Ship Resistance Computations using OpenFOAM

Muhammad Tufail Shahzad

Master Thesis

presented in partial fulfillment
of the requirements for the double degree:
“Advanced Master in Naval Architecture” conferred by University of Liege
"Master of Sciences in Applied Mechanics, specialization in Hydrodynamics, Energetics and Propulsion” conferred by Ecole Centrale de Nantes

developed at West Pomeranian University of Technology, Szczecin
in the framework of the

“EMSHIP”
Erasmus Mundus Master Course
in “Integrated Advanced Ship Design”

Ref. 159652-1-2009-1-BE-ERA MUNDUS-EMMC

Supervisor: Dr. Maciej Taczła, West Pomeranian University of Technology, Poland
Co- supervisor: Prof. Dr. Serdar Beji, Istanbul Technical University, Turkey

Reviewer: Prof. Robert Bronsart, University of Rostock, Germany

Reviewer: Prof. Pierre Ferrant, École Centrale de Nantes, France

Szczecin, May 2017

PREFACE

This thesis is a part of the requirements for the European Masters Course in Integrated Advanced Ship Design and Offshore Structure (EMSHIP) organized by University of Liège (ULG), Belgium –Coordinator, Ecole Centrale de Nantes (ECN),France, ,University of Rostock (URO), Germany, Zachodniopomorski Uniwersytet Technologiczny (ZUT), (West Pomeranian University of Technology), Poland, “Dunarea de Jos” University of Galati (UGAL), Romania, University of Genoa (UNIGE), Italy.

And in collaboration with ICAM - School of mechanical and manufacturing engineering(FRANCE), University of Michigan (NAME, USA), University of Osaka (NAOE, Japan), Universidade Federal do Rio de Janeiro (UFRJ, Rio de Janeiro), University of New South Wales (UNSW, Australia), Pusan National University, KOSORI (PNU, Korea), Vietnam Maritime University (VIMARU, Vietnam), Istanbul Technical University (ITU, Turkey), University of Sciences and Technology of Oran Mohamed Boudiaf (USTO-MB, Algeria), WEGEMT - European Association of Universities in Marine Technology and Related Sciences (UK).

This Thesis has been carried out at Delta Marine, Turkey in collaboration with Zachodniopomorski Uniwersytet Technologiczny (ZUT), (West Pomeranian University of Technology), Poland and Istanbul Technical University (ITU, Turkey).

NOMENCLATURE

A Wetted surface area, thin ship theory (m^2)

A0 Propeller disc area [πD2/4]

AD Propeller developed blade area ratio, or developed blade area (m^2)

AE Propeller expanded blade area ratio

AP Projected bottom planing area of planing hull (m^2) or projected area of propeller blade (m^2)

AT Transverse frontal area of hull and superstructure above water (m^2)

AX Midship section area (m^2)

b Breadth of catamaran demihull (m), or mean chine beam of planing craft (m)

B Breadth of monohull or overall breadth of catamaran (m)

Bpa Mean breadth over chines [= AP/LP] (m)

Bpx Maximum breadth over chines (m)

BWL Breadth on waterline (m)

c Section chord (m)

CA Model-ship correlation allowance coefficient

CB Block coefficient

CDair Coefficient of air resistance [Rair/1/2ρaATV2]

Cf Local coefficient of frictional resistance

CF Coefficient of frictional resistance [RF/1/2ρWSV2]

CL Lift coefficient

CM Midship coefficient [AX/(B× T)]

CP Prismatic coefficient [∇/(L× AX)] or pressure coefficient

CR Coefficient of residuary resistance [RR/1/2ρSV2]

CS Wetted surface coefficient [S/√∇ · L]

CT Coefficient of total resistance [RT/1/2ρSV2]

CV Coefficient of viscous resistance [RV/1/2ρSV2]

CW Coefficient of wave resistance [RW/1/2ρSV2]

CWP Coefficient of wave pattern resistance [RWP/1/2ρSV2]

D Propeller diameter (m)

Dair Aerodynamic drag, horizontal (planing craft) (N)

DF Planing hull frictional resistance, parallel to keel (N)

E	Energy in wave front
FH	Hydrostatic pressure acting at centre of pressure of planing hull (N)
FP	Pressure force over wetted surface of planing hull (N)
Fr	Froude number [V/√g · L]
Frh	Depth Froude number [V/√g · h]
Fr∇	Volume Froude number [V/g · ∇1/3]
Fx	Yacht sail longitudinal force (N)
Fy	Yacht sail transverse force (N)
g	Acceleration due to gravity (m/s2)
G	Gap between catamaran hulls (m)
GM	Metacentric height (m)
h	Water depth (m)
H	Wave height (m)
HT	Transom immersion (m)
iE	Half angle of entrance of waterline (deg.), see also 1/2 αE
J	Propeller advance coefficient (VA/nD)
k	Wave number
KT	Propeller thrust coefficient (T/ρn2D4)
KQ	Propeller torque coefficient Q/ρn2D5)
L	Length of ship (m)
Lair	Aerodynamic lift, vertically upwards (planing craft) (N)
LAPP	Appendage lift (N)
LBP	Length of ship between perpendiculars (m)
lc	Wetted length of chine, planing craft (m)
LCB	Longitudinal center of buoyancy (%Lforward or aft of amidships)
LCG	Longitudinal center of gravity (%Lforward or aft of amidships)
Lf	Length of ship (ft)
lK	Wetted length of keel, planing craft (m)
lm	Mean wetted length, planing craft [= (lK + lc)/2]
LOA	Length of ship overall (m)
lp	Distance of center of pressure from transom (planing craft)(m)
LP	Projected chine length of planing hull (m)
LPS	Length between pressure sources
LWL	Length on waterline (m)

n	Propeller rate of revolution (rps)
N	Propeller rate of revolution (rpm), or normal bottom pressure load on planing craft (N)
P	Propeller pitch (m)
PAT	Atmospheric pressure (N/m²)
P/D	Propeller pitch ratio
PD	Delivered power (kW)
PE	Effective power (kW)
PL	Local pressure (N/m²)
PS	Installed power (kW)
PV	Vapour pressure (N/m²)
Q	Propeller torque (Nm)
Rair	Air resistance (N)
Rapp	Appendage resistance (N)
Re	Reynolds Number (ρVL/μ or VL/ν)
RF	Frictional resistance (N)
RFh	Frictional resistance of yacht Hull (N)
RInd	Induced resistance of yacht (N)
rps	Revolutions per second
rpm	Revolutions per minute
RR	Residuary resistance (N)
RRh	Residuary resistance of yacht Hull (N)
RRK	Residuary resistance of yacht keel (N)
RT	Total hull resistance (N)
RV	Viscous resistance (N)
RVK	Viscous resistance of yacht keel (N)
RVR	Viscous resistance of yacht rudder (N)
RW	Wave resistance (N)
RWP	Wave pattern resistance (N)
S	Wetted surface area (m²)
SAPP	Wetted area of appendage (m²)
SC	Wetted surface area of yacht canoe body (m²) or separation between catamaran semi hull centrelines (m)
sfc	Specific fuel consumption
SP	Propeller/hull interaction on planing craft (N)

t	Thrust deduction factor, or thickness of section (m)
T	Draught (m), or propeller thrust (N), or wave period (secs)
TC	Draught of yacht canoe body (m)
U	Speed (m/s)
V	Speed (m/s)
Va	Wake speed (VS(1 − wT)) (m/s)
VA	Relative or apparent wind velocity (m/s)
VK	Ship speed (knots)
VR	Reference velocity (m/s)
VS	Ship speed (m/s)
W	Channel width (m)
wT	Wake fraction
Z	Number of blades of propeller
(1+k)	Form-factor, monohull
(1+βk)	Form factor, catamaran
1/2 αE	Half angle of entrance of waterline (deg.), see also iE
β	Viscous resistance interference factor, or appendage scaling factor, or deadrise angle of planing hull (deg.) or angle of relative
or	apparent wind (deg.)
δ	Boundary layer thickness (m)
ε	Angle of propeller thrust line to heel (deg.)
ηD	Propulsive coefficient (η0ηHηR)
ηO	Open water efficiency (JKT/2π KQ)
ηH	Hull efficiency (1 − t)/(1 − wT)
ηR	Relative rotative efficiency
ηT	Transmission efficiency
γ	Surface tension (N/m), or wave height decay coefficient, or course angle of the yacht (deg.), or wave number
φ	Heel Angle (deg.), or hydrodynamic pitch angle (deg.)
λ	Leeway angle (deg.)
μ	Dynamic viscosity (g/ms)
ν	Kinematic viscosity (μ/ρ) (m^2/s)
ρ	Density of water (kg/m3)
ρa	Density of air (kg/m3)

σ Cavitation number, or source strength, or allowable stress (N/m^2)

τ Wave resistance interference factor (catamaran resistance/monohull resistance), or trim angle of planing hull (deg.)

τ_c Thrust/unit area, cavitation (N/m^2)

τ_R Residuary resistance interference factor (catamaran resistance/monohull resistance)

τ_W Surface or wall shear stress (N/m^2)

θ Wave Angle (deg.)

ζ Wave elevation (m)

∇ Ship displacement volume (m3)

∇_C Displacement volume of yacht canoe body (m3) Ship displacement mass ($\nabla\rho$) (tons), or displacement force ($\nabla\rho g$) (N)

ABBREVIATIONS

ABS	American Bureau of Shipping
AEW	Admiralty Experiment Works (UK)
AFS	Antifouling systems on ships
AHR	Average hull roughness
AP	After perpendicular
ARC	Aeronautical Research Council (UK)
ATTC	American Towing Tank Conference
BDC	Bottom dead centre
BEM	Boundary element method
BN	Beaufort Number
BSRA	British Ship Research Association
CAD	Computer-aided design
CCD	Charge-coupled device
CFD	Computational fluid dynamics
CG	Centre of gravity
CLR	Centre of lateral resistance
CODAG	Combined diesel and gas
CP	Controllable pitch (propeller)
CSR	Continuous service rating
DES	Detached eddy simulation
DNS	Direct numerical simulation
DNV	Det Norske Veritas
DSYHS	Delft systematic yacht hull series
DTMB	David Taylor Model Basin
EFD	Experimental fluid dynamics
FEA	Finite element analysis
FP	Forward perpendicular, or fixed pitch (propeller)
FRP	Fibre-reinforced plastic
FV	Finite volume
HP	Horsepower
HSVA	Hamburg Ship Model Basin

IESS	Institute of Engineers and Shipbuilders in Scotland
IMarE	Institute of Marine Engineers (became IMarEST from 2001)
IMarEST	Institute of Marine Engineering, Science and Technology
IMechE	Institution of Mechanical Engineers
LDA	Laser Doppler anemometry
LDV	Laser Doppler velocimetry
LR	Lloyd's Register of Shipping
MAA	Mean apparent amplitude
MCR	Maximum continuous rating
NTUA	National Technical University of Athens
ORC	Offshore Racing Congress
P	Port
PIV	Particle image velocimetry
QPC	Quasi propulsive coefficient
RANS	Reynolds Averaged Navier–Stokes
RB	Round back (section)
RINA	Royal Institution of Naval Architects
ROF	Rise of floor
Rpm	Revolutions per minute
rps	Revolutions per second
S	Starboard
SAC	Sectional area curve
SCF	Ship correlation factor
SG	Specific gravity
SNAJ	Society of Naval Architects of Japan (later to become JASNAOE)
SNAK	Society of Naval Architects of Korea
SNAME	Society of Naval Architects and Marine Engineers (USA)
SP	Self-propulsion
TBT	Tributyltin
TDC	Top Dead Centre
TDW	Tons deadweight
TE	Trailing edge of foil or fine
TEU	Twenty-foot equivalent unit [container]
ULG	University of Liege

CONVERSION OF UNITS

1m	3.28 ft
1 in	25.4 mm
1 mile	5280 ft
1 ft	12 in
1 km	1000 m
1 nautical mile (Nm)	6078 ft
1 kg	2.205 lb
1 tonne	2240 lb
1 tonne	1000 kg
1 UK gal	4.546 litres
1 mile/hr	1.61 km/hr
1 knot	1 Nm/hr
1 knot	0.5144 m/s
1 lb	4.45 N
1 bar	14.7 lbs/in^2
1 lbs/in^2	6895 N/m^2
1 HP	0.7457 kW
Fr	0.2974 VK/$\sqrt{Lf}$

DECLARATION OF AUTHORSHIP

I declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

Where I have consulted the published work of others, this is always clearly attributed.

Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

I have acknowledged all main sources of help.

Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

This thesis contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree or diploma.

I cede copyright of the thesis in favor of the Zachodniopomorski Uniwersytet Technologiczny (ZUT), Poland.

Date: 31st May, 2017 Signature: Muhammad Tufail Shahzad

ACKNOWLEDGEMENTS

This thesis was developed in the frame of the European Master Course in “Integrated Advanced Ship Design” named “EMSHIP” for “European Education in Advanced Ship Design”, Ref.: 159652-1-2009-1-BE-ERA MUNDUS-EMMC.

I would like to acknowledge and thank my thesis supervisor Prof. Maciej Taczala at Faculty of Maritime Technology and Transport, Zachodniopomorski Uniwersytet Technologiczny (ZUT), (West Pomeranian University of Technology), Poland, Co-supervisor, Prof. Dr. Serdar Beji at Faculty of Naval Architecture and Ocean Engineering, Istanbul Technical University (ITU, Turkey), for all their patience, motivation, enthusiasm, Immense knowledge, help and support throughout the project.

My sincere Thanks also goes to Dr. Levent Kaydihan, Director of Advanced Engineering Department at Delta Marine (Mesh), Istanbul (Turkey), for offering me internship opportunity in their group and leading me working on the diverse exciting project. I would also like to show my gratitude to everyone at Delta Marine who allowed me to develop this project in a pleasant work environment and especially thanks to Mr. T. G. Canyurt for all help concerning both support in the use of software and valuable theory discussions time to time.

Most importantly, I would like to thank all my EMSHIP 6th Cohort and overseas friends who helped me to get through this master’s study.

Last but not least, I would like to thank my family: my parents Mr. Zahoor Ahamd and Shamim Bagum for supporting me spiritually throughout my life.

Szczecin (PL), June 2017
Muhammad Tufail Shahzad

Future of CFD

The times are changing. Lifestyles are changing. The industry is changing. The CAE software market is changing. And yes, also the CFD codes are changing !!!

CONTENTS

TABLE OF FIGURES

LIST OF TABLES

ABSTRACT

Computational Fluid Dynamics (CFD) for ship hydrodynamics has been advancing significantly within the last decade towards providing design tools which are capable of full-scale modeling and simulations with much less cost and more accuracy. For ship resistance and powering, CFD has become increasingly important part of the design process. Establishing the reliability of CFD for specific applications requires test runs and comparisons with experimentally measured data. In this work, ship resistance components such as viscous frictional and pressure forces and wave resistance are to be obtained by virtual towing tank simulation technique by using OpenFOAM CFD code. In numerical simulations, optimum mesh arrangement and different turbulence models such as k-omega (k- ω), and SST (Shear Stress Transport) are considered first for a definite test run and the most accurate combination is determined. Then, a definite hull form designed by Delta Marine Company of Istanbul, Turkey is investigated under the decided setting of parameters for its total resistance characteristics for a number of different speeds. Finally, a speed versus resistance curve for the selected ship form is obtained from the results of numerical computations and compared with the available towing test results. Concluding comments are made on the performance and reliability of the OpenFOAM as a numerical tool in determining the resistance characteristics of a ship. The suggested mesh arrangements and turbulence parameter settings for possible best results are also pointed out in closing;

Keywords: CFD, Hydrodynamics, Performance, Resistance calculations, OpenFOAM, Turbulence model

THESIS STRUCTURE

The thesis begins with an abstract written in English. This is followed by a short introduction in Chapter 1, which summarizes the purpose and aim of the thesis, the motivation for the project, Approach towards the concerned topic and outcomes benefits in terms of ship hydrodynamic.

In chapter 2, history and recent development (Literature review) of Computational Fluid Dynamic (CFD) in terms of ship hydrodynamics (resistance) is explained.

In Chapter 3, Basic knowledge of CFD, pre-explanation of Navier-Stokes equations, their approach and their versatility have been explained.

In chapter 4, and introduction of OpenFOAM, structure of its platform, Solvers importance, geometry formation, mesh generations (Used protocols) has been explained.

In Chapter 5, a detailed information about the computations resources and its available capacity for the current project has been presented. A brief introduction for supercomputer facility (LSF-Used facility) with its working flowchart is explained. Scaling rules and its effect are explained here as well.

In Chapter 6, two different solvers are used to perform the resistance investigation of KCS Ship and the comparison of the two different turbulence solvers is drawn, Post-processing results for each turbulence models are also presented.

In Chapter 7, Investigation is performed on a Delta Ship using SST K-omega as turbulence model and CFD results are compared with EFD data along with their post-processing results.

In Chapter 8, Conclusion is drawn for the performed investigation which followed by the Future work in Chapter 9, self-recommendation learned while performing this investigation in Chapter 10 and used references in Chapter 11.

Note: The thesis is addressed at the reader with a basic knowledge of Fluid dynamics in terms of ship hydrodynamic applications, but not certainely the detailed effective parameters.

1. INTRODUCTION

Hydrodynamic problems for ship designs are very well known to be the most complex. The complexity of this problem is due to its conventional approach because it's very expensive, time-consuming while performing Towing tank test or in any other laboratories. Ship Hydrodynamics is one of the most important tools and today demand. Since a couple of decades, it is seen that the industrial and commercial interest of ships and related floating structures in Seaway has been expanded considerably. nowadays, it's getting more usual to inspect a ship or other related offshore structures on its seakeeping performance. Among other things, the behavior of a ship in the sea has a significant impact on the various parameters and some are listed below type:

1. Safeness of passengers, crew, cargo and ship;
2. Satisfaction of passengers and crew;
3. Dynamic loads effects on the structure (cargo or equipment) and,
4. Uninterrupted sea speed of a ship and the fuel consumption.

Accurate prediction of difference hydrodynamic forces on a ship in motion has a great importance in ship design. Water resistance at a specific speed determines the required engine power and hence the consumption of the fuel as well. Therefore, in the design of the hull, minimization of these hydrodynamic forces is very important.

The very basic simplification in ship design is to consider separately the performance of the ship in the calm water and also in the sea. Optimization of a hull (hydrodynamically) primarily requires the resistance calculations. While doing this, calm sea and the open sea effects are generally taken into account as an added wave resistance. The prediction of ship hydrodynamic performance can be split into the main general areas; Resistance and propulsion, Seakeeping and Maneuvering etc. in reality, resistance and propulsion are closely interdependent on each other; resistance determines the thrust which is required of propulsion and hence propulsion deals with to provide that thrust. The basic problem for the naval architects is the prediction of resistance at very early stages in order to make as much as a possible change in the design phase of the ship. When a body moves through fluids, it experiences forces which are opposing to its motion so when a ship moves through water or air it experiences the both air and water forces. The resistance calculations are usually performed in still water with no wind (Banks, Phillips, & Stephen, 2010).

Talking about any propulsion system, it interacts with the hull of the ship and then the flow field is changed by the hull, usually upstream hence the propulsion systems changes the flow field at the ship hull. It has been seen that the naval architects have always considered the propeller and ship separately and have introduced considerations (factors, special efficiencies) for the interaction effect. Which decomposition has been experienced by many as an important assistance in order to construct the difficult problem of ship hydrodynamics? The engine power required to drive the ship at a certain speed is only dependent on the ship resistance, there is an important factor also called the propulsive efficiency which means the performance of propeller and its interaction with the hull. However, resistance is the single most important parameters determining the required propulsive power. The main focus of this current work will be on resistance and propulsion. In order to reach required tasks there are different approaches which can be followed:

Empirical Approach: Engineers require the simple and most reasonable accurate estimations. Considering the power requirement of the ship, it has been seen that common approaches combine much simpler physical model and statistical process, for the estimation of relations between different variable, to determine the required coefficient. The coefficient may be given in form of constants, formulas or curves.

Experimental Approach: Experimentations have always been the core of technologies since decades. In the shipbuilding industry, when it comes to extracting the full information about the ship at full scale the basic idea of model testing is to experiment with a scale model. It's mandatory to have a certain degree of empirical approach first in order to keep continuing the research and standardization efforts. Usually, the prediction accuracy of ship resistance is increased by some empirical means, a method of model-to-ship correlation is openly used. Total resistance can be seen in various ways. It has seen that model basin were developed to adopt the approaches that seem the most accurate and relevant to their respective organization's collective experience and with their database.

Numerical approaches (CFD): This can be regarded as the most efficient and current work approach. When the computations techniques were expanding globally, another CFD approach, by solving the differential equations (numerically) of viscid flow (Navier-Stokes Equations), were developed. Since 1980 numerical methods in Computational Fluid Dynamics (CFD) are being used in order to solve the ship design and other offshore related complex problems. For ship resistance and powering of the ship, CFD has become the most important and a crucial part of the design process.

Computational Fluid Dynamic (CFD), which provides a cost-effective way for the simulation of flow field based on numerical methods and governing method. Computational techniques have replaced the governing equations. Experimental methods have played an important role in the process of validation and exploring the limits of approximation to these governing equations. It complements experimental and theoretical fluid dynamics by providing the alternative, less expensive and with more accuracy, means of testing fluid flow system. Talking about its worthiness, main advantages of CFD today is that it is providing a tremendous assistance in the solution of problems which wasn't that easy in the past. Today Engineers can analyses and model an extensive range of problems on the computer with low expenses and less time to approach the capable visualization. (Sayma, Abdulnaser, 2009)

In just over three decades Computational Fluid Dynamics (CFD) for ship ***hydrodynamics*** has overpowered all the expectations heading toward the tremendous progress, capabilities and the milestone providing the best design tool for full-scale and model simulations in order to determine the ship performance accurately. This is due to enabling technologies such as free surface capturing, six ***Degree of Freedom*** (6DOF) motion prediction, adaptive grid refinement, dynamic overset grids, high performance of computation, environmental modeling and different methods for optimization in order to reach the optimum results. Verification and validation procedures with their applications, including ***resistance*** and ***propulsion***, ***seakeeping***, ***maneuvering,*** stability also make it best in this regard. (F. Stern, 2012)

CFD is becoming very important and being widely used in many complex designs and analysis of equipment related to flow of the fluid and to heat transfer. There is a wide range of commercial software available in the market to approach these purposes, which are user-friendly and by using these tools one can be to generate a number of colorful results. But in the meantime, it is also important to understand these CFD packages completely. Taking into account the safety factors, the behavior of the ship in water is very important, when we talk about hydrodynamics mostly one referring to the term resistance and propulsion and in the current, work a brief study about resistance will be studied under the CFD Code.

Hence, Hydrodynamic behavior based on CFD simulation becomes the major factor for minimizing ship resistance. Reducing the resistance leads to less consumable power, more speed, fewer emissions, and noises.

Calculation of resistance characteristic of the ship is one of the most beneficial topics in Naval Architecture, Offshore and Ocean Engineering. In general, the resistance of a ship Phenomenon occurs when the fluid force acting on the ship is being disturbed in its onward motion, resulting in an exerting pressure on the body of ship hull that may cause deformation in the ship structure and alter the flow of the fluid itself. At the very initial stage of design, it is highly recommended that to determine the total drag (resistance) in order to obtain the maximum speed of the ship and then to determine what size of the engine is required to reach a desired cruising speed. By doing this, it helps the designers to make sure whether the drag of the ship is an acceptable level from a financial as well as a physical standpoint. Because it directly effects on the cost of engine, fuel capacity of the engine in order to meet the requirement. Total drag (resistance) measurement is a very crucial parameter that has been studied by Naval Architects since this determines the power which is required to propel a ship (AKTAR, 2012).

In the preliminary stage, the model test is expensive and time-consuming. CFD computation might be more practical which can come up more accurate and more reliable full-scale power prediction simulating with effective cost in a very least time as compared with traditional model testing techniques.

1.1.Objectives

A very precisely explanation of CFD always leads us to a branch which is a relatively young science where the fluid dynamics is being simulated on a computer. The futuristic challenges in CFD research for ship hydrodynamics focusing in the development of fast, more stable and computationally (numerically) efficient methods which are able to perform the analysis at the different scales which are mostly involved in the analysis of practical ship hydrodynamics situations. CFD for ship hydrodynamic has been well benchmarked. Computational ship hydrodynamic is reviewed with a different perspective and special focus on the numerical methods, critical assessment and High-Power Computing (HPC) both nowadays and way forward. The main objective of the thesis to perform an investigation on ship hydrodynamic Applications in terms of resistance. Proceeding work will be carried out by virtual towing tank simulation techniques by using an open source CFD solver OpenFOAM. Our main focus of this work is to investigate a passenger ship designed by Delta Marine, turkey. In order to proceed with the benchmark, we will try two different Turbulence models, SST-Omega and Standard-k-Omega for different speeds. After the comparison of these two turbulence models results we will choose the best one, which gives us less error mean more accurate results, for the passenger ship.

In any work or research, the most important thing is to validate the proceeding work to make sure that upcoming results will be as much accurate as expecting. In the current work, to validate the investigation, experimental and CFD workshop results will be used for comparison. The validation process is dividing into four stages mentioned below:

1. Virtual test results will be investigated of common ship hull, KCS using two different turbulence models; Standard K-Omega & SST K-Omega;
2. After the selection of turbulence model, virtual test results will be performed at 7 different speed for a ship designed by Delta Marine and their data will be compared with Experimental results;
3. Discussion over the accuracy and user-friendly approach toward OpenFOAM when calculating the ship resistance of a ship;
4. In closing, the basis on the current research some recommendation will be written for the improvement in order to examine the futuristic prospects of CFD analysis in ship design, by using OpenFOAM.

For the grid dependency, optimum mesh arrangement will be applied to the virtual model and results will also be compared with experimental results.

1.2.Scope of Study

From the engineering point of view, the problem needs to be the simplest as possible in order of accurate explanation of the actual real world system Hydrodynamic has always been an important and essential subject for many disciplines. In the proceeding work, Hydrodynamic applications in terms of resistance are being investigated under CFD program OpenFOAM. Computations for the current work will be carried out by using Supercomputer techniques provided by High computation center of Turkey (Uhem). In OpenFOAM, we will be mostly working with InterFoam and InterDyMFoam solver and for the good postprocessing results, SnappyHexMesh will be used. After the simulation, data will be reorganized manually and will be compared with container ship design by Delta Marine. suitable turbulence model investigation is also being performed for this process.

1.3.Benefits

Computational Fluid Dynamics (CFD) is a very important computational tool to solve the basic equation that models the flow movements. Although the impetus to this approach was basically provided by the aerospace industry, it has then expanded widely become the essential tools for nearly every industry today. The basic objective of this tool is to solve approximately the flow and the basic equation that give the movement and other characteristics of the flow. CFD provides comprehensive virtual analyses for engineers. Using these analyses engineers can upgrade many objects around us. Approximation of total resistance in the very early design phase becomes very important due to the need of a ship design for more efficient. Knowledge about flow details is increasing interest not only in hull form design but also in the design of appendages (propellers, rudders or energy-saving devices).

Nowadays these techniques are found in almost all the fields ranging from engineering to medical research. In Ship Hydrodynamic Applications several important aspects are taken into account: Resistance (Area of interest), and Propulsion, Seakeeping, Propeller design, and Maneuverability.

1.4.Motivation

Applications of newly built ships continually developed in response to social, technical and economic factors which includes changing in operational speed and fluctuations in fuel prices. These changes are dependent on the reliable estimation of ship propulsive power. There is a growing need to reduce the fuel consumption, reduce power and other operating costs. There is various kind of approaches in order to reach the goal and various kind of methods whereby ship resistance and propulsion are evaluated but these evaluations can never be an exact science, it always require a combination of analysis, experiments performed, accuracy in computations and empiricism.

The resistance of a ship is of course not a new concept, but still being used for the concern problems of vessels, a perfect detail design in this regard requires continuous efforts to improve the prediction of ship resistance. Resistance can affect significantly on the velocity of the vessel which always comes up with an additional requirement of more power and fuel. In some cases, it can also cause the damage to the hull (Appendages) and then a heavy cost to repair this damage. It is obvious that it's very expensive in order to run the project and up to date such a large container ship always. Although the model testing of ships in towing tanks has always been an excellent approach for ship resistance. In today world, a fast growing need for accuracy, in particular, the wave resistance on the determination, a well-defined approach is needed for the industry.

The main goal of any CFD analysis is to approach the deepest understanding of the problem under consideration. Continuously increasing computational power leads to the use of Computational Fluid Dynamics (CFD) methods like Reynolds-Averages-Navier-Stokes-Equations (RANSE) as a common tool in order to predict the hydrodynamic performance of a ship. Talking about CFD, it has been a tremendous revolution since decades in every field of the world, starting from a tennis ball heading towards the dynamics and jet propulsion and motion of ships, as long as the right set of the equation is applied, CFD can come up with more accuracy than any other existing tools. A vessel must be designed in a way where with a minimum of external force it should move efficiently through the water, by using CFD codes, it becomes very efficient and more reliable for the design process of Shipbuilding, Ship hydrocyanic in terms of resistance and propulsion is an area of active research and there is strong motivation to further development for computational techniques for these virtual simulations.

1.5.Approach

The general process for CFD analysis for ship hydrodynamic is shown in the Figure below and the current work approach will also be the same from pre-processing to post-processing.

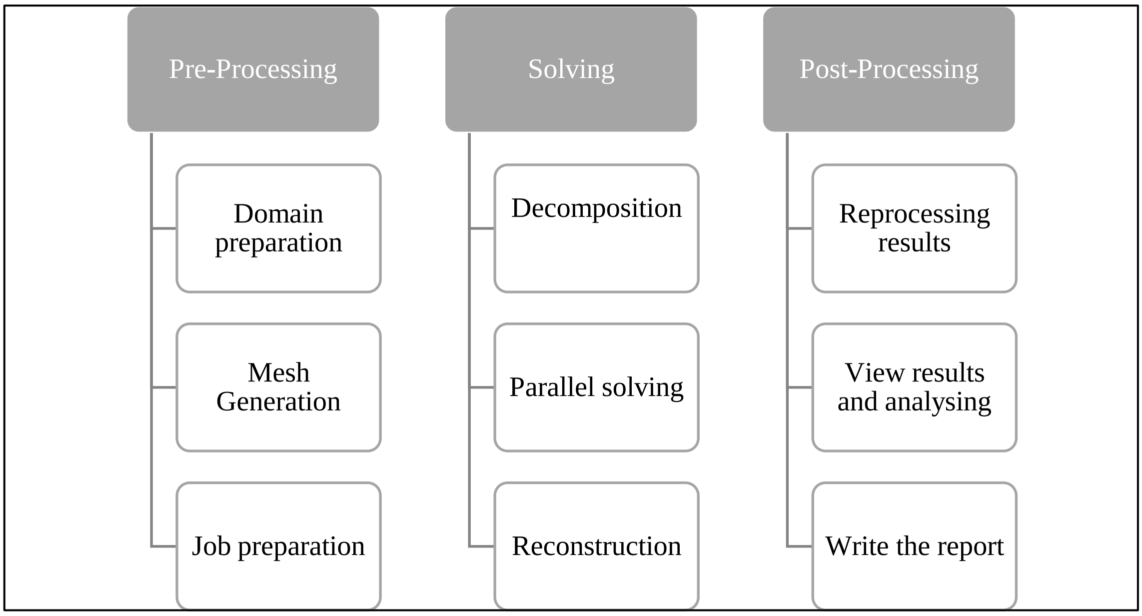


Figure 1 - General Process of CFD analysis

In order to approach the required task, I will go through very basic knowledge of CFD and resistance to study the brief knowledge of the physical aspects of CFD and how it is used in the estimations of resistance of a ship. In order to perform the numerical simulations, KRISO Container Ship (KCS) hull is selected for this research, I will investigate the performance of KCS hull with two different turbulence model, SST-K-Omega, and Standard-k-Omega, in order to get the best one and then to keep that numerical setup for the Delta ship. Discretization will be performed for a better idea of resistance components and foam factor. After a detailed investigation, the result will be compared with the literature data and comparison will be made. A container ship designed by Delta Marine will be simulated by using OpenFOAM and hence the results will be compared with its experimental test, Performed at CTO, Poland.

This work gives the algorithm details that computes the resistance of ship in a two-phase fluid environment including 6 degrees of freedom (DoF) motion. Selected numerical tool, software, to perform this investigation is OpenFOAM, an Open Source object-oriented library for numerical simulations in continuum mechanics, written in the C++ programming language.

2. LITERATURE REVIEW

The resistance of a ship is of course not any new approach in naval architecture, but as it's mentioned before, a container ship requirement is always based on continual effort into the improving resistance prediction methods. A simple understanding can be described as; Resistance of a ship can affect the forward velocity of the ship which in turn requires more power and fuel (costly). When this phenomenon happened, it always results in damage to the hull and requires repair work. It is clearly obvious how much of an expensive operation running and up keeping large container ships is becoming. Even though model testing of ships in towing tanks has been an excellent approach to determining resistance values (and sinkage and trim) up to now, with an increasing need for accuracy especially of wave resistance (Wortley , 2013). whether the matter is made from discrete very tiny particles or is a continuum, divisible or infinite, ancient Greeks have debated over 2000 years. This question was not answered until the 19th century (Brownian motion, etc.), and in the meantime, different very useful thermodynamics theories were developed. Later on, with the development of statistical mechanics, it becomes easy to understand where these laws are come from, in terms of fundamental physics.

According to the Braithwaite, the same theory is true of hydrodynamics as well, the study of fluid flow, which was also developed preliminary to the conclusion of the atom vs. continuum debate. Now, we know that fluids are made from particles, we can explain some fluid phenomena in terms of more fundamental physics, for instance by consideration of particles we can predict the viscosity of the gas, mean-free paths etc. However, classical hydrodynamics, meaning without consideration for the particle nature of matter, making only occasional reference to particles (Braithwaite, 2014).

In 1870, W. Froude did an investigation on ship resistance with the use of models. He experienced that the configuration of waves around the similar geometry was similar at corresponding speeds which mean the speeds proportional to the square root of the model length. From his work, he mentioned that the total resistance can be further divided into skin friction resistance and residual, mainly wave resistance, resistance (Froude, 1872). He noticed that the specific residuary resistance would remain same at corresponding speeds between the model and ship. This theory was not appreciated well in the start but got attention when the full-scale model test had been carried out. His proposal was initially not well received but gained favor after full-scale tests had been carried out. HMS Greyhound (100 ft) was towed by a larger vessel and the results showed a substantial level of agreement with the model predictions (Froud, 1874).

In between 1890, research towards the resistance tests using model had been realized by applying the full potential theory. Various kind of routine test for the specific ships was performed in order to progress in this era and also tests were performed on series of models. A very significant and primarily contribution to this era was done by (Taylor, 1908) where the influence of the shape of the midship section of a vessel upon its resistance was concerned and his method had been most widely used hitherto in order to determine the ship resistance. Although Taylor method was based on experiments performed with differing hull form. Later on, this research was followed by (Baker, 1913), where the main purpose of his research was to show the value of tank experiments to the shipbuilding industry.

The word 'Hydrodynamics' comes from the simple water formula $H_2O.$ This is hydro, from this the word hydrodynamics is derived, because hydrogen is the dominating term. That's the reason we will always look to the dynamics of the hydrogen basically of water and hence we call it hydrodynamics. we often say that the fluid is incompressible and we take this supposition in major hydrodynamics studies, we most of time assume that fluid is non-viscous, and when we are saying this Of course, there are some branches, where we suppose that the fluid viscosity. Particularly, where the fluid viscosity is important whereas, in a major case like, on the like. Taking an example about aircraft, it is seen that boundary near the structure give viscous. Viscous forces are very important so we have a boundary layer beyond the boundary layer, a certain layer then this fluid can have considered as inviscid. Also, when we look at the motion of a ship, around the ship, viscous forces play a very certain role. The role of computational techniques is taking place in the industry because it is fulfilling every modern need with an accurate precision and most relevant solution of the problem. This does not totally replace the need for model testing and data collection at full-scale, that's the reason Hydrodynamic institutions are making effort to enlarge their activities through a very fine integration of experimental and computational approach so that the increasing demands of industries can be well operated.

In 2009, ship-speed performance based on a computational method was examined by (Choi, Kim, Lee, Choi, & Lee, 2009), in this research computation were performed under identical conditions (model). A series of model tests have been performed and hence again it comes up with the good agreement of using RANS equation of ship speed problems. In order to perform the validation and effectiveness of CFD approaches completely based on solving Reynolds Average Naiver-Stokes (RANS) equations, three different ship models are offered to perform this investigation. These models are KRISO Container Ship KCS, DTMB 5415 (US Naval Combatant) and KVLCC2 (KRISO Crude Carrier) which have complex curvatures (Hull). All of the mentioned models have the geometry; transom sterns and bulbous bows which make it difficult to solve the flow problem with the methods based on potential theory. It has been investigated that KCS gives good agreement numerical results comparing with its experimental data by using compiling vortex lattice method (VLM) and RANS solver. (Prakash & Subramanian, 2010). In the same year, Free surface flow around various commercial ships is examined where the main focus of the work was resistance components, free surface deformation, velocity profile on a propeller plane, with the help of RANSE based solver a good comparison with the experimental results is achieved. In this research, the computational predictions result in similar tendencies with the experimental approach. These obtained differences hence then ensure that there is potential to apply the computational method to predicting speed performances in the initial hull-form design stage. It has seen that the obtained results show that streamlines along the ship hull can be accurately estimated with this technique. (Choi, Min , Kim, Lee , & Seo , 2010). Following the current ongoing research, a detail about flow around container ship KCS using three grids and their relevant results, in this research verification and validation of resistance and wave profile are calculated using the ITTC recommended procedures. The successful practice of uncertainty analysis accumulated valuable experiences and help to improve the calculation (Zhang, Verification and validation for RANS simulation of KCS container ship, 2010).

The theoretical background of CFD came into sight in the early 1960s, however, its applications in the industry become only possible due to the rapid increase in computer technology. Resistance and Propulsion of a ship are referred to a core and futuristic scientific approach which is evaluating resistance and propulsion (numerically) of a ship. An introduction to the work which includes the latest developments from the applied research, including those in experimental and CFD techniques, and then provides guidelines for the practical estimation (Model testing). It also includes enough published standard series data for the hull resistance and also for the propeller performance to enable more experimental approaches to making ship power predictions based on material and based on virtual simulation test. (Anthony F. Molland, 2011).

3. GOVERNING EQUATIONS

The billion-dollar question comes into mind that what exactly computational fluid dynamic is looking for? Any problem where CFD applications are needed, all we need is to solve its control volume, velocity field, and pressure fields. More precisely we are always looking for a solution in order to find out u, v, w and p throughout the domain. If the problem is unsteady in nature it will be a function of x, y, z, and t.

3.1.Navier-Stokes Equations – Equations to solve the unknowns

However, in this work, we will be working on Modern Hydrodynamics if we are able to solve the unknown u, v, w and p then we are almost done with CFD calculations. In order to solve these unknowns, we must need to have four equations. For the development of these four equations, we will use the conservation principle on a small control volume.

The very first principle is conservation of mass, the *rate of increase of mass at a given point is mass flux in minus mass flux out,* and this will lead us to the first equation.

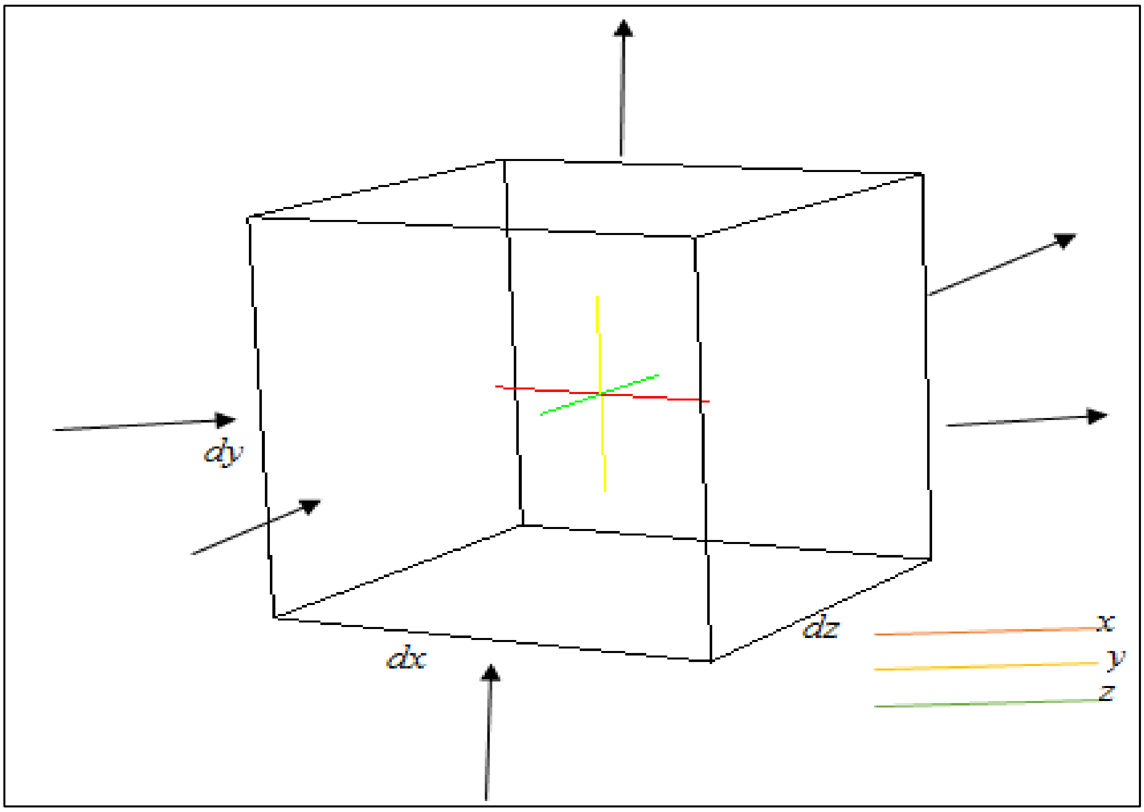


Figure 2 - Differential control volume considered for derivation of conservative equation

The velocity of the fluid entering the control volume from the left side is u. (note that this is the *average velocity* of the plane, not local velocity). The velocity leaving the control volume at the right side is then $u + \frac{\partial u}{\partial x} dx$. Partial derivate are used here although only the derivatives in x directions are relevant. Considering the total volume change dV in a time-step dt, we can have

$$dV = \left(\frac{\partial u}{\partial x} dx\right) dydzdt + \left(\frac{\partial v}{\partial y} dy\right) dxdzdt + \left(\frac{\partial w}{\partial z} dz\right) dxdydt \tag{1}$$

In the compressible flow, this volume change must be zero for any volume and any time step, it can be represented in differential form like this.

$$\frac{\partial p}{\partial t} + \frac{\partial}{\partial x}(pu) + \frac{\partial}{\partial y}(pv) + \frac{\partial}{\partial z}(pw) = 0 \tag{2}$$

This is continuity equation for incompressible flow and this equation can be used in the formulation of the equation of motion.

Remaining three questions can be derived from the conversation of momentum, which can be described from Newton 2nd law of motion. Since the momentum is a vector quantity, there will be three components to it and it will come up with three independent equations.

The total derivative of the velocity u can be written in its partial derivatives so that the formulation of the equation in space yields:

$$\frac{\partial(pu)}{\partial t} + \frac{\partial(pu^2)}{\partial x} + \frac{\partial(puv)}{\partial y} + \frac{\partial(puw)}{\partial z} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(\lambda\nabla.V + 2\mu\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)\right] + \frac{\partial}{\partial z}\left[\mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)\right] + pfx \tag{3}$$

$$\frac{\partial(pv)}{\partial t} + \frac{\partial(puv)}{\partial x} + \frac{\partial(pv^2)}{\partial y} + \frac{\partial(puw)}{\partial z} = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left[\mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)\right] + \frac{\partial}{\partial y}\left(\lambda\nabla.V + 2\mu\frac{\partial v}{\partial y}\right) + \frac{\partial}{\partial z}\left[\mu\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right)\right] + pfy \tag{4}$$

$$\frac{\partial(pw)}{\partial t} + \frac{\partial(puv)}{\partial x} + \frac{\partial(pvw)}{\partial y} + \frac{\partial(pw^2)}{\partial z} = -\frac{\partial p}{\partial z} + \frac{\partial}{\partial x}\left[\mu\left(\frac{\partial v}{\partial z} + \frac{\partial u}{\partial x}\right)\right] + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right)\right] + \frac{\partial}{\partial z}\left(\lambda\nabla.V + 2\mu\frac{\partial w}{\partial z}\right) + pfz \tag{5}$$

These equations are called the Navier-Stokes Equations because the French mathematician Navier and English mathematician Stokes developed this equation independently in the middle of 19th century. (G.Kuiper, 1997)

3.1.1. The approach toward N-S Eq.

Next question comes to mind is how to solve these obtained equations in order to reach what is desired. One can solve N-S equation together and can find out the 4 unknowns having the 4 equations. Actually, in nature, there is no analytical solution exists for their solution because these are highly nonlinear. So the only left is the numerical approach, where instead of solving for general case we will solve the problem for particular case at discreet points.

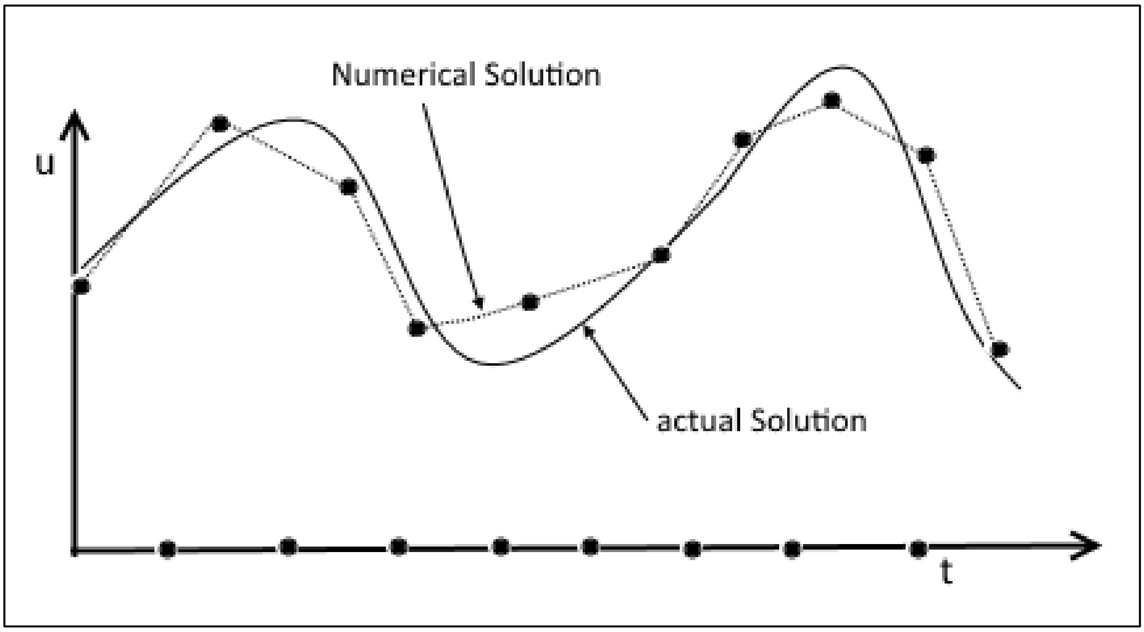


Figure 3 - Numerical Vs Actual solution

Nowadays there are a number of mathematical techniques like Finite Element Method (FEM), Finite Difference Method (FDM) or Finite Volume Method (FVM) available for Numerical Solutions.

As we have mentioned, the exact solution of Navier-Stokes equations is too much of accuracy. It captures every possible minute detail of turbulent flow which is a huge number of data and not very easy to draw the conclusion. But as an engineer, of course, we are not interested in such a solution, what we need is an averaged solution as shown below:

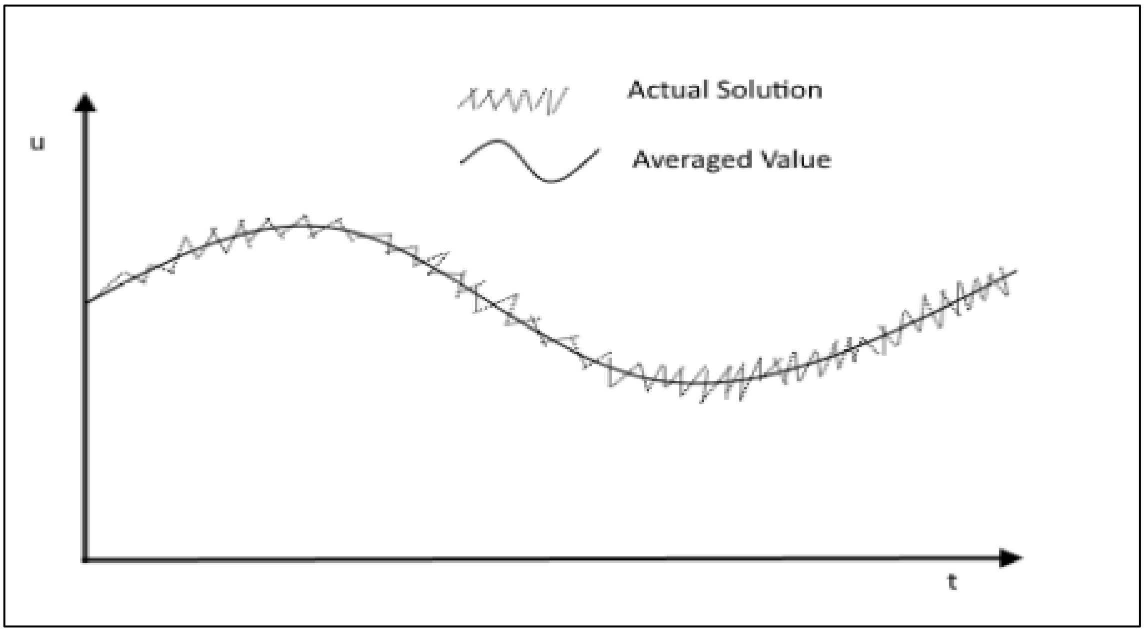


Figure 4 - Actual Vs Averaged solution

From the above figure, we can see that any variable in a turbulent flow can be represented as the sum of mean value and the averaged value (fluctuating value).

To obtain these averaged values, instead of solving N-S equations we can solve something called, Averaged Navier-Stokes equation. N-S equations generated after the averaging operation known as Reynolds-Averaged Navier-Stokes Equations(RANS). We will briefly explain these equations in the forward section.

3.2.RANS Models

Engineers are always interested in knowing just a few properties of a turbulent flow, such as forces distribution on the body and degree of mixing in between two incoming streams etc. In general, Navier-Stokes Equations maybe be solved by numerical Finite volume or Finite Element Methods but direct access to these approach puts a very heavy demand on the resources in order to implement the practical projects. That's the reason special numerical methods based on the Navier- Stokes Equations were developed in order to simplify the problem with more accuracy and decreased the computational efforts. Nowadays, Reynolds Average Navier Stokes Equations(RANS) methods are the most popular one in practical ship hydrodynamic. This good demand approach provides the most suitable and exact simulating conditions since the 2000s. (Larsson & Raven, 2010) Any variable in a turbulent flow field can be demonstrated as the sum of its mean value and the fluctuating value. We can write the variable u in term of

$$u = \bar{u} + u' \tag{6}$$

Where the average value can be represented as follow:

$$\bar{u} = \frac{1}{\Delta t}\int_{t}^{t+\Delta t} u.\,dt \tag{7}$$

A pictorial representation of above equation is listed below.

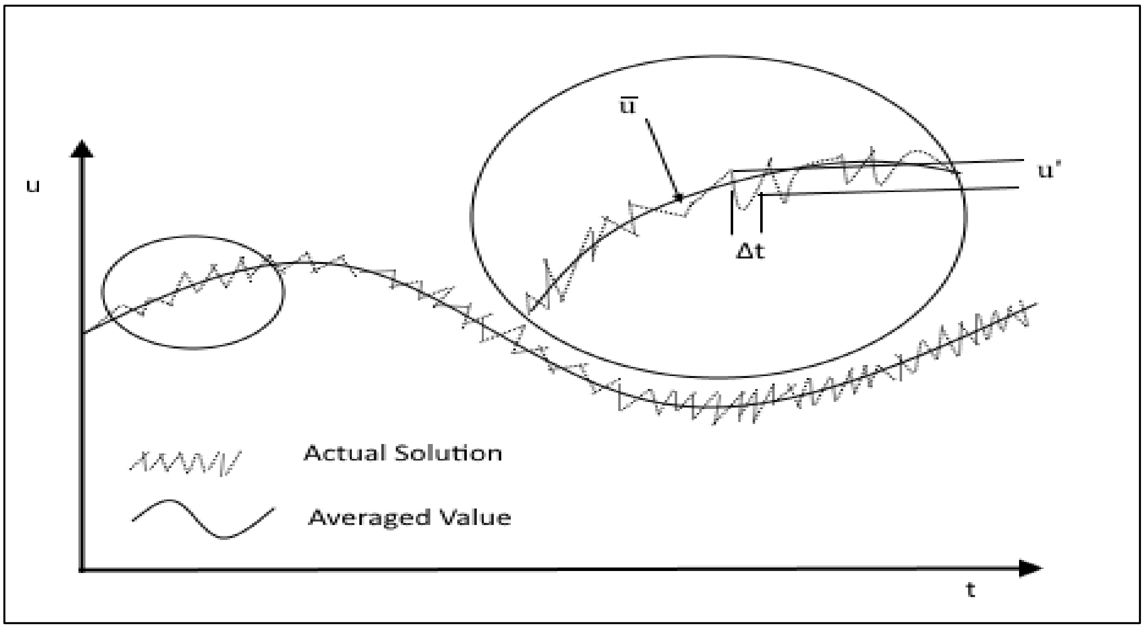


Figure 5 - Pictorial representation of RANSE

Here the interval of integration should be carefully selected. It should be small enough in order to capture any unsteadiness in the flow and big enough to smooth out the fluctuations due to turbulence. RANS are not so difficult to solve numerically. Current CFD packages use mainly 3 numerical methods. They are FEM, FVM, and FDM. (Bakker, 2002)

It is clear that FVM (Finite Volume Method) is the most common method and in the current work, we will be mainly focused on this method as well.

3.3.Finite Volume Method

The finite volume method is a very common approach widely used in CFD codes because it has advantages in memory usage in also in the speed of solutions. Concerning the large problems, it's very efficient in presence of high Reynold numbers turbulent flows. The popularity of the Finite Volume Method (FVM) in CFD is based on its high flexibility for discretization method, directly; without any transformation between the physical and computational coordinate system. FVM assumed a particularly significant role in the fluid flow simulations and its related transport phenomenon. Like other numerical methods which are developed in order to simulate the fluid flow, FVM transforms the set of partial differentiation equations. In addition, adoption of collocated arrangement made it more suitable for the complex geometries. (Rhie & Chow, 1983)

OpenFOAM implements a Finite Volume Method of the 2nd order of convergence with the help of arbitrary unstructured meshes which allows the user to discretize the flow domains if an even very high complex geometry. The finite-volume equations yield the governing equations in the form of:

$$\frac{\partial}{\partial t} \iiint \boldsymbol{Q} \boldsymbol{dv} + \iiint \boldsymbol{F} \boldsymbol{da} = \boldsymbol{0} \tag{8}$$

Where Q is the vector of the conserved variable, F is the vector of fluxes, V is the volume of control volume element and A is the surface area.

The unstructured mesh allows for a very fast, sometimes automatic, mesh generation procedure which is very important for industrial applications. This content describes a very basic detail for FVM in OpenFOAM, FVM, and unstructured mesh topology may be both complexes for CFD.

Finite element method is not used in the ongoing project as it requires greater computational resources and CPU effort than equivalent finite volume method(FVM).

3.4.Stages of CFD Analysis

Any analysis based o CFD can usually be split into five major components; Problem Definition, Mathematical modeling, Pre- Processing and mesh generation, Solving, and Post-Processing, some of these steps must need to be performed multiple times in order to finally reach the desired quality of results. Pragmatic questions that might come up before performing a CFD analysis are:

General Considerations:

- What is to be the conclusion from CFD analysis?
- How the validate will be performed?
- How will the degree of accuracy of results be defined?
- How much time is available for the project?

Thermo-Physics:

- Is the flow laminar, turbulent or transitional?
- Compressible or incompressible?
- Multiple fluid phase or chemical species?
- What is the role of heat transfer?
- Are material properties functions of dependent variables?

Geometry and Mesh:

- Can an accurate representation of flow domain is possible?
- What will happen to the computational domain, deforming or moving, during the simulation?
- How to reduce the complexity without impacting the solution accuracy?

Computational Resources:

- How much time is available for the simulation?
- Availability of distributed resources?
- How many simulations will be enough in order proceed?

These mentioned queries participate to a complete and well accurate CFD analysis. As computers and numerical methods are being very common and popular, it can be seen that direct computations of ship maneuvers and seakeeping are becoming more feasible as well. The open source computational fluid dynamics(CFD)code OpenFOAM is very attractive as a tool for computational ship hydrodynamics since it has good capabilities for free surfaces flow problems and it is good enough to answer all the mentioned questions in a CFD analysis. OpenFOAM is developed and released by Open CFD Ltd at ESI Group and is one of the most popular open sources, anyone has the access to the source code, CFD packages today. However, OpenFOAM lacks the ability to perform arbitrary motions which are needed to move ships, rudders, propellers, and other appendages simultaneously as required for free model simulations of ships. (Zhirong, Decheng, & Pablo, 2015)

4. OPENFOAM

OpenFOAM (Open Field Operation and Manipulation) is a special computer software, an open-source software, for CFD simulations. Its complete version is freely available to anyone (distributed under General Public License - GPL) and hence in this study, OpenFOAM is used. There are few good reasons that make me confident to consider OpenFOAM for my CFD simulations. OpenFOAM is generally considered to be the best alternative to the commercial solutions.

Although OpenFOAM is mostly used for CFD problems it can also be used in almost all continuum mechanics problems where Finite volume method (FVM) can be applied. The program runs on Linux system(recommended) but Windows versions are also available.

OpenFOAM is supplied with pre- and post-processing environments. The interface to pre- and post-Processing's are themselves its utilities so that consistent data handling across all the environment make it more useful. The overall structure of OpenFOAM is shown below:

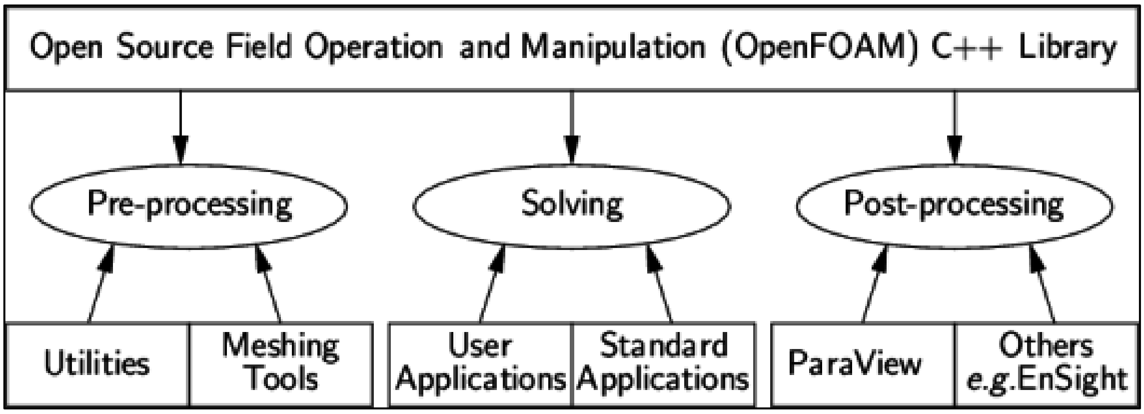


Figure 6 - Overview of the OpenFOAM structure. Available from http://en.wiki.laduga.ru/1_Introduction [Accessed 08 October 2016].

Syntax: One distinguishing feature of OpenFOAM is its syntax for tensor operations and partial differential equations (PDI). For example:

$$\frac{\partial \rho U}{\partial t} + \nabla . \emptyset U - \nabla . \mu \nabla U = -\nabla p \tag{9}$$

Can be represented by the code in the following figure.

```
solve
(
Fvm::ddt(rho, U)
+fvm::div(phi, U)
-fvm::laplacian(mu, U)
==
-fvc::grad(p)
);
```

Figure 7 – Simple representation of OpenFOAM (Code)

Which shows that it's easily readable language. This syntax enables users to create custom solvers with the relative ease. However, codes are becoming more challenging with the increasing depth of the OpenFOAM library and heavy use of template metaprogramming.

OpenFOAM comes with different tools, meshing, e.g. BlockMesh, and for pre- and post-processing, e.g. ParaView. The software is licensed under the General Public License, as an open source library, advantages are that the user is free to extend and change the library as wished. This provides a robust and very flexible development environment for a viscous model. All the processes are run parallel so the user can also maximize the computer power accordingly. (J. Greenshields, 2015)

Background: Originally, OpenFOAM was developed at Imperial College, London, in the late 1980s. the main task was to find a more flexible and powerful simulation platform than the existing solvers for computations. At that time, it was firstly written in FORTRAN, as C++ also was developing very fast and was providing a lot of assist to many computations, the development of OpanFOAM took large steps. Later on, Nabla Ltd. Commercialized this development under the name of FOAM. In 2004 the package was released, under the new name OpenFoam. OpenFOAMis now developed and distributed by The OpenFOAM Foundation, which holds the copyright to the source code, still as an open-source toolbox. (Revolvy, 2016)

4.1.OpenFOAM structure

It is very important to distinguish between the actual mesh geometry and the geometry that comes out from CAD program. Here we will briefly explain an overview of how the actual mesh is stored in the file system of OpenFOAM. There are three basic directories in OpenFOAM, i.e. 0, constant and system.

0: containing individual files of data for particular fields, e.g. velocity*(u)* and pressure*(p).* The data can be either, initial values and boundary conditions that the user must need to define the concerned problem, or results are written to file by OpenFOAM.

Constant: contains the full description of mesh in subdirectory *poly mesh* and properties for the application concerned, e.g. *transportProperties.*

System: This directory is associated with simulation procedure and contains at least the three following files: control Dict where the run control parameters are defined including start/end time, time step and parameter for data output; *fvschemes* where discretization schemes used; and, *fvSolution* where equation solvers, tolerances, and another algorithm are set for run. (Puig, Gamez, & Gustavo, 2014)

As long as static meshes are being used, the computational grid is always stored in *constant/polyMesh* directory. Word static means that the meshes are not changing over the course of simulation either through point displacement or connectivity changes.

Users having worked with CFD codes may be missing the per-cell addressing because they mostly worked with the codes which are based on structured meshes. OpenFOAM constructs the meshes on the per-face basis rather than per cell due to the untrusted Finite Volume Method(FVM). (Puig, Gamez, & Gustavo, 2014).

When solved, OpenFOAM creates a folder for each time step where the flow properties, i.e. velocity, pressure and phase fraction are stored in different files. These files contain the respective values at each grid point. Additional properties, e.g. vorticity can be calculated based on these values. There are many different solvers in OpenFOAM, which is suitable for different types of problems, e.*g interFoam* for two-phase flow, *InterDyMFoam* for dynamic flow, *pisoFoam* for one-phase incompressible flow, *dieselFoam* for diesel spray etc. (Magnus, 2013)

4.1.1. Solvers

OpenFOAM has an extensive range of features, it includes tools for the meshing in and around complex geometries, data processing, visualization and more. OpenFOAM solvers include:

1. Basic CFD solvers;
2. Incompressible flow with RANS and LES capabilities;

3. Compressible flow solvers with RANS and LES capabilities;
4. DNS and LES;
5. Multiphase flow, solvers;
6. Solid dynamic solvers;
7. Electromagnetic solvers;
8. Buoyancy-driven flow driven solvers;
9. Particle tracking solvers and,
10. Direct Simulations Monte Carlo Solvers.

Besides the standard solvers, OpenFOAM's syntax allows users for the creation of custom solvers. In the current work, we will be mostly working on two solvers that are InterFOAM and InterDyMyFoam.

InterFoam: (Fixed)

Solver for two incompressible, isothermal immiscible fluids, capturing the interface by using Volume of Fluid (VOF) method. In the presented work InterFoam solver has been used for the simulations from 0 to 5 second in order to capture effect while being fixed.

InterDyMFoam: (Dynamic)

Solver for two incompressible, isothermal immiscible fluids, capturing the interface by using Volume of Fluids (VOF) with optional changes in mesh motion, mesh topology, and adaptive re-meshing. InterDyMyFoam has to be used from 5 to 29 second in order to get the dynamic effect of whole simulations.

4.2. Geometry

Importing a geometry which has been generated in any external computational aided design (CAD) software is a common task for any CFD engineer and while using OpenFOAM this is usually performed with *snappyHexMesh*. The very basic concept, for now, is to understand is that it deals with the import of Stereolithographic (STL) files.

4.2.1. *STL Format*

A file format that is able to store the surfaces of geometries in *triangulated* form. Both binary and ASCII encoded files are possible but here we will be wringing with ASCII encoding for the sake of simplicity. STL file is perhaps the single most efficient item of any 3D printing workflow as it contains the 3D model.

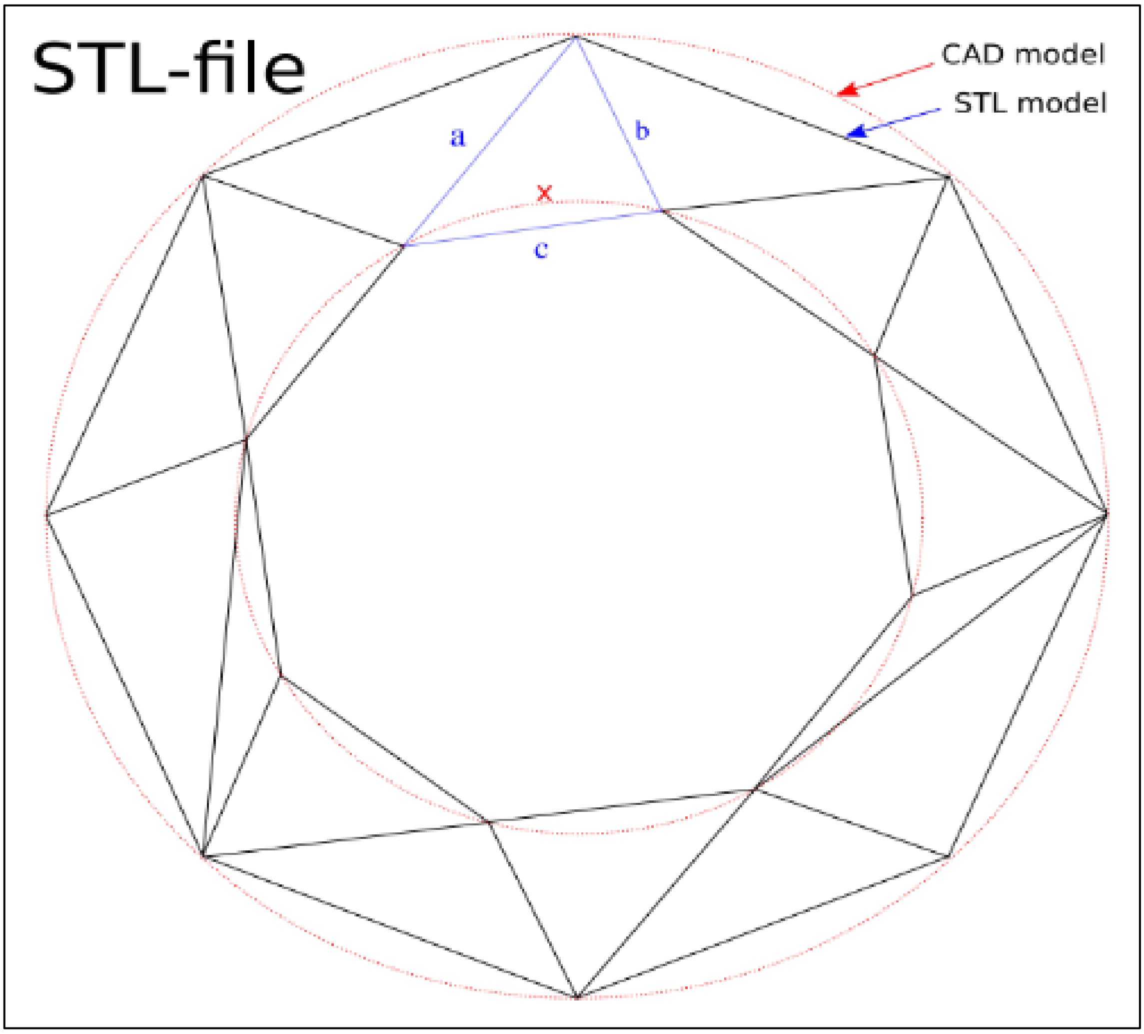


Figure 8 - STL vs CAD Model. Available from https://en.wikipedia.org/wiki/STL_(file_format)#/media/File:The_differences_between_CAD_and_STL_Models.svg [Accessed 10 November 2016].

As an example, an STL file is shown in the next figure.

```
solid      KCS_MOERI

  facet normal -0 -0 -1

    outer loop

      vertex 0.015123417600989342 -1.1594028163339253e-17 0

      vertex 0.014204167760908604 -1.1594028163339253e-17 0

      vertex 0.014204167760908604 0.00015773337509017438  0

    endloop

  endfacet

  facet normal -0 -0 -1

    outer loop

      vertex 0.015123417600989342 -1.1594028163339253e-17 0

      vertex 0.014204167760908604 0.00015773337509017438  0

      vertex 0.015123417600989342 0.00015773337509017438  0

    endloop

  endfacet

  facet normal -0 0 -1

    outer loop

      vertex 0.014204167760908604 0.00015773337509017438  0

      vertex 0.014204167760908604 -1.1594028163339253e-17 0

      vertex 0.013284917920827866 -1.1594028163339253e-17 0

    endloop

  endfacet
```

Figure 9 – An example of STL file, the very three objects from KCS_MOERI geometry

As mentioned in the script, only one solid is defined as KCS, geometry we will be working, though. An STL file can have multiple solids as well and can be defined one after the other. One can see that each of the composed surfaces has a normal vector and three points.

An advantage of STL format is its simplicity. Even the most complex design can be reduced to the simple geometry: that it always contains the *triangulated* surface mesh.

The disadvantage of using ASCII STL format is that the file size grows rapidly with the increasing resolution of the surface. Edges are not included because only triangles are stored. Due to this reason, identifying and extracting feature edges from an STL formats is sometimes a challenging task.

4.3.Meshes

Before getting into the details of OpenFOAM, some notations have to be put in mind in order to understand its simplicity. A *geometry* in CFD context is basically a 3D representation of the flow, a *mesh* can have multiple meaning but keeping the focus on the current topic, a mesh is 3D *volume mesh*. There are small variations to this term like *surface mesh* which is discretization of the surface.

4.3.1. Mesh Generation

In general sense, the main task of a mesh generator is to generate the *polyMesh* files as described before.There are various kind of mesh generators specially designed for the OpenFOAM and are spread into the main two development branches.

1. *BlockMesh, snappyHexMesh, foamHexMesh, foamyQuadMesh*
2. *cfMesh*

there are some remaining tools, extrudeMesh and extrud2DMesh but will not be explained here because they are not widely used. blockMesh and snappyHexMesh will be described here briefly because the topic was covered by using these two (mainly).

BlockMesh: When calling the executable *blockMesh* the *blockMeshDict* is read (automatically) from the *constant/polyMesh* directory where it must be present. Generation of grids is very difficult and sometimes is impossible because *blockMesh* generates the block-structured hexahedral meshes which are then converted to the arbitrary unstructured format of OpenFOAM. Hence, only simple meshes are generated by using this and the further discretization of actual geometry is shifted by using the *snappuHexMesh*, which makes the *blockMesh* a great tool.

As an example, how the blockMeshDict is set up correctly, KCS ship hull has been discretized and a prepared case can be found in Annex-II.

In order to explain the blockMeshDict, a simple table with keywords explanation can be seen below:

Table 1 - Keywords used in blockMeshDict.

Keyword	Description
convertToMeters	Scaling factor for the vertex coordinates (Usually set to 1)
vertices	1 - Prescribe the vortex locations for blocks 2- Cartesian coordinates (x, y, z) 3 - Each vertex in list = 1 index
Edges	1 - Prescribe block topology and mesh settings 2 - Edge is straight if not listed 3 - Arc and Spline available
Block	Ordered list of vertex labels and mesh size
Patches	Prescribe surface patches for boundary condition
mergePatchPairs	1 - Merge multiple (separated) blocks into one mesh 2 - One-to-one correspondence if blocks are not listed

The directory itself consists of one keyword, convertToMeters， and four sub-directories (Vertices, edges, blocks, patches).

SnappyHexMesh: As compared with blockMesh, snappyHexMesh may not require that much of work so the user has the less control over it. SnappyHexMesh is a mesh generator that takes an already existing mesh, usually created with blockMesh, and forms it into the mesh we want (Montorfano & Federico , 2015). To do so, it requires:

1. A well-defined dictionary, namely system/snappyHexMeshDict;
2. Good geometrical definitions, such as STL/OBJ files with well-defined surfaces and,

3. eMesh feature edge files.
4. parallel execution is available by using mpirun in a terminal window (Used in the current work)

Execution flow of SnappyHexMesh can be split into 5 steps. A simple explanation of its working flow can be seen below.

1. BlockMesh:
    - Creation of base mesh;
    - Custom made by using blockMesh.
2. refineMesh:
    - Refinement of base mesh;
    - Surface refinement i.e. feature lines and for curvature
    - Refinement of volume in every shape (closed, surfaces, geometric shapes).
3. CastellatedMesh:
    - Remove unused cell;
    - User specification refinement;
    - Resulting mesh consists only of hexahedrons.
4. SnapControls:
    - Snap mesh to surface;
    - Topology changes from hexahedron to polyhedron;
    - Cell near the surfaces may delete or merged.
5. Add layers:
    - Additional cells on geometry surface;
    - Pre-existing cells are moved away from geometry.

A snappyHexMesh file has been added to the Annex-II with detailed comments for each step, the reader is highly recommended to go through in order to have a very keen idea about each term used in this utility.

4.4.OpenFOAM Advantages

The main advantages of OpenFOAM are:

1. No license cost;
2. Easy syntax for partial differential equations(PDI);
3. Polyhedral grid capabilities (Unstructured);
4. Automatic parallelization;
5. Commercial support and training provided by the developers;
6. C++ Code, easy to read;
7. Results with more accuracy;
8. Built-in post-processing (Paraview);
9. Due to the open source nature and GPL one can keep using OpenFOAM software forever;
10. A great time and cost reduction;
11. Unmatched insight into the concerned problem that may be difficult to prototype or test through experimentation;
12. Ability to foresee implications of design changes and accordingly optimization;
13. The resistance prediction simulations are carried out for a wide range of applications and conditions;
14. Reducing risk i.e. building confidence and,
15. The best for last- with OpenFOAM, 40% reduction in cost associated with CFD on average after its development.

5. COMPUTATION AND CAPACITY

With the increasing power and popularity of modern computers, CFD is becoming a possible tool for the engineers in order to approach the desired results with much accuracy as expected. Almost all computations can be executed in parallel as standard to take the full advantages of today multi-core processor computation technology. *"We are literally at a significant point in the history. A 3rd branch of the scientific method, computer simulation, is emerging as a day-to-day tool. It is taking its place next to the experimental development and mathematical theory as a way to new discoveries in the science and engineering"*. This was the part of speech of John Rollwagen, Chairman, and CEO of Cray Research, to the opening session of supercomputing 89. In general, CFD problems have a high data cost, which requires several cores for the stimulating. Talking about OpenFOAM, it allows the user to have full control of data usage. In the current work, Computing Resources used were provided by the National Centre for High-Performance Computing of Turkey (UHeM). Approved details for the project are listed below:

Topic	• Ship Resistance Computations using OpenFOAM
Project Name	• shaur
Capacit y	• 10,000 CPU kernel-hours
Grant Number	• 4004202016
Under	• Istanbul Technical University (ITU), Turkey

Figure 10 – UHem Project details

All the computations have been done while accessing the supercomputer by using NX Client with a safe VPN access.

Istanbul Technical University cluster has been installed with LSF (Load Sharing Facility) for computation platform.

The solver used was InterDyMyFoam, typed directly into the terminal (Linux) in order to execute the simulation. This task always based on local time stepping for the solution of two incompressible, isothermal and immiscible fluids (OpenFOAM 2011). In short, we can say that this is the brief process for the multiphase situations (Setup and Solved cases). Some of the simulations are not performed on supercomputer so it required a lot of time and computer storage to complete. Therefore, they were run in parallel using mpirun command, where 1-6 nodes were used. Each node had 8 cores with 16GB of RAM available. Simulations for these cases took roughly 350 CPU hours.

5.1.LSF

LSF is a workload management product from Platform Computing Corporation. it includes LSF Batch (Job processing system), LSF JobScheduler (Integration of heterogeneous server), LSF MultiCluster (resource sharing), LSF Make (dispatched task to multiple hosts), and LSF Analyzer (produce statistical reports) and all are running on top of the LSF Base system, a software Upon which all other LSF products are based.

Below diagram illustrates the components in an LSF desktop support environment and cluster concept with its execution procedure.

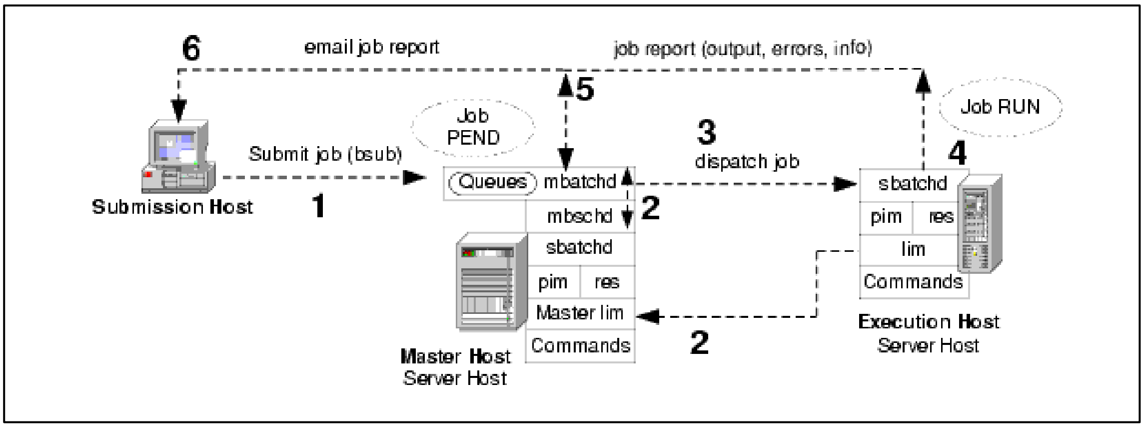


Figure 11 - Cluster concept of LSF Available from https://assoc.ictp.it/icts/manuals/lsf6/images/A_termsa.gif [Accessed 08 December 2016].

Below are some of its main features:

1. Network of heterogeneous computers as a single system;
2. Automatically selection of hosts in order to perform the required task;
3. Maintains full control over the jobs;

4. Able to perform both sequential and parallel jobs and,
5. Transparently running of software which is not available on the local host.

LSF understanding needs a very continuous focus and here will not be explaining that much but the reader is highly recommended to get a vast knowledge about LSF by going through its user guide (LSF Batch User's Guide, 1998).

Submitting Jobs: In order to submit a job on lsf client bsub command is used. A script of one LSF job submission used for the simulation of KCS can be found in Annex-III with detailed comments.

The overall life cycle of lsf job execution can be seen in the below figure.

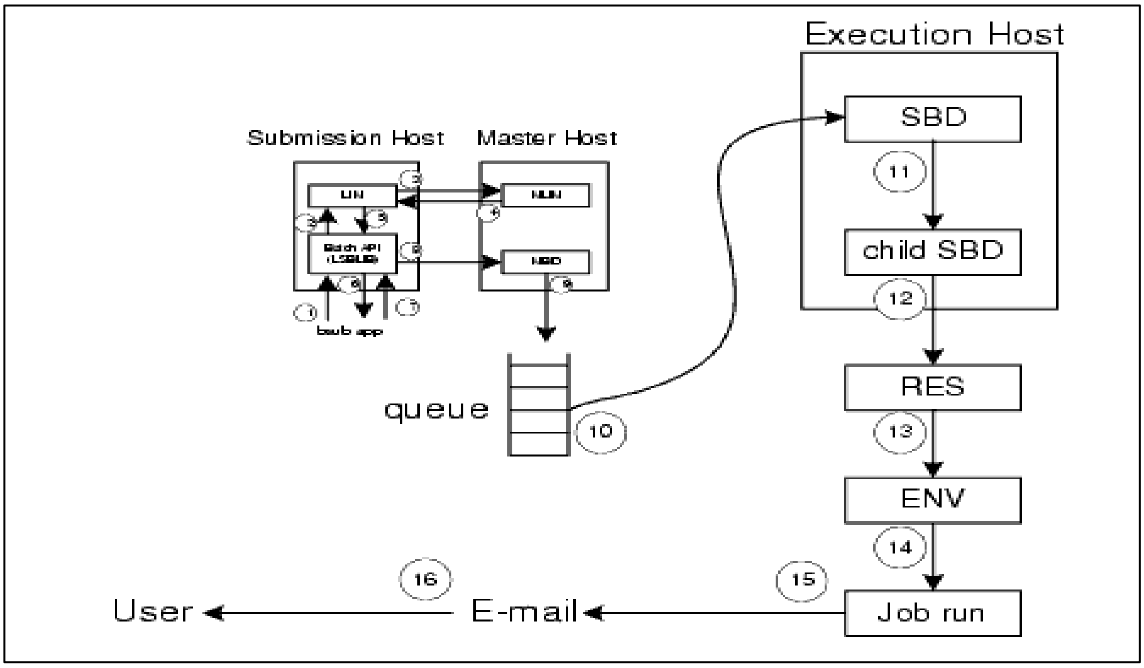


Figure 12 - LSF job life cycle

As a summary of repeated work done by using LSF scripts, here we are listing some very often used LSF commands in order to perform our computation.

Table 2 - commands used in LSF

Command	Description
Bsub	Submit of jobs
Mbatchd	Sends the jobs for scheduling
mbschd	Evaluates jobs and makes scheduling decisions based on: 1 - Job priority 2 - Scheduling policies 3 - Available resources
bstop	Suspends a job
bresume	Resume a suspended job
bgadd	Creates job groups
bgdel	Delete job groups
btop	Moves a pending job relative to the first job in the queue
bjobs	Displays information about jobs
bjgroup	Displays information about job groups

5.2.Scaling Rules

When two models of different size are compared, the comparison should be made in similar conditions, which means that various forces involvement should be kept in the same proportion. For model tests, there are some special rules called scaling laws. When the models at different scales are compared, each size of the model with same hull form has different resistance. As has been being seen that the resistance can be made non-dimensional as:

$$\boldsymbol{CT} = \frac{\boldsymbol{R}}{\frac{\boldsymbol{1}}{\boldsymbol{2}}} * \rho * \boldsymbol{Vs^2} * \boldsymbol{S} \tag{10}$$

Where:

R = resistance in N;

ρ = specific mass of water in $\frac{kg}{m^3}$;

Vs = Ship speed in m/sec and,

S = wetted surface area of the ship in³.

In order to represent the frictional resistance coefficient as a function of ship or model speed in a single curve, the velocity Vs has to be expressed non-dimensionally, Reynolds number,

$$\boldsymbol{Rn} = \frac{\boldsymbol{Vs}}{\boldsymbol{v}} \tag{11}$$

Similarly, wave resistance coefficient reduces to the one function for all the size of ships when plotted as function of Froude number,

$$\boldsymbol{Fn} = \frac{\boldsymbol{Vs}}{\sqrt{\boldsymbol{gL}}} \tag{12}$$

Until now, these parameters have been determining experimentally and then proved a well, one can also approach to the desired task by dimensional analysis as well.

The situations are similar when the non-dimensional parameters are kept same. The non-dimensional parameters, therefore, act as scaling laws. Each non-dimensional parameters leads to a scaling law:

Maintain the Froude number at the model scale means that:

$$\frac{\boldsymbol{Vs}}{\sqrt{\boldsymbol{gLs}}} = \frac{\boldsymbol{Vm}}{\sqrt{\boldsymbol{gLm}}} \tag{13}$$

Simplification for the model scale leads to

$$\boldsymbol{Vm} = \frac{\boldsymbol{Vs}\sqrt{\boldsymbol{Lm}}}{\boldsymbol{Ls}} = \boldsymbol{Vs}/\sqrt{\boldsymbol{\alpha}} \tag{14}$$

Where the index m represents the model and s for the ship.

5.2.1. Scale Effect

It is not possible to maintain the scaling rules with 100% accuracy. Deviation in scaling law results in dissimilarities between the model and full scale and these dissimilarities are call scale effect. A deviation in Rn will result in differences in the regions where viscosity is important so in the boundary layer and a deviation in Fn will result in dissimilarity of a wave system.

These effects are a major source of problems in model testing.

6. COMPUTATIONAL STUDY FOR KCS

Accurate assessment of the ship resistance is the fundamental problem in naval hydrodynamics. Towing tank techniques are very common in order to determine the ship resistance and propulsion but the computational cost is way far. In order to approach within a reasonable economic domain, flow around a propeller geometry and hull, numerical simulations are used for the commercial needs. Basically, it is essential to develop and validate a numerical model for calculating the resistance components. This requires the need of a free surface model that allows the wave pattern so the resistance can be accessed. In order to capture the form drag and frictional resistance, accurate modeling of the boundary layer growth is very important. Extensive research has been performed in this area and different CFD workshop has also been documented. SVA container ship KCS, one of the most common hull geometry is used in today research, has been selected for the current work. The main reason for using this hull is that their test data exist for different Froude numbers.

6.1.KCS

The KRISO Container Ship (KCS) is a well-known case for which measurements have been made in Korea and Japan (Kume et al. 2000). KCS hull form was very first selected for a modern ship case at the Gothenburg workshop 2000.KCS hull form was conceived to provide data for both physics and CFD validation fo a modern container ship. Various kind of experimental tests has been performed at Korea Research Institute for Ship and Ocean Engineering (Now MOIRI) to obtain the resistance, free surface waves, and mean flow. KCS hull is characterized by its long bow (bulbous)) and extending stern which makes a new design but at the same time produces complex flow behind the Hull and Wakefield. As its data is open to everyone, after a refine refinement of its hull, we will start the simulations in order to have better results. KRISO Container Ship (KCS) is presented below:

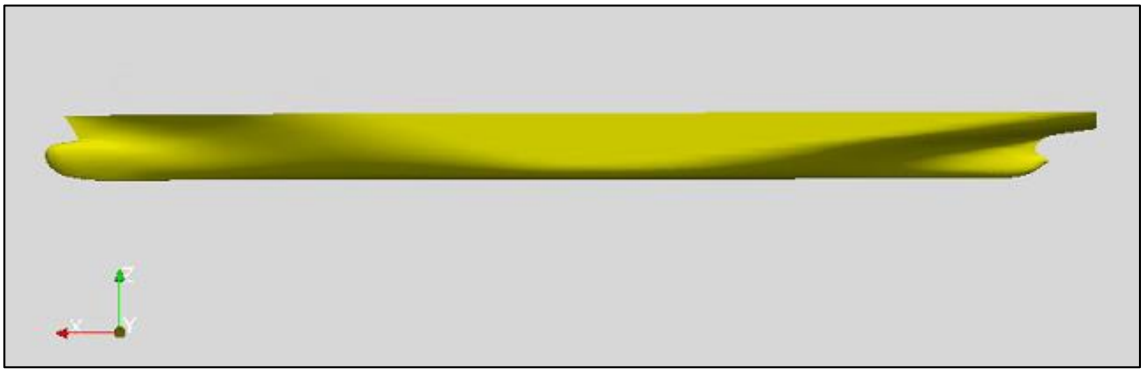


Figure 13 - KCS Hull (STL geometry)

KCS is one of three hull forms explored in Gothenburg 2010 workshop in numerical ship hydrodynamics. Ship propulsion test has been performed at ship research institute (now NMRI) in Tokyo and has been present in the Proceedings of CFD workshop Tokyo in 2005.

The main characteristic of the hull form is present below:

Table 3 Characteristic of KCS Hull

Main Characteristic	Full scale	Model scale (MOERI)
Length of water line, Lwl, [m]	232,5	7,357
Length between perpendiculars, Lpp, [m]	230	7,2786
Ixx	0,40B = 138,98	0,40B = 138,98
Izz	2730,02	0,25Lpp = 2730,02
Beam, B, [m]	32,2	1,019
Draft, T, [m]	10,8	0,3418
Depth, D, [m]	19	0,6013
Volumentric Displacement [m^3]	52030	824,5 kg
Hull wetted surface, S, [m^2]	9424	9
Speed, v	24 knots	2,196
Froude Number, Fn	0,26	0,26
Block Coefficient, C_B	0,651	0,651
Midship Section Coefficient, C_M	0,985	0,985

After a fine refinement of KCS, we will first start with the resistance part.

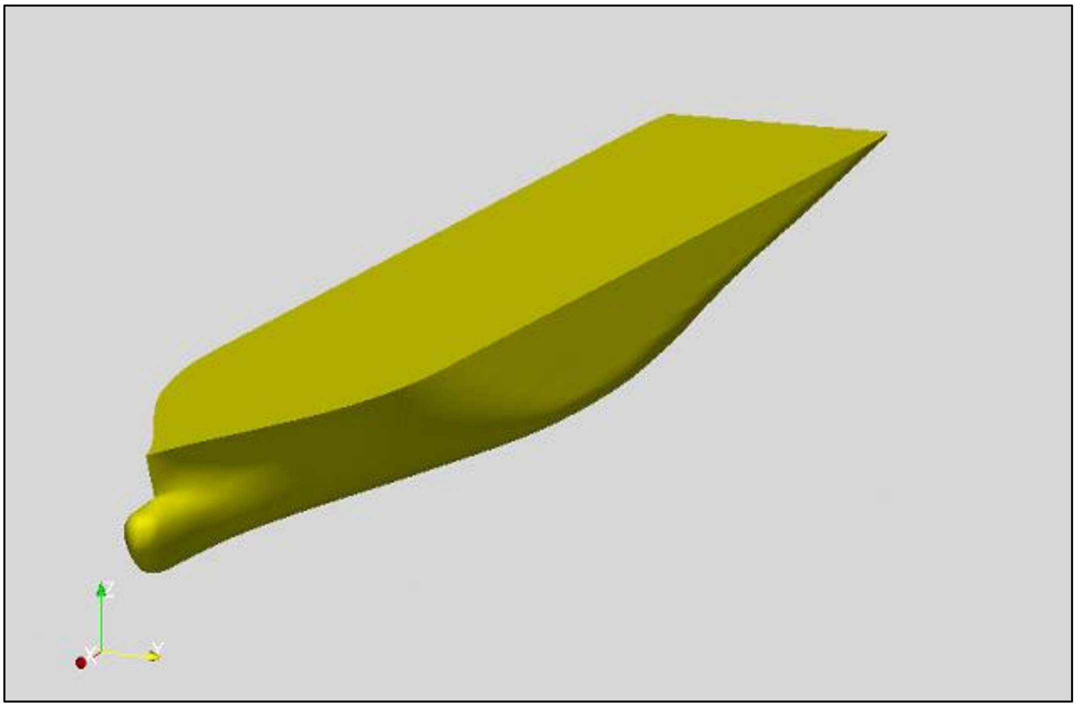


Figure 14 - 3D view of KCS Hull-1

KCS geometry is available online for any user who is willing to work on, All we did is the good refinement of its edges and then present it in the STL format which is user-friendly in terms of OpenFOAM work. Refinement of KCS is done by using Rhinoceros software.

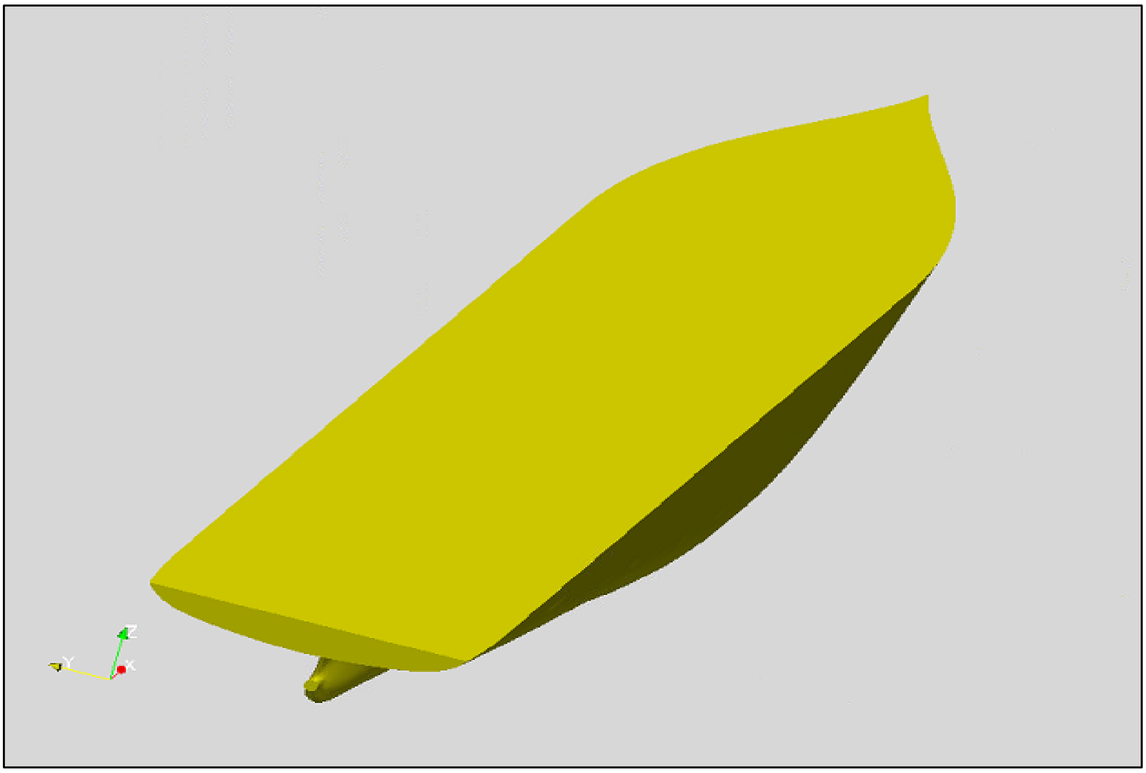


Figure 15 - 3D View of KCS Hull-2

In order to use the KCS hull form to compare the results with Delta Ship, we will first investigate the KCS hull itself and will compare two turbulence models (K-omega and SST-omega), results will be compared with literature in order to select one best turbulence model for the Delta Ship.

6.2.Numerical Mesh (KCS)

The two important files used in OpenFOAM to create the mesh are the blockMeshDict (constant folder) and the snappyHexMeshDict (system folder). The blockMeshDict creates by namesake 3D meshes that are blocks or cubes. An example can be seen in Annex. The mesh that is generated contains equal sized cells throughout that make up the discretized grid.

Table 4 - Mesh Status

Mesh Status	
Points	5223327
Faces	14695616
Internal Faces	14440717
Cells	4743873
Faces per cell	6,14189
Boundary Patches	7
Point zones	0
Face zones	0
Cell zones	0

This whole data is defined precisely in a polyMesh folder (Annex-IV) and different types of cell used in simulations are listed in the table below:

Table 5 Overall number of cells of each type (KCS)

Types of Cells	
Hexahedra	4387903
Prisms	10837
Wedges	0
Pyramids	0
Tet Wedges	19
Tetrahedra	0
Polyhedra	345114

The snappyHexMeshDict utility is used in order to mesh an obstacle internally into the blockMesh which was created previously. The container ship here is the obstacle and must be in STL form. This function is very useful to create a new patch that is the surface of the hull. Below figures show mesh around the bow and stern of the KCS Hull.

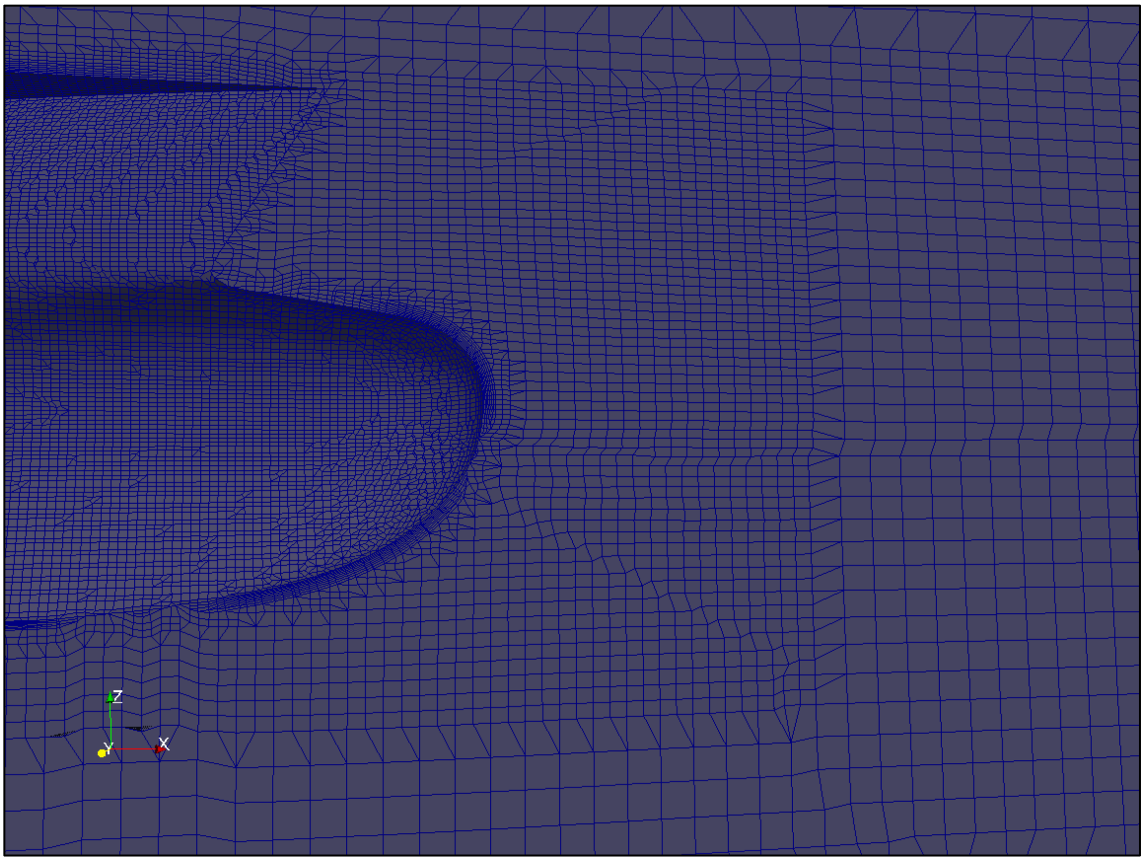


Figure 16 - Mesh around the Bow

As it can be seen from the figure above, a very coarse mesh is applied near the bow and stern for a better capturing of surface effect.

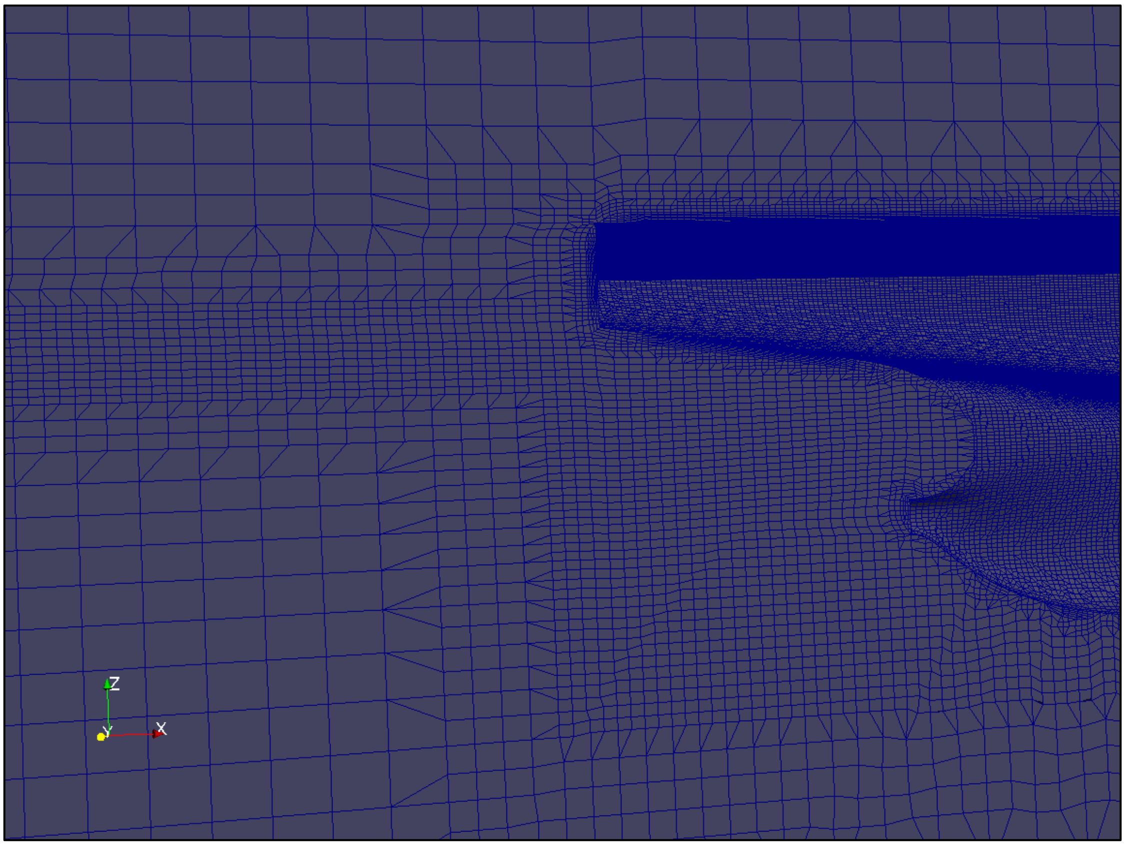


Figure 17 - Mesh Around the Stern

6.3.Fluid Properties

Prediction of the free surface, the interaction between water and air, can often become unstable. hence, an appropriate technique is chosen in order to make sure it has done (meshes) this section more precisely and in a right way. As we are dealing with the multiphase, VOF Method, a good method of modeling ships that can produce breaking of waves, because in general it is mostly applied for two immiscible fluids to calculate the interface interaction throughout the simulation.

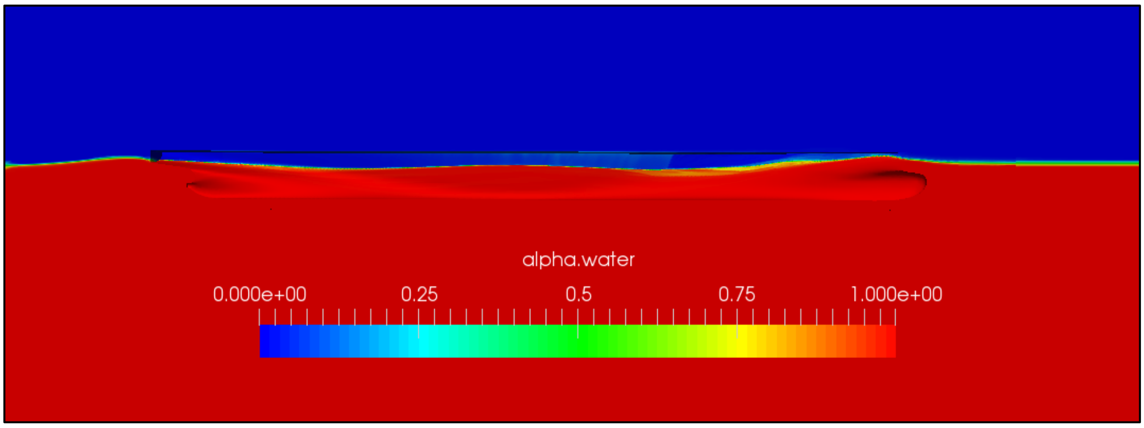


Figure 18 - Alpha water distribution

For values of 0 the fluid is air and for values of 1, the fluid is water. Anything between this is a mixture of the two and hence there will be an interface (Zhang, Hui, Song Ping, & Feng, 2006).

The utility setFields was used to calculate the values of alpha, specifying a box that is all of a value 1, else was set to 0. The table below has the fluid properties as specified in the dictionary transportProperties.

Table 6 - Fluid Properties

Fluid	ν (m^2/s)	ρ (kg/m^3)
Water	1,04E-06	998,4
Air	1,85E-05	1,2

6.4.Validation and verification for KCS hull form

It is not possible to calculate the resistance of full ship directly and the only method leads us to get through this approach is model test calculation. It has observed and seen that the measured calm water resistance is usually decomposed into its various components, most of them cannot be calculated directly. A comprehensive resistance calculation flowchart decomposed by Larsson and Baba in 1996.

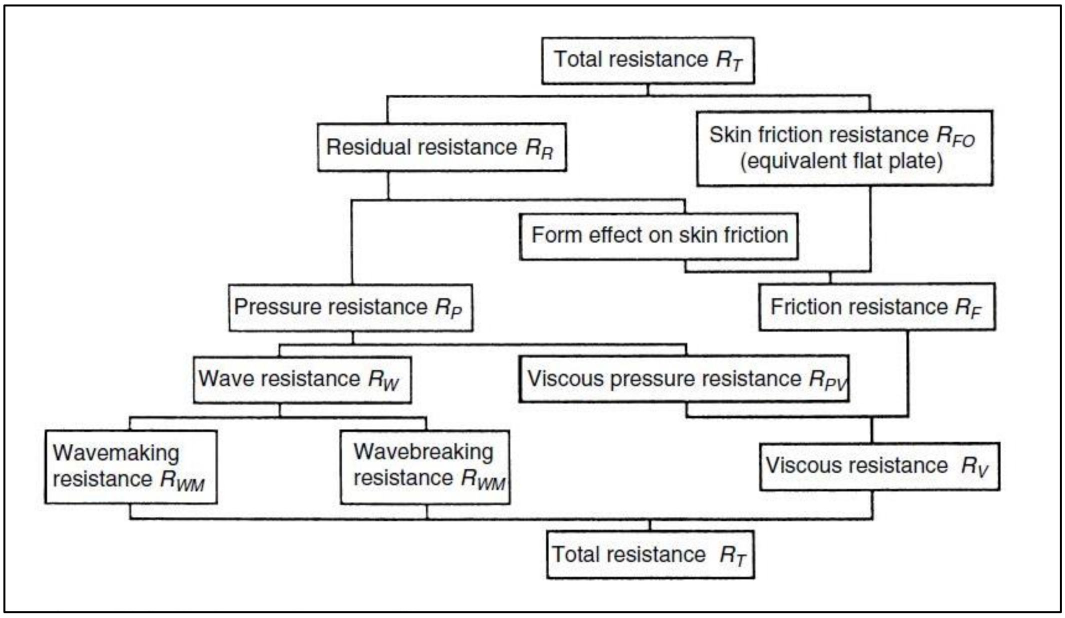


Figure 19 - Resistance decomposition flow chart

One of the most competent recommendations in the field of naval architecture for CFD (Validations and Verifications) were developed within an international workshop on the numerical prediction of ship viscous flow. Gothenburg-2000, three Modern Hull forms (KCS, DTC, KVLCC2) were introduced with reliable experimental data for the validation cases. The most general case among them is KCS, Used in the current work.

6.5.Introduction to Turbulence Modeling

Since the 19th century, to find a suitable simulation model which describes the turbulence phenomenon perfectly has proven to be a very difficult. In any fluid flow problem, engineers need many ways to simulate the turbulent flow in order to optimize their designs with respect to the real-world problem. A number of empiric turbulence models have been created to help engineers so they can find the best model to fit their system of study, but to apply this procedure it could take a lot of trials, possible errors, and physical testing techniques.

Let’s start by assuming the flow over a flat plate shown in the figure below. The uniform velocity fluid hits the leading edge and a laminar boundary layer begins to develop. The flow in this area is very much predictable. After some distance, some oscillations begin to develop in the fluid field, and the flow begins to turbulence and hence becoming fully turbulent.

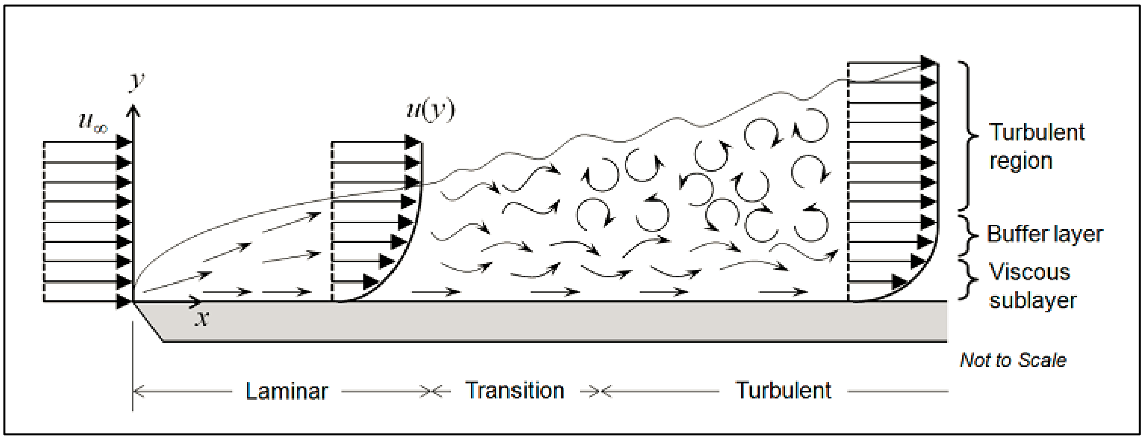


Figure 20 - Flow over a Flat Plate Available from https://cdn.comsol.com/wordpress/2013/09/Flow-of-a-fluid-over-a-flat-plate.png [Accessed 03 March 2017
].

The transition between these three regions is always defined in terms of the Reynolds number. Let’s assume that the fluid under consideration is Newtonian means viscosity is constant with respect to shear rate.

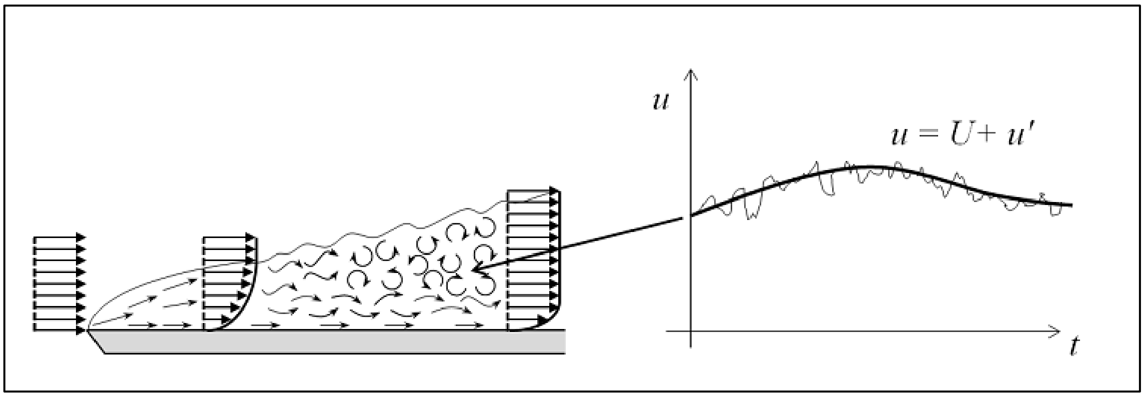


Figure 21 - Accurate prediction of flow behavior Available from https://cdn.comsol.com/wordpress/2013/09/Using-a-Reynolds-Averaged-Navier-Stokes-formulation.png

In the laminar region, the flow can always be predicted by solving the steady-state Navier-Stokes equations, which predict the velocity and the pressure fields. But when the flow begins to pass through transition region, small oscillations appear in the flow and at this stage, it’s not possible to assume that the flow is invariant with time. Hence, it is necessary to solve the problem in the time domain.

One of the most concern problems in turbulence modeling is the accurate prediction of flow separation from a smooth surface. Standard two-equation turbulence models often fail to predict the onset and the amount of flow separation under adverse pressure gradient conditions. This is an important phenomenon in many technical applications. Separation prediction is important in many technical applications both for internal and external flows

Ther are several different formulations techniques (models) for solving turbulent flow problems like the L-VEL, yPlus, Spalart-Allmaras, k-epsilon, k-omega, Low Reynolds number k-epsilon, and Menter k-omega SST models. To Learn about why we want to use these various turbulence models and how to choose between them, and how to use them effectively we will perform various speeds simulations for KCS ship hull for the two different turbulence models and then a comparison will be drawn upon their results with a brief discussion.

6.5.1 Standard K-Omega

In CFD, the k–omega (k–ω) turbulence model is a common two-equation turbulence model used as a closure for the Reynolds-averaged Navier–Stokes equations (RANS equations). It is used to predict turbulence by two partial differential equations for two variables, k (Turbulence kinetic energy) and ω (specific rate of dissipation). (Wilcox, 1988)

Simulations were performed at three Froude speeds;0.22,0.26,0.28. in order to keep an eye the ongoing simulations a simple Python code(Annex-V) is written and hence an overall process of the simulation was being monitored. Figures below shows the overall simulations results with respect to their time.

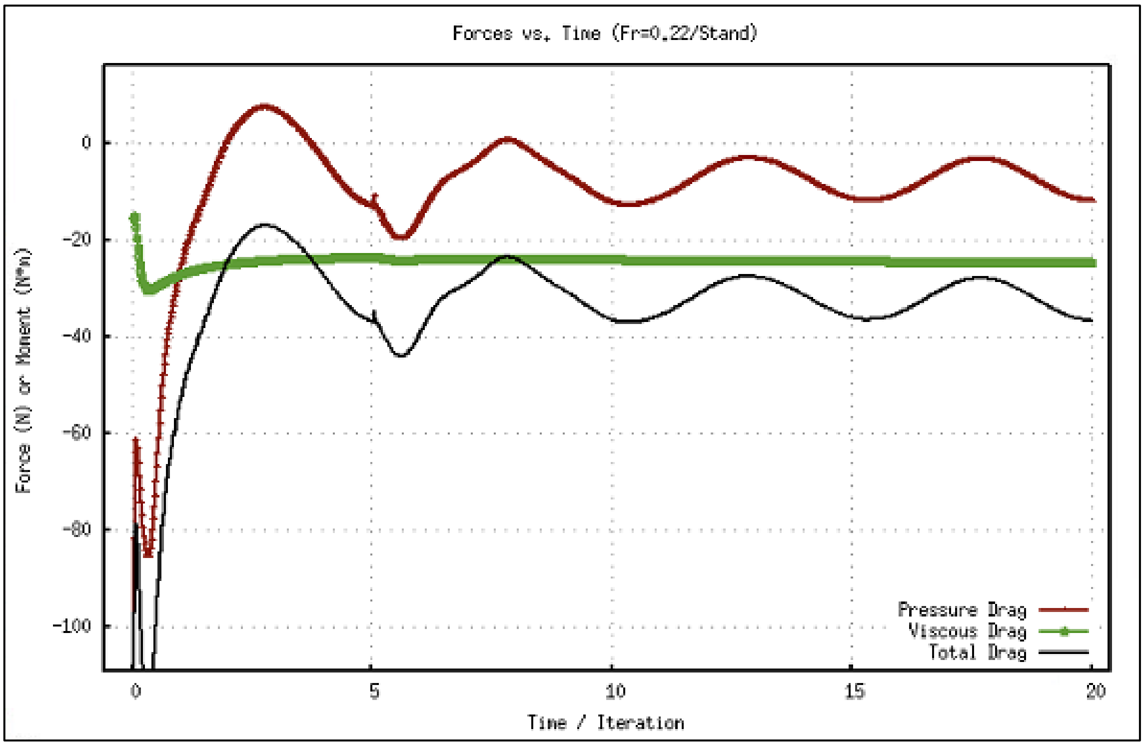


Figure 22 - Forces vs Time at Fr=0.22 for standard k-omega

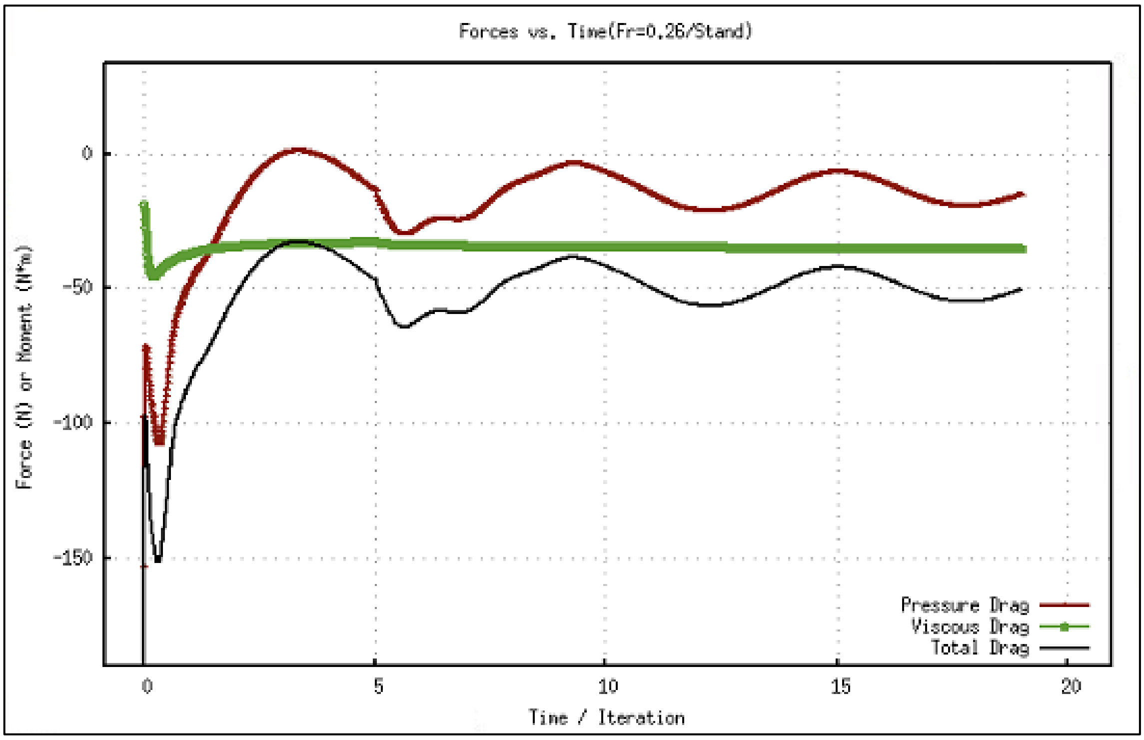


Figure 23 - Forces vs Time at Fr=0.26 for standard k-omega

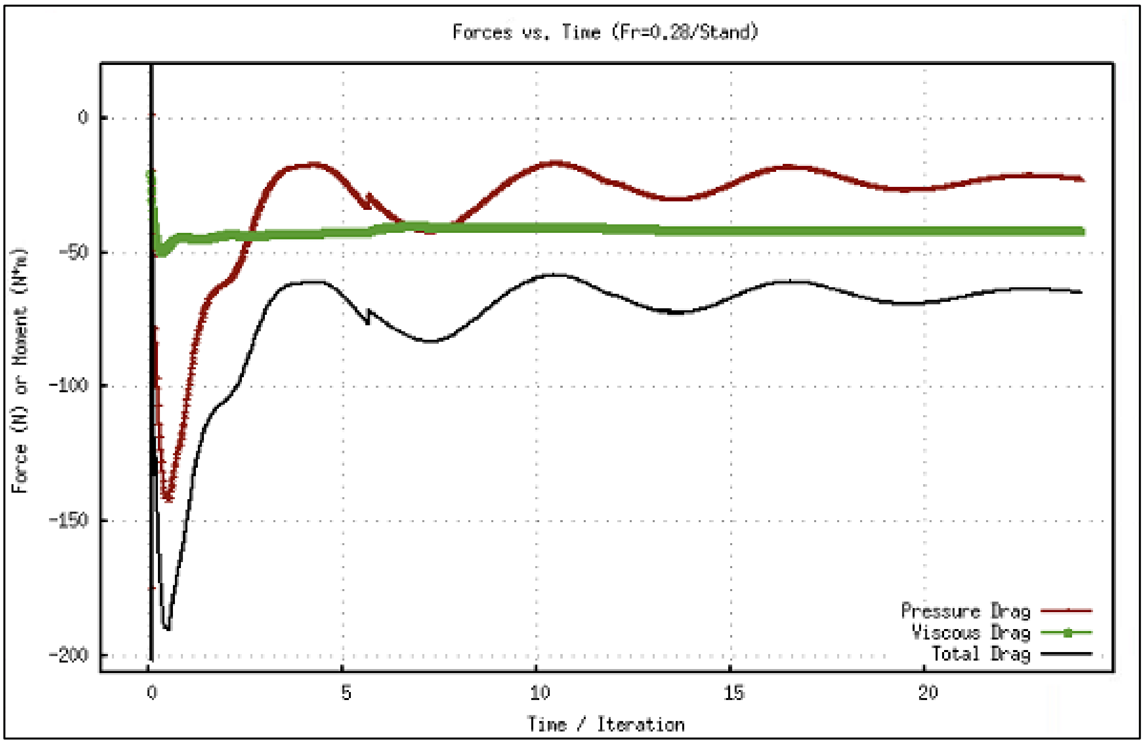


Figure 24 - Forces vs Time at Fr=0.28 for standard k-omega

In the above figures, three different curves can be seen which represent the pressure drag (Red), Viscous drag (Green) and the total drag (black) respectively. It can also be seen that from 0-5 second there is a disturbance in simulations and after a number of iteration become smooth. In each simulation two different solvers was used, InterFoam for first 5sec which capture the phenomenon in fixed position and from 5 onward InterDyMFoam which capture the dynamic behavior of ongoing simulations.

Total resistance values have been taken from the average values of each total drag curve using formula:

ABS(Crest+Trough)/2.

The table below gives the details about each Total resistance value for their respective speeds.

Table 7 - Total resistance values for Standard k-Omega

TOTAL RESISTANCE TAKEN VALUES - Stand				
Speed (m/s)	Froude (Fn)	Crest	Trough	R_T (KN)
1,83	0,22	-26	-32.01	29.01
2,196	0,26	-41.01	-50.14	45.58
2,38	0,28	-60.21	-70.21	65.21

Results obtained from CFD simulations (using Standard K-Omega) are presented below with their experimental data by using ITTC 1957 formulas.

Table 8 - Comparison of CFD and EFD results (standard K-omega)

Speed		EFD (KCS)		CFD (KCS)-OpenFOAM		Error in C_T
V(kn)	Fr	C_T *10^3	C_R*10^3	C_T *10^3	C_R*10^3	%
3.6	0.22	3.40	0.48	3.62	0.57	6.48
4.3	0.26	3.66	0.83	3.95	0.72	7.85
5	0.28	4.45	1.66	4.82	1.57	8.25

After the comparison with the experimental data, it has been seen that although the error is fine a bit of high and cannot be neglected so in order to improve the results we will perform the same simulations with different turbulence model which is SST-K omega.

6.5.2 Post - processing results for Standard K Omega

In the following figure, there are six different pictures representing the free surface elevation in two different views for three different Froud numbers using Standard K-Omega as turbulence model. For the picture A1: Fr = 0.22/ Top view; A2: Fr = 0.22/ Front view; B1: Fr = 0.26/ Top view; B2: Fr = 0.26/ Front view; C1: Fr = 0.28/ Top view and C2: Fr = 0.28/ Front view.

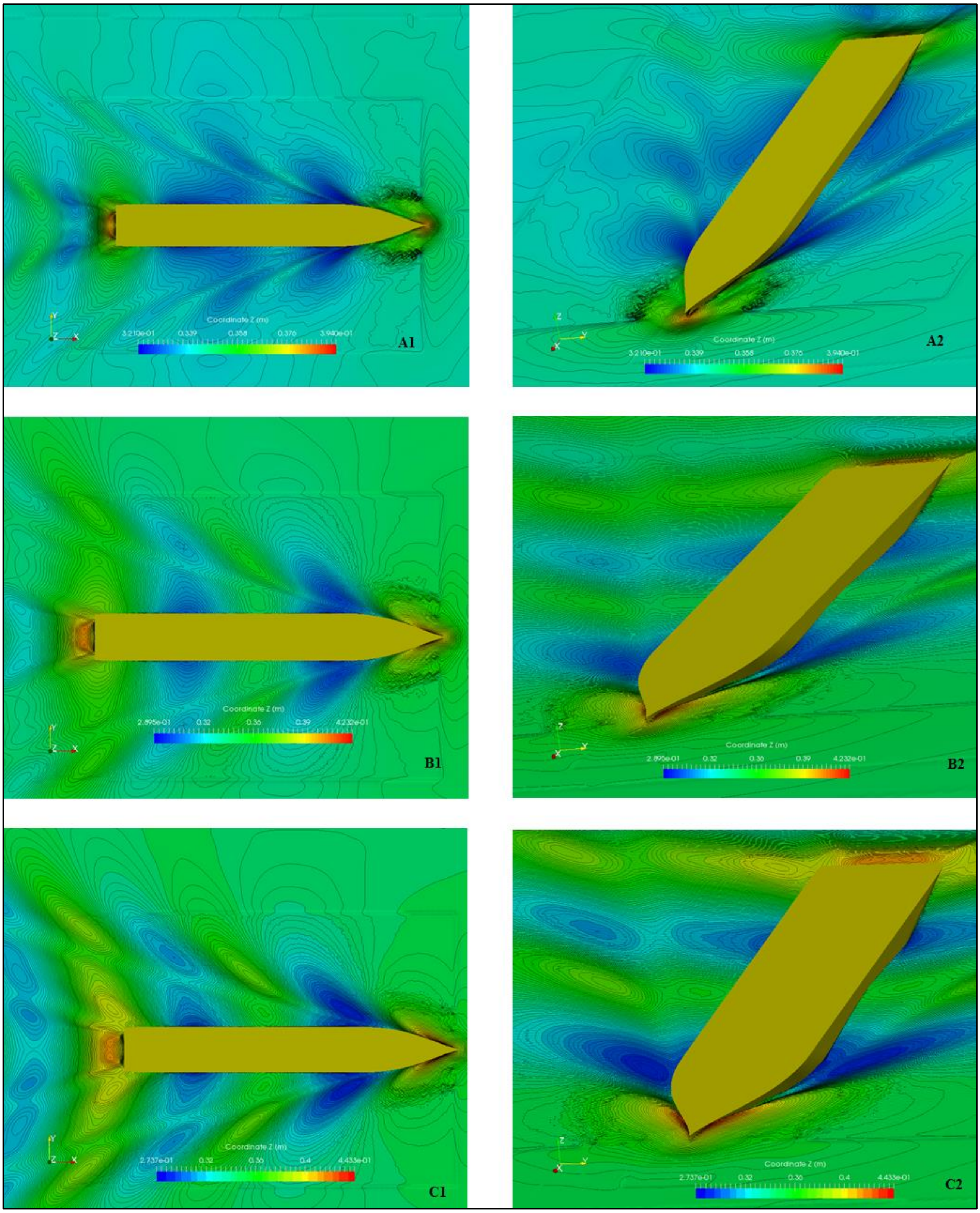


Figure 25 – Free surface elevation at three different speed for standard K-Omega

Free surface elevation for different speed here although doesn't actually represent the complete phenomena as it can be seen from the numerical data but it gives a clear idea at some rage. It can be seen that at low Froude (Fr=022), the max encounter wave elevation is around 3 meter which goes high for the later Froude numbers from the figures it can be observed that the mesh is not refined enough and hence there are some blown-up areas which make the elevation worse at some point. At some speeds, it can also be observed that the wave elevation is very high for this ship which is of course not very good but still acceptable. At low Froude, the wave elevation

Now the next thing we will do is to try the same procedure by using another turbulence solver named as SST K-Omega and then we will compare our obtained results to make sure which models we will adopt for this project.

6.5.3 SST-K-omega

The SST k-ω turbulence model is a Two-equation eddy-viscosity model which has become very popular. SST formulation combines the best of two worlds. SST k-ω model can be used as a Low-Re turbulence model without any extra damping functions.

The simulations were performed at three Froude speeds;0.22,0.26,0.28. In order know the spectrum of ongoing simulations a simple Python code is written which present the every detail of the whole process and hence an overall process of the simulation was being monitored graphically. Figures below shows the overall simulations results with respect to their time.

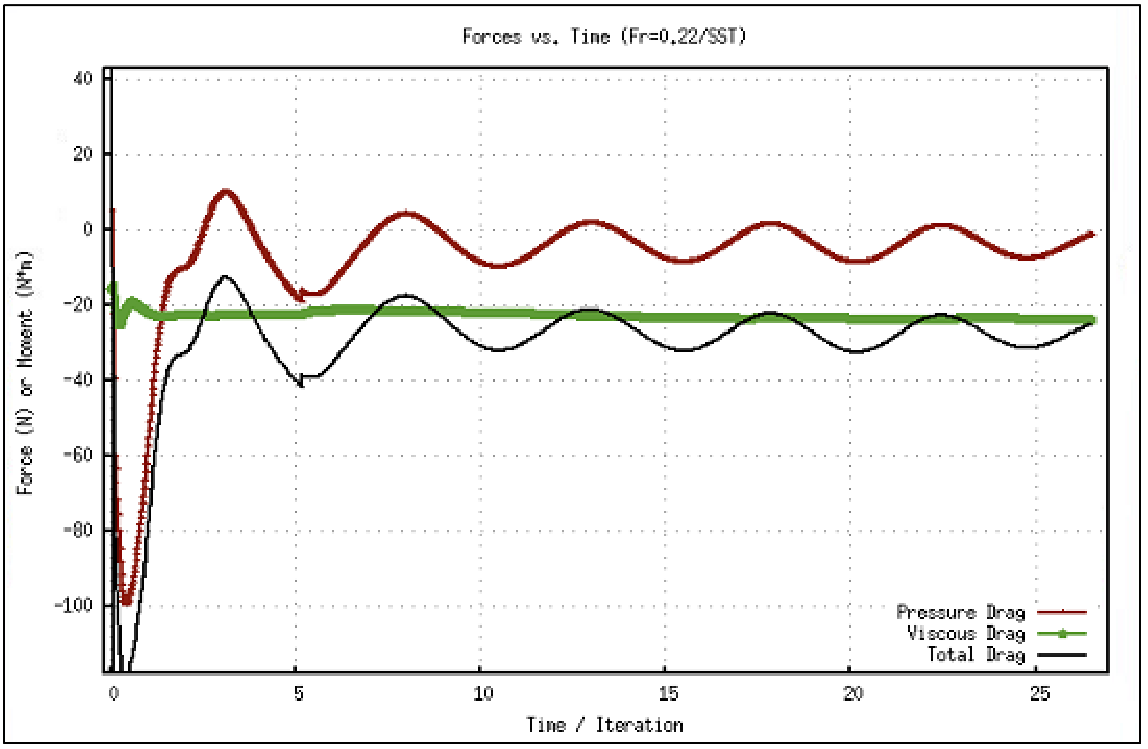


Figure 26 - Forces vs Time at Fr=0.22 for SST k-omega

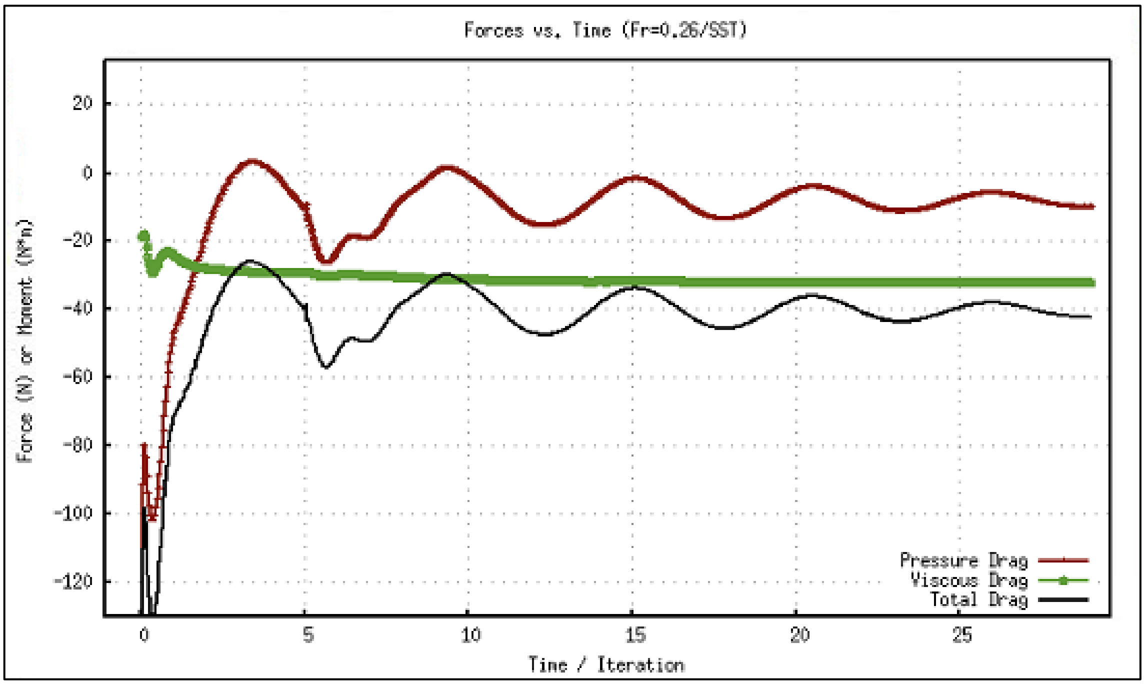


Figure 27 - Forces vs Time at Fr=0.26 for SST k-omega

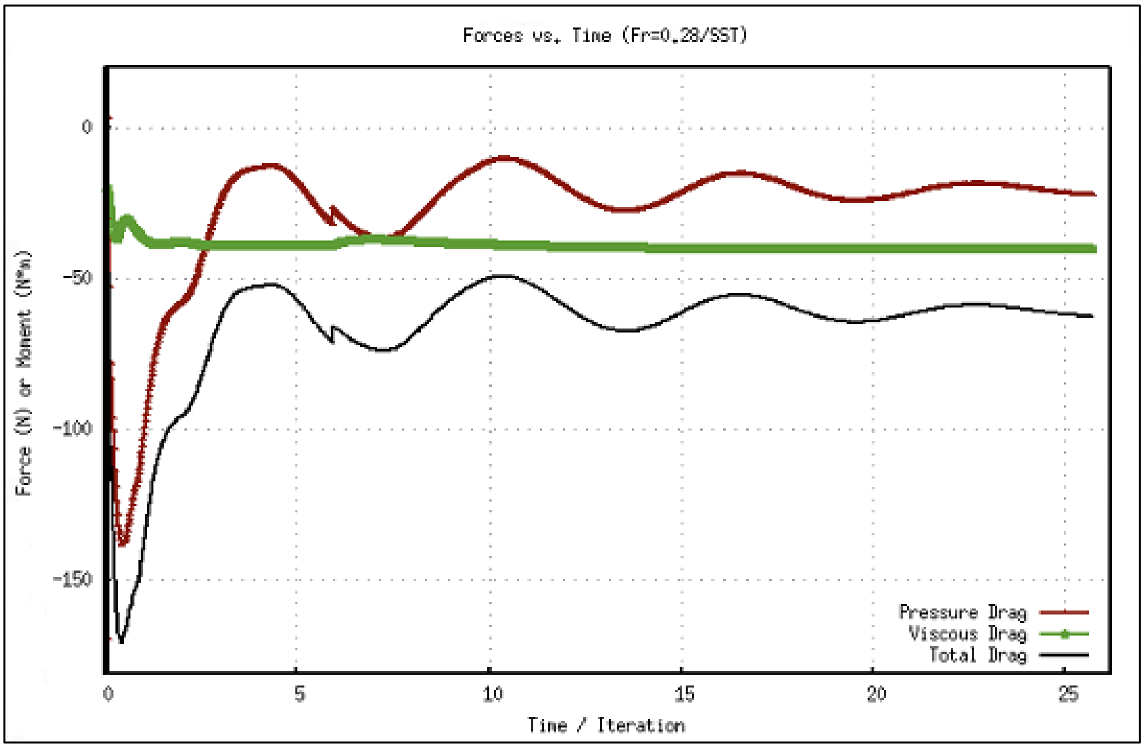


Figure 28 - Forces vs Time at Fr=0.28 for SST k-omega

In the figure, three different curves can be seen which represent the pressure drag (Red), Viscous drag (Green) and the total drag (black) respectively. It can also be seen that from 0-5 second there is a disturbance in simulations and after few hundreds of iteration become smooth. In each simulation two different solvers was used, InterFoam for first 5sec which capture the phenomenon in fixed position and from 5 onward InterDyMFoam which capture the dynamic behavior of ongoing simulations.

Convergence is also being monitored for each iteration for the overall process. In the figure below convergence ratio for Froude 0.22 for SST K-omega is shown which shows good and smooth results as expecting.

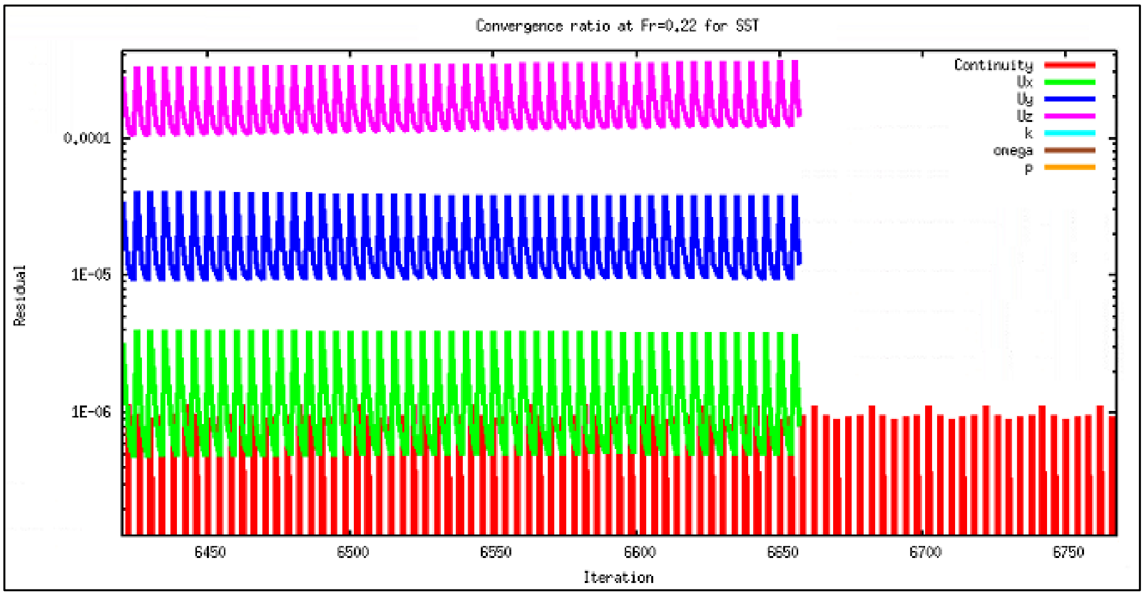


Figure 29 - Residuals for Fr=0.22

Total resistance values have been taken from the average values of each total drag curve using formula:

ABS(Crest+Trough)/2.

The table below gives the details about each Total resistance value for their respective speeds.

Table 9 - Total resistance values for SST k-Omega

TOTAL RESISTANCE TAKEN VALUES - SST				
Speed (m/s)	Froude (Fr)	Crest	Trough	RT (KN)
1,83	0,22	-20.21	-33.41	26.81
2,196	0,26	-44.8	-35.9	40.35
2,38	0,28	-55.5	-58.5	57

After the manipulations of data below is the comparison of our CFD values with the experimental data (Gothenburg, 2010):

Table 10 - Comparison of CFD and EFD results (SST K-omega)

Speed		EFD (KCS)		CFD (KCS)-OpenFOAM		Error in C_T
V(kn)	Fr	C_T *10^3	C_R*10^3	C_T *10^3	C_R*10^3	%
3.6	0.22	3.40	0.48	3.35	0.53	1.46
4.3	0.26	3.66	0.83	3.51	0.79	4.17
5	0.28	4.45	1.66	4.22	1.62	5.23

Which is also reasonable value and in comparison, of K-Standard omega (error percentage was quite high), it has seen that the most prominent two-equation models in this area are the second one. it gives a highly accurate prediction of the onset and the amount of flow separation under adverse pressure gradients by the inclusion of transport effects into the formulation of the eddy-viscosity. This results in a major improvement in terms of flow separation predictions.

6.5.4 Post – processing results for SST K Omega

The wave elevation of KCS at Froude Numbers 0.22, 0.26 and 0.28 are shown in the next figure. These results are computed by steadyNavalFoam and the post-processing was carried out using paraFoam.

For the picture A1: Fr = 0.22/ Top view; A2: Fr = 0.22/ Front view; B1: Fr = 0.26/ Top view; B2: Fr = 0.26/ Front view; C1: Fr = 0.28/ Top view and C2: Fr = 0.28/ Front view.

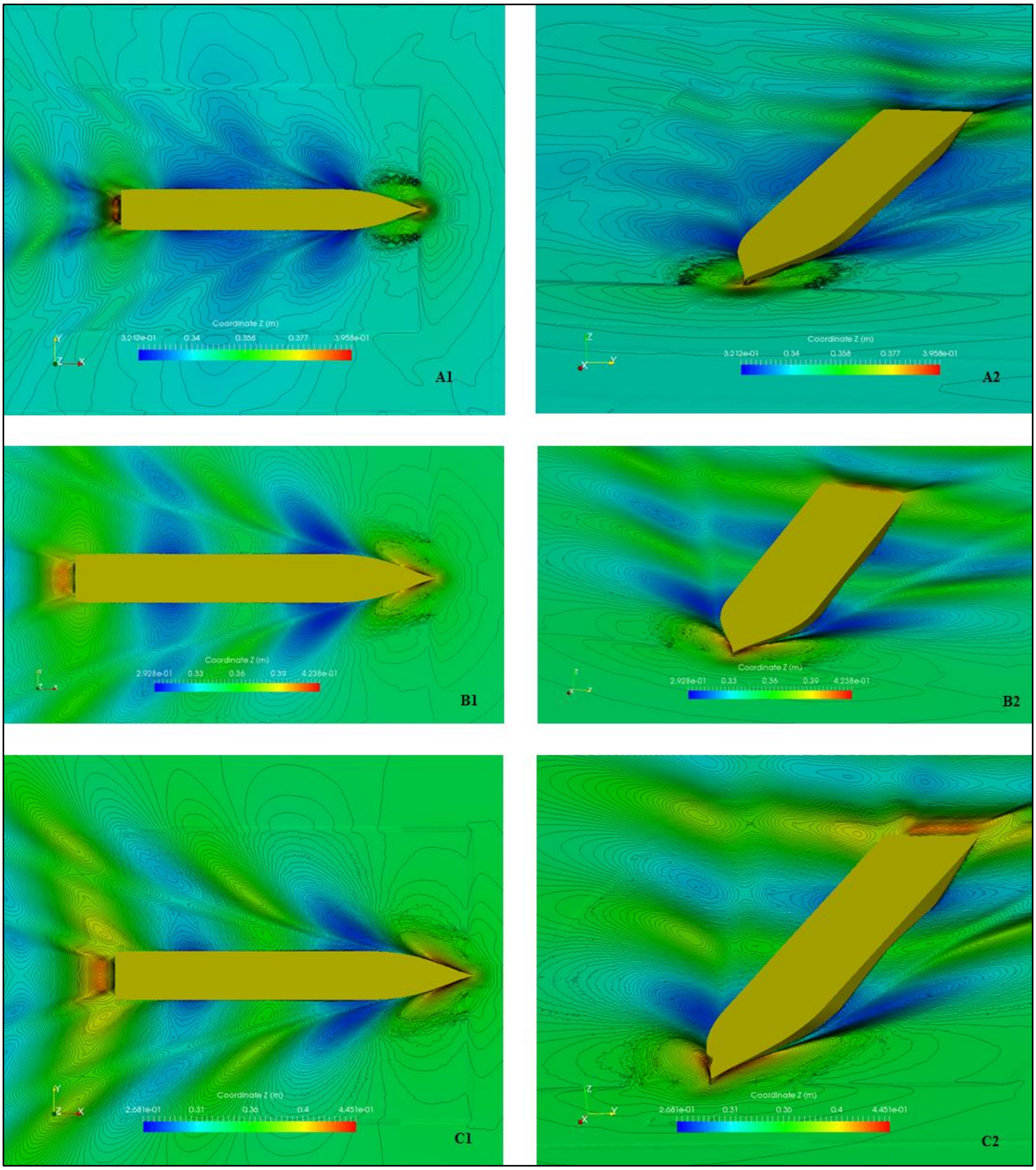
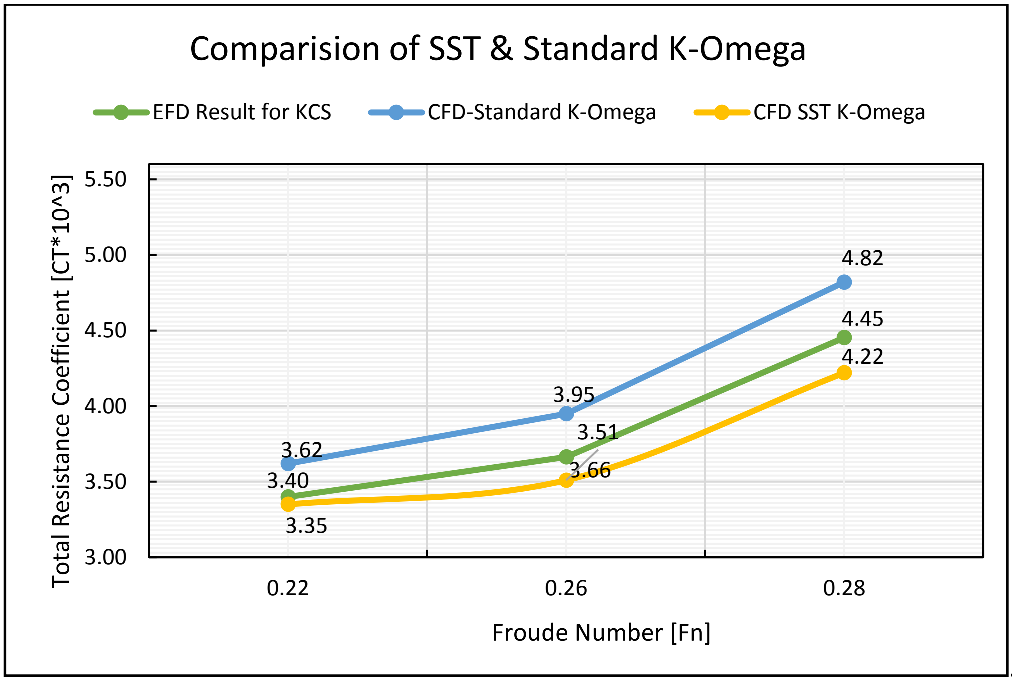


Figure 30 – Free surface elevation at different Fn numbers using SST K-Omega

From the above figures, it can be observed that the computations, using SST K-Omega, are extended through the sublayer (Viscous) close to the wall. Here a clear difference can be seen in compared with Standard k omega which is not a good in capturing of the adverse pressure gradient. A very fine mesh in the near-wall zone and the correspondingly large number of nodes are required for a low Re approach. Computer-storage and run-time requirements are higher (Very Important) and care must be taken to make sure a good numerical resolution near the wall region to capture the rapid variation in variables.



Fogure 31- Comparision of Turbulance Models

In the above figure, CFD and EFD values are compared for different turbulence models with respect to their total coefficient of resistance and it is seen that SST gives the better results in each and every aspect concerning the current problem.

It is observed that the SST k-Omega model is an enhancement of the original k-Omega model. Plus, It has the advantage that it can be applied to the viscous-affected region without further modification, which actually makes it a popular choice in aerospace applications.

Due to the good behavior in adverse pressure gradients and separating flow, Pressure contour of KCS hull is presented and it is seen that SST K-omega would be the best selection for the current work. Investigation for different turbulence models requirement was selected actually on the idea which is based on that highest positive pressure on the bottom surface (Hull) appears at the stern that results in a high squat at the stern as well. Which simply leads to the concept that at the bow high pressure is not completely underneath.

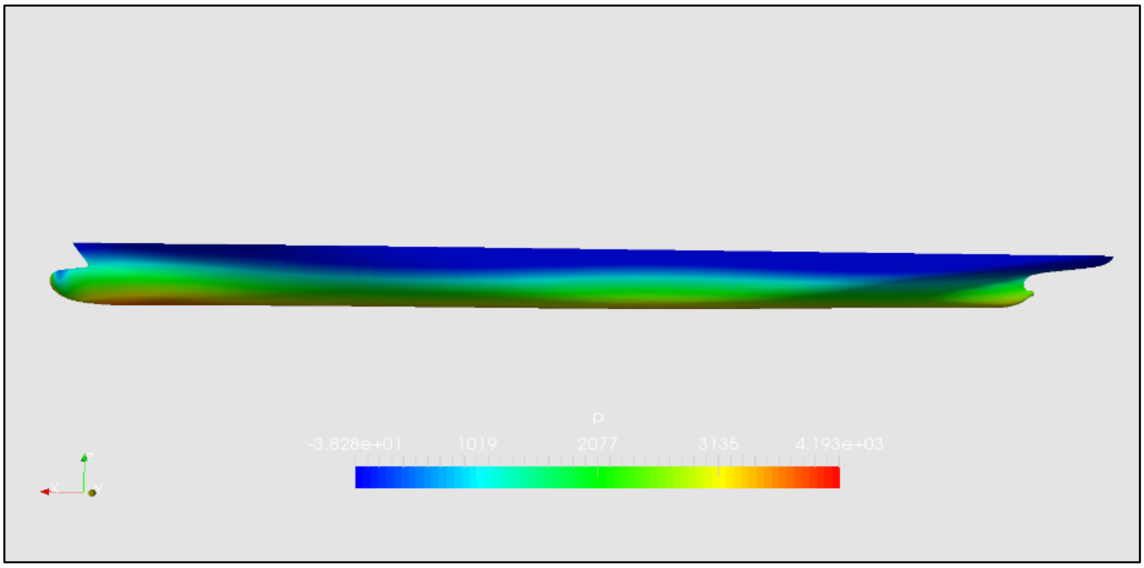


Figure 31 - Pressure contour of KCS Hull

The superior performance of this model has been demonstrated in a large number of validation studies and for our further work, we will be using SST K- Omega as our turbulence model because it has overcome the deficiency precision given by K-Omega.

7. COMPUTATIONAL STUDY FOR DELTA SHIP (CONTAINER)

All the previous procedure was adopted in order to find the suitable turbulence model for the Delta Ship which is a container ship, most of the data has been kept confidential due to trademark. In 2012, Towing tank tests have been performed in 'Ship Design and Research Centre - CTO, Poland' CTO, and their data has been listed along with the ship main characteristic, in order to compare the results.

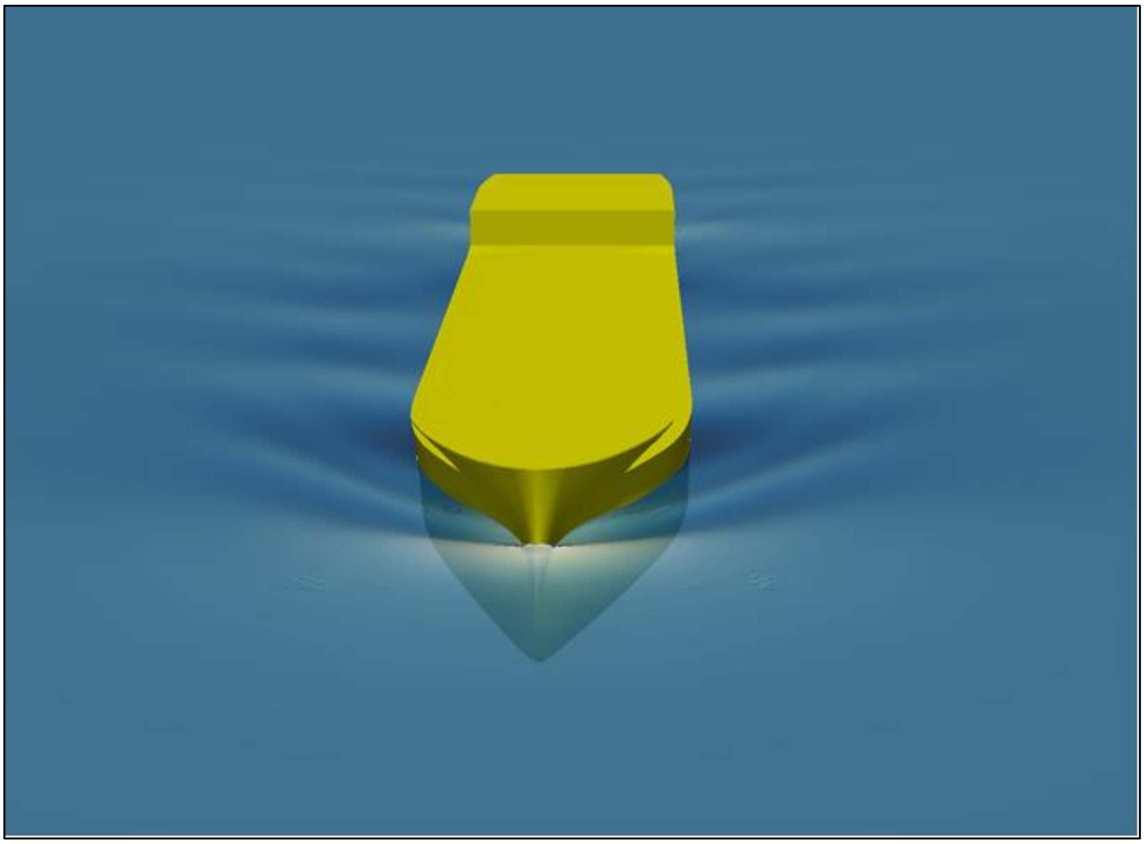


Figure 32 - Hull geometry of Delta ship

The main particulars for model and ship are given in Table 9. In the proceeding work, 1/16 model scale was used for CFD calculations.

Table 11 - Main particulars of Delta Ship

Item	Symbol	Unit	Full Scale	Model Scale
Length Between Perpendicular	Lbp	[m]	107,7	6,731
Length On Water Line	Lwl	[m]	109,603	6,850
Submerged Length	Los	[m]	109,947	6,872
Breadth on Water Line	Bwl	[m]	17,5	1,094
Draught	T	[m]	6,5	0,406
Draught at Aft Perp	Tap	[m]	6,5	0,406
Draught at Fore Perp	Taf	[m]	6,5	0,406
Displacement Volume	∇	[m3]	9284,262	580,266
Longituninal Center Of Bouyancy (% of Lwl)	LCB	[%]	-1,631	-0,102
Block Coefficient (Refer to Lwl)	Cb	[-]	0,745	0,047
Midship Coefficient	Cm	[-]	0,989	0,062
Prismatic Coefficient	Cp	[-]	0,753	0,047
Waterplane Coefficient	Cwl	[-]	0,832	0,052
Wetted Surface Area 'Bare Hull'	S	[m2]	2769,485	173,093
Bulbous Sectional Aarea	Ab	[m2]	1,186	0,074
Centre of Bulbous Sectional Area	Hb	[m]	4,254	0,266
Transom Area	At	[m2]	0,119	0,007
Transverse projected area above the waterline	Apt	[m2]	350	21,875
Roughness		[mm]	0,12	0,008

The 3D model of the bare hull can be seen in the following figure. (Refined by using Rhinoceros)

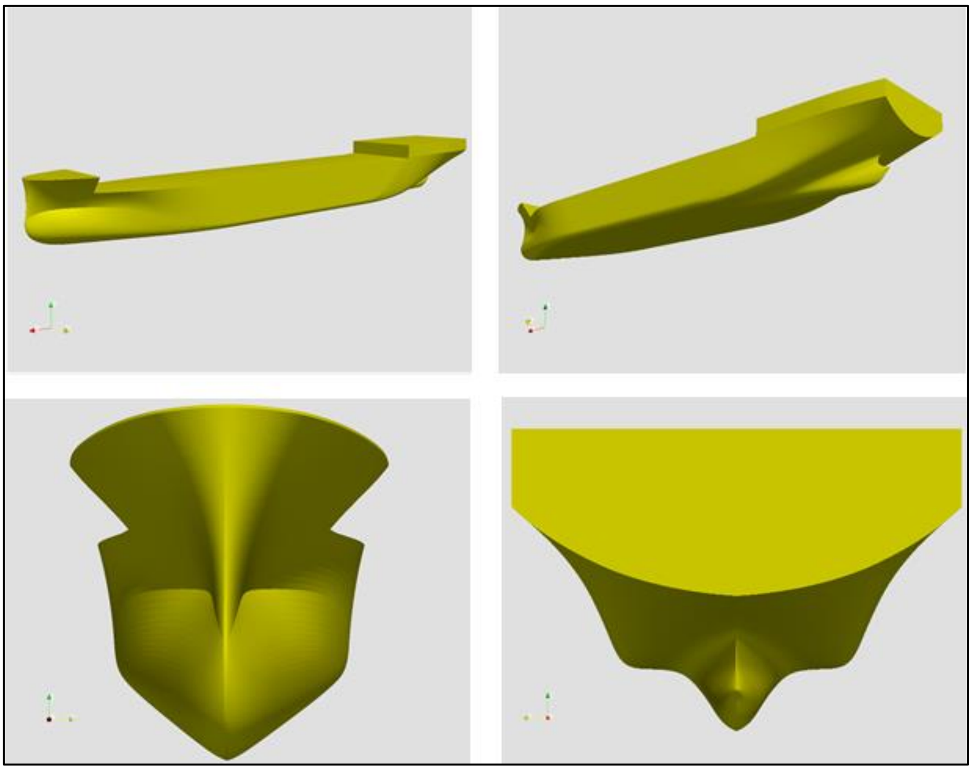


Figure 33 – 3D Geometry of the Delta Ship

7.1.Numerical mesh (Delta)

In order to create the numerical mesh, two important files used in OpenFOAM; blockMeshDict and the snappyHexMeshDict (Annex-II is added for the complete understanding of this two files). The first one is situated in the constant folder while the other one is in the system folder of the case folder (Detail of these two meshes have been explained in the Chap 4).

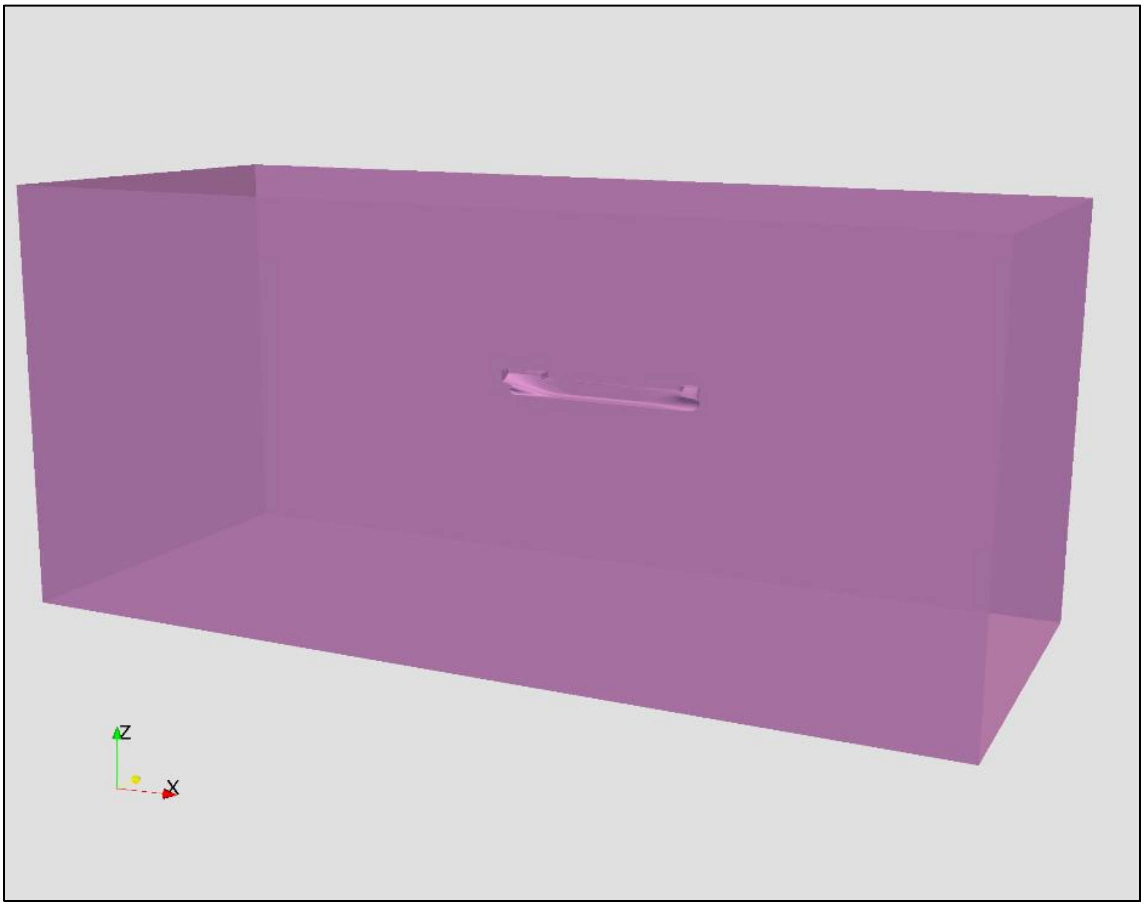


Figure 34 - Numerical Mesh

7.2.Mesh Generation

Grid generation is always considered as the most important and most time-consuming part of CFD computations. The quality of the performed grid plays a direct proportional role on the quality of the analysis, regardless of the flow solver used. Furthermore, the solver will be more robust and efficient when using a well-constructed mesh.

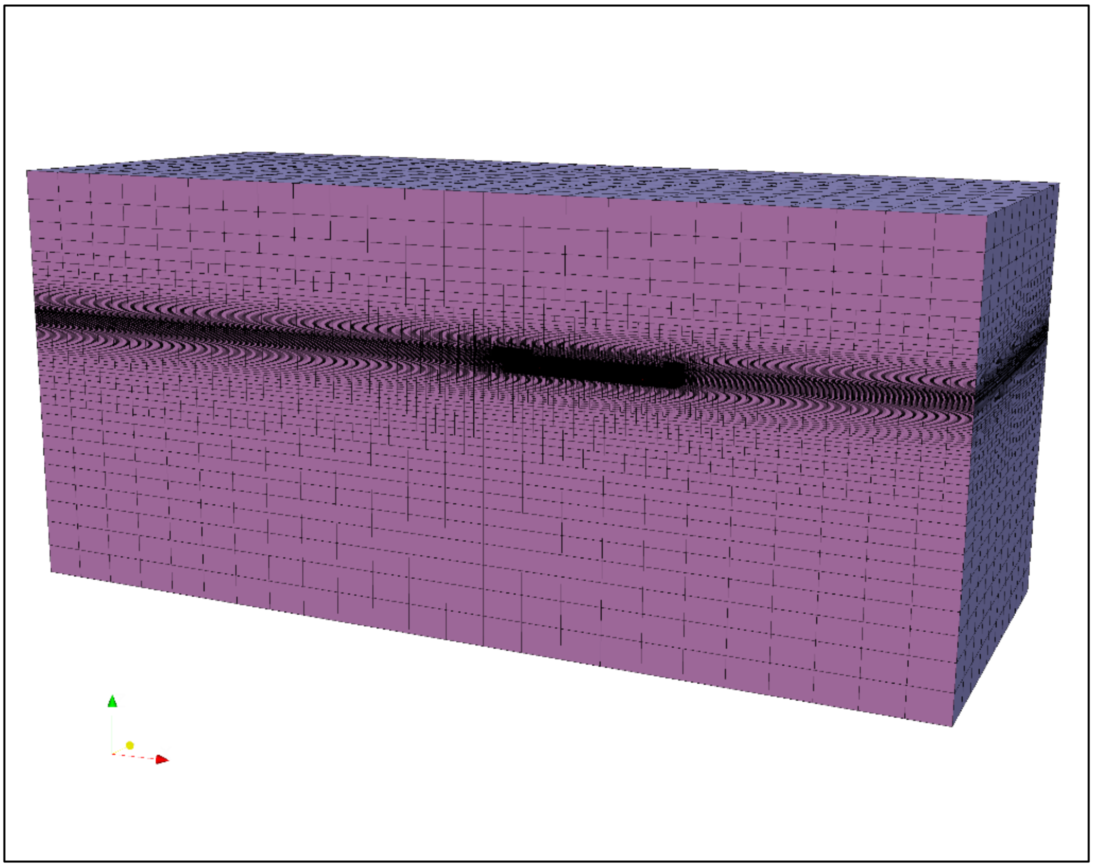


Figure 35 - General View of Mesh

The generated mesh contains the equal size of cells everywhere that results in the discretized grid. As we have explained before every possible explanation about the mesh generation in Chap 4, here we will go with some extra information. Talking about the points which are used to distinguish the 8 vertices for each block, as well as the number of cells in each the x, y and z directions. At last the boundary patches are defined using the vertices. Conditions of the simulation are usually defined by these boundaries. It consists of an inlet, outlet, sides of the channel and the atmosphere. The number of points, cells, and faces created by each blockMesh for the Delta ship simulations are listed in the table below.

Table 12 - Mesh status (Delta Ship)

Mesh Status	
Points	4161600
Faces	11891937
Internal Faces	11729962
Cells	3870107
Faces per cell	6,10368
Boundary Patches	7
Point Zones	0
Face Zones	0
Cell Zones	0

This whole data is defined precisely in a polyMesh folder and different types of cell used in simulations are listed in the table below:

Table 13 - Overall number of cells of each type (Delta Ship)

Types of Cells	
Hexahedra	3653549
Prisms	5498
Wedges	0
Pyramids	0
Tet Wedges	135
Tetrahedra	0
Polyhedra	210925

As explained before once again snappyHexMeshDict is used to mesh an obstacle internally into the blockMesh which is created previously. Detail explanation of SnappyHexMeshDict has been explained in the chap 4. In order to recognize the surfaces clearly, the ship must be in STL format. Using this utility, one is able to create a new patch which is hull surface.

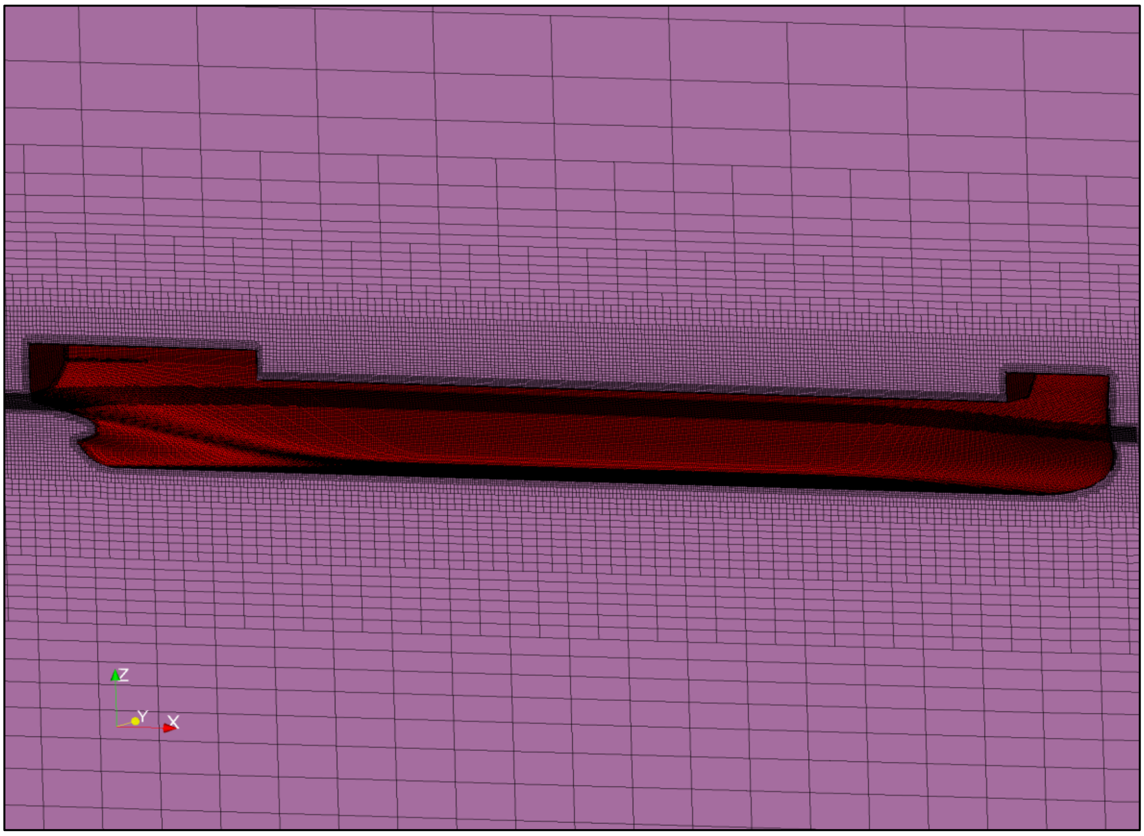


Figure 36 - General View of Mesh

In the above table, different types of cells are explained and it can be seen that they are completely hexes so they must need split into different parts so they can interact with the ship hull. After this splitting of the cells, they are then refined at the surface. This procedure is done by specifying a min. level to applied across the whole surface and a max. the level that can be applied to those cells that are not very simple (OpenFOAM User Guide, 2011). Figures below shows the mesh underneath the hull; bow and stern which is fair, cells having good size and shape.

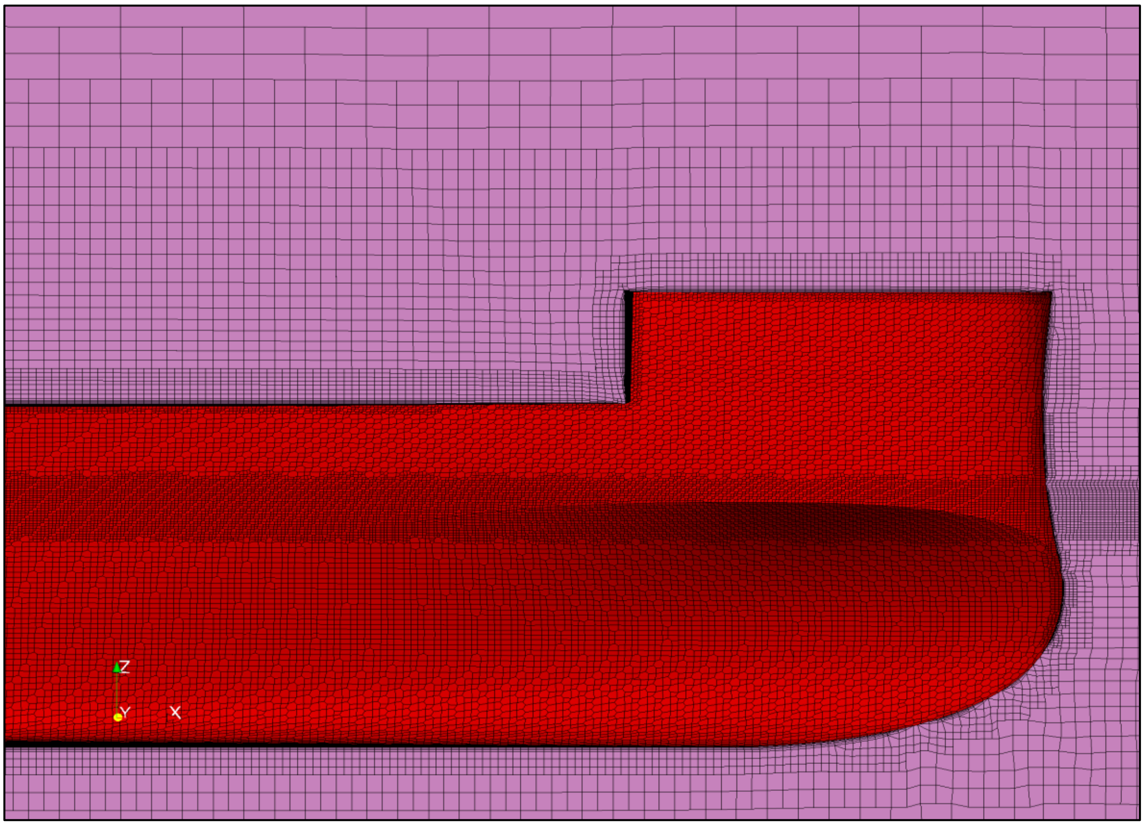


Figure 37 - General View of Mesh (Bow)

A very good refinement is performed for the bow and stern of ship hull which will be capturing the effect, to do so snappyHexMesh was used and a detail of this term is already explained in chap 4 already.

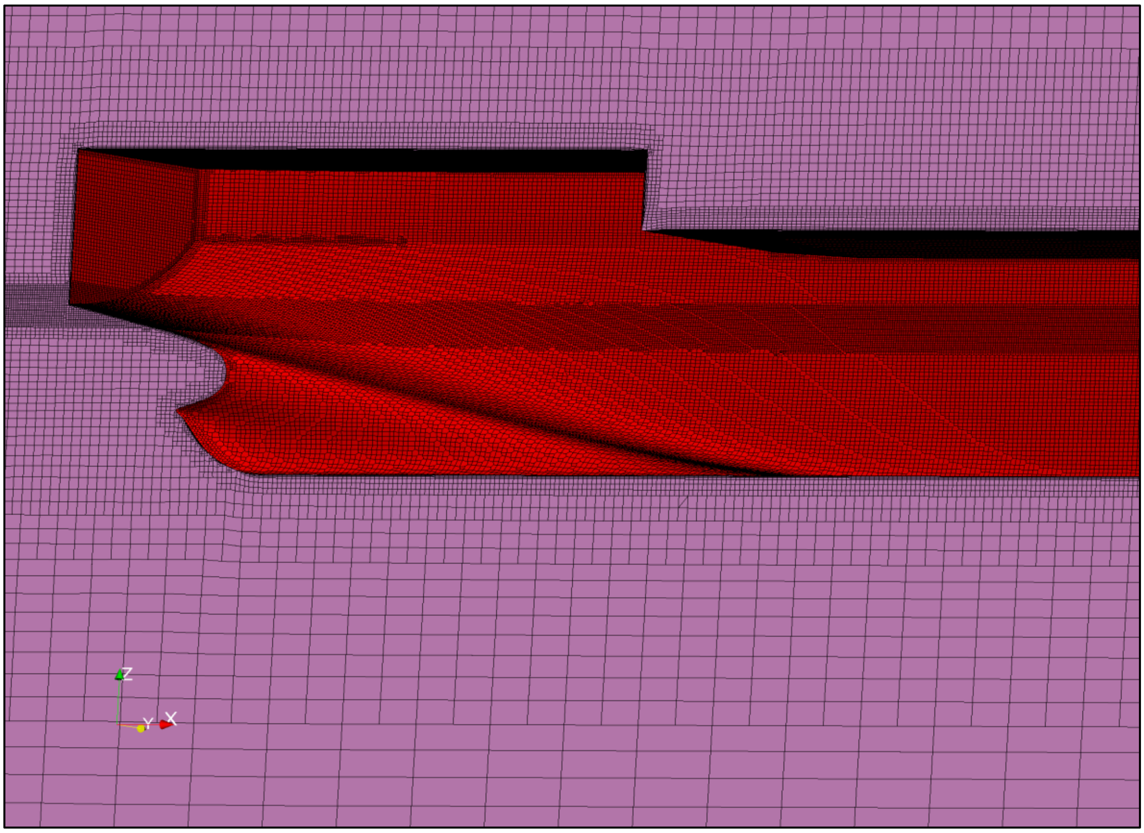


Figure 38 - General View of Mesh (Stern)

All the cells inside of the hull wall are first removed, and more refinement is done in other areas. In the presented case, the free surface is the main region which is refined for all the cases, in another word it’s the interface actually between the air and water. This the most important because of this, these two fluids interact due to the difference in their properties.

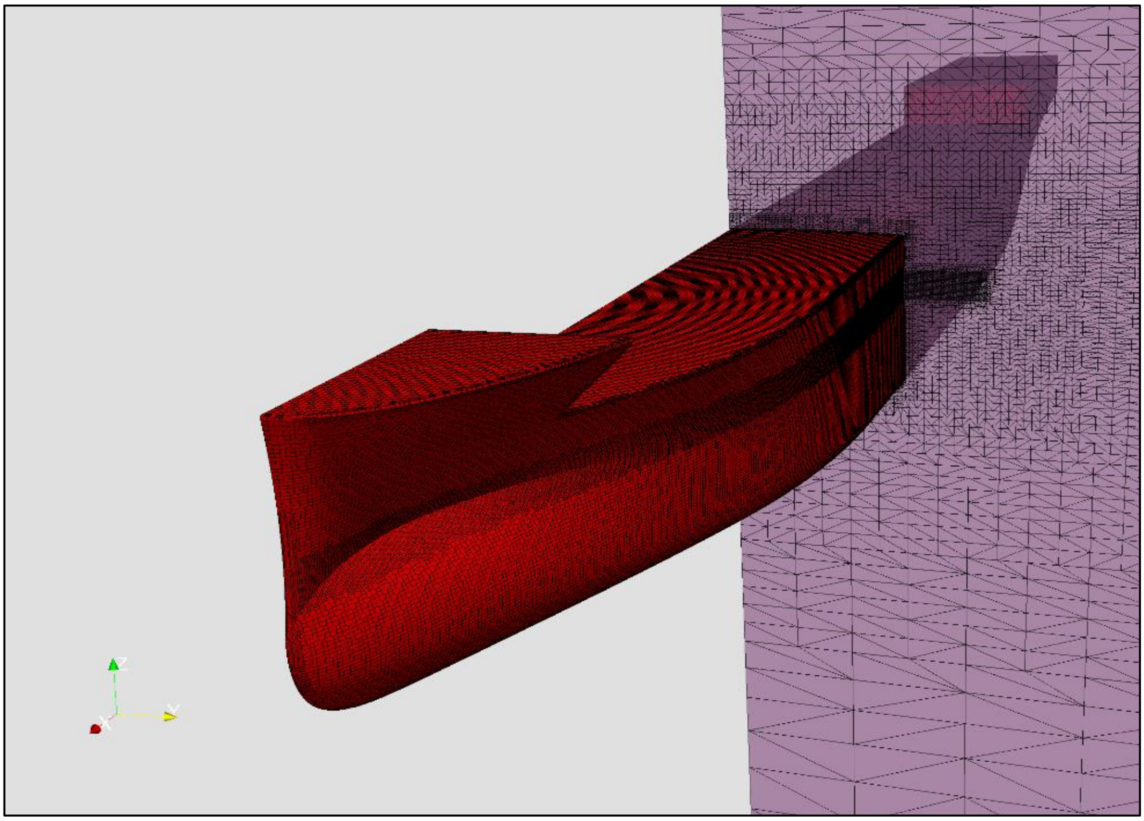


Figure 39 - Transverse view of mesh

From the figure below it can be seen that here may still be a small number of cells which are left inside of the hull and snapping is the only one way so these cells are removed. Here the surface is smoothed enough so there are no visible sharp edges.

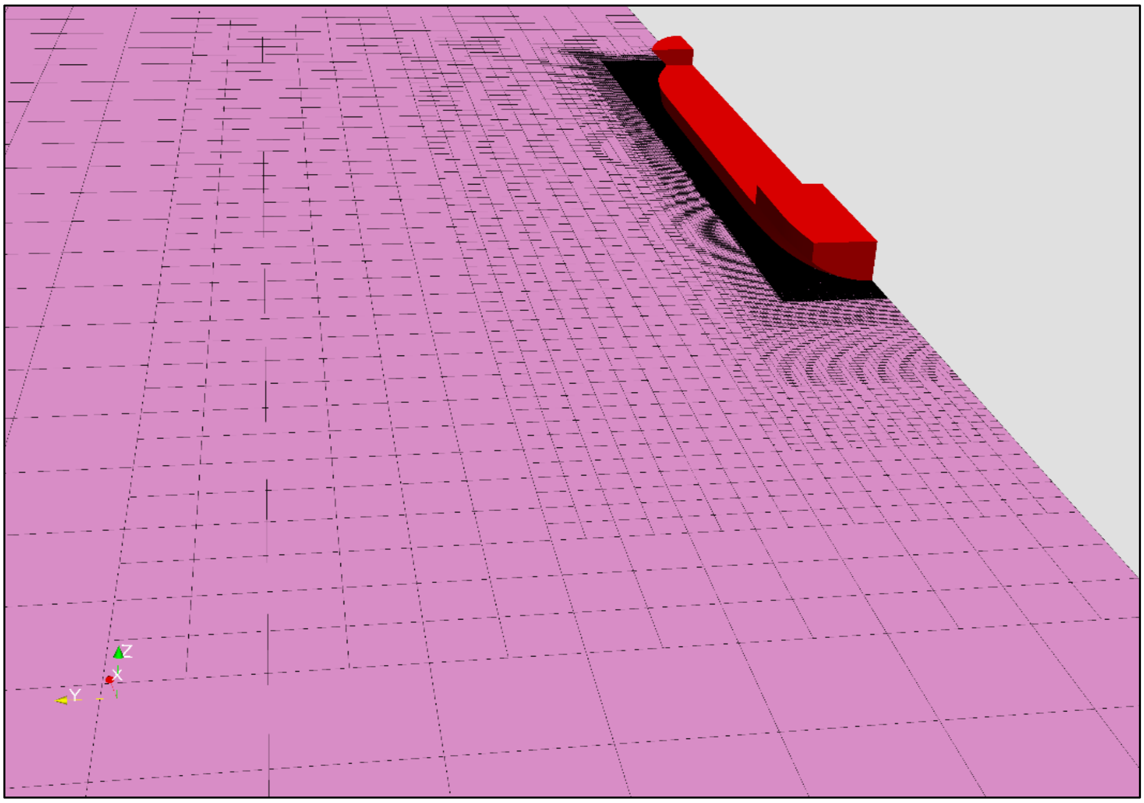


Figure 40 - Free surface view of mesh

The overall process of mesh quality control which means the process if meshing and snapping follow the following procedure:

I. After completing the mesh, If the cells are still outside then they are continued to be refined;
II. Some controls (for all cases) included a max non-orthogonal angle of 63°, max skewness at 24 (Boundary) and then of 6 (Intern);
III. The quality of the overall mesh is always based on the aspect ratio, and this should be as close to 1 as possible.

7.3.Fluid Properties

The interaction between water and air and prediction of the free surface, sometimes become unstable. In order to get a possible accuracy, an appropriate technique is chosen in order to make sure it models (meshes) this section (unstable) more precisely and correctly. In this project, we are only dealing with the multiphase flow, VOF Method, a good way of modeling ships that can produce breaking waves, because it can also be used for the two immiscible fluids and can calculate the interface interaction (overall simulations).

Double body analysis was run in order to get the value of foam factor (1+k) in the same way as we have done for the KCS ship and The volume fraction is used to calculate the value of alpha.

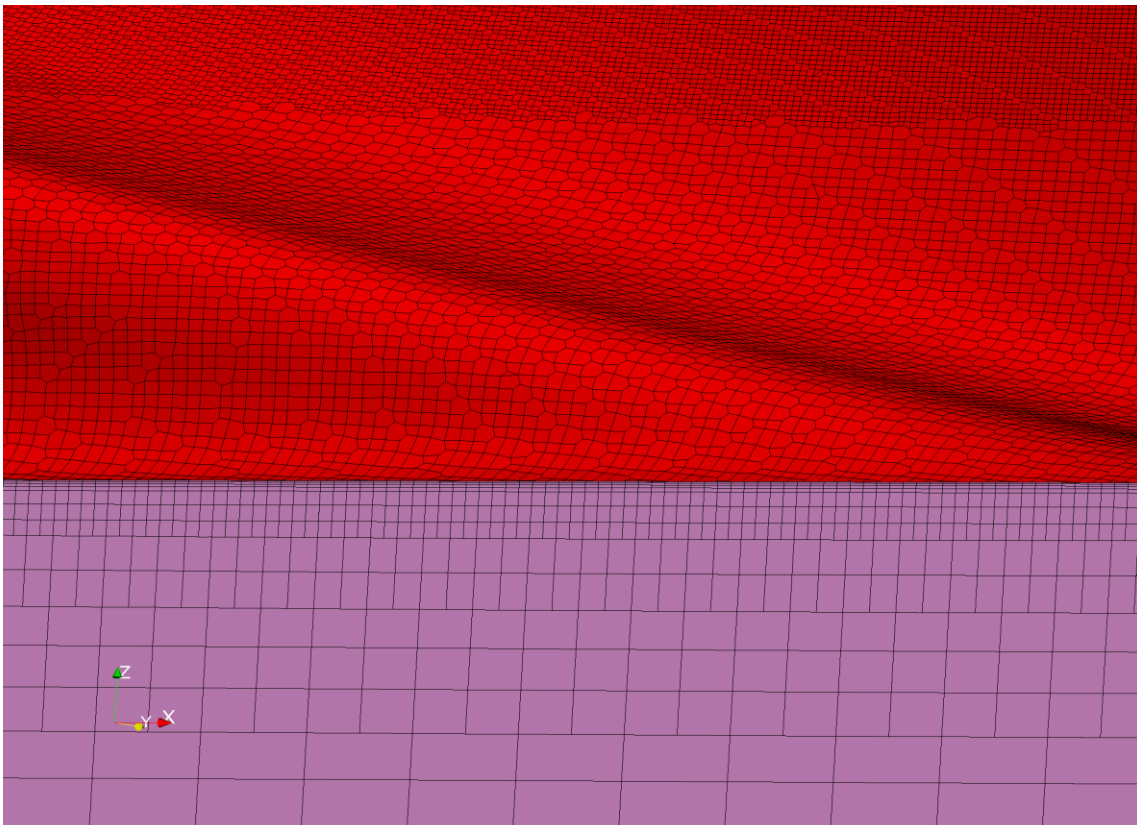


Figure 41 - Boundary layer mesh

Here the utility setFields was used to calculate the values of alpha by simply specifying a box that has the value of 1 overall and everything else was 0. The table below shows the fluid properties as specified in the dictionary transportProperties.

Table 14 - Fluid Properties (Delta Ship)

Water	Density	ρ	kg/m^3	998,4
	Viscosity	μ	kg/m-s	0,00104
Air	Density	ρ	kg/m^3	1,2
	Viscosity	μ	kg/m-s	1,85E-05

7.4. Comparison of Residual Resistance

Residual resistance comprises wave resistance, energy loss caused by waves created by the vessel and viscous pressure resistance. This residual resistance normally represents 10-30% of the total resistance for low-speed ships and up to 40-70% for high-speed ships.

Comparison of residual resistance coefficient has been present below in the table where it can be seen that the error is very much of perfect as it's a container ship.

Table 15 – Comparison of residual resistance

V_S (kn)	V_S(m/s)	F_n	C_R (CFD)	C_F(CFD)	C_T (CFD)	C_F (EFD)	C_T(EFD)	C_R (EFD)	Error (%)
9	4.63	0.141	1.37E-04	3.134E-03	3.27E-03	3.13E-03	3.27E-03	1.34E-04	2.06
10	5.144	0.157	1.90E-04	1.19E-03	1.38E-03	1.19E-03	1.37E-03	1.86E-04	2.14
11	5.658	0.173	2.79E-04	3.03E-03	3.30E-03	3.03E-03	3.31E-03	2.86E-04	2.45
12	6.173	0.188	3.82E-04	2.98E-03	3.36E-03	2.98E-03	3.36E-03	3.82E-04	0.03
13	6.687	0.204	3.96E-04	2.94E-03	3.34E-03	2.94E-03	3.33E-03	3.94E-04	0.61
14	7.202	0.22	6.44E-04	2.90E-03	3.55E-03	2.90E-03	3.54E-03	6.42E-04	0.36
15	7.716	0.235	1.17E-03	2.87E-03	4.04E-03	2.87E-03	4.03E-03	1.16E-03	0.22

From the obtained results, it is seen that error is relatively low and acceptable with much of accuracy and it's all due to the very fine mesh and computations resources used for this work (compared with KCS where the refinement of mesh around the hull wasn't good enough).

In general, residual resistance comprises wave resistance which actually refers to the energy loss caused by waves created by the vessel and viscous pressure (resistance).

In the below graph, performed simulations results for Delta ship are shown in term of residual resistance coefficient compared with their experimental data. Relatively low error confirm that the study performed for this work is good enough.

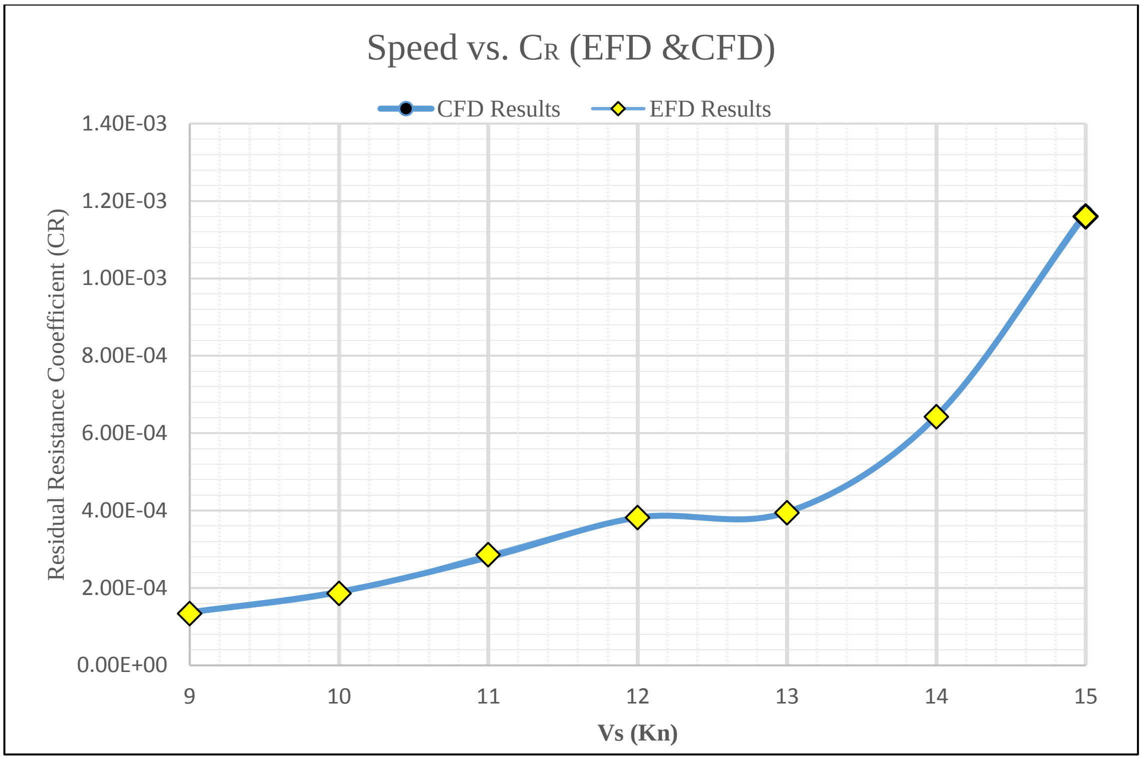


Figure 42 - Residual Resistance Coefficient Comparison of EFD and CFD

7.5.Post- processing results for Delta Ship

The wave elevation of a container ship at Froude Numbers 0.141, 0.188, and 0.235 are shown in the next figure. The results are computed by OpenFOAM and the post-processing is carried out using paraFoam (Paraview).

For the picture A1: Fr = 0.141/ Top view; A2: Fr = 0.141/ Front view; B1: Fr = 0.188/ Top view; B2: Fr = 0.188/ Front view; C1: Fr = 0.235/ Top view and C2: Fr = 0.235/ Front view

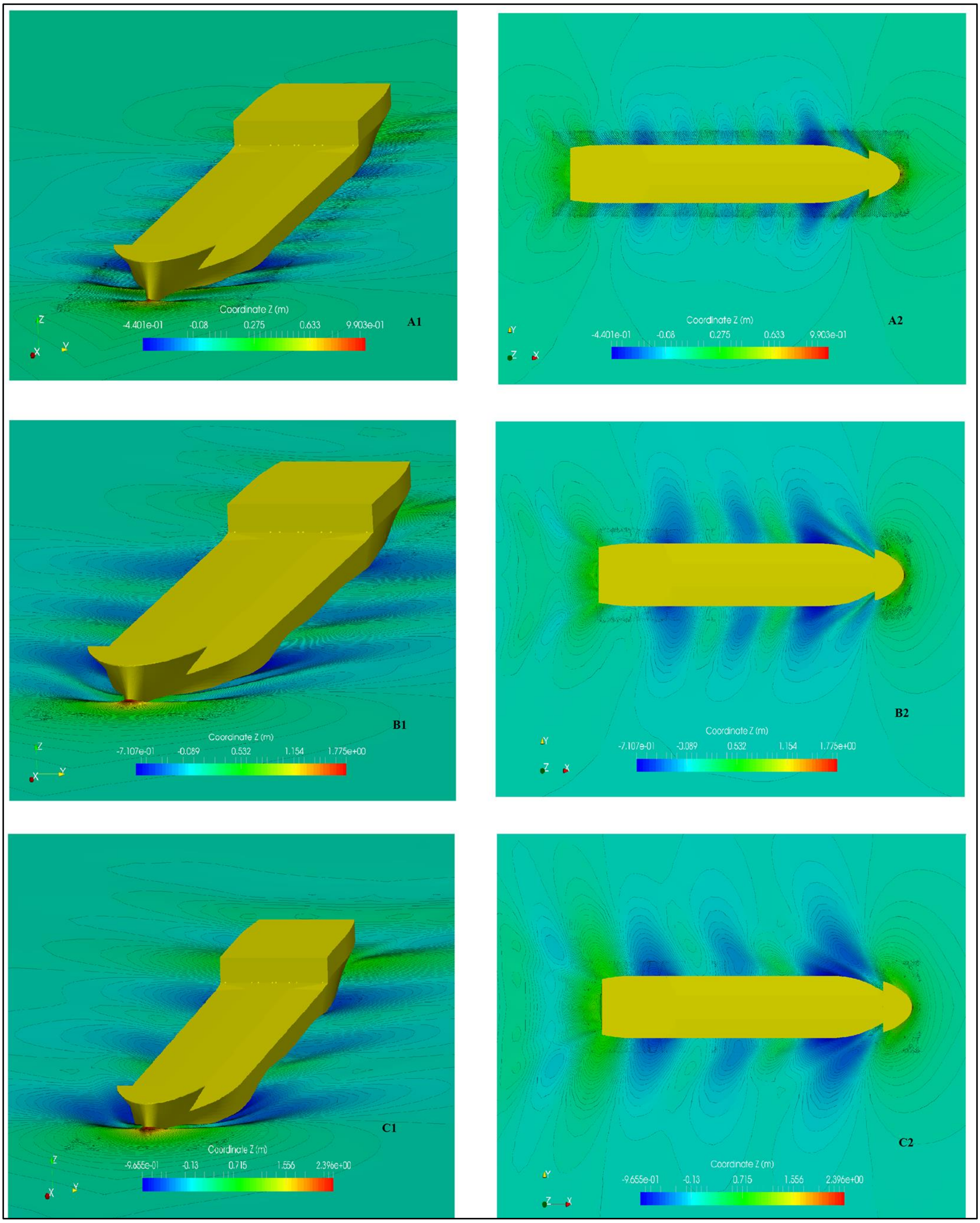


Figure 43 – Free surface elevation for T = 6.5m for different Fn numbers and speeds.

The Delta Ship is 6.731 m long and is tested at seven different Froude numbers ranging from Fr =0.041−0.235. The converged solution for all the simulations is presented above in the table along with their respective speeds. So for the sake of simplicity, we represent postprocessing results for the three different Froude numbers only (low, medium & high).The grid contains approximately 3870107cells.

In the above-presented figures, it is observed that at very low Froude Numbers as 0.141 and 0.157, the wave resistance is less significant. The wave pattern is barely visible forming only a slight disturbance in the water (bow). But wave pattern becomes more apparent as the ship speed is increased. Hence, the wave pattern of the rest Froude Numbers follows that of the Kelvin wave pattern, which is consisting of clearly visible diverging and transverse waves. These waves travel outwards from the bow and stern to form the V shape. Larger waves are created both by the bow and stern with the increasing of ship speed, while clear troughs are formed by fore and aft shoulders. Due to the steady nature of the forward speed tests, the simulations are performed in a steady, ship-fixed frame of reference. Some variations are observed in the size of the stern wave leaving the transom and magnitude of the stern wave pattern further downstream of the hull.

8. CONCLUSION

The main goal of this project was to investigate the hydrodynamics applications while using OpenFOAM, This CFD toolbox has been used to model a number of benchmark free surface flow cases with a very acceptable accuracy, Sometimes the simulations stops due to some online bugs and to resolve these errors we need a rich experience of this software. The results were good enough as they provided a very detailed information of where OpenFOAM was accurate and where it had problems. In any CFD analysis, the most important task is the computation time and its accuracy which was also one of the main concern of this work. In order to perform the simulations, supercomputer facility (LSF) has been adopted to save the time for the computations. After finalizing the complete study, below conclusion can be drawn:

1. Finite volume method is very much perspective for simulation of flow around ship in both hexahedral and tetrahedral grid resolution;

2. Two different turbulence models such as Standard k-omega and Shear Stress Transport (SST) k-ω models are used to capture boundary layer in the simulation of steady flow around hulls in the viscous turbulent flow. It is observed that the Standard k-omega model with standard wall functions computes the drag coefficient accurately. However, only SST k-ω model with the transitional flow is used for simulation of turbulent flow because of its better performance. Finally, due to the good results in adverse pressure gradients and separating flow, SST K- ω was selected for the container Ship, designed by Delta Marine;

3. For the KCS model, our results are generally in good agreement with the data. It has been observed that at Froude 0.22,0.26,0.28, error in C_T for SST are 3.2%, 3.78%, and 4.97%, and for Standard K-Omega were 4.75% 5.08% and 6.03% respectively The wave pattern is well predicted, with very little numerical damping and good agreement at larger distances from the ship While working with container ship designed by Delta Marine, it has been seen that it gives the most accurate results for the coefficients of resistance and the comparison with their experimental results with an error of its residual coefficient of resistance of 2.06%, 2.14%, 2.45%, 0.03%, 0.61%, 0.36% and 0.22% at speed of 9, 10, 11, 12, 13 and 14 Knots respectively.

4. The results of grid study for KCS and Delta hip proved that the computational demands of the wave generation by interFoam is rather high. In OpenFOAM, The order of wave

generation convergence varies around 1 and is lower than the desired value that is 2 (2^{nd} order discretization). This is because of the first order type of initialization of the wave field in the model. It has been observed that the grid resolution is highly dependent on the wave steepness. The method used in this project, between 400 to 600 grids are needed for the wave generation in OpenFOAM. This implies the use of grid refinement at least around the interface.

5. It is observed that even though the wave deformation is the same almost for all the cases for KCS (turbulence modeling) and for Delta Ship (Seven speeds), turbulent quantities can still be compared due to the variety of different turbulence models which are used during the simulations for this project.

6. We arrived at the conclusion that the selection of OpenFOAM for the Ship Hydrodynamic applications (resistance calculations) is good enough to perform the simulations and result in postprocessing. There is a good perspective for OpenFOAM and CFD in the calculation of ship hydrodynamics.

7. At last, With any CFD toolbox (softwere), you can come up with both bad and excellent results and the results does not depend on the tool people use but the results depend on the skills people have.

9. FUTURE WORK

1. OpenFOAM will be extensively used in the future to help engineers in various computations tasks, especially when it comes to shipping hydrodynamic applications, resistance, propulsion etc. It is also expected that research and education in the field of simulations will continue to be supported by the governments and industry.

2. With the increasing computations power of modern computers, fine grids, smallest time steps and different kind of turbulence models will be used to study the performance of complete systems. Performing overset grids (latest), it will also be possible to study the motion of the ship in extreme waves, slamming and fluid-structure interaction. Furthermore, by coupling these CFD-methods, in which the RANSE-approach has used, we would be able to approach the wave propagation over a very larger distance;

Let’s make a brainstorming on what might be the future of the CFD codes. Yes, of course, this is just a science fiction. A thought experiment.

10.RECOMMENDATION

1. All the performed simulations are done with the same control settings except for the few speed of KCS Hull using a computer of 16GB RAM instead of a supercomputer. While performing the simulations it is observed that some schemes (FV Schemes mostly) will blow up because of the time derivation. It is observed that is we chose a smaller time step for this scheme, it will come up with a good result. This behavior is because of the fact that some dimensionless numbers like the Courant No. or the Fourier No. are needed to be in a certain acceptable range and in any case If we go below a limit, we have two options:
   I. Decreasing the time step;
   II. Numerical stabilization (By using different loops and relaxation).

   This is the reason why we observed different results using different.

2. OpenFOAM is an amazing tool in order to work with CFD related problems. But OpenFOAM is just a set of tools. The question is: How to use these tools properly and efficiently? I've noticed, that many people fail when adopting OpenFOAM. In Delta Marine, I've been delivering OpenFOAM training for approx. 3 month and many time face a new problem while working with it. I believe when learning OpenFOAM there exists a certain point in time – the breaking point. It is a very important milestone in OpenFOAM learning process. A newcomer starts with OpenFOAM and before reaching this point, many things go wrong, there is a new environment, there are many new terms, within each new step there is lots of uncertainty and many unknown elements.

3. During the completion of this project and months of sitting in the front of the computer by performing CFD calculations, I am pleased to recommend the most valuable experiences and ideas about CFD codes for the future.

One can notice that the CFD has drastically changed since a couple of years. Users' requirements are totally different for current CFD performance, comparing to their requirements a few years ago. Considering the latest trends, I have found four key features of the CFD code of the future. This is the list of those features. The future CFD code has to be:

- Automatable;
- Capable;
- Inferable;

- Scalable.

Hence, CFD codes should grow these features as soon as possible in order to be able to compete in the future demanding market at its very best.

4. The SST model is very well suitable for cross-diffusion which gives better outcomes as compared with the k-epsilon and k-omega turbulence models. Here based on my own experience, I realized that by using a blended function, one can also add the cross-diffusion when away from the wall.

5. My personal experience precisely indicates that lack of OpenFOAM skills sooner or later turns in a lack of CFD skills (not acceptable at any rate). People who know CFD, they learn OpenFOAM much faster than those who do not.

11. REFERENCES

1. Greenshields, C. (2015). *OpenFOAM, The Open Source CFD Toolbox.* OpenFOAM Foundation Ltd. Retrieved December 22, 2016

2. Aktar, S. (2012). *Drag analysis of different ship models using computational fluid dynamics tools.* Master Thesis, Bangladesh University of Engineering and Technology, Department of Mathematics, Dhaka. Retrieved January 24, 2017

3. Anthony F. Molland, S. R. (2011). *Ship Resistance and Propulsion.* New York: Cambridge University Press.

4. Baker, G. (1913). Methodical experiments with Mercantile Ship Forms. *Transactions of the Royal Institution of Naval Architects, 55*, pp. 164-178. New York. Retrieved December 17, 2016

5. Bakker, A. (2002). *Applied Computational Fluid Dynamics.* Lecture on Turbulence Models. Retrieved January 2017, from http://www.bakker.org/

6. Banks, J., Phillips, A., & Stephen, T. (2010). Free surface CFD prediction of components of Ship Resistance for KCS. *13th Numerical Towing Tank Symposium.* Germany. Retrieved 04 15, 2017, from http://www.uni-due.de/IST/ismt_nutts

7. Braithwaite, J. (2014). An Introduction to Hydrodynamics and Astrophysical Magnetohydrodynamics. CreateSpace Independent Publishing Platform.

8. Choi, J., Kim, J., Lee, H., Choi, B., & Lee, D. (2009). Computational predictions of ship-speed performance. *Journal of Maritime Science and Technology, 14*(3), 322–333. Retrieved 03 15, 2017

9. Choi, J., Min, K.-S., Kim, J., Lee, S., & Seo, H. (2010, 05). Resistance and propulsion characteristics of various commercial ships based on CFD results. *Ocean Engineering, 37*(7), 549–566. doi:https://doi.org/10.1016/j.oceaneng.2010.02.007

10. F. Stern, J. Y. (2012). Computational Ship Hydrodynamics: Nowadays and way Forward. *29th Symposium on Naval Hydrodynamics.* Gothenburg, Sweden.

11. Froud, W. (1874). *Experiment with HMS Greyhound.* Transaction of the Royal Institution of Naval Architects.

12. Froude, W. (1872). *Experiment on the surface-friction experienced by a plan moving through water.* Brighton: 42nd Report of the British Association for the Advancement of Science.

13. G.Kuiper, P. D. (1997). *Resistance and Propulsion of Ships.* Delft, Wageningen, Netherlands: MARIN, Maritime Research Institute Netherlands.

14. Larsson, L., & Raven, H. (2010). *Ship Resistance and Flow.* The Society of Naval Architects and Marine Engineers.

15. LSF Batch User's Guide. (1998, August). (Sixth). Canada: Platform Computing Corporation. Retrieved December 28, 2016, from http://ls11-www.informatik.uni-dortmund.de/people/hermes/manuals/LSF/users.pdf

16. Magnus, W. (2013). *Benchmark and validation of Open Source CFD codes, with a focus on compressible and rotating capabilities, for integration on the SimScale platform.* Master Thesis, CHALMERS UNIVERSITY OF TECHNOLOGY, Department of Applied Mechanics, Gothenburg. Retrieved January 2016

17. Maric, T., Hopken, J., & Mooney, K. (2014). *The OpenFOAM Technology Primer*. Source flux. Retrieved December 27, 2016, from http://www.sourceflux.de/book

18. Montorfano, A., & Federico, P. (2015). *SnappyHexMesh: scalable & automatic mesh generation for OpenFOAM.* Lecture, Politecnico de Milano, Dipartimento di Energia, Retrieved 2016, from https://hpc-forge.cineca.it/files/CoursesDev/public/2015/Workshop_HPC_Methods_for_Engineering/snappyHexMesh.pdf

19. (2011). *OpenFOAM User Guide.* OpenFOAM Foundation. Retrieved May 29, 2017, from http://www.openfoam.org/docs/user/

20. Prakash, S., & Subramanian, A. (2010). Simulation of Propeller – hull Interaction using Ranse Solver. *The International Journal of Ocean and Climate Systems, 1,* 188-210.

21. Puig, J., Gamez, P., & Gustavo, R. (2014). *OpenFOAM Guide for beginners.* Retrieved December 27, 2016, from Foam House.

22. *Revolvy*. (2016, December 22). Retrieved December 22, 2016, from https://www.revolvy.com/main/index.php?s=OpenFOAM&item_type=topic

23. Rhie, C., & Chow, W. (1983, 11). A Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA, 21*(11), 1525-1530. doi:10.2514/3.8284

24. Sayma, A. (2009). *Computational Fluid Dynamic.* ISBN: 978-87-7681-430-4.

25. Simonsen, C., Otzen, J., Joncquez, S., & Stern, F. (2013). EFD and CFD for KCS heaving and pitching in regular head waves. *Journal of Marine Science and Technology, 18*(4), 435-455. doi:10.1007/s00773-013-0219-0

26. Taylor, D. (1908). The influence of midship section shape upon the resistance of Ship. *Society of Naval Architects and Marine Engineers*, *16*, pp. 162-175. New York. Retrieved December 26, 2016

27. Wilcox, D. (1988). Reassessment of the scale-determining equation for advanced turbulence models. *AIAA Journal, 26*(11), 1296-1360.

28. Wortley, S. (2013). *CFD Analysis of Container Ship Sinkage, Trim and Resistance.* Thesis, Curtin University, Department of Mechanical Engineering. Retrieved April 13, 2017

29. Zhang, Z.-R. (2010). Verification and validation for RANS simulation of KCS container ship. *9th International Conference on Hydrodynamics*, (pp. 932-939). Shanghai. doi:10.1016/S1001-6058(10)60055-8

30. Zhang, Z.-R., Hui, L., Song Ping, Z., & Feng, Z. (2006). Application of CFD in ship engineering design practice and ship hydrodynamics. *Journal of Hydrodynamics*, 317-321.

31. Zhirong, S., Decheng, W., & Pablo, M. (2015). Dynamic Overset grids in OpenFOAM with application to KCS self-propulsion and maneuvering. *Ocean Engineering, 108*, 287-306. doi:10.1016/j.oceaneng.2015.07.035

ANNEX

Annex-I-blockMesh

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F idle         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
    (-20 0 -10)
    (16 0 -10)
    (16 16 -10)
    (-20 16 -10)

    (-20 0 -1)
    (16 0 -1)
    (16 16 -1)
    (-20 16 -1)

    (-20 0 0.1418)
    (16 0 0.1418)
    (16 16 0.1418)
    (-20 16 0.1418)

    (-20 0 0.3418)
    (16 0 0.3418)
    (16 16 0.3418)
    (-20 16 0.3418)

    (-20 0 0.5418)
    (16 0 0.5418)
    (16 16 0.5418)
    (-20 16 0.5418)

    (-20 0 1.6)
    (16 0 1.6)
    (16 16 1.6)
    (-20 16 1.6)

    (-20 0 6)
    (16 0 6)
    (16 16 6)
    (-20 16 6)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (18 10 15) simpleGrading (1 1 0.15)
```

```
    hex (4 5 6 7 8 9 10 11) (18 10 15) simpleGrading (1 1 0.3)
    hex (8 9 10 11 12 13 14 15) (18 10 6) simpleGrading (1 1 1)
    hex (12 13 14 15 16 17 18 19) (18 10 6) simpleGrading (1 1 1)
    hex (16 17 18 19 20 21 22 23) (18 10 15) simpleGrading (1 1 2.5)
    hex (20 21 22 23 24 25 26 27) (18 10 10) simpleGrading (1 1 7.5)
);

edges
(
);

boundary
(
    atmosphere
    {
        type patch;
        faces
        (
            (24 25 26 27)
        );
    }
    inlet
    {
        type patch;
        faces
        (
            (1 2 6 5)
            (5 6 10 9)
            (9 10 14 13)
            (13 14 18 17)
            (17 18 22 21)
            (21 22 26 25)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
            (4 8 11 7)
            (8 12 15 11)
            (12 16 19 15)
            (16 20 23 19)
            (20 24 27 23)
        );
    }
    bottom
    {
        type symmetryPlane;
        faces
        (
            (0 3 2 1)
        );
    }
    side
    {
        type symmetryPlane;
        faces
        (
            (0 1 5 4)
            (4 5 9 8)
            (8 9 13 12)
            (12 13 17 16)
            (16 17 21 20)
```

```
            (20 21 25 24)
        );
    }
    midPlane
    {
        type symmetryPlane;
        faces
        (
            (3 7 6 2)
            (7 11 10 6)
            (11 15 14 10)
            (15 19 18 14)
            (19 23 22 18)
            (23 27 26 22)
        );
    }
);

mergePatchPairs
(
);

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

Annex-II-snappyHexMesh

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.3.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}

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

// Which of the steps to run
castellatedMesh false;
snap            false;
addLayers       true;


// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
{
    kcs_moeri.stl
    {
        type triSurfaceMesh;
        name hull;

        patchInfo
        {
            type wall;
        }
    }
    freesurface_1
    {
        type searchableBox;
      min (-0.5 -2 0.25);
      max (6.9 2 0.45);
    }
    freesurface_2
    {
        type searchableBox;
      min (-1.5 -3 0.3);
      max (7.2 3 0.4);
    }
    bow_1
    {
        type searchableBox;
      min (6.4 -0.3 -0.2);
      max (8.4 0.3 0.55);
    }
    bow_2
    {
        type searchableBox;
      min (7 -0.25 -0.1);
      max (7.8 0.25 0.5);
    }
    stern_1
```

```
    {
        type searchableBox;
       min (-0.4 -0.4 -0.2);
       max (2.7 0.4 0.55);
    }
    stern_2
    {
        type searchableBox;
       min (-0.25 -0.3 -0.1);
       max (2.5 0.3 0.5);
    }
};



// Settings for the castellatedMesh generation.
castellatedMeshControls
{

    // Refinement parameters
    // ~~~~~~~~~~~~~~~~~~~~~

    // If local number of cells is >= maxLocalCells on any processor
    // switches from from refinement followed by balancing
    // (current method) to (weighted) balancing before refinement.
    maxLocalCells 1000000;

    // Overall cell limit (approximately). Refinement will stop immediately
    // upon reaching this number so a refinement level might not complete.
    // Note that this is the number of cells before removing the part which
    // is not 'visible' from the keepPoint. The final number of cells might
    // actually be a lot less.
    maxGlobalCells 20000000;

    // The surface refinement loop might spend lots of iterations refining just a
    // few cells. This setting will cause refinement to stop if <= minimumRefine
    // are selected for refinement. Note: it will at least do one iteration
    // (unless the number of cells to refine is 0)
    minRefinementCells 20;

    // Number of buffer layers between different levels.
    // 1 means normal 2:1 refinement restriction, larger means slower
    // refinement.
    nCellsBetweenLevels 3;



    // Explicit feature edge refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies a level for any cell intersected by its edges.
    // This is a featureEdgeMesh, read from constant/triSurface for now.
    features
    (
         {
             file "kcs_moeri.eMesh";
             level 3;
         }
    );



    // Surface based refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies two levels for every surface. The first is the minimum level,
    // every cell intersecting a surface gets refined up to the minimum level.
    // The second level is the maximum level. Cells that 'see' multiple
```

```
// intersections where the intersections make an
// angle > resolveFeatureAngle get refined up to the maximum level.

refinementSurfaces
{
    hull
    {
        // Surface-wise min and max refinement level
        level (3 3);
    }
}

resolveFeatureAngle 60;


// Region-wise refinement
// ~~~~~~~~~~~~~~~~~~~~~

// Specifies refinement level for cells in relation to a surface. One of
// three modes
// - distance. 'levels' specifies per distance to the surface the
//   wanted refinement level. The distances need to be specified in
//   descending order.
// - inside. 'levels' is only one entry and only the level is used. All
//   cells inside the surface get refined up to the level. The surface
//   needs to be closed for this to be possible.
// - outside. Same but cells outside.

refinementRegions
{
   hull
    {
        mode        distance;
        levels      ((0.02 1));
    }
    freesurface_1
    {
        mode        inside;
        levels      ((1E15 1));
    }
    freesurface_2
    {
        mode        inside;
        levels      ((1E15 2));
    }
    bow_1
    {
        mode        inside;
        levels      ((1E15 1));
    }
    bow_2
    {
        mode        inside;
        levels      ((1E15 2));
    }
    stern_1
    {
        mode        inside;
        levels      ((1E15 1));
    }
    stern_2
    {
        mode        inside;
        levels      ((1E15 2));
    }
}
```

```
    // Mesh selection
    // ~~~~~~~~~~~~~~

    // After refinement patches get added for all refinementSurfaces and
    // all cells intersecting the surfaces get put into these patches. The
    // section reachable from the locationInMesh is kept.
    // NOTE: This point should never be on a face, always inside a cell, even
    // after refinement.
    locationInMesh (-0.7 0 0);


    // Whether any faceZones (as specified in the refinementSurfaces)
    // are only on the boundary of corresponding cellZones or also allow
    // free-standing zone faces. Not used if there are no faceZones.
    allowFreeStandingZoneFaces true;
}



// Settings for the snapping.
snapControls
{
    //- Number of patch smoothing iterations before finding correspondence
    //  to surface
    nSmoothPatch 3;

    //- Relative distance for points to be attracted by surface feature point
    //  or edge. True distance is this factor times local
    //  maximum edge length.
    //    tolerance 4.0;
    tolerance 1.0;

    //- Number of mesh displacement relaxation iterations.
    nSolveIter 300;

    //- Maximum number of snapping relaxation iterations. Should stop
    //  before upon reaching a correct mesh.
    nRelaxIter 5;

    nFeatureSnapIter 10;
}



// Settings for the layer addition.
addLayersControls
{
    // Are the thickness parameters below relative to the undistorted
    // size of the refined cell outside layer (true) or absolute sizes (false).
    relativeSizes true;

    // Per final patch (so not geometry!) the layer information
    layers
    {
        hull
        {
            nSurfaceLayers 8;
        }
    }

    // Expansion factor for layer mesh
    expansionRatio 1.1;


    // Wanted thickness of final added cell layer. If multiple layers
    // is the thickness of the layer furthest away from the wall.
    // Relative to undistorted size of cell outside layer.
    // See relativeSizes parameter.
```

```
    finalLayerThickness 0.5;

    // Minimum thickness of cell layer. If for any reason layer
    // cannot be above minThickness do not add layer.
    // See relativeSizes parameter.
    minThickness 0.07;

    // If points get not extruded do nGrow layers of connected faces that are
    // also not grown. This helps convergence of the layer addition process
    // close to features.
    // Note: changed(corrected) w.r.t 17x! (didn't do anything in 17x)
    nGrow 0;


    // Advanced settings

    // When not to extrude surface. 0 is flat surface, 90 is when two faces
    // are perpendicular
    featureAngle 180;

    // a maximum number of snapping relaxation iterations. Should stop
    // before upon reaching a correct mesh.
    nRelaxIter 5;

    // Number of smoothing iterations of surface normals
    nSmoothSurfaceNormals 1;

    // Number of smoothing iterations of interior mesh movement direction
    nSmoothNormals 3;

    // Smooth layer thickness over surface patches
    nSmoothThickness 10;

    // Stop layer growth on highly warped cells
    maxFaceThicknessRatio 0.5;

    // Reduce layer growth where ratio thickness to medial
    // distance is large
    maxThicknessToMedialRatio 0.3;

    slipFeatureAngle 20;

    // Angle used to pick up medial axis points
    // Note: changed(corrected) w.r.t 17x! 90 degrees corresponds to 130 in 17x.
    minMedianAxisAngle 90;

    // Create buffer region for new layer terminations
    nBufferCellsNoExtrude 0;


    // Overall max number of layer addition iterations. The mesher will exit
    // if it reaches this number of iterations; possibly with an illegal
    // mesh.
    nLayerIter 50;

    // Max number of iterations after which relaxed meshQuality controls
    // get used. Up to nRelaxIter it uses the settings in meshQualityControls,
    // after nRelaxIter it uses the values in meshQualityControls::relaxed.
    nRelaxedIter 20;
}



// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls
{
    #include "meshQualityDict"
```

```
}


// Advanced

// Flags for optional output
// 0 : only write final meshes
// 1 : write intermediate meshes
// 2 : write volScalarField with cellLevel for postprocessing
// 4 : write current intersections as .obj files
debug 0;


// Merge tolerance. Is fraction of the overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1E-6;


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

Annex-III-lsf script

```
#!/bin/bash --login
# initializing bash shell as login so that it would remember environment variables

#BSUB -a intelmpi          #To get information about recently completed jobs.
#BSUB -q mid               #Submits job to specified queues
#BSUB -m karadeniz         #Runs job on one of the specified hosts.
#BSUB -P shauor            #Assigns job to specified project
#BSUB -n 8                 #Specifies min & max no. of processors for a parallel job
#BSUB -J "KCS"              #Assigns the specified name to the job.
#BSUB -e KCS.e.%J          #Appends the standard input to a file
#BSUB -o KCS.o.%J          #Appends the standard output to a file

# source file location
# ------------------------------
source /rs/progs/openfoam/openfoam-2.3.0_gcc_impi/foam-extend-3.0/etc/bashrc

# DESCRIPTION OF THE EXECUTION OF PROGRAM
# ------------------------------
mpirun.lsf interDyMFoam-parallel-case /RS/users/mmuhammad_tufail_shahzad/shauor/kCS/
```

Annex-IV PolyMesh Discription- Boundary Condition

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.1                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      binary;
    class       polyBoundaryMesh;
    location    "0.02/polyMesh";
    object      boundary;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

7
(
    atmosphere
    {
        type            patch;
        nFaces          180;
        startFace       14440717;
    }
    inlet
    {
        type            patch;
        nFaces          1260;
        startFace       14440897;
    }
    outlet
    {
        type            patch;
        nFaces          1260;
        startFace       14442157;
    }
    bottom
    {
        type            symmetryPlane;
        inGroups        1(symmetryPlane);
        nFaces          180;
        startFace       14443417;
    }
    side
```

```
    {
        type            symmetryPlane;
        inGroups        1(symmetryPlane);
        nFaces          37971;
        startFace       14443597;
    }
    midPlane
    {
        type            symmetryPlane;
        inGroups        1(symmetryPlane);
        nFaces          2268;
        startFace       14481568;
    }
    hull
    {
        type            wall;
        inGroups        1(wall);
        nFaces          211780;
        startFace       14483836;
    }
)

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

Annex-V-FV Scheme

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
    limitedGrad     cellLimited Gauss linear 1;
}

gradSchemes
{
    default         Gauss linear;
    grad(U) cellMDLimited Gauss linear 1;
}

divSchemes
{
    div(rhoPhi,U)   Gauss linearUpwindV grad(U);//Gauss upwind
    div(phi,alpha)  Gauss interfaceCompression vanLeer01;
    div(phirb,alpha) Gauss linear;//Gauss upwind
    div(phi,k)      Gauss linear; //Gauss upwind;
    div(phi,omega)  Gauss linear; //Gauss upwind;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default Gauss linear limited corrected 0.33;
}
```

```
interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         none;
    snGrad(p_rgh)        limited corrected 0.33;
    snGrad(rho)            limited corrected 0.33;
    snGrad(alpha.water)    limited corrected 0.33;
}

wallDist
{
    method meshWave;
}


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

Annex-VI-FV Solution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     3.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "alpha.water.*"
    {
        nAlphaCorr      1;
        nAlphaSubCycles 3;
        cAlpha          0.5;
        icAlpha         0;

        MULESCorr       yes;
        nLimiterIter    10;
        alphaApplyPrevCorr  yes;

        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-8;
        relTol          0;
        minIter         1;
    }

    pcorr
    {
        solver          PCG;

        preconditioner
        {
            preconditioner  GAMG;
```

```
        smoother        GaussSeidel;
        agglomerator    faceAreaPair;
        mergeLevels     1;
       smoother DICGaussSeidel;
       nPreSweeps 0;
       nPostSweeps 2;
       nBottomSweeps 2;
        nCellsInCoarsestLevel 10;
        cacheAgglomeration true;

        tolerance       1e-5;
        relTol          0;
    };

    tolerance       1e-5;
    relTol          0.001;
};
pcorrFinal
{
    $pcorr;
    relTol          0;
}

p_rgh
{
    solver          GAMG;

    smoother        DIC;
    agglomerator    faceAreaPair;
    mergeLevels     1;
    nCellsInCoarsestLevel 10;
    cacheAgglomeration true;

    tolerance       1e-7;
    relTol          0.01;
};

p_rghFinal
{
    $p_rgh;
    relTol          0;
}

"(U|k|omega).*"
{
```

```
        solver          smoothSolver;

        smoother        symGaussSeidel;
        nSweeps         1;

        tolerance       1e-7;
        relTol          0;
        minIter         1;
    };
}

relaxationFactors
{
    "(p_rgh).*"     0.3;
    "(U).*"         0.7;
    "(k).*"         0.6;
    "(omega).*"             0.6;
    alpha           0.2;
}

PIMPLE
{
    momentumPredictor   yes;

    nOuterCorrectors    5;
    nCorrectors         3;
    nNonOrthogonalCorrectors 0;
    correctPhi      yes;
    moveMeshOuterCorrectors yes;
    turbOnFinalIterOnly yes;
}



cache
{
    grad(U);
}


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

Annex-VII-DynamicMeshdict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      dynamicMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dynamicFvMesh       dynamicMotionSolverFvMesh;

motionSolverLibs    ("libsixDoFRigidBodyMotion.so");

solver              sixDoFRigidBodyMotion;

sixDoFRigidBodyMotionCoeffs
{
    patches         (hull);
    innerDistance   0.3;
    outerDistance   1;

    centreOfMass    (3.532 0 0.2304);
    mass            824.5;
    momentOfInertia (138.98 2730.02 2730.02);
    rhoInf          1;
    report          on;

    value           uniform (0 0 0);

    accelerationRelaxation 0.4;

    solver
    {
        type Newmark;
    }
```

```
    constraints
    {
        zAxis
        {
            sixDoFRigidBodyMotionConstraint line;
            direction     (0 0 1);
        }
        yPlane
        {
            sixDoFRigidBodyMotionConstraint axis;
            axis          (0 1 0);
        }
    }
}


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

Annex-VIII-Python Scripts for Visualization

```
#!/usr/bin/python

import os
import sys
import math

forces_file = "postProcessing/forces/0/forces.dat"

if not os.path.isfile(forces_file):
	print "Forces file not found at "+forces_file
	print "Be sure that the case has been run and you have the right directory!"
	print "Exiting."
	sys.exit()

def line2dict(line):
	tokens_unprocessed = line.split()
	tokens = [x.replace(")","").replace("(","") for x in tokens_unprocessed]
	floats = [float(x) for x in tokens]
	data_dict = {}
	data_dict['time'] = floats[0]
	force_dict = {}
	force_dict['pressure'] = floats[1:4]
	force_dict['viscous'] = floats[4:7]
	force_dict['porous'] = floats[7:10]
	moment_dict = {}
	moment_dict['pressure'] = floats[10:13]
	moment_dict['viscous'] = floats[13:16]
	moment_dict['porous'] = floats[16:19]
	data_dict['force'] = force_dict
	data_dict['moment'] = moment_dict
	return data_dict

time = []
press = []
```

```
viscous = []
drag = []
with open(forces_file,"r") as datafile:
	for line in datafile:
		if line[0] == "#":
			continue
		data_dict = line2dict(line)
		time += [data_dict['time']]
		press += [data_dict['force']['pressure'][0]]
		viscous += [data_dict['force']['viscous'][0]]
		drag += [data_dict['force']['pressure'][0] + data_dict['force']['viscous'][0]]
datafile.close()

outputfile = open('forces.txt','w')
for i in range(0,len(time)):
	outputfile.write(str(time[i])+'	'+str(press[i])+'	'+str(viscous[i])+'	'+str(drag[i])+'\n')
outputfile.close()

os.system("./gnuplot_script.sh")
```