Politecnico di Milano

Facoltà di Ingegneria Industriale
Corso di Laurea in Ingegneria Aeronautica

Numerical Simulations of a Boat's Hull With an Open-Source CFD Code

Relatore:
Prof. Alfio QUARTERONI
Correlatori:
Prof. Nicola PAROLINI
Ing. Matteo LOMBARDI

Tesi di Laurea di:
Davide Lupo CONTI
Matr. 734455

Anno Accademico 2010-2011

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Alfio Quarteroni, for offering me the possibility of working on a subject that is my real passion: fluid dynamics applied to sailing boats. Working on this thesis gave me the possibility to have a look ˜from the inside˝to one of the greatest America's Cup teams of all time and that week at the Alinghi base in Valencia will probably be the closest thing to the Cup i will ever live in my entire career.

I feel in great in debt with Prof. Nicola Parolini who gave me a great hand, especially in the second half of this 'long' thesis. His knowledge and passion for this subject is something really remarkable. I really appreciated his patience and his capability and love for teaching, qualities not always present in researchers.

A huge thank you goes to Eng. Teo Lombardi who gave me a massive support. His help was crucial, in every moment of this almost two year work, especially at the beginning when I came to Lausanne with very little knowledge of CFD and almost none in Linux and C++. His patience and comprehension despite all my mistakes was very remarkable. I still owe you a ride on my 49er, don't think I forgot.

A big sign of affection and appreciation goes to Prof. Massimiliano Lanz, for understanding my need for following the dream of the America's Cup and for giving me the ˜green card ˝for doing my thesis with another department. I'll always feel grateful for that and for all the non-standard academic experiences concerning sailing I managed to do thanks to him: Mille e Una Vela, wind tunnel testing, and all the rest. Thank you.

I'll always be thankful towards C.C.C.C.. The writing of this thesis would still be in an embryonic stage without her help.

I would like to thank all the CMCS group in Lausanne: Gian, Simone, Capitan Lesinaigo, Paolo, Laura, Cristiano, Andrea, and all the rest I forgot. A big hug goes to The Gionnis (Anwar and Alberto) and La Giovanna (Cristina). Their help was fundamental to survive in Lausanne. Thank you.

I can't do without remembering the friends of all these years spent in Bovisa, il Catta e il Cassa, Filippo, il Biondo ed il Chiozz, il Costa, L'Eleonora,

Dario, il Droa, Andrea, my partner for the bachelor thesis, and the folks from PGV: Alessandro, Giuseppe and San Piva obviously. I can't forget Fabio, Rocco, Vincenzo, Paolo, Luca and Sara. Studying i Mante with them was a bit easier. Let's hope that all this suffering will be useful one day. Thanks also to Giulio Romanelli for his help and suggestions.

A big hug goes to V. and all the Carle and Carli... probably the most solid group I've ever met. Marco, my crew, Silvia and Vittorio deserve to be remembered also. It's quite helpful to have sailors to talk to in Uni! A big thank you to Lucrezia for her affection and her support, despite my bad character and the unfortunate timing...

Finally I would like to say hello to Chicca e gli Zii, Luca e la Bea, i Nonni e la Nonna, always present when needed. My Mom and Dad's patience, understanding and support has always been something admirable, especially when I "decided "to last my university two years more. The biggest hug goes to my sisters, Gaia and Camilla. Life wouldn't be the same without you.

Thank You.

Abstract

The study of the performance and dynamics of a boat's hull has always been object of interest in many fields, ranging from commercial shipyards to high performance sail crafts. This type of analysis was initially carried out with the use of empirical models, simplified formulas and towing-tank tests. Nowadays, thanks to technological progress, complex numerical tools are employed. Computational Fluid Dynamics (CFD) has become, in the last decade, much more accurate, cheaper and faster compared to the past. The research that has been carried out permits, nowadays, to simulate almost any condition a vessel can sail in. Towing-tank testing still remains very important to validate numerical results, but has become much less employed because of the high costs (instrumentation, model realization, delivery time, ...), of the smaller range of conditions that can be tested and of the model-ship correlation (which does not exist in CFD, since full-scale virtual model can be simulated).

The state of the art of Computational Fluid Dynamics for naval engineering problems has been reached with commercial CFD programs, which have a high license cost and do not permit ad-hoc modifications of the source code by the user. The goal of this thesis is to show that these two problems can be avoided adopting a "free-approach"to Computational Fluid Dynamics. For this purpose the Open-Source CFD library OpenFOAM was used. OpenFOAM solves the Reynolds Averaged Navier-Stokes Equations (RANS) using the Finite Volume discretization (FV) and a segregated approach. The two solvers used (*interFoam* and *interDyMFoam*) implement the Volume Of Fluid (VOF) method to model the free-surface. The dynamic free surface-flow solver *interDyMFoam* is coupled with a Six Degree Of Freedom (6DOF) solver which uses quaternions ad septernions to integrate the rigid body motion equations that model the hull dynamics.

In order to complete the simulation procedure independently of commercial softwares, the adoption of Open-Source mesh generation tool has also been considered. The meshing process is based on: the *snappyHexMesh* mesher and different utilities embedded in OpenFOAM, other Open-Source functionalities, and some ad-hoc coding. The results from simulations carried

out using these meshes are compared to others performed with meshes made using a commercial grid generator. Simulations with fixed hull in flat water and in presence of waves were carried out and results (drag coefficients, wave lines and wave patterns) are compared to experimental ones and to those obtained by commercial CFD codes found in literature. Free trim and free sink and trim simulations were computed, and drag coefficients, trim angles and sinkage are compared to available data.

Results show that a "free-approach"to Computational Fluid Dynamics for naval engineering problems is possible and the state of the art is fully reached. Drag coefficient shows an error of less than 6% with respect to experimental data. Trim angles and sinkage are correctly predicted and the hull dynamics in presence of waves is correctly simulated.

Keywords

Open-Source, Computational Fluid Dynamics, OpenFOAM, Free-Surface.

Sommario

Lo studio delle prestazioni e della dinamica dello scafo di una barca è sempre stato oggetto di interesse per tutti i settori del mondo nautico-navale, dalle navi commerciali fino ad imbarcazioni a vela ad alte prestazioni. Inizialmente questo tipo di analisi veniva fatto per mezzo di modelli empirici, formule semplificate e prove in vasca navale. Oggigiorno, grazie allo sviluppo tecnologico in campo informatico, vengono utilizzati strumenti numerici decisamente complessi. La Fluidodinamica Computazionale (CFD) nell'ultima decade è diventata sempre più economica e veloce e ha fornito risultati sempre più accurati. La ricerca in questo campo ha permesso di simulare numericamente qualsisasi condizione in cui una barca si trovi a dover navigare. Prove sperimentali in vasca navale vengono sempre meno utilizzate a causa degli elevati costi (di strumentazione, realizzazione del modello, per esempio), del minor numero di condizioni diverse di prova e del problema della correlazione modello in scala-modello reale (problema che non esiste in campo numerico, vista la possibilità di simulare in scala reale), ma rimangono sempre e comunque di vitale importanza, soprattutto per la validazione dei risultati numerici.

Lo stato dell'arte per la Fluidodinamica Computazionale è stato raggiunto con l'utilizzo di codici commerciali, che hanno alti costi di licenza e non offrono alcuna possibilità all'utilizzatore di cambiare i codici sorgenti, se necessario. L'obiettivo di questa tesi è quello di dimostrare che un approccio"libero"alla Fluidodinamica Computazionale è possibile. Per questo scopo, è stato utilizzato OpenFOAM, una libreria CFD Open-Source. Con OpenFOAM vengono risolte le equazioni mediate di Reynolds (RANS) con un approccio segregato, utilizzando una discretizzazione a Volumi Finiti (FV). Nei due solutori utilizzati, statico (*interFoam*) e dinamico (*interDyMFoam*), la modellazione della superficie libera viene fatta con il metodo della Frazione Di Volume (VOF, dall'inglese Volume Of Fluid). Il solutore dinamico per flussi a superficie libera è accoppiato a un solutore a sei gradi di libertà (6DOF) che utilizza septermioni e quaternioni per integrare le equazioni di moto dell'imbarcazione (modellata come un corpo rigido).

Volendo completare la procedura di simulazione in maniera completa-

mente ¨libera¨, quindi senza l'utilizzo di codici commerciali si è vagliato l'utilizzo di generatori di griglia Open-Source. Il processo di grigliatura si basa sull'utilizzo del meshatore *snappyHexMesh* e di alcune applicazioni fornite con OpenFOAM, di altre Open-Source non fornite in OpenFOAM e di alcune funzioni scritte appositamente. I risultati delle simulazioni con le griglie generate con questo processo, vengono confrontati con quelli forniti da simulazioni con griglie generate da un meshatore commerciale. Sono state effettuate simulazioni a scafo fissato, in acqua piatta e con moto ondoso imposto, ed i risultati, in termini di coefficienti di resistenza, linee d'onda e moto ondoso, sono stati confrontati con risultati sperimentali e numerici di codici commerciali, disponibili in letteratura. Il confronto con risultati numerici e sperimentali disponibili è stato fatto anche per i coefficienti di resistenza, gli angoli di beccheggio ed affondamento ottenuti da simulazioni dinamiche, ovvero con i gradi di libertà di beccheggio ed affondamento non vincolati.

I risultati mostrano che un approccio ¨libero¨alla Fluidodinamica Computazionale per problemi tipici dell'ingegneria navale è possibile e che lo stato dell'arte è stato raggiunto. I coefficienti di resistenza si discostano dai valori sperimentali per meno del 6%, gli angoli di beccheggio ed i valori dell'affondamento sono congruenti con i valori disponibili in letteratura e il comportamento dinamico dello scafo in presenza di onde è correttamente simulato.

Parole Chiave

Open-Source, Fluidodinamica Computazionale, OpenFOAM, Superficie Libera.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

The study of the performance and dynamics of a boat's hull has always been object of interest in many fields, ranging from commercial shipyards to high performance sail crafts. This type of analysis was initially carried out using empirical models and simplified formulas. Nowadays, thanks to technological progress, complex numerical tools are employed.

Commercial shipyards are interested in the analysis of a hull's behavior in many different conditions, especially extreme ones, when the safety of the vessel must be guaranteed. This type of analysis also allows the designers to assess the ship maneuverability: a ship that can hardly be maneuvered without the use of tugboats may result inappropriate. Furthermore, performance prediction has recently become significantly important, because of the energy crisis and the relative increase in oil price. At cruising speed an accurate hull design can significantly reduce drag, bringing to a lower need for propulsion, therefore to higher efficiency and lower cost.

The study of hull dynamics is very important in the field of race sailing boats, where performance is the top priority: predicting the performance of a hull as accurately as possible, both in the presence of waves and during maneuvering is fundamental. During the last decade, hydro-elastic behavior has become very influent in high performance boats, due to the large employment of low weight high-tech materials, such as carbon fiber and titanium. This implies the need to correctly account for the influence of hydro-elastic interaction of bulbs, dagger board or fins on drag (performance) during the design process. This topic is well beyond the reach of this thesis. The evaluation of loss of speed during maneuvering and of the time needed to reaccelerate to maximum speed is important as well: in match-races such as America's Cup, for example, up to sixty maneuvers can occur during a one hour race.

The analysis of hull behavior is also relevant in aeronautics applications. A huge number of seaplanes exists and the design of their floats is crucial. Floats that make landing on water safe and easy for the pilot typically are not the most efficient when flying: this is why the effective behavior of a seaplane during take-off and landing needs to be carefully evaluated. Concluding, the filling of water tanks of Canadair is an extremely delicate maneuver, and the design of different hulls-fuselage and water-intakes can make these maneuvers safer.

1.2 CFD Tool: OpenFOAM

The OpenFOAM (Open Field Operation and Manipulation) CFD Toolbox is a free, open source CFD (Computational Fluid Dynamics) software package. It has a large user base across many areas of engineering and science, both from commercial and academic organizations. OpenFOAM has an extensive range of features to solve different problems, such as complex fluid flows involving chemical reactions, turbulence and heat transfer as well as solid dynamics and electromagnetic.

The core technology of OpenFOAM is a flexible set of efficient C++ modules. Their structure and hierarchical design is such that its solvers, utilities and libraries are fully extensible. Modules are used to build a wealth of tools; solvers to simulate specific problems in engineering mechanics; utilities to perform pre- and post-processing tasks such as simple data manipulations, visualization and mesh processing; libraries implementing constitutive laws and physical models, to create toolboxes that are accessible to the solvers and utilities. Numerous pre-configured base solvers, utilities and libraries are supplied, so that OpenFOAM can also be used like any typical simulation package without the need of extra coding.

OpenFOAM is based on finite volume discretization to solve systems of partial differential equations on a vast range 3D structured and unstructured mesh of polyhedral cells. The fluid flow solvers are developed within a robust, implicit, pressure-velocity, iterative solution framework, although alternative techniques are applied to other continuum mechanics solvers. Domain decomposition based parallelism is fundamental to the design of OpenFOAM and is integrated at a low level so that solvers can generally be developed without the need for any 'parallel-specific' coding.

One of the main reasons why OpenFOAM was adopted for this project is that it is Open-Source, distributed under GNU license. Commercial CFD codes typically have two main problems: very high license costs and inaccessibility to source code, making it impossible to modify the code if necessary.

These problems are not present in OpenFOAM and control over the code is (theoretically speaking) complete. OpenFOAM has a large user base and the community, which is rapidly expanding, communicates on-line on a quite active forum, www.cfd-online.com. However, there are different concerns regarding this CFD library. OpenFOAM is at first not very user-friendly and does not have a graphical interface, usually found in commercial codes. It requires a good C++ knowledge to understand the sources and very small official documentation is available.

1.3 Thesis Outline

In this thesis the mathematical and practical problem regarding the dynamics of a floating body in free surface flow is introduced. In Chapter 2 the base equations modelling the flow field are presented, followed by the model for viscous, incompressible flow with the free-surface modelling. In Chapter 3 the numerical discretization and solution of the equations implemented in OpenFOAM is described. The mathematical model for the dynamics of a hull is discussed, together with the coupling for fluid-structure interaction in Chapter 4. In Chapter 5 the numerical results that we have botained are reported and discussed.

Chapter 2

A Mathematical Model For Free-Surface Flows

The mathematical model for free surface flows in naval engineering problems must correctly represent the large number of physical phenomena involved in the motion of a boat. Therefore, in order to accurately simulate the hydrodynamic flow, the model must account for viscosity effects, turbulence, free-surface wave motion and, possibly, fluid-structure interaction.

The fluid dynamic model adopted for this work is presented in this chapter. The Navier-Stokes equations are introduced, as well as their Reynolds-averaged form (RANS) which are effectively implemented in OpenFOAM. Finally, the $\kappa - \omega SST$ turbulence model, which was adopted for all the simulations, is briefly described as well as the Front-Capturing Method for the free-surface description.

2.1 Continuum Mechanics Equations

2.1.1 Navier-Stokes Equations

The physical nature of the problems we are concerned with is in the field of continuum mechanics, since the characteristic length-scales and time-scales are considerably higher with respect to the discrete structure of the matter. The equations that model the fluid behavior are the so called Navier-Stokes Equations. Typical velocity and length scales are such that Mach number is nearly 0 ($M \approx 0$) and Reynolds number results particularly high (Re $[10^6 \div 10^7]$). This implies that Navier-Stokes Equations for viscous incompressible flows are the governing equations that correctly describe the flow field studied (in the domain Ω occupied by the fluid):

$$\begin{cases} \dfrac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U} \otimes \mathbf{U}) = -\dfrac{1}{\rho}\nabla p + \nabla \cdot (\nu \nabla \mathbf{U}) + \mathbf{g} \\ \\ \nabla \cdot \mathbf{U} = 0 \end{cases} \qquad in \quad \Omega \times [0, T] \tag{2.1}$$

where: ρ is the density, p is the pressure, $\mathbf{U}$ is the velocity, t is time, ν is the kinematic viscosity and $\mathbf{g}$ is the gravitational acceleration. As the Mach number is nearly zero, the fluid can be considered as incompressible, therefore the continuity equation is described by the divergence of the velocity being equal to zero ($\nabla \cdot \mathbf{U} = 0$ in equation (2.1)). Problem (2.1) is well-posed, if suitable boundary and initial condition are given. These are discussed in subsection 5.1.

2.1.2 RANS Equations

As previously stated, in typical naval problems, the Reynolds number is particularly high ($Re \approx 10^7$). This involves the presence of turbulence. Turbulence can be considered as a state of continuous instability of the flow, where fluctuations can be separated by mean values of the field. It is characterized by several features, such as three-dimensionality, instationarity, the presence of vortices (turbulent structures responsible for rapid change of turbulence intensity, through a strong mixing). Different time-scales and length-scales are also present, from the smallest, that characterize the Kolmogorov structures, to the biggest, that are comparable to the size of the physical problem. In order to correctly simulate a turbulent flow, the smallest turbulent structures also have to be correctly kept into account. Different approaches can be adopted: Direct Numerical Simulation (DNS), Large Eddy Simulation (LES) or Reynolds Averaged Navier-Stokes (RANS).

The DNS approach consists in a numerical integration of the governing equations over the whole range of turbulent scales. This implies the number of nodes to be at least equal to L/η , in each coordinate direction, where L is the geometrical length scale (length of the boats hull) and η is Kolmogorov's length-scale. This ratio is proportional to a power of Reynolds number ($L/\eta \sim Re^{3/4}$). This means that a three dimensional mesh should have at least $Re^{9/4}$ nodes, which implies an unaffordable computational cost for any engineering problem.

The LES approach separates the big scales from the small scales of the turbulent field. In order to do this a spatial filter is applied. Therefore large scale structures (super-grid scales) are resolved by a numerical integration on

a given mesh, while small scales (sub-grid scales) are modelled. As the mesh becomes finer, sub-grid scales become smaller and convergence to the DNS approach is achieved. The underlying idea of LES approach is that small turbulent scales can be considered fairly homogeneous and isotropic, thus relatively easy to model. Small scales are modeled from information obtained by the resolved field, based on the idea if similarity. From a computational point of view, the LES approach is far less expensive than the DNS approach, but still often is too expensive to be applied to typical naval and engineering problems. For these reasons the RANS approach is still the most commonly adopted.

The RANS approach relies on the idea of separating the variables in two components, a mean part and a fluctuating part:

$$\phi(x,t) = \overline{\phi}(x,t) + \phi'(x,t) \tag{2.2}$$

where $\phi'(x,t)$ is the fluctuating component while $\overline{\phi}(x,t)$ is the mean value, given by:

$$\overline{\phi}(x,t) = \lim_{T\to\infty} \frac{1}{T} \int_0^T \phi(x,t) \tag{2.3}$$

Applying this decomposition to all the variables of the incompressible viscous Navier-Stokes equations, keeping in mind the average operators properties:

- Linearity: $\overline{u + \lambda v} = \overline{u} + \lambda \overline{v}$
- Commutation between average and derivative: $\overline{\frac{\partial u}{\partial x} u} = \frac{\partial u}{\partial x} \overline{u}$
- Average of product: $\overline{u\overline{v}} = \overline{uv}$
- Invariance of average: $\overline{\overline{u}} = \overline{u}$
- Average of fluctuation: $\overline{u'} = 0$

we obtain the Reynolds Averaged Navier-Stokes equations:

$$\begin{cases} \dfrac{\partial \overline{\mathbf{U}}}{\partial t} + \nabla \cdot (\overline{\mathbf{U}} \otimes \overline{\mathbf{U}}) = -\dfrac{1}{\rho} \nabla \overline{p} + \nabla \cdot (\nu \nabla \overline{\mathbf{U}}) + \nabla \cdot (\overline{\mathbf{U}' \otimes \mathbf{U}'}) + \mathbf{g} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad on \quad \Omega \times [0,T] \\ \nabla \cdot \overline{\mathbf{U}} = 0 \end{cases} \tag{2.4}$$

where the term $\overline{\mathbf{U}' \otimes \mathbf{U}'}$ is the Reynold Stress Tensor, which needs to be modelled in order to close the problem. There are two main approaches to solve this problem. The first one consists in directly modelling transport

equations for the Reynolds Stress Tensor components. The second method is based on the Boussinesq's hypothesis which assumes that the turbulent stress tensor can be expressed as a linear function of the strain rate tensor, in analogy with Stokes law for laminar flows. Therefore:

$$\overline{\mathbf{U}' \otimes \mathbf{U}'} = \nu_t(\nabla \mathbf{U} + (\nabla \mathbf{U})^T) + \frac{2}{3}\kappa I \tag{2.5}$$

where ν_t is the turbulent eddy viscosity and $\kappa = \frac{1}{2}\|\overline{\mathbf{U}'}\|^2$ is the turbulent kinetic energy. Thanks to Boussinesq's hypothesis the problem of evaluating Reynolds Stress Tensor is brought back to the determination of a new variable: turbulent eddy viscosity: ν_t. Different models of turbulent eddy viscosity exist:

- zero-equation (Algebraic) models (Prandtl's mixing length)
- one-equation models (Spalart-Allmaras)
- two-equations models ($\kappa - \epsilon$, $\kappa - \omega$)

in this work, a particular type of two-equations model has been adopted: $\kappa - \omega SST$ Turbulence Model.

2.1.3 The $\kappa - \omega$ *Shear Stress Transport* ($\kappa - \omega SST$) Turbulence Model

The $\kappa - \omega SST$ turbulence model (by Menter [1], [2]) is a two-equation eddy-viscosity model which has become widely used. The Shear Stress Transport (SST) formulation combines the best aspects of the $\kappa - \epsilon$ and $\kappa - \omega$ turbulence models. The $\kappa - \omega$ turbulence model gives very good results near walls, without the need of extra damping functions, while its accuracy gets worse moving further away from walls. The $\kappa - \epsilon$ model instead, gives better results in free-stream. The SST formulation adopts two combination functions to switch from one model to the other, avoiding the common $\kappa - \omega$ problem that the model is too sensitive to the inlet free-stream turbulence properties. Authors who use the $\kappa - \omega SST$ model often praise it for its good behavior in adverse pressure gradients and separating flow. The $\kappa - \omega SST$ model does produce slightly too large turbulence levels in regions with large normal strain, like stagnation regions and regions with strong acceleration. This tendency is much less pronounced than with a normal $\kappa - \epsilon$ model though.

The equations for turbulent kinetic energy (κ) and specific dissipation rate (ω) are:

$$\begin{cases} \dfrac{\partial \kappa}{\partial t} + U_j \dfrac{\partial \kappa}{\partial x_j} = P_\kappa - \beta^* \kappa \omega + \dfrac{\partial}{\partial x_j}\left[(\nu + \sigma_k \nu_t)\dfrac{\partial k}{\partial x_j}\right] \\ \dfrac{\partial \omega}{\partial t} + U_j \dfrac{\partial \omega}{\partial x_j} = \alpha S^2 - \beta \omega^2 + \dfrac{\partial}{\partial x_j}\left[(\nu + \sigma_\omega \nu_t)\dfrac{\partial \omega}{\partial x_j}\right] \\ \qquad\qquad +2(1-F_1)\sigma_{\omega 2}\dfrac{1}{\omega}\dfrac{\partial \kappa}{\partial x_j}\dfrac{\partial \omega}{\partial x_j} \end{cases} \tag{2.6}$$

where the production term (P_κ) is given by:

$$P_\kappa = \min\left(\tau_{ij}\frac{\partial U_i}{\partial x_j}, 10\beta^*\kappa\omega\right) \tag{2.7}$$

The first combination (or interface) function yields:

$$\begin{cases} F_1 = \tanh\left\{\left\{\min\left[\max\left(\dfrac{\sqrt{\kappa}}{\beta^*\omega y}, \dfrac{500\nu}{y^2\omega}\right), \dfrac{4\sigma_{\omega 2}\kappa}{CD_{\kappa\omega}y^2}\right]\right\}^4\right\} \\ CD_{\kappa\omega} = \max\left(2\rho\sigma_{\omega 2}\dfrac{1}{\omega}\dfrac{\partial \kappa}{\partial x_i}\dfrac{\partial \omega}{\partial x_i}, 10^{-10}\right) \end{cases} \tag{2.8}$$

where $CD_{\kappa\omega}$ is a redistributive term. Finally the kinematic eddy viscosity is given by:

$$\nu_t = \frac{a_1 k}{\max(a_1\omega, SF_2)} \tag{2.9}$$

where F_2 is the second interface function:

$$F_2 = \tanh\left[\left[\max\left(\frac{2\sqrt{\kappa}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}\right)\right]^2\right] \tag{2.10}$$

Closure Coefficients are provided in the following table:

$\alpha_1 = \frac{5}{9}$	$\beta_1 = \frac{3}{40}$	$\beta^* = \frac{9}{100}$	$\sigma_{\kappa 1} = 0.85$	$\sigma_{\omega 1} = 0.5$
$\alpha_2 = 0.44$	$\beta_2 = 0.0828$		$\sigma_{\kappa 2} = 1$	$\sigma_{\omega 2} = 0.856$

Table 2.1: $\kappa - \omega SST$ Closure Coefficients

Initial and boudary conditions are generally imposed as a fixed value, given by the empirical formulas:

$$\kappa = \frac{3}{2}(UI)^2, \qquad \omega = C_\mu^{-\frac{1}{4}} \frac{\sqrt{\kappa}}{l} \tag{2.11}$$

where U is the free-stream velocity modulus, I is the free-stream turbulence intensity (typical values range from 1% to 5%), C_μ is a constat which has a value of 0.09 and l is the turbulent lenght-scale.

2.2 Free-Surface Flows

The peculiarity of flows of naval interest is clearly the presence of a free-surface. The determination of the position of the free-surface is of crucial importance because the value of resulting forces on a boat's hull noticeably depends on the effective waterline, therefore on the free-surface position. Several numerical methods for the determination of the free-surface have been presented in literature. They are typically classified into two major categories, Front Tracking and Front Capturing methods.

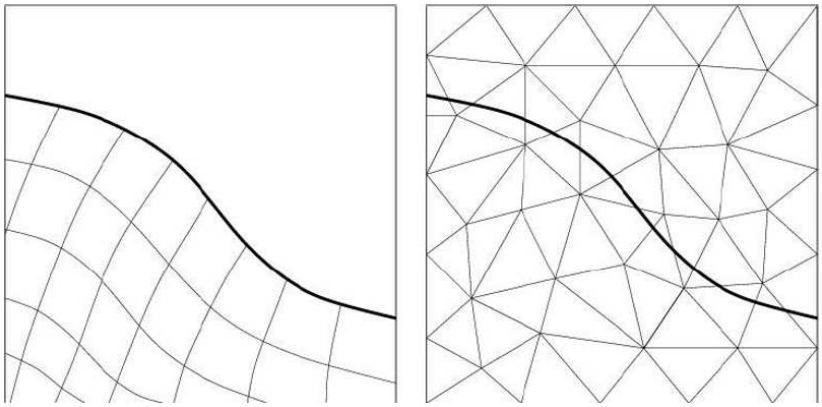


Figure 2.1: Grids for Front Tracking Method (left) and Front Capturing method (right)

2.2.1 Front Tracking Methods

Front Tracking Methods were the most used ones in the past. The free-surface is considered as a boundary of the computational domain, therefore only water is simulated. Adopting a Lagrangian approach, the computational mesh is adapted at each time-step in order to follow the free surface deformation correctly. As aforementioned, the free-surface is considered as a

mobile boundary of the domain and its position is defined by a function of time and two spatial coordinates:

$$z = h(x, y, t) \tag{2.12}$$

This approach is quite accurate in simulating free-surface flows with relatively simple free-surfaces, but is inadequate in case of breaking waves, which occur quite often in naval problems.

2.2.2 Front Capturing Methods

Front Capturing Methods are becoming the most widely used methods in naval engineering problems, thanks to their flexibility in dealing with complex free-surface problems. These methods are based on an Eulerian approach, therefore the computational domain and mesh are fixed. The interface between the two fluids is "captured" by solving an additional advection equation for a new proper variable. The two most used Front Capturing Methods are the Level-Set and Volume Of Fluid (VOF) methods; the latter is the one adopted in this work.

In Volume Of Fluid Method, the interface is captured solving an additional advection equation for the new variable, $\alpha = \alpha(\mathbf{x}, t)$, function of space ($\mathbf{x}$) and time ($t$), which represents the volume fraction of one fluid at each point:

$$\frac{\partial \alpha}{\partial t} + \mathbf{U} \cdot \nabla \alpha = 0 \quad in \quad \Omega \times [0, T] \tag{2.13}$$

Therefore α will vary between one and zero: when $\alpha = 1$ the cell will be full of water and when $\alpha = 0$ the cell will be full of air, while cells where $0 < \alpha < 1$ are the ones across the interface. Fluid properties at each point will be given by a weighted-average on the volume fraction:

$$\begin{cases} \rho(\mathbf{x}, t) = \rho_{H_2O}\alpha(\mathbf{x}, t) + \rho_{air}(1 - \alpha(\mathbf{x}, t)) \\ \nu(\mathbf{x}, t) = \nu_{H_2O}\alpha(\mathbf{x}, t) + \nu_{air}(1 - \alpha(\mathbf{x}, t)) \end{cases} \tag{2.14}$$

As previously stated, in order to correctly evaluate the resulting forces, the computation of the free-surface position has to be particularly accurate. Therefore, the interface (the zone where $0 < \alpha < 1$) must be as sharp as possible and solutions with diffused interface must be avoided. In order to do this, an accurate mesh refinement will be necessary near the zone where the interface will probably be captured and interface-compression schemes (which force the interface to span only few cells) are usually adopted.

2.3 Resolution of Fluid Dynamic Problems in a Moving Domain

The real position of a hull moving with respect to the flow is a priori unknown. This is because the sink and trim position of a hull depend on the resultant forces acting on it, which in turn depend on the hull's configuration. Therefore, when considering the interaction between the fluid and a moving structure, mesh motion should be considered. The Arbitrary Lagrangian Eulerian approach (ALE) for fluid dynamic equations allows to keep the mesh motion into account. The structural movement computed from the six degrees of freedom (6DOF) model implies a change in the fluid dynamic domain at each time-step. Mesh movement (or mesh morphing) must entirely follow the movement of the boundary, but at the same time it must avoid excessive cell distortions (especially skewness) and preserve a positive volume for each control volume, in order to avoid numerical issues. This can be done by solving a Laplace problem for mesh movement, in analogy with a generic elastic problem. The boundary position is known and by imposing an appropriate mesh stiffness, the elastic problem can be solved and the new mesh position can be computed. Mesh stiffness generally follows an inverse square law in order to deform the smallest cells of the domain near the hull as little as possible. As a consequence, Navier-Stokes equations must be reformulated to keep into account such mesh motion, and this is done with the ALE approach.

2.3.1 The ALE Approach

The following is a short discussion of how Navier-Stokes equations are treated in the ALE approach. Refer to [4] for further details. A continuous and bijective map is now introduced:

$$A_t \colon \Omega_0 \to \Omega_{A_t}, \mathbf{Y} \to \mathbf{y}(t, \mathbf{Y}) = A_t(\mathbf{Y}) \tag{2.15}$$

which transforms ALE coordinates $(t, \mathbf{Y})$ in $(t, \mathbf{y})$ coordinates for each $t \in I$, where I is the time-step. Furthermore

$$\Omega_{A_t} \equiv A_t(\Omega_0) = \Omega_t \tag{2.16}$$

where ω_0 is the original domain and Ω_t is the domain at time t. Domain velocity is defined as:

$$\widetilde{\mathbf{w}}(t, Y) = \frac{\partial}{\partial t}\mathbf{y}(t, \mathbf{Y}) \tag{2.17}$$

that corresponds to

$$\mathbf{w} = \widetilde{\mathbf{w}} \circ A_t^{-1}, i.e. \mathbf{w}(t, \mathbf{y}) = \widetilde{\mathbf{w}}(t, A_t^{-1}\mathbf{y}). \tag{2.18}$$

Let $\widetilde{f}$ be the composition of a function f with the ALE map, $\widetilde{f} = f \circ A_t$ and let $T_{\mathbf{Y}}$ be the ALE trajectory for each $\mathbf{Y} \in \Omega_0$ as

$$T_{\mathbf{Y}} = \{(t, \mathbf{y}(t, \mathbf{Y})), t \in I\}. \tag{2.19}$$

Let's define the ALE derivative of a function f as the derivative along the trajectory T_Y:

$$\frac{D^A}{Dt} f \colon I \times \Omega_t \to R, \frac{D^A}{Dt} f(t, \mathbf{y}) = \frac{\partial f}{\partial t}(t, \mathbf{Y}). \tag{2.20}$$

Therefore, we obtain:

$$\frac{D^A}{Dt} f = \frac{\partial f}{\partial t} + \mathbf{w} \cdot \nabla f \tag{2.21}$$

where the gradient is with respect to $\mathbf{y}$. The Jacobian matrix of the ALE map is defined as:

$$J^{A_t} = det(\frac{\partial \mathbf{y}}{\partial \mathbf{Y}}) \tag{2.22}$$

and satisfies the following equation:

$$\frac{D^A}{Dt} J^{A_t} = J^{A_t} \nabla \cdot \mathbf{w}. \tag{2.23}$$

Finally, it is shown that:

$$\frac{d}{dt} \int_{\Omega_t} f = \int_{\omega_t} \frac{\partial f}{\partial t} + \int_{\Omega_t} \nabla \cdot (f\mathbf{w}). \tag{2.24}$$

The Navier-Stokes equations can be rewritten for each $t \in I$ on the domain Ω_t as:

$$\frac{D^A}{Dt}(\mathbf{U}) + \nabla \cdot ((\mathbf{U} - \mathbf{w}) \otimes \mathbf{U}) = -\frac{1}{\rho}\nabla p + \nabla \cdot (\nu \nabla \mathbf{U}) + \mathbf{g} \quad on \quad \Omega \times [0, T] \tag{2.25}$$

where the ALE derivative has been highlighted.

2.3.2 OpenFOAM 6DOF Solver

The main focus of this work is to determine the dynamics of a boat's hull, therefore a convenient model has to be adopted. The hull can be considered as a rigid body, since possible structural deformations are in first approximation neglected. The equations describing the dynamics of a rigid body which is free to move in space are the classic Newton's laws for linear and angular momentum:

$$\begin{cases} m\ddot{\mathbf{X}}_{CG} = m\mathbf{g} + \mathbf{F}_{flow} + \mathbf{F}_{ext} \\ J\dot{\omega} = \mathbf{M}_{flow} + \mathbf{M}_{ext} \end{cases} \tag{2.26}$$

where $\ddot{\mathbf{X}}_{CG}$ is the center of gravity's acceleration m is the mass, $\mathbf{g}$ is the gravitational acceleration, J is the inertia tensor, $\dot{\omega}$ is the angular acceleration, $\mathbf{F}_{flow}$ and $\mathbf{M}_{flow}$ are the resultant force and momentum of the fluid acting on the hull and $\mathbf{F}_{ext}$ and $\mathbf{M}_{ext}$ are eventual external forces and moments.

OpenFOAM has a built-in six degrees of freedom (6DOF) solver which integrates Newton's laws of motion using quaternions and septernions. Quaternions and septernions are mathematical objects that can store a translation or rotation and when called they return a tensor for rotation or a vector for translation (for further details on quaternions and rotation parametrization, [3] is suggested). An object of the class septernion consists of a vector and a quaternion. The resulting ordinary differential equations are integrated with respect to time with a Runge-Kutta scheme. Constraints (fixed rotations or fixed translations, for example), if imposed by the user, are applied iteratively. This means that at each time-step the necessary constraint force is calculated iteratively, until a given tolerance on the blocked degree of freedom is reached. Usually a couple of iterations are sufficient to reach convergence. This type of scheme permits to apply restraints too, thus it is possible to also model springs and dampers of an eventual experimental set-up.

The following algorithm describes the extraction of the forces and the implementation of the mesh moving scheme.

1. Extract forces and moments from pressure.
2. Extract forces and moments from viscous effects.
3. Apply estimated constraints and/or restraints forces.
4. Solve ODE, get linear and angular velocity (V and Ω respectively).
5. Evaluate new boat position and orientation.
6. If movement along constraint direction is less then a given a tolerance, go to new time-step, or else go back to point 3.

Chapter 3

Numerical Discretization and Solution of Free-Surface Flow Equations

The purpose of a discretization procedure is to transform one or several differential equations into a corresponding system of algebraic equations. The solution of this algebraic system will produce a set of values which will correspond to the solution of the differential equation (or of the system of differential equations) in pre-determined points in space and in pre-determined moments in time. The discretization procedure can be divided into two stages: time and space domain discretization and equation discretization. The domain discretization produces a discrete description of the computational domain, splitting it in a finite number of Control Volumes (CV) or cells over which equations will be numerically solved (in our case the center of CV). Equation discretization transforms the terms governing differential equations into algebraic expressions.

3.1 The Finite Volume Method

OpenFOAM adopts the Finite Volume Method (FVM) technique as space discretization. This method has the following properties:

- FVM is based on the discretization of the integral form of the governing equations over each Control Volume. Basic quantities are therefore conserved over every cell.

- Equations are solved in a Cartesian coordinate system on the mesh. FVM is applicable either for steady or transient problems.

- Meshes can be arbitrarily unstructured, therefore CVs will be general polyhedrals with an arbitrary number of faces. A co-located (or non-staggered) variable arrangement is adopted, therefore the same CV are shared by all dependent variables.

- Systems of partial differential equations are treated in a segregated way. They are thus solved one at a time and inter-equation coupling is treated in the explicit manner. Non-linear equations are linearized before discretization and non-linear terms are lagged.

3.1.1 Solution Domain Discretization

As aforementioned, the discretization of the computational domain is done through the construction of a mesh on which governing equations are evaluated and solved. The Finite Volume method requires a subdivision of the physical domain into a finite number of Control Volumes, which do not overlap and completely fill the domain. The centroid of each volume (point P), where the solution is sought, is such that:

$$\int_{V_p} (\mathbf{x} - \mathbf{x}_P) dV = 0 \tag{3.1}$$

Each CV is bounded by a finite number of flat faces, each shared with one or more neighbouring CV (if it is not a boundary face). Each face is defined by a face area vector, $\mathbf{S}_f$, pointing outwards from the cell with the lower label, which is called the owner. The other cell sharing the same face (but with a higher label) is called the neighbour. The owner centroid is defined as P while the neighbour's centroid is N. The solution for each variable is computed in cell centroids. In order to avoid spurious pressure modes due to the co-located grid, a technique equivalent to the Rhie-Chow interpolation is adopted. Therefore fluxes through each cell face are evaluated interpolating velocity on the values assumed in cell centers belonging to cells sharing the same face (i.e. P and N in figure 3.1).

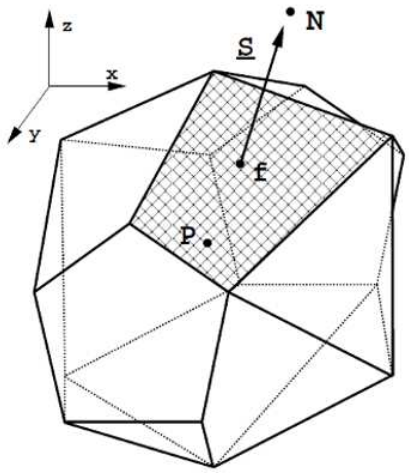


Figure 3.1: Control Volume.

3.1.2 Equation Discretization

The discretization of a general transport equation is shown in this section. The same discretization procedure for Navier-Stokes equations is straightforward, except for the non-linear term and the pressure-velocity coupling, that will be discussed separately. The standard form of the transport equation for a scalar property ϕ is:

$$\frac{\partial \rho\phi}{\partial t} + \nabla \cdot (\rho \mathbf{U}\phi) - \nabla \cdot (\rho\Gamma_\phi \nabla\phi) = S_\phi(\phi) \tag{3.2}$$

where ρ is the density, $\mathbf{U}$ is the velocity, Γ_ϕ is the fluid's diffusivity and $S_\phi(\phi)$ is the source term.

The accuracy of the discretization depends on the assumed variation of the function $\phi = \phi(\mathbf{x}, t)$ in space and time around point P. Since a second-order accurate model is preferable, this variation must be linear in space and time. Therefore, if $\phi_P = \phi(\mathbf{x}_P)$ and $\phi^t = \phi(t)$, it is assumed that:

$$\begin{aligned} \phi(\mathbf{x}) &= \phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla\phi)_P \\ \phi(t + \Delta t) &= \phi^t + \Delta t \left(\frac{\partial\phi}{\partial t}\right)^t \end{aligned} \tag{3.3}$$

The finite volume method requires the governing equation(s) to be satis-

fied over the control volume V_P around the point P in the integral form:

$$\int_t^{t+\Delta t}\left[\frac{\partial}{\partial t}\int_{V_P}\rho\phi dV+\int_{V_P}\nabla\cdot(\rho\mathbf{U}\phi)dV-\int_{V_P}\nabla\cdot(\rho\Gamma_\phi\nabla\phi)dV\right]dt=\int_t^{t+\Delta t}\left(\int_{V_p}S_\phi(\phi)dV\right)dt \tag{3.4}$$

Each term of this equation can be now discretized singularly:

$$\begin{aligned}\int_{V_P}\phi(\mathbf{x})dV&=\int V_P[\phi_P+(\mathbf{x}-\mathbf{x}_P)\cdot(\nabla\phi)_P]dV\\&=\phi_P\int_{V_P}dV+\left[\int_{V_P}(\mathbf{x}-\mathbf{x}_P)dV\right]\cdot(\nabla\phi)_P\\&=\phi_P V_P\end{aligned} \tag{3.5}$$

The discretized convection term is:

$$\int_{V_P}\nabla\cdot(\rho\mathbf{U}\phi)dV=\sum_f\mathbf{S}_f\cdot(\rho\mathbf{U}\phi)_f=\sum_f\mathbf{S}_f\cdot(\rho\mathbf{U})_f\phi_f=\sum_f F\phi_f \tag{3.6}$$

where F is the mass flux through the face: $F=\mathbf{S}_f\cdot(\rho\mathbf{U})_f$

The discretized diffusion term is:

$$\int_{V_P}\nabla\cdot(\rho\Gamma_\phi\nabla\phi)dV=\sum_f\mathbf{S}_f\cdot(\rho\Gamma_\phi\nabla\phi)_f=\sum_f(\rho\Gamma_\phi)_f\mathbf{S}_f\cdot(\nabla\phi)_f \tag{3.7}$$

Before the actual discretization, the source term must be linearized:

$$S_\phi(\phi)=S_u+S_p\phi \tag{3.8}$$

Therefore:

$$\int_{V_p}S_\phi(\phi)dV=S_uV_P+S_pV_P\phi_p \tag{3.9}$$

The transport equation becomes:

$$\int_t^{t+\Delta t}\left[\left(\frac{\partial\rho\phi}{\partial t}\right)_P V_P+\sum_f F\phi_f-\sum_f(\rho\Gamma_\phi)_f\mathbf{S}_f\cdot(\nabla\phi)_f\right]dt=\int_t^{t+\Delta t}(S_uV_P+S_pV_P\phi_P)dt \tag{3.10}$$

which is also known as the semi-discrete form of the transport equation. It must be noted that, in order to discretize the previous terms, the Gauss theorem has been used to move from volume integral to surface integral.

Finally, time discretization is necessary, therefore, applying a Crank-Nicolson scheme (second order in time) we have:

$$\begin{aligned}\frac{\rho_P\phi_P^n - \rho_P\phi_P^o}{\Delta t}V_P + \frac{1}{2}\sum_f F\phi_f^n - \frac{1}{2}\sum_f(\rho\Gamma_\phi)_f\mathbf{S}_f(\nabla\phi)_f^n \\ + \frac{1}{2}\sum_f F\phi_f^o - \frac{1}{2}\sum_f(\rho\Gamma_\phi)_f\mathbf{S}_f(\nabla\phi)_f^o \\ = S_U V_P + \frac{1}{2}S_P V_P\phi_P^n + \frac{1}{2}S_P V_P\phi_P^o\end{aligned} \tag{3.11}$$

Therefore, for each control volume we have an algebraic equation:

$$a_P\phi_P^n + \sum_N a_N\phi_N^n = R_P \tag{3.12}$$

since the value of ϕ_P^n depends on the values of the neighbouring cells, it will result in a linear system:

$$[A][\phi] = [R] \tag{3.13}$$

The discretization of Navier-Stokes Equations presents two issues requiring particular attention: the non-linear term and the pressure-velocity coupling. The discretization of the non-linear term is quadratic in velocity in the algebraic equation, thus, the resulting system is non-linear. This term is therefore linearized and discretized as follows:

$$\begin{aligned}\int_{V_P}\nabla\cdot(\mathbf{U}\mathbf{U})dV &= \sum_f \mathbf{S}_f(\mathbf{U})_f(\mathbf{U})_f \\ &= \sum_f F(\mathbf{U})_f \\ &= a_P\mathbf{U}_P + \sum_N a_N\mathbf{U}_N\end{aligned} \tag{3.14}$$

Linearisation of the convection term implies that an existing velocity (flux) field that satisfies the continuity equation will be used to calculate a_P and a_N. The linearisation does not have any effects on steady-state calculations. When the steady-state is reached, the fact that a part of the non-linear term has been lagged is not significant. In transient flows two different approaches can be adopted: either to iterate over non-linear terms or to neglect the lagged non-linearity effects. Non-linear iteration can significantly increase the computational cost, but are necessary if the time-step is large and the accurate simulation of the transient solution is sought. On the other hand, if the time-step is small, the change between consecutive solutions will also

be small and it is therefore possible to lag the non-linearity without any significant effects. The semi-discretized form of momentum equation is:

$$a_P U_P = \mathbf{H}(\mathbf{U}) - \nabla p \tag{3.15}$$

where $\mathbf{H}(\mathbf{U})$ is given by:

$$\mathbf{H}(\mathbf{U}) = -\sum_N a_N \mathbf{U}_N + \frac{\mathbf{U}^o}{\Delta t} \tag{3.16}$$

Equation (3.15) is used to express $\mathbf{U}_P$:

$$\mathbf{U}_P = \frac{\mathbf{H}(\mathbf{U})}{a_P} - \frac{1}{a_P}\nabla p \tag{3.17}$$

therefore velocities on the cell faces are given by:

$$\mathbf{U}_f = \left(\frac{\mathbf{H}(\mathbf{U})}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla p)_f \tag{3.18}$$

Substituting equation (3.18) in the discretized form of continuity equation

$$\nabla \cdot \mathbf{U} = \sum_f \mathbf{S}_f \mathbf{U}_f = 0 \tag{3.19}$$

the pressure equation is obtained:

$$\nabla \cdot \left(\frac{1}{a_P}\nabla p\right) = \nabla \cdot \left(\frac{\mathbf{H}(\mathbf{U})}{a_P}\right) = \sum_f \mathbf{S}_f \left(\frac{\mathbf{H}(\mathbf{U})}{a_p}\right)_f \tag{3.20}$$

The laplacian on the l.h.s. can be discretized in the standard way and the final form of the incompressible Navier-Stokes system is:

$$a_P \mathbf{U}_P = \mathbf{H}(\mathbf{U}) - \sum_f \mathbf{S}_f (p)_f \tag{3.21}$$

$$\sum_f \mathbf{S}_f \left[\left(\frac{1}{a_P}\right)_f (\nabla p)_f\right] = \sum_f \mathbf{S}_f \left(\frac{\mathbf{H}(\mathbf{U})}{a_P}\right)_f \tag{3.22}$$

Face fluxes F are expressed using equation (3.18):

$$F = \mathbf{S}_f \mathbf{U}_f = \mathbf{S}_f \left[\left(\frac{\mathbf{H}(\mathbf{U})}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f - (\nabla p)_f\right] \tag{3.23}$$

and are conservative when the pressure equation (equation (3.20)) is satisfied.

Volume Of Fluid (VOF) Equation Discretisation

Particular attention has to be given to the discretisation of the equation for α (equation (2.13)), i.e. the volume fraction equation. As previously stated, the interface has to be as sharp as possible, therefore compression schemes for the interface have to be adopted. This is obtained by introducing an extra artificial compression in equation (2.13):

$$\frac{\partial \alpha}{\partial t} + \mathbf{U} \cdot \nabla \alpha + \nabla \cdot (\mathbf{U}_r \alpha (1 - \alpha)) = 0 \tag{3.24}$$

where $\mathbf{U}_r$ is a velocity field that is suitable to compress the interface. The advantage of equation (3.24) is that its solution (α) can be bounded between zero and one. Using the discretisation schemes previously adopted and treating time advancement explicitly, the MULES (Multidimensional Universal Limiter for Explicit Solution) scheme is obtained:

$$\frac{\alpha^{n+1} - \alpha^n}{\Delta t} V_P + \sum_f [\alpha_f^n \phi_f^n + \alpha_f^n (1 - \alpha_f^n) \phi_{rf}^n] = 0 \tag{3.25}$$

where ϕ_f^n is the face flux calculated from the pressure-velocity coupling and $\phi_{rf}^n = \mathbf{U}_r^n \cdot S_f$ is the 'compression flux', given by:

$$\phi_{rf}^n = n_f \min \left[C_\alpha \frac{|\phi|}{|\mathbf{S}_f|}, \max \left(\frac{|\phi|}{|\mathbf{S}_f|} \right) \right] \tag{3.26}$$

C_α is an adjustment constant. n_f, the face unit normal flux, yields:

$$n_f = \frac{(\nabla \alpha)}{|(\nabla \alpha)_f + \delta_n|} \cdot \mathbf{S}_f \tag{3.27}$$

where δ_n is a stabilization factor. Further details can be found in [5], [6], [7] and [8].

3.2 Solution of Navier-Stokes Equations

A procedure for the solution of the Navier-stokes equations, together with an additional transport equation (e.g. turbulence model or volume-fraction equation) will now be described. In order to do this, an iterative scheme for the solution of the pressure-velocity coupling will be shown.

3.2.1 PISO Algorithm

To solve unsteady Navier-Stokes equations, an iterative procedure for solving equations for velocity and pressure is adopted: the Pressure-Implicit Split-Operator (PISO) scheme. The PISO algorithm can be described as follows:

- Momentum Predictor Step: the pressure gradient is not known, therefore pressure field from the previous time-step is used in order to solve momentum equation. An approximation of the velocity field is obtained with equation (3.21).
- Pressure Solution Step: predicted velocities are used to assemble the $\mathbf{H}(\mathbf{U})$ operator in order to obtain an estimate of pressure from the pressure equation.
- Explicit Velocity correction: a set of conservative fluxes are computed (equation (3.23)) using the new pressure field. Velocity field is explicitly corrected as a consequence of the new pressure distribution using equation (3.17).

A closer look at equation (3.17) shows that the correction depends on two terms: a pressure gradient change ($\frac{1}{a_P}\nabla p$) and the transport influence of corrections of neighbor velocities ($\frac{\mathbf{H}(\mathbf{U})}{a_P}$). Correcting explicitly the velocity means neglecting the latter part, so the error is assumed to be dependent on pressure only. It is therefore necessary to also correct the $\mathbf{H}(\mathbf{U})$ term, by reformulating the pressure equation and repeating the procedure. Thus, the PISO algorithm con be considered as an implicit momentum predictor followed by a series of pressure solutions and explicit velocity corrections. This loop is repeated until tolerance is reached. It should be noticed that the $\mathbf{H}(\mathbf{U})$ coefficients should be re-calculated after each pressure solution. This is not actually done, and coefficients are kept constant until a new Moment Predictor step is computed. This is because the non-linear coupling is considered much less important than the pressure-velocity coupling, consistently with the linearisation of the momentum equation.

3.2.2 Solution Procedure for the Navier-Stokes System

Finally a transient solution procedure for Navier-Stokes Equations will be shown. It must be noticed that all inter-equation couplings apart from the pressure-velocity system are lagged. If a closer coupling between some of the equations is needed, equations should be included in the PISO loop. A transient solution procedure for incompressible turbulent flows can be summarized as follows:

1. Set up the initial conditions for all field values.
2. Start the calculation of the new time-step values.
3. Assemble and solve the momentum predictor equation with the available face fluxes.
4. Go through the PISO loop until tolerance for the pressure-velocity system is reached. At this stage, pressure and velocity fields for the current time-step are obtained, as well as the new set of conservative fluxes.
5. Using the conservative fluxes, solve all other equations in the system (turbulence quantities, volume fraction). If the flow is turbulent, calculate the effective viscosity from the turbulence variables. If a VOF method is used for interface capturing, equation for volume fraction is solved.
6. If the final time is not reached, return to step 2.

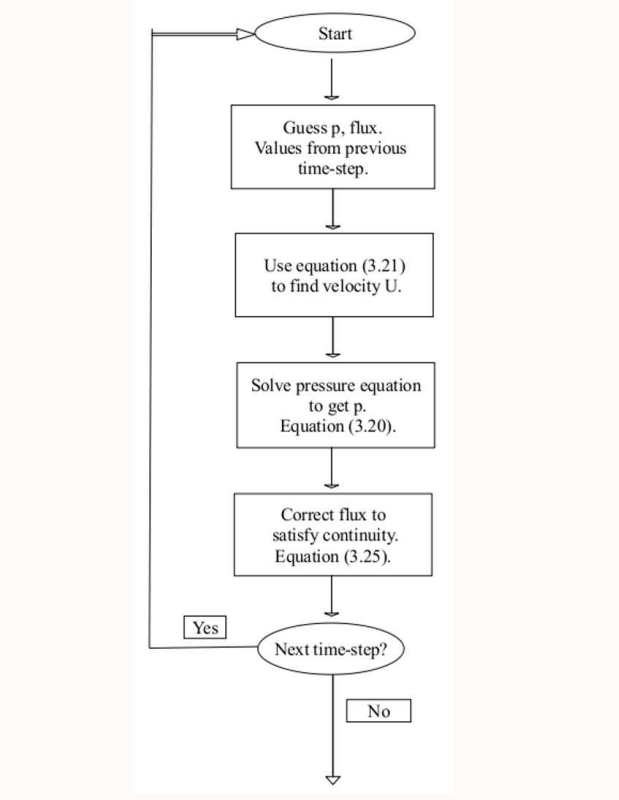


Figure 3.2: PISO loop.

Chapter 4

Mesh Generation

Grid resolution and grid quality play a crucial role for the accuracy of viscous flow calculations. The results of different types of meshes found in literature show that most of the best grid quality is given by block-structured meshes. Despite the good control that structured meshes can give on local refinement, it takes a large amount of man-hours to make such a mesh on a complex geometry. Furthermore, commercial grid generators have to be used, since no free or open-source structured meshers for complex geometries are available. Unstructured meshers are typically easier to use, they require less man-hours and can be easily integrated in an automatized analysis process. However, the risk of generating meshes with a higher number of cells compared to a structured mesh of equivalent quality is present. Different free or open-source unstructured meshers are available, and in this work *snappyHexMesh*, the mesh generation utility supplied with OpenFOAM was adopted. The use of *snappyHexMesh* on its own wasn't enough to obtain a correct geometry reproduction and a satisfying refinement control. *SnapEdge*, a utility not supplied in the OpenFOAM release, developed by Niklas Nordin [9], was ad-hoc modified and used to obtain the desired accuracy in geometry reproduction. Some coding was needed in order to perform local refinement without exceeding the number of cells.

4.1 Peculiarities of Meshes for Naval Simulations

The most important issues that must be taken into account when generating a mesh are: hull refinement, dimension and grading of boundary layer cells, the choice of control volumes shapes and free-surface refinement. Sufficient refinement of the grid on the hull surface is needed in order to have an

accurate evaluation of pressure and viscous forces. The choice of cell dimension and grading near the hull is crucial to correctly compute the boundary layer, especially when using turbulence models with wall-functions. It is desirable that control volumes are chosen as hexahedrals as this will produce a smoother free-surface than the one given by tetrahedrals, and anisotropic refinement is easier to perform in the former approach. Since a front-capturing method is employed, an adequate free-surface refinement is needed in order to correctly capture the shape of the wave. This is extremely important because the pressure forces acting on hulls strongly depend on the wave pattern.

4.2 Meshing Utilities

The *snappyHexMesh* is a mesh generation utility supplied with OpenFOAM. This utility automatically generates three-dimensional meshes containing hexahedra and split-hexahedra cells from triangulated surface geometries in Stereolithography (STL) format. The mesh approximately conforms to the surface by iteratively refining a starting mesh and morphing the resulting split-hex mesh to the surface. An optional phase shrinks back the resulting mesh and inserts cell layers. The specification of mesh refinement level is flexible and the surface handling is robust with a pre-specified final mesh quality.

The utility *snappyHexMesh* has one major problem: it is unable to identify geometry edges and therefore to correctly reproduce them. Sharp edges are a major feature in naval architecture (bow, stern, spray rails for example), therefore they must be precisely represented. The utility *snapEdge*, not included in the OpenFOAM release, is used to snap cell vertex points to geometry edges. This utility moves the nearest points to the geometry edges on the edges themselves. This is done iteratively in order to maintain grid quality. A slight change was brought to the original source of snapEdge, in order to better identify geometry edges from the STL file.

Free-surface refinement is done by running a manipulation utility available within OpenFOAM: *refineMesh*. This utility splits cells belonging to a user-defined set. Splitting can be done both isotropically or along user-defined directions. Since our main interest is to capture the correct free-surface elevation, the refinement was done only along the direction normal to the free-surface. A simple utility called *createSet* was implemented in order to easily select the refinement zone near the free-surface, leaving out the cells previously refined by *snappyHexMesh*. This process allows to produce a satisfactory mesh refinement along defined directions and within certain zones of the computational domain, without exceeding in the number of cells.

4.2.1 Meshing Process

The meshing process can be summarized in the following steps:

1. Definition of the computational domain, base level blocking and mesh density, creating a background hex mesh using *blockMesh*, a built-in utility in OpenFOAM, to create structured meshes in very simple domains. (Figure 4.1).

2. The mesh is iteratively refined, in user-defined zones (typically a box around the hull), by a pre-defined number of isotopic cell splitting. Cells not belonging to the computational domain (i.e. cells inside the hull) are removed. This step is called castellatedMesh. (Figure 4.2).

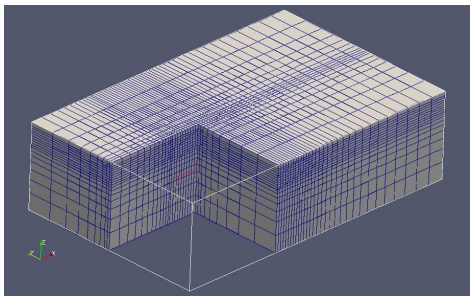
3. The cell vertex points are moved onto surface geometry to remove the jagged castellated surface from the mesh. This process is called snappy and is performed iteratively in order to respect grid quality parameters when moving cell vertex points. (Figure 4.3).

4. Sharp edges are not identified by the *snappyHexMesh* utility, therefore cell vertex are moved on the geometry edges by the modified version of *snapEdge*. (Figure 4.4).

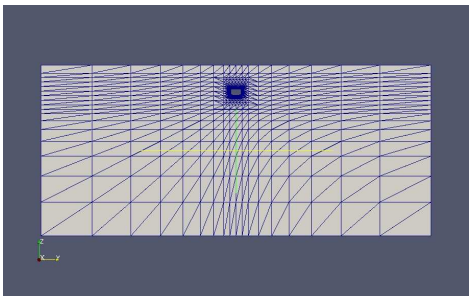
5. Free-surface refinement is generally done only in the direction normal to the free-surface, by running the *refineMesh* utility on the only cells belonging to the free-surface zone. The application createSet was used in order to make this selection of cells. (Figure 4.5).

6. Hexahedral cell layers are introduced, aligned with the boundary surface, in order to obtain the desired cell dimension and grading where the boundary layers will be. This step is called addLayers. (Figure 4.6).

The extreme importance of snapEdge step (n° 4) is highlighted in figure 4.7. Figure 4.8 shows the final mesh.

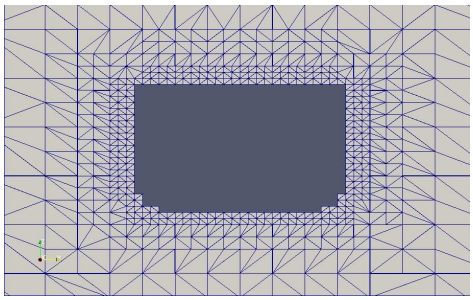


(a) Initial Background Mesh.

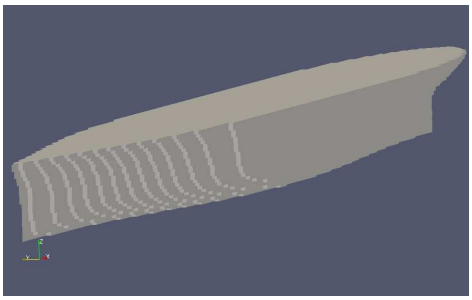


(b) CastellatedMesh Step. Section.

Figure 4.1: Initial Background Mesh and CastellatedMesh Step, Section.

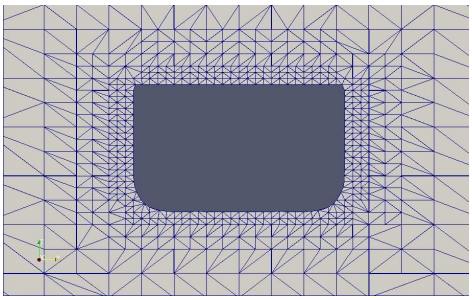


(a) Section, Zoom.

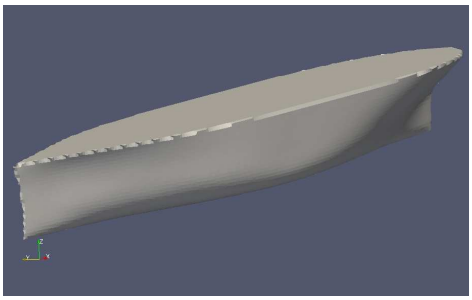


(b) Hull.

Figure 4.2: CastellatedMesh Step.

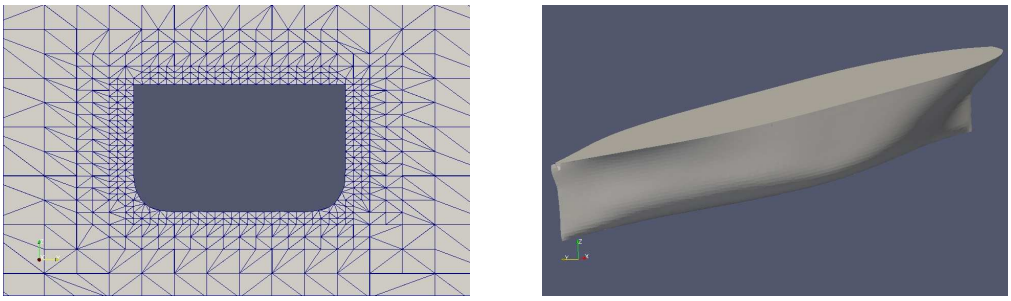


(a) Section, Zoom.



(b) Hull.

Figure 4.3: Snappy Step.



(a) Section, Zoom. (b) Hull.

Figure 4.4: SnapEdge Step.

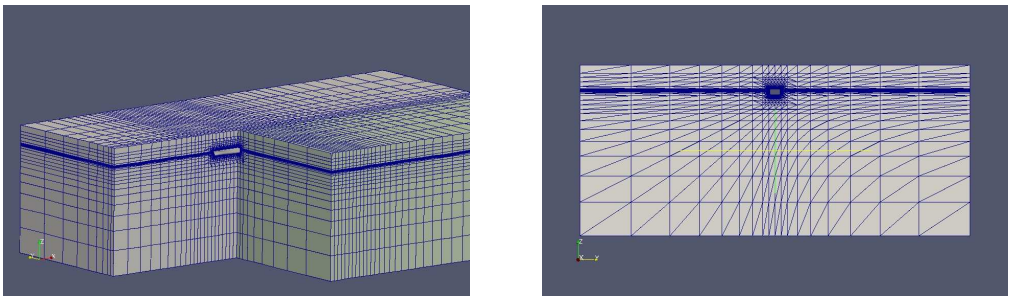


Figure 4.5: RefineMesh Step.

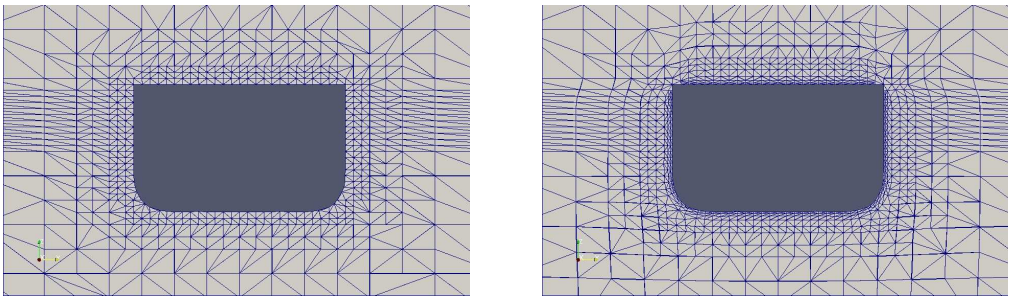
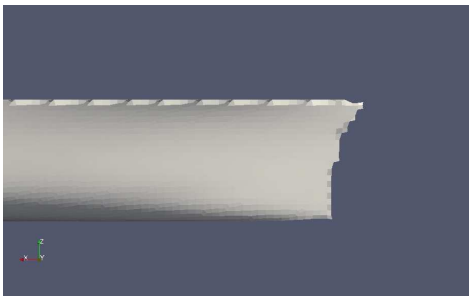
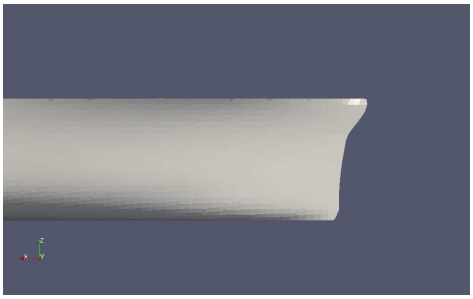


Figure 4.6: RefineMesh Step and addLayers Steps. Section, Zoom.

(a) CastellatedMesh Step.



(b) Snappy Step.



(c) SnapEdge Step.

Figure 4.7: Different Meshing Steps. Bow Detail.

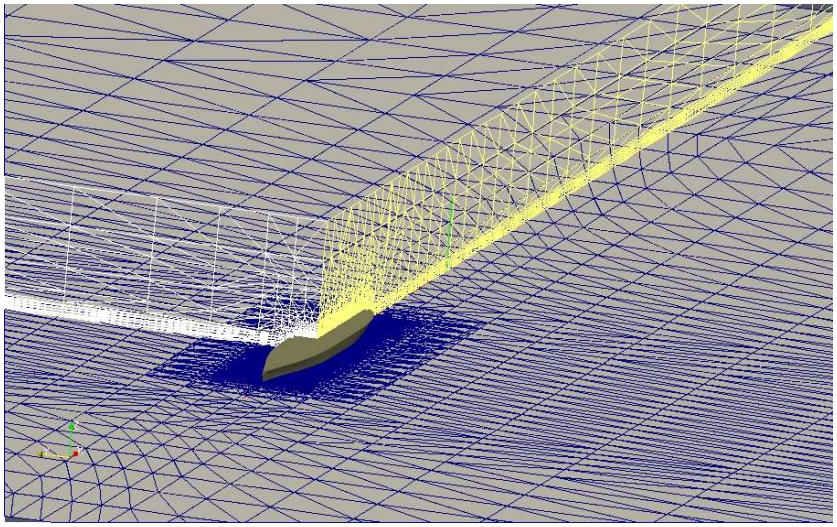


Figure 4.8: Final Mesh.

4.2.2 Structured Meshes

In Chapter 5 results from unstructured meshes made with the meshing process described in subsection 4.2.1 are compared with results given by structured grids made with a commercial mesher, Ansys ICEM CFD. As previously stated, block structured meshes allow for a good quality control and a good mesh refinement without exceeding in number of cells. Structured grids are made by dividing the computational domain in hexahedral blocks. On each block the number and distribution of CVs along the three principal directions is imposed. In order to accurately refine the mesh near the hull and in the free-surface zone, without propagating the refinements throughout the whole domain, a particular type of blocking was adopted: O-grid blocking (figure 4.10).

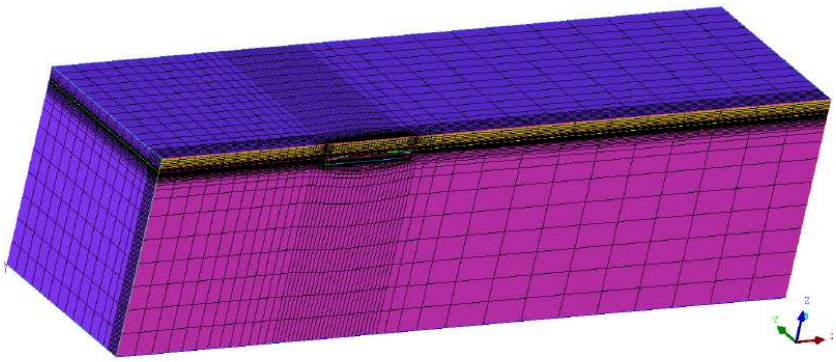
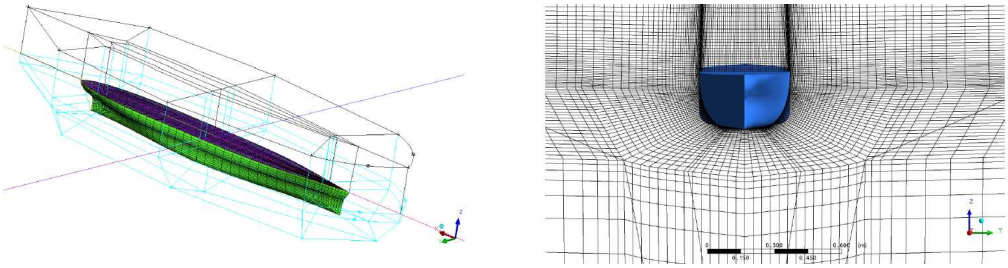


Figure 4.9: Example of an ICEM Mesh.



(a) O-Grid blocking of an ICEM Mesh.

(b) Section of an O-Grid structured ICEM Mesh.

Figure 4.10: O-Grid Structured Mesh.

4.3 Grids Used

In order to perform space and time refinement convergence studies on the steady solver (interFoam), three different meshes were generated. They were all computed starting from the same background mesh (last column in table 4.1) imposing different hull and free surface refinement levels. The boundary layers were kept the same for all meshes. Dynamic simulations (interDyMFoam solver) were done using the Coarse mesh. A fourth mesh, refined in the vertical and stream wise direction equally, was necessary to compute simulations with waves. In tables 4.1 and 4.3 features of meshes made in snappyHexMesh and ICEM are summarized. It must be noticed that the domain is symmetric and only symmetric problems have been solved. Therefore it is sufficient to divide the domain with a plane corresponding to the symmetry-plane and to impose symmetry-plane type of boundary conditions in order to reduce the number of CVs, i.e. unknowns. This was possible only in ICEM-made meshes, and the number of CV given in table 4.3 are referred to the half domain. Unfortunately, when making unstructured meshes with *snappyHexMesh*, the division of the computational domain was impossible. Therefore, simulations with snappyHexMesh-made meshes were computed on the entire domain.

Grid	CVs	Ref. Level	n° BL	BL Grad.	Min Y^+	Max Y^+	Ave. Y^+	x · y · z
Coarse	144.887	4	3	1.5	2.8	120.5	24.0	53 · 28 · 17
Medium	666.184	5	3	1.3	1.6	64.0	16.7	53 · 28 · 17
Fine	2.752.053	6	1	1	1.0	35.4	27.1	53 · 28 · 17
Waves	769.092	4	3	1.5	3.3	71.6	17.1	108 · 28 · 17

Table 4.1: snappyHexMesh Meshes' Characteristics

Grid	CVs	MinY^+	MaxY^+	Ave.Y^+
Coarse	70.749	2.1	77.3	27.8
Medium	186.974	1.6	78.5	17.7
Fine	652.670	0.7	91.8	21.7
Waves	462.686	2.6	67.3	27.8

Table 4.2: ICEM Meshes' Characteristics

Chapter 5

Numerical Simulations

The purpose of this thesis is to show that an ¨Open-Source approach¨to Computational Fluid Dynamics for naval engineering problems is possible and competitive in terms of results' accuracy, compared to licensed softwares. In order to show this, numerical simulations on the Series 60 hull (figure 5.1) were carried out. The Series 60 hull was chosen as a benchmark because a large number of numerical and experimental results are available in literature. In this chapter numerical results from OpenFOAM simulations are compared to experimental towing-tank results ([10],[11],[12] and [13]) and to commercial software (ANSYS CFX[1]) results ([14], [15] and [16]). Furthermore, a comparison between OpenFOAM simulations with unstructured meshes (made with *snappyHexMesh*) and structured meshes (made with ANSYS ICEM CFD) is carried out. In Section 5.2 wave patterns and wave lines from different mesh refinements are compared to experimental ones. In Subsection 5.2.2 time refinement convergence and grid refinement convergence are shown in subsection 5.2.1. In Subsection 5.2.3 results from static[2] simulations (fixed hull) in presence of waves are given. In Section 5.3 results form dynamic simulations (free trim only and free sink and trim) are compared to experimental ones.

[1] CFX simulations were computed with ANSYS ICEM made meshes.

[2] It must be noticed that simulations with fixed hull are called static despite the fact that the solver (*interFoam*) is transient, while simulations done with the boat free to move are called dynamic. The solver for dynamic simulations (*interDyMFoam*) is obviously transient.

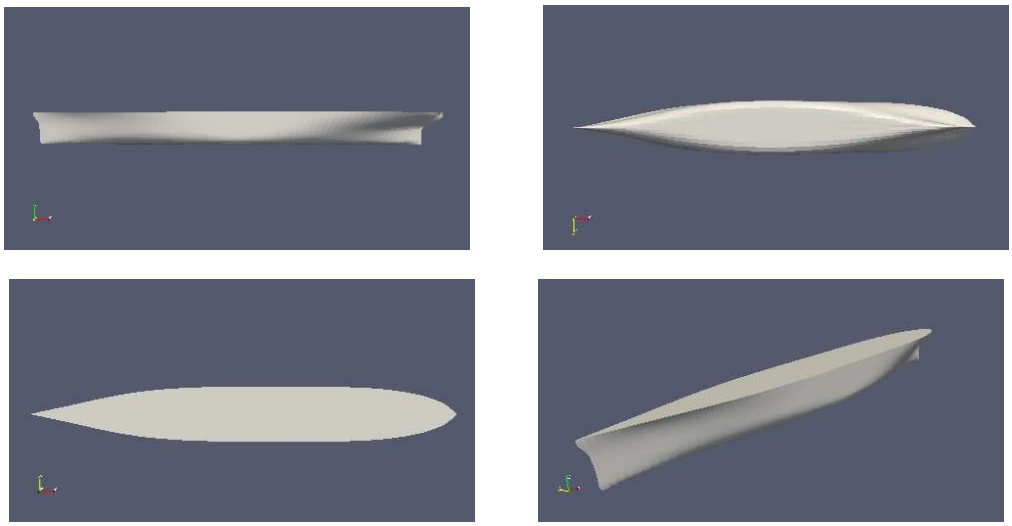


Figure 5.1: Series 60 Hull

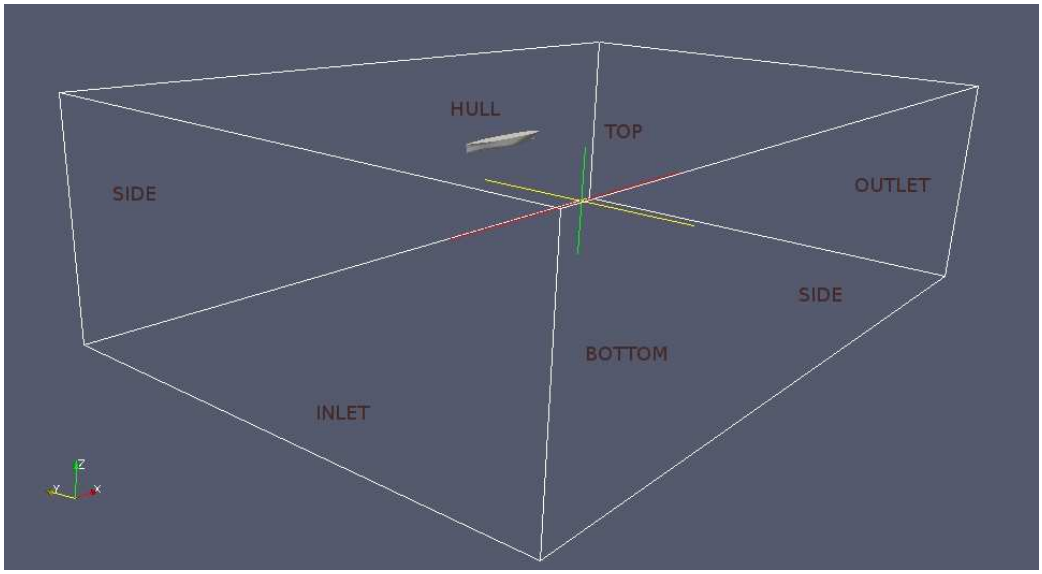


Figure 5.2: Computational Domain. Boundaries.

5.1 Numerical Set-Up

As Described in Section 4.3, the computational domain is a prism that is sufficiently large to avoid any type of wave reflection. As described in figure 5.2, the domain is delimited by an INFLOW and OUTFLOW section, the TOP, BOTTOM, the two SIDES and the HULL, forming the boundary.

Given the symmetry of the problem, simulations on half domain were carried out by imposing symmetry-plane type of conditions. This was done only for the ICEM-made meshes, while simulations on the entire domain when using the snappy-made meshes.

Initial and Boundary Conditions

If the computational domain is sufficiently large, on the TOP, BOTTOM, SIDE and INLET the flow can be considered undisturbed: the imposed velocity is therefore aligned with the free-stream, with a magnitude equal to the asymptotic conditions (*fixedValue* boundary condition). On the OUTLET the flow can not be considered undisturbed, and a condition of *zeroGradient* is imposed. The *zeroGradient* boundary conditions impose the normal derivative to be zero on the boundary:

$$\left.\frac{\partial \mathbf{U}(\mathbf{x},t)}{\partial n}\right|_{Boundary} = 0. \tag{5.1}$$

On the HULL, the fluid velocity is imposed to be equal to the boat velocity, therefore $U = 0$ for static simulations (fixed hull) or $U = U_{boat}$ for dynamic simulations. Obviously, U_{boat} is obtained by solving the 6DOF at each time-step. Pressure is given a fixed value on the TOP, while *zeroGradient* boundary conditions are imposed on the rest. The volume fraction is imposed on the INLET and SIDE with a value of zero if the face cell is over the free-surface and one if under:

$$\alpha(\mathbf{x},t)|_{INLET,SIDE} = \begin{cases} 0 & if \quad z > z_{fs} \\ 1 & if \quad z < z_{fs} \end{cases} \tag{5.2}$$

on the rest of the domain the boundary condition for the volume fraction is *zeroGradient*. As previously stated, in simulations with ICEM-generated meshes, the computational domain was split and symmetry-plane type of boundary conditions were imposed. Symmetry-plane conditions impose the normal component of the velocity and the normal derivative of the tangential component of the velocity to be null:

$$\mathbf{U}_n(\mathbf{x},t)|_{symmetry-plane} = 0, \qquad \left.\frac{\partial \mathbf{U}_\tau(\mathbf{x},t)}{\partial n}\right|_{symmetry-plane} = 0 \tag{5.3}$$

To simulate the turbulence effects on the mean flow, the $\kappa-\omega SST$ turbulence model was used and standard OpenFOAM wall-functions were adopted to impose velocity distribution in the boundary layer (HULL). A corresponding value to 1% of turbulence intensity was imposed to the turbulent variables

on the INFLOW. The *zeroGradient* condition was imposed on the rest of the domain.

To start the simulation, the INLET boundary conditions are imposed over the entire domain. This means that at $T = 0$ the velocity is not aligned with the hull and strong velocity gradients will appear after the initialization. This shock is fairly managed by the numerical method, therefore an acceleration of the flow from rest is not necessary.

Numerical Schemes and Linear System Solvers

The chosen discretization practice is the standard Gaussian finite volume integration. Gaussian integration is based on summing values on cell faces, which must be interpolated from cell centers. The chosen interpolation schemes are second order accurate, and these are: SFCD (Self-Filtered Central Differencing, a blended scheme, involving a combination of higher-order differencing and upwind differencing scheme.) for the divergence terms, linear (central differencing) for gradients and laplacians. Temporal discretization was done by means of Euler implicit scheme.

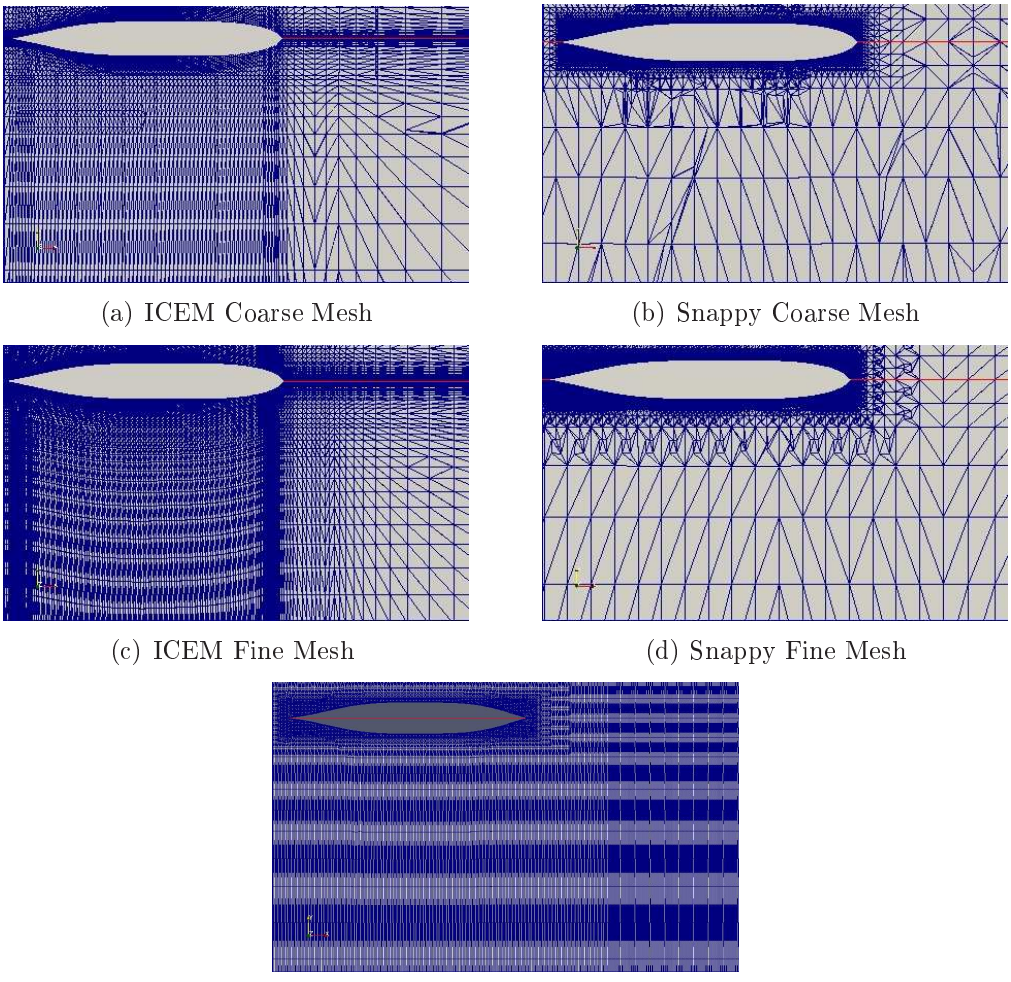
The resulting linear systems were solved with the Preconditioned Bi-Conjugate Gradient (PBiCG) solver, using the Diagonal Incomplete-LU (DILU) preconditioner, except for the pressure, which was solved with a Preconditioned Conjugate Gradient (PCG) using the Diagonal incomplete-Cholesky (DIC) preconditioner.

5.2 Static Simulations

Numerical simulations of the flow around the Series 60 hull were computed using the six meshes described in tables 4.1 and 4.3. The time-step was chosen as variable, so that the maximum Courant number at each iteration resulted less than 0.9. This was done in order to reduce the computational time, by taking the biggest time-steps possible, and so avoiding the risk of instabilities due to the integration with a too large time-step.

In figure 5.4 wave patterns at $Fr = 0.316$ are compared to experimental ones (upper half of each figure). It is evident how the computed wave pattern becomes similar to the experimental one for finer meshes. Computed wave patterns from ICEM-made meshes are much more regular and converge much faster to the experimental results with respect to the Snappy-made meshes. Convergence of the wave pattern computed with the Snappy-made meshes to the experimental results is not so evident because the first type of meshes are locally refined near the boat and at the interface zone in the vertical direction only (normal to the free-surface), and not in stream wise and transverse

direction. This is shown in figure 5.3. Indeed, improvements are obtained if refinement is also done in the stream wise direction, like in the Snappy-made Waves mesh: this mesh is equally refined in the stream wise and vertical directions and has the same local refinement as the Coarse one.



(a) ICEM Coarse Mesh

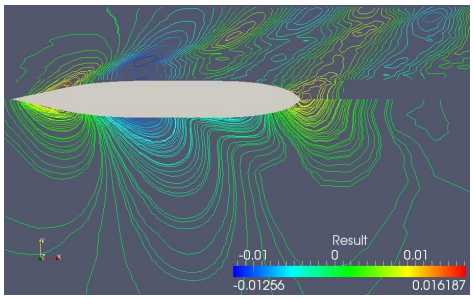
(b) Snappy Coarse Mesh

(c) ICEM Fine Mesh

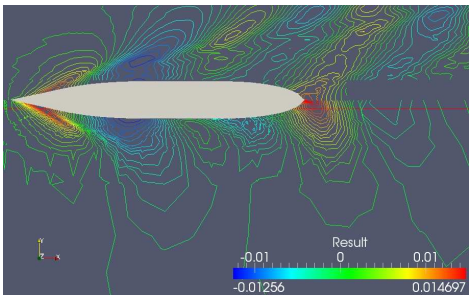
(d) Snappy Fine Mesh

(e) Snappy Waves Mesh

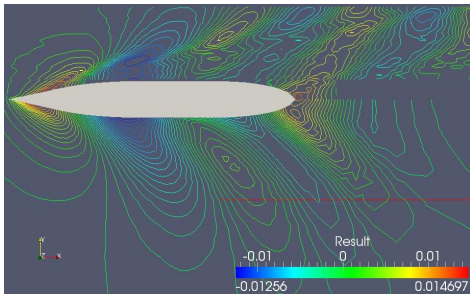
Figure 5.3: Interface Zone Refinements



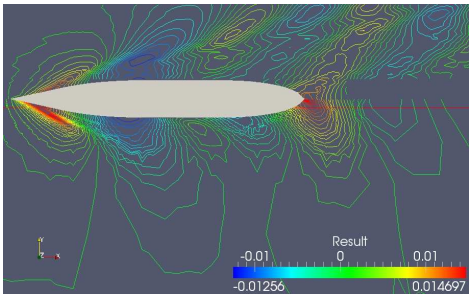
(a) ICEM Coarse Mesh Vs Experimental



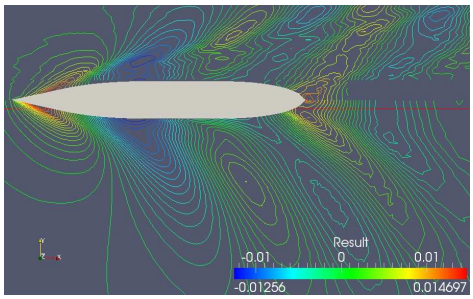
(b) Snappy Coarse Mesh Vs Experimental



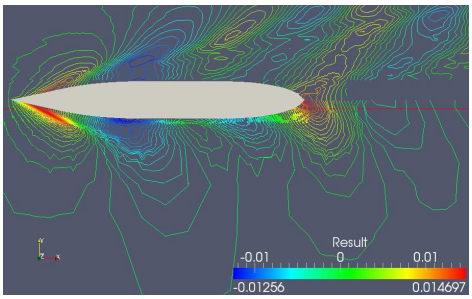
(c) ICEM Medium Mesh Vs Experimental



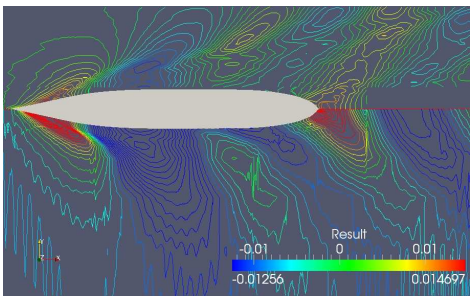
(d) Snappy Medium Mesh Vs Experimental



(e) ICEM Fine Mesh Vs Experimental



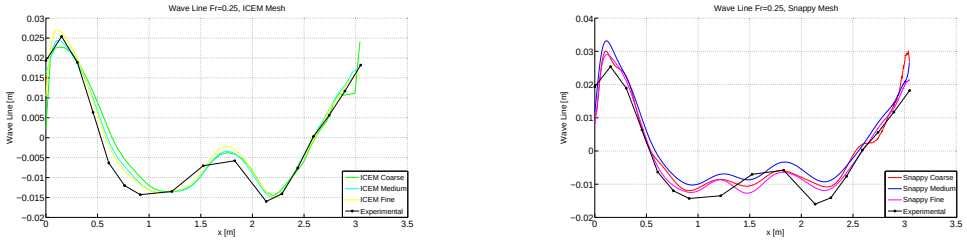
(f) Snappy Fine Mesh Vs Experimental



(g) Snappy Waves Mesh Vs Experimental

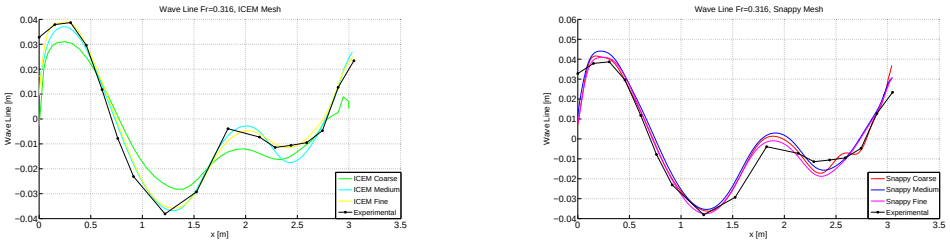
Figure 5.4: Wave Patterns. Numerical Vs Experimental. Fr=0.316.

In figures 5.5 and 5.6 wave lines at $Fr = 0.25$ and $Fr = 0.316$ respectively, are shown. Convergence of computed wave lines to experimental ones is evident also for the Snappy-made meshes. This is because, as aforementioned, mesh refinement near the hull is isotropic and not only in the vertical direction, like in the interface zone. Figure 5.7 shows wave lines for the finer meshes, ICEM and Snappy made.



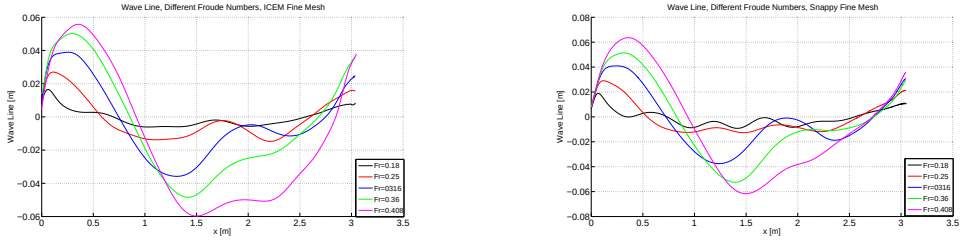
(a) ICEM Meshes (b) Snappy Meshes

Figure 5.5: Wave Lines, Fr=0.25.



(a) ICEM Meshes (b) Snappy Meshes

Figure 5.6: Wavel Lines, Fr=0.316.



(a) ICEM Fine Mesh (b) Snappy Fine Mesh

Figure 5.7: Wave Lines, Different Froude Numbers, Fine Meshes.

5.2.1 Effects of Grid Refinement on Resistance

The ultimate goal for these types of analyses is to determine the hull's resistance, this is why it is important to determine the level of accuracy depending on grid refinement. Figures 5.8 to 5.13, friction coefficient (CD_v), pressure drag coefficient (CD_p) and total drag coefficient (CD) values in function of Froude number are shown for each mesh [3]. Results given by the ICEM-made meshes are quite concerning. No convergence is highlighted, and the results given by the Coarse mesh are nonsense. Such a drag reduction for higher Froude numbers is definitely not physical. This could be caused by the fact that mesh quality constraints imposed by OpenFOAM are much more restrictive with respect to ANSYS CFX ones. Although OpenFOAM mesh quality checks were all passed, a high number of non-orthogonal faces was found. This could have caused the low quality of the results given by these meshes. Conversely, results given by Snappy-made meshes show an excellent convergence and are far closer to the experimental data compared to CFX simulations (results given in [15] and [16]). Figure 5.14 shows a comparison of results from ICEM meshes and Snappy meshes and the error with respect to the experimental data. Error on friction coefficient is particularly low, less then 2.5% for the medium and fine mesh. The drag pressure coefficient error is a bit higher, but in all cases lower than 10%. Total drag coefficient error is less then 6% an in particular less then 1% for low Froude numbers.

[3] It must be noticed that the split in CD_v and CD_p of the experimental CD is done using empirical formulas. Therefore, CD_v and CD_p values are to be considered indicative.

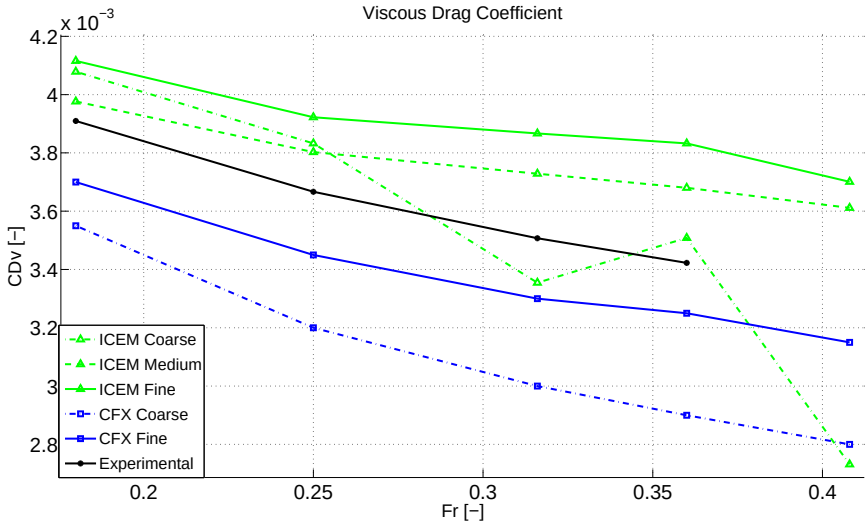


Figure 5.8: Viscous Drag Coefficient (CD_v). ICEM Meshes.

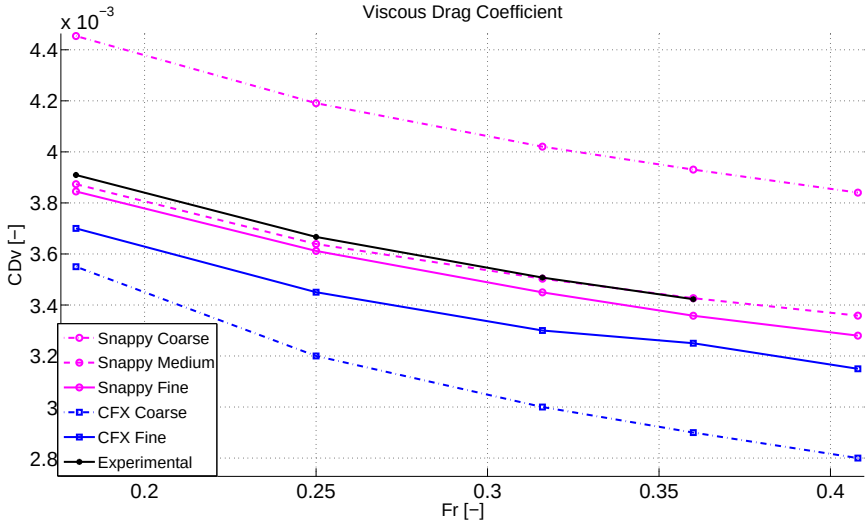


Figure 5.9: Viscous Drag Coefficient (CD_v). Snappy Meshes.

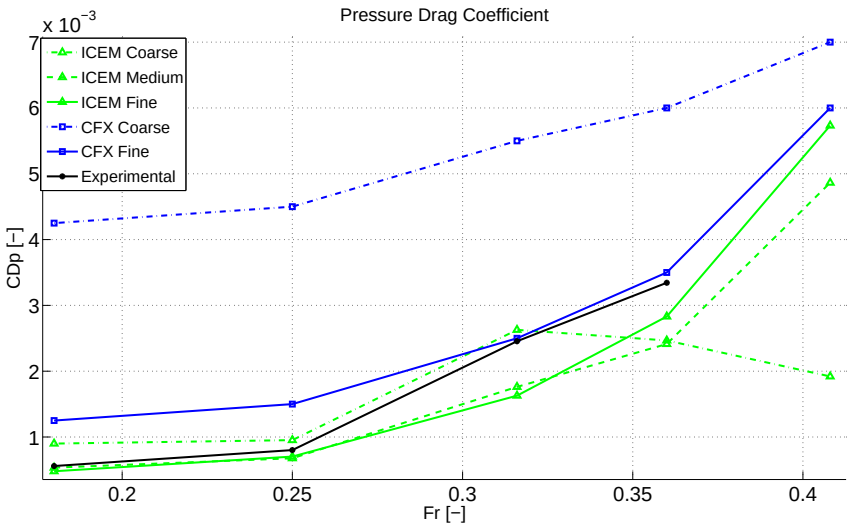


Figure 5.10: Pressure Drag Coefficient (CD_p). ICEM Meshes.

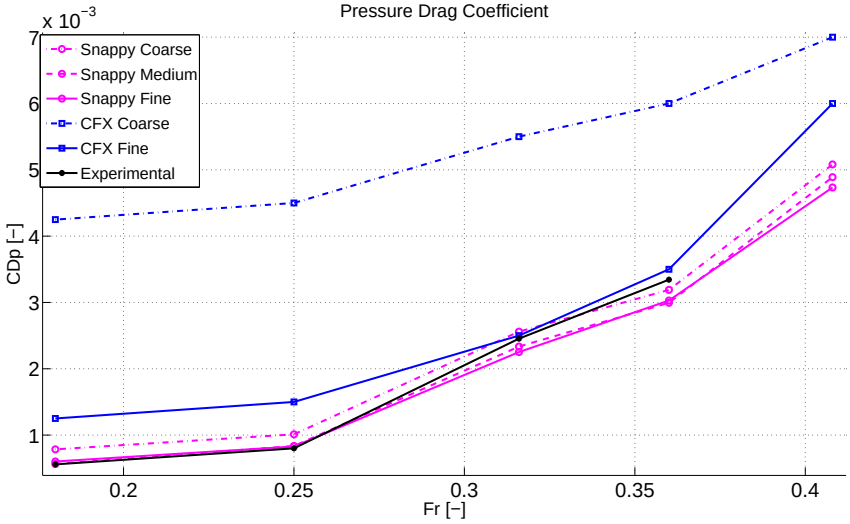


Figure 5.11: Pressure Drag Coefficient (CD_p). Snappy Meshes.

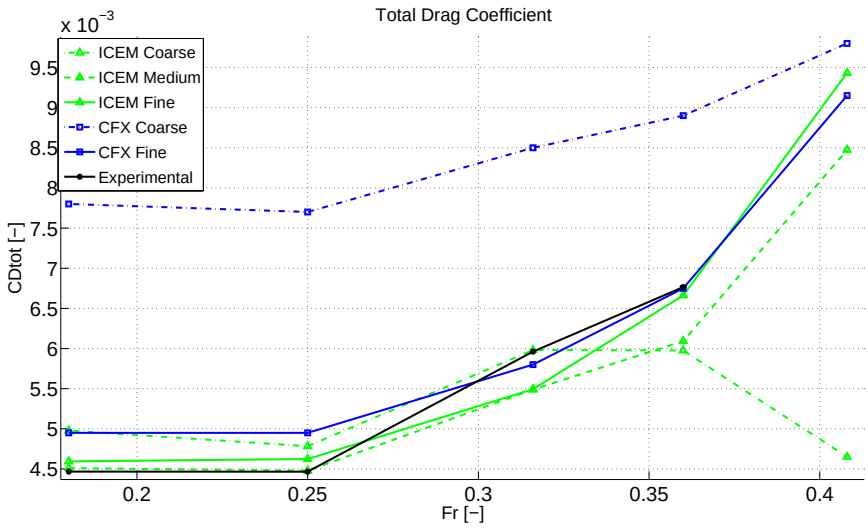


Figure 5.12: Total Drag Coefficient (CD). ICEM Meshes.

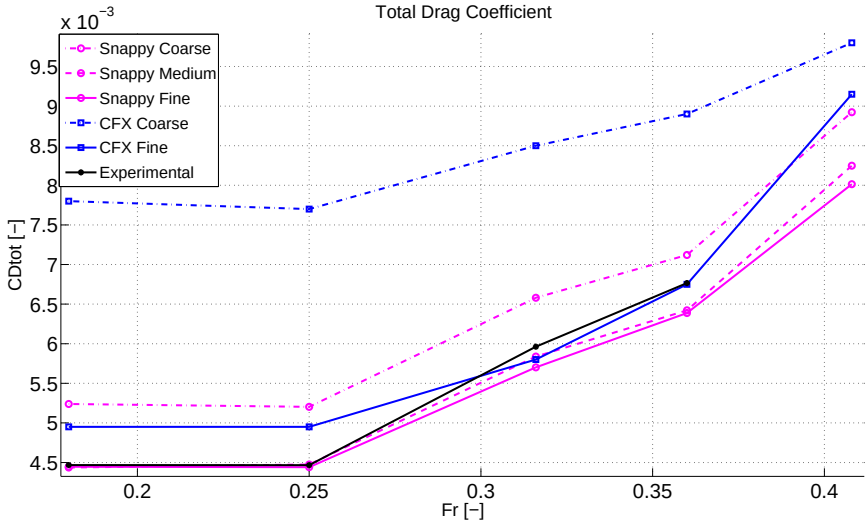
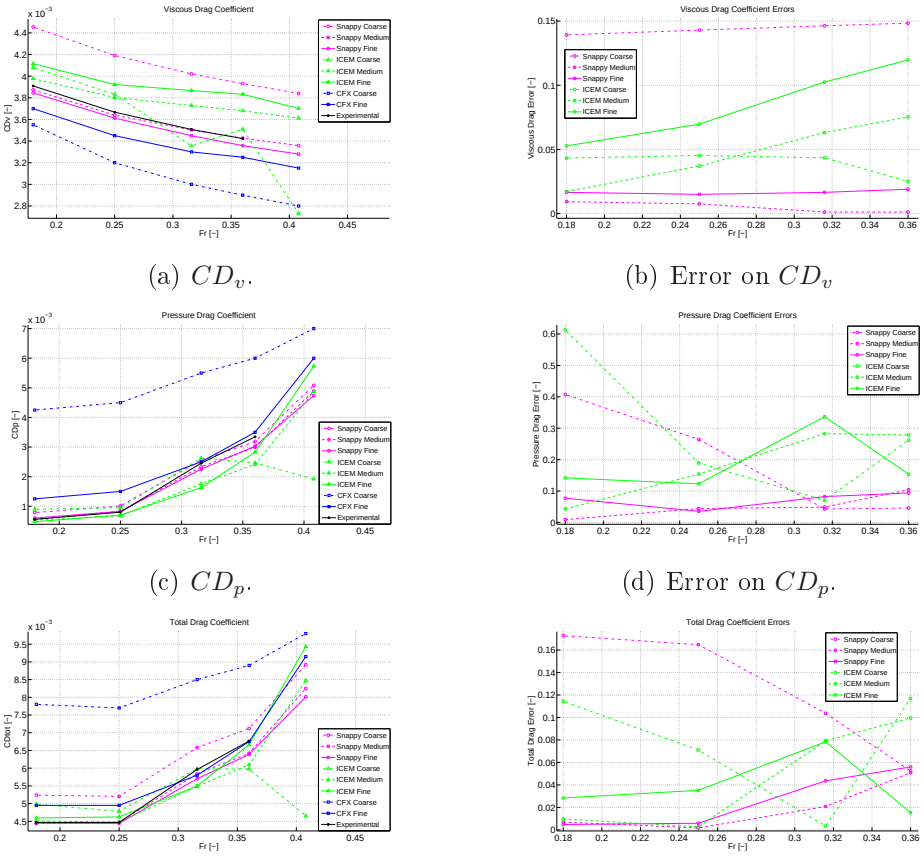


Figure 5.13: Total Drag Coefficient (CD). Snappy Meshes.



(a) CD_v.

(b) Error on CD_v

(c) CD_p.

(d) Error on CD_p.

(e) CD.

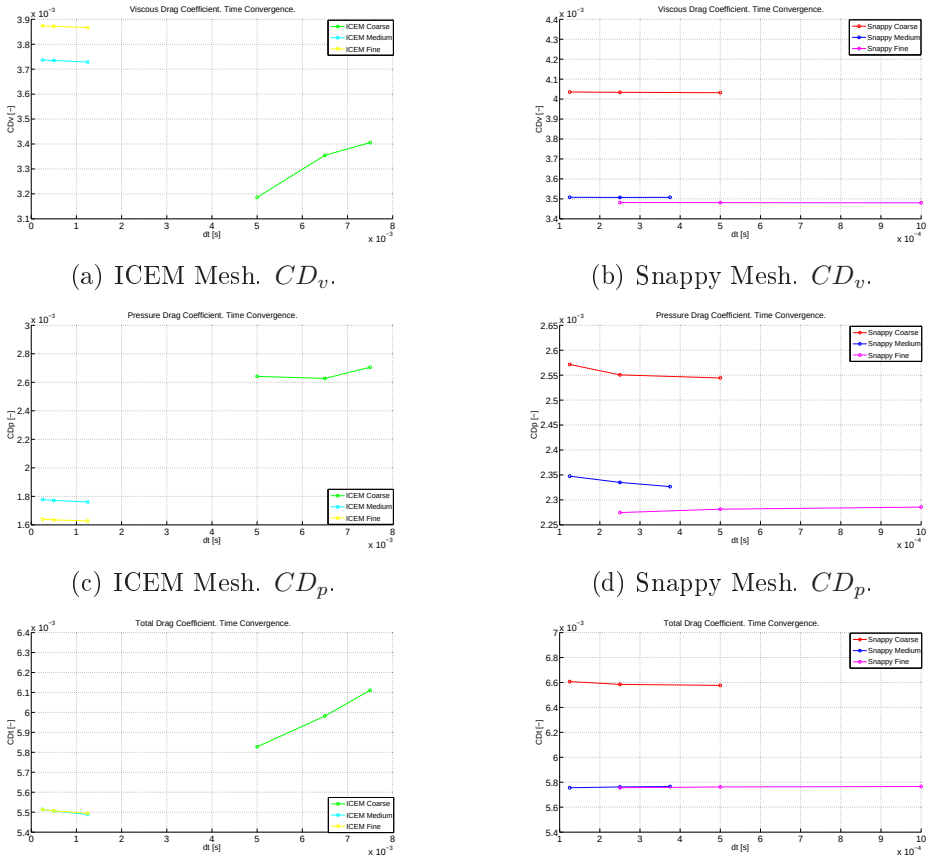
(f) Error on CD.

Figure 5.14: Mesh Comparison. CD_v, CD_p, CD and Errors.

5.2.2 Effects of Time-Step Refinement on Resistance

The choice of the time-step for time integration is quite important. Infact it has to be chosen within certain limits: if it is too large, the time integration procedure becomes unsteady (Courant number < 1 is not respected). If it is too small the Rhie and Chow correction will not properly work and the pressure and velocity fields will decouple and calculations will diverge. Furthermore, pressure and viscous resistance depend on the time-step chosen, therefore several runs have to be computed in order to determine the level of uncertainty by varying the time-step. Figure 5.15 shows results for different time-steps for each mesh. The variation in drag coefficient values is in general very small but little convergence exists. The only mesh that shows a non-negligible variation in drag coefficient is ICEM Coarse, which is the coarsest. This is reasonable because the dependency on time-step is higher for coarser meshes. ICEM Coarse mesh was the only one that showed instability for

small time-steps, in particular for time-steps smaller than 0.005s.



(a) ICEM Mesh. CD_v.

(b) Snappy Mesh. CD_v.

(c) ICEM Mesh. CD_p.

(d) Snappy Mesh. CD_p.

(e) ICEM Mesh. CD.

(f) Snappy Mesh. CD.

Figure 5.15: Time-Refinement Convergence

5.2.3 Static Simulations in the Presence of Waves

A further step towards a complete modelisation of a ship's running attitude is to simulate its behavior in the presence of external waves. In this section, a brief description of the theory of wave modelling is presented and results of simulations in the presence of waves with a fixed hull are shown. Further details on the Linear Wave theory and fifth order Stokes' solution can be found in [17], [18] and [19], while different numerical results are shown in [16].

Linear Wave Theory

Linear wave theory with Stokes' fifth order solution is now briefly described. The characteristic parameters of a wave are: amplitude a, wavelength λ, period T, propagation direction, phase and slope, defined as $2a/\lambda$. A wave is both a function of time and space. It is therefore possible to describe the wave's height η with the following equation:

$$\eta(\mathbf{x}, t) = \sin\left(\frac{2\pi}{T}t - \frac{2\pi}{\lambda}\mathbf{x}\right) \tag{5.4}$$

where $\frac{2\pi}{T}$ is the frequency (ω) and $\frac{2\pi}{\lambda}$ is the wave number (k). Frequency and wave number are related by the dispersion relation, given by:

$$\omega^2 = gk\tanh(hk) \tag{5.5}$$

where h is the height of the free-surface from the bottom and g is the gravitational acceleration. In case of deep water, since $\tanh(hk) \simeq 1$, the equation (5.5) yields:

$$\omega^2 = gk \tag{5.6}$$

The objective is to generate a steady periodic wave train, i.e. a train of bi-dimensional periodic stationary waves, propagating with constant velocity without any change of shape. These waves are homogeneous in the span wise direction, therefore one coordinate will be neglected. x is the longitudinal coordinate and z is the vertical one, positive pointing upwards. Furthermore, a frame fixed with the wave is adopted, therefore the problem will not be time dependent.

The simplifying hypothesis of incompressible and irrotational fluid permits to describe the velocity as a gradient of a scalar function Φ:

$$\mathbf{U} = \nabla\Phi \tag{5.7}$$

which, thanks to the continuity equation, gives the Laplace equation for the potential:

$$\nabla^2\Phi = 0 \tag{5.8}$$

In order to close the problem, boundary conditions must be defined. The

equations correctly describing the wave motion yield [4]:

$$\begin{cases} \dfrac{\partial^2\Phi}{\partial x^2} + \dfrac{\partial^2\Phi}{\partial z^2} = 0 & in \quad \Omega \\ \dfrac{\partial\Phi}{\partial x} = 0 & on \quad \partial\Omega_f = \{(x,z) \in \Omega : z = -h\} \\ \dfrac{\partial z}{\partial t} + \dfrac{\partial\Phi}{\partial x}\dfrac{\partial z}{\partial x} = \dfrac{\partial\Phi}{\partial z} & on \quad \partial\Omega_{SF} = \{(x,z) \in \Omega : z = -h\} \\ \dfrac{\partial\Phi}{\partial t} + \dfrac{p}{\rho} + \dfrac{\frac{\partial\Phi}{\partial u}^2 + \frac{\partial\Phi}{\partial z}^2}{2} + gz = 0 & on \quad \partial\Omega_{SF} = \{(x,z) \in \Omega : z = -h\} \end{cases} \tag{5.9}$$

where Ω is the computational domain corresponding to water only, Ω_f is the boundary corresponding to the bottom placed at $z = -h$ in the adopted frame and Ω_{SF} is the boundary corresponding to the free-surface. The second equation of system (5.9) imposes the vertical velocity on the bottom to be equal to zero. The last two equations of system (5.9) are conditions on the free-surface: the first one imposes the belonging of a particle to the free surface, a true condition until the wave doesn't brake. The second one is an equilibrium of forces, imposing the pressure on the free surface (calculated with Bernoulli's equation) to be equal to the atmospheric pressure. System (5.9) can be solved with a linear approximation, or, better, with the fifth order series expansion by Stokes. Stokes' theory uses the fifth order series expansion of the potential function with respect to the adimensional parameter ϵ for the resolution of system (5.9). ϵ is defined as:

$$\epsilon = ka = \frac{2\pi a}{\lambda} \tag{5.10}$$

From a physical point of view, ϵ describes the wave's slope. The smaller ϵ, the less suitable the linear theory.

The potential is given by:

$$\Phi(x,z) = C_0 \left(\frac{g}{k^3}\right)^{\frac{1}{2}} \sum_{i=1}^{5} \epsilon^i \sum_{j=1}^{i} A_{ij} cosh(jkz)\sin(jkx) + \vartheta(\epsilon^6) \tag{5.11}$$

In the specific case of deep water the potential yields:

$$\begin{aligned} \left(\frac{k^3}{g}\right)^{\frac{1}{2}} \Phi(x,z) =& \epsilon e^{kz}\sin(kx) - \frac{1}{3}\epsilon^3 e^{kz}\sin(kx) \\ &+ \frac{1}{2}\epsilon^4 e^{2kz}\sin(2kx) \\ &+ \epsilon^5(-\frac{37}{24}e^{kz}\sin(kx) + \frac{1}{12}e^{3kz}\sin(3kx)) \\ &+ \vartheta(\epsilon^6) \end{aligned} \tag{5.12}$$

[4] The derivation of system (5.9) is well described in [16]

while the free-surface is given by:

$$\begin{aligned} k\eta(x) =& \epsilon\cos(kx) + \epsilon^2\cos(2kx) \\ &+ \frac{3}{8}\epsilon^3(\cos(3kx) - \cos(kx)) + \frac{1}{3}\epsilon^4(\cos(2kx) - \cos(4kx)) \\ &+ \frac{1}{348}\epsilon^5(-422\cos(kx) + 297\cos(3kx) + 125\cos(5kx)) \\ &+ \vartheta(\epsilon^6) \end{aligned} \tag{5.13}$$

These equations are written in the wave-fixed frame. To write them in the original frame it is sufficient to apply a simple change of variable: $x = \overline{x} - ct$, where $\overline{x}$ is the coordinate of the wave-fixed frame in the original frame and $c = \dfrac{\lambda}{\tau} = \dfrac{\omega}{k}$ is the wave's speed of propagation. In our case, the frame moves with constant velocity U_{Re}, therefore, with another change of variables we have: $\overline{x} = \widetilde{x} - U_{Re}t$. The new angular velocity is: $\widetilde{\omega} = \omega + \kappa U_{Re}$. For simplicity, in the following, $\widetilde{\omega}$ and $\widetilde{x}$ are called ω and x.

Finally, the boundary conditions for velocity that need to be imposed in order to have the desired Steady Periodic Wave Train are [16]:

$$\begin{cases} u = \left(\dfrac{k}{g}\right)^{-\frac{1}{2}} \left[\epsilon e^{kx}\cos(kx) - \dfrac{1}{2}\epsilon^3 e^{kz}\cos(kx) \right. \\ \qquad \left. +\epsilon^4 e^{2kz}\cos(2kx) + \epsilon^5\left(-\dfrac{37}{24}e^{kz}\cos(kx) + \dfrac{1}{4}e^{3kz}\cos(3kz)\right)\right] \\ v = 0 \\ w = \left(\dfrac{k}{g}\right)^{-\frac{1}{2}} \left[\epsilon e^{kz}\sin(kx) - \dfrac{1}{2}\epsilon^3 e^{kz}\sin(kx) \right. \\ \qquad \left. +\epsilon^4 e^{2kz}\sin(2kx) + \epsilon^5\left(-\dfrac{37}{24}e^{kz}\sin(kx) + \dfrac{1}{4}e^{3kz}\sin(3kz)\right)\right] \end{cases} \tag{5.14}$$

and the phase fraction is given by:

$$\alpha = H(\eta(x,t) - z) \tag{5.15}$$

where H is the Heaviside function and $\eta(x,t)$ is defined in equation (5.13).

Boundary Conditions

Boundary and initial conditions for simulations in presence of waves are the same as described in section 5.1 except for the velocity and volume fraction. System (5.14) and (5.15) are imposed on the INFLOW while *zeroGradient* conditions are imposed on the remaining boundaries.

Results

A total of three different simulations with fixed boat were computed with the snappy Waves mesh, all at Froude number 0.316. The first one, with no waves imposed, was necessary to compare the results to the ones given by the non stream wise-refined meshes and to evaluate the variation of drag due to the presence of waves. The other two simulations had two different wave amplitude imposed: 0.0125m and 0.1m. This was done to verify the robustness of the numerical set-up in conditions that are 'tough' to simulate, such as in presence of big breaking waves with water flowing over the boat's deck also. This type of simulations (with free sink and trim) is necessary to verify the sea-keeping capabilities of commercial ships.

Figure 5.16 shows the stationary wave generated by the boat in flat water conditions. Figures 5.17 and 5.18 show, at different time-steps, the wave elevation for the simulation with 0.0125m waves. The overlap of the following two systems of waves is quite evident: the periodic wave train imposed at the inflow and the stationary wave generated by the presence of the hull. These figures suggest the periodicity of the entire system, and this is confirmed by figure 5.19 which plots viscous and pressure drag in function of time. Drag values are periodic and have the same period of the imposed waves. This means that forces can be described by an harmonic function having the same angular frequency of the imposed wave train and suitable amplitude and phase corrections:

$$F_i(t) = a_i \sin(\omega t + \Delta\phi_i) \tag{5.16}$$

The mean value of the viscous drag is higher with respect to the value of the flat water case. This is a reasonable result, since in presence of waves, the wet surface of the hull is bigger. A last thing should be highlighted: the waves' amplitude is reduced when moving in stream wise direction. This would seem to contradict the theory, which guarantees a constant wave amplitude. However, this is true only for a potential flow. The numerical dissipation responsible for damping can be reduced by refining the interface zone a lot, but this would impose an excessive number of CVs. Few numerical tests can be carried out in order to evaluate the correct wave amplitude that must be imposed on the INFLOW to have the desired wave amplitude when waves reach the boat.

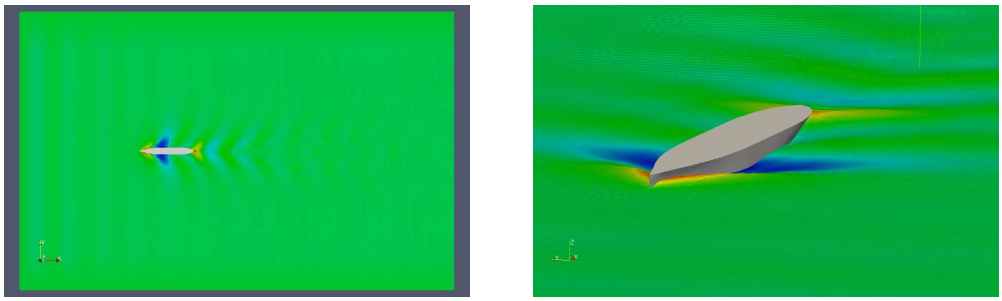


Figure 5.16: Wave Mesh. Simulations with no Waves. Top and Front View.

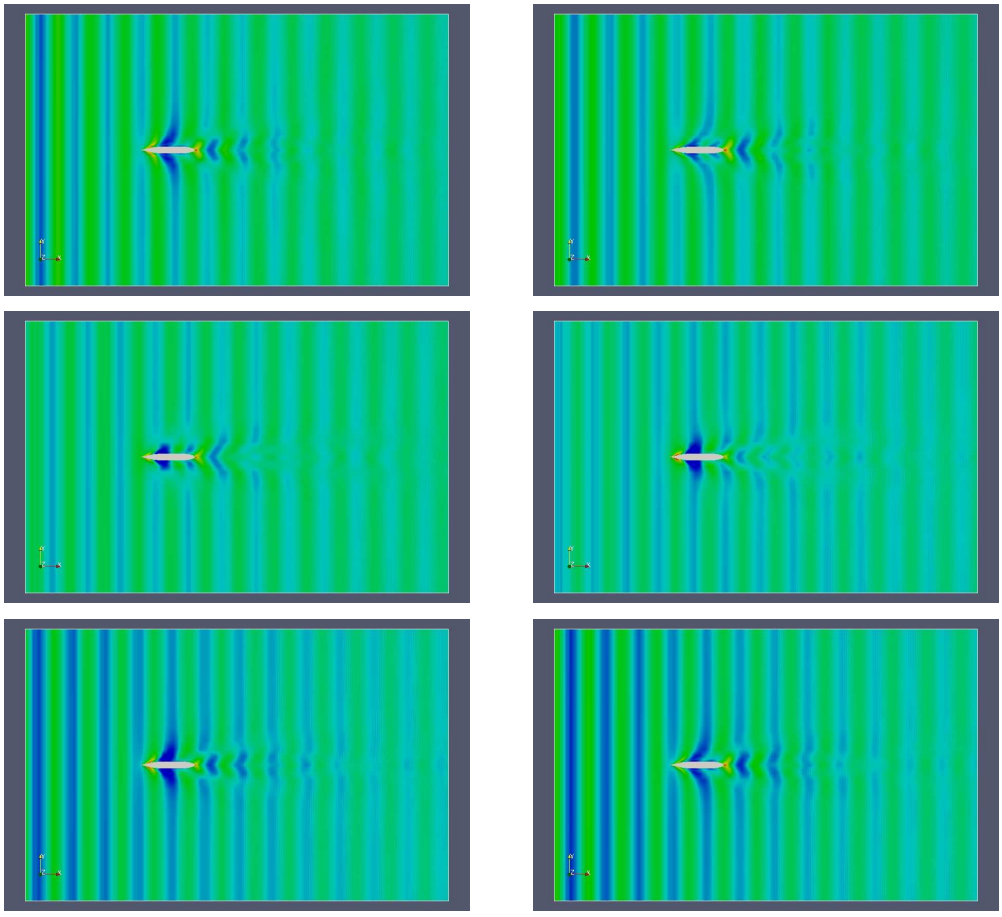


Figure 5.17: Simulation with Waves. Fr=0.316. Wave Amplitude=0.0125m. Top View. Different Time Steps.

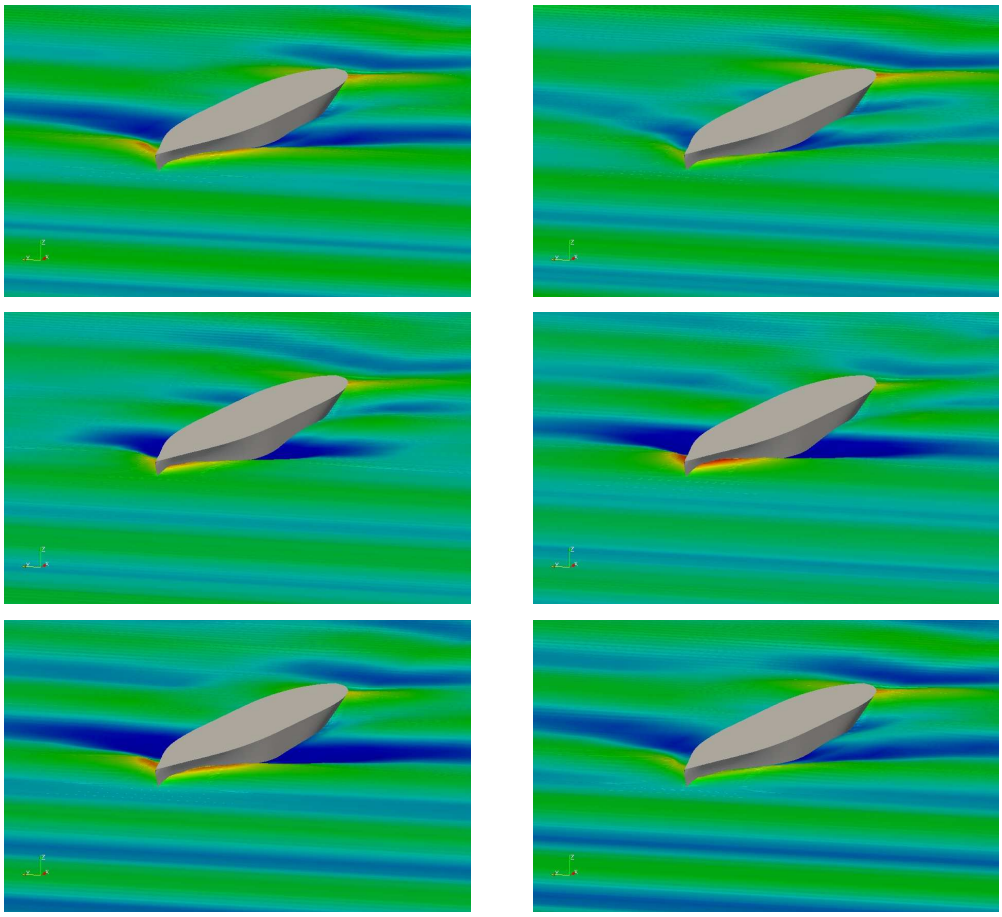
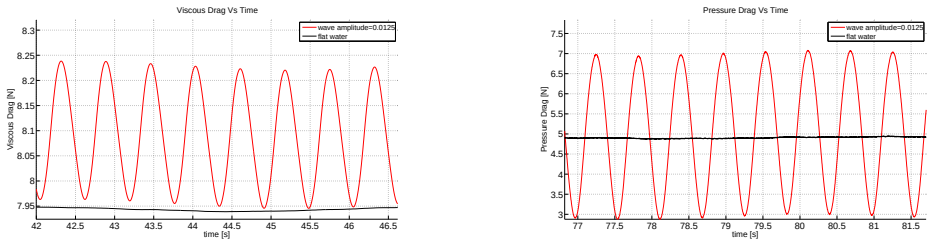


Figure 5.18: Simulation with Waves. Fr=0.316. Wave Amplitude=0.0125m. FrontView. Different Time Steps.



(a) Viscous Drag.

(b) Pressure Drag.

Figure 5.19: Simulation with Waves. Fr=0.316. Wave Amplitude=0.0125m. Viscous and Pressure Drag

Figures 5.20 and 5.21 show the free-surface elevation for the 0.1m wave amplitude simulation at different time-steps. The same considerations made

previously, concerning the periodicity of the phenomena, can be made. The interesting thing is that by choosing such a wave amplitude there is water flowing over the boat's deck. These cases often happen in reality and are paramount when treating sea keeping analysis. The influence of the presence of water on the boat's deck is evident in figure 5.22, where plots of the vertical force and pitching moment against time are shown. Vertical force and pitching moment are not simple sine type functions but are influenced by the water's weight flowing over its deck. Viscous and pressure drag are shown in figure 5.23. Their behavior remains substantially sinusoidal and the evident rise of viscous drag is due once again to the consistent rise of the wet surface.

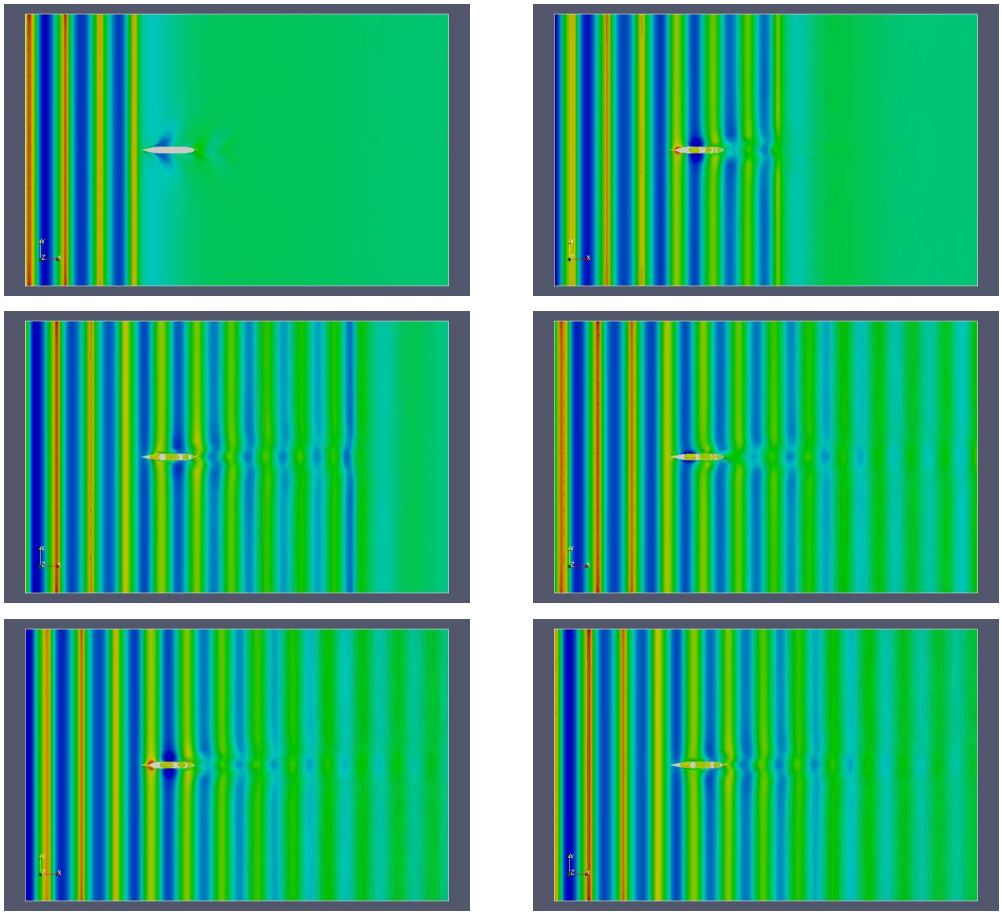


Figure 5.20: Simulation with Waves. Fr=0.316. Wave Amplitude=0.1m. Top View. Different Time Steps.

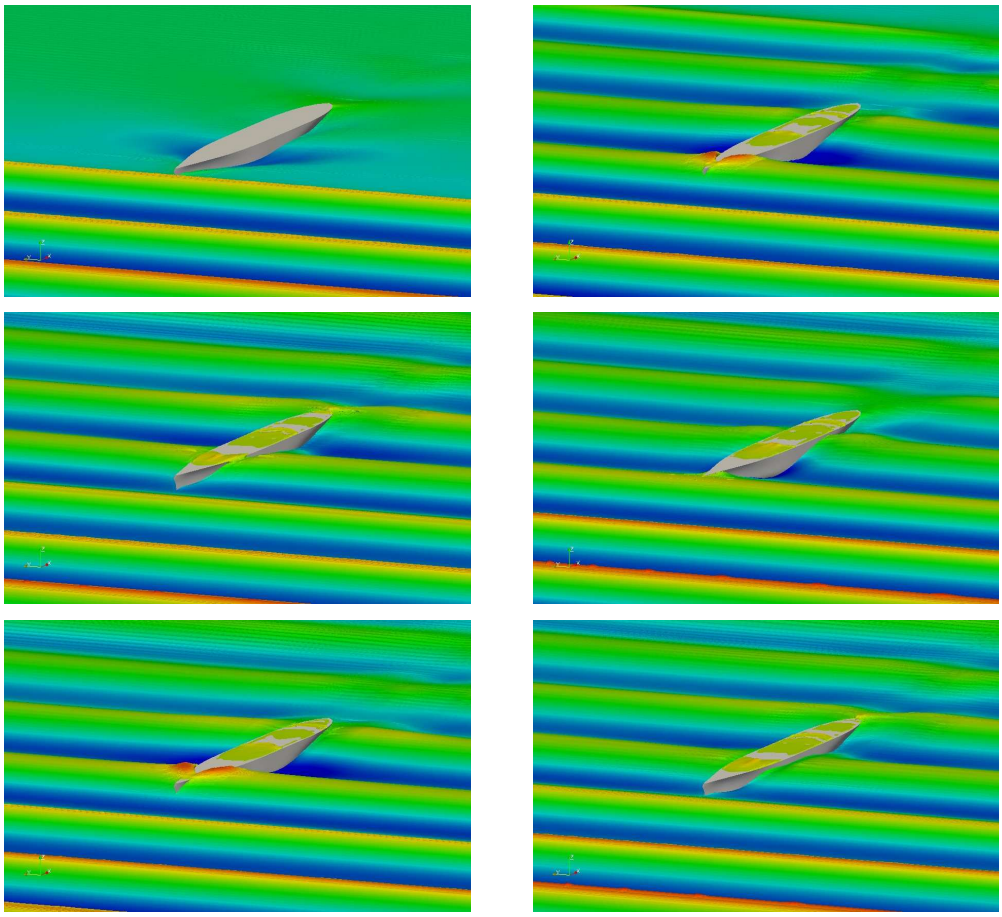
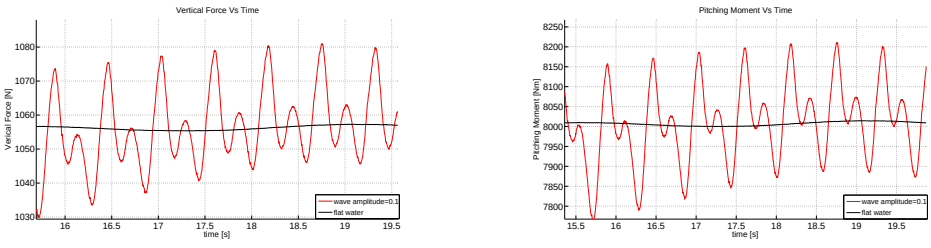


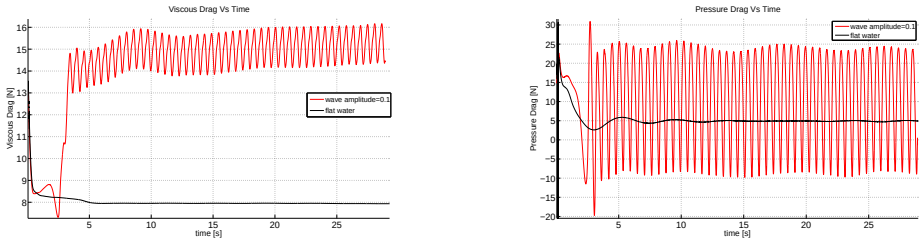
Figure 5.21: Simulation with Waves. Fr=0.316. Wave Amplitude=0.1m. FrontView. Different Time Steps.



(a) Vertical Force.

(b) Pitching Moment.

Figure 5.22: Simulation with Waves. Fr=0.316. Wave Amplitude=0.1m. Vertical Force and Pitching Moment.



(a) Viscous Drag.

(b) Pressure Drag.

Figure 5.23: Simulation with Waves. Fr=0.316. Wave Amplitude=0.1m. Viscous and Pressure Drag.

5.3 Dynamic Simulations

When the vessel sails at constant speed, the pressure field around the hull changes compared to still water conditions. As a result, the vertical force and its distribution changes. Consequently, the boat's attitude (in terms of sink and trim position) changes, implying a significant change in drag [5]. Therefore, to predict the real drag value, free sink and trim simulations have to be computed. Simulations computed to predict the hydrodynamic equilibrium attitude become crucial for ships sailing in restricted and shallow waters, that may risk touching the bottom as a consequence of the relevant change in sink position.

In the following, free trim simulations have been carried out on the Snappy-made meshes, both in flat water and in the presence of waves. Free sink and trim simulations were computed on the ICEM-made mesh and compared to experimental [20] and commercial code results [15], [16]. It was impossible to compute free sink and trim and free sink simulations on the Snappy-made meshes because an evident instability of forces caused the simulations to crash. It is still not clear why this occurred, since no mesh-checks were failed (also on the deformed mesh) and the numerical schemes and the numerical set-up was the same for the other simulations. Different types of mesh diffusivities and numerical schemes were tested, without success. This behaviour strictly depends on the hull's geometry, infact free sink and trim simulations with Snappy-made meshes is possible, as shown in section 5.3.5 where preliminary results form simulations on the Politecnico di Milano's 4.6 meters R3 class skiffs are given.

[5] In this work, 1DOF (trim) and 2DOF (sink and trim) simulations are computed, but, infact, it is possible to simulate a free body (6DOF).

5.3.1 Free Trim Simulations in flat water

Results from free trim simulations are now presented. The boat translations are fixed and the only unconstrained degree of freedom is the pitch rotation. The center of rotation is coincident with the center of gravity of the boat. Trim position is considered positive if the bow moves upwards. The boat's motion is a damped harmonic motion which converges to the equilibrium value, a behavior well shown in figure 5.24. Pitch angle and pitch velocity are correctly in counterphase and equilibrium value is reached in less then 10 s. Figure 5.25 plots viscous and pressure drag against time. Oscillations are negligible in the viscous drag, and pressure drag stabilizes after 70 s approximatively.

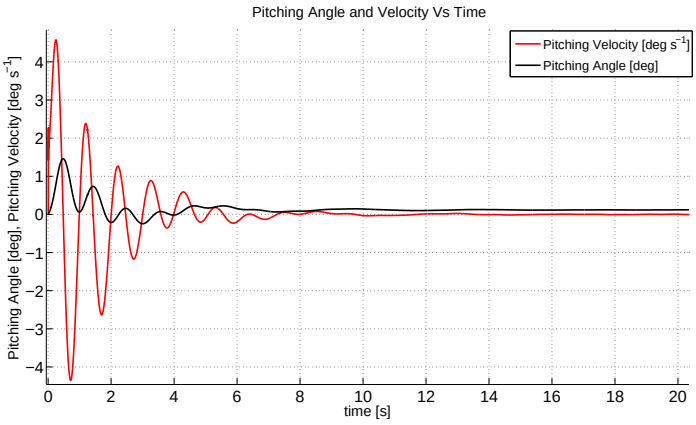


Figure 5.24: Free Trim Simulation. Fr=0.316. Pitching Velocity and Pitching Angle.

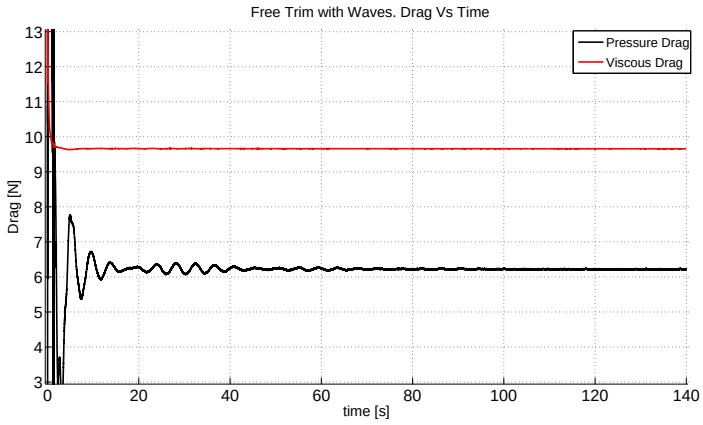


Figure 5.25: Free Trim Simulation. Fr=0.316. Viscous and Pressure Drag.

5.3.2 Free Trim Simulations in Presence of Waves

Free trim simulations in presence of waves were computed on the Snappy-made Waves mesh. The expected result is an harmonic behavior for drag, pitch angle and velocity. This is evident in figure 5.26 where pressure and viscous drag are plotted in function of time. Figure 5.27 shows the harmonic behavior of pitch velocity and pitch angle. It is evident that the mean trim angle is converging to a value different from zero. This is reasonable since the center of rotation was imposed as coincident with the center of gravity. Fluid dynamic moment with respect to the center of gravity is null in case of still water only: this is why the mean trim angle is converging to a non-zero position.

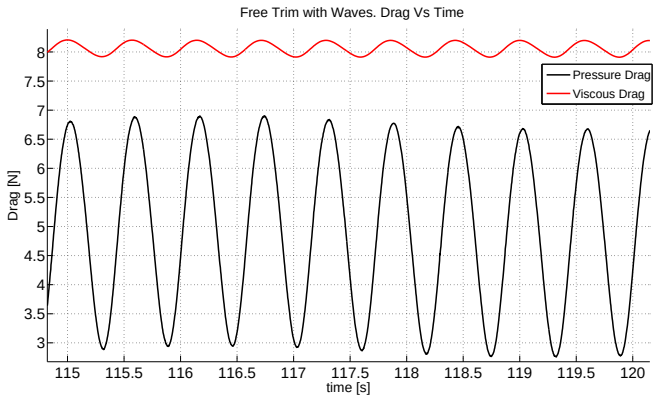


Figure 5.26: Free Trim Simulation with Waves. Fr=0.316. Wave Amplitude=0.0125m. Viscous and Pressure Drag.

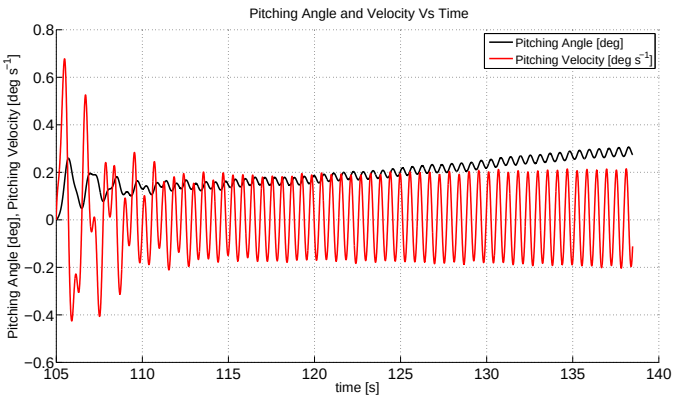


Figure 5.27: Free Trim Simulation with Waves. Fr=0.316. Wave Amplitude=0.0125m. Pitching Velocity and Pitching Angle.

5.3.3 Free Sink and Trim Simulations

Sink and trim simulations were computed using the ICEM-made Waves mesh. As initial conditions the solution of the fixed-hull simulations were imposed. Results of simulations in flat water are now compared to experimental and commercial CFD codes results. In figures 5.28, 5.29 and 5.30 typical forces sink and trim transitories are given. As shown in figure 5.28 the trim angle oscillates and converges to the equilibrium solution after approximatively 7 s. The equilibrium value is very close to experimental results. Figure 5.29 plots sink and sink velocity in function of time. Equilibrium is reached after 5 s, but the sink's value is slightly higher with respect to experimental ones. Figure 5.30 shows that pressure drag needs a bit more time to stabilize to the equilibrium value, approximately 30 s. The total drag is higher compared to experimental data. This is certainly caused by the higher sink obtained with OpenFOAM's simulation. This difference could be due to slightly different values of the mass used when modelling the hull, since such data was not available from experimental reports.

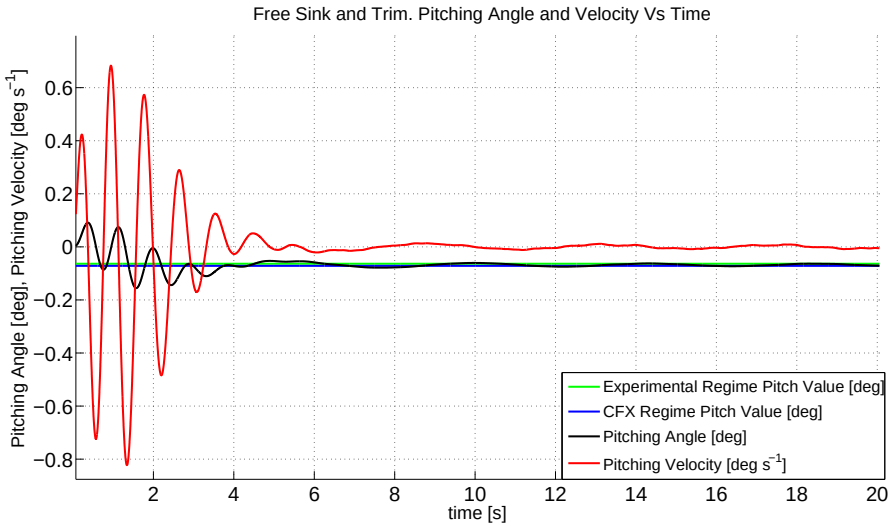


Figure 5.28: Free Sink and Trim Simulation. Fr=0.316. Pitching Velocity and Pitching Angle.

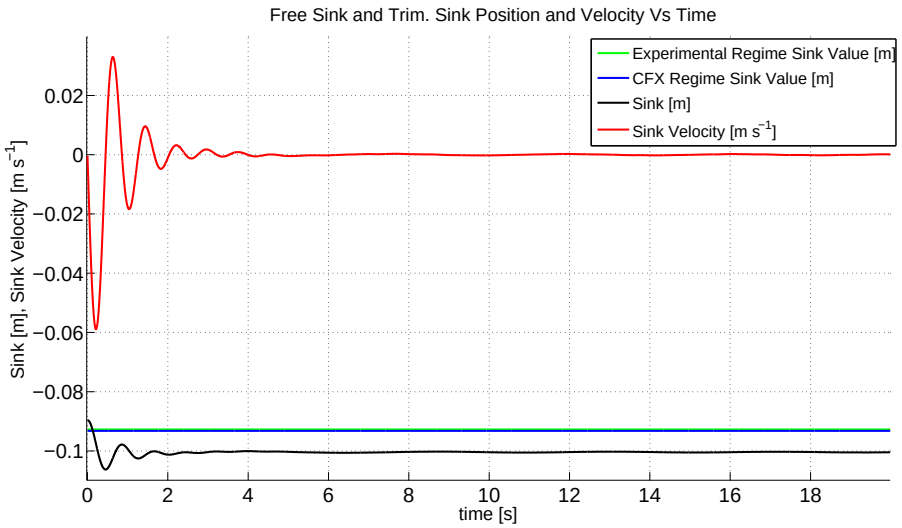


Figure 5.29: Free Sink and Trim Simulation. Fr=0.316. Sink and Sink Velocity.

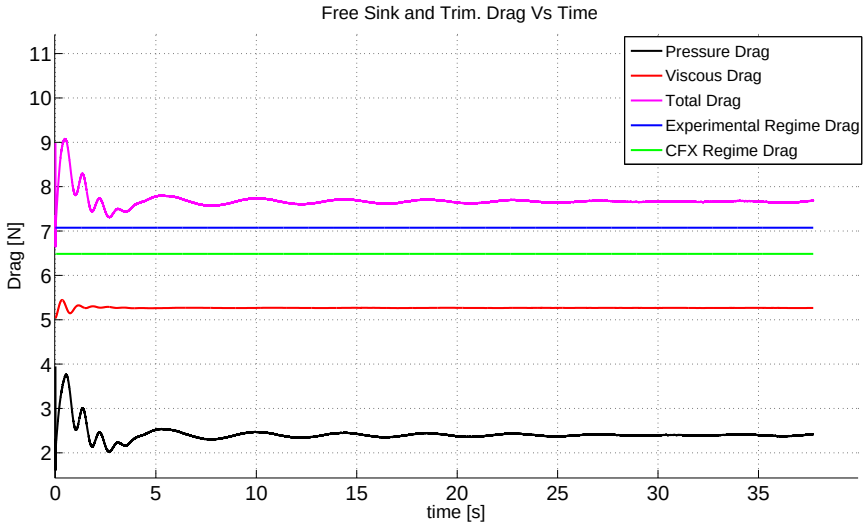
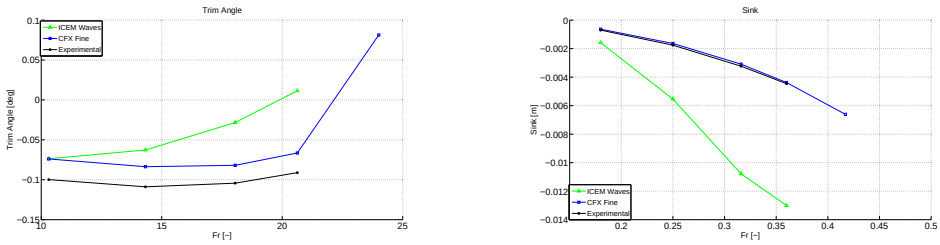


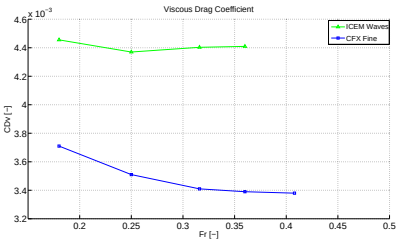
Figure 5.30: Free Sink and Trim Simulation. Fr=0.316. Viscous and Pressure Drag.

Figure 5.31 shows sink and trim equilibrium values for varying Froude number. The values computed by the OpenFOAM dynamic solver (*interDyMFoam*) with a mesh made with ICEM are once again a bit concerning. It is possible to affirm that the trend is correct, but numerical values are over-extimated. Figure 5.32 shows the values of CD_v, CD_p and CD at different Froude numbers. The viscous drag coefficient si quite over-extimated while pressure drag values seem congruent with experimental data. These results can not be considered satisfactory and the lack of quality in results is due to the mesh. Infact, the mesh used for these simulations had a high number (980) of *nonOrthogonalFaces*, although there were no failures during the mesh check. This is confirmed once again by the results given by the static solver (*interFoam*) using the same mesh (Figure 5.33). Same considerations made in section 5.2.1 can be done for these results. Furtermore, it took a longer time for forces to stabilise, sometimes showing non usual transitories and in certain cases ($Fr = 0.408$) the value of forces diverged and it was impossible to reach a steady state. This shows once again that particular attention must be given to the meshing process and results' sensitivity to the number of *nonOrthogonalFaces* is shown to be very high.

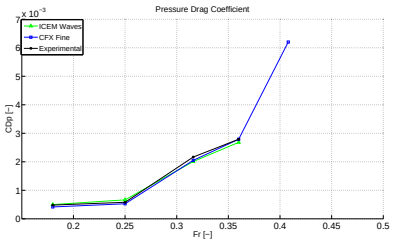


(a) Trim Angle. (b) Sink.

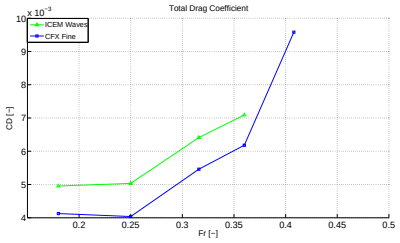
Figure 5.31: Free Sink and Trim Simulation. Trim Angle and Sink Vs Froude number.



(a) CD_v.

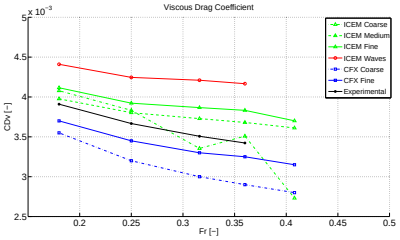


(b) CD_p.

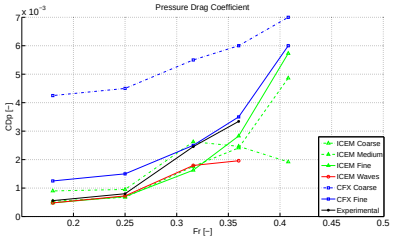


(c) CD.

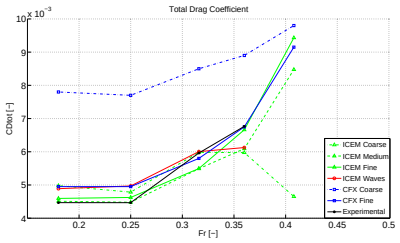
Figure 5.32: Free Sink and Trim Simulation. CD_v, CD_p and CD Vs Froude number.



(a) CD_v.



(b) CD_p.



(c) CD.

Figure 5.33: Waves Mesh. Fixed Hull simulations. CD_v, CD_p and CD Vs Froude number.

5.3.4 Free Sink and Trim Simulations in Presence of Waves

Free sink and trim simulations in presence of waves of an amplitude of 0.0025 m were computed on the ICEM Waves mesh. Again, an harmonic response with the same frequency of the imposed wave train is expected. Figures 5.34, 5.35 and 5.36 widely confirm previous statements: indeed, sink, trim and drag oscillate with the same frequency of the wave motion.

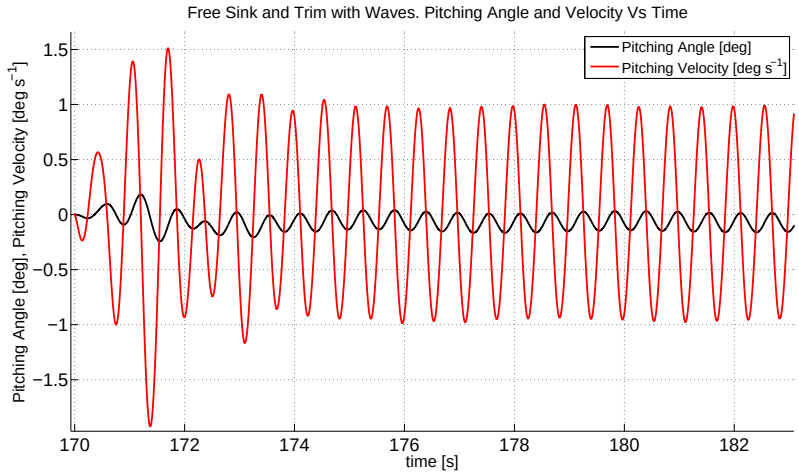


Figure 5.34: Free Sink and Trim Simulation with Waves. Fr=0.316. Wave Amplitude=0.025m. Pitching Velocity and Pitching Angle.

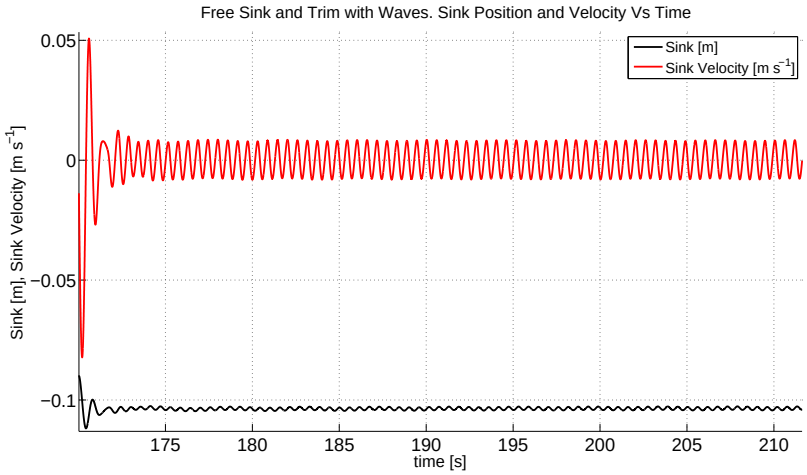


Figure 5.35: Free Sink and Trim Simulation with Waves. Fr=0.316. Wave Amplitude=0.025m. Sink and Sink Velocity.

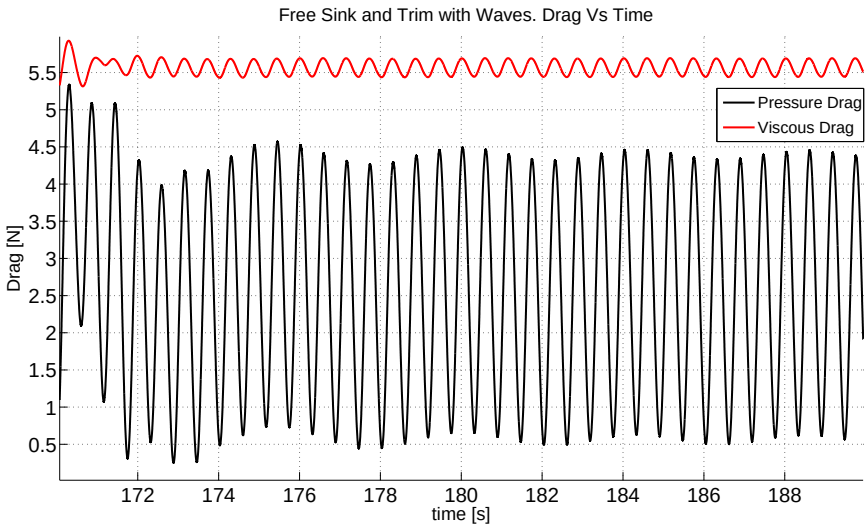


Figure 5.36: Free Sink and Trim Simulation with Waves. Fr=0.316. Wave Amplitude=0.025m. Viscous and Pressure Drag.

5.3.5 Simulations of Politecnico di Milano's 4.6m R3 class skiffs

Modern sailing boats sail in planning conditions, while commercial ships sail at slower speed, therefore, modern sailing boat hulls are particularly different from the Series 60 presented above. Planning hulls have very little immersion and edges can be found near the water line. These two features introduce some issues, when computing a numerical simulations, not present in the case previously presented. Edges need to be carefully represented, therefore particular attention when making the mesh is needed. The little immersion of these hulls an their development in the free-surface parallel direction can easily bring to a phenomena called ¨numerical ventilation ˝when simulating. Numerical ventilation occurs when the air-water interface penetrates under the hull, therefore a non physical mixture of air and water will flow near the keel. This is a purely numerical phenomena, thus called numerical ventilation. Numerical ventilation can easily cause big errors when calculating the forces acting on the hull, since forces depend on the properties (viscosity and density) of the fluid acting on the keel. The error when computing resultant forces depend on the extension of the phenomena and on the volume fraction. The presence of numerical ventilation strongly depends on the mesh and the use of compression schemes for the interface.

In this section a preliminary work on the Politecnico di Milano's skiffs is done. These boats were designed, built and sailed by students of Politecnico

di Milano for the competition ¨Mille e Una Vela per L'Università˝, a competition open to European universities. These hulls (figures 5.37 and 5.38) are characterized by the two features aforementioned: little immersion and the presence of an edge near the water line. The choice of these to hull shapes is due also to the presence of previous work done with commercial codes ([21] and [22]).

Figure 5.39 shows the volume fraction value on the Version 2.0 skiff hull, when the interface compression is or is not adopted for in a static simulation ($Fr = 0.3$). When using the compression scheme, the volume fraction is forced to stay in limited areas, therefore the particular pattern for the volume fraction shown in figure 5.39 (a). On the counter part, when the compression factor is not used, the volume fraction is smeared out on all the keel, and a value corresponding to 1 is present only at the center of the hull.

The effect of the compression factor is well shown in figure 5.40, where the pressure, viscous and total drag values are given as well as the vertical force. As shown, when the compression factor is adopted, the pressure drag is almost $1N$ lower (3% difference), while the viscous drag is higher by $0.3N$, (1.5%) and the total drag difference is about 0.5%. The difference in the vertical force is about 1.2%.

It is possible to affirm that the presence of numerical ventilation depends on the quality of the mesh. It must be noticed that the interface is ¨sucked ˝under the hull more ore less one meter behind the bow, in correspondence of a point where the mesh does not perfectly respect the edge present in the hull geometry. Therefore, the first thing to do should be to assure a full respect of geometry when meshing. Secondly, when a compression scheme is not adopted, refining further more the mesh would certainly improve results and the volume fraction value on the keel would probably be much closer to one. These are all tests that should be carried out. Figure 5.41 shows the volume fraction value on the keel of th Version 1.0 skiff. This simulation was computed on a mesh where the edge was even less respected with respect to the Version 2.0 in figure 5.39. Figure 5.42 shows different instants of a free trim and sink simulations for the Version 2.0 skiff. Further studies have to be carried out to verify the stability of the computation and the quality of results.

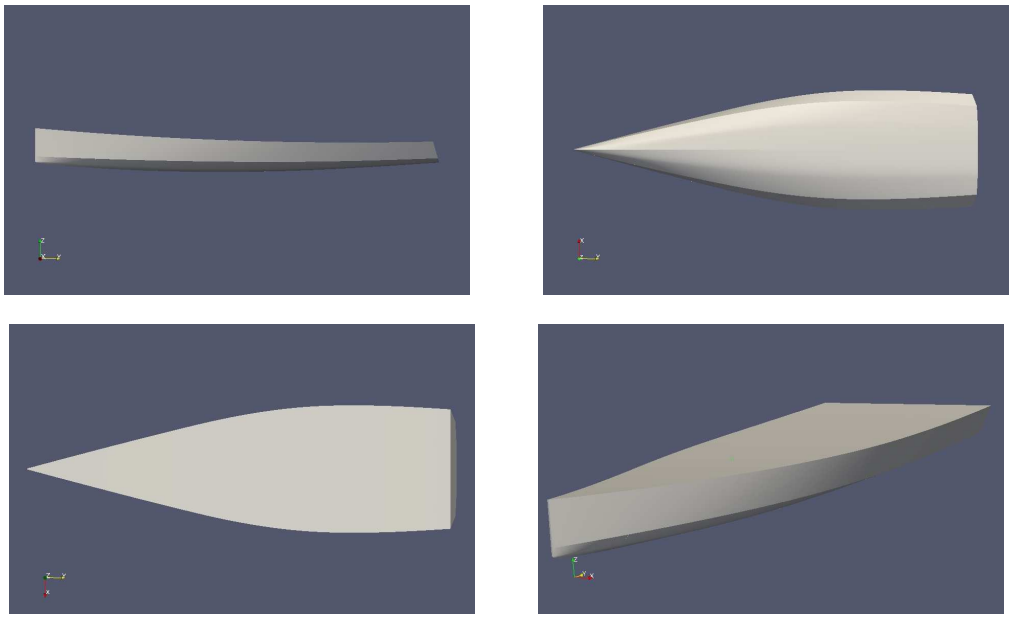


Figure 5.37: PoliMi R3 skiff Version 1.0

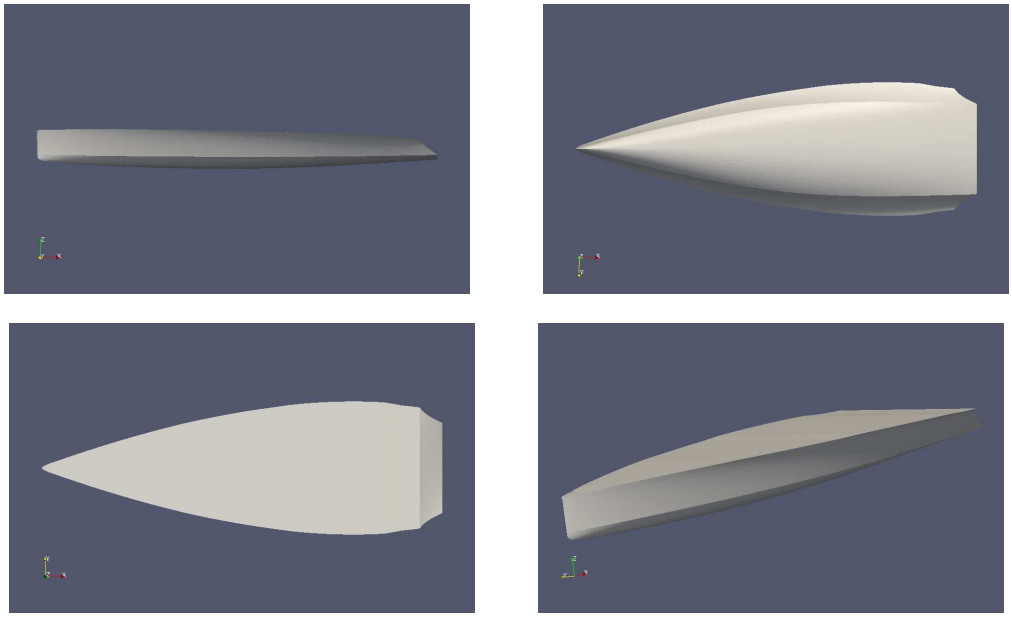
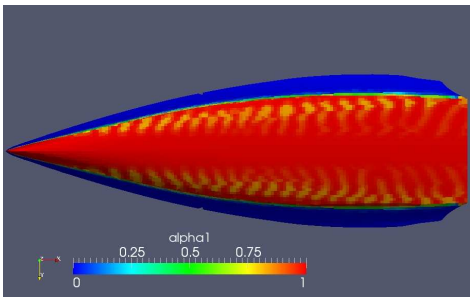
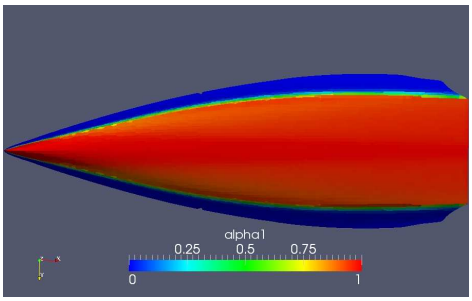


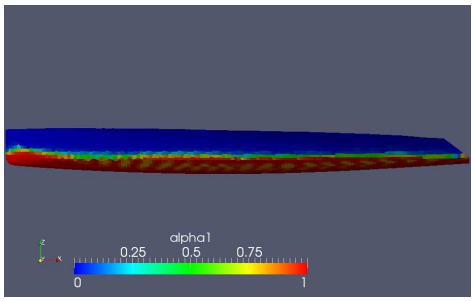
Figure 5.38: PoliMi R3 skiff Version 2.0



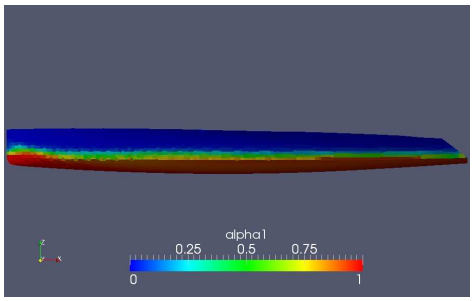
(a) Compression Factor On



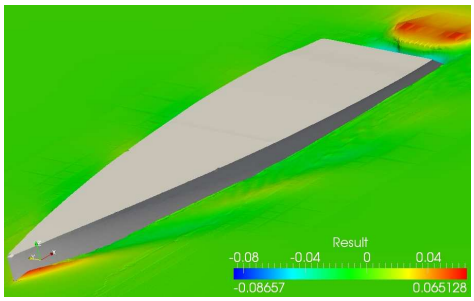
(b) Compression Factor Off



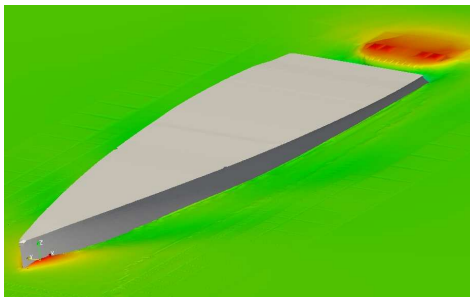
(c) Compression Factor On



(d) Compression Factor Off



(e) Compression Factor On



(f) Compression Factor Off

Figure 5.39: PoliMi R3 skiff Version 2.0. Numerical Ventilation.

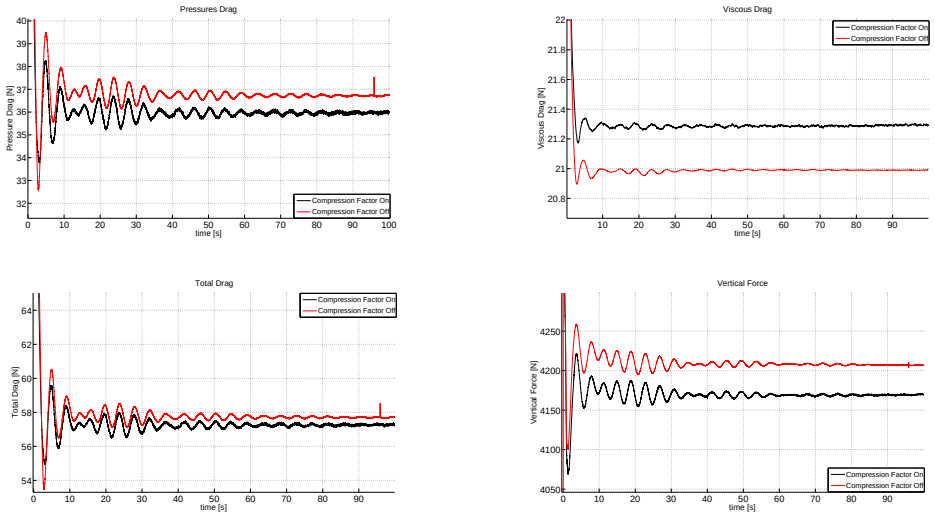


Figure 5.40: PoliMi R3 skiff Version 2.0. Compression Factor Effects on Forces.

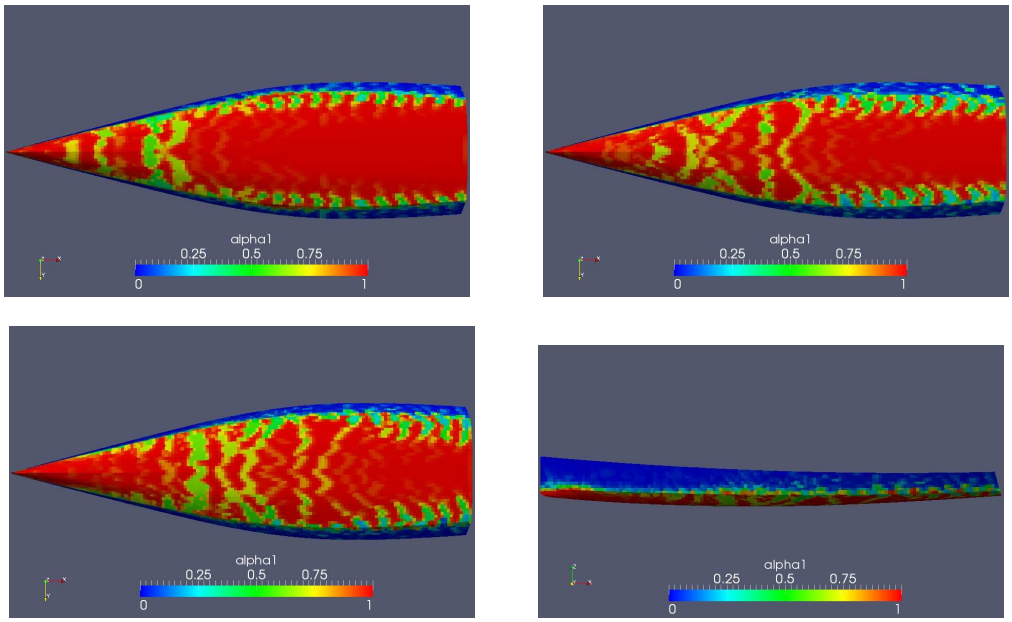


Figure 5.41: PoliMi R3 skiff Version 1.0. No Compression Factor. Numerical Ventilation.

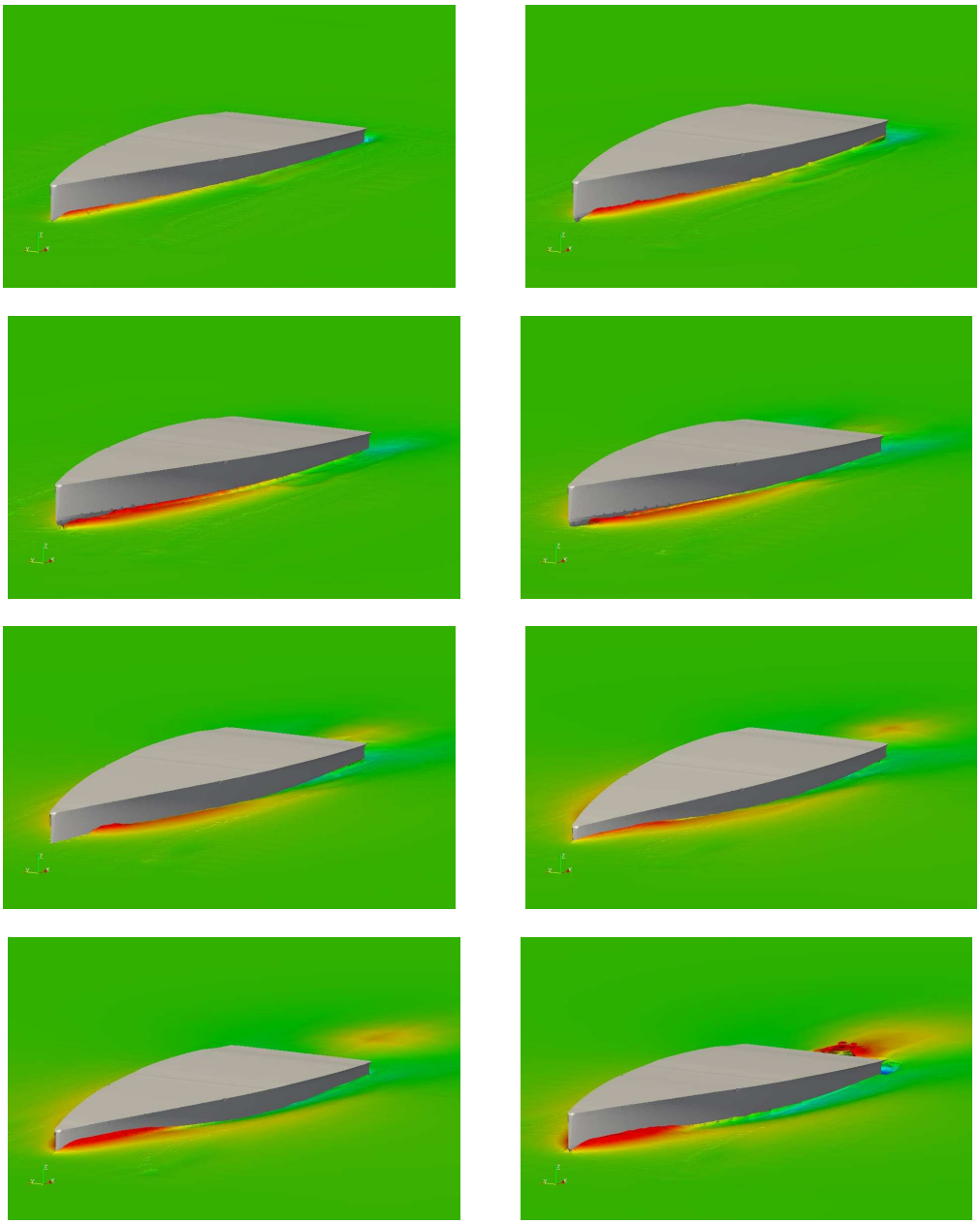


Figure 5.42: PoliMi R3 skiff Version 2.0. No Compression factor. Free Sink and Trim Simulation. Different Instants.

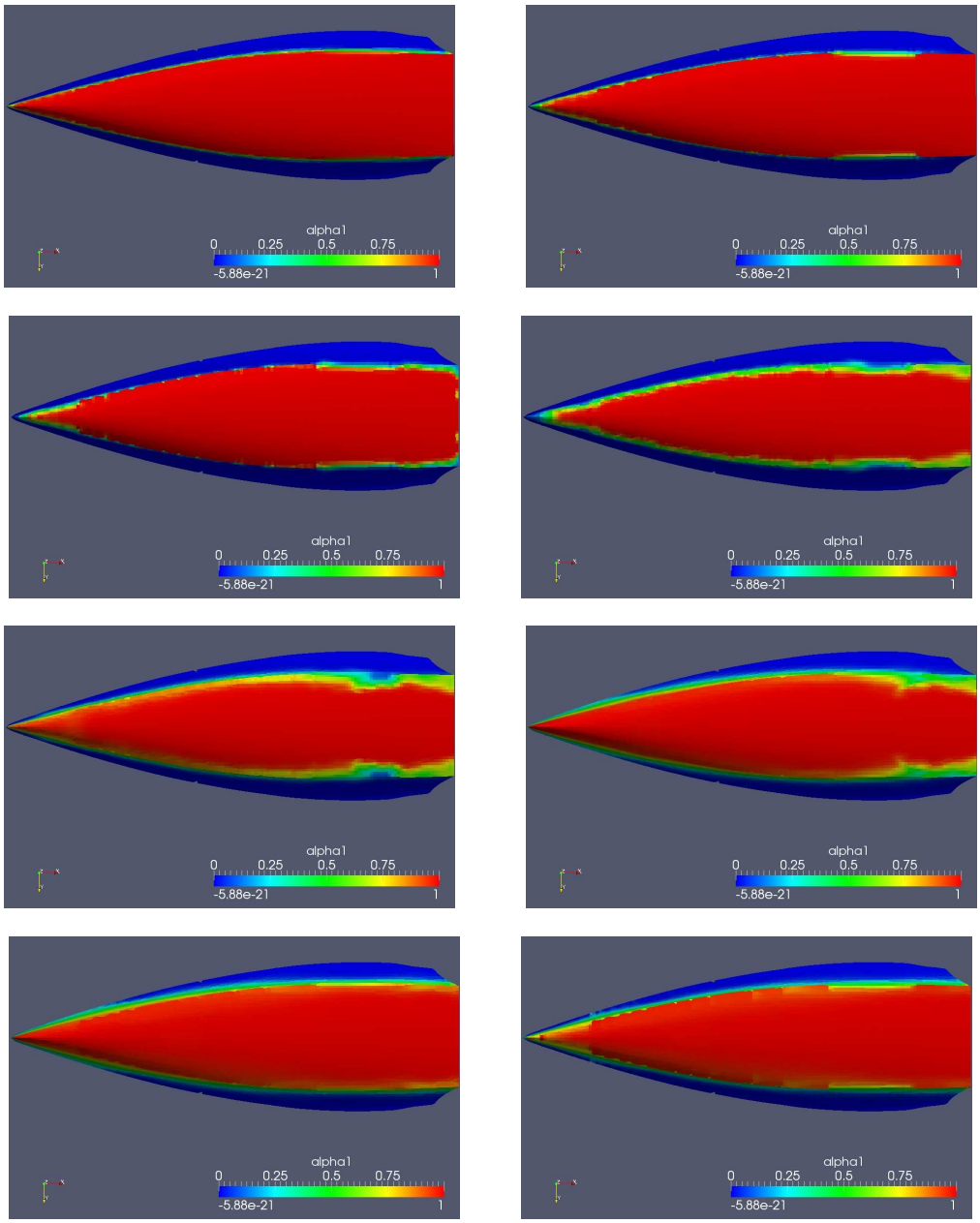


Figure 5.43: PoliMi R3 skiff Version 2.0. No Compression Factor. Free Sink and Trim Simulation. Different Instants. Bottom View

Chapter 6

Closure

In the following, the work done is summarized, conclusions are drawn and suggestions for future work are given.

Summary and Conclusions

First of all, the development of the OpenFOAM grid generator has been carried out. In order to obtain a satisfactory quality in geometry reproduction, the original mesher, *snappyHexMesh*, has been integrated with a modified version of *snapEdge*. Non-isotropical refinement in the interface zone was done using the embedded utility called *refineMesh*. The selection of CV belonging to the interface zone, excluding the ones previously refined by the meshing process, was done by *createSet*, a utility ad-hoc written for this purpose.

The static two-phase solver (interFoam) has been extensively validated. Meshes from snappyHexMesh show very good grid-refinement convergence, and numerical results show a discrepancy of less than 6% from experimental data. Results show very little sensitivity to time refinement. On the counterpart, structured meshes made with the commercial mesher ICEM, by ANSYS, did not show an evident convergence to grid refinement and results were quite disappointing. Probably, this depends on the difficulty of fully satisfying the mesh quality requests by OpenFOAM concerning the non-orthogonality.

The dynamic two-phase solver (interDyMFoam) has been successfully tested with unconstrained sink and trim degrees of freedom, both in flat water conditions and with imposed waves, using the ICEM-made mesh. The difference in sink value and, consequently, in drag, with respect to experimental data, is probably caused by a slight difference in the boat's mass used in simulations and the high number of *nonOrthogonalFaces* present in the mesh. Sink and trim simulations done with snappy-made meshes showed

forces instability and crashed after a few iterations. The reason for this is still not clear, since no mesh failures were found and the numerical set-up was the same as in all other simulations. Furthermore, simulations with only the trim degree of freedom left unconstrained were carried out without any problems, also in presence of waves.

A preliminary study on the Polimi R3 skiffs has been carried out. Problems concerning "numerical ventilation"(problem quite often when simulating modern planning hulls) are present and are found to be strongly dependant on the mesh quality and the eventual use of the compression factor for the inteface.

Future Work

Meshes made with the meshing process developed in this work give good results, but the meshing process itself is still too case-sensitive, therefore very hard to automatise. Unstructured meshes are widely used in industry because of the little men-hours required and the possibility to insert the grid generator in an automatized processes. Further work should be done in this direction, in order to make this mesh generator fully competitive with commercial ones.

The interFoam solver has been validated, but only with the $\kappa-\omega SST$ turbulence model. Tests with different turbulence models, such as the Spalart-Allmaras are strongly suggested. The study of the sensitivity of results to the boundary layer cell's dimension is quite interesting. On complex geometries, it is often difficult to impose a correct and uniform Y^+ [1] on the entire surface and an evaluation of how results change in function of the Y^+ is quite important. Furthermore, the introduction of adaptive wall-functions (work done by Mafioletti [23] and Benelli [24] for example) could seriously improve results' quality. The interFoam solver (PISO scheme) is still slower compared to commercial codes, mainly because of the high number of pressure-velocity corrections needed to stabilize results. In order to reduce computational time, different strategies could be employed: adopt a PIMPLE scheme, (a PISO scheme with SIMPLE scheme sub-iterations), implement a local time-stepping scheme (which only sub-iterates where the cells are smaller, to locally satisfy the condition of Courant number < 1) or to switch to a multi-grid approach.

For the interDyMFoam solver, the same considerations for the static solver, concerning turbulence models and computational time, can be done.

[1] Y^+ is defined by: $Y^+ = Y_c U_\tau / \nu$ where ν is the fluid viscosity, Y_c is the cell dimension in the wall normal direction and U_τ is the friction velocity, given by $U_\tau = \sqrt{\tau_w/\rho}$. τ_w is the shear stress on the wall.

Other tests regarding free sink simulations with unstructured grids should be performed to understand the reasons behind the numerical instabilities that made the simulations crash.

Bibliography

[1] F. R. Menter, *Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications* AIAA Journal, Vol. 32, No. 8,pp. 1598-1605, August 1994.

[2] F. R. Menter, *Zonal Two Equation $\kappa - \omega$ Turbulence Models for Aerodynamic Flows*, AIAA Paper 93-2906, 1993.

[3] C. L. Bottasso, *Three-Dimensional Rotations*, Dispense del Corso di Meccanica del Volo 2

[4] F. Nobile, *Numerical Approximation of Fluid Structure Interaction Problems With Application to Haemodynamics*, Ph.D. Thesis, EPFL Lausanne, 2001.

[5] H. Rusche, *Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions*, Ph.D. Thesis, Imperial College of Science Technology and Medicine, Department of Mechanical Engineering, 2002.

[6] S. M. Damián, D. E. Ramajo and N.M. Nigro, *Volume Of Fluid Simulation Of Borda Mouthpieces*, Mecánica Computacional Vol XXIX, pp. 3741-3750.

[7] P. B. Rodriguez de Medina, *Study and Numerical Simulation of Sediment Transport in Free-Surface Flow*, Ph.D. Thesis, University of Malaga, E.T.S. Ingenieros Industriales Department of Mechanical Engineering and Fluid Mechanics, 2008.

[8] F. P. Karrholm, *Numerical Modelling of Diesel Spray Injection, Turbulence Interaction and Combustion*, Ph.D Thesis, Department of Applied Mechanics Chalmers University of Technology Goteborg, Sweden, 2008.

[9] N. Nordin, http://openfoamwiki.net/index.php/Contrib_snapEdge.

[10] Y. Toda, F. Stern, J. Longo, *Mean-Flow Measurement in the Boundary Layer and Wake and Wave Field of a Series 60 CB = 0.6 Ship Model. Part*

1:Froude Numbers 0.16 and 0.36, Journal of Ship Research, vol 36, No. 4, pp. 360-377, 1992.

[11] F. Stern and J. Longo, *Evaluation of Surface-Ship Resistance and Propulsion Model-Scale Database for CFD Validation*, Proceedings 1st Symposium on Marine Application of Computational Fluid Dynamics, 19-21 May 1998, McLean, VA.

[12] F. Stern, R. V. Wilson, H. W. Coleman and E. G. Paterson, *Verification And Validation of CFD Simulations*, Iowa Institute of Hydraulic Research and Propulsion Research Center Mechanical and Aerospace Engineering Department University of Alabama in Huntsville.

[13] IIHR-Hydroscience and Engineering, Hydroscience and Engineering Ship Hydrodynamics Website, Series 60 steady flow, http://www.iihr.uiowa.edu/

[14] R. Azcueta Repetto, *Computation of Turbulent Free-Surface Flows Around Ships and Floating Bodies*, Ph.D Thesis, Technischen Universitat Hamburg , 2001.

[15] M. Lombardi, *Simulazione Numerica della Dinamica di uno Scafo*, Master Thesis, MOX, Politecnico di Milano, 2006.

[16] S. Piazza, *Simulazioni Numeriche della Dinamica di uno Scafo in Mare Ondoso*, Master Thesis, MOX, Politecnico di Milano, 2006.

[17] J. D. Fenton, *A Fifth-Order Stokes Theory For Steady Waves*, J. Waterway, Port, Coastal and Ocean Engineering, 111, pp. 216-234, 1985.

[18] T. Kleesfsman, *Water Impact Loading on Offshore Structure*, Rijksuniver-siteit Groningen, ISBN 90-367-2385-X, 2005.

[19] R. G. Dean, R. A. Dalrymple, *Water Wave Mechanics for Engineers and Scientists*, World Scientific, 1984.

[20] C. Yang and R. Lohner, *Calculation of Ship Sinkage and Trim Using a Finite Element Method and Unstructured Grids, International Journal of Computational Fluid Dynamics*, Vol. 16 (3), pp.217-227, 2002.

[21] S. Bartesaghi, *Hydrodynamic Performance di uno Skiff classe R3*, http://www.simonebartesaghi.com, February 2010.

[22] S. Bartesaghi, *Performance Prediction di uno skiff R3 Class mediante Artificial Neural Network*, http://www.simonebartesaghi.com, December 2010.

[23] M. Mafioletti *Simulazione di Flussi Turbolenti in Ambito Automotive con Codice Open-Source*, Mater Thesis, DIA, Politecnico di Milano, to be published.

[24] S. Benelli, *Implementazione e verifica di un modello di turbolenza* $\overline{v'^2} - f$ *in OpenFOAM*, Master Thesis, DIA, Politecnico di Milano, 2010.

[25] D. Detomi, N. Parolini and A. Quarteroni, *Numerical Models and Simulations in Sailing Yacht Design*, Lecture Notes in Computational Science and Engineering, 2009.

[26] F. R. Manter, *Zonal-Two Equation* $\kappa - \omega$ *Turbulence Models for Aerodynamic Flow*, AIAA Technical paper, 93-2906, 1993.

[27] C. Kassiotis, *Which Strategy to Move the Mesh in the Computational Fluid Dynamic Code OpenFOAM*, April 12, 2008.

[28] M. Lombardi, N. Parolini, G. Rozza and A. Quarteroni, *Numerical Simulation of a Sailing Boats: Dynamics and Shape optimization*, to be published.

[29] H. Jasak, *Dynamic Mesh Handling in OpenFOAM*, American Institute of Aeronautics and Astronautics, 2009.

[30] G. Passoni, R. Ponzini and I. M. Viola, *Simulazioni Fluidodinamiche di una Imbarcazione di Coppa America ai Limiti delle Possibilità Computazionali*, Bollettino del CILEA N. 114, December 2009.

[31] K. Kissling, J. Springer, H. Jasak, S. Schutz, K. Urban, M. Piesche, *A Coupled Pressure Based Solution Algorithm Based on The Volume-Of-Fluoid Approach for Two or More Immiscible Fluids*, V European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010.

[32] C. M. Rhie and W. L. Chow, *Numerical Study of the Turbulent Flow Past an Airfoil With Trailing Edge Separation*, AIAA Journal, vol. 21, pp. 1525-1532, 1983.

[33] F. Fossati, *Aero-Hydrodynamics and the Performance of Sailing Yachts: The Science Behind Sailboats and Their Design*, McGraw-Hill, 2009.

[34] J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics*, Springer, 2002.

[35] A. Quarteroni and F.Saleri, *Introduzione al Calcolo Scientifico*, Springer, 2004.

[36] A. Quarteroni, *Modellistica Numerica per Problemi Differenziali*, Springer, 2006.

[37] B. Stroustrup, *The C++ Programming Language*, Addison-Wesley, 2004.

[38] A. Baron *Dispense del Corso di Fluidodinamica*.www.aero.polimi.it.

[39] M. Quadrio *Dispense del Corso di Turbolenza*, www.aero.polimi.it.

[40] OpenCFD, *Open Foam Programmer's Guide*, 2009.

[41] OpenCFD, *Open Foam User Guide*, 2009.

[42] N. Parolini, *Computational Fluid Dynamics for Naval Engineering Problems*, Ph.D Thesis, CMCS ,École Polytechnique Federale de Lausanne, 2004.

[43] H. Jasak, *Error Analysis and Extimation for Finite Volume Method with Applications to Fluid Flow*, Ph.D Thesis, Department of Mechanical Engineering, Imperial College of Science, Technology and Medicine, 1996.

[44] R. Azcueta, *RANSE Simulations For Sailing Yachts Including Dynamic Sinkage and Trim and Unsteady Mptions In Waves*, High Performance Yacht Design Conference, Auckland, 4-6 December 2002.