NTNU

Kunnskap for en bedre verden

Department of Marine Technology

TMR4520 - Marine Hydrodynamics

Thrust loss due to interaction with ship hull and bottom for ships in shallow waters

Author:
Brage Møller-Pettersen

16.05.2022

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

MASTER THESIS IN MARINE TECHNOLOGY

SPRING 2022

FOR

Brage Møller-Pettersen

Thrust loss due to interaction with ship hull and bottom for ships in shallow waters

It is suspected that wind turbine installation ships and ships for wind turbine service, might experience severe thrust loss during operations in shallow water. It is believed that the thrust loss is caused by interaction between the propeller slip-stream (propeller jet) with both ship hull and the sea bottom. Interaction between hull, propeller and sea bottom is relevant also for other types of ships and operations, like ships travelling in rivers and inland waterways. It is important to understand the interaction effects, both to dimension the propulsion system properly, and to take the right actions to reduce those detrimental effects. The overall aim of the combined project and master thesis is to get an understanding of the physics of the effects, and establish simple methods to quantify the interaction.

To investigate the interaction between propeller slip-stream, ship hull and bottom, the following activities are foreseen in the master thesis: a thorough literature review of both literature on thrust loss and propeller hull interaction, as well as a review of the theory of turbulent jets and how they interact with boundaries. The main work will focus on numerical investigations using CFD..

The project and master will be performed in co-operation with Kongsberg Maritime University Technology Centre (UTC) at NTNU.

In the thesis the candidate shall present his personal contribution to the resolution of problem within the scope of the thesis work.

Theories and conclusions shall be based on mathematical derivations and/or logic reasoning identifying the various steps in the deduction.

The thesis work shall be based on the current state of knowledge in the field of study. The current state of knowledge shall be established through a thorough literature study, the results of this study shall be written into the thesis. The candidate should utilize the existing possibilities for obtaining relevant literature.

The thesis shall be organized in a rational manner to give a clear exposition of results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Telegraphic language should be avoided.

The thesis shall contain the following elements: A text defining the scope, preface, list of contents, summary, main body of thesis, conclusions with recommendations for further work, list of symbols and acronyms, reference and (optional) appendices. All figures, tables and equations shall be numerated.

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

The supervisor may require that the candidate, in an early stage of the work, present a written plan for the completion of the work. The plan shall include a budget for the use of laboratory or other resources that will be charged to the department. Overruns shall be reported to the supervisor.

The original contribution of the candidate and material taken from other sources shall be clearly defined. Work from other sources shall be properly referenced using an acknowledged referencing system.

The thesis shall be submitted electronically (pdf) in Inspera:

- Signed by the candidate
- The text defining the scope (this text) (signed by the supervisor) included

Supervisor : Professor Sverre Steen
Advisors : Leif Vartdal, Geir Åge Øye (Kongsberg Maritime)
Start : 17.01.2022
Deadline : 10.06.2022

Trondheim, 17.01.2022

Sverre Steen
Supervisor

ABSTRACT

The result envisioned for this project is a clear answer to the question: *Is there a particular effect, or a combination thereof occurring in shallow waters that could cause severe thrust loss*?

After studying a broad spectrum of literature and discussing the impact of the different ways thrust loss might be achieved in shallow waters, the Coanda effect seems the most likely candidate to affect the thrust directly. Both ventilation and thruster-to-thruster effects might be equally possible, but are not a specific danger for shallow waters, which has remained the main focus of this thesis project.

The hypothesis proposes no thrust loss specifically related to shallow waters. However, thrust loss due to the Coanda effect is considered since the propeller jet may interact with the new surface, namely the sea bottom, implying a counter interaction to the Coanda effect interacting with the hull.

By studying CFD techniques and analyzing a 2D flow case, the thesis attempts to demonstrate the Coanda effect with a RANS simulation; the goal has been to investigate the impact of shallow waters on the Coanda effect. In addition, the thesis focuses on simulating the flow realistically and considering the debate regarding the limits of the approach.

Besides a thorough dive into CFD methods, the thesis is concerned with the essence of the problem. Mainly what types of thrust loss are typical and known from the literature. The primary considerations are towards the nature of the interaction and if and how shallow water might impact the interaction.

Thruster-to-thruster interaction is here considered mainly a design problem. Ventilation is a possibility and has been a subject of interest, but since the relevant vessels considered are service or installation ships operating in dynamic positioning with strict restrictions regarding the climate ahead of operations, the necessary climate for ventilation to occur is an unlikely encounter.

SAMMENDRAG

Målet med denne master avhandlingen er et klart svar på spørsmålet: *Er det en spesifikk effekt, eller kombinasjon av flere som forekommer på grunt vann og forårsaker thrust tap?*

Etter et relativt bredt litteratursøk og diskusjoner om forskjellige måter thrust tap kan forekomme på grunt vann, er Coanda effekten ansett for å være den mest trolige kilden til interaksjon med propellen. Både ventilasjon og thruster-til-thruster effekter er mulige, men interaksjonene forblir det samme på grunt vann som på dypt vann, og blir derfor nedprioritert.

Hypotesen foreslår at det ikke blir noe thrust tap spesifikt relatert til grunt vann, men thrust tap relatert til Coanda effekten blir vurdert. Ettersom propellstrålen kan interagere med den nye overflaten som blir introdusert ved grunt vann, havbunnen. Impliserer dette en motreaksjon til eventuelle thrust tap fra interaksjon mellom propellstråle og skrog.

Ved å studere metoder brukt innen CFD og analysere en 2D strømning, blir det gjort et forsøk på å demonstrere Coanda effekten med en RANS simulasjon. Målet har vært å undersøke effekten av grunt vann på Coanda effekten. Det rettes fokus mot å simulere strømningen realistisk og diskutere begrensningene til tilnærmingen.

I tillegg til mye bakgrunn innen CFD er det lagt innsats i hva slags typer thrust tap som er kjent eller typisk fra literatur. Fokuset er da på opphavet til interaksjonen, for å kunne anslå hva grunt vann vil kunne bety for effekten.

Thruster-til-thruster interaksjoner er hovedsakelig vurdert som et design problem. Ventilasjon er av stor interesse, men med bakgrunn i de skipene som denne studien retter seg mot, service og installasjonsskip som opererer med dynamisk posisjonering, er det såpass strenge reglementer som dikterer handlingsrommet for en operasjon, at forholdene som skal til for at ventilasjon skal kunne forekomme er usannsynlig å møte på.

PREFACE

This thesis represents the last piece of the puzzle for my Master of Science in Marine Technology specializing in Marine Hydrodynamics. The project was conducted throughout the spring semester of 2022 at the Norwegian University of Science and Technology (NTNU) in Trondheim.

First and foremost, I would like to express my gratitude to Professor Sverre Steen at the department of Marine Technology. He provided essential insight and guidance in the effort through weekly meetings, which I have enjoyed a lot. Furthermore, I would like to thank Professor Lars Erik Holmedal for assisting with the theoretical background and introducing me to Postdoctoral Fellow Jianxun Zhu. He provided essential insight into the different solvers and techniques for mesh construction.

Furthermore, I greatly appreciate the help from Leif Vartdal and Geir Åge Øye from Kongsberg Maritime for taking the time to discuss the project and share earlier studies on the subject.

Lastly, I would like to thank my fellow office colleagues and my girlfriend Lise for all the support, patience, and attention this last semester, not to mention the last five years.

TABLE OF CONTENTS

Project description **i**

Abstract **iii**

Sammendrag **iv**

Preface **v**

List of Figures **ix**

List of Tables **xii**

Nomenclature **xiii**

1 Introduction **1**

2 Theoretical background **4**

2.1 Reynolds Averaged Navier Stokes 4

2.2 Generalized eddy viscosity 6

2.2.1 $k-\varepsilon$ model 9

2.2.2 $k-\omega$ model 9

2.2.3 SST $k-\omega$ model . . . 10
2.3 Jet flow . . . 12
2.3.1 Plane jet . . . 12
2.3.2 Circular jet . . . 14

3 Known sources of thrust loss **15**
3.1 Ventilation . . . 15
3.2 Thruster to thruster interaction . . . 17
3.3 The influence of current . . . 18
3.4 Increased resistance in shallow waters . . . 18
3.5 Interactions related to the Coanda effect . . . 19

4 Computational Fluid Dynamics Experiment **21**
4.1 Experimental setup for 2D simulation . . . 22
4.2 Numerical solver . . . 25
4.3 Boundary conditions . . . 27
4.4 Mesh sizes . . . 29
4.5 Validation . . . 31

5 Results **37**
5.1 Mesh refinement analysis . . . 37
5.1.1 The drag coefficient . . . 38
5.1.2 Lift coefficient . . . 39
5.1.3 Roll moment coefficient . . . 39
5.1.4 Velocity profiles from mesh refinement analysis . . . 40
5.2 Validation . . . 42

6 Discussion **47**
6.1 Mesh refinement analysis . . . 48
6.1.1 Drag coefficients . . . 48
6.1.2 Lift and roll moment coefficients . . . 48

6.1.3 Thoughts on the improvement of mesh convergence analysis . 49
6.2 Validation 49
6.3 The applied method 49

7 Conclusion **51**

Bibliography **52**

A Kongsberg products used for comparison **I**

B Figures produced in paraView **IV**

C Codes used in `OpenFoam` **VIII**

C.1 Equal for both solvers the 0 directory VIII
C.2 The constant directory XV
C.3 The system directory XVI
C.3.1 Used in `pimpleFOAM` XVI
C.3.2 Used in `simpleFOAM` XX

D Codes used in `GMSH` **XXVI**

D.1 Mesh of 2D hull with propeller XXVI
D.2 Mesh of 2D hull XLVI

LIST OF FIGURES

2.1 Early development of a turbulent jet, copy of a figure found in F. M. White and Majdalani 2006 13

3.1 Schematic presentation of different ventilation zones, Koushan 2004 . 16

3.2 The limit for ventilation at bollard condition, for n = 16Hz, from Kozlowska and Steen 2017 17

4.1 Conceptual drawing of the domain for numerical experiment 22

4.2 A conceptual plot of the velocity profile along a wall, along with the plotted wall functions when applied correctly; created by user Aokomoriuta 2022 24

4.3 Some named properties to avoid confusion 29

4.4 The different meshing line groups, r symbolises the expansion ratio for each consecutive cell. All meshing lines are defined so that the clustering of cells happen towards the hull. 30

4.5 Influence of rounding radius upon the drag coefficient of various blunt bodies; Hoerner 1965 32

4.6 A mirrored rectangular cross section 32

4.7 Influence of splitter plates (and similar devices) on the drag coefficients without wake interference. $\mathfrak{Re} \in [10^4,\ 10^5]$; Hoerner 1965 . . . 33

4.8 Simple vortex system with an image flow above the free surface so the rigid free surface condition is satisfied, from Faltinsen 1993 34

4.9 Wake development for a particular hull cross section. Calculations are shown for different time instants t and area based on numerical calculations by Aarsnes et al. 1985 with a thin free shear layer model; found in Faltinsen 1993 34

4.10 Calculated and estimated drag coefficients C_D for two dimensional cross flow past cross sections along a particular ship; from Faltinsen 1993 35

4.11 The different meshing line groups, r symbolises the expansion ratio for each consecutive cell. All meshing lines are defined so that the clustering of cells happen towards the hull. 36

5.1 Average C_D for the front of the hull, with plotted divergence from mean of the last four calculated values 38

5.2 Average C_D for the rear part of the hull, with plotted divergence from mean of the last four calculated values 38

5.3 Average C_D for the entire hull, with plotted divergence from mean of the last four calculated values 39

5.4 Average C_L for the hull, with plotted divergence from mean of the last four calculated values 39

5.5 Average C_{mRoll} for the hull, with plotted divergence from mean of the last four calculated values 40

5.6 Velocity profile of $u_2 = v$ from the transient solver pimpleFoam with $\frac{W_D}{T} = 1.85$, $t = 39$s, the lowest resolution with 10^4 cells 40

5.7 Velocity profile from the transient solver pimpleFoam with $\frac{W_D}{T} = 1.85$, $t = 23$s, second lowest resolution with $2.5 \cdot 10^4$ cells 41

5.8 Velocity profile from the transient solver pimpleFoam with $\frac{W_D}{T} = 1.85$, $t = 87$s, third largest resolution with $5 \cdot 10^5$ cells 41

5.9 Velocity profile from the transient solver pimpleFoam with $\frac{W_D}{T} = 1.85$, $t = 69$s, second to largest resolution with $7.5 \cdot 10^4$ cells 42

5.10 Velocity profile from the transient solver pimpleFoam with $\frac{W_D}{T} = 1.85$, $t = 16$s, the largest resolution with 10^5 cells 42

5.11 The validation results from the pimpleFoam solver, with the expected value of the drag coefficient outlined 43

5.12 Instantaneous snapshot of the velocity field from pimpleFoam transient simulation, $\mathfrak{Re} = 10^7$, $\frac{W_D}{T} = 4$ and $t = 100$s 44

5.13 The validation results from the simpleFoam solver, with the expected value of the drag coefficient outlined 44

5.14 Snapshot of the steady state velocity field from the simpleFoam simulation, $\mathfrak{Re} = 10^5$, $\frac{W_D}{T} = 3$ and $t = 100$s 45

5.15 C_D as a function of time plotted with expected value range found in 4.5, computed with the pimleFoam solver, $\frac{W_D}{T} = 3$, $\mathfrak{Re} = 10^7$, n_{nodes} is the number of nodes used and k is meant to symbolize 10^3 46

A.1 Example vessel used to create generic hull shape III

B.1 Instantaneous snapshot of the velocity field from pimpleFoam transient simulation, $\mathfrak{Re} = 10^7$, $\frac{W_D}{T} = 3$ and $t = 100$s IV

B.2 Instantaneous snapshot of the velocity field from pimpleFoam transient simulation, $\mathfrak{Re} = 10^5$, $\frac{W_D}{T} = 3$ and $t = 100$s V

B.3 Instantaneous snapshot of the velocity field from pimpleFoam transient simulation, $\mathfrak{Re} = 10^5$, $\frac{W_D}{T} = 4$ and $t = 100$s V

B.4 Snapshot of the steady state velocity field from the simpleFoam simulation, $\mathfrak{Re} = 10^5$, $\frac{W_D}{T} = 3$ and $t = 100$s VI

B.5 Snapshot of the steady state velocity field from the simpleFoam simulation, $\mathfrak{Re} = 10^7$, $\frac{W_D}{T} = 3$ and $t = 100$s VI

B.6 Snapshot of the steady state velocity field from the simpleFoam simulation, $\mathfrak{Re} = 10^7$, $\frac{W_D}{T} = 4$ and $t = 100$s VII

LIST OF TABLES

1.1 Different transition Reynolds numbers predicted by an assortment of methods 3

2.1 Constants of incorporated $k-\omega$ model 11

2.2 Constants of incorporated $k-\varepsilon$ model 12

4.1 Example ship main dimensions 22

4.2 Flow domain characteristics 22

4.3 Boundary types 28

4.4 Initial conditions 28

4.5 Characteristic sizes for mesh refinement process 31

5.1 Amount of time required for each simulation to reach 100 simulated seconds 37

A.1 Kongsberg US TYPE AZIMUTHING THRUSTER II

NOMENCLATURE

Abbreviations

$\overline{\text{overbar}}$	The line over a certain quantity signifies the mean component, page 5
BP	Bollard pull, page 25
DNS	Direct Numerical Simulation, page 27
DP	Dynamic positioning, page 1
NS	Navier-Stokes equations , see equation (2.2), page 5
prime$'$	The prime symbol is used to signify the fluctuating component of the quantity, page 5
RPM	Revolutions per minute, almost always converted to RPS, page 16
RPS	Revolutions per second , page 16 Hz

Dimensionless numbers

$\mathcal{F}n$	Froude number , see equation (4.6), page 25
$\mathcal{S}t$	The Strouhal number, page 33
$\mathcal{W}e$	The Weber number , see equation (3.2), page 16
$\mathfrak{Re}$	The Reynolds number, page 2
$\mathfrak{Re}_{x,\text{tr}}$	Reynolds number signifying the fully turbulent flow transition, page 2
J	The advance coefficient , see equation (3.1), page 16
y^+	The y-pluss value , see equation (4.1), page 23

Empirical constants

α	Empirical constant used in the $k-\omega$ method, page 10	
β	Empirical constant used in the $k-\omega$ method, page 10	
β^*	Empirical constant used in the $k-\omega$ method, page 10	
ϕ	Representing the combination of constants in the $k-\omega$ SST method , see equation (2.23), page 11	
ϕ_1	Representing the constants of the $k-\omega$ method incorporated in the $k-\omega$ SST method , see equation (2.23), page 11	
ϕ_2	Representing the constants of $k-\varepsilon$ method incorporated in the $k-\omega$ SST method , see equation (2.23), page 11	
σ_ω	Empirical constant used in the $k-\omega$ method, page 10	
σ_ω^*	Empirical constant used in the $k-\omega$ method, page 10	
σ_ε	One of five empirical constants used in the $k-\varepsilon$ method, page 9	
σ_k	Constant related to diffusion when applying the Kolmogorov-Prandtl relation, page 8	
$c_{1\varepsilon}$	One of five empirical constants used in the $k-\varepsilon$ method, page 9	
c_1	The factoring of constants C_μ and C_{di} results in one of the five empirical constants used in the $k-\varepsilon$ method, page 9	
$c_{2\varepsilon}$	One of five empirical constants used in the $k-\varepsilon$ method, page 9	
C_μ	Constant from Kolmogorov-Prandtl relation , see equation (2.10), page 7	
C_{di}	Constant related to dissipation when applying the Kolmogorov-Prandtl relation, page 8	
F_1	Blending function representing the amount priority given to the $k-\omega$ method incorporated in the $k-\omega$ SST method, page 11	

Other symbols

$\dot{m}_S$	Mass flow through propeller in full scale, page 25	
$\mathcal{L}$	A characteristic length , page 2	m
$\mathcal{Q}_J$	Jet flow momentum , page 13	kgm/s
ν_{CFD}	Kinematic viscosity used in CFD simulations, page 24	
ν_T	The eddy viscosity. For the formulation used in CFD experiments see equation (2.27), for earliest mention , see equation (2.8), page 6	
τ_{ij}	The stress tensor , page 10	$\mathrm{N/m^2}$

τ_w	Wall shear stress , see equation (4.1), page 23	
ε	Dissipation terms for the turbulent kinetic energy , see equation (2.14), page 7	
B	Beam or breadth of hull, page 24	
b_J	The half width of the jet , page 13	m
D_k	Diffusion terms for the turbulent kinetic energy , see equation (2.12), page 7	
d_p	Propeller diameter , page 16	m
I	Turbulence intensity , page 27	%
l_m	Characteristic length scale of the largest turbulent eddies , page 6	m
n	Revolutions per second , page 16	Hz
n_{thruster}	Number of thrusters, page 25	
p	Pressure , page 5	N/m^2
P_k	Production terms for the turbulent kinetic energy , see equation (2.13), page 7	
T	Draft or draught , page 1	m
u_τ	The friction velocity , see equation (4.1), page 23	
u_i	Velocity components in Cartesian coordinates, using Einstein summation convention , page 5	m/s
V_m	Characteristic velocity scale of the largest turbulent eddies , page 6	m/s
W_D	Water depth , page 1	m
x_i	Cartesian coordinates using Einstein summation convention, page 5	
z	Vertical coordinate with origin at the fluid surface, page 19	

Physics constants

δ_{ij}	The Dirac delta function, page 6	
ν	Kinematic viscosity , page 2	m^2/s
ρ	Density of seawater, considered constant throughout the thesis, page 5	
S	Fluid surface tension , page 16	N/m

CHAPTER

1

INTRODUCTION

In most marine operations, available thrust and global loads are paramount for positioning and performance. Deep water conditions are typically the only considered environment during the design phase and model testing. Meaning many systems in use are optimized for deep water maneuvering. In shallow water, where this report refers to a water depth to draft ratio $\frac{W_D}{T} < 3$, blockage of the flow underneath the hull leads to increased global loads applied to the hull. During operations involving dynamic positioning, or DP, thruster systems are the primary tool to counteract the effects of waves and currents. A thorough understanding of potential risks related to thruster performance reinforces operational safety, especially during DP. This thesis aims to study the effects of shallow water and attempt to describe how the jet of a thruster might interact with the shallow water conditions. The primary tool used in CFD simulations is OpenFOAM.

In order to achieve the goal, numerical investigations will be made towards a 2D generic hull experiencing cross-flow at different depths of water, with a control volume underneath the hull simulating a propeller disk. Furthermore, by studying the jet flow behind the hull, the effect of the bottom concerning the Coanda effect will be assessed.

The flow domain is considered to be turbulent. In F. M. White and Majdalani 2006 a few different methods for computing the transitional Reynolds numbers over a flat plate was demonstrated. By comparing different methods, namely Michel 1952, Granville 1953, Cebeci and Smith 1974 and Wazzan et al. 1981 the final onset of turbulent flow can be predicted, $\mathfrak{Re}_{x,\,\mathrm{tr}}$, the results are displayed in table 1.1. The predicted values are noticeably lower than equation (1.1). As this project revolves around jet flow and otherwise displaced flow between the hull and the sea bottom, the flows of interest are always considered turbulent due to the large velocities and length scales involved. For example, the least turbulent flow considered will be a

transverse flow underneath a vessel using DP, where the characteristic length is the vessel's beam, assumed for typical service ships to be larger than 15 m. Current velocity is assumed to be 1m/s, and kinematic viscosity of water, ν can be found for simplicity in 28th ITTC 2011.

$$\mathcal{L} \geq 15\text{m} \quad , \quad u \geq 1\text{m/s} \quad \rightarrow \quad \mathfrak{Re} = \frac{u \cdot \mathcal{L}}{\nu} > 10^7 \tag{1.1}$$

Although theoretical velocity distribution of plane and three-dimensional jets exists, these mathematical representations assume calm surroundings. The influence of, for example, a transversal flow crossing the jet is not as easily predicted. According to F. M. White and Majdalani 2006, higher-order CFD solvers are expected to be able to predict the mean velocity in turbulent flows and model the turbulence to a satisfactory degree. As found in Cutler and J. White 2001 there are limitations to the reliability of the results, but it is undoubtedly a valuable tool concerning flow behavior.

An important simplifaction for the CFD analysis is the rigid free surface, modeled as a frictionless rigid wall instead of a free surface. The potential impact of free surface waves is only considered theoretically through experiments and studies performed and published by others, for example, Kozłowska 2019. The significant thrust loss associated with the free surface is ventilation, which can be avoided by demanding certain submergence of the propeller and avoiding operations in overly difficult environmental conditions. Furthermore, according to Erik Rotteveel et al. 2017, the wave resistance is typically increased in shallow water, which is a vital aspect to consider, meaning, although thrust loss might not occur, an increased amount of resistance will undoubtedly require more thrust.

Considering small clearances between the keel and bottom has shown to increase the current loads due to blockage of the flow underneath the hull. In turn, this increases the flow velocities closer to the surface. Hence, increasing resistance and altering the pressure distribution around the hull, as described by the experiments and CFD simulations performed in Koop 2015. However, how this change in pressure and velocity distribution might impact a thruster underneath the hull is unclear.

Accelerated flows tend to cling or be attracted by surrounding surfaces. In the case of a jet flow, this clinging can lead to a displaced momentum, where entrainment of surrounding fluid creates a low-pressure region between the surface and the jet, enabling the flow to curve along the outside of convex geometry. This clinging effect is often called the Coanda effect and is known to cause loss of thrust, as discussed in Vartdal and Garen 2001. Free shear flows can be predicted in CFD, although the interaction with solid surfaces does pose a real challenge.

This report seeks to combine previous knowledge acquired from model and CFD experiments with some CFD experiments conducted by the author to assess the effects that might make a difference in shallow water. The effects mentioned above will be the subjects of interest for this report. Hopefully, combining the knowledge of how each of these acts on the vessel performance will lead to a conclusion towards

how the resulting combined effects might make a difference in practice.

Table 1.1: Different transition Reynolds numbers predicted by an assortment of methods

Method	predicted $\mathfrak{Re}_{x,\ \mathrm{tr}}$
Granville 1953	$2.505 \cdot 10^6$
Michel 1952	$2.525 \cdot 10^6$
Cebeci and Smith 1974	$2.027 \cdot 10^6$
Wazzan et al. 1981	$4.753 \cdot 10^6$

This thesis studies the potential thrust losses that might impact performance when operating in shallow waters. The first topic of interest is a theoretical background for CFD computations, explaining the background for the theory of the methods applied later. Equally important is knowledge about the limitations of CFD. A theoretical discussion of free shear flows is also included in order to be able to discuss the impact of the difference between two and three-dimensional effects. After the purely theoretical part, there is a discussion regarding the different sources of known thrust loss, resulting in a hypothesis describing the possible interactions and how best to test their validity in shallow water. Most of the time invested in this thesis has gone into establishing a working CFD model.

CHAPTER

2

THEORETICAL BACKGROUND

This chapter is included to clarify the background of applied theory later in the thesis. It covers the essential background for CFD, leading up to the specific methods used. The background for the CFD theory is, to a large extent, a rewritten account of the theoretical background covered in Holmedal 2002. The formulation of the equations is kept the same, and the general story interpreting the equations is told the same way. The specific portion rewritten from Holmedal 2002 is section 2.1 until the end of section 2.2.1. The author has tried to keep the general formulation and use of variable names as constant as possible throughout the theoretical background chapter, as well as in later chapters to avoid confusion, as well as conform to the wish of co-supervisor professor Holmedal that the theoretical background should be formulated as done in Holmedal 2002.

Following the background and formulation of the methods used in the CFD experiment, a short discussion of two-dimensional and three-dimensional free shear flows is presented, mainly to serve as background for later arguments regarding the results and validity of eventual results achieved from CFD analysis.

2.1 Reynolds Averaged Navier Stokes

Big whirls have little whirls
that feed on their velocity,
and little whirls have lesser whirls
and so on to viscosity.
–Lewis Fry Richardson

The ocean water, which makes out the fluid flow encountering the hull, is considered a homogeneous, isotropic Newtonian fluid. The fluid is also assumed incompressible with a density ρ and kinematic viscosity ν. Considering conservation of mass and momentum (Newton's second law), the incompressible Navier Stokes equations can be derived. Considering zero external body forces and the absence of temperature fields, using Cartesian coordinates with the Einstein summation convention, $x_i(i = 1, 2, 3) = (x, y, z)$ we can write the Navier Stokes equations as follows

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2.1}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu\frac{\partial^2 u_i}{\partial x_j \partial x_j} \tag{2.2}$$

In equation (2.1) and (2.2), $u_i(i = 1, 2, 3) = (u, v, w)$ signify the velocity components in Cartesian coordinates and p is the pressure.

Since Navier Stokes equations are derived from such fundamental theory, it is generally assumed to encompass every aspect of information about the flow. However, since the computational cost of directly solving them is substantial, simplifications and approximations are commonly used to lessen the effort needed to find a solution.

Due to the diversity in cases of interest, most flows are turbulent. A turbulent flow is characterized by velocity and pressure fluctuations in three dimensions. Reminiscent of a chaotic or random system. A common way to decompose the flow is

$$u(x_i, t) = \overline{u(x_i, t)} + u'(x_i, t) \tag{2.3}$$

$$p(x_i, t) = \overline{p(x_i, t)} + p'(x_i, t) \tag{2.4}$$

Where the $\overline{\text{overbar}}$ signifies the mean, or average value, and the prime$'$ signifies the turbulent, or fluctuating component. Ideally, the averaging should be an ensemble average, but in experiments, the more practical time average is often used once the turbulent flow is independent of time. By inserting equations (2.3) and (2.4) into the Navier Stokes equations (2.1) and (2.2) yields the Reynolds averaged Navier Stokes equations, abbreviated RANS equations

$$\frac{\partial \overline{u_i}}{\partial x_i} = 0 \tag{2.5}$$

$$\frac{\partial \overline{u_i}}{\partial t} + \frac{\partial \overline{u_i}\,\overline{u_j}}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu\frac{\partial^2 \overline{u_i}}{\partial x_j \partial x_j} - \frac{\partial \overline{u_i' u_j'}}{\partial x_j} \tag{2.6}$$

Important to note in equation (2.6) is the last term, which is a single-point correlation that acts as a stress on the fluid and is therefore called the Reynolds stress. The stress is a flow property determined by the fluctuating velocity components, not the fluid. In order to predict this term, modeling has to be applied. There are several ways to model this term, and each particular way introduces a new turbulence model.

2.2 Generalized eddy viscosity

The first method for modeling the Reynolds stress was made by Boussinesq 1877, assuming proportionality between the Reynolds stress and the mean velocity gradients of the flow. The most common method today is the generalized eddy viscosity

$$-\overline{u_i' u_j'} = \nu_T \left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i} \right) - \frac{2}{3} k \delta_{ij} \quad , \quad k = \frac{1}{2} \overline{u_i' u_i'} \tag{2.7}$$

Where δ_{ij} is the Kroenecker delta function, which equals zero for $j \neq i$, and one for $j = i$. k is the turbulent kinetic energy.

The advantage of formulating the equation with the use of the δ_{ij} function is that the normal and transverse stresses are separated, enabling the last term in equation (2.7) to be absorbed into the pressure term when the equation is substituted into equation (2.6). The resulting formulation reduces the problem of turbulence modeling to the specification of an appropriate eddy viscosity ν_T.

The fundamental form of the eddy viscosity used is the one assumed by Prandtl 1925

$$\nu_T = l_m V_m \tag{2.8}$$

l_m is the characteristic length scale, and V_m is the velocity scale of the largest turbulent eddies.

There are several ways to resolve the eddy viscosity; the different methods are classified into three groups. *zero-, one-* and *two*-equation models. The zero-equation model specifies V_m and l_m from experiments or empirical formulas. A one-equation model means a transport equation is applied to V_m, while l_m is specified differently. Lastly, a two-equation model involves transport equations for both factors. Zero-equation models are often used for simplified versions of the Navier Stokes equations. One and two-equation models are more often used with RANS equations.

For the first eddy viscosity distribution Prandtl 1925 used the well-known Prandtl's mixing length hypothesis

$$\nu_T = l_m^2 \left| \frac{\partial \overline{u}}{\partial z} \right| \tag{2.9}$$

The mixing length concept assumes that turbulent eddies interact by collisions, much like molecules collide in a gas, making the mixing length l_m comparable to the mean free path from kinetic gas theory. The more modern view is that turbulent eddies interact more or less continuously. Eddies can also encompass the same length or width as the flow domain leading to the conclusion that the hypothesis is false, as described by Mathieu and Scott 2000. It should be noted that the mixing length models have successfully predicted many flows.

The biggest limitation of zero-equation models is that they do not account for the transport of the turbulent length and velocity scale from equation (2.8), which would be insufficient to describe, for example, boundary layer turbulence produced close to the body and then diffusing outwards from the body. The diffusive and convective transport of the velocity scale is taken into account in one-equation models, but it is not before two-equation models are used that we see both velocity and length scales diffusion and convection taken into account. Both Kolmogorov 1941 and Prandtl 1945 independently suggested to use the turbulence velocity scale $V_m \sim \sqrt{k}$, giving us the Kolmogorov-Prandtl relation

$$\nu_T = C_\mu \sqrt{k} l_m \tag{2.10}$$

Where C_μ is an empirical constant. The relation leads to new transport equations, which can be found by manipulating the Navier Stokes equations, (2.6) to get

$$\frac{\partial k}{\partial t} + \overline{u_j} \frac{\partial k}{\partial x_j} = D_k + P_k - \varepsilon \tag{2.11}$$

where the diffusion D_k, the production P_k and the dissipation ε of turbulent kinetic energy is specified in equations (2.12), (2.13) and (2.14) respectively

$$D_k = -\frac{\partial}{\partial x_j} \left(\overline{u_i' \left(\frac{p'}{\rho} + k \right)} \right) + \nu \frac{\partial}{\partial x_i} \left(\overline{u_j' \left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i} \right)} \right) \tag{2.12}$$

$$P_k = -\overline{u_i' u_j'} \frac{\partial \overline{u_i}}{\partial x_j} \tag{2.13}$$

$$\varepsilon = \nu \overline{\left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i} \right) \left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i} \right)} \tag{2.14}$$

Each of these equations has to be modeled, utilizing the concept of eddy viscosity shown in equation (2.7), as well as the averaged version of Poisson's equation for incompressible fluids in equation (2.5). Resulting in the modeled production term

$$P_k = \nu_T \left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i} \right) \frac{\partial \overline{u_i}}{\partial x_j} \tag{2.15}$$

The diffusion term is treated a little differently, especially for higher Reynolds numbers, where the last term in equation (2.12) is often assumed negligible. The turbulent transport is assumed to be proportional to the gradient of k. Moreover, the proportionality constant is also assumed to be proportional to the eddy viscosity. Resulting in the modeled diffusion term

$$D_k = \frac{\partial}{\partial x_j} \left(\frac{\nu_T}{\sigma_k} \frac{\partial k}{\partial x_j} \right) \tag{2.16}$$

Where σ_k is an empirical constant.

The poem shown at the beginning of this chapter may best exemplify the dissipation term. The dissipation is constantly positive, meaning that turbulent kinetic energy is at all times deteriorating from the largest whirls or eddies down to such small eddies that the viscosity, which is negligible for large eddies, becomes the dominant factor. All turbulent energy is, in the end, dissipated into heat. Using dimensional analysis as discussed in Tennekes et al. 1972, we have $\varepsilon \sim \frac{V_m^3}{l_m}$, using $V_m \sim \sqrt{k}$, the modeled dissipation becomes

$$\varepsilon = C_{di} \frac{k^{\frac{3}{2}}}{l_m} \tag{2.17}$$

Where C_{di} is an empirical constant.

Substituting equation (2.15), (2.16) and (2.17) into equation (2.11), we get the modeled turbulent transport equation

$$\frac{\partial k}{\partial t} + \overline{u_j} \frac{\partial k}{\partial x_j} = \frac{\partial}{\partial x_j} \left(\frac{\nu_T}{\sigma_k} \frac{\partial k}{\partial x_j} \right) + \nu_T \left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i} \right) \frac{\partial \overline{u_i}}{\partial x_j} - C_{di} \frac{k^{\frac{3}{2}}}{l_m} \tag{2.18}$$

A characteristic length l_m must still be specified, depending on the flow. In addition, C_μ and C_{di} need to be determined from experiments. Even though these constants also depend on the flow, for engineering purposes, most reference values are determined from experiments of steady flow over a flat plate, which is a trade-off that enables an easier way to assess the method's robustness.

2.2.1 $k-\varepsilon$ model

By specifying a set characteristic length, as earlier discussed in chapter 2.2, a one-equation method is chosen. Which is considered to be less realistic than a two-equation method. When a two equation method is utilized, a transport equation for $k^a l_m^b$ is used, not a transport equation for l_m by itself, where the constants a and b are decided through dimensional analysis. Using equation (2.17) and equation (2.10) we get another relation for the eddy viscosity

$$\nu_T = C_{di} C_\mu \frac{k^2}{\varepsilon} \equiv c_1 \frac{k^2}{\varepsilon} \tag{2.19}$$

The transport equation for the dissipation ε can, similarly to the turbulent kinetic energy transport equation, be deduced from the RANS equations (2.6), as done in Harlow and Nakayama 1967. However, it is subject to heavy modeling, as discussed in Ueda and Hinze 1975. The most common form of the high Reynolds number k-ε model is

$$\frac{\partial \varepsilon}{\partial x_j} + \overline{u_j} \frac{\partial \varepsilon}{\partial x_j} = \frac{\partial}{\partial x_j} \left(\frac{\nu_T}{\sigma_\varepsilon} \frac{\partial \varepsilon}{\partial x_j} \right) + c_{1\varepsilon} \frac{\varepsilon}{k} \nu_T \left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i} \right) \frac{\partial \overline{u_i}}{\partial x_j} - c_{2\varepsilon} \frac{\varepsilon}{k} \tag{2.20}$$

Both equation (2.18) and equation (2.20) must be solved simultaneously with the RANS equations. Important to note is the five constants that need to be determined from experiments.

$$\left\{ \begin{array}{ccccc} c_1 & \sigma_k & \sigma_\varepsilon & c_{1\varepsilon} & c_{2\varepsilon} \end{array} \right\}$$

The flow will still influence the appropriate initial values for k and ε. Near solid walls, wall functions are used to approximate the boundary layer with a logarithmic velocity profile (in the direction that's normal to the wall), as well as demand equilibrium between production and dissipation of turbulent kinetic energy.

2.2.2 $k-\omega$ model

According to F. M. White and Majdalani 2006, The k-ε model predicts the mean velocity, growth rate, and shear stress moderately well and are only fair to poor for the turbulence components. In order to avoid those shortcomings Wilcox 1988 created another model, called the k-ω model. It is a two-equation model, just like the k-ε model, but instead of using the transport equation for ε, the specific dissipation ω is given the transport equation

$$\frac{\partial \omega}{\partial t} + \overline{u_j} \frac{\partial \omega}{\partial x_j} = \alpha \frac{\omega}{k} \tau_{ij} \frac{\partial \overline{u_i}}{\partial x_j} - \beta \omega^2 + \frac{\partial}{\partial x_j} \left[(\nu + \sigma_\omega \nu_T) \frac{\partial \omega}{\partial x_j} \right] \tag{2.21}$$

$$\tau_{ij} = \nu_T \left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i} - \frac{2}{3} \frac{\partial \overline{u_k}}{\partial x_k} \delta_{ij} \right) - \frac{2}{3} k \delta_{ij} \tag{2.22}$$

Involving a new set of constants that needs to be validated through experiments

$$\left\{ \begin{array}{ccccccc} \alpha & \beta & \beta^* & \sigma_\omega & \sigma_\omega^* & \varepsilon \end{array} \right\}$$

More of a physical interpretation of the specific turbulent dissipation ω might be that it is the ratio of the turbulent dissipation rate ε to the turbulent mixing energy. In other words, ω is the rate of turbulence dissipation per unit of energy.

2.2.3 SST $k - \omega$ model

The shear stress transport (SST) formulation, proposed by Florian R. Menter 1993 and improved for engineering purposes by Florian R Menter 1994, attempts to combine the best of two worlds regarding the weaknesses mentioned at the beginning of section 2.2.2. First, using a $k - \omega$ formulation in the inner parts of the boundary layer could enable a higher accuracy down to the wall through the viscous sub-layer, avoiding extra damping functions. The SST formulation then switches to a $k - \varepsilon$ behavior in the free stream region, which is advantageous since the $k - \omega$ model is susceptible to the inlet free-stream turbulence properties.

Following the reasoning from Florian R Menter 1994, the $k - \omega$ model does not involve damping functions and allows simple Dirichlet boundary conditions to be specified. Due to its simplicity, the $k-\omega$ model is superior to other models, especially concerning numerical stability. Furthermore, it is as accurate as any other model in predicting the mean flow profiles. However, the $k - \omega$ model cannot predict the asymptotic behavior of the turbulence as it approaches the wall, and it does not accurately represent the k and ε distribution.

Conclusively, Florian R Menter 1994 argues that in the sublayer and the logarithmic part of the boundary layer, the $k - \omega$ model is superior to the $k - \varepsilon$ model in equilibrium adverse pressure gradient flows and incompressible flows. Therefore, in the wake region of the boundary layer, the $k-\omega$ model is abandoned in favor of the $k - \varepsilon$ model.

In order to achieve the desired features, the $k-\varepsilon$ model is transformed into a $k-\omega$ formulation and then multiplied with a blending function, equation (2.23), and added to the original $k-\omega$ model multiplied with F_1. The blending function is then created so that the value of F_1 equals 1 in the sublayer and parts of the logarithmic region of the boundary layer. Followed by a gradual decrease to 0 towards the wake region. Lastly, the definition of the eddy viscosity is modified to account for the transport of the principal turbulent shear stress

$$\phi = \underbrace{F_1\phi_1}_{k-\omega} + \underbrace{(1-F_1)\,\phi_2}_{k-\varepsilon} \tag{2.23}$$

ϕ represents the constants from each set combined, ϕ_1 constants of the incorporated $k-\omega$ method, and ϕ_2 the constants of the manipulated $k-\varepsilon$ method.

The transport equation for the turbulent kinetic energy (2.18) and specific dissipation are therefore modeled a little differently from both 2.2.1 and 2.2.2

$$\frac{\partial k}{\partial t} + \overline{u_j}\frac{\partial k}{\partial x_j} = \tau_{ij}\frac{\partial \overline{u_i}}{\partial x_j} - \beta^*\omega k + \frac{\partial}{\partial x_j}\left((\nu + \sigma_k\nu_T)\frac{\partial k}{\partial x_j}\right) \tag{2.24}$$

$$\frac{\partial \omega}{\partial t} + \overline{u_j}\frac{\partial \omega}{\partial x_j} = \frac{\gamma}{\rho\nu_T}\tau_{ij}\frac{\partial \overline{u_i}}{\partial x_j} - \beta\omega^2 + \frac{\partial}{\partial x_j}\left[(\nu + \sigma_\omega\nu_T)\frac{\partial \omega}{\partial x_j}\right] + 2\,(1-F_1)\,\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} \tag{2.25}$$

Where

$$\tau_{ij} = \nu_T\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i} - \frac{2}{3}\frac{\partial \overline{u_k}}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}k\delta_{ij} \tag{2.26}$$

The definition of the eddy viscosity is defined as

$$\nu_T = \frac{a_1 k}{\max\begin{pmatrix} a_1\omega & \omega F_2 \end{pmatrix}} \quad , \quad F_2 = \tanh\left(\left(\max\begin{pmatrix} \frac{2\sqrt{k}}{0.09\omega y} & \frac{500\nu}{y^2\omega} \end{pmatrix}\right)^2\right) \tag{2.27}$$

where Florian R Menter 1994 has suggested an optimized value for each constant, see tables 2.1 and 2.2. These values are found mainly by comparing direct numerical simulation (DNS) results to that of the model, but at the same time also state that adding small changes to the constants can lead to significant improvement or deterioration of the model predictions. None of the available empirical tools, including DNS, can provide constants to that degree of accuracy. The only way to establish the validity is to carefully test the resulting model against several challenging and well-documented research flows. Even this is hard, as it is unclear if improvements in one type of flow, for example, boundary layer flow, might lead to deterioration of results for other types of flows, like free shear flow.

Table 2.1: Constants of incorporated $k-\omega$ model

Constant	σ_{k1}	$\sigma_{\omega 1}$	β_1	a_1	β^*	κ	γ_1
Value	0.85	0.5	0.075	0.32	0.09	0.41	$\frac{\beta_1}{\beta^*} - \sigma_{\omega 1}\frac{\kappa^2}{\sqrt{\beta^*}}$

Table 2.2: Constants of incorporated $k-\varepsilon$ model

Constant	σ_{k2}	$\sigma_{\omega 2}$	β_2	β^*	κ	γ_2
Value	1.0	0.856	0.0828	0.09	0.41	$\frac{\beta_2}{\beta^*} - \sigma_{\omega 2}\frac{\kappa^2}{\sqrt{\beta^*}}$

As a more pointed remark towards the work described in chapter 4, according to Florian R Menter 1992, it has been shown that the eddy viscosity in boundary and free shear layers can change by more than 100% by simply reducing the value of ω that is applied at the inlet, while also showing that the $k-\varepsilon$ model does not suffer from the same deficiency. According to Florian R Menter 1994 there does not seem to be a model that accurately predicts the free shear flows, so the $k-\varepsilon$ model seems to be a fair compromise.

2.3 Jet flow

For this part of the theory chapter, the RANS equations (2.6) are no longer considered. Therefore the notation will change a little, even though the velocities discussed are arguably the same. However, the velocities used from here on out will be the instantaneous velocity, i.e., following the same form as the regular NS equations (2.2).

2.3.1 Plane jet

The following theoretical considerations are mainly collected from F. M. White and Majdalani 2006. Most steps between assumptions and results will be omitted to avoid a lengthy and elaborate derivation. In figure 2.1 there is a depiction of how the initial formation of a 2D jet is formed. In theoretical considerations, it is the self-preserving region that is considered.

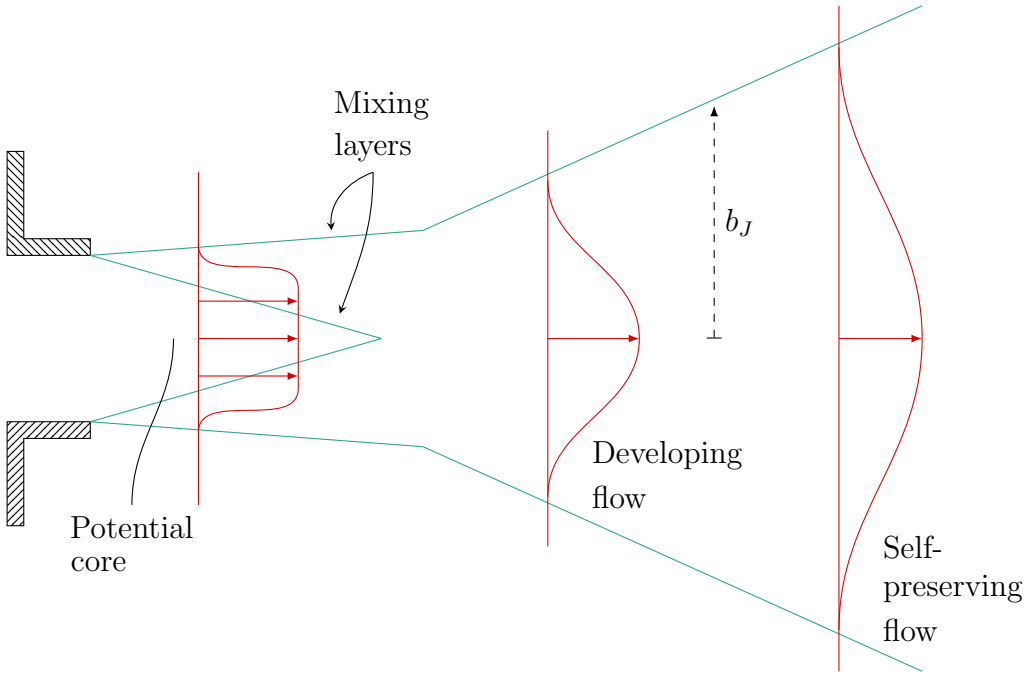


Figure 2.1: Early development of a turbulent jet, copy of a figure found in F. M. White and Majdalani 2006

The potential core represents an area where the jet issues at a nearly flat, fully developed, turbulent velocity. As the jet emerges, mixing layers of both still ambient flow and the nearly inviscid potential core form at the edge of the source. These mixing layers increase in size until approximately one diameter length downstream is reached. Further downstream, the potential core has vanished, and the velocity profile assumes the "typical" jet velocity profile. Eventually, at around 20 diameters downstream, the velocity profile achieves and maintains a self-preserving shape. Which can be formulated with a flow-dependent function f, as in equation (2.28). Important to note here is that the velocity profiles at two different locations have the same momentum but not the same mass flow because of entrainment of the surrounding fluid.

$$\frac{u}{U_{\max}} \simeq f\left(\frac{y}{b_J}\right) \tag{2.28}$$

The central assumption for the self-similar region is that the centerline velocity and jet width should only depend upon jet momentum, density, and distance. *Not* upon molecular viscosity due to the absence of walls. Since there is no pressure gradient, the jet momentum $\mathcal{Q}_J$ must remain constant at each cross-section. By dimensional analysis, the width can only have a linear growth.

$$\mathcal{Q}_J = \int_{-\infty}^{\infty} \rho \overline{u}^2 dA = C \quad \vee = C\rho b_J u_{\max}^2 \tag{2.29}$$

C represents an arbitrary constant.

The maximum flow velocity and jet half-width are functions of the distance from the source in the flow direction, momentum, and density. By dimensional analysis, the plane jet can only experience linear growth, that is, regardless of the Reynolds number

$$b_J = Cx \quad , \quad u_{\max} = C \left(\frac{J}{\rho} \right)^{\frac{1}{2}} x^{-\frac{1}{2}} \tag{2.30}$$

There are different ways to derive the jet half-width and deterioration of velocity. In F. M. White and Majdalani 2006 the forms of the maximum flow velocity and b_J are used to solve turbulent boundary layer continuity and momentum relations, i.e. equation (2.5) and (2.6) for two dimensions. The resulting half-angle is calculated to 13°.

$$\frac{b_J}{x} = \tan\left(13^\circ\right) \tag{2.31}$$

2.3.2 Circular jet

The analysis for the circular jet is quite similar to the plane jet. However, the deterioration of the velocity and the boundary condition for axisymmetric jets look a little different. These results are also found in F. M. White and Majdalani 2006. The velocity profile is again formulated by using the maximum flow velocity

$$\mathcal{Q}_J = \int_{-\infty}^{\infty} \rho \overline{u}^2 dA = C \quad \vee = C\rho b_J^2 u_{\max}^2 \tag{2.32}$$

$$b_J = Cx \quad , \quad u_{\max} = C \left(\frac{J}{\rho} \right)^{\frac{1}{2}} x^{-1} \tag{2.33}$$

The resulting form is similar to the laminar jet, although some discrepancies in the outer regions exist, quite possibly due to the intermittency of turbulence. Because of the long time it takes for the turbulence components to develop, the jet cannot reach the self-preserving region before $50 \geq \frac{x}{D} \geq 70$. Along the centerline, the maximum velocity decreases as if the flow began from a virtual origin seven diameters ahead of the source. That would result in a nozzle exit angle of 4°. The basis for these results is a comparison of the results of Wygnanski and Fiedler 1969 with calculations of F. M. White and Majdalani 2006.

CHAPTER

3

KNOWN SOURCES OF THRUST LOSS

Dynamic positioning has been increasing in importance for different types of marine operations. Different kinds of propulsors are used, and the characteristics of the propulsors are usually known only for the open water case. An important aspect to consider, especially regarding dynamic positioning, are environmental and interaction effects, which might affect the characteristics known from the open water case.

3.1 Ventilation

As described in Kozlowska and Steen 2017, a ship propeller, performing under a significant loading condition, might develop unsteady line vortex cavitation between the propeller tip and the hull, known to cause significant vibrations and noise near the stern of the ship. In addition, a vortex might form between the propeller and the free surface if the propeller is situated with low submergence. Through such a vortex, air can be drawn down into the propeller, resulting in ventilation of the propeller. Typically this scenario occurs for propellers subjected to high loading conditions, with limited submergence, and subjected to large relative motions due to heavy seas.

Propeller ventilation depends primarily on submergence, propeller loading, and forward speed, as concluded in Kozłowska 2019. A closely related cause of severe thrust loss is the emergence of the propeller.

The phenomena can be categorized through three zones, following the description by Koushan 2004, sub-critical, critical, and super-critical. The dependence is illustrated by the thrust and torque over the advance coefficient, J in figure 3.1. The

advance coefficient is defined in equation (3.1). Thrust reduction becomes a time-dependent function of relative motion, wave direction, and propeller loading. The figure shows a hysteresis for the critical zone, depending on which direction the propeller is evolving. If the propeller moves from sub-critical to critical, the ventilation phenomenon is delayed while the thrust is increased. In the other case, the propeller evolves from fully developed ventilation in the super-critical zone and reasserts into the critical zone. The critical advance coefficient becomes higher, resulting in lower thrust.

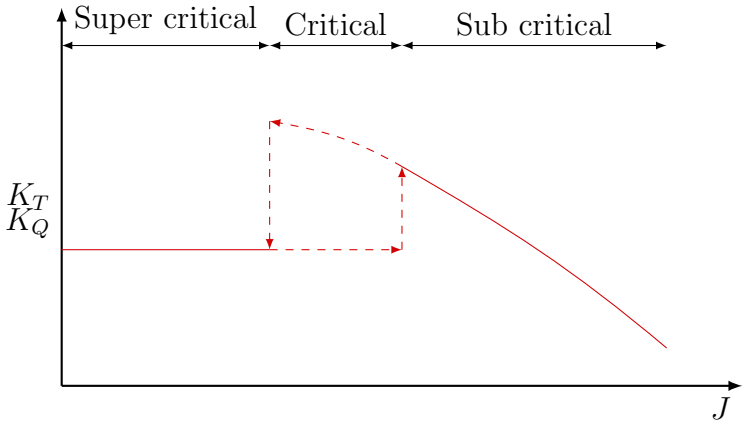


Figure 3.1: Schematic presentation of different ventilation zones, Koushan 2004

$$J = \frac{u}{nD} \tag{3.1}$$

Where n is RPS and d_p is the propeller diameter.

From the extensive tests, considering different immersion ratios and rate of revolutions, performed by Shiba 1953. It was found that the effect of the Weber number, equation 3.2, becomes insignificant for a Weber number over 180

$$\mathcal{W}e = nD_p\sqrt{\frac{\rho}{S}D_p} \tag{3.2}$$

S is the fluid surface tension.

Seeing as most propellers operate at a Weber number much larger than 180, this might not be as useful. For example, a propeller with a diameter of 2.5 m running at 120 RPM will have a Weber number of around 924. Therefore the Weber number effect might be ignored for practical purposes regarding full-scale vessels considered in this project.

Considering some of the experiments presented in Kozlowska and Steen 2017, it was found that ventilation does not occur for bollard condition $J = 0$, when the propeller submergence $\left(\frac{h_p}{R_p}\right)$ is larger than 3.4. Where h_p is the distance from the propeller

axis to the surface, and R_p is the propeller radius. Plotted for different diameters in figure 3.2.

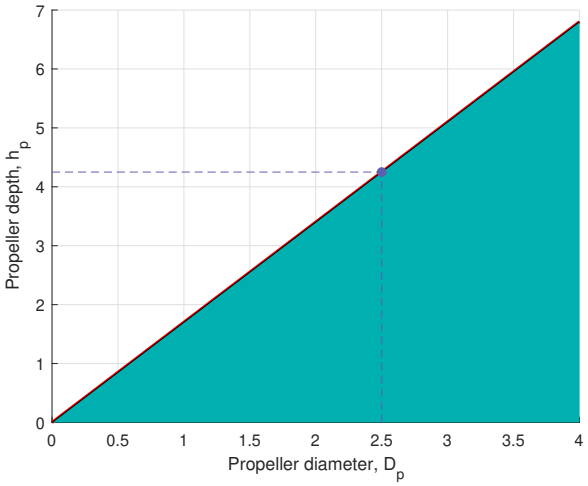


Figure 3.2: The limit for ventilation at bollard condition, for $n = 16$Hz, from Kozlowska and Steen 2017

3.2 Thruster to thruster interaction

In Lehn 1985, it is argued that for a propeller operating behind another propeller, a considerable amount of thrust is forfeit if the thruster in front directs the propeller race towards the hindmost propeller. It also causes a loss of torque, which is a source of problems with regards to systems that control the RPM. The worst case scenario is the so-called tandem configuration, i.e. the propellers are mounted in front of each other. This implies a larger advance coefficient, and as can be seen from figure 3.1 a loss of thrust can indeed be expected. Intuitively an increase in incoming flow to the propeller, wrt. J, equation (3.1), can be compensated for by a higher number of revolutions per second. An obvious problem arises of course for very high velocity in the incoming flow.

From experimental results, achieved by Lehn 1985, it was found that at a distance of $5 \cdot d_p$ from the foremost to hindmost propeller in tandem configuration, an approximate 50% of the thrust is lost in bollard condition. It was also found that at a distance of $6 \cdot d_p$, there was no loss of thrust, if the hindmost propeller was directed at a 15° angle to the incoming propeller race. At $3 \cdot d_p$ distance, the hindmost propeller had to be directed at an angle between $25° - 30°$ in order to avoid thrust loss.

In Koushan 2004 it is recommended to make serious evaluation of propulsion units placement relative to each other and operational limits wrt. the direction of the propeller races, in order to avoid the interaction between a propeller race and another propeller. The penalty is not just with regards to the available force, but the overall efficiency of the propeller as well. In the end, this is an important problem to consider at the time of design and placement of the propellers, not after the vessel

is fully seaborne. It is also an independent effect from the $\frac{W_D}{T}$ ratio.

3.3 The influence of current

According to Lehn 1985 the largest contribution from interaction losses occur for thruster to thruster interactions, or when a propeller race impedes onto parts of the hull. The propeller race can in this regard be compared to a very speedy current or an obstacle that that the jet interacts with. Since currents rarely, if at all achieves speeds comparable to that of a propeller race, the increase in advance coefficient can most likely be compensated for by an increase in RPM. What remains with regards to current interaction is a potential deflection of the propeller race, which therefore is important to consider.

As argued by Lehn 1985, the propeller race can be diverted by $2° - 4°$ in vertical direction when encountering a pontoon. Depending on the velocity and direction of the current, a similar result could be achieved with the right parameters. However, the influence is unique for every heading and velocity.

Another angle to consider is the bottom boundary layer at sea, which is very difficult to measure, not at all a controlled environment. From Johns 1983,

Nevertheless, as far as can be judged, the boundary layer beneath tidal and other currents on the continental shelf appears to exhibit many similarities with boundary layers in the atmosphere and the laboratory. However, it would be presumptuous to suppose that every facet of the marine boundary layer is an identical analogue to those found in better known boundary layers.

An important difference from typical lab conditions is the so called bursting phenomenon, which describes an event where series of turbulent fluctuations contain more energy than the average flow turbulence. Despite considerable research into the bursting phenomenon, a practical and usable method that incorporates the processes involved into a theoretical framework for the bottom boundary layer does not exist, as stated in Johns 1983. For the present work, this implies a certain uncertainty for the exact $\frac{W_D}{T}$ ratio involving interactions between the bottom boundary layer, propeller jet and hull.

3.4 Increased resistance in shallow waters

Similarily explained as in Amdahl et al. 2015, where an analogy for a vessel moving in deep water is used to explain how the hull is able to produce the waves that result in wave resistance. The analogy is a body floating on the surface, and instead of moving with a forward velocity, a current or flow is encountering and moving past the hull. The displacement of the hull alter the movement of the water particles, displacing their movement to the side and below the hull.

Near the bow the flow is retarded due to the encounter with the hull, although this

intuitively indicates an increase in pressure, the pressure at the surface (atmospheric pressure) remains constant, according to Bernoullis[1] equation (3.3), the only possible compensation for a lower velocity is an increase in elevation z. Indicating a wave crest near the bow. The volume displacement influences the velocity along the hull, forcing the flow outwards, effectively increasing the velocity which then indicates a decrease in elevation or a wave trough. In reality waves are formed all along the hull, but they become "visible" only where there are changes in the curvature of the hull.

$$\frac{\rho}{2}u^2 + \rho g z + p = \text{constant} \tag{3.3}$$

Where z is the vertical distance from the calm water surface.

According to Rotteveel and Hekkenberg 2015 the increase of ship resistance in shallow water can be divided into two parts. The increase in wave resistance and additional resistance related to blockage of the flow underneath the hull. In Koop 2015 the blockage effect was investigated by the use of both CFD and experiments. For shallow water, and incoming current from a heading between $40° - 130°$ the resistance was found to increase by $30\% - 50\%$. However, these experiments utilized basin side walls, meaning the fluid had very little available space when encountering the hull. For deeper water, and basin side walls, the increase was found to be less than 5%.

An interesting find, with regards to the blockage effect from Koop 2015, was the difference between towing and current experiments. The boundary layer present for the current case seems to increase the blockage of the flow, resulting in 10% increase in resistance. Since all of these experiments were conducted with basin side walls, the implications for shallow water without the presence of side walls become much less specific. Regardless, an argument can be made for the importance of increased resistance, not only decrease in thrust.

3.5 Interactions related to the Coanda effect

Due to entrainment of the surrounding fluid in a free shear flow, as discussed in section 2.3.1 the velocity of the surrounding fluid around the jet is accelerated. When a surface of some kind happens to be in the vicinity of a free shear flow, or jet, the larger velocities decrease the pressure between the surface and the jet, pulling on the flow. This phenomenon is often described as a tendency to cling to surrounding surfaces, and called the Coanda effect. The consequence of clinging to e.g. the surface of the hull could be deflection of the jet, and in some cases include impingement on parts of the hull.

The coanda effect is just as particularly flow dependent as currents are in 3.3. And the effects therefore vary in character. If the flow clings to the hull in a way that

[1]For incompressible flow along a streamline

results in impingement on the hull, the thrust loss could potentially be dramatic. However, that is very avoidable through design considerations. From Lehn 1985 it is found that the simple redirection of mass flow, caused by propeller jet clinging to the hull can cause around 5% thrust loss.

CHAPTER

4

COMPUTATIONAL FLUID DYNAMICS EXPERIMENT

At this stage in the project, a hypothesis has had the time to develop and be formulated, motivating a strategy for testing, ideally generating results that either support or oppose the statement: *thruster, hull, and bottom interactions in shallow water causes thrust loss.* The primary sources of thrust loss are ventilation, thruster-to-thruster, thruster-to-current, and thruster-to-hull interactions. The thruster-to-hull interactions involve various types, like additional frictional loss, impingement on the hull surface, and Coanda effect losses.

Firstly the issue of ventilation is detrimental but assumed not to be typical for shallow waters. As shortly discussed in Erik Rotteveel et al. 2017 regarding the design of a ship stern, measures are taken to prevent ventilation at the small draft, which occurs in case the ship is empty, partially loaded, or if the ship cannot be fully loaded due to limited water depth. For service vessels, which plan to operate at a certain depth, a minimum draft should be an acceptable solution to avoid the ventilation unless rough seas intervene. At which point service operations should be postponed. Therefore ventilation is not considered part of the scope of this thesis.

Thruster-to-thruster interactions can be pretty severe. As discussed in 3.2, during critical conditions, thruster-to-thruster interaction could be catastrophic but is avoidable through automated systems and clever design. These effects are also considered not particularly typical for shallow waters since they can occur anywhere.

However, the Coanda effect needs surfaces for the interaction to occur. Since shallow waters introduce a new surface to the flow domain, the implication for the Coanda effect is an exciting topic to explore.

The computational fluid dynamics (CFD) experiment attempts to simulate the jet's

behavior ahead and behind the thruster. In order to derive a relatively generic model, both for the thruster and the hull, actual ship dimensions have been used to derive a scaled-down generic version with fitting dimensions. The ship chosen is shown in figure A.1, and the propeller used as a model is seen in table A.1. The main particulars of the hull can be seen in table 4.1, while data regarding the domain surrounding the hull is found in table 4.2.

Table 4.1: Example ship main dimensions

Length over all	L_{OA}	82.0	m
Length between perpendiculars	L_{PP}	72.1	m
Breadth	B	18.0	m
Draft	T	5.0	m
Propeller diameter	d_p	2.5	m

Table 4.2: Flow domain characteristics

Transverse current velocity	U_∞	1.0	m/s
Turbulence intensity	I	2	%
Scale ratio	C	1 : 20	[−]

4.1 Experimental setup for 2D simulation

A discretization scheme is put in place, first involving the domain's meshing. Conceptually, see figure 4.1, the domain will consist of a generic hull cross-section, with a propeller disk situated right underneath with a forced amount of volume flow entering and exiting. In 2D, the disk will essentially be a small box. A current will be forced to enter through an inlet, and the same amount of fluid will have to exit at an outlet. The top or surface of the domain would ideally be a free surface but will instead be modeled as a slip-wall boundary for the flow since the goal will be to study the interactions underneath the hull. The hull and bottom will be modeled as walls, generating a boundary layer demonstrating how the flow interacts with the two surfaces at different levels of depth.

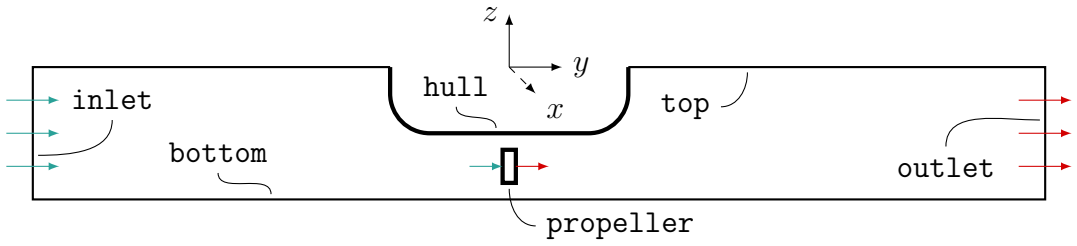


Figure 4.1: Conceptual drawing of the domain for numerical experiment

The hull and bottom boundary will be modeled as a solid wall. The specific functions and restrictions used are discussed in the following section about boundary

conditions; 4.3. The thing that sets the characterization of a wall in CFD apart is the so-called wall functions. They represent the means to approximate the flow in the viscous sub-layer as described near the end of 2.2. From Murad 2022, the wall functions all rely on the *universal law of the wall*, first formulated by Von Kármán 1931. Which in essence states that sufficiently close to a wall, nearly all turbulent flows have similar velocity profiles. The most common parameter to define the valid region for a wall function is the y-plus value, y^+; see equation (4.1). The equation is comparable in form to a $\mathfrak{Re}$ number for the boundary layer itself. Both Professor Lars Erik Holmedal and Murad 2022 suggests a $30 > y^+ > 300$.

$$y^+ = \frac{yu_\tau}{\nu}, \qquad u_\tau = \sqrt{\frac{\tau_w}{\rho}}, \qquad \tau_w = \rho\nu \left(\frac{du}{dy}\right)_{y=0} \tag{4.1}$$

In this particular instance, y is the distance away from the wall, not as defined in figure 4.1. u_τ is the frictional velocity, and τ_w is the wall shear stress.

The big drawback with wall functions is the loss of flexibility. The wall functions themselves are optimized from experiments based on flat plates. For similar reasons, as mentioned in the discussion below, equation (2.18) simplifies the method's general use when standardized constants are used. Considering figure 4.2, the wall functions, when applied correctly, can provide excellent results. Unfortunately, the velocity profile looks quite different when, for example, an adverse pressure gradient is acting on the flow or if the boundary layer contains backflow for some other reason.

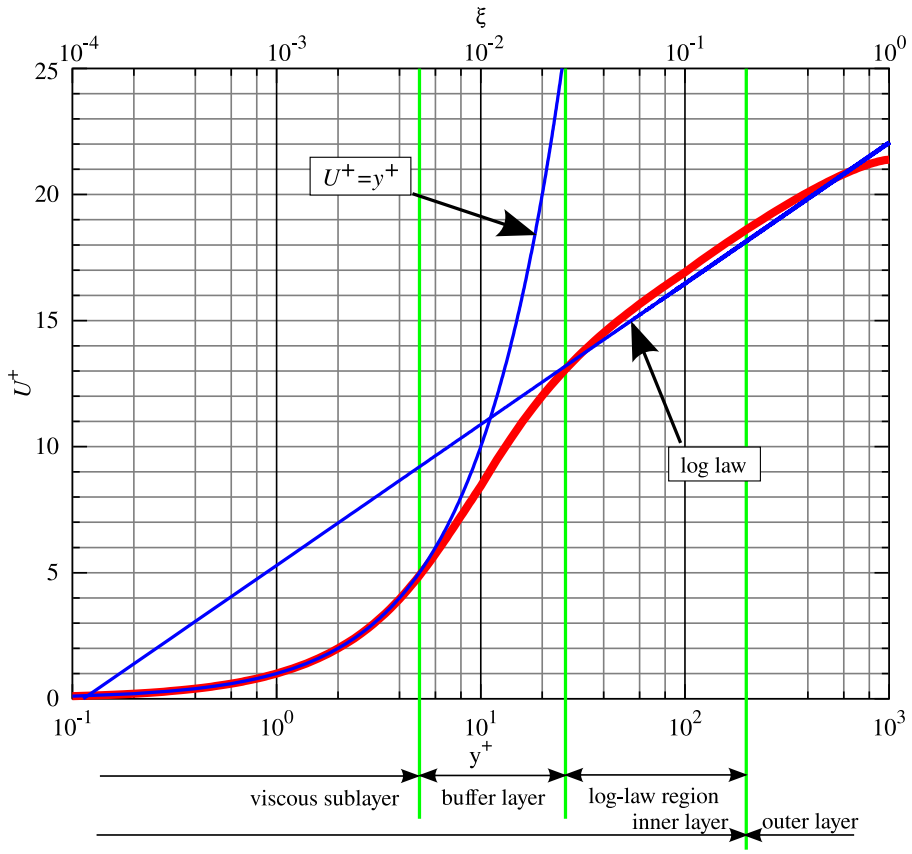


Figure 4.2: A conceptual plot of the velocity profile along a wall, along with the plotted wall functions when applied correctly; created by user Aokomoriuta 2022

The y^+ value can only be obtained upon completion of the analysis and can therefore often be a source of lost computing time, as discussed later in 6.

The inlet, outlet, and propeller are modeled as sinks and sources. To read about the specific functions used on the boundaries, a more detailed discussion is found in 4.3. An essential characteristic of a control volume in a CFD analysis is that the flow obeys the conservation laws; in essence, the same amount of flow that enters the domain must also exit. Regarding the flow parameters, it is not common practice to alter the Reynolds number by changing the incoming flow velocity; instead, the more common way is to change the kinematic viscosity of the flow. This way, the incoming flow velocity can remain 1m/s for all desired Reynolds numbers. This way, the desired Reynolds number is found from equation (1.1), and the kinematic viscosity used in the analyses become

$$\nu_{\mathrm{CFD}} = \frac{u \cdot B \cdot \texttt{scaleRatio}}{\mathfrak{Re}} = 6.802 \cdot 10^{-8} \mathrm{m}^2/\mathrm{s} \tag{4.2}$$

ν_{CFD} is the applied kinematic viscosity used in the analyses, and B is the beam or breadth of the hull.

The total amount of fluid can also be necessary to consider, in this case, essential for the propeller disk. The propeller is modeled after a real propeller as a control volume.

The chosen model can be found marked in red in table A.1. As an appropriate scenario, one thruster performs at 80% bollard condition, BP. Meaning the mass flow through the propeller in full scale is

$$\dot{m}_S = \frac{\text{BP}}{2 \cdot n_{\text{thrusters}}} \cdot 0.8 \tag{4.3}$$

Where $\dot{m}_S$ is the mass flow through the propeller in full scale, and n_{thruster} is the number of thrusters.

In order to find an approximate propeller flow velocity, the mass flow, equation (4.3), is divided by the area of the propeller disk and the density

$$\overline{u_S} = \frac{\dot{m}_S}{\pi \dfrac{d^2}{4} \cdot \rho} \tag{4.4}$$

d is the diameter of the propeller chosen as a model in full scale

The Froude number, equation (4.6) is a dimensionless relation that can be used to relate similarity for all inertial forces at different scales. The process of adapting the velocities is called Froude scaling, and in equation (4.5) the velocity is adapted to the model size.

$$\overline{u_M} = \overline{u_S} \cdot \sqrt{\texttt{scaleRatio}} \tag{4.5}$$

$$\mathcal{F}n = \frac{u}{\sqrt{g\mathcal{L}}} \tag{4.6}$$

The volume and mass flow rate becomes

$$\overline{u_M} \cdot \pi \frac{(d \cdot \texttt{scaleRatio})^2}{4} \begin{cases} \cdot 1 & = \dot{V}_M \\ \cdot \rho & = \dot{m}_M \end{cases} \tag{4.7}$$

It does not matter which, but one of these values is then used as a boundary condition for the control volume used to simulate the propeller, where an equal amount of flow volume has to enter and leave the control volume per unit of time.

4.2 Numerical solver

The program used to resolve the flow system is OpenFOAM, which is free and open-source software released under the *GNU General Public License*. It is a `C++` toolbox created for developing customized numerical solvers and pre-/post-processing utilities to solve continuum mechanics problems. In order to visualize the results, a

program called ParaView has been used, which is an open-source multiple-platform application for interactive, scientific visualization.

As a solver, the initial choice was `PimpleFOAM`, a transient solver, meaning it takes time dependence into account and solves the equations for each time step. Postdoctoral Fellow Jianxun Zhu recommended the solver due to uncertainty regarding vortex shedding. However, large velocities are present in the analysis, which might hinder the development of vortex shedding. Therefore, a transient solver is picked to check if the analysis produces any vortex shedding, which calculates all coefficients for each volume for every timestep. These solvers require more time to compute but provide a more realistic simulation if the flow changes over time, for example, experience cyclic changes like that of vortex shedding.

Another solver is chosen if no vortex shedding is apparent from these simulations. Namely `SimpleFOAM`, which is a steady-state solver. It calculates the steady-state solution for each timestep, which results in much faster computations.

`SimpleFOAM` uses a so-called segregated solution strategy, meaning the velocity field, pressure field, and the variables characterizing the turbulence are solved sequentially. The solution of the preceding equations is then inserted into the following equation. The algorithm starts by approximating a velocity field from the momentum equation. The velocity field is likely not divergence-free, meaning it does not satisfy the continuity equation. Therefore the momentum and continuity equations are used to construct an equation for the pressure. Then it generates a pressure field, which, if inserted into the momentum equation, delivers a divergence-free velocity field. After correcting the velocity field, the equations for turbulence are solved. The above iterative solution procedure is repeated until convergence. [1]

As a general method, the applied model is described in 2.2.3. The SST $k-\omega$ model has been chosen first and foremost due to recommendations from co-supervisor Professor Holmedal, but also due to online statements on forums where people with more experience than the author write down their opinion of the perceived performance of each model.

Authors who use the SST $k-\omega$ model often merit it for its good behavior in adverse pressure gradients and separating flow[2]

The method's appeal is mainly based on the combination of $k-\varepsilon$ and $k-\omega$ models. By using the strengths of each model for different parts of the domain, a better result is most likely achieved than using a full-fledged model of either variant. The result depends on the flow in question, and even though $k-\varepsilon$ supposedly outperforms the $k-\omega$ model for free shear flows, the hull is a blunt body that the flow has to encounter, where the $k-\omega$ model outperforms the $k-\varepsilon$ model. Therefore the choice seems clear-cut, given the minimal presentation of methods described in this thesis. Other methods do exist, although there are certain drawbacks to all of them. For example, the ideal solution would be to use direct numerical simulation, DNS, where the governing equations, i.e., equation (2.2), are solved numerically, without turbulence modeling; this implies that the whole range of the energy cascade, i.e.,

[1] Found on OpenFOAMWiki
[2] Found on CFD wiki

all of the spatial and temporal scales, needs to be resolved. In other words, by utilizing discretization only, the mesh must contain small enough volumes and be resolved over small enough time steps to resolve the behavior of the smallest eddies where dissipation from kinetic energy to heat occurs. The method also requires the computational power to cover a large enough grid to encompass the largest eddies, where most of the turbulent kinetic energy exists. This method is very demanding computationally and, essentially for the scale within the scope of this project, impossible to compute due to the lack of computational resources.

4.3 Boundary conditions

The boundary conditions are given for all necessary components of the numerical solver, meaning k, ν_T, ω, p and $\overline{u}$. The conditions have to be imposed on every single surface bordering the domain. For a general overview of the conditions imposed, see tables 4.3 and 4.4.

The inlet is given a simple and constant entry velocity, uniform across the depth. The inlet is also given the property of introducing turbulence to the flow entering the domain. From Tian et al. 2013 expressions for the two initial conditions that have to be specified can be found in equation (4.8), with a chosen turbulence intensity I. The turbulence intensity used can be found in table 4.2.

$$k_{\text{inlet}} = 1.5\left(u_\infty I\right)^2 \quad , \quad \omega_{\text{inlet}} = \frac{\sqrt{k_{\text{inlet}}}}{C_\mu^{0.25} \cdot 0.07\mathcal{L}} \tag{4.8}$$

As an initial condition, the values extracted from equation (4.8) are also applied to the internal field, which in OpenFOAM indicates the entire volume inside the domain that is not specified as anything but a patch of volume, i.e. all volume except the propeller and hull.

For the top, or surface boundary, which in this project is kept static needs to be modeled as a barrier, but without the boundary layer near the surface. Therefore the top boundary is given a slip condition, meaning it does not affect the velocities close to the surface.

The bottom and hull boundaries are modeled identically. As described in 2.2.3, the model is able to simulate the flow all the way down to the boundary, the purely modeled part, based on experiments are the wall-functions used to determine the flow dependent coefficients, like k, ν_T and ω.

Table 4.3: Boundary types

Boundary	`type`
inlet	`patch`
top	`patch`
hull	`wall`
outlet	`patch`
bottom	`wall`
propInlet	`patch`
propOutlet	`patch`
propTopAndBottom	`patch`
frontAndBack	`empty`

- The `patch` class is a basic condition that contains no geometric or topological information about the mesh.
- The `wall` class defines the border similarily as a patch, but defines a distance from the wall of the cell centers next to the wall, so that the cells are stored as part of the patch, not the wall. Also needed for the use of wall functions in turbulence modeling.
- The `empty` class defines which directions where no flow occurs for 2D or 1D flow. OpenFOAM always generates geometries in 3 dimensions, but it can be used to solve in 2 (or 1) dimensions by specifying a special empty condition on each patch whose plane is normal to the 3rd (and 2nd) dimension for which no solution is required.

Table 4.4: Initial conditions

Boundary	k	ν_T	ω	p	$\overline{u}$
inlet	`fV`	`calc`	`fV`	`zG`	`fV`
top	`zG`	`zG`	`zG`	`zG`	`zG`
hull	`wF`	`wF`	`wF`	`zG`	`nS`
outlet	`zG`	`calc`	`zG`	`fV`	`zG`
bottom	`wF`	`wF`	`wF`	`zG`	`nS`
propInlet	`zG`	`zG`	`zG`	`zG`	`fR`
propOutlet	`zG`	`zG`	`zG`	`zG`	`fR`
propTopAndBottom	`zG`	`zG`	`zG`	`zG`	`s`
frontAndBack	`empty`	`empty`	`empty`	`empty`	`empty`

`fV fixedValue`: This boundary condition assignes a fixed value constraint throughout the simulation.

`calc calculated`: This boundary condition is not designed to be evaluated; it is assmued that the value is assigned via field assignment

`zG zeroGradient`: This boundary condition applies a zero-gradient condition from the patch internal field onto the patch faces

`wF` `wallFunction`: There are different wall functions for each flow property.

`kqRWallFunction`: This boundary condition provides a simple wrapper around the zero-gradient condition, which is used for the turbulent kinetic energy k

`nutUSpaldingWallFunction`: This boundary condition provides a wall constraint on the turbulent viscosity, ν_T, based on velocity

`omegaWallFunction`: This boundary condition provides a wall constraint on the specific dissipation rate, ω

`nS` `noSlip`: This boundary condition fixes the velocity to zero at walls

`s` `slip`: This boundary condition provides a slip constraint

`fR` `flowRateInletVelocity`: This boundary condition specifies a volumetric or mass flow rate

`empty` This boundary condition provides an "empty" condition for reduced dimensions cases

4.4 Mesh sizes

In order to determine the most fitting mesh size distribution, a mesh convergence study is performed. In order to determine the convergence, an arbitrary coefficient calculated by the algorithm is compared for each mesh refinement. The meshes are characterized by the smallest cell length in spanwise direction. The smallest cell side-wall lengths are shown in table 4.5.

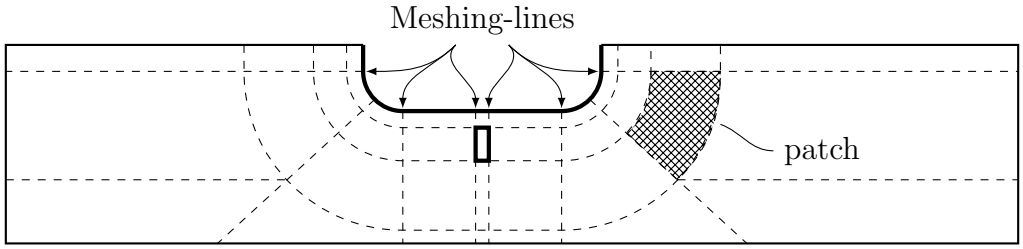


Figure 4.3: Some named properties to avoid confusion

The mesh sizes are scaled evenly, increasing the number of cells or volumes used gradually. They can be categorized through the smallest cell side-wall, or by the number of cells. The meshing is based on the smallest cell side-wall length, which are used around the hull. The meshing-lines make out the boundaries of each patch of volume and cells, the ends have to be located at the outer boundaries. Essentially stretching from boundary to boundary. This implies some limitations, with regards to the cell sizes in different regions. Two patches that share a border, must necessarily also share a mesh refinement in the normal direction to that border. To view a visual representation of the mesh patches and lines, see figure 4.3.

Each meshing line have to cross from one boundary to another, and due to a desire to mesh the immediate area around the hull as accurately as possible, the mesh cell sizes are set to different values throughout the mesh. The cells nearest the hull are defined by having the "finest", or smallest cell length of the mesh. Some mesh lines are also given the attribute that each consecutive cell length, away from the hull is enlarged a little in size. This stepwise scaling is termed the expansion ratio, r_{exp}. The meshing lines given an expansion ratio, and what the ratio is, becomes more apparent in figure 4.4.

For clarity, the arrows symbolize the direction of the meshing lines, the density of the lines are determined at the boundaries, and are different for each color. The blue arrows indicate the meshing lines placed perpendicular to the hull surface, these lines have the highest density, with equally distributed lines. The red arrows indicate the meshing lines oriented parallel to the hull surface. They start with the same density as the blue lines near the hull, and expands gradually outwards. The expansion is also constant across the borders of the patches. Notice that the expansion is in a perpendicular direction to the arrows. For the red arrows, meaning expansion in the direction away from the hull. The green arrows indicate the outer regions of the domain, from top to bottom along the inlet and outlet. The mesh density here is determined by matching the spacing for the outermost region of the red arrows, and quite a bit larger expansion ratio. This way the computational power is focused on the hull and propeller. The last set of meshing lines are represented by the cyan arrow. The region that stretches from inlet to outlet over the bottom. This region has the least dense mesh, but especially for shallow water, as in figure 4.4, the region becomes a little packed close to the bottom. The lines are equally spaced, meaning the expansion ratio equals zero.

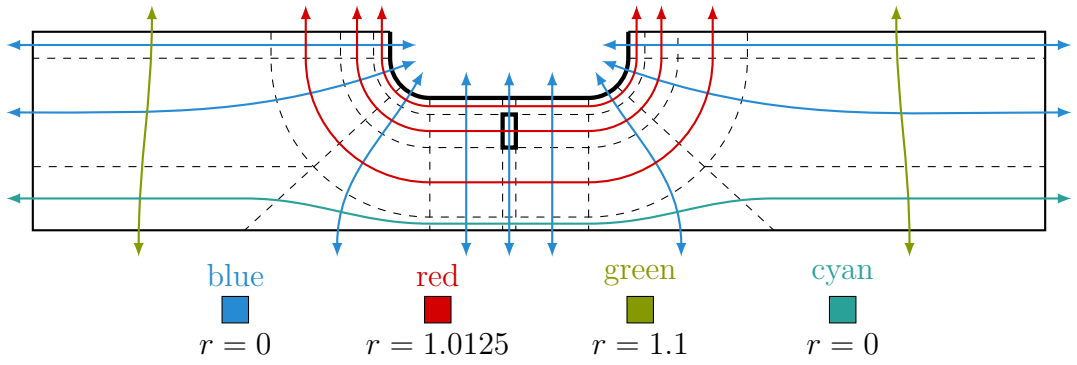


Figure 4.4: The different meshing line groups, r symbolises the expansion ratio for each consecutive cell. All meshing lines are defined so that the clustering of cells happen towards the hull.

When it comes to deriving the number of cells in the domain, and the size of each cell, a convenient parameter is the Courant-Friedrich-Lewy number; equation (4.9). It is a non dimensional parameter that reveals the length traveled over a single time step in terms of the cell size. In essence, when the CFL number equals 1, it means a fluid particle traverses the entire length of a cell during one timestep. If the CFL number exceeds the value of 1, entire cell volumes can be surpassed by a fluid particle during a single timestep, which leads to inaccuracy and untrustworthy

results. For the present analysis the correct CFL number is not easily predicted correctly. The inlet velocity is applied, and therefore known, but the accelerated flow inside the domain on the other hand is not. That is for the analysis to derive.

$$CFL = \Delta t \left(\sum_{i=1}^{n} \frac{u_i}{\Delta x_i} \right) \leq CFL_{\max} \tag{4.9}$$

Where n is symbolizing the number of dimensions for the simulation to take place.

In practice, many configurations of cell sizes and distributions will simply be tested. As the cell sizes change, so does the CFL number, and in order to avoid the simulation exploding[3], or the consequences of untrustworthy results the timestep is altered parallel to the mesh refinement. This is achieved by a simplified CFL number computed for one dimension, using the known inlet velocity and the size of the smallest cell in the mesh. Shown for clarity in equation (4.10). The mesh sizes that are run in the mesh refinement analysis, see 5.1, are posted in table 4.5.

Table 4.5: Characteristic sizes for mesh refinement process

Smallest cell size	Number of cells	Order
$6.400 \cdot 10^{-3}$	9966	$1.00 \cdot 10^{4}$
$3.550 \cdot 10^{-3}$	24965	$2.50 \cdot 10^{4}$
$2.250 \cdot 10^{-3}$	49467	$5.00 \cdot 10^{4}$
$1.710 \cdot 10^{-3}$	75149	$7.50 \cdot 10^{4}$
$1.400 \cdot 10^{-3}$	100366	$1.00 \cdot 10^{5}$

$$\Delta t_{\max} = \frac{CFL_{\max} \cdot \Delta y}{U_{\infty}} \tag{4.10}$$

Where $CFL_{\max}$ is the desired CFL number for the analysis, Δy is the size, or really the length of the smallest cell in the domain and U_{∞} is the free stream velocity. As discussed earlier, even though a desired CFL number is chosen, it will not equal the actual CFL number from the analysis. Emphasizing that this is a simplified approach to find an appropriate timestep.

4.5 Validation

The present case is difficult to validate, much because of the presence of the propeller. In order to properly validate a CFD experiment, a comparison to an actual experiment should be conducted. Therefore a propellerless case is considered. Then the drag coefficient of the hull can be compared to a conducted experiment, like one of the many described in Hoerner 1965.

For the 2D case the experimental results described in figure 4.5 are considered. The hull considered in the CFD experiments has a defined curvature on the corners of

[3]meaning that the derived values asymptotically approach infinity

the hull, and from the draft and curvature of the model, a fitting case for comparison can be made.

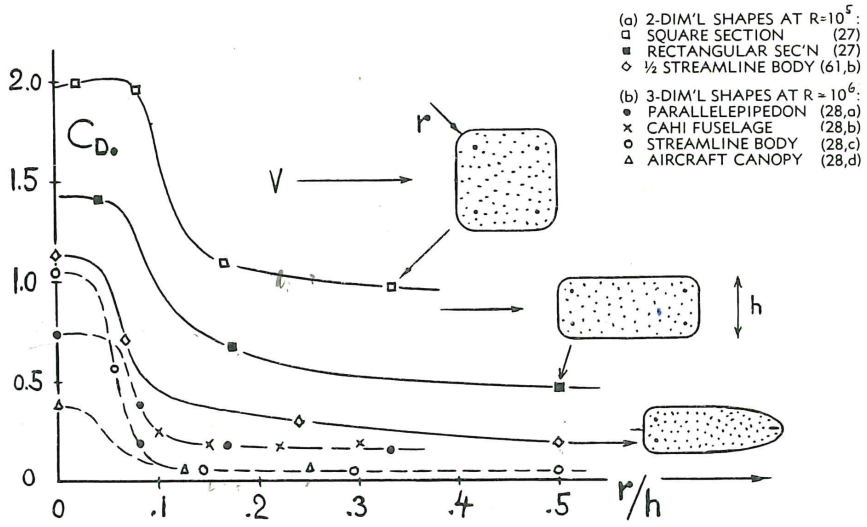


Figure 4.5: Influence of rounding radius upon the drag coefficient of various blunt bodies; Hoerner 1965

Considering figure 4.6, a possible way to think of this is the hull mirrored along the waterline. Assuming the flow traverses past the hull with a particular drag coefficient. If the entire mirrored body were submerged and surrounded by water, the drag coefficient would be equal to the rectangular cross-section, figure 4.5 for a certain rounding radius, and h equal to two times the draft. By considering half the drag coefficient at the submerged level, with half the reference area in equation (4.11), the drag coefficient of the hull floating on the surface is found, as in equation (4.12).

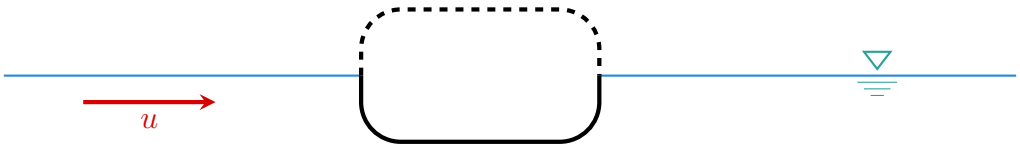


Figure 4.6: A mirrored rectangular cross section

$$C_D = \frac{F_D}{\frac{1}{2}\rho u^2 A} \tag{4.11}$$

$$\frac{r}{h} = \frac{3\text{m}}{2 \cdot 5\text{m}} = 0.3 \overset{\text{figure 4.5}}{\Rightarrow} C_D \simeq 0.5 \tag{4.12}$$

By applying a rounding radius, the drag of the original geometry is reduced, but according to Hoerner 1965, it is a less than perfect method of reducing drag. A cause of worry is how it affects the separation of the flow and a potential vortex street following in the wake. It makes complete sense that the frictional drag is split

in half, as half of the geometry is removed, but if there is a presence of a vortex street for the entire geometry and not for the half of it, the pressure drag would not be simply half the original value.

In Hoerner 1965 there are some measurements of the drag coefficient, C_D for some vortex-street producing shapes, where a "trick" to avoid vortex shedding has been applied. The trick is a so-called, "splitter" plate, which prohibits or delays the vortex shedding process for higher $\mathfrak{Re}$ than otherwise. The splitter plates effectively reduce the Strouhal number, equation (4.13) to the order of half the usual value, and the drag coefficient is noticeably decreased.

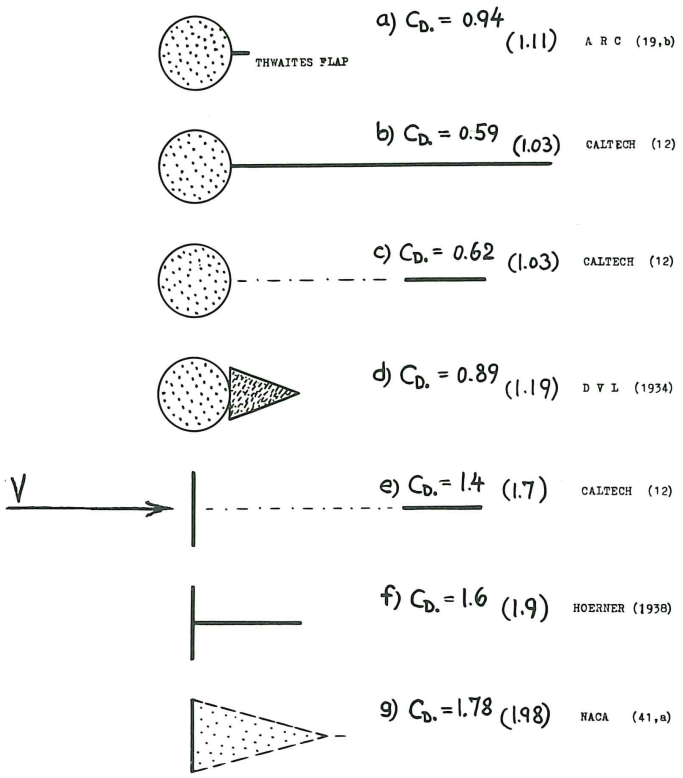


Figure 4.7: Influence of splitter plates (and similar devices) on the drag coefficients without wake interference. $\mathfrak{Re} \in [10^4,\ 10^5]$; Hoerner 1965

$$\mathcal{S}t = \frac{f\mathcal{L}}{u} \tag{4.13}$$

According to Faltinsen 1993, the free surface tends to act as an infinitely long splitter plate, and a simple way to explain this behavior is by using figure 4.8. Let the shed vortex be called Γ, which is a function of time. Then, an image vortex is applied above the surface to explain how the free surface effect can decrease the drag coefficient. This image vortex has a more decisive influence over the motion of the natural vortex than vortices in a vortex street have on each other. Since it is known that a splitter plate reduces the drag, and we know that there is a correlation between the velocity of the shed vortices and the force experienced by the vessel,

it becomes clear that the free surface will influence the drag coefficient. Thereby reducing it, compared to the whole body submerged.

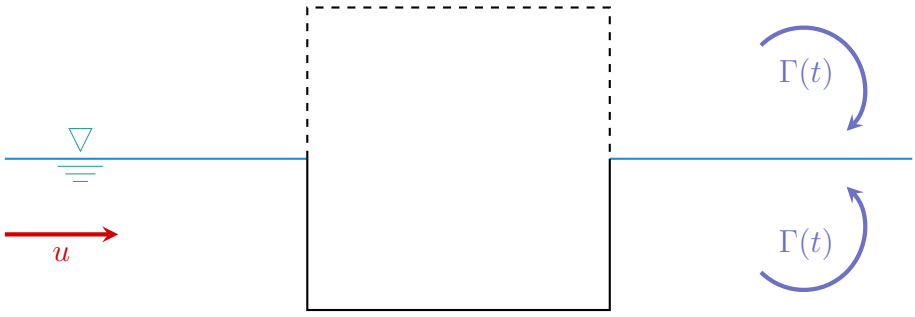


Figure 4.8: Simple vortex system with an image flow above the free surface so the rigid free surface condition is satisfied, from Faltinsen 1993

It was early discovered that the flows of interest in this project are turbulent, see equation (1.1). Furthermore, turbulence does affect the separation point of the boundary layer, as well as the drag coefficient. According to Faltinsen 1993, the critical Reynolds number for a smooth cylinder is $2 \cdot 10^5$, which is around the same value as the drag coefficients in figure 4.5, which means the geometries are most likely experiencing laminar separation. In Aarsnes et al. 1985 the results of drag coefficient are separated into three regions, defined by different ranges of the Reynolds number, *subcritical*, *transcritical* and *supercritical*.

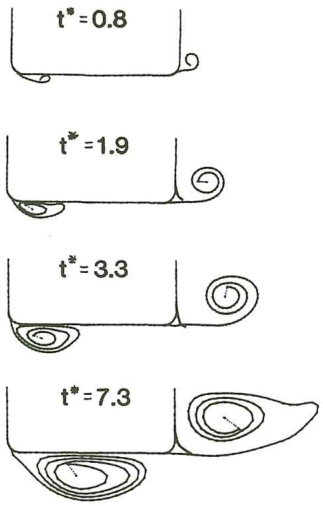


Figure 4.9: Wake development for a particular hull cross section. Calculations are shown for different time instants t and area based on numerical calculations by Aarsnes et al. 1985 with a thin free shear layer model; found in Faltinsen 1993

In Faltinsen 1993, it is shown how the boundary layer separates as early as the leading edge for subcritical flow based on a numerical study on a particular hull by Aarsnes et al. 1985, see figure 4.9. Turbulent boundary layers, on the other hand, have a more remarkable ability to sustain adverse pressure gradients before separation occurs. According to Faltinsen 1993, this is the reason why there is no

separation at the leading bilge for transcritical flow, which is the case for most hulls in full scale. From Aarsnes et al. 1985 some calculations were performed to find the 2D drag coefficient for a hull section in cross-flow conditions, see figure 4.10

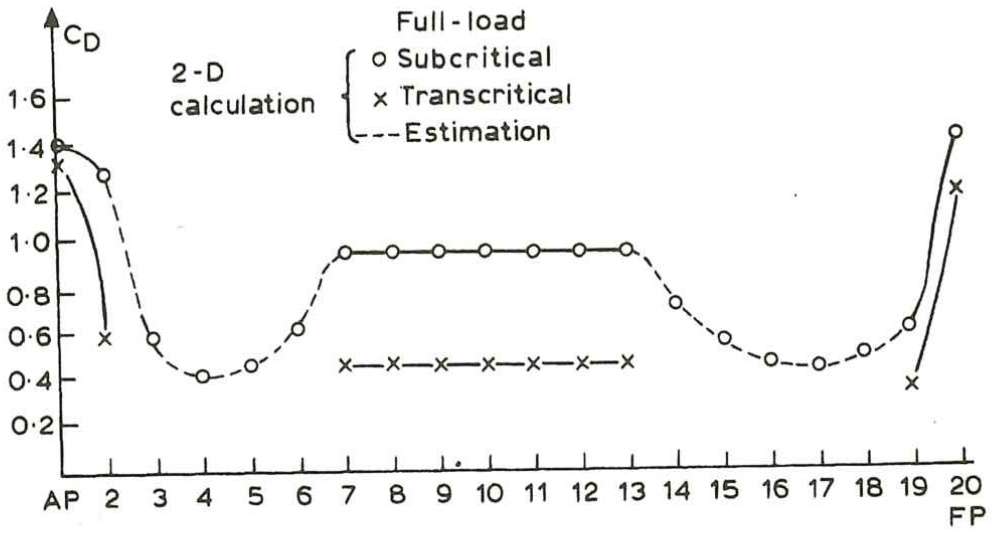


Figure 4.10: Calculated and estimated drag coefficients C_D for two dimensional cross flow past cross sections along a particular ship; from Faltinsen 1993

The advantage of the results in figure 4.10 relies on taking into account the free surface and turbulent boundary layer that impacts the value of C_D. The major drawback is the lack of consideration for the rounding radius applied to the hull used in the experiments, which is a significant advantage when considering the plot from Hoerner 1965 in figure 4.5. Since splitter plates, and thereby the free surface, decrease the drag of vortex street-producing shapes, C_D ought to be a bit lower than what is presented in figure 4.10.

The goal of the validation is to check that the established method solves the equations correctly and can be trusted to a higher degree to solve the equations correctly for other cases; this is a necessary assumption as long as additional model tests are not performed, although it is not a guarantee for reliable results. The simulation in question utilizes turbulence modeling, and the applied method of turbulence model is the desired target for validation. For this reason, there is no point in creating a laminar flow model for validation with the results shown in figure 4.5. The easiest solution is to alter the rounding radius of the hull to conform with the results presented in figure 4.10; since the rounding radius is not presented, a couple of close relatives must therefore be constructed, and from there be compared in likeness to the results of figure 4.10. A second alternative that can also be implemented is a drag coefficient found by comparing the results of figures 4.5 and 4.10.

Considering equation (4.12), C_D seems to achieve about the same value as the transcritical drag coefficient from figure 4.10. Since the large rounding radius chosen in the project implies a certain reduction in drag, combined with the fact that due to the surface and turbulent boundary layer, the drag coefficient should be even lower. By considering the difference between subcritical and transcritical in figure

4.10, an estimate for the "improvement" in drag can be extracted

$$\left.\begin{array}{l} C_{D,\text{ sub}} \simeq 0.95 \\ C_{D,\text{ trans}} \simeq 0.5 \end{array}\right\} \rightarrow \quad \frac{C_{D,\text{ trans}}}{C_{D,\text{ sub}}} \simeq 0.5 \;\Rightarrow\; 50\% \text{ decrease} \tag{4.14}$$

Now there are two different drag coefficients available for comparison, based on equations (4.12) and (4.14). These results will be used as a lower and upper bound for what to expect of the drag coefficient found in the CFD experiment.

Since both cases from Hoerner 1965 and Aarsnes et al. 1985 are for shapes in free-flow conditions, the $\frac{W_D}{T}$ ratio needs to be large enough to avoid the influence of the bottom boundary layer, i.e., deep water conditions. Therefore this test also provides a valuable opportunity to define what deep water conditions are for the main test subject. The mesh should be kept as similar as possible to the original mesh, and the number of cells used will be the same as found to be necessary from the mesh refinement analysis discussed in 4.4, the meshing line composition and expansion ratio are shown in figure 4.11. The main difference lies in the removal of the propeller. The inner meshing line situated parallel to the hull replaces the propeller, while the cells' expansion remains the same to keep the mesh as similar as possible. The curvature is adapted to the experiments measured against the drag coefficient from figure 4.10.

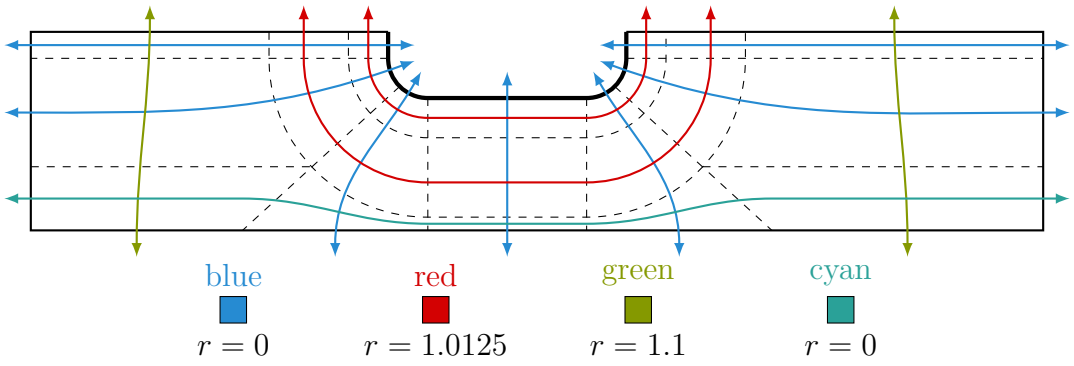


Figure 4.11: The different meshing line groups, r symbolises the expansion ratio for each consecutive cell. All meshing lines are defined so that the clustering of cells happen towards the hull.

CHAPTER

5

RESULTS

In this chapter plots and figures generated from the results in OpenFoam are presented. First the results of the mesh refinement analysis is posted; 5.1, and then the results from the validation of the method; 5.2. All of the results have a short description, before a more thorough discussion is considered in chapter 6.

5.1 Mesh refinement analysis

The mesh refinement was performed for the 2D hull with incoming current flow, as well as a constant volume flow through the propeller disk for the smallest $\frac{W_D}{T}$ ratio that did not outright crash, or explode during computations. The number of cells tested are visible in table 4.5, and in the horizontal axis of the plots. All of the simulations was programmed to stop when a 100 seconds of simulated time was computed. The amount of time required for each simulation to reach this limit is shown in table 5.1.

Table 5.1: Amount of time required for each simulation to reach 100 simulated seconds

Number of nodes	days	hours	minutes
$1.00 \cdot 10^4$	0	5	39
$2.50 \cdot 10^4$	0	22	44
$5.00 \cdot 10^4$	1	12	39
$7.50 \cdot 10^4$	1	23	34
$1.00 \cdot 10^5$	2	18	21

5.1.1 The drag coefficient

OpenFoam separates the coefficients into two components along the defined direction in the code. Figure 5.1 demonstrates drag coefficient for the front of the hull, or side closest to the inlet. The values of the drag are uncharacteristically large, which will be further discussed in 6.1.

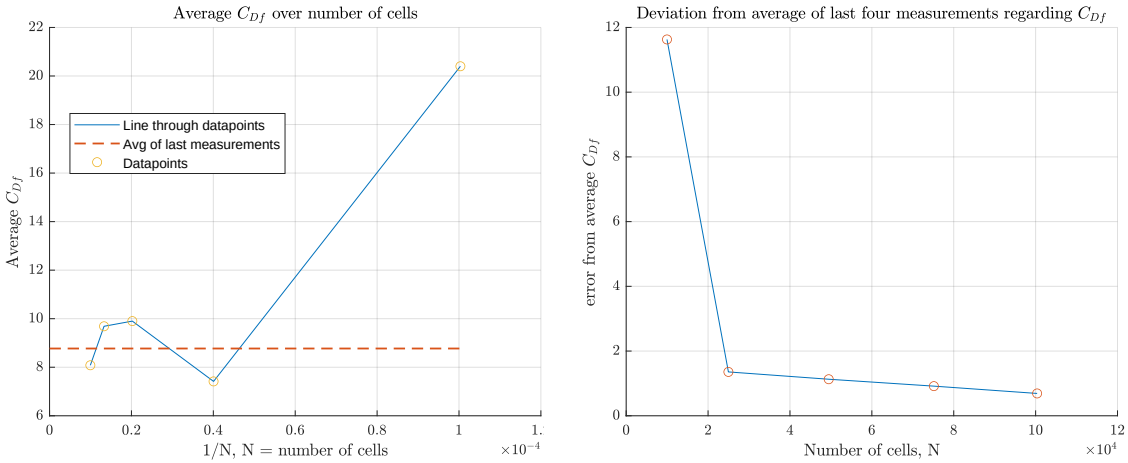


Figure 5.1: Average C_D for the front of the hull, with plotted divergence from mean of the last four calculated values

Figure 5.2 demonstrates the drag coefficient for the hindmost part of the hull, or side closest to the outlet. The values are in the ballpark of an expected drag coefficient for the entire hull, i.e. to large, this is part of the discussion in 6.1.

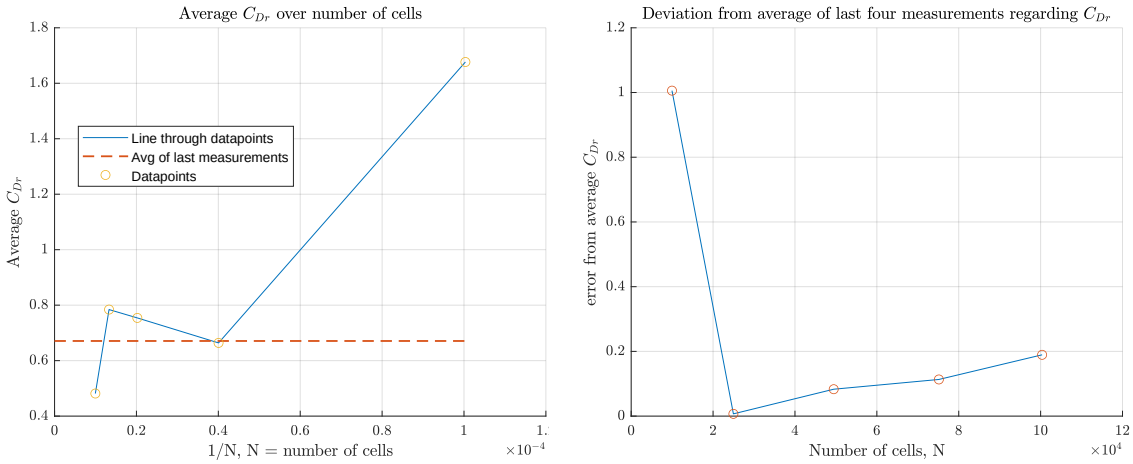


Figure 5.2: Average C_D for the rear part of the hull, with plotted divergence from mean of the last four calculated values

Figure 5.3 demonstrates the drag coefficient for entire hull. The values of the drag are uncharacteristically large, which will be further discussed in 6.1.

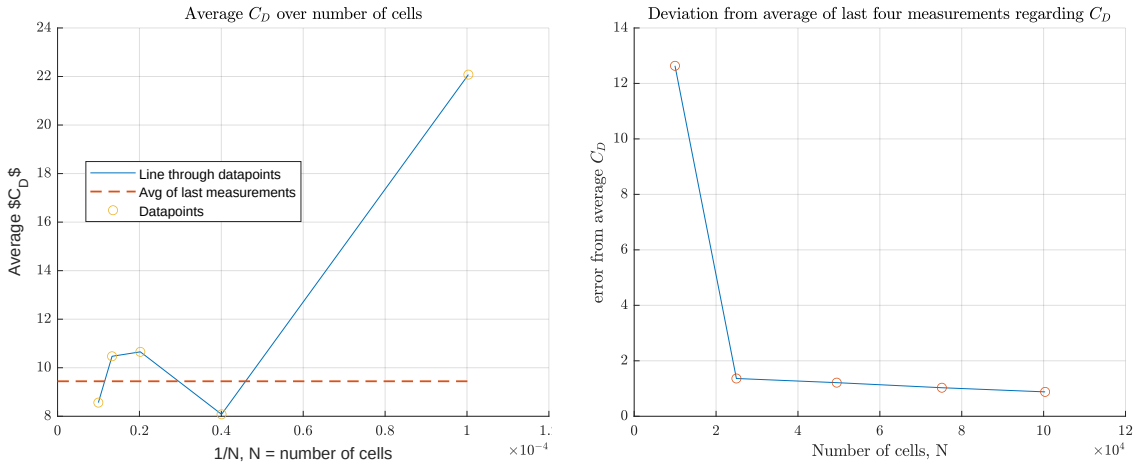


Figure 5.3: Average C_D for the entire hull, with plotted divergence from mean of the last four calculated values

5.1.2 Lift coefficient

Figure 5.4 shows the derived lift coefficient for the hull. The coefficient itself does not offer much usefull information, but is included to support the assessment of convergence of the method. The result is further discussed in 6.1.

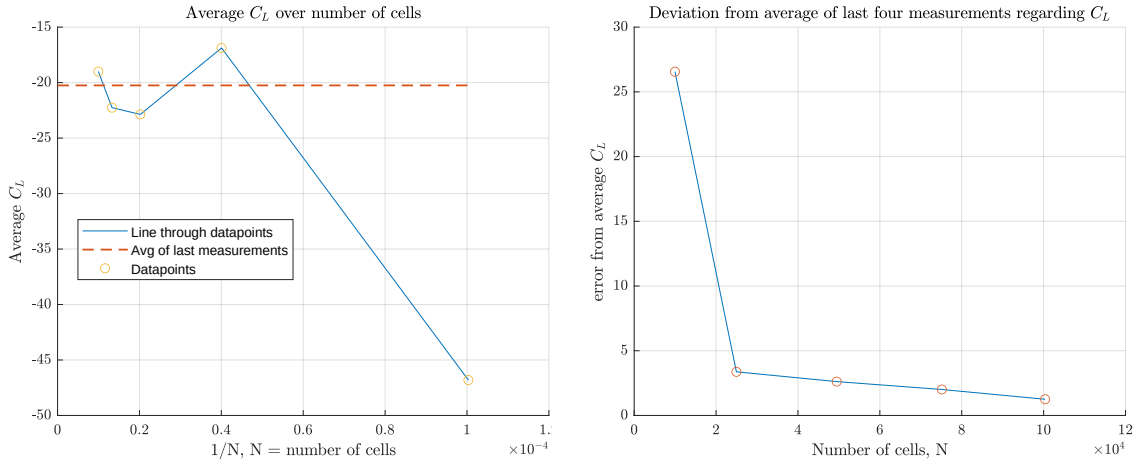


Figure 5.4: Average C_L for the hull, with plotted divergence from mean of the last four calculated values

5.1.3 Roll moment coefficient

The roll moment is defined the same way as for a full scale ship, rotation around the x-axis. This is also a value that is mainly included for the assessment of the convergence.

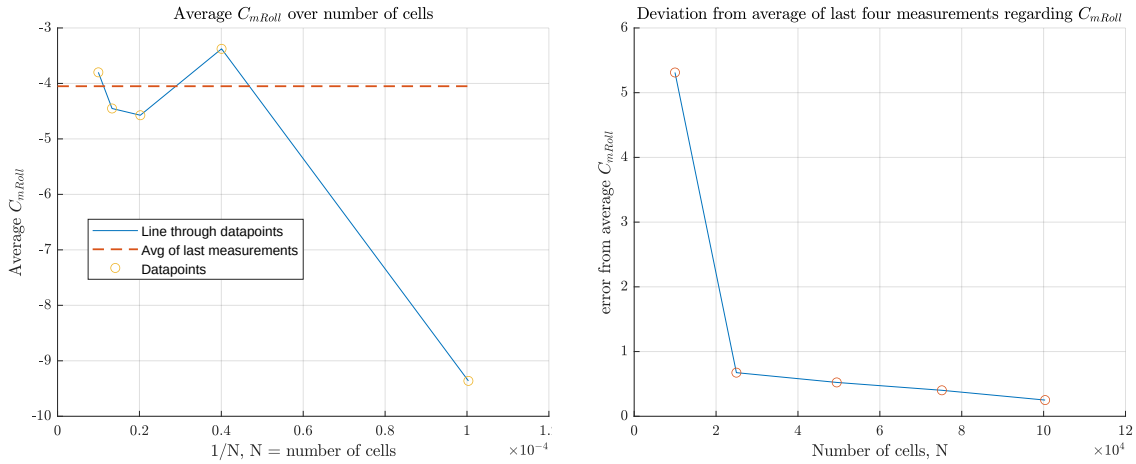


Figure 5.5: Average C_{mRoll} for the hull, with plotted divergence from mean of the last four calculated values

5.1.4 Velocity profiles from mesh refinement analysis

A velocity profile is extracted for each mesh at an arbitrary point in time in order to show how these profiles look in terms of flow velocity. Important to note that these profiles are just snapshots, and therefore unable to represent the entire time series.

Figure 5.6 shows the least refined resolution. An event in the time series where the jet flow appear to curve towards the surface. The rest of the flow domain seems to be still or moving slightly backwards.

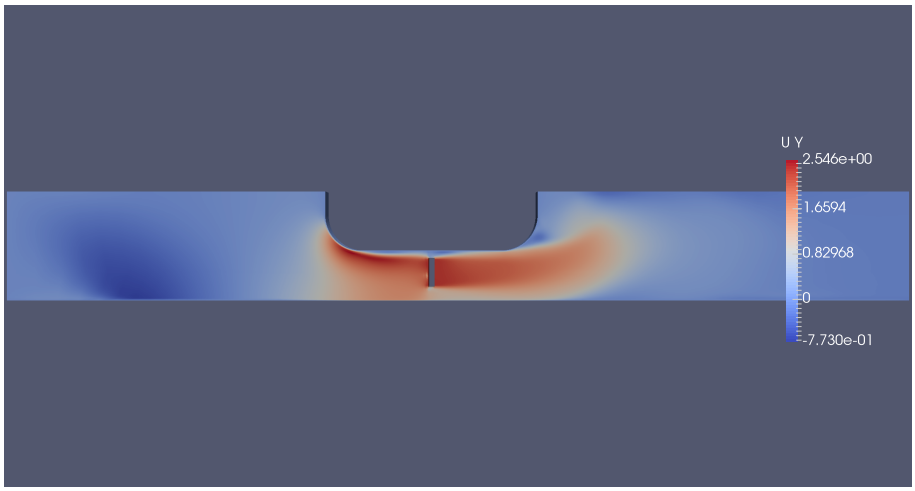


Figure 5.6: Velocity profile of $u_2 = v$ from the transient solver pimpleFoam with $\frac{W_D}{T} = 1.85$, $t = 39$s, the lowest resolution with 10^4 cells

Figure 5.7 is an attempt to capture the same event as in figure 5.6. The event seems to happen a little earlier in this simulation, and the surrounding fluid in the rest of

the domain seems to be, if not still, almost still. Some regions are also moving in negative y-direction.

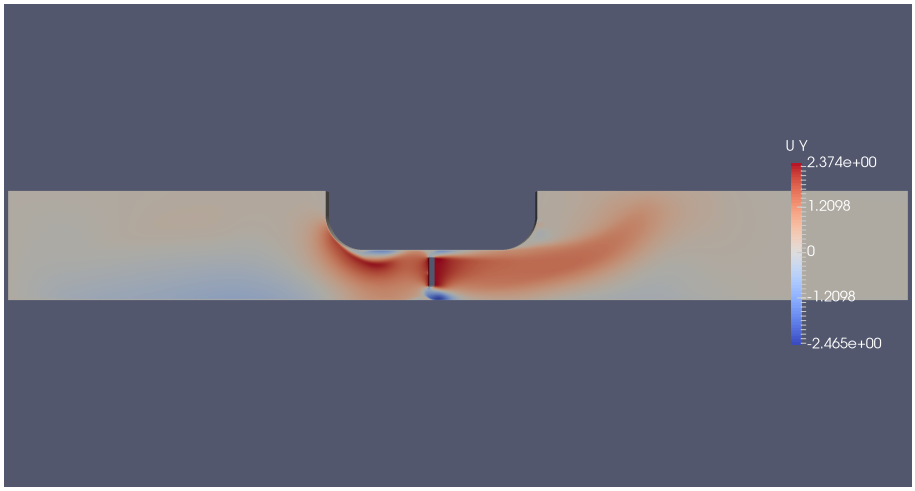


Figure 5.7: Velocity profile from the transient solver pimpleFoam with $\frac{W_D}{T} = 1.85$, $t = 23$s, second lowest resolution with $2.5 \cdot 10^4$ cells

Figure 5.8 displays another event, much later in the simulation. This event also involves the jet flow curving upwards towards the surface, while the rest of the domain stays either still or moving in negative y-direction. Note that the largest velocity found in the domain is in negative direction.

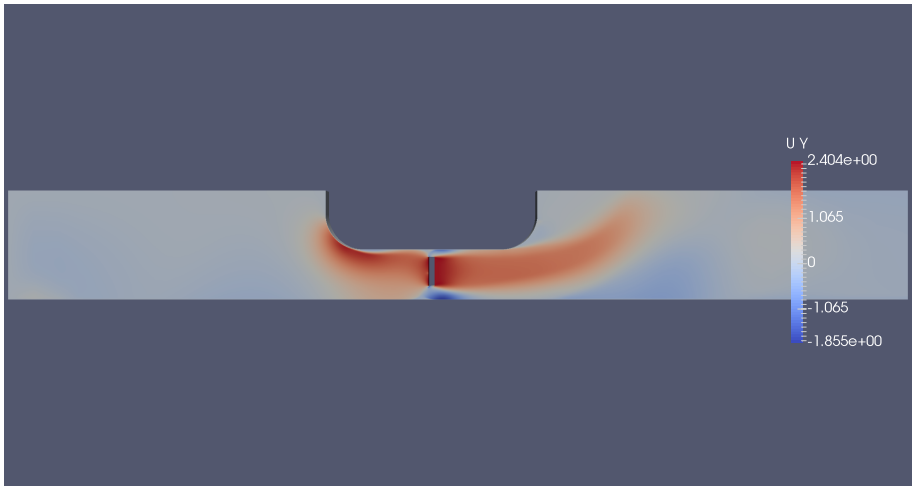


Figure 5.8: Velocity profile from the transient solver pimpleFoam with $\frac{W_D}{T} = 1.85$, $t = 87$s, third largest resolution with $5 \cdot 10^5$ cells

Figure 5.9 depicts the second largest refinement tested. The same events seem to happen, but earlier for finer mesh refinements. The same sort of curving motion, while the rest of the domain is either still or moving in negative y-direction. This snapshot also shows that the largest velocity component is in negative direction.

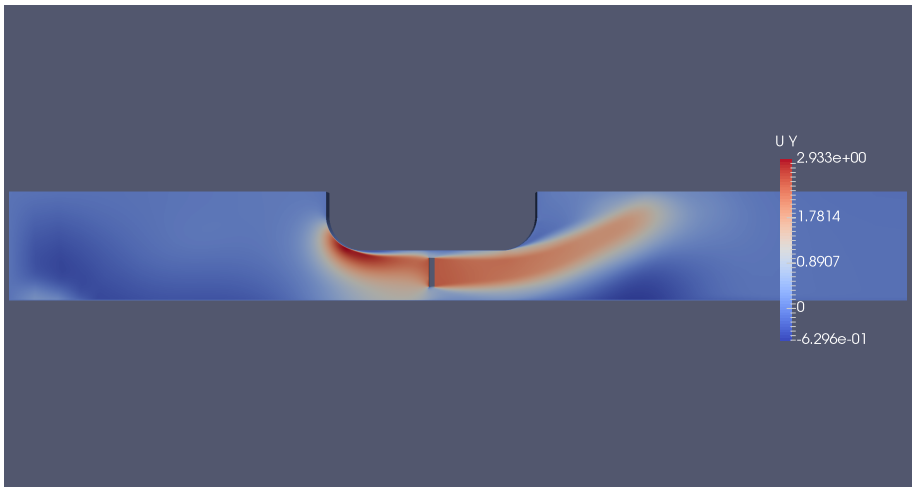


Figure 5.9: Velocity profile from the transient solver pimpleFoam with $\frac{W_D}{T} = 1.85$, $t = 69$s, second to largest resolution with $7.5 \cdot 10^4$ cells

Figure 5.10 shows a snapshot from the last mesh refinement simulation. In this snapshot the curve seems less pronounced, clinging a bit more against the bottom. The velocities in the rest of the domain are still, or contain almost the same magnitude in negative y-direction.

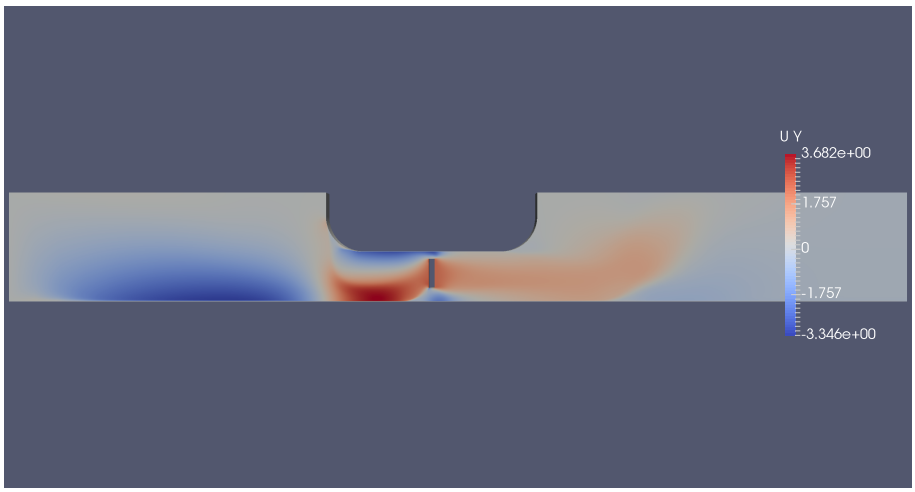


Figure 5.10: Velocity profile from the transient solver pimpleFoam with $\frac{W_D}{T} = 1.85$, $t = 16$s, the largest resolution with 10^5 cells

5.2 Validation

The following figures display a comparison between the drag coefficient computed by the altered simulation without the propeller disk underneath the hull. The same mesh density as found favorable through the mesh refinement analysis; 5.1 has been used. The plots are further discussed in the next chapter; 6.

Figure 5.11 shows the numerically computed drag coefficients from the transient

pimpleFoam solver, in the form of a tower diagram. Unfortunately, all of the drag coefficients were found to equal zero by the algorithm, indicating some error, further discussed in 6.2. The gray outlined area is the range of the expected value of the drag, discussed in 4.5.

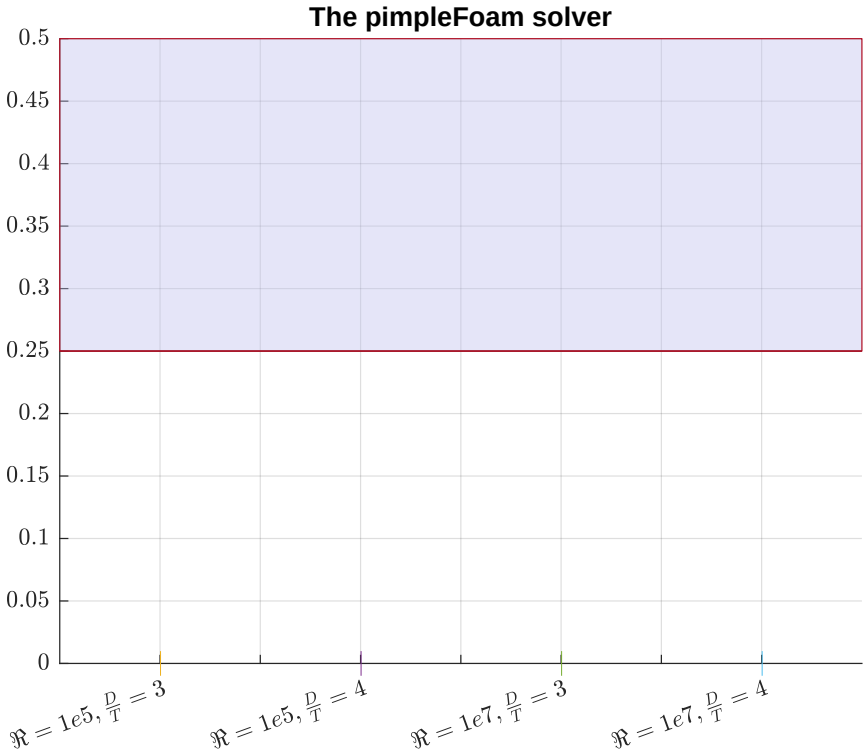


Figure 5.11: The validation results from the pimpleFoam solver, with the expected value of the drag coefficient outlined

Figure 5.12 shows the velocity profile through a contour plot, generated in paraView. The profile is generated at the last second of simulated time. The plot demonstrates at least one error apparent in the method, since no apparent flow seems to actually traverse the domain. The figure is further discussed in 6.2.

Figure 5.12: Instantaneous snapshot of the velocity field from pimpleFoam transient simulation, $\mathfrak{Re} = 10^7$, $\frac{W_D}{T} = 4$ and $t = 100$s

Figure 5.13 shows the numerically computed drag coefficients from the steady state simpleFoam solver, in the form of a tower diagram. The analysis was created to explore a more efficient, and different approach to the flow simulation. Unfortunately, all of the drag coefficients were found to equal zero by the algorithm, indicating that the same error is most likely present for this case too, further discussed in 6.2. The gray outlined area is the range of the expected value of the drag, discussed in 4.5.

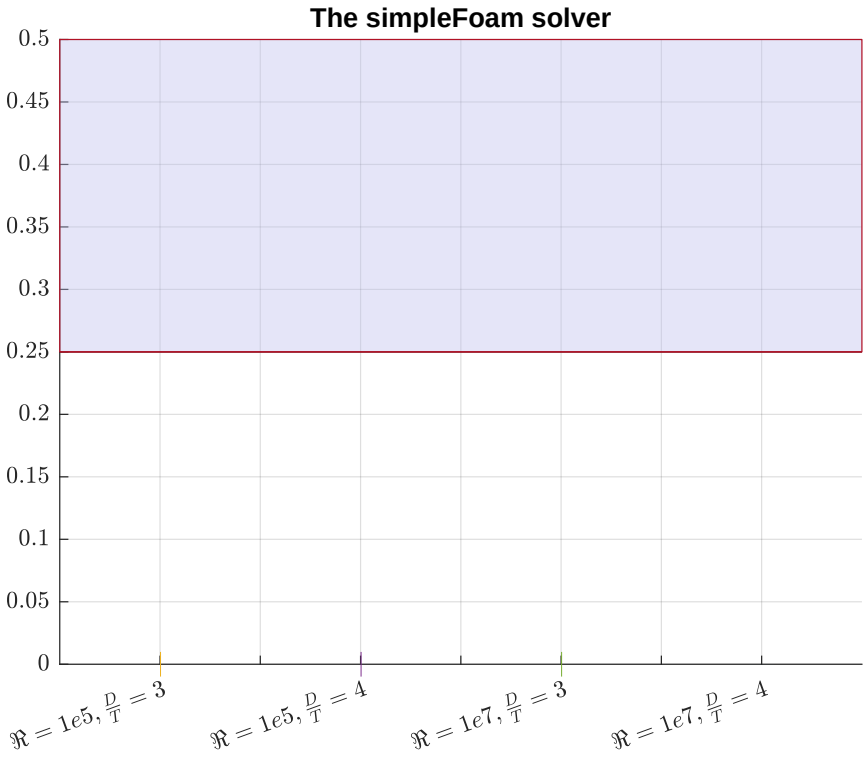


Figure 5.13: The validation results from the simpleFoam solver, with the expected value of the drag coefficient outlined

Figure 5.14 shows the velocity profile through a contour plot, generated in paraView. The profile is generated at the last second of simulated time. The plot confirms that at least one of the same problems apparent in 5.12 is still present with the steady state solver. The figure is further discussed in 6.2.

Figure 5.14: Snapshot of the steady state velocity field from the simpleFoam simulation, $\mathfrak{Re} = 10^5$, $\frac{W_D}{T} = 3$ and $t = 100$s

Due to the lack of conclusive results, another attempt was made at validating the method, see figure 5.15. The alterations made to the analysis are discussed in 6.2. With regards to the test regime, the experiments have moved into a debugging phase. The drag coefficients seem to converge towards a too large value.

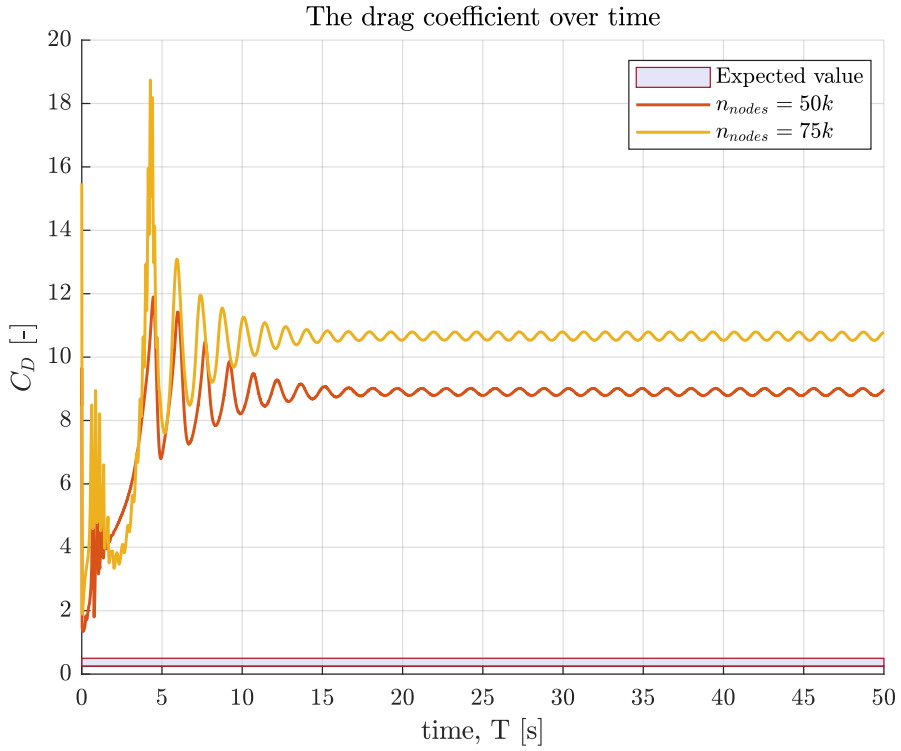


Figure 5.15: C_D as a function of time plotted with expected value range found in 4.5, computed with the pimleFoam solver, $\frac{W_D}{T} = 3$, $\mathfrak{Re} = 10^7$, n_{nodes} is the number of nodes used and k is meant to symbolize 10^3

CHAPTER

6

DISCUSSION

As stated in the introduction, the thesis aims to explore the influence of shallow waters on thrust loss using CFD simulations and knowledge from the literature. Thrust loss proved to be a pretty diverse subject, as many possible ways for thrust loss to occur were explored in 3.

Ventilation is influenced by the wave elevation, as discussed in 3.1, and therefore a subject of interest since both wave dynamics and global forces change for shallow waters. However, the regulations for service and installation vessels are rigorous concerning the safety of any operations. Therefore, nearly all conditions where ventilation is possible are avoided altogether, not only due to the risk of ventilation. Ventilation becomes impossible in a calmer environment with certain submergence on the propeller.

Thruster-to-thruster interactions can be detrimental for thrust capacity, as discussed in 3.2. However, the effects are mostly caused by weaknesses in the configuration and placement of the thrusters. These effects can be completely, or at least to a satisfactory degree, removed by decent design solutions. Therefore these effects are not considered to be of any interest to shallow water interactions.

Currents will also be able to interact with the propeller race, as discussed in 3.3. The flow will, at specific flow velocities and heading angles, be able to divert the propeller jet, but to what extent is fully flow-dependent and therefore very difficult to quantify. Although it is not directly relevant to thrust loss, an important aspect to consider is the increased resistance related to currents in shallow waters. As discussed in 3.4, the blockage of the flow might, in some cases, lead to substantially increased resistance, but an emphasis should be directed to the small likelihood of these extreme situations of blockage described in Koop 2015. However, the influence of the bottom boundary layer is an interesting aspect that could interact with the

propeller and increase the blockage.

6.1 Mesh refinement analysis

As the CFD computations were completed, the author quickly found the required time for OpenFOAM to complete the simulations very demanding for the strict thesis schedule, as seen in table 5.1. The next refinement level of the analysis was also attempted to be run, as only five data points were considered insufficient when assessing the method's convergence level. After 12 hours of running, OpenFOAM computed no more than 0.5 seconds of simulated time. At that point, the author aborted the mesh refinement analysis for more exemplary mesh distributions to test how the method performed for other parameters, namely the validation case.

6.1.1 Drag coefficients

The drag coefficients' value seems too large and is, at this stage, a significant cause of worry. C_D for the entire hull, figure 5.3, almost seems to start converging at a value of around 8, which is unrealistic, even with the higher velocities stimulated by the propeller disk. An alternative interpretation is that the method converges extremely slowly wrt the number of elements, which is very unfortunate when each analysis takes up so much time to compute. In figure 5.1, it is clear where the main contribution to the total drag coefficient comes from, as this value seems to make up about 90% of the total drag coefficient. As discussed in section 4.5, the drag coefficient is expected to be below 1, especially in a turbulent flow. The drag coefficient for the rear; figure 5.2 is only slightly larger than the expected drag for the entire hull. The fact that it is larger than the expected value for the whole hull is an indication that something is most likely wrong.

6.1.2 Lift and roll moment coefficients

The other plots from the mesh refinement, figure 5.4 and 5.5 are not subject to comparison with any known results. They are mainly included to assess the convergence of the method. That said, they display a very similar likeness to the drag coefficient results. Essentially they seem large and not completely converged. From the last four data points in both analyses, it appears that if the author had plotted the method with a higher resolution between the existing points, it would oscillate around some value close to the average. In both plots, the average of the last four values is also plotted, indicating some oscillation around the final value, given by a sufficiently large mesh density.

6.1.3 Thoughts on the improvement of mesh convergence analysis

In hindsight, these results are only valuable when the method functions correctly. In other words, without any values to compare against, it is more difficult to determine if something is faulty or wrong with the method. The amount of time available for debugging was debilitated by the amount of time required to complete an analysis, which the author should have handled differently. In this case, it was the lack of experience that impacted the decisions of the author. This entire process should have been omitted until a correct method for the case described in 4.5 was completed.

6.2 Validation

The validation results all show 0 drag. The unrealistic result is due to an error in the simulation, proving that the applied method is not ready and needs repair before more experiments are conducted. Contrary to the mesh refinement, these values are useful regardless of whether the method is working. The resulting drag coefficients, plotted for both solvers used, namely `PimpleFOAM` in figure 5.11 and `SimpleFOAM` in figure 5.13.

After some debugging a fatal and embarrassing error was discovered. The current velocity was defined to move 1m/s in the x-direction, and 0m/s in the correct y-direction, as defined in figure 4.1. Correcting the mistake resulted in figure 5.15. The values are all much larger than the expected drag coefficient range derived in 4.5, and strangely the value seems to increase for finer mesh resolutions.

6.3 The applied method

Turbulence is a three-dimensional effect. Fluctuations have a chaotic nature and act, therefore, in all directions. The three-dimensionality of turbulence is part of why circular and plane jets spread at different angles, as discussed in 2.3.1 and 2.3.2. The two cases are fundamentally different, which is the reason the original goal of the thesis was to investigate the interactions with a 3D CFD simulation. As may be evident from the results, producing reliable results in CFD is challenging and time-consuming.

Simulating this flow with CFD was initially chosen as an exciting way to study the effects involved in thrust loss. The potential of CFD may be substantial, but as the results are pointing out, very time-consuming to produce and possibly useless if some detail in the discretization or method is imprecise. From the authors' point of view, more experience in the subject was the secondary goal of this project and, therefore, worth the effort no matter the results. However, the project lost a lot of time due to inexperience.

The step down from 3D to 2D propeller and hull was an insufficient simplification.

Considering the time necessary to run the mesh refinement analysis and the lack of results to compare, it would have been much more efficient to start with the more straightforward case described in 4.5 and debug the method from there.

A reoccurring problem during the experiments has been analyses that crash due to asymptotic values in the results, especially regarding the `SimpleFOAM` solver. As if a steady-state solution of the 2D flow does not exist. Since `pimpleFOAM` was able to complete the simulation, it may imply that the flows are unsteady and subject to oscillations. However, the more likely scenario is something wrong with the method or discretization. Unfortunately, no solution to the problem has been found, and the analysis will remain incomplete.

The CFD solvers are built with complex algorithms, which this project has treated to some extent as a black box, considering input and output without enough consideration for the method applied. Why the results appear to be wrong is so far only speculation. One hypothesis is the influence of the wall functions applied to the specific dissipation ω, eddy viscosity ν_T and turbulent kinetic energy k. The wall functions are based upon boundary layer flow tests on flat plates and might therefore influence the results quite drastically due to the lack of compatibility with the blunt impact on the hull. The y^+ value discussed in 4.1 should have been optimized so that a value between $30 > y^+ > 300$ was achieved.

Another possible source of errors in the results could be the integration scheme. The integration algorithms are inadequately covered by the thesis, even though they matter for the results. However, the integration should not increase the value of the drag as drastically as seen in the results. The solver algorithms are only compatible with specific integration schemes, so different algorithms were used for each solver; see C for the codes used, and specifically C.3 to view the integration algorithms used.

Although it is fascinating due to the implied potential of CFD simulations, we should remember that the alternative to CFD was model tests, which could have been a more efficient approach to this subject. Nevertheless, model tests are noted since every CFD method is susceptible to the input variables and therefore needs to be validated by comparing it to a model experiment subjected to the same kind of flow.

CHAPTER

7

CONCLUSION

Shallow water affects the flow domain around the hull, influencing some but not all thruster interactions that cause thrust loss. For example, the Coanda effect will be influenced by the bottom surface, although the degree of influence and distance required for an interaction to occur is still unclear.

A vessel's total resistance will increase upon entering shallow waters and therefore require more thrust available to maneuver as effectively as in deeper waters, depending on the conditions.

Studying the interactions through CFD is an inefficient approach due to the lack of model test results available for use in the validation of the code. Simple model experiments with a propeller underneath a curved surface, preferably a hull, and little clearance between the surface and the bottom are advised to achieve decent results for comparison.

The results of this thesis are inconclusive due to a lack of validation and bugs. Therefore, debugging of the code should focus on integration schemes and optimizing the y^+ value for the wall functions.

BIBLIOGRAPHY

28th ITTC, Quality Systems Group of the (2011). 'ITTC–Recommended Procedures Fresh Water and Seawater Properties'. In: *Proceedings of ITTC*.

Aarsnes, JV, O Faltinsen and B Pettersen (1985). 'Application of a vortex tracking method to current forces on ships'. In: *Proceedings of Proc. Conf. Separated Flow around Marine Structures, Trondheim*. Vol. 309346.

Amdahl, Jørgen et al. (2015). 'TMR4105 - Marin Teknikk Grunnlag Kompendium'. In: *Marin Teknisk Senter, NTNU* 6, pp. 10.14–10.17.

Aokomoriuta (2022). *Law Of The Wall*. Wikipedia. URL: 'https://en.wikipedia.org/wiki/File:Law_of_the_wall_(English).svg' (visited on 8th June 2022).

Boussinesq, Joseph (1877). *Essai sur la théorie des eaux courantes*. Vol. 2. Imprimerie nationale.

Cebeci, T and AMO Smith (1974). 'Analysis of turbulent boundary layers'. In: *NASA STI/Recon Technical Report A* 75, p. 46513.

Cutler, A and J White (2001). 'An experimental and CFD study of a supersonic coaxial jet'. In: *39th Aerospace Sciences Meeting and Exhibit*, p. 143.

Faltinsen, Odd (1993). *Sea loads on ships and offshore structures*. Vol. 1. Cambridge university press.

Granville, Paul S (1953). *The calculation of the viscous drag of bodies of revolution*. Tech. rep. DAVID TAYLOR MODEL BASIN WASHINGTON DC.

Harlow, Francis H and Paul I Nakayama (1967). 'Turbulence transport equations'. In: *The Physics of Fluids* 10.11, pp. 2323–2332.

Hoerner, Sighard F (1965). 'Fluid Dynamic Drag, published by the author'. In: *Midland Park, NJ*, pp. 3.6–3.13.

Holmedal, Lars Erik (2002). 'Wave-current Interactions in the Vicinity of the Sea Bed'. In: pp. 3–8.

Johns, Bryan (1983). *Physical oceanography of coastal and shelf seas*. Elsevier, pp. 189–262.

Kolmogorov, Andrej Nikolaevich (1941). 'Equations of turbulent motion in an incompressible fluid'. In: *Dokl. Akad. Nauk SSSR*. Vol. 30.

Koop, Arjen (2015). 'Shallow water current loads on a LNG carrier using CFD'. In: *International Conference on Offshore Mechanics and Arctic Engineering*. Vol. 56482. American Society of Mechanical Engineers, V002T08A037.

Koushan, K (2004). 'Environmental and interaction effects on propulsion systems used in dynamic positioning, an overview'. In: *Proceedings of 9th International Symposium on Practical Design of Ships and other Floating Structures PRADS*, pp. 1013–1020.

Kozłowska, Anna Maria (2019). 'Hydrodynamic Loads on Marine Propellers Subject to Ventilation and Out of Water Condition'. In.

Kozlowska, Anna Maria and Sverre Steen (2017). 'Experimental analysis on the risk of vortex ventilation and the free surface ventilation of marine propellers'. In: *Applied Ocean Research* 67, pp. 201–212. ISSN: 0141-1187. DOI: https://doi.org/10.1016/j.apor.2017.07.006. URL: https://www.sciencedirect.com/science/article/pii/S0141118717300445.

Lehn, Erik (1985). *On the propeller race interaction effects*. NSFI.

Mathieu, Jean and Julian Scott (2000). *An introduction to turbulent flow*. Cambridge University Press.

Menter, Florian R (1992). 'Influence of freestream values on k-omega turbulence model predictions'. In: *AIAA journal* 30.6, pp. 1657–1659.

— (1994). 'Two-equation eddy-viscosity turbulence models for engineering applications'. In: *AIAA journal* 32.8, pp. 1598–1605.

— (1993). 'Zonal two equation kw turbulence models for aerodynamic flows'. In: *23rd fluid dynamics, plasmadynamics, and lasers conference*, p. 2906.

Michel, Roger (1952). 'Etude de la transition sur les profils d'aile-establissment d'un point de transition et calcul de la tranée de profil en incompressible'. In: *ONERA publication* 58.

Murad, Jousef (2022). *What is y+ (yplus)?* Ed. by SimScale Team. SimScale. URL: 'https://www.simscale.com/forum/t/what-is-y-yplus/82394' (visited on 8th June 2022).

Prandtl, L (1925). 'Uber die ausgebildete Turbulenz'. In: *ZAMM, 5p* 136.

— (1945). 'Uber ein Neues Formel-System fur die Ausgebildete Turbulenz, Nachr Akad Wiss, Gottingen, Math'. In: *Phys. Kl* 1945, p. 6.

Rotteveel, E and RG Hekkenberg (2015). 'The influence of shallow water and hull form variations on inland ship resistance'. In: *IMDC 2015: Proceedings of the 12th International Marine Design Conference, Tokyo, Japan, 11-14 May 2015*.

Rotteveel, Erik, Robert Hekkenberg and Auke van der Ploeg (2017). 'Inland ship stern optimization in shallow water'. In: *Ocean Engineering* 141, pp. 555–569.

Shiba, Hisamitsu (1953). 'Air-drawing of marine propellers'. In: *Report of transportation technical research institute* 9.

Tennekes, Hendrik, John Leask Lumley, Jonh L Lumley et al. (1972). *A first course in turbulence*. MIT press.

Tian, Xinliang et al. (2013). 'Unsteady RANS simulations of flow around rectangular cylinders with different aspect ratios'. In: *Ocean Engineering* 58, pp. 208–216.

Ueda, H and Julius Oscar Hinze (1975). 'Fine-structure turbulence in the wall region of a turbulent boundary layer'. In: *Journal of Fluid Mechanics* 67.1, pp. 125–143.

Vartdal, Leif and Rune Garen (2001). 'A thruster system which improves positioning power by reducing interaction losses'. In: *Dynamic Positioning Conference, Houston, TX, USA*.

Von Kármán, Theodore (1931). *Mechanical similitude and turbulence*. 611. National Advisory Committee for Aeronautics.

Wazzan, Ahmed R, C Gazley Jr and Apollo Milton Olin Smith (1981). 'HR/x/-method for predicting transition'. In: *AIAA Journal* 19.6, pp. 810–812.

White, Frank M and Joseph Majdalani (2006). *Viscous fluid flow*. Vol. 3. McGraw-Hill New York. Chap. 6-9, pp. 473–479.

Wilcox, David C (1988). 'Reassessment of the scale-determining equation for advanced turbulence models'. In: *AIAA journal* 26.11, pp. 1299–1310.

Wygnanski, Israel and Ho Fiedler (1969). 'Some measurements in the self-preserving jet'. In: *Journal of Fluid Mechanics* 38.3, pp. 577–612.

APPENDIX

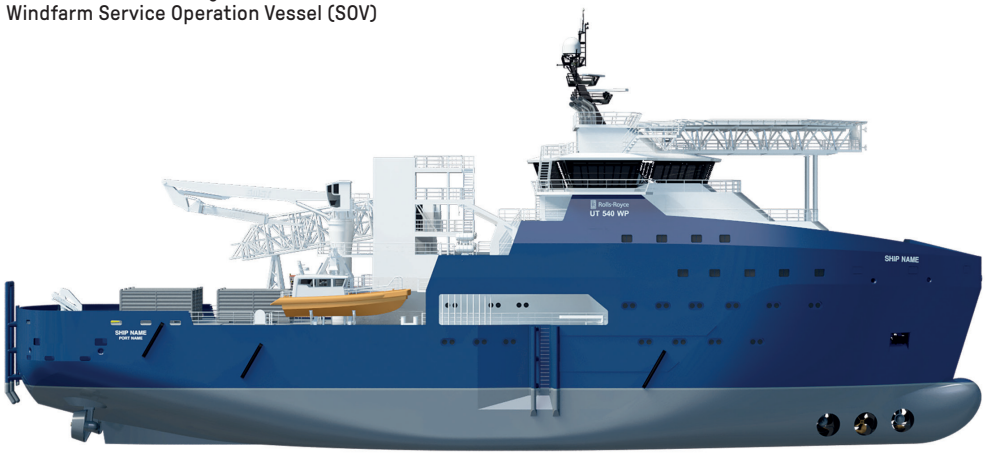
A

KONGSBERG PRODUCTS USED FOR COMPARISON

Table A.1: Kongsberg US TYPE AZIMUTHING THRUSTER

US Thruster Type	Propeller Dia. Ducted [mm]	Max. power. [kW]	Bollard Pull [mtons] with 2 x AZM, max Power
55	1050	350	11
105P6	1300	500	17
105P9	1500	750	24
	1600	750	25
155P12	1600	900	29
	1800	1100	35
155P14	1800	1150	37
	2000	1370	45
165	2200	1471	50
	2400	1340	50
205	2300	1870	60
	2400	2000	65
	2500	2000	67
	2800	2000	70
255	2600	2390	77
	2700	2560	81
	2800	2560	85
	3000	2600	91
265	3000	2800	99
35	2800	2790	91
	3000	2900	97
305	3000	3200	104
	3200	3300	111
355	3200	3600	117
	3500	4050	135
	3600	4050	140
60	3800	5500	174
	4000	5500	180

UT 5400 WP vessel range
Windfarm Service Operation Vessel (SOV)



MAIN DIMENSIONS

Length overall	82.0 m
Length between p.p.	72.1 m
Breadth moulded	17.0-18.0 m
Depth main deck	7.4 m
Design draught	5.0 m

MANOEUVRABILITY ENHANCEMENT

- Diesel-electric with four variable RPM main generator sets
- Twin azimuthing thrusters aft
- Three (super-silent) tunnel thrusters forward
- Optional: Retractable azimuthing thruster replacing 3rd tunnel thruster

CAPACITIES (PRELIMINARY)

Deadweight	1500 t
Fuel oil	890 m^3
Fresh water	500 m^3
Cargo store (Main deck)	400 m^2
Cargo deck (A-deck)	340 m^2

OTHER PARTICULARS

- One 9-1 1m daughter craft/workboat
- One fast rescue craft
- Cargo and personell lift
- Deck crane 3 t at 16 m
- 3D-motion compensated crane (1 t at 25 m)
- Two CTV landing areas
- Optional helicopter landing deck

PERFORMANCE

Max. speed	13.5 kn
Service speed	10-11 kn

ACCOMMODATION

- Up to 60 person accommodation, (40 technicians and 20 crew)
- Optional above 60 persons
- Comfort Class V(2)C(2)

Kongsberg Maritime
P.O.Box 483, NO-3601
Kongsberg, Norway

Switchboard: +47 815 73 700
Global support 24/7: +47 33 03 24 07
E-mail sales: km.sales@km.kongsberg.com
E-mail support: km.support@km.kongsberg.com

01.UT-2 of 2-03.09.16 V.1

Figure A.1: Example vessel used to create generic hull shape

APPENDIX

B

FIGURES PRODUCED IN PARAVIEW

Figure B.1: Instantaneous snapshot of the velocity field from pimpleFoam transient simulation, $\mathfrak{Re} = 10^7$, $\frac{W_D}{T} = 3$ and $t = 100$s

Figure B.2: Instantaneous snapshot of the velocity field from pimpleFoam transient simulation, $\mathfrak{Re} = 10^5$, $\frac{W_D}{T} = 3$ and $t = 100$s

Figure B.3: Instantaneous snapshot of the velocity field from pimpleFoam transient simulation, $\mathfrak{Re} = 10^5$, $\frac{W_D}{T} = 4$ and $t = 100$s

Figure B.4: Snapshot of the steady state velocity field from the simpleFoam simulation, $\mathfrak{Re} = 10^5$, $\frac{W_D}{T} = 3$ and $t = 100$s

Figure B.5: Snapshot of the steady state velocity field from the simpleFoam simulation, $\mathfrak{Re} = 10^7$, $\frac{W_D}{T} = 3$ and $t = 100$s

Figure B.6: Snapshot of the steady state velocity field from the simpleFoam simulation, $\mathfrak{Re} = 10^7$, $\frac{W_D}{T} = 4$ and $t = 100$s

APPENDIX

C

CODES USED IN OPENFOAM

C.1 Equal for both solvers the 0 directory

```
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0.0006;

boundaryField
{
    frontAndBack
    {
        type            empty;
    }
    top
    {
        type            zeroGradient;
```

```
    }
    inlet
    {
        type            fixedValue;
        value           uniform 0.0006;
    }
    hull
    {
        type            kqRWallFunction;
        value           uniform 0.0006;
    }
    propTopAndBottom
    {
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    propInlet
    {
        type            zeroGradient;
    }
    propOutlet
    {
        type            zeroGradient;
    }
    bottom
    {
        type            kqRWallFunction;
        value           uniform 0.0006;
    }
}


// ************************************************************************* //
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 1.25e-07;

boundaryField
{
    frontAndBack
    {
        type            empty;
    }
    top
    {
        type            zeroGradient;
    }
    inlet
    {
        type            calculated;
        value           uniform 1.25e-07;
    }
    hull
    {
        type            nutUSpaldingWallFunction;
        value           uniform 1.25e-07;
    }
    propTopAndBottom
    {
        type            zeroGradient;
    }
    outlet
    {
        type            calculated;
        value           uniform 1.25e-07;
    }
    propInlet
    {
        type            zeroGradient;
    }
    propOutlet
    {
        type            zeroGradient;
    }
    bottom
    {
        type            nutUSpaldingWallFunction;
        value           uniform 1.25e-07;
    }
}
```

```
56 // ************************************************************************* //

1  FoamFile
2  {
3      version     2.0;
4      format      ascii;
5      arch        "LSB;label=32;scalar=64";
6      class       volScalarField;
7      location    "0";
8      object      omega;
9  }
10 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

11 dimensions      [0 0 -1 0 0 0 0];

12 internalField   uniform 2.5555063;

13 boundaryField
14 {
15     frontAndBack
16     {
17         type            empty;
18     }
19     top
20     {
21         type            zeroGradient;
22     }
23     inlet
24     {
25         type            fixedValue;
26         value           uniform 2.5555063;
27     }
28     hull
29     {
30         type            omegaWallFunction;
31         blended         true;
32         value           uniform 2.5555063;
33     }
34     propTopAndBottom
35     {
36         type            zeroGradient;
37     }
38     outlet
39     {
40         type            zeroGradient;
41     }
42     propInlet
```

```
    {
        type            zeroGradient;
    }
    propOutlet
    {
        type            zeroGradient;
    }
    bottom
    {
        type            omegaWallFunction;
        blended         true;
        value           uniform 2.5555063;
    }
}


// ************************************************************************* //

FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    frontAndBack
    {
        type            empty;
    }
    top
    {
        type            zeroGradient;
    }
    inlet
    {
        type            zeroGradient;
    }
    hull
```

```
    {
        type            zeroGradient;
    }
    propTopAndBottom
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    propInlet
    {
        type            zeroGradient;
    }
    propOutlet
    {
        type            zeroGradient;
    }
    bottom
    {
        type            zeroGradient;
    }
}


// ************************************************************************* //
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    frontAndBack
    {
```

```
        type                empty;
    }
    top
    {
        type                zeroGradient;
    }
    inlet
    {
        type                fixedValue;
        value               uniform (0 1 0);
    }
    hull
    {
        type                noSlip;
    }
    propTopAndBottom
    {
        type                slip;
    }
    outlet
    {
        type                zeroGradient;
    }
    propInlet
    {
        type                flowRateInletVelocity;
        volumetricFlowRate  constant -0.001227;
        value               uniform (0 0 0);
    }
    propOutlet
    {
        type                flowRateInletVelocity;
        volumetricFlowRate  constant 0.001227;
    }
    bottom
    {
        type                noSlip;
    }
}


// ************************************************************************* //
```

C.2 The constant directory

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              6.802e-08;
rho             1025;


// ************************************************************************* //
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType      RAS;

RAS
{
    RASModel        kOmegaSST;

    turbulence      on;

    printCoeffs     on;
}


// ************************************************************************* //
```

C.3 The system directory

C.3.1 Used in pimpleFOAM

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pimpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         50;

deltaT          1e-4;

writeControl    adjustableRunTime;

writeInterval   0.5;

purgeWrite      0;

writeFormat     ascii;

writePrecision  7;

writeCompression no;

timeFormat      general;

timePrecision   6;

runTimeModifiable no;

adjustTimeStep  no;

maxCo           1;
```

```
maxDeltaT       1;

functions
{
    // #include "relVelocity"
    forceCoeffs1
    {
        // ------------------
        // Mandatory Entries
        // ------------------
        type        forceCoeffs;
        libs        ( "libforces.so" );
        patches     ( hull );
        // ----------------
        // Optional Entries
        // ----------------
        // Field names
        pName           p;
        Uname           U;
        rho             rhoInf;
        rhoInf          1000;
        // Reference pressure [Pa]
        pRef            0;
        // Porosity effects
        porosity        no;
        writeControl    adjustableRunTime;
        writeInterval   0.5;
        // Freestream velocity magnitude [m/s]
        magUInf         1.00;
        log             true;
        // Lift direction
        liftDir         (0 0 1);
        // Drag direction
        dragDir         (0 1 0);
        // Centre of rotation for moment calculations
        CofR            (0 0 0);
        // Pitch axis
        pitchAxis       (0 1 0);
        // Reference length [m]
        lRef            0.25;
        // Reference area [m²]
        Aref            0.025;
    }
}

// ************************************************************************* //
```

```
FoamFile
```

```
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 10;

method          scotch;

scotchCoeffs
{
        // processorWeights
        //(
        //        1
        //        1
        //        1
        //        1
        //)
        //writeGraph        true;
        //strategy        "b";
}

// ************************************************************************* //
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
```

```
    default         none;

    div(phi,U)      Gauss linearUpwind grad(U);
    div(U)          Gauss linear;

    div(phi,k)      Gauss linearUpwind grad(U);
    div(phi,omega)  Gauss linearUpwind grad(U);

    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method          meshWave;
}


// ************************************************************************* //
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "(p|pcorr)"
    {
        solver          GAMG;
```

```
        smoother        DICGaussSeidel;
        tolerance       1e-06;
        relTol          0.1;
    }

    "(p|pcorr)Final"
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|omega)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0.1;
    }

    "(U|k|omega)Final"
    {
        $U;
        tolerance       1e-06;
        relTol          0;
    }
}

PIMPLE
{
    momentumPredictor        yes;
    nOuterCcorrectors        3;
    nCorrectors              1;
    nNonOrthogonalCorrectors 0;
}


// ************************************************************************* //
```

C.3.2 Used in `simpleFOAM`

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
```

```
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     simpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         100;

deltaT          1e-4;

writeControl    adjustableRunTime;

writeInterval   0.5;

purgeWrite      0;

writeFormat     ascii;

writePrecision  7;

writeCompression no;

timeFormat      general;

timePrecision   6;

runTimeModifiable no;

adjustTimeStep  no;

maxCo           1;

maxDeltaT       1;

functions
{
    // #include "relVelocity"
    forceCoeffs1
    {
        // ------------------
        // Mandatory Entries
        // ------------------
        type       forceCoeffs;
```

```
        libs        ( "libforces.so" );
        patches     ( hull );
        // ----------------
        // Optional Entries
        // ----------------
        // Field names
        pName           p;
        Uname           U;
        rho             rhoInf;
        rhoInf          1000;
        // Reference pressure [Pa]
        pRef            0;
        // Porosity effects
        porosity        no;
        writeControl    adjustableRunTime;
        writeInterval   0.5;
        // Freestream velocity magnitude [m/s]
        magUInf         1.00;
        log             true;
        // Lift direction
        liftDir         (0 0 1);
        // Drag direction
        dragDir         (0 1 0);
        // Centre of rotation for moment calculations
        CofR            (0 0 0);
        // Pitch axis
        pitchAxis       (0 1 0);
        // Reference length [m]
        lRef            0.25;
        // Reference area [m²]
        Aref            0.025;
    }
}


// ************************************************************************* //
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 10;
```

```
method          scotch;

scotchCoeffs
{
        // processorWeights
        //(
        //      1
        //      1
        //      1
        //      1
        //)
        //writeGraph        true;
        //strategy        "b";
}

// ************************************************************************* //
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;

    div(phi,U)      bounded Gauss linearUpwindV grad(U);
    div(U)          Gauss linear;

    div(phi,k)      bounded Gauss upwind;
    div(phi,omega)  bounded Gauss upwind;

    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}
```

```
laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method          meshWave;
}


// ************************************************************************* //
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "(p|pcorr)"
    {
        solver          GAMG;
        smoother        DICGaussSeidel;
        tolerance       1e-06;
        relTol          0.1;
    }

    "(p|pcorr)Final"
    {
        $p;
        tolerance       1e-06;
        relTol          0;
```

```
    }

    "(U|k|omega)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0.1;
    }

    "(U|k|omega)Final"
    {
        $U;
        tolerance       1e-06;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors    0;
    consistent                  yes;
}
relaxationFactors
{
    equations
    {
        U       0.9;
        k       0.7;
        omega   0.7;
    }
}
cache
{
    grad(U);
}

// ************************************************************************* //
```

APPENDIX

D

CODES USED IN GMSH

D.1 Mesh of 2D hull with propeller

```
//+-------------------------------------------------------------------
// Helpfull function for deciding mesh size
//+-------------------------------------------------------------------

// a1: mesh size of the first cell
// aN: mesh size of the last cell
Macro CalExpansionRatio
   NumGrid = Log(1 - Sn * (1-q) / a0) / Log(q);
   NumGrid = Round(NumGrid) + 1;
   aN      = a0 * q^(NumGrid - 1);
   NumNode = NumGrid;
Return

//+-------------------------------------------------------------------
//  GMSH project created by Brage Møller-Pettersen
//+-------------------------------------------------------------------

//+-------------------------------------------------------------------
// Original vessel used as a mockup model:
// UT 5400 WP vessel range Windfarm Service Operation Vessel (SOV)
//+-------------------------------------------------------------------
LoA                 = 82.0;  // [m]
L_pp                = 72.1;  // [m]
Breadth             = 18.0;  // [m]
```

```
Draft               =  5.0;  // [m]
thrusterDiameter    =  2.5;  // [m]
currentVelocity     =  1.0;  // [m/s]
turbulenceIntensity =  0.02; // [%]
scaleRatio          = 1/20;  // [-]
//------------------------------------------------------------------

////////////////////////////////////
// Coefficients Deciding Mesh Shape
////////////////////////////////////
//+
// DoT = Depth/Draft
DoT = 1.85;
// Radii of curvature in hull corners
Radii = 3; // [m] // Must be smaller than draft
k = Radii * scaleRatio;

////////////////////////////
// Outer points of the hull
////////////////////////////
//+
yh = Breadth * 0.5 * scaleRatio;
zh = Draft * scaleRatio;

/////////////
// Waterdepth
/////////////
//+
d          = DoT*zh;
clearance  = d - zh;
pDia       = thrusterDiameter * scaleRatio;
outerCoeff = clearance * 0.9;

//////////////////////////////
// z coordinates from the top
//////////////////////////////
//+
z1   = zh - k;
zOut = zh + outerCoeff;

////////////////////////////
// y coordinates from inlet
////////////////////////////
//+
ySt  = yh * 4;
yEn  = yh * 4.5;
yOut = yh + outerCoeff;
y1   = yh - k;
```

```
65  ////////////////////////////////////
66  // y & z coordinates for circular arcs
67  ////////////////////////////////////
68  //+
69  Oy   = y1;
70  Oz   = z1;
71  rh   = yh - Oy;
72  rOut = yOut - Oy;

73  ////////////////////////////////////////////////////////////
74  // y & z coordinates for the diagonal line intersecting arcs
75  ////////////////////////////////////////////////////////////
76  //+
77  yd1 = Oy + rh * Sqrt(2) * 0.5;    zd1 = Oz + rh * Sqrt(2) * 0.5;
78  yd3 = Oy + rOut * Sqrt(2) * 0.5;  zd3 = Oz + rOut * Sqrt(2) * 0.5;
79  yd4 = Oy + (d-z1) * Tan(Pi/4);    zd4 = d;

80  ////////////////////////////////////
81  // y & z coordinates for the propeller
82  ////////////////////////////////////
83  //+
84  pRad      = pDia * 0.5;
85  pCyl      = pDia * 0.2;
86  pDistHull = pDia * 0.25;

87  pY        = pCyl * 0.5;
88  pZtop     = zh + pDistHull;
89  pZbottom  = pZtop + pDia;

90  ////////////////////////////////////////////////
91  // y & z coordinates for the propeller boundaries
92  ////////////////////////////////////////////////
93  //+
94  ypT1 = Oy + (pZtop-z1) * Sqrt(2) * 0.5;     zpT1 = Oz + (pZtop-z1) *
    ↪  Sqrt(2) * 0.5;
95  ypT2 = yh + pDistHull;

96  ypB1 = Oy + (pZbottom-z1) * Sqrt(2) * 0.5;  zpB1 = Oz + (pZbottom-z1)
    ↪  * Sqrt(2) * 0.5;
97  ypB2 = yh + pDistHull + pDia;

98  ////////////////////////////////
99  // The Length Of Each Meshing Line
100 ////////////////////////////////
101 //+
102 L_vec = {
103     ySt - yd3,                      //  0
```

```
    zOut - pZbottom,                    //  1
    pZbottom - pZtop,                   //  2
    pZtop - zh,                         //  3
    yEn - yOut,                         //  4
    z1,                                 //  5
    Pi * 2 * k * 0.125,                 //  6
    Pi * 2 * k * 0.125,                 //  7
    y1 - pY,                            //  8
    pY * 2,                             //  9
    y1 - pY,                            // 10
    Pi*2*k*0.125,                       // 11
    Pi*2*k*0.125,                       // 12
    z1,                                 // 13
    d - zd3                             // 14
};

//////////////////////////////////////////
// Calculate Appropriate Number Of Elements
//////////////////////////////////////////
//+

////////////////////////////////////////////////
//  Indexes:                                  //
//  0 = surface from inlet to outer           //
//  1 = outer subdomain                       //
//  2 = propeller                             //
//  3 = inner subdomain                       //
//  4 = surface from outlet to outer          //
//  5 = inlet to top of hull                  //
//  6 = inlet to corner of hull               //
//  7 = bottom to corner of hull, inlet side  //
//  8 = bottom to hull, inlet side            //
//  9 = bottom to hull, propeller             //
// 10 = bottom to hull, outlet side           //
// 11 = bottom to corner of hull, outlet side //
// 12 = outlet to corner of hull              //
// 13 = outlet to top of hull                 //
// 14 = inlet to outlet                       //
////////////////////////////////////////////////

// Define the smallest cell side length:

meshSizes = {       // nCells:      id:
        6.4e-3,     //    9966       0
        3.55e-3,    //   24965       1
        2.25e-3,    //   49467       2
        1.71e-3,    //   75149       3
        1.4e-3,     //  100366       4
```

```
        1.21e-3,    //  124897      5
        1.06e-3,    //  150130      6
        9.5e-4,     //  175046      7
        8.7e-4,     //  199205      8
        7.99e-4,    //  224759      9
        7.4e-4,     //  249700     10
        4.45e-4,    //  499418     11
        3.27e-4,    //  749689     12
        2.625e-4    // 1000411     13
};

fine       = meshSizes[2];
semiFine   = fine * 1.25;
coarseFine = fine * 1.5;
coarse     = fine * 2;
veryCoarse = fine * 12;

/*  Suggestion
    fine       = meshSizes[];
    semiFine   = fine * 1.1;
    coarseFine = semiFine * 1.1;
    coarse     = coarseFine * 1.1;
    veryCoarse = coarse * 1.1;
*/

cellMin = {
    veryCoarse,   //  0              // Matches largest cell in 1
    semiFine,     //  1              // Matches largest cell in 2
    semiFine,     //  2              // Matches largest cell in 3
    fine,         //  3
    semiFine,     //  4              // Matches largest cell in 1
    fine,         //  5
    fine,         //  6
    fine,         //  7
    fine,         //  8
    fine,         //  9
    fine,         // 10
    fine,         // 11
    fine,         // 12
    coarseFine,   // 13
    fine          // 14
};
// Define expansion ratios
r_exp = {
    1.1,          //  0
    1.0125,       //  1
    1.0125,       //  2
    1.0125,       //  3
```

```
    1.1,          //  4
    1.025,        //  5
    1,            //  6
    1,            //  7
    1,            //  8
    1,            //  9
    1,            // 10
    1,            // 11
    1,            // 12
    1.025,        // 13
    1             // 14
};
// Preallocate vectors for largest grid cells and number of nodes
cellMax = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
nCells = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

//  3 = inner subdomain
Sn          = L_vec[3];
q           = r_exp[3];
a0          = cellMin[3];
Call CalExpansionRatio;
cellMax[3] = aN;
nCells[3]  = NumNode;

//  2 = propeller
Sn          = L_vec[2];
q           = r_exp[2];
a0          = cellMax[3];  cellMin[2] = cellMax[3];
Call CalExpansionRatio;
cellMax[2] = aN;
nCells[2]  = NumNode;

//  1 = outer subdomain
Sn          = L_vec[1];
q           = r_exp[1];
a0          = cellMax[2];  cellMin[1] = cellMax[2];
Call CalExpansionRatio;
cellMax[1] = aN;
nCells[1]  = NumNode;

//  0 = surface from inlet to outer
Sn          = L_vec[0];
q           = r_exp[0];
a0          = cellMax[1]*r_exp[0];  cellMin[0] = cellMax[1]*r_exp[0];
Call CalExpansionRatio;
cellMax[0] = aN;
nCells[0]  = NumNode;
```

```
236 //  4 = surface from outlet to outer
237 Sn          = L_vec[4];
238 q           = r_exp[4];
239 a0          = cellMax[1];  cellMin[4] = cellMax[1];
240 Call CalExpansionRatio;
241 cellMax[4] = aN;
242 nCells[4]  = NumNode;

243 //  5 = inlet to top of hull
244 Sn          = L_vec[5];
245 q           = r_exp[5];
246 a0          = cellMin[5];
247 Call CalExpansionRatio;
248 cellMax[5] = aN;
249 nCells[5]  = NumNode;

250 //  6 = inlet to corner of hull
251 cellMax[6] =  cellMin[6];
252 nCells[6]  = Round(L_vec[6]/cellMin[6]);

253 //  7 = bottom to corner of hull, inlet side
254 cellMax[7] =  cellMin[7];
255 nCells[7]  = Round(L_vec[7]/cellMin[7]);

256 //  8 = bottom to hull        , inlet side
257 cellMax[8] =  cellMin[8];
258 nCells[8]  = Round(L_vec[8]/cellMin[8]);

259 //  9 = bottom to hull, propeller
260 cellMax[9] =  cellMin[9];
261 nCells[9]  = Round(L_vec[9]/cellMin[9]);

262 // 10 = bottom to hull, outlet side
263 cellMax[10] =  cellMin[10];
264 nCells[10]  = Round(L_vec[10]/cellMin[10]);

265 // 11 = bottom to corner of hull, outlet side
266 cellMax[11] =  cellMin[11];
267 nCells[11]  = Round(L_vec[11]/cellMin[11]);

268 // 12 = outlet to corner of hull
269 cellMax[12] =  cellMin[12];
270 nCells[12]  = Round(L_vec[12]/cellMin[12]);

271 // 13 = outlet to top of hull
272 Sn          = L_vec[13];
273 q           = r_exp[13];
274 a0          = cellMin[13];
```

```
Call CalExpansionRatio;
cellMax[13] = aN;
nCells[13]  = NumNode;

// 14 = inlet to outlet
cellMin[14] = cellMax[1];
cellMax[14] =  cellMin[14];
nCells[14]  = Round(L_vec[14]/cellMin[14]);

/*  Suggestion
Sn          = L_vec[14];
q           = r_exp[14];
a0          = cellMax[1];  cellMin[14] = cellMax[1];
Call CalExpansionRatio;
cellMax[14] = aN;
nCells[14]  = NumNode;
*/

//+-------------------------------------------------------------------
// Points
//+-------------------------------------------------------------------
//+
Point(1) = {0, -yh, 0, 1.0};
//+
Point(2) = {0, -yh, -z1, 1.0};
//+
Point(3) = {0, -y1, -zh, 1.0};
//+
Point(4) = {0, y1, -zh, 1.0};
//+
Point(5) = {0, yh, -z1, 1.0};
//+
Point(6) = {0, yh, 0, 1.0};

// Second surrounding mesh structure
//+
Point(7)  = {0, -yOut, 0, 1.0};
//+
Point(8)  = {0, -yOut, -z1, 1.0};
//+
Point(9)  = {0, -y1, -zOut, 1.0};
//+
Point(10) = {0, y1, -zOut, 1.0};
//+
Point(11) = {0, yOut, -z1, 1.0};
//+
Point(12) = {0, yOut, 0, 1.0};
```

```
// Outer borders for the domain
//+
Point(13) = {0, -ySt, 0, 1.0};
//+
Point(14) = {0, -ySt, -z1, 1.0};
//+
Point(15) = {0, -ySt, -d, 1.0};
//+
Point(16) = {0, yEn, -d, 1.0};
//+
Point(17) = {0, yEn, -z1, 1.0};
//+
Point(18) = {0, yEn, 0, 1.0};
//+
Point(19) = {0, -y1, -d, 1.0};
//+
Point(20) = {0, y1, -d, 1.0};

// origo for circular arcs
//+
Point(21) = {0, -Oy, -Oz, 1.0};
//+
Point(22) = {0, Oy, -Oz, 1.0};

// Points of intersection with diagonal and arcs
//+
Point(23) = {0, -yd1, -zd1, 1.0};
//+
Point(24) = {0, yd1, -zd1, 1.0};
//+
Point(25) = {0, -yd3, -zd3, 1.0};
//+
Point(26) = {0, yd3, -zd3, 1.0};
//+
Point(27) = {0, -yd4, -zd4, 1.0};
//+
Point(28) = {0, yd4, -zd4, 1.0};
//+
// Horizontal support for diagonals
Point(29) = {0, -ySt, -zd3, 1.0};
//+
Point(30) = {0, yEn, -zd3, 1.0};
//+
// Propeller
Point(31) = {0, -pY, -pZtop, 1.0};
//+
Point(32) = {0, pY, -pZtop, 1.0};
//+
```

```
Point(33) = {0, pY, -pZbottom, 1.0};
//+
Point(34) = {0, -pY, -pZbottom, 1.0};
//+
// Propeller boundary points
Point(35) = {0, -y1, -pZtop, 1.0};
Point(36) = {0, -ypT1, -zpT1, 1.0};
Point(37) = {0, -ypT2, -z1, 1.0};
Point(38) = {0, -ypT2, 0, 1.0};
//+
Point(39) = {0, y1, -pZtop, 1.0};
Point(40) = {0, ypT1, -zpT1, 1.0};
Point(41) = {0, ypT2, -z1, 1.0};
Point(42) = {0, ypT2, 0, 1.0};
//+
Point(43) = {0, -y1, -pZbottom, 1.0};
Point(44) = {0, -ypB1, -zpB1, 1.0};
Point(45) = {0, -ypB2, -z1, 1.0};
Point(46) = {0, -ypB2, 0, 1.0};
//+
Point(47) = {0, y1, -pZbottom, 1.0};
Point(48) = {0, ypB1, -zpB1, 1.0};
Point(49) = {0, ypB2, -z1, 1.0};
Point(50) = {0, ypB2, 0, 1.0};
//+
Point(51) = {0, -pY, -zh, 1.0};
Point(52) = {0, -pY, -zOut, 1.0};
Point(53) = {0, -pY, -d, 1.0};
//+
Point(54) = {0, pY, -zh, 1.0};
Point(55) = {0, pY, -zOut, 1.0};
Point(56) = {0, pY, -d, 1.0};

//+-----------------------------------------------------------------
// Lines
//+-----------------------------------------------------------------
//+
Line(1)    = {2, 1};
//+
Circle(2)  = {23, 21, 2};
//+
Circle(3)  = {3, 21, 23};
//+
Line(4)    = {51, 3};
//+
Line(5)    = {51, 54};
//+
Line(6)    = {54, 4};
```

```
//+
Circle(7)  = {4, 22, 24};
//+
Circle(8)  = {24, 22, 5};
//+
Line(9)    = {5, 6};
//+
Line(10)   = {37, 38};
//+
Circle(11) = {36, 21, 37};
//+
Circle(12) = {35, 21, 36};
//+
Line(13)   = {31, 35};
//+
Line(14)   = {31, 32};
//+
Line(15)   = {32, 39};
//+
Circle(16) = {39, 22, 40};
//+
Circle(17) = {40, 22, 41};
//+
Line(18)   = {41, 42};
//+
Line(19)   = {45, 46};
//+
Circle(20) = {44, 21, 45};
//+
Circle(21) = {43, 21, 44};
//+
Line(22)   = {34, 43};
//+
Line(23)   = {34, 33};
//+
Line(24)   = {33, 47};
//+
Circle(25) = {47, 22, 48};
//+
Circle(26) = {48, 22, 49};
//+
Line(27)   = {49, 50};
//+
Line(28)   = {8, 7};
//+
Circle(29) = {25, 21, 8};
//+
Circle(30) = {9, 21, 25};
```

```
//+
Line(31)   = {52, 9};
//+
Line(32)   = {52, 55};
//+
Line(33)   = {55, 10};
//+
Circle(34) = {10, 22, 26};
//+
Circle(35) = {26, 22, 11};
//+
Line(36)   = {11, 12};
//+
Line(37)   = {14, 13};
//+
Line(38)   = {29, 14};
//+
Line(39)   = {29, 15};
//+
Line(40)   = {27, 15};
//+
Line(41)   = {19, 27};
//+
Line(42)   = {53, 19};
//+
Line(43)   = {53, 56};
//+
Line(44)   = {56, 20};
//+
Line(45)   = {20, 28};
//+
Line(46)   = {28, 16};
//+
Line(47)   = {30, 16};
//+
Line(48)   = {30, 17};
//+
Line(49)   = {17, 18};
//+
Line(50)   = {1, 38};
//+
Line(51)   = {2, 37};
//+
Line(52)   = {23, 36};
//+
Line(53)   = {3, 35};
//+
Line(54)   = {51, 31};
```

```
//+
Line(55)   = {54, 32};
//+
Line(56)   = {4, 39};
//+
Line(57)   = {24, 40};
//+
Line(58)   = {5, 41};
//+
Line(59)   = {6, 42};
//+
Line(60)   = {38, 46};
//+
Line(61)   = {37, 45};
//+
Line(62)   = {36, 44};
//+
Line(63)   = {35, 43};
//+
Line(64)   = {31, 34};
//+
Line(65)   = {32, 33};
//+
Line(66)   = {39, 47};
//+
Line(67)   = {40, 48};
//+
Line(68)   = {41, 49};
//+
Line(69)   = {42, 50};
//+
Line(70)   = {46, 7};
//+
Line(71)   = {45, 8};
//+
Line(72)   = {44, 25};
//+
Line(73)   = {43, 9};
//+
Line(74)   = {34, 52};
//+
Line(75)   = {33, 55};
//+
Line(76)   = {47, 10};
//+
Line(77)   = {48, 26};
//+
Line(78)   = {49, 11};
```

```
//+
Line(79)   = {50, 12};
//+
Line(80)   = {7, 13};
//+
Line(81)   = {8, 14};
//+
Line(82)   = {25, 29};
//+
Line(83)   = {25, 27};
//+
Line(84)   = {9, 19};
//+
Line(85)   = {52, 53};
//+
Line(86)   = {55, 56};
//+
Line(87)   = {10, 20};
//+
Line(88)   = {26, 28};
//+
Line(89)   = {26, 30};
//+
Line(90)   = {11, 17};
//+
Line(91)   = {12, 18};

////////////////////////////////////////
// Set number of elements along the lines
////////////////////////////////////////
//+
//  0 = surface from inlet to outer
Transfinite Curve {80, 81, 82, 40} = nCells[0]-1 Using Progression
    ↪  r_exp[0];
//+
//  1 = outer subdomain
Transfinite Curve {70, 71, 72, 73, 74, 75, 76, 77, 78, 79} =
    ↪  nCells[1] Using Progression r_exp[1];
//+
//  2 = propeller
Transfinite Curve {60, 61, 62, 63, 64, 65, 66, 67, 68, 69} =
    ↪  nCells[2]-1 Using Progression r_exp[2];
//+
//  3 = inner subdomain
Transfinite Curve {50, 51, 52, 53, 54, 55, 56, 57, 58, 59} =
    ↪  nCells[3]-1 Using Progression r_exp[3];
//+
//  4 = surface from outlet to outer
```

```
598 Transfinite Curve {91, 90, 89, 46} = nCells[4]-1 Using Progression
    ↪  r_exp[4];
599 //+
600 //  5 = inlet to top of hull
601 Transfinite Curve {1, 10, 19, 28, 37} = nCells[5]-1 Using Progression
    ↪  r_exp[5];
602 //+
603 //  6 = inlet to corner of hull
604 Transfinite Curve {2, 11, 20, 29, 38} = nCells[6] Using Progression
    ↪  r_exp[6];
605 //+
606 //  7 = bottom to corner of hull, inlet side
607 Transfinite Curve {3, 12, 21, 30, 41} = nCells[7] Using Progression
    ↪  r_exp[7];
608 //+
609 //  8 = bottom to hull        , inlet side
610 Transfinite Curve {4, 13, 22, 31, 42} = nCells[8] Using Progression
    ↪  r_exp[8];
611 //+
612 //  9 = bottom to hull, propeller
613 Transfinite Curve {5, 14, 23, 32, 43} = nCells[9] Using Progression
    ↪  r_exp[9];
614 //+
615 // 10 = bottom to hull, outlet side
616 Transfinite Curve {6, 15, 24, 33, 44} = nCells[10] Using Progression
    ↪  r_exp[10];
617 //+
618 // 11 = bottom to corner of hull, outlet side
619 Transfinite Curve {7, 16, 25, 34, 45} = nCells[11] Using Progression
    ↪  r_exp[11];
620 //+
621 // 12 = outlet to corner of hull
622 Transfinite Curve {8, 17, 26, 35, 48} = nCells[12] Using Progression
    ↪  r_exp[12];
623 //+
624 // 13 = outlet to top of hull
625 Transfinite Curve {9, 18, 27, 36, 49} = nCells[13]-1 Using
    ↪  Progression r_exp[13];
626 //+
627 // 14 = inlet to outlet
628 Transfinite Curve {47, 88, 87, 86, 85, 84, 83, 39} = nCells[14] Using
    ↪  Progression r_exp[14];

629 //+-----------------------------------------------------------------
630 // Set Surfaces
631 //+-----------------------------------------------------------------
632 //+
633 Curve Loop(1)     = {1, 50, -10, -51};
```

```
Plane Surface(1)  = {1};
Transfinite Surface {1};
Recombine Surface {1};
//+
Curve Loop(2)     = {51, -11, -52, 2};
Plane Surface(2)  = {2};
Transfinite Surface {2};
Recombine Surface {2};
//+
Curve Loop(3)     = {3, 52, -12, -53};
Plane Surface(3)  = {3};
Transfinite Surface {3};
Recombine Surface {3};
//+
Curve Loop(4)     = {4, 53, -13, -54};
Plane Surface(4)  = {4};
Transfinite Surface {4};
Recombine Surface {4};
//+
Curve Loop(5)     = {5, 55, -14, -54};
Plane Surface(5)  = {5};
Transfinite Surface {5};
Recombine Surface {5};
//+
Curve Loop(6)     = {6, 56, -15, -55};
Plane Surface(6)  = {6};
Transfinite Surface {6};
Recombine Surface {6};
//+
Curve Loop(7)     = {7, 57, -16, -56};
Plane Surface(7)  = {7};
Transfinite Surface {7};
Recombine Surface {7};
//+
Curve Loop(8)     = {8, 58, -17, -57};
Plane Surface(8)  = {8};
Transfinite Surface {8};
Recombine Surface {8};
//+
Curve Loop(9)     = {59, -18, -58, 9};
Plane Surface(9)  = {9};
Transfinite Surface {9};
Recombine Surface {9};
//+
Curve Loop(10)    = {69, -27, -68, 18};
Plane Surface(10) = {10};
Transfinite Surface {10};
Recombine Surface {10};
```

```
682 //+
683 Curve Loop(11)    = {60, -19, -61, 10};
684 Plane Surface(11) = {11};
685 Transfinite Surface {11};
686 Recombine Surface {11};
687 //+
688 Curve Loop(12)    = {61, -20, -62, 11};
689 Plane Surface(12) = {12};
690 Transfinite Surface {12};
691 Recombine Surface {12};
692 //+
693 Curve Loop(13)    = {12, 62, -21, -63};
694 Plane Surface(13) = {13};
695 Transfinite Surface {13};
696 Recombine Surface {13};
697 //+
698 Curve Loop(14)    = {13, 63, -22, -64};
699 Plane Surface(14) = {14};
700 Transfinite Surface {14};
701 Recombine Surface {14};
702 //+
703 Curve Loop(15)    = {14, 65, -23, -64};
704 Plane Surface(15) = {15};
705 Transfinite Surface {15};
706 Recombine Surface {15};
707 //+
708 Curve Loop(16)    = {15, 66, -24, -65};
709 Plane Surface(16) = {16};
710 Transfinite Surface {16};
711 Recombine Surface {16};
712 //+
713 Curve Loop(17)    = {16, 67, -25, -66};
714 Plane Surface(17) = {17};
715 Transfinite Surface {17};
716 Recombine Surface {17};
717 //+
718 Curve Loop(18)    = {68, -26, -67, 17};
719 Plane Surface(18) = {18};
720 Transfinite Surface {18};
721 Recombine Surface {18};
722 //+
723 Curve Loop(19)    = {70, -28, -71, 19};
724 Plane Surface(19) = {19};
725 Transfinite Surface {19};
726 Recombine Surface {19};
727 //+
728 Curve Loop(20)    = {71, -29, -72, 20};
729 Plane Surface(20) = {20};
```

```
Transfinite Surface {20};
Recombine Surface {20};
//+
Curve Loop(21)    = {21, 72, -30, -73};
Plane Surface(21) = {21};
Transfinite Surface {21};
Recombine Surface {21};
//+
Curve Loop(22)    = {22, 73, -31, -74};
Plane Surface(22) = {22};
Transfinite Surface {22};
Recombine Surface {22};
//+
Curve Loop(23)    = {23, 75, -32, -74};
Plane Surface(23) = {23};
Transfinite Surface {23};
Recombine Surface {23};
//+
Curve Loop(24)    = {24, 76, -33, -75};
Plane Surface(24) = {24};
Transfinite Surface {24};
Recombine Surface {24};
//+
Curve Loop(25)    = {25, 77, -34, -76};
Plane Surface(25) = {25};
Transfinite Surface {25};
Recombine Surface {25};
//+
Curve Loop(26)    = {78, -35, -77, 26};
Plane Surface(26) = {26};
Transfinite Surface {26};
Recombine Surface {26};
//+
Curve Loop(27)    = {79, -36, -78, 27};
Plane Surface(27) = {27};
Transfinite Surface {27};
Recombine Surface {27};
//+
Curve Loop(28)    = {80, -37, -81, 28};
Plane Surface(28) = {28};
Transfinite Surface {28};
Recombine Surface {28};
//+
Curve Loop(29)    = {81, -38, -82, 29};
Plane Surface(29) = {29};
Transfinite Surface {29};
Recombine Surface {29};
//+
```

```
778 Curve Loop(30)    = {82, 39, -40, -83};
779 Plane Surface(30) = {30};
780 Transfinite Surface {30};
781 Recombine Surface {30};
782 //+
783 Curve Loop(31)    = {30, 83, -41, -84};
784 Plane Surface(31) = {31};
785 Transfinite Surface {31};
786 Recombine Surface {31};
787 //+
788 Curve Loop(32)    = {31, 84, -42, -85};
789 Plane Surface(32) = {32};
790 Transfinite Surface {32};
791 Recombine Surface {32};
792 //+
793 Curve Loop(33)    = {32, 86, -43, -85};
794 Plane Surface(33) = {33};
795 Transfinite Surface {33};
796 Recombine Surface {33};
797 //+
798 Curve Loop(34)    = {33, 87, -44, -86};
799 Plane Surface(34) = {34};
800 Transfinite Surface {34};
801 Recombine Surface {34};
802 //+
803 Curve Loop(35)    = {34, 88, -45, -87};
804 Plane Surface(35) = {35};
805 Transfinite Surface {35};
806 Recombine Surface {35};
807 //+
808 Curve Loop(36)    = {89, 47, -46, -88};
809 Plane Surface(36) = {36};
810 Transfinite Surface {36};
811 Recombine Surface {36};
812 //+
813 Curve Loop(37)    = {90, -48, -89, 35};
814 Plane Surface(37) = {37};
815 Transfinite Surface {37};
816 Recombine Surface {37};
817 //+
818 Curve Loop(38)    = {91, -49, -90, 36};
819 Plane Surface(38) = {38};
820 Transfinite Surface {38};
821 Recombine Surface {38};


822 //+-----------------------------------------------------------------
823 // Extrude Surfaces
```

```
//------------------------------------------------------------------
//+
Extrude {-0.1, 0, 0} {
  Surface{1};   Surface{2};   Surface{3};   Surface{4};
  Surface{5};   Surface{6};   Surface{7};   Surface{8};
  Surface{9};   Surface{10};  Surface{11};  Surface{12};
  Surface{13};  Surface{14};  Surface{15};  Surface{16};
  Surface{17};  Surface{18};  Surface{19};  Surface{20};
  Surface{21};  Surface{22};  Surface{23};  Surface{24};
  Surface{25};  Surface{26};  Surface{27};  Surface{28};
  Surface{29};  Surface{30};  Surface{31};  Surface{32};
  Surface{33};  Surface{34};  Surface{35};  Surface{36};
  Surface{37};  Surface{38};
  Layers {1};
  Recombine;
}

//------------------------------------------------------------------
// Classify Boundaries
//------------------------------------------------------------------
//+
Physical Surface("frontAndBack", 928)      = {
    707,  28, 729,  29, 751,  30,  19, 509,
     20, 531,  21, 553,  31, 773,  12, 355,
     11, 333, 113,   1, 135,   2,   3, 157,
     13, 377,   4, 179, 399,  14,  22, 575,
     32, 795,   5, 201,  15, 421,  23, 597,
     33, 817,   6, 223,  16, 443,  24, 619,
     34, 839, 245,   7,  17, 465,  25, 641,
    861,  35, 267,   8, 487,  18,  26, 663,
     37, 905, 883,  36, 289,   9, 311,  10,
    685, 27, 38, 927
};
//+
Physical Surface("inlet", 929)             = {
    742, 720, 698
};
//+
Physical Surface("outlet", 930)            = {
    874, 896, 918
};
//+
Physical Surface("top", 931)               = {
    914, 672, 298, 276, 104, 320, 496, 694
};
//+
Physical Surface("bottom", 932)            = {
    746, 768, 790, 812, 834, 856, 878
```

```
};
//+
Physical Surface("hull", 933)              = {
    288, 254, 232, 210, 188, 166, 144, 134, 100
};
//+
Physical Surface("propInlet", 934)         = {
    398
};
//+
Physical Surface("propOutlet", 935)        = {
    412
};
//+
Physical Surface("propTopAndBottom", 936) = {
    196, 416
};

//------------------------------------------------------------------------
// Create Volume
//------------------------------------------------------------------------
//+
Physical Volume("internalField", 937) = {
     1,  2,  3,  4,  5,  6,  7,  8,
     9, 10, 11, 12, 13, 14, 16, 17,
    18, 19, 20, 21, 22, 23, 24, 25,
    26, 27, 28, 29, 30, 31, 32, 33,
    34, 35, 36, 37, 38
};

//------------------------------------------------------------------------
// Engage Meshing In GUI
//------------------------------------------------------------------------
//+
Mesh 3;
```

D.2 Mesh of 2D hull

```
//-------------------------------------------//
// Helpfull function for deciding mesh size  //
//-------------------------------------------//

// a1: mesh size of the first cell
// aN: mesh size of the last cell
Macro CalExpansionRatio
   NumGrid = Log(1 - Sn * (1-q) / a0) / Log(q);
```

```
   NumGrid = Round(NumGrid) + 1;
   aN            = a0 * q^(NumGrid - 1);
   NumNode = NumGrid;
Return

//-----------------------------------------------//
//  GMSH project created by Brage Møller-Pettersen  //
//-----------------------------------------------//

//+-------------------------------------------------------------------
//         Original vessel used as a mockup model:
//         UT 5400 WP vessel range Windfarm Service Operation Vessel (SOV)
//+-------------------------------------------------------------------
LoA                 = 82.0;  // [m]
L_pp                = 72.1;  // [m]
Breadth             = 18.0;  // [m]
Draft               = 5.0;   // [m]
thrusterDiameter    = 2.5;   // [m]
currentVelocity     = 1.0;   // [m/s]
turbulenceIntensity = 0.02;  // [%]
scaleRatio          = 1/20;  // [-]
//+-------------------------------------------------------------------

////////////////////////////////////
// Coefficients Deciding Mesh Shape
////////////////////////////////////
//+
// DoT = Depth/Draft
DoT = 3;
// Radii of curvature in hull corners
Radii = 1;                                  // [m]        // Must be smaller than draft
k = Radii * scaleRatio;

////////////////////////////
// Outer points of the hull
////////////////////////////
//+
yh = Breadth * 0.5 * scaleRatio;
zh = Draft * scaleRatio;

/////////////
// Waterdepth
/////////////
//+
d = DoT*zh;                clearance = d - zh;
```

```
pDia = thrusterDiameter * scaleRatio;
outerCoeff = clearance * 0.9;

////////////////////////////
// z coordinates from the top
////////////////////////////
//+
z1 = zh - k;
zInn = zh + clearance*0.3;  zOut = zh + outerCoeff;

////////////////////////
// y coordinates from inlet
////////////////////////
//+
ySt = yh * 4;                                yEn = yh * 4.5;
yInn = yh + clearance*0.3;  yOut = yh + outerCoeff;
y1 = yh - k;

////////////////////////////////////
// y & z coordinates for circular arcs
////////////////////////////////////
//+
Oy = y1;
Oz = z1;
rh = yh - Oy;
rInn = yInn - Oy;
rOut = yOut - Oy;

////////////////////////////////////////////////////////////
// y & z coordinates for the diagonal line intersecting arcs
////////////////////////////////////////////////////////////
//+
yd1 = Oy + rh * Sqrt(2) * 0.5;
zd1 = Oz + rh * Sqrt(2) * 0.5;
yd2 = Oy + rInn * Sqrt(2) * 0.5;
zd2 = Oz + rInn * Sqrt(2) * 0.5;
yd3 = Oy + rOut * Sqrt(2) * 0.5;
zd3 = Oz + rOut * Sqrt(2) * 0.5;
yd4 = Oy + (d-z1) * Tan(Pi/4);
zd4 = d;

////////////////////////////////
// The Length Of Each Meshing Line
////////////////////////////////
//+
L_vec = {
    ySt - yd3,                      //  0
    yOut - yInn,                    //  1
```

```
    yInn - yh,                          //  2
    yEn - yOut,                         //  3
    z1,                                 //  4
    Pi * 2 * k * 0.125,                 //  5
    Pi * 2 * k * 0.125,                 //  6
    2 * y1,                             //  7
    Pi * 2 * k * 0.125,                 //  8
    Pi * 2 * k * 0.125,                 //  9
    z1,                                 // 10
    d - zd3                             // 11
};

//////////////////////////////////////////
// Calculate Appropriate Number Of Elements
//////////////////////////////////////////
//+
////////////////////////////////////////////
//      Indexes:                          //
// 0 = Surface to bottom inlet side       //
// 1 = Outer subdomain                    //
// 2 = Inner subdomain                    //
// 3 = Surface to bottom outlet side      //
// 4 = Inlet to top of hull               //
// 5 = Inlet to corner of hull            //
// 6 = bottom to corner of hull inlet side //
// 7 = bottom to hull                     //
// 8 = bottom to corner of hull outlet side //
// 9 = Outlet to corner of hull           //
// 10 = Outlet to top of hull             //
// 11 = Inlet to outlet                   //
////////////////////////////////////////////

// Define the smallest cell side length:

meshSizes = {       // nCells:      id:
        6.04e-3,    //   10076       0
        3.4e-3,     //   24943       1
        2.15e-3,    //   50399       2
        1.64e-3,    //   75485       3
        1.36e-3,    //   99550       4
        1.17e-3,    //  124598       5
        1.03e-3,    //  149216       6
        9.23e-4,    //  174446       7
        8.41e-4,    //  200013       8
        7.74e-4,    //  225075       9
        7.17e-4,    //  250218      10
        4.32e-4,    //  500075      11
        3.19e-4,    //  749268      12
```

```
        2.56e-4     // 1001452      13
};

fine = meshSizes[4];
semiFine = fine * 1.25;
coarseFine = fine * 1.5;
coarse = fine * 2;
veryCoarse = fine * 12;

/*  Suggestion
    fine = meshSizes[];
    semiFine = fine * 1.1;
    coarseFine = semiFine * 1.1;
    coarse = coarseFine * 1.1;
    veryCoarse = coarse * 1.1;
*/

cellMin = {
    veryCoarse,   //  0              // Matches largest cell in 1
    semiFine,     //  1              // Matches largest cell in 2
    semiFine,     //  2              // Matches largest cell in 3
    fine,         //  3
    semiFine,     //  4              // Matches largest cell in 1
    fine,         //  5
    fine,         //  6
    fine,         //  7
    fine,         //  8
    fine,         //  9
    fine,         // 10
    fine,         // 11
    fine,         // 12
    coarseFine,   // 13
    fine          // 14
};
// Define expansion ratios
r_exp = {
    1.1,          //  0
    1.0125,       //  1
    1.0125,       //  2
    1.1,          //  3
    1.025,        //  4
    1,            //  5
    1,            //  6
    1,            //  7
    1,            //  8
    1,            //  9
    1.025,        // 10
    1             // 11
```

```
};
// Preallocate vectors for largest grid cells and number of nodes
cellMax = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
nCells = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

// 2 = Inner subdomain
Sn = L_vec[2];
q = r_exp[2];
a0 = cellMin[2];
Call CalExpansionRatio;
cellMax[2] = aN;
nCells[2] = NumNode;

// 1 = Outer subdomain
Sn = L_vec[1];
q = r_exp[1];
a0 = cellMax[2];       cellMin[1] = cellMax[2];
Call CalExpansionRatio;
cellMax[1] = aN;
nCells[1] = NumNode;

// 0 = Surface to bottom inlet side
Sn = L_vec[0];
q = r_exp[0];
a0 = cellMax[1];       cellMin[0] = cellMax[1];
Call CalExpansionRatio;
cellMax[0] = aN;
nCells[0] = NumNode;

// 3 = Surface to bottom outlet side
Sn = L_vec[3];
q = r_exp[3];
a0 = cellMax[1];       cellMin[3] = cellMax[1];
Call CalExpansionRatio;
cellMax[3] = aN;
nCells[3] = NumNode;

// 4 = Inlet to top of hull
Sn = L_vec[4];
q = r_exp[4];
a0 = cellMin[4];
Call CalExpansionRatio;
cellMax[4] = aN;
nCells[4] = NumNode;

// 5 = Inlet to corner of hull
cellMax[5] =  cellMin[5];
nCells[5] = Round(L_vec[5]/cellMin[5]);
```

```
// 6 = bottom to corner of hull inlet side
cellMax[6] =  cellMin[6];
nCells[6] = Round(L_vec[6]/cellMin[6]);

// 7 = bottom to hull
cellMax[7] =  cellMin[7];
nCells[7] = Round(L_vec[7]/cellMin[7]);

// 8 = bottom to corner of hull outlet side
cellMax[8] =  cellMin[8];
nCells[8] = Round(L_vec[8]/cellMin[8]);

// 9 = Outlet to corner of hull
cellMax[9] =  cellMin[9];
nCells[9] = Round(L_vec[9]/cellMin[9]);

// 10 = Outlet to top of hull
Sn = L_vec[10];
q = r_exp[10];
a0 = cellMin[10];
Call CalExpansionRatio;
cellMax[10] = aN;
nCells[10] = NumNode;

// 11 = Inlet to outlet
cellMin[11] = cellMax[1];
cellMax[11] =  cellMin[11];
nCells[11] = Round(L_vec[11]/cellMin[11]);

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Points
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+
Point(1) = {0, -yh, 0, 1.0};
//+
Point(2) = {0, -yh, -z1, 1.0};
//+
Point(3) = {0, -y1, -zh, 1.0};
//+
Point(4) = {0, y1, -zh, 1.0};
//+
Point(5) = {0, yh, -z1, 1.0};
//+
Point(6) = {0, yh, 0, 1.0};

// First surrounding mesh structure
//+
```

```
263 Point(7) = {0, -yInn, 0, 1.0};
264 //+
265 Point(8) = {0, -yInn, -z1, 1.0};
266 //+
267 Point(9) = {0, -yd2, -zd2, 1.0};
268 //+
269 Point(10) = {0, -y1, -zInn, 1.0};
270 //+
271 Point(11) = {0, y1, -zInn, 1.0};
272 //+
273 Point(12) = {0, yd2, -zd2, 1.0};
274 //+
275 Point(13) = {0, yInn, -z1, 1.0};
276 //+
277 Point(14) = {0, yInn, 0, 1.0};

278 // Second surrounding mesh structure
279 //+
280 Point(15) = {0, -yOut, 0, 1.0};
281 //+
282 Point(16) = {0, -yOut, -z1, 1.0};
283 //+
284 Point(17) = {0, -y1, -zOut, 1.0};
285 //+
286 Point(18) = {0, y1, -zOut, 1.0};
287 //+
288 Point(19) = {0, yOut, -z1, 1.0};
289 //+
290 Point(20) = {0, yOut, 0, 1.0};

291 // Outer borders for the domain
292 //+
293 Point(21) = {0, -ySt, 0, 1.0};
294 //+
295 Point(22) = {0, -ySt, -z1, 1.0};
296 //+
297 Point(23) = {0, -ySt, -d, 1.0};
298 //+
299 Point(24) = {0, yEn, -d, 1.0};
300 //+
301 Point(25) = {0, yEn, -z1, 1.0};
302 //+
303 Point(26) = {0, yEn, 0, 1.0};
304 //+
305 Point(27) = {0, -y1, -d, 1.0};
306 //+
307 Point(28) = {0, y1, -d, 1.0};
```

```
// origo for circular arcs
//+
Point(29) = {0, -Oy, -Oz, 1.0};
//+
Point(30) = {0, Oy, -Oz, 1.0};

// Points of intersection with diagonal and arcs
//+
Point(31) = {0, -yd1, -zd1, 1.0};
//+
Point(32) = {0, yd1, -zd1, 1.0};
//+
Point(33) = {0, -yd3, -zd3, 1.0};
//+
Point(34) = {0, yd3, -zd3, 1.0};
//+
Point(35) = {0, -yd4, -zd4, 1.0};
//+
Point(36) = {0, yd4, -zd4, 1.0};
//+
// Horizontal support for diagonals
Point(37) = {0, -ySt, -zd3, 1.0};
//+
Point(38) = {0, yEn, -zd3, 1.0};
//+

//+-----------------------------------------------------------------
// Lines
//+-----------------------------------------------------------------
//+
Line(1) = {2, 1};
//+
Circle(2) = {31, 29, 2};
//+
Circle(3) = {3, 29, 31};
//+
Line(4) = {3, 4};
//+
Circle(5) = {4, 30, 32};
//+
Circle(6) = {32, 30, 5};
//+
Line(7) = {5, 6};
//+
Line(8) = {8, 7};
//+
Circle(9) = {9, 29, 8};
//+
```

```
Circle(10) = {10, 29, 9};
//+
Line(11) = {10, 11};
//+
Circle(12) = {11, 30, 12};
//+
Circle(13) = {12, 30, 13};
//+
Line(14) = {13, 14};
//+
Line(15) = {16, 15};
//+
Circle(16) = {33, 29, 16};
//+
Circle(17) = {17, 29, 33};
//+
Line(18) = {17, 18};
//+
Circle(19) = {18, 30, 34};
//+
Circle(20) = {34, 30, 19};
//+
Line(21) = {19, 20};
//+
Line(22) = {1, 7};
//+
Line(23) = {2, 8};
//+
Line(24) = {31, 9};
//+
Line(25) = {3, 10};
//+
Line(26) = {4, 11};
//+
Line(27) = {32, 12};
//+
Line(28) = {5, 13};
//+
Line(29) = {6, 14};
//+
Line(30) = {7, 15};
//+
Line(31) = {8, 16};
//+
Line(32) = {9, 33};
//+
Line(33) = {10, 17};
//+
```

```
Line(34) = {11, 18};
//+
Line(35) = {12, 34};
//+
Line(36) = {13, 19};
//+
Line(37) = {14, 20};
//+
Line(38) = {15, 21};
//+
Line(39) = {16, 22};
//+
Line(40) = {33, 37};
//+
Line(41) = {33, 35};
//+
Line(42) = {17, 27};
//+
Line(43) = {18, 28};
//+
Line(44) = {34, 36};
//+
Line(45) = {34, 38};
//+
Line(46) = {19, 25};
//+
Line(47) = {20, 26};
//+
Line(48) = {23, 37};
//+
Line(49) = {37, 22};
//+
Line(50) = {22, 21};
//+
Line(51) = {24, 38};
//+
Line(52) = {38, 25};
//+
Line(53) = {25, 26};
//+
Line(54) = {35, 23};
//+
Line(55) = {27, 35};
//+
Line(56) = {27, 28};
//+
Line(57) = {28, 36};
//+
```

```
Line(58) = {36, 24};

////////////////////////////////////////
// Set number of elements along the lines
////////////////////////////////////////
//+
// 0 = Surface to bottom inlet side
Transfinite Curve {38, 39, 40, 54} = nCells[0]-1 Using Progression r_exp[0];
//+
// 1 = Outer subdomain
Transfinite Curve {30, 31, 32, 33, 34, 35, 36, 37} = nCells[1]-1 Using Progression r_exp[1];
//+
// 2 = Inner subdomain
Transfinite Curve {22, 23, 24, 25, 26, 27, 28, 29} = nCells[2]-1 Using Progression r_exp[2];
//+
// 3 = Surface to bottom outlet side
Transfinite Curve {47, 46, 45, 58} = nCells[3]-1 Using Progression r_exp[3];
//+
// 4 = Inlet to top of hull
Transfinite Curve {1, 8, 15, 50} = nCells[4]-1 Using Progression r_exp[4];
//+
// 5 = Inlet to corner of hull
Transfinite Curve {2, 9, 16, 49} = nCells[5] Using Progression r_exp[5];
//+
// 6 = bottom to hull inlet side
Transfinite Curve {3, 10, 17, 55} = nCells[6] Using Progression r_exp[6];
//+
// 7 = bottom to hull
Transfinite Curve {4, 11, 18, 56} = nCells[7] Using Progression r_exp[7];
//+
// 8 = bottom to hull outlet side
Transfinite Curve {5, 12, 19, 57} = nCells[8] Using Progression r_exp[8];
//+
// 9 = Outlet to corner of hull
Transfinite Curve {6, 13, 20, 52} = nCells[9] Using Progression r_exp[9];
//+
// 10 = Outlet to top of hull
```

```
Transfinite Curve {7, 14, 21, 53} = nCells[10]-1 Using Progression r_exp[10];
//+
// 11 = Inlet to outlet
Transfinite Curve {48, 41, 42, 43, 44, 51} = nCells[11] Using Progression r_exp[11];

//+-----------------------------------------------------------------
// Set Surfaces
//+-----------------------------------------------------------------
//+
Curve Loop(1) = {22, -8, -23, 1};
Plane Surface(1) = {1};
Transfinite Surface {1};
Recombine Surface {1};
//+
Curve Loop(2) = {23, -9, -24, 2};
Plane Surface(2) = {2};
Transfinite Surface {2};
Recombine Surface {2};
//+
Curve Loop(3) = {3, 24, -10, -25};
Plane Surface(3) = {3};
Transfinite Surface {3};
Recombine Surface {3};
//+
Curve Loop(4) = {4, 26, -11, -25};
Plane Surface(4) = {4};
Transfinite Surface {4};
Recombine Surface {4};
//+
Curve Loop(5) = {5, 27, -12, -26};
Plane Surface(5) = {5};
Transfinite Surface {5};
Recombine Surface {5};
//+
Curve Loop(6) = {6, 28, -13, -27};
Plane Surface(6) = {6};
Transfinite Surface {6};
Recombine Surface {6};
//+
Curve Loop(7) = {29, -14, -28, 7};
Plane Surface(7) = {7};
Transfinite Surface {7};
Recombine Surface {7};
//+
Curve Loop(8) = {30, -15, -31, 8};
Plane Surface(8) = {8};
```

```
Transfinite Surface {8};
Recombine Surface {8};
//+
Curve Loop(9) = {9, 31, -16, -32};
Plane Surface(9) = {9};
Transfinite Surface {9};
Recombine Surface {9};
//+
Curve Loop(10) = {10, 32, -17, -33};
Plane Surface(10) = {10};
Transfinite Surface {10};
Recombine Surface {10};
//+
Curve Loop(11) = {11, 34, -18, -33};
Plane Surface(11) = {11};
Transfinite Surface {11};
Recombine Surface {11};
//+
Curve Loop(12) = {12, 35, -19, -34};
Plane Surface(12) = {12};
Transfinite Surface {12};
Recombine Surface {12};
//+
Curve Loop(13) = {13, 36, -20, -35};
Plane Surface(13) = {13};
Transfinite Surface {13};
Recombine Surface {13};
//+
Curve Loop(14) = {14, 37, -21, -36};
Plane Surface(14) = {14};
Transfinite Surface {14};
Recombine Surface {14};
//+
Curve Loop(15) = {15, 38, -50, -39};
Plane Surface(15) = {15};
Transfinite Surface {15};
Recombine Surface {15};
//+
Curve Loop(16) = {16, 39, -49, -40};
Plane Surface(16) = {16};
Transfinite Surface {16};
Recombine Surface {16};
//+
Curve Loop(17) = {40, -48, -54, -41};
Plane Surface(17) = {17};
Transfinite Surface {17};
Recombine Surface {17};
//+
```

```
Curve Loop(18) = {17, 41, -55, -42};
Plane Surface(18) = {18};
Transfinite Surface {18};
Recombine Surface {18};
//+
Curve Loop(19) = {18, 43, -56, -42};
Plane Surface(19) = {19};
Transfinite Surface {19};
Recombine Surface {19};
//+
Curve Loop(20) = {19, 44, -57, -43};
Plane Surface(20) = {20};
Transfinite Surface {20};
Recombine Surface {20};
//+
Curve Loop(21) = {45, -51, -58, -44};
Plane Surface(21) = {21};
Transfinite Surface {21};
Recombine Surface {21};
//+
Curve Loop(22) = {20, 46, -52, -45};
Plane Surface(22) = {22};
Transfinite Surface {22};
Recombine Surface {22};
//+
Curve Loop(23) = {21, 47, -53, -46};
Plane Surface(23) = {23};
Transfinite Surface {23};
Recombine Surface {23};


//+------------------------------------------------------------------
// Extrude Surfaces
//+------------------------------------------------------------------
//+
Extrude {-0.1, 0, 0} {
  Surface{1};   Surface{2};   Surface{3};   Surface{4};
  Surface{5};   Surface{6};   Surface{7};   Surface{8};
  Surface{9};   Surface{10};  Surface{11};  Surface{12};
  Surface{13};  Surface{14};  Surface{15};  Surface{16};
  Surface{17};  Surface{18};  Surface{19};  Surface{20};
  Surface{21};  Surface{22};  Surface{23};
  Layers {1};
  Recombine;
}

//+------------------------------------------------------------------
// Classify Boundaries
```

```
624 //+------------------------------------------------------------------
625 //+
626 Physical Surface("frontAndBack", 565) = {15, 388, 16, 410, 17, 432,
    ↪  8, 234, 9, 256, 10, 278, 18, 454, 1, 80, 2, 102, 3, 124, 4, 146,
    ↪  11, 300, 19, 476, 5, 168, 12, 322, 20, 498, 6, 190, 13, 344, 7,
    ↪  212, 14, 366, 23, 564, 22, 542, 21, 520};
627 //+
628 Physical Surface("inlet", 566) = {383, 405, 423};
629 //+
630 Physical Surface("outlet", 567) = {559, 537, 511};
631 //+
632 Physical Surface("top", 568) = {555, 357, 199, 67, 221, 379};
633 //+
634 Physical Surface("bottom", 569) = {515, 493, 471, 449, 427};
635 //+
636 Physical Surface("hull", 570) = {211, 177, 155, 133, 111, 101, 79};


637 //+------------------------------------------------------------------
638 // Create Volume
639 //+------------------------------------------------------------------
640 //+
641 Physical Volume("internalField", 571) = {
642      1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
643     11, 12, 13, 14, 15, 16, 17, 18,
644     19, 20, 21, 22, 23
645 };

646 //+------------------------------------------------------------------
647 // Engage Meshing In GUI
648 //+------------------------------------------------------------------
649 //+
650 Mesh 3;
```