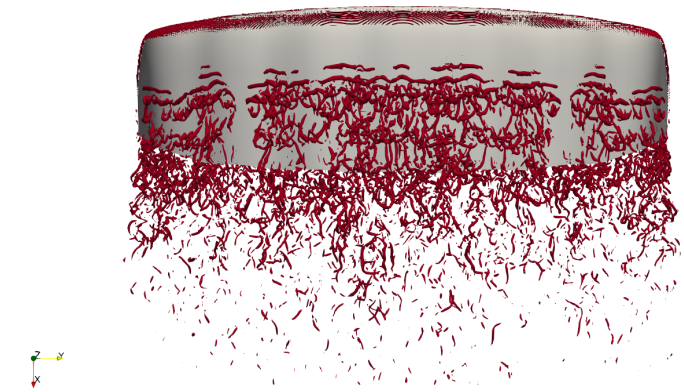
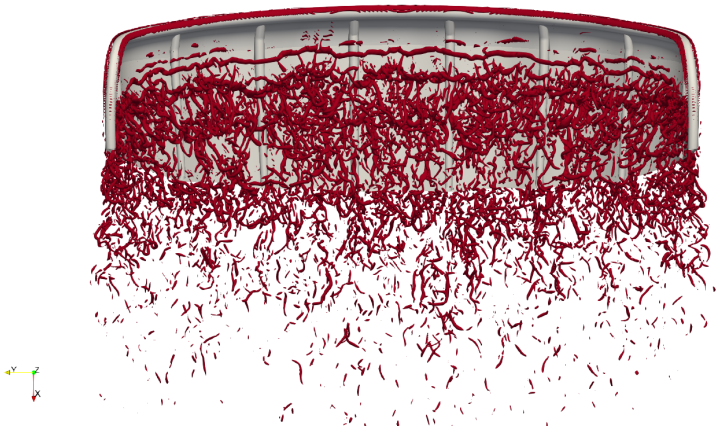


Figure 4.21: Vorticity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 0°$



(a) Suction side



(b) Pressure side

Figure 4.22: 3D representation of the Q-criterion iso-contours of $Q_{crit} = 0.01$ for $Re = 1 \times 10^5$ and $\alpha = 0°$

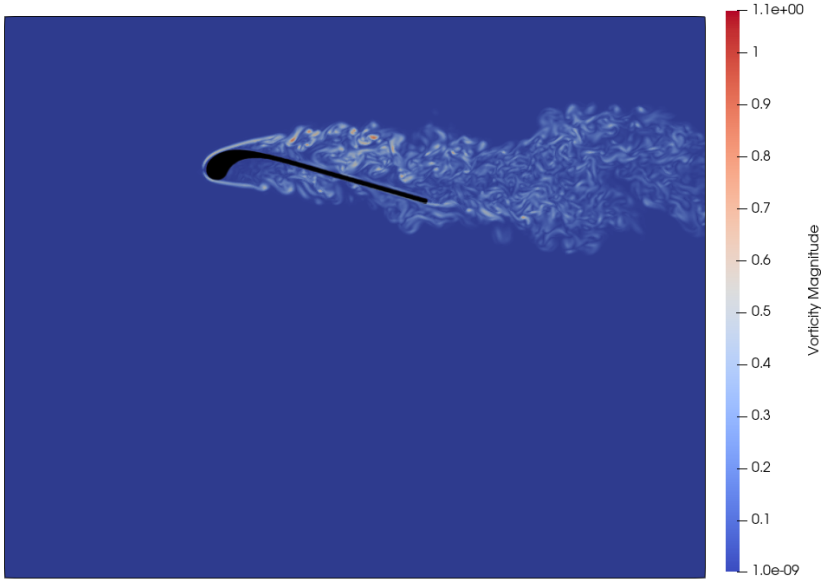
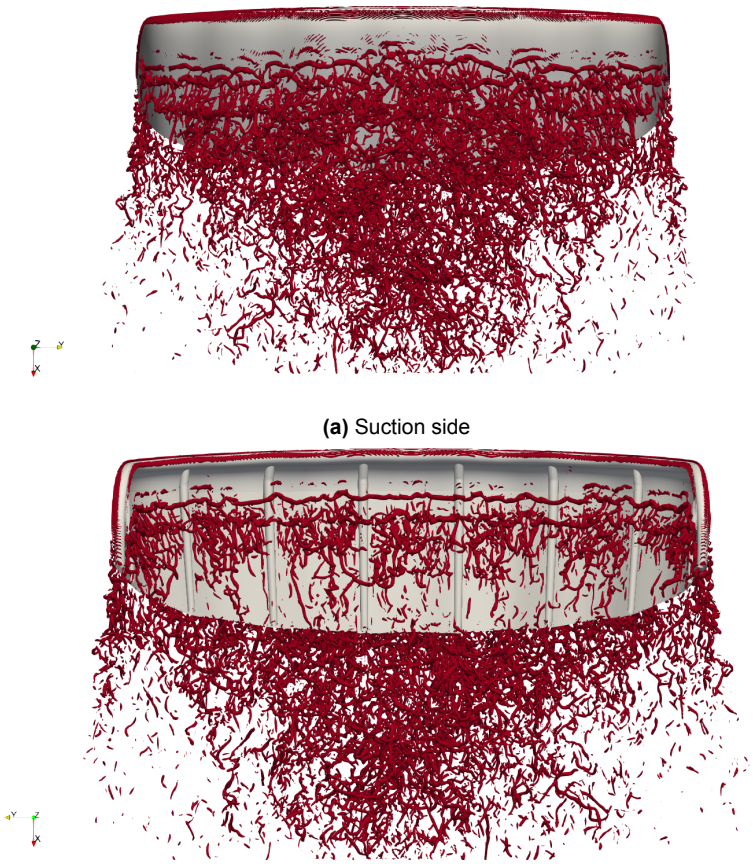


Figure 4.23: Vorticity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 7.5°$



(a) Suction side

(b) Pressure side

Figure 4.24: 3D representation of the Q-criterion iso-contours of $Q_{crit} = 0.01$ for $Re = 1 \times 10^5$ and $\alpha = 7.5°$

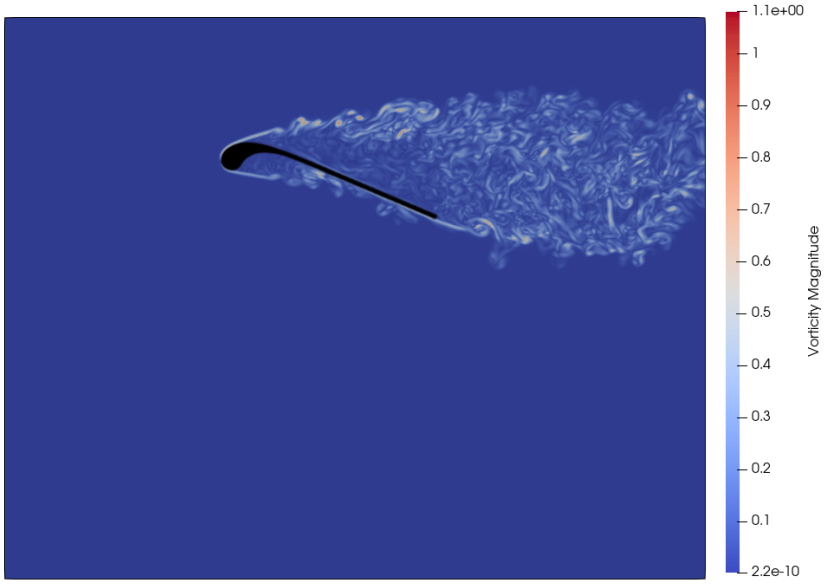
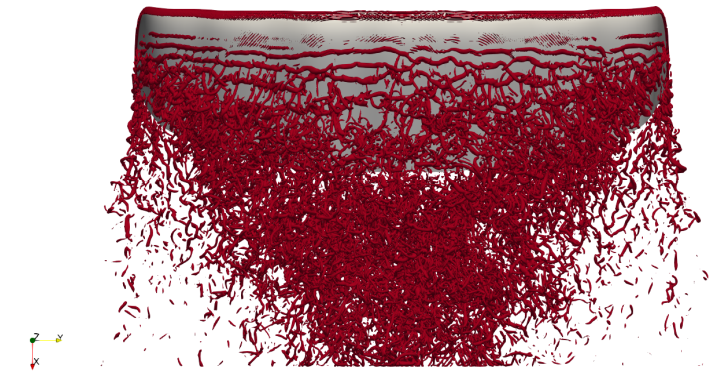
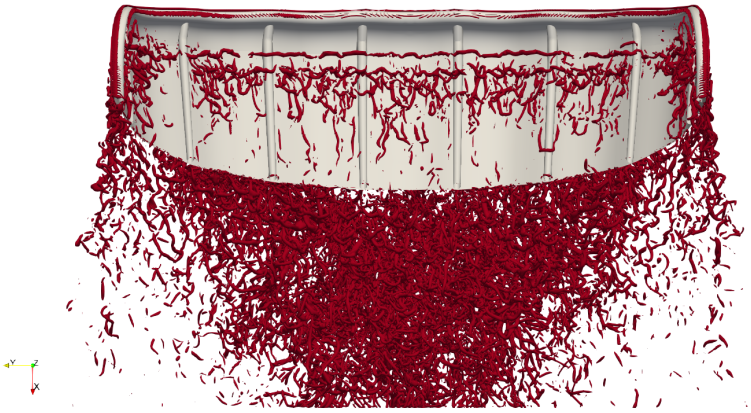


Figure 4.25: Vorticity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 15°$



(a) Suction side



(b) Pressure side

Figure 4.26: 3D representation of the Q-criterion iso-contours of $Q_{crit} = 0.01$ for $Re = 1 \times 10^5$ and $\alpha = 15°$

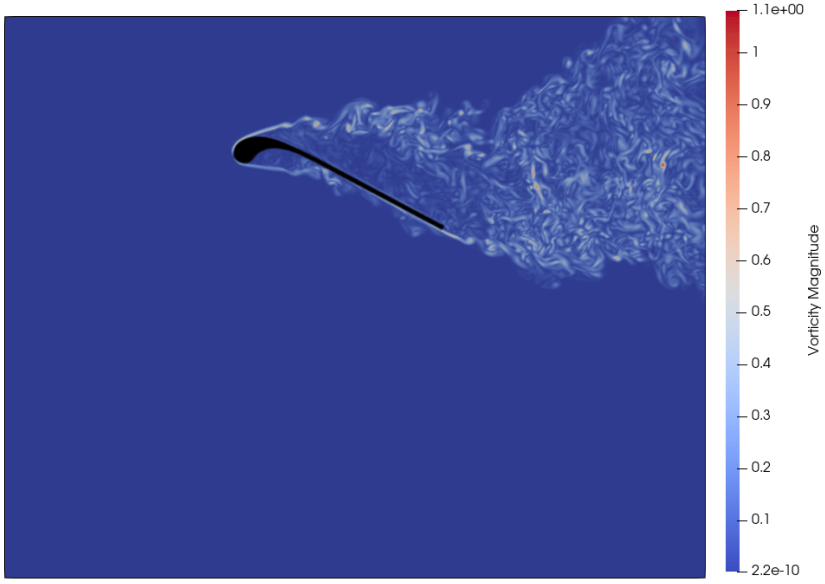
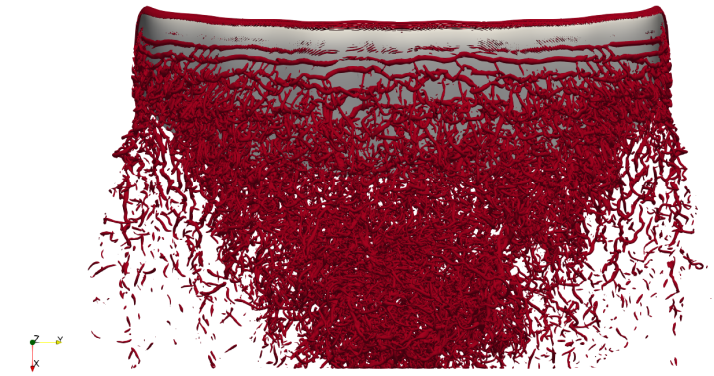
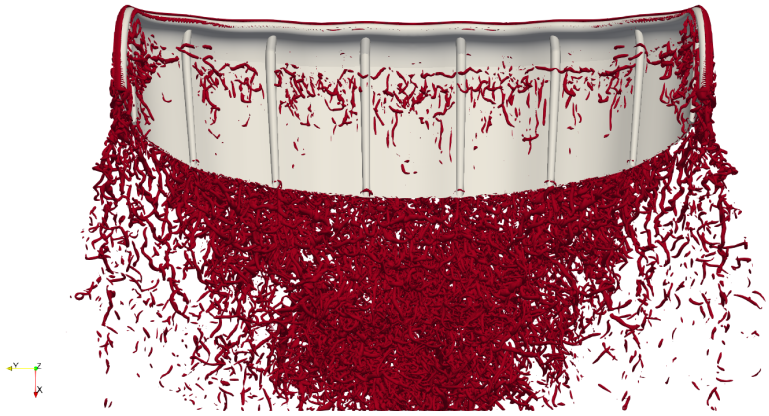


Figure 4.27: Vorticity magnitude in the wing symmetry plane for $Re = 1 \times 10^5$ and $\alpha = 20°$



(a) Suction side



(b) Pressure side

Figure 4.28: 3D representation of the Q-criterion iso-contours of $Q_{crit} = 0.01$ for $Re = 1 \times 10^5$ and $\alpha = 20°$

Analyzing the flow fields has shown a lot about the kite aerodynamics that RANS or other similar methods that don't actually simulate turbulence can't reproduce. We have clearly seen the effects of flow separation and reattachment, especially on the pressure side where the flow detaches from the leading edge and reattaches on the canopy. The effect of the flow reattachment has been identified as the reason for the discrepancy between other methods and the presented solution. Well defined vortex shedding is observed, mainly coming from the separation point on the suction side of the kite. We have seen that increasing the angle of attack, α significantly changes where the flow is turbulent. For low angles of attack the pressure side is turbulent due to the flow separation at the inflated leading edge. Increasing the angle of attack leads to flow separation on the suction side and increasing the angle of attack enough leads to reattachment at the pressure side, making only the suction side turbulent.

4.2.4. Fluctuating forces

In order to determine the degree to which the observed turbulence affects the kite's drag and lift coefficients, we analyze the fluctuations in their time traces. All data used in this section is sampled only after 20 time steps to ensure that only the statistically converged portion of the solution is included.

We begin by estimating the power spectral density (PSD) of the drag and lift coefficient time traces using Welch's method. Given our sampling frequency, $f_s = 40$ samples per time step, we can resolve frequencies up to the Nyquist frequency of $20\,\text{Hz}$. This approach improves upon the standard Fourier spectrum by dividing the time signal into overlapping segments, applying a Hann window to each segment, computing the squared magnitude of the discrete Fourier transform, and averaging the results. The PSD estimate can be written as (Welch 1967 [64]):

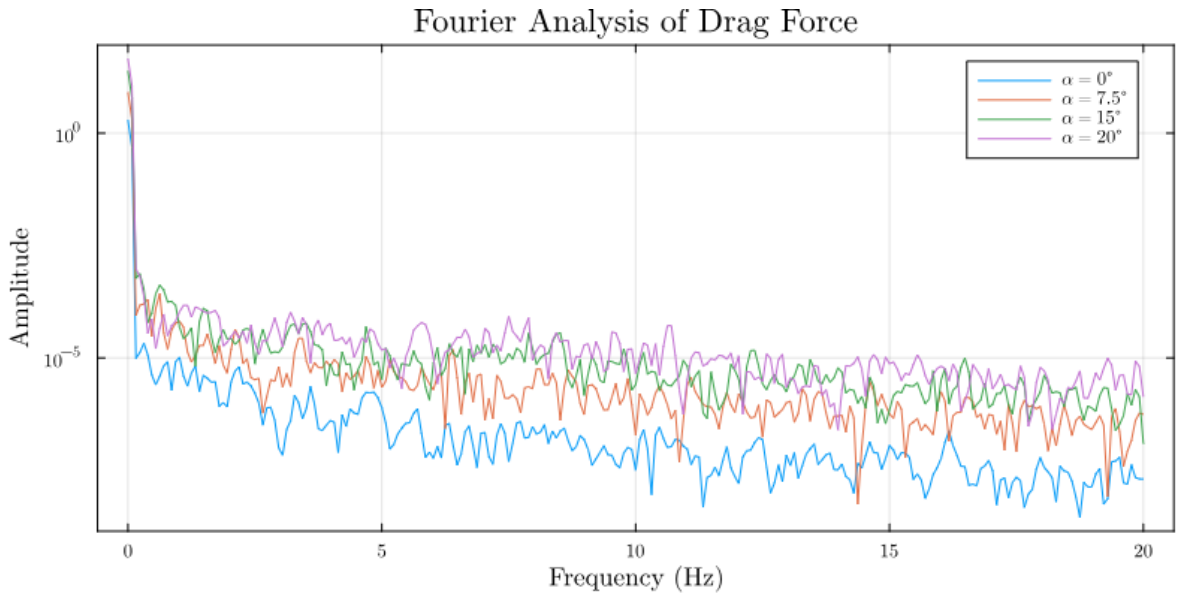


Figure 4.29: Fourier analysis of the time trace of the drag coefficient for different angles of attack

$$\hat{P}_{xx}^{(\text{Welch})}[m] = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{U}\left|\sum_{n=0}^{N-1} x_k[n]\, w[n]\, e^{-i2\pi mn/N}\right|^2$$

where:

- $x_k[n]$ is the k-th overlapping segment of the signal,
- $N = 512$ is the window (segment) length,
- $w[n]$ is a Hann window of length N,
- A 10% overlap is used between segments, corresponding to 51 overlapping samples,
- $U = \sum_{n=0}^{N-1} |w[n]|^2$ is the window normalization factor,
- The FFT is computed using the real-input discrete Fourier transform (rFFT),
- K is the number of segments over which the spectrum is averaged,
- m indexes the frequency bins in the one-sided spectrum.

The resulting spectra for both drag and lift show no dominant frequencies across any of the tested angles of attack. All signals exhibit a broadband shape, with a steep decay in amplitude at the lower frequencies followed by a relatively flat, noisy distribution. We do not see any narrow peaks that would indicate the presence of periodic vortex shedding or coherent structures. This means that the small vortices emitted from the leading edges do not have a significant effect on the drag and lift characteristics. The energy is spread across the spectrum, which is typical for turbulent flows with no dominant modes. This confirms that the force fluctuations are stochastic and that the flow is in a fully developed turbulent regime. The rapid decay of the lower frequencies is seen in both Figure 4.29 and Figure 4.30.

As the angle of attack increases, we observe a clear rise in the overall amplitude of the drag spectrum. This is consistent with the earlier flow field observations, where higher angles led to earlier flow separation, slight vortex shedding, and a more turbulent wake all affecting the drag on the kite. At low angles, the flow remains mostly attached with limited pressure difference and weak vortex activity, resulting in lower fluctuations. As the angle of attack increases, the flow becomes increasingly detached on the suction side, producing stronger unsteady aerodynamic forces. This leads to a broadband increase in spectral energy and a clear rise in turbulence intensity across the entire spectrum.

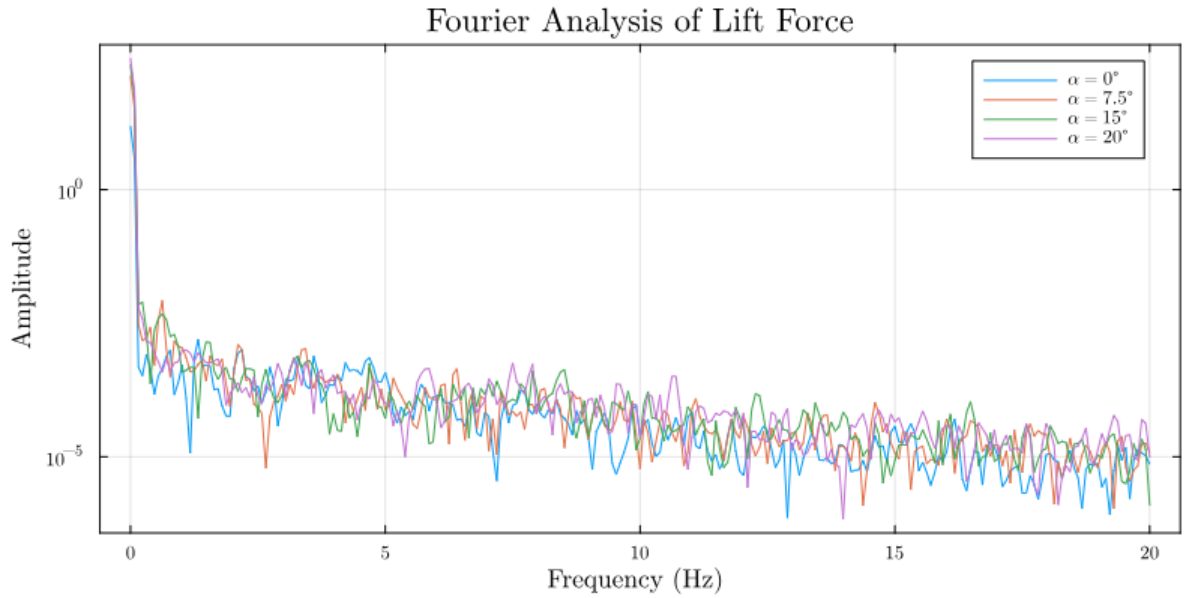


Figure 4.30: Fourier analysis of the time trace of the lift coefficient for different angles of attack

To quantify how often and by how much the force on the kite deviates from the mean we calculate a kernel density estimate, KDE, of the force fluctuations. These plots show a Kernel Density Estimate, not a histogram. The ”Frequency” axis represents probability density, not actual counts. Unlike a histogram, the area under each KDE curve integrates to 1, and the height reflects the relative likelihood of values—not their rate of occurrence. The fluctuations used for the analysis are defined as the difference between the force at a time and the mean force.

$$C'_{D,t} = C_{D,t} - C_D$$
$$C'_{L,t} = C_{L,t} - C_L$$

Where $C'_{D,t}$ and $C'_{L,t}$ are the fluctuations at time t and $C_{D,t}$ and $C_{L,t}$ are the total measured force at that time. We make a KDE of the force fluctuations. This shows us, for each simulation, the likelihood of a fluctuation.

The fluctuations in lift in Figure 4.31 show a clear dependence on the angle of attack, and supports earlier findings regarding from the spectral and flow field analysis. For $\alpha = 0°$ we see a high peak centered around 0 and a narrow band around it. This indicates that the total force on the kite is often close to or exactly the mean (only exact for $C'_L = 0$). When the force does fluctuate it does so only slightly. Increasing the angle of attack leads to more fluctuating forces. Seen by fewer occurrences of $C'_L = 0$ and a more spread out KDE. This also means that the force fluctuates more often, but also that when it does it has a larger deviation. However if we look at the fluctuations relative to the mean lift coefficient , $\frac{C'_L}{C_L}$, we see tat the relation flips. In Figure 4.32 we see that for $\alpha = 0°$ the relative fluctuations are a lot higher for the lower angles of attack.

We can also see the effect of the lack of flow reattachment for $\alpha = 7.5°$. While we see that the absolute fluctuations increase with the angle of attack, the fluctuations for $\alpha = 7.5°$ are even more spread out than for the higher angles of attack. This is a clear effect of the lack of reattachment. While the suction side gets more turbulent the pressure side becomes more laminar due to the reattachment seen for $\alpha = 15°$ & $\alpha = 20°$. The large amounts of turbulence on both the pressure and the suction side for $\alpha = 7.5°$ are a likely reason for the more fluctuation forces seen here.

In the fluctuations of the drag coefficient we see fairly similar results. The drag coefficient is more steady at low angles of attack, where we are extremely likely to have no fluctuation at all, Figure 4.33. Compared to the lift coefficients the fluctuations in drag are also most steady at low angles of when comparing

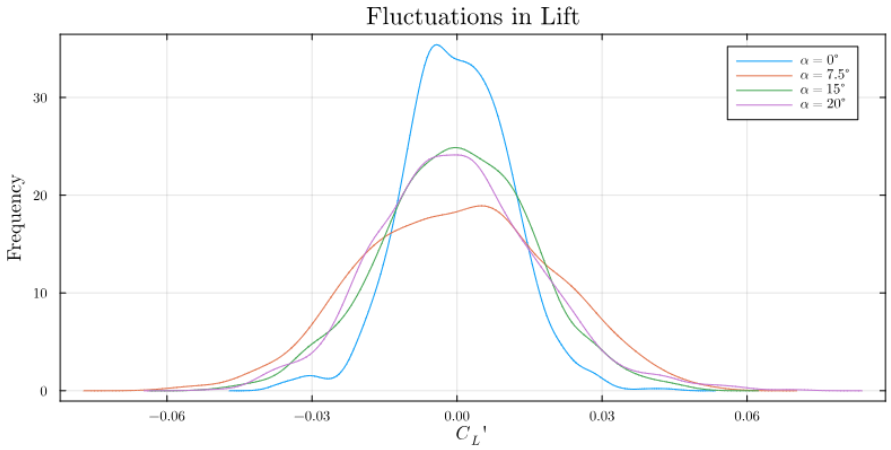


Figure 4.31: Kernel density estimate of the fluctuations in the lift coefficient, C'_L for different angles of attack

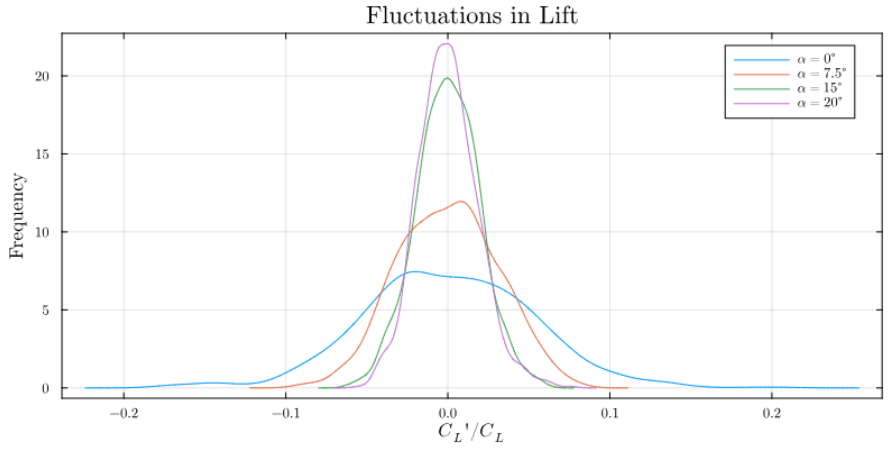


Figure 4.32: Kernel density estimate of the fluctuations in the lift coefficient, C'_L relative to the mean lift coefficient C_L for different angles of attack

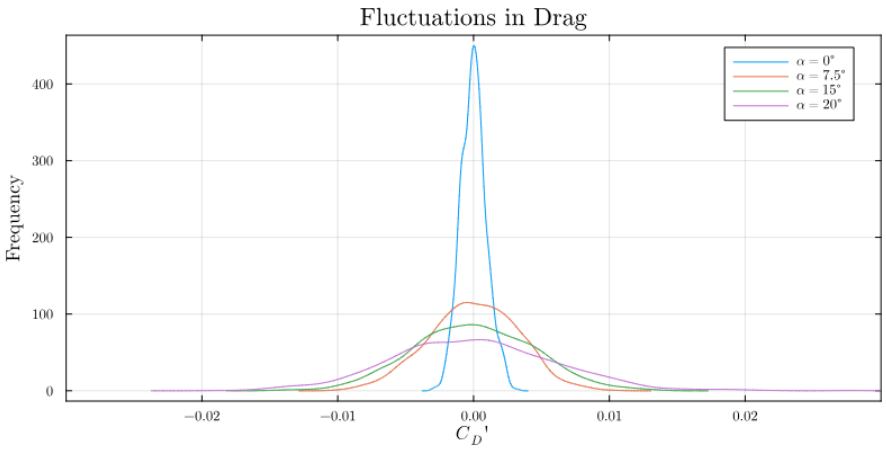


Figure 4.33: Kernel density estimate of the fluctuations in drag force, C_D' for different angles of attack

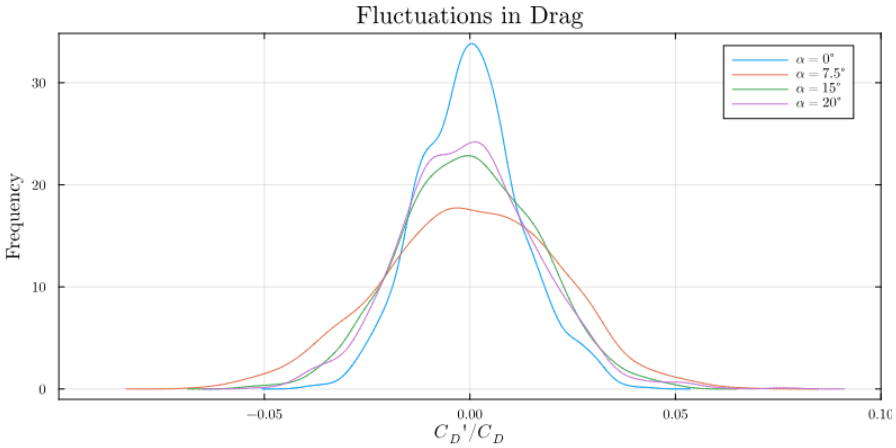


Figure 4.34: Kernel density estimate of the fluctuations in drag force, C_D' relative to the mean lift coefficient C_D for different angles of attack

relative fluctuations in Figure 4.34. This does not go for $\alpha = 7.5°$ where the relative fluctuations in drag are more extreme. We believe this is also a consequence of the detachment underneath the leading edge.

4.2.5. Pressure probes

For the final part of the analysis we will investigate 15 points along the the pressure and suction side of the kite. The points ar located on the centerline of the kite and the locations are chosen to be just outside of the transition zone such that:

$$|\mu_0(x)| = \sqrt{3}$$

This means all we are out of the transition zone in all three directions. We use the simulation for $\alpha = 15°$ and run it for an additional 15 time steps during which we record the pressure at the points every 0.1 time steps. The location of the points on the $\mu_{0,x}$ field can be seen in Figure 4.35.

The time traces in Figure 4.36 of for each point show that the Pressure side has more variation across space than the suction side which corresponds with the earlier observation that the suction side exhibits a full turbulent regime while the pressure side has a turbulent region that is largely dissipated farther down. Applying a Fourier transform to these time traces reveals some interesting phenomena. Figure 4.37 shows only part of the Fourier transform for clarity. Overall we see a clear decay for higher frequencies. At the Suction side at $x = 260$, which is near the front of the kite. We see a peak around 3 Hz. This peak is the result of the regular vortex shedding coming from the leading edge which could be seen in Figure 4.19. Farther down the chord we only see a steady decay and a low magnitude. We saw that

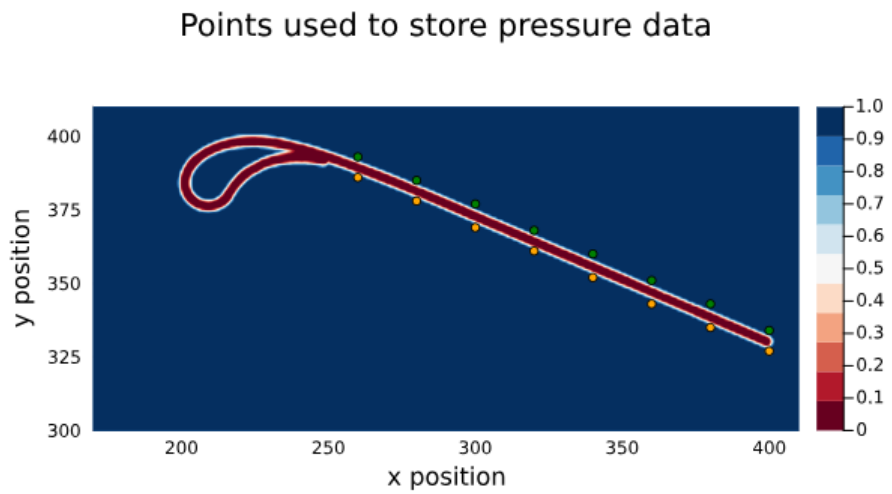
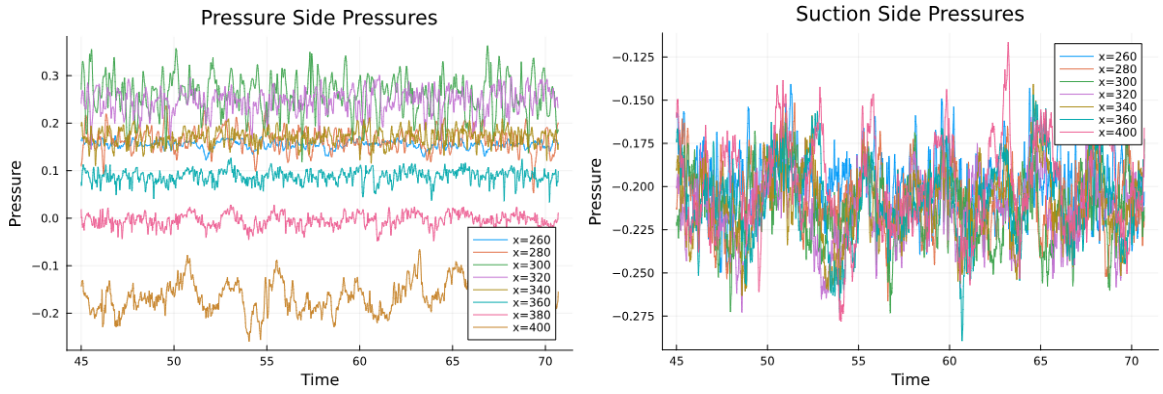


Figure 4.35: Points at which the pressure was recorded for 15 additional time steps



(a) Pressure side

(b) Suction Side

Figure 4.36: Time traces of the recorded pressure, labels indicate the x position of the point at which the data was recorded

for higher angles of attack that the suction side gets more turbulent. However we now see that while the flow is turbulent the pressure at the kites surface is fairly steady. This also explains the low force fluctuations. At the pressure side we do not see any clear peaks. However we do see that there is more higher frequency behavior clos to the kite. At $x = 300$ we see the pressure fluctuates a lot. This point corresponds to where the turbulence from the leading edge hits the canopy leading to higher frequency changes. Farther down the kite we see generally very little fluctuations at all indicating the flow has re-attached itself.

We have also plotted the time averaged pressure over the surface of the kite. We see that at the suction side the pressure is very steady over the length of the canopy. However at the pressure side we see the clear effect of the turbulent region behind the leading edge. Close to the leading edge we see a high pressure area, but farther down the kite where we have already established the flow re-attachment, the flow runs parallel to the kite again and is able to pick up more speed leading to the lower pressures we see.

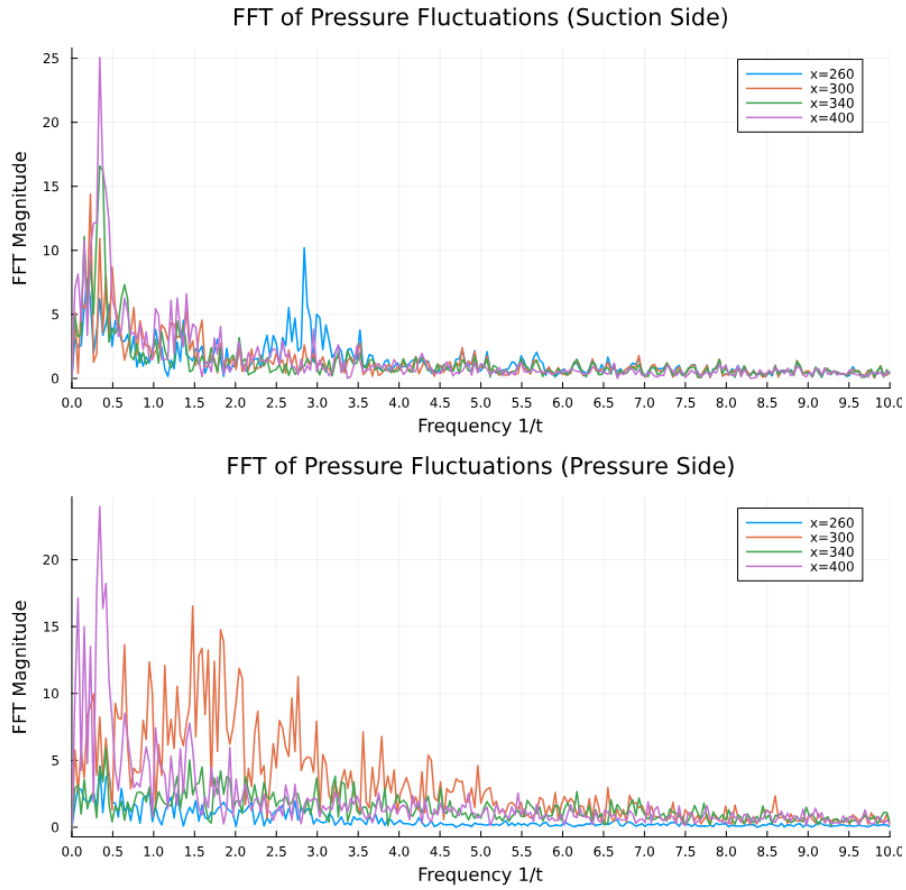


Figure 4.37: Fourier transforms of the pressure value at different points along the kite

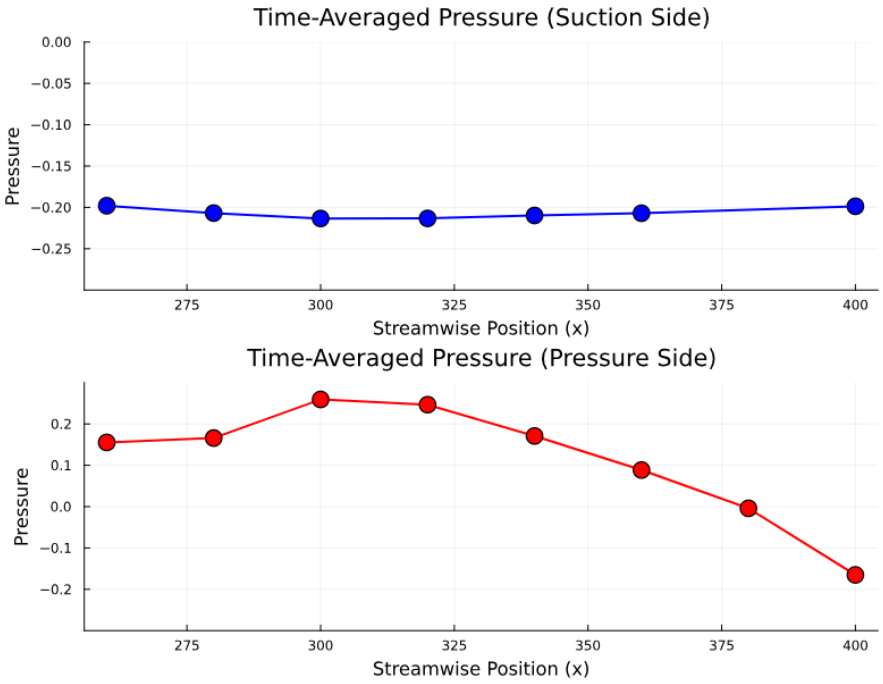


Figure 4.38: Time averaged pressure on the pressure and suction side of the kite

5

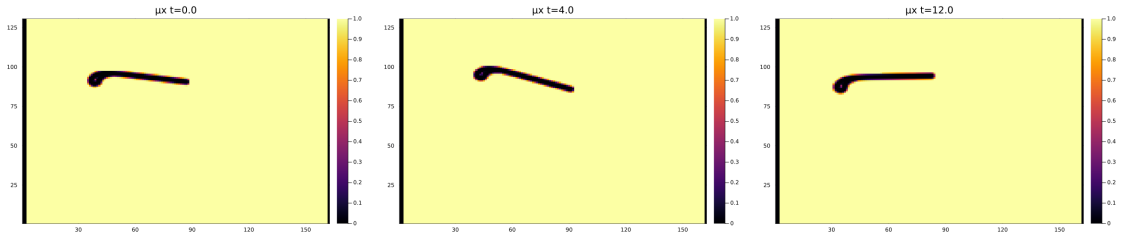
Proof of concept: Dynamic simulation

In order to show the benefit of the BDIM we will perform a simulation in which the kite moves around within the domain. Since the solver relies solely on a Cartesian grid we do not need to construct a new mesh at every time step, which can be very computationally expensive. Instead we need to reconstruct the signed distance function and calculate the kernel moments, μ_0 and μ_1.

5.1. Simulation set-up

We use a small simulation domain of 160 by 256 by 128 cells in which we put a kite with a chord length of 50 cells at 60 cells from the inflow. This small chord length results in a thick kite due to the minimum thickness, this is not representative of the actual geometry. This simulation is not meant to provide high fidelity data, but rather to show the capabilities of the solver and to justify the use of a BDIM solver.
We will force the kite to pitch directly by prescribing the motion. The kite will be pitching between an angle of attack of $\pm 10°$ with a period of 15 time steps. Since we are simulating at a low resolution we scale the Reynolds number down to $Re = 1e4$ to ensure the flow is resolved at the wall. Snapshots of the $\mu_{0,x}$ field at different time steps, at the centerline can be seen in Figure 5.1. We only run the simulation for the duration of one period.

$$\alpha(t) = 10 * sin(2\pi * t/15)$$



(a) $t = 0.0$ s **(b)** $t = 4.0$ s **(c)** $t = 12.0$ s

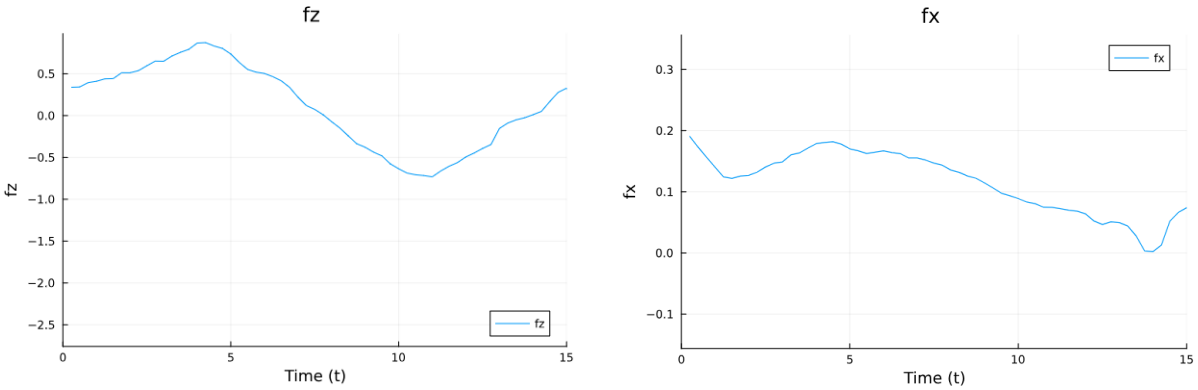
Figure 5.1: $\mu_{0,x}$ field at the centerline of the kite at different timesteps.

Domain size	160 x 256 x 128
Number of cells	5,242,880
Chord	50 cells
A	7491.908 cells^2
Re	10000

Table 5.1: Domain parameters for dynamic simulation

5.2. Results

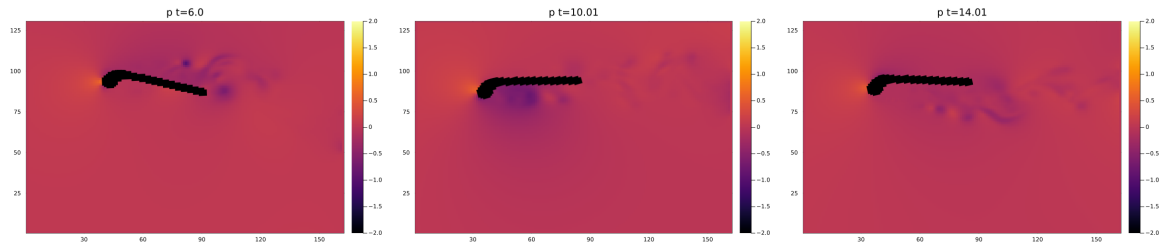
Figures 5.2a and 5.2b show that we can obtain force readings from the dynamic simulation. Part of this process is that the normal field has to be remeasured every time we want to calculate the forces which is optimized using the BVH method but is still has a large computational cost. We see in the time traces that the force on the kite is highly dependent on the angle of attack and the effect of the pitch can clearly be seen. Figures 5.3,5.4 and 5.5 show the pressure, velocity in x direction and velocity in z direction respectively. The main goal of these figures is to show that the solver is able to handle the changing immersed boundaries at each time step.



(a) Time trace of drag force during one period.

(b) Time trace of lift force during one period.

Figure 5.2: Time histories of drag and lift forces during one period of motion.

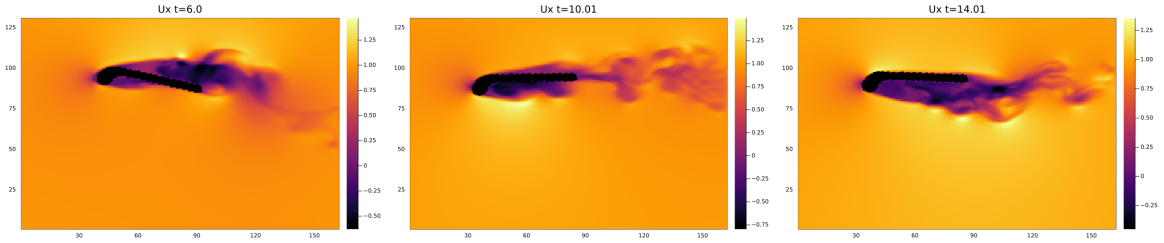


(a) $t = 6.0$ s

(b) $t = 10.01$ s

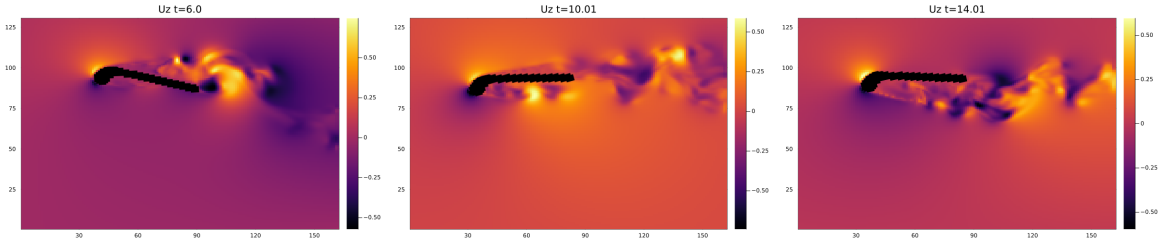
(c) $t = 14.01$ s

Figure 5.3: Pressure field snapshots during dynamic motion.



(a) $t = 6.0$ s **(b)** $t = 10.01$ s **(c)** $t = 14.01$ s

Figure 5.4: Streamwise velocity (u_x) field snapshots during dynamic motion.



(a) $t = 6.0$ s **(b)** $t = 10.01$ s **(c)** $t = 14.01$ s

Figure 5.5: Vertical velocity (u_z) field snapshots during dynamic motion.

This simulation is in no way a good representation of the flow dynamics, but clearly demonstrates the ability to perform dynamic simulations with the BDIM and *WaterLily*. We have provided only a proof of concept here for such future simulations that want to analyze complex movements of the kite or other immersed objects.

6

Conclusion

The objective of this thesis was to improve the efficiency of submerging complex meshes in *WaterLily*. We then applied the approach to a kite geometry as a use case. The motivation is clear. If we can submerge a complex shape such as a kite in an LES solver at practical cost, we can study the fluctuating forces that arise at high Reynolds number.

We developed a Bounding Volume Hierarchy based pipeline that reduces the number of distance evaluations between grid points and mesh elements. This led to a significant speed up while preserving the accuracy of the signed distance function. The method was validated for both bounding and non bounding meshes. We also introduced a new pressure force routine that integrates loads by looping over mesh elements rather than grid points. This routine matches the existing results and is faster when the grid size is much larger than the mesh size. It does not yet measure forces accurately on non bounding meshes, which is a limitation.

The pipeline was used to submerge the TU Delft LEI V3 kite [1]. We performed wall resolved simulations and found that mean forces agree with existing data. The forces fluctuate in time, but the amplitude is only a few percent of the mean. Increasing the angle of attack makes the flow more turbulent and increases the absolute fluctuations. Relative to the larger mean forces at higher angles, the fluctuations are smaller in percentage terms.

The results show that for steady flight, Large Eddy Simulation is consistent with other methods. The added turbulence resolution affects the forces only slightly. For static performance estimates we therefore recommend computationally cheaper approaches such as RANS. For dynamic flight, where the flow is more unsteady, our proof of concept shows that the current framework can handle the problem. Since turbulence does influence the loads, we recommend further LES studies of dynamic flight.

Future work should extend the force routine to non bounding meshes, assess grid and bandwidth sensitivity at higher Reynolds numbers, and explore wall modeled LES to reach larger scales within the same computational budget.

References

[1] Airborne Wind Energy Research Group. *TUDELFT_V3_KITE: TU Delft V3 Leading□Edge Inflatable Kite Data Repository*. https://github.com/awegroup/TUDELFT_V3_KITE. Accessed 2025-06-08. 2025.

[2] *Airseas: Seawing system*. 2023. url: https://airseas.com/en/seawing-system/.

[3] B. Akay et al. “An actuator sector method for efficient transient wind turbine simulation”. In: *Wind Energy* March 2014 (2013), pp. 1–20. doi: 10.1002/we.

[4] J. Andreas Bærentzen and Henrik Aanæs. “Signed Distance Computation Using the Angle Weighted Pseudonormal”. In: *IEEE Transactions on Visualization and Computer Graphics* 11.3 (2005), pp. 243–253. doi: 10.1109/TVCG.2005.49.

[5] Sanjeeb T. Bose and George I. Park. “Wall-Modeled Large-Eddy Simulation for Complex Turbulent Flows”. In: *Annual Review of Fluid Mechanics* 50 (2018), pp. 535–561. doi: 10.1146/annurev-fluid-122316-045241.

[6] Evert A. Bouman et al. “State-of-the-art technologies, measures, and potential for reducing GHG emissions from shipping – A review”. In: *Transportation Research Part D: Transport and Environment* 52 (2017), pp. 408–421. doi: https://doi.org/10.1016/j.trd.2017.03.022.

[7] Robert H. Bush et al. “Recommendations for future efforts in RANS modeling and simulation”. In: *AIAA Scitech 2019 Forum* (2019), pp. 1–19. doi: 10.2514/6.2019-0317.

[8] Oriol Cayon, Mac Gaunaa, and Roland Schmehl. “Fast Aero-Structural Model of a Leading-Edge Inflatable Kite”. In: *Energies* 16.7 (2023), pp. 1–19. doi: 10.3390/en16073061.

[9] Antonello Cherubini et al. “Airborne Wind Energy Systems: A review of the technologies”. In: *Renewable and Sustainable Energy Reviews* 51 (2015), pp. 1461–1476. doi: 10.1016/j.rser.2015.07.053.

[10] Floyd M. Chitalu, Christophe Dubach, and Taku Komura. “Binary Ostensibly-Implicit Trees for Fast Collision Detection”. In: *Computer Graphics Forum* 39.2 (2020), pp. 509–521. doi: 10.1111/cgf.13948.

[11] George S. Constantinescu and Kyle D. Squires. “LES and DES investigations of turbulent flow over a sphere”. In: *Flow,Turbulence and Combustion* (2003), pp. 267–298. doi: 10.2514/6.2000-540.

[12] George M. Dadd, Dominic A. Hudson, and R. A. Shenoi. “Comparison of two kite force models with experiment”. In: *Journal of Aircraft* 47.1 (2010), pp. 212–224. doi: 10.2514/1.44738.

[13] George M. Dadd, Dominic A. Hudson, and R. A. Shenoi. “Determination of kite forces using three-dimensional flight trajectories for ship propulsion”. In: *Renewable Energy* 36.10 (2011), pp. 2667–2678. doi: 10.1016/j.renene.2011.01.027.

[14] EDGAR/JRC. *Carbon dioxide emissions from international shipping worldwide from 1970 to 2023 (in million metric tons) [Graph]*. In Statista. Retrieved January 23, 2025, from https://www.statista.com/statistics/1291468/international-shipping-emissions-worldwide/. 2024.

[15] Dylan Eijkelhof, Nicola Rossi, and Roland Schmehl. “Optimal Flight Pattern Debate for Airborne Wind Energy Systems : Circular or Figure-of-eight ”. In: December (2024), pp. 1–24.

[16] Michael Erhard and Hans Strauch. “Control of towing kites for seagoing vessels”. In: *IEEE Transactions on Control Systems Technology* 21.5 (2013), pp. 1629–1640. doi: 10.1109/TCST.2012.2221093. arXiv: 1202.3641.

[17] Michael Erhard and Hans Strauch. “Flight control of tethered kites in autonomous pumping cycles for airborne wind energy”. In: *Control Engineering Practice* 40 (2015), pp. 13–26. doi: 10.1016/j.conengprac.2015.03.001.

[18] Michael Erhard and Hans Strauch. “Theory and Experimental Validation of a Simple Comprehensible Model of Tethered Kite Dynamics Used for Controller Design”. In: *Green Energy and Technology*. 2013. isbn: 9783642399657. doi: 10.1007/978-3-642-39965-7.

[19] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. “Distance Transforms of Sampled Functions”. In: *Theory of Computing* 8 (2012), pp. 415–428. doi: 10.4086/toc.2012.v008a019.

[20] Mikko Folkersma, Roland Schmehl, and Axelle Viré. “Boundary layer transition modeling on leading edge inflatable kite airfoils”. In: *Wind Energy* 22.7 (2019), pp. 908–921. doi: 10.1002/we.2329.

[21] Jennifer A. Franck and Kenneth S. Breuer. “Unsteady high-lift mechanisms from heaving flat plate simulations”. In: *International Journal of Heat and Fluid Flow* 67 (2017), pp. 230–239. doi: 10.1016/j.ijheatfluidflow.2017.08.012. arXiv: 1703.10906.

[22] Jochen Fröhlich and Dominic von Terzi. “Hybrid LES/RANS methods for the simulation of turbulent flows”. In: *Progress in Aerospace Sciences* 44.5 (2008), pp. 349–377. doi: https://doi.org/10.1016/j.paerosci.2008.05.001.

[23] Eric Garnier et al. “On the use of shock-capturing schemes for large-eddy simulation”. In: *Journal of Computational Physics* 153.2 (1999), pp. 273–311. doi: 10.1006/jcph.1999.6268.

[24] Massimo Germano et al. “A dynamic subgrid-scale eddy viscosity model”. In: *Physics of Fluids A* 3.7 (1991), pp. 1760–1765. doi: 10.1063/1.857955.

[25] Sandip Ghosal. “An analysis of numerical errors in large-eddy simulations of turbulence”. In: *Journal of Computational Physics* 125.1 (1996), pp. 187–206. doi: 10.1006/jcph.1996.0088.

[26] T. Haas et al. “Wake characteristics of pumping mode airborne wind energy systems”. In: *Journal of Physics: Conference Series* 1256.1 (2019). doi: 10.1088/1742-6596/1256/1/012016.

[27] Michal Hapala et al. “Efficient stack-less BVH traversal for ray tracing”. In: *Proceedings - SCCG 2011: 27th Spring Conference on Computer Graphics* May 2014 (2013), pp. 7–12. doi: 10.1145/2461217.2461219.

[28] Homer. *The Odyssey*. Trans. by Samuel Butler. Public domain translation. London: A. C. Fifield, 1900. url: https://www.gutenberg.org/ebooks/1727.

[29] Boris Houska and Moritz Diehl. “Optimal control of towing kites”. In: *Proceedings of the IEEE Conference on Decision and Control* 3 (2006), pp. 2693–2697. doi: 10.1109/cdc.2006.377210.

[30] ICCT. *Maritime shipping - International Council on Clean Transportation*. url: https://theicct.org/sector/maritime-shipping/.

[31] Kiran Jadhav and Abhilash J. Chandy. “Assessment of SGS Models for Large Eddy Simulation (LES) of a Stratified Taylor–Green Vortex”. In: *Flow, Turbulence and Combustion* 106.1 (2021), pp. 37–60. doi: 10.1007/s10494-020-00175-5.

[32] S. Jakirlić and R. Maduta. “Extending the bounds of ‘steady’ RANS closures: Toward an instability-sensitive Reynolds stress model”. In: *International Journal of Heat and Fluid Flow* 51 (2015). Theme special issue celebrating the 75th birthdays of Brian Launder and Kemo Hanjalic, pp. 175–194. doi: https://doi.org/10.1016/j.ijheatfluidflow.2014.09.003.

[33] Mark W. Jones, J. Andreas Baerentzen, and Milos Sramek. “3D Distance Fields: A Survey of Techniques and Applications”. In: *IEEE Transactions on Visualization and Computer Graphics* 12.4 (2006), pp. 581–599. doi: 10.1109/TVCG.2006.56.

[34] Mher M. Karakouzian, Mojtaba Kheiri, and Frédéric Bourgault. “A survey of two analytical wake models for crosswind kite power systems”. In: *Physics of Fluids* 34.9 (2022). doi: 10.1063/5.0102388. arXiv: 2111.04115.

[35] Mojtaba Kheiri et al. “Power Systems Power Systems”. In: 40.5 (2021), pp. 1183–1190.

[36] B C D Peck L Khan J J R Macklin, O Morton, and J-B R G Souppez. “a Review of Wind-Assisted Ship Propulsion for Sustainable Commercial Shipping :” in: *Wind Propulsion Conference, 15th-16th September 2021, London, UK* September (2021).

[37] W. J. Rider L. G. Margolin and F. F. Grinstein. “Modeling turbulent flow with implicit LES”. In: *Journal of Turbulence* 7 (2006), N15. doi: 10.1080/14685240500331595.

[38] Marin Lauber. *Computational Fluid-Structure Interaction of Membranes and Shells with Application to Bat Flight*. ORCID: 0000-0003-2191-9318. Mar. 2023.

[39] Geert Lebesque. “Steady-State RANS Simulation of a Leading Edge Inflatable Wing with Chordwise Struts”. Available via Google Scholar. Master’s Thesis. Delft, The Netherlands: Delft University of Technology, 2020.

[40] D. K. Lilly. “A proposed modification of the Germano subgrid-scale closure method”. In: *Physics of Fluids A* 4.3 (1992), pp. 633–635. doi: 10.1063/1.858280.

[41] Audrey P. Maertens and Gabriel D. Weymouth. “Accurate Cartesian-grid simulations of near-body flows at intermediate Reynolds numbers”. In: *Computer Methods in Applied Mechanics and Engineering* 283 (2015), pp. 106–129. doi: 10.1016/j.cma.2014.09.009.

[42] Claudio Mannini et al. “URANS and DES simulation of flow around a rectangular cylinder”. In: *Notes on Numerical Fluid Mechanics and Multidisciplinary Design* 96 (2022), pp. 36–43. doi: 10.1007/978-3-540-74460-3_5.

[43] Peter Naaijen and Vincent Koster. “Performance of auxiliary wind propulsion for merchant ships using a kite”. In: *2nd International Conference on Marine Research and Transportation* January 2007 (2007), p. 10.

[44] D. Peng et al. “A PDE Based Fast Local Level Set Method”. In: *Journal of Computational Physics* 155.2 (1999), pp. 410–438. doi: 10.1006/jcph.1999.6345.

[45] W. F. Phillips and D. O. Snyder. “Modern Adaptation of Prandtl’s Classic Lifting-Line Theory”. In: *Journal of Aircraft* 37.4 (2000), pp. 662–670. doi: 10.2514/2.2649.

[46] Ugo Piomelli and Elias Balaras. “Wall-layer models for large-eddy simulations”. In: *Annual Review of Fluid Mechanics* 34 (2002), pp. 349–374. doi: 10.1146/annurev.fluid.34.082901.144919.

[47] Eduard Pujol and Antonio Chica. “Adaptive Approximation of Signed Distance Fields Through Piecewise Continuous Interpolation”. In: *Computers & Graphics* 114 (2023), pp. 337–346. doi: 10.1016/j.cag.2023.06.020.

[48] Inigo Quilez. *Inigo Quilez*. url: https://iquilezles.org/articles/distfunctions/.

[49] G. Della Rocca and Grégoire Blanquart. “Level Set Reinitialization at a Contact Line”. In: *Journal of Computational Physics* 265 (2014), pp. 34–49. doi: 10.1016/j.jcp.2014.01.040.

[50] Giovanni Russo and Peter Smereka. “A Remark on Computing Distance Functions”. In: *Journal of Computational Physics* 163.1 (2000), pp. 51–67. doi: 10.1006/jcph.2000.6553.

[51] S. Sarkar. “Large Eddy Simulation of Flows of Engineering Interest: A Review”. In: *50 Years of CFD in Engineering Sciences: A Commemorative Volume in Memory of D. Brian Spalding*. Ed. by Akshai Runchal. Singapore: Springer Singapore, 2020, pp. 363–400. isbn: 978-981-15-2670-1. doi: 10.1007/978-981-15-2670-1_11.

[52] Stefan C. Schlanderer, Gabriel D. Weymouth, and Richard D. Sandberg. “The boundary data immersion method for compressible flows with application to aeroacoustics”. In: *Journal of Computational Physics* 333 (2017), pp. 440–461. doi: 10.1016/j.jcp.2016.12.050.

[53] J. A. Sethian. “A Fast Marching Level Set Method for Monotonically Advancing Fronts”. In: *Proceedings of the National Academy of Sciences of the USA* 93.4 (1996), pp. 1591–1595. doi: 10.1073/pnas.93.4.1591.

[54] *SkySails Marine: Kite Systems for Commercial Shipping*. https://skysails-marine.com/.

[55] J. SMAGORINSKY. “GENERAL CIRCULATION EXPERIMENTS WITH THE PRIMITIVE EQUATIONS: I. THE BASIC EXPERIMENT”. In: *Monthly Weather Review* 91.3 (1963), pp. 99–164. doi: 10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.

[56] Stanford Computer Graphics Laboratory. *The Stanford 3D Scanning Repository*. https://graphics.stanford.edu/data/3Dscanrep/. Accessed: 2025-05-21. 1994.

[57] Mark Sussman, Peter Smereka, and Stanley Osher. “A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow”. In: *Journal of Computational Physics* 114.1 (1994), pp. 146–159. doi: 10.1006/jcph.1994.1155.

[58] The Engineering ToolBox. *Air Temperature, Pressure and Density vs. Altitude*. Online. Accessed 20 June 2025. 2003. url: https://www.engineeringtoolbox.com/air-altitude-temperature-d_461.html.

[59] THE MARINE ENVIRONMENT PROTECTION COMMITTEE. *RESOLUTION MEPC.377(80) - 2023 IMO STRATEGY ON REDUCTION OF GHG EMISSIONS FROM SHIPS*. July 7, 2023. url: https://wwwcdn.imo.org/localresources/en/OurWork/Environment/Documents/annex/MEPC%2080/Annex%2015.pdf.

[60] Paul Thedens, Gael de Oliveira, and Roland Schmehl. “Ram-air kite airfoil and reinforcements optimization for airborne wind energy applications”. In: *Wind Energy* 22.5 (2019), pp. 653–665. doi: 10.1002/we.2313.

[61] J. E. Thomas and J-B. R. G. Souppez. “Comparative performance prediction of historical Thames A Rater class designs”. In: (2018).

[62] Greg Turk and Marc Levoy. *Stanford Bunny 3D Model*. https://graphics.stanford.edu/data/3Dscanrep/. Accessed: 2025-05-21. 1994.

[63] UNCTAD. *Review of Maritime Report 2024*. 2024. url: https://unctad.org/publication/review-maritime-transport-2024.

[64] P. Welch. “The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms”. In: *IEEE Transactions on Audio and Electroacoustics* 15.2 (1967), pp. 70–73. doi: 10.1109/TAU.1967.1161901.

[65] J.F. Wellicome and S. Wilkinson. *Propulsive kites – an initial study*. Technical Report SSSU 19. University of Southampton, Department of Ship Science, Faculty of Engineering and Applied Science, 1984.

[66] G. D. Weymouth and Dick K.P. Yue. “Boundary data immersion method for Cartesian-grid simulations of fluid-body interaction problems”. In: *Journal of Computational Physics* 230.16 (2011), pp. 6233–6247. doi: 10.1016/j.jcp.2011.04.022.

[67] G.D. Weymouth and Dick K.P. Yue. “Boundary data immersion method for Cartesian-grid simulations of fluid-body interaction problems”. In: *Journal of Computational Physics* 230.16 (2011), pp. 6233–6247. doi: https://doi.org/10.1016/j.jcp.2011.04.022.

[68] Gabriel D. Weymouth and Bernat Font. “WaterLily.jl: A differentiable and backend-agnostic Julia solver to simulate incompressible viscous flow and dynamic bodies”. In: 2022 (2024), pp. 1–12. arXiv: 2407.16032. url: http://arxiv.org/abs/2407.16032.

[69] Gabriel D. Weymouth and Marin Lauber. “Using Biot-Savart boundary conditions for unbounded external flow on Eulerian meshes”. In: *Journal of Computation Physics* (2024). eprint: 2404.09034. url: http://arxiv.org/abs/2404.09034.

[70] Yizi Wu, Jiaju Man, and Ziqing Xie. “A Double Layer Method for Constructing Signed Distance Fields from Triangle Meshes”. In: *Graphical Models* (2014). doi: 10.1016/j.gmod.2014.04.011.

[71] Tianju Xue et al. “A New Finite Element Level Set Reinitialization Method Based on the Shifted Boundary Method”. In: *Journal of Computational Physics* 438 (2021), p. 110360. doi: 10.1016/j.jcp.2021.110360.

[72] Xiang I.A. Yang and Kevin P. Griffin. “Grid-point and time-step requirements for direct numerical simulation and large-eddy simulation”. In: *Physics of Fluids* 33.1 (2021), pp. 1–11. doi: 10.1063/5.0036515. arXiv: 2010.15307.

[73] I. R. Young. “Seasonal variability of the global ocean wind and wave climate”. In: *International Journal of Climatology* 19.9 (1999), pp. 931–950. doi: 10.1002/(SICI)1097-0088(199907)19:9<931::AID-JOC412>3.0.CO;2-O.