FOSSEE, IIT Bombay
OpenFOAM Case Study Project
August 3, 2022

Multiphase simulation in OpenFOAM for calculating Ship Resistance

Rishabh Sharma
Indian Institute of Technology Madras

Abstract

This is a comprehensive study to calculate the Ship resistance when sailing in water. The CFD tool used is OpenFOAM which is an opensource and very robust software.The study focuses on the mesh generation, fields initialization and comparison between different solvers and finite volume solutions.All the verification of the resistance calculation is done through the ITTC 1987 empirical formula. Numerical accuracy of under 8% is achieved through the calculations.

1 Introduction

With the advent of Computational Fluid Dynamics Techniques it has become increasingly cheaper to test out new design models for their efficiency and resistance. The expensive towing tank tests to test out new hull models can now be eliminated with cheaper alternatives like CFD to test out even small changes in the models effectively.

OpenFOAM is an opensource and widely used CFD software used in both industry and academia extensively for various computational modeling.The aim of this study is also to prove the power and reliabilty of the opensource software in ship resistance calculations.

We start with first familiarizing ourselves with the governing equations involved with the simulations and turbulence models.We then proceed with the Geometry and Mesh which play a very important role in giving accurate results and if done wrong will blow out the solutions.Then we move on to defining the boundary conditions of the problem which also deals with turbulence modelling which is vital in such a high turbulent flow.Lastly we move onto selecting the finite volume schemes and solutions and experiment with various parameters to get a fairly accurate solution.

2 Governing Equations and Models

The solver we will be using for our simulation is the *interFoam* solver built in OpenFOAM. As we are dealing with simulation between the *interface* of Air and Water, hence the name *interFoam*.

Basis of solving any fluid mechanics problem is solving the Navier-Stokes equation along with other boundary equations. For our case wherein we have an air-water fluid interface, we'll be using Volume of Fluid (VOF) method. One of the key difference in our simulation is the interface between the air and water.The VOF method is a numerical technique to track the interface.

The three equations of importance to us are the Continuity equation(1), Momentum equation(2) and the Interface equation(4).

$$\frac{\delta u_j}{\delta x_j} = 0 \tag{1}$$

$$\frac{\delta(\rho u_i)}{\delta t} + \frac{\delta}{\delta x_j}(\rho u_j u_i) = -\frac{\delta \rho}{\delta x_i} + \frac{\delta}{\delta x_j}(\tau_{ij} + \tau_{ij}) + \rho g_i + f_{\sigma i} \tag{2}$$

Since we are dealing with multiphase simulation, the density equation will change depending on what phase a particular cell is in.

$$\rho = \alpha\rho_1 + (1 - \alpha)\rho_2 \tag{3}$$

The value of α varies between 0 to 1. ρ_1 is water density and ρ_2 is Air density. α takes the value of 1 if cell is inside water and takes value 0 if cell is inside Air. At the *interface* the value of α varies between 0 and 1.

$$\frac{\delta \alpha}{\delta t} + \frac{\delta(\alpha u_j)}{\delta x_j} = 0 \tag{4}$$

Coming to the turbulence modelling we have three parameters which we need to give as input for the solver viz *turbulent kinetic energy (k)*, *turbulence specific dissipation (ω)* and *eddy viscocity (ν_t)*.We'll be using the k-omegaSST turbulent model since it is widely used and tested for such applications.The key variables which helps in deciding the above three parameters are *Turbulent length (T_L)*, *Reynolds number (R_e)*, *Turbulent Intensity (I)*, *kinematic viscosity (ν)* and *Mean velocity (U_m)*

Reynolds number is given by

$$R_n = \frac{vl}{\nu} \tag{5}$$

where v is the velocity of the body (in our case ship/boat), l is the length of the water line, and ν is the kinematic viscosity of water.

A highly turbulent flow results in eddies of various size. The size of these eddies gives an idea about the Turbulent length scale.To approximate the turbulent length we use

$$T_{U_L} = 0.07 R_L$$

where R_L is the relative length in the simulation, in our case the Length of water line (LWL).

Turbulent intensity gives an idea about the variation of the water particle velocity from the mean velocity (velocity of the ship in our case). Since we'll be moving at low froude numbers and have a denser fluid (water) involved, we'll take the Turbulence intensity T_U = 0.5 %.

Since our flow is turbuent and we are having turbulent fluctuations in velocity (u'), Turbulent kinetic energy (k) gives an idea about the kinetic energy per unit mass of the fluctuations.It is numerically given by

$$k = \frac{3}{2}u'^2$$

Coming to specific turbulent dissipation (ω), it gives an idea about the time rate at which the turbulent kinetic energy is converted into thermal internal energy.

To calculate all the above quantaties, there are sophisticated online calculators [1].

3 Simulation Procedure

3.1 Geometry and Mesh

The geometry can be of any ship/boat. The important things to note for a successful mesh and simulation is a closed and well defined geometry. It is also recommended to slice the hull from the main ship body for a simpler, accurate and faster simulation.To check if your geometry (stl or obj file) is correct and **closed** use OpenFOAM's *surfaceCheck* utility. Now

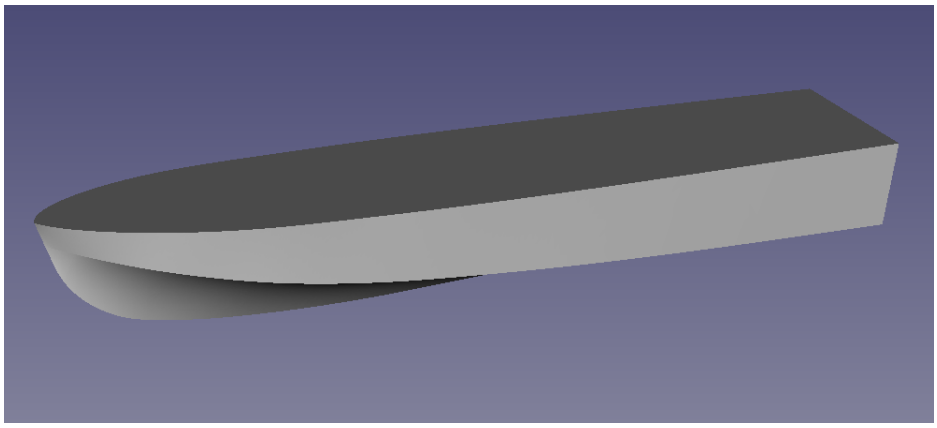
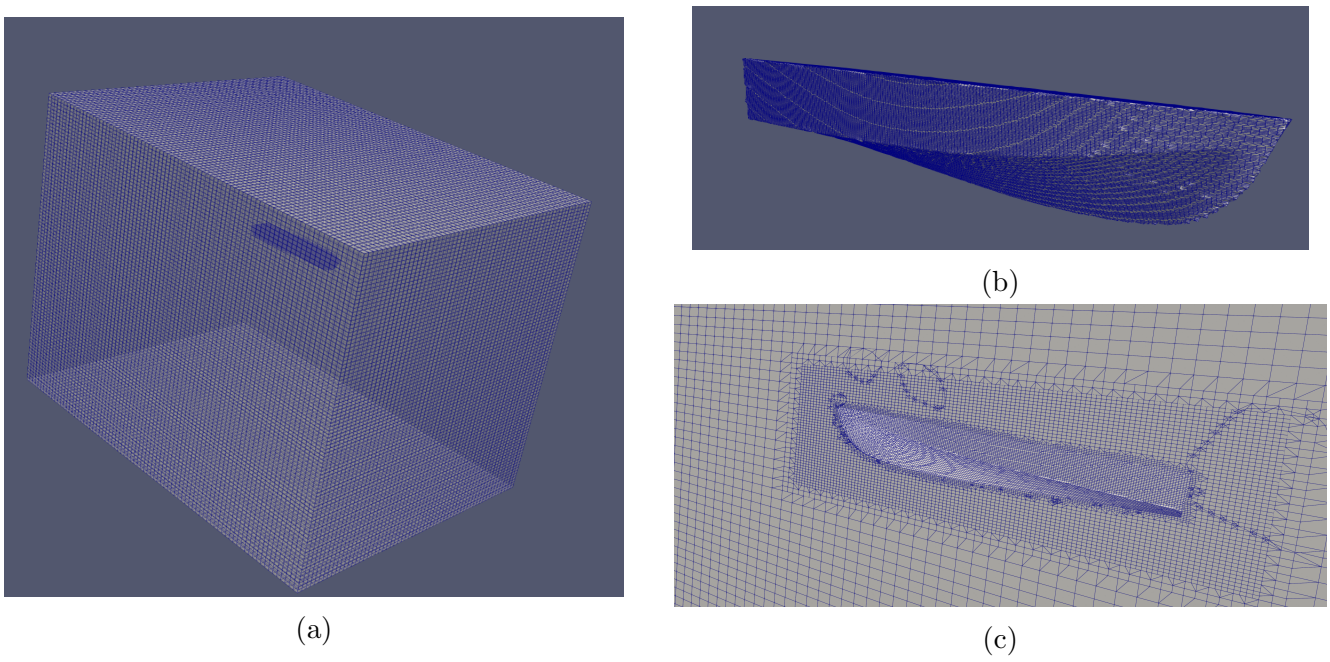


Figure 1: Boat Geometry

that we have our geometry, we proceed to the next important step, ***Meshing***. Infact, this is one of the most crucial steps as the simulation time, simulation result and simulation accuracy depends on the mesh. We will use OpenFOAM's inbuilt utilities *blockMesh* and *snappyHexMesh* as the meshing tools.We use *refinement* blocks in snappyHexMesh for a finer mesh near the boat.This saves us from the extra computation required to refine the whole domain.

One might say that the more refine mesh we have the better the accuracy, which is, quite suprisingly not true. Having a more refined mesh is more computationaly demanding and might even blow up since very low *Courant number* would be required at that region.*Courant* number gives an idea about many cells does the flow pass through per time step. It is recommended to have a low (≤ 0.7) Courant number for a stable and accurate flow, but we will see how we different solvers accomodate for a higher Courant number. Coming back to

our mesh, a mesh cell count around 570K to 2M cells works just fine. We can also add an additional refinement near the interface region to capture the flow near the interface well.



(a) (b) (c)

Figure 2: Mesh, a)The domain mesh b)Boat mesh c)Refinement regions

3.2 Initial and Boundary Conditions

The boundaries of the domain are respectively, the sides, inlet, outlet, bottom and atmosphere (As shown in Fig 3).To complete the solution, we specify the initial velocity, hydrostatic pressure, alpha.water and turbulent properties.

We take the velocity to be 1m/s which gives a Froude number of

$$F_r = 0.2257$$

taking the length to be 2m. The realtive pressure is set to be zero with different conditions at different boundaries (see table). The alpha.water input is set as 0 and finally the turbulent parameters are calculated as discussed in *Section 2*.

Boundary	U	p_rgh	alpha.water
Inlet	fixedValue	fixedFluxPressure	fixedValue, 0
Outlet	outletPhaseMeanVelocity	zeroGradient	variableHeightFlowrate
Side1/Side2	symmetryPlane	symmetryPlane	symmetryPlane
Bottom	symmetryPlane	symmetryPlane	symmetryPlane
Atmosphere	pressureInletOutletVelocity	totalPressure, 0	inletOutlet
Boat	movingWallVelocity	fixedFluxPressure	zeroGradient

Table 1: Boundary conditions

Value	Fr = 0.2257
k	0.0000375
omega	0.04374
nut	5e-07
	nutkWallFunction

Table 2: Turbulent properties

3.3 Solver

As discussed in *section 2* we'll be using *interFoam* for the solver. The main parameters which will affect our results are the *finite volume schemes and solutions*. We will test out distinctly three different time derivatives schemes, different parameters in the PIMPLE solver for pressure and also different divergence schemes (*divSchemes*).

We'll also be using the RAS solver for turbulent solving.

To begin with we use the *localEuler* time derivative scheme.As the name suggests this is a Local-Time-Step solver meaning that the time-step would be adjusted locally to keep the courant number under control which inturn affects the solution.One other advantage of using LTS is that the solution may be more accurate than Global time step and also higher Courant numbers can be used, speeding up the solution.

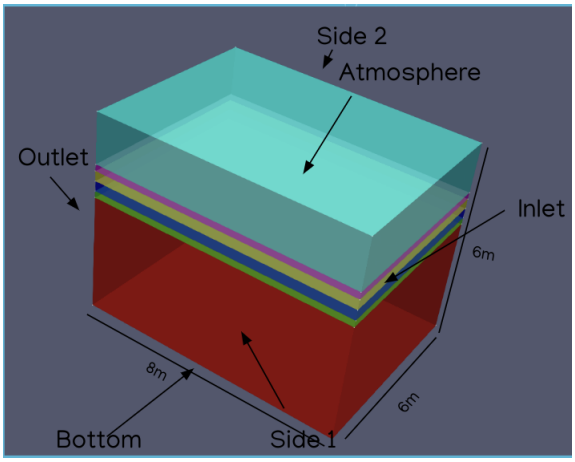


Figure 3: Domain

One thing to note is that before running the simualtion we need to initialize the water field. So instead of water flowing in from the inlet, we set the water to already be in the domain which saves time! To do this we use OpenFOAM's *setFields* utility.

Using this in OpenFOAM v2112 we set the *divSchemes* for alpha as *Gauss interfaceCompression* or *Gauss linear*. Both are tested and perform well, although the former is recommened as it is specially designed for interface problems. Coming to the *fvSolution* we leave the *alpha.water* scheme to be default values found in the *DTCHull* tutorial found in 'OpenFOAM multiphase RAS' tutorial directory.

Coming to the PIMPLE solver (which is combination of the SIMPLE and PISO solver) we have different parameters to experiment with, which affects the solution by a lot.First we see the *momentumPredictor* option which is set to "no".As recommended in [2] it is better to keep this option off for multiphase simulation. However it doesnt affect the solution in the current *divSchemes* and *ddtShemes*, it is found to affect the solution negatively for other schemes.

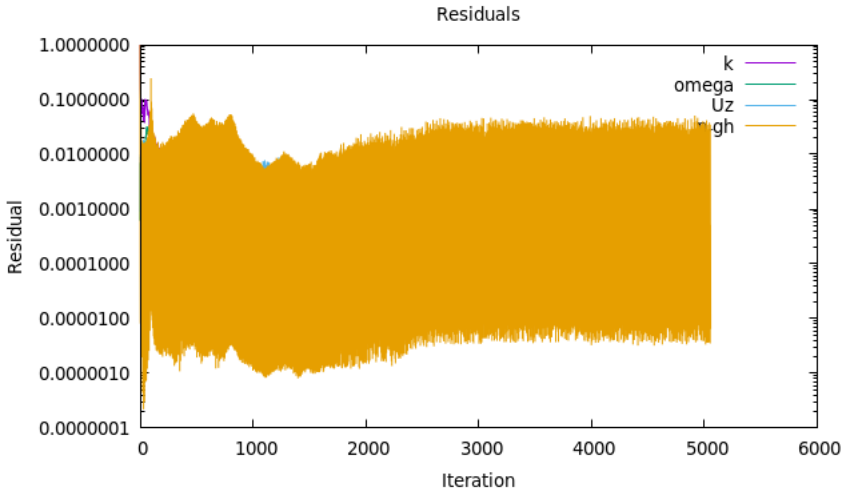


Figure 4: Interface Compression, Local Euler

Next we use the Global Time Step Schemes. We change the time scheme to *Euler* and add Courant number and timeStep controls in the *ControlDict*.As we'll discuss in the next section, this solution runs with quite stability. We also experiment with *crankNicolson* which is a second-order scheme and thus gives more accuracy. Although the parameters have to be decided quite carefully or else the solution blows up.By "blow up" we mean that the alpha (α) values aren't bounded between 0 and 1. If you get an alpha value greater than 1, that would have no physical significance and thus we say the solution "blew up" .It also has to do with the Residuals values (solution diverges).The *crankNicolson* time scheme take in a parameter which ranges from 0 to 1. 0 meaning Euler and 1 meaning completely crankNicolson.For a sweet balance between stability and accuracy we set the parameter to 0.7.Also one more thing to note is that that *crankNicolson* does not support Alpha SubCycling (temporal filter to improve accuracy) thus one needs to set *nAlphaSubCycles* to 1 in the *fvSolutions*.

The above simulations are carried out in OpenFOAM v2112. However, there are some neat and useful functionalities which were introduced from OpenFOAM v8 onwards. Namely

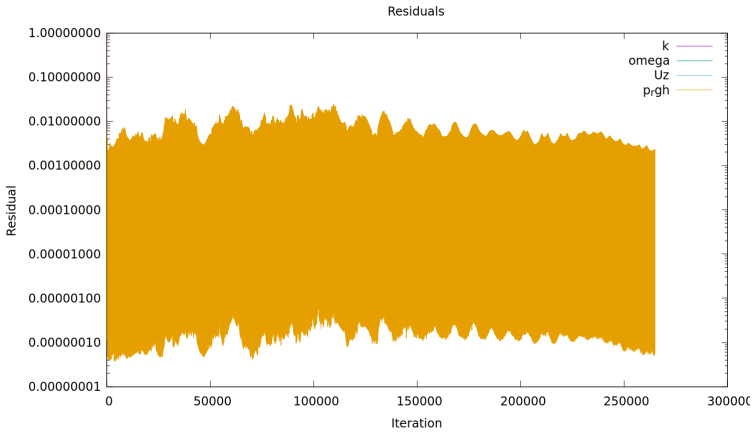


Figure 5: interFace Compression, Global Time Step (Crank)

the PLIC and MPLIC divergence schemes. PLIC stands for *Piecewise Linear Interface Calculation*. What it does is that it carries out a single surface cut on a cell so that the volume fraction (alpha) is bounded in that particular cell. MPLIC (*Multicut PLIC*) cuts the cell by more than one surface which inturn increases stablity at the cost of computation power.

The global time schemes take aprroximately 5 hours to converge on a 80 core parallel computing system, while the MPLIC local time step takes around 16 hours on a 20 core parallel computing system. Also it is evident from Fig 4,5 and 6 that the Residuals in the

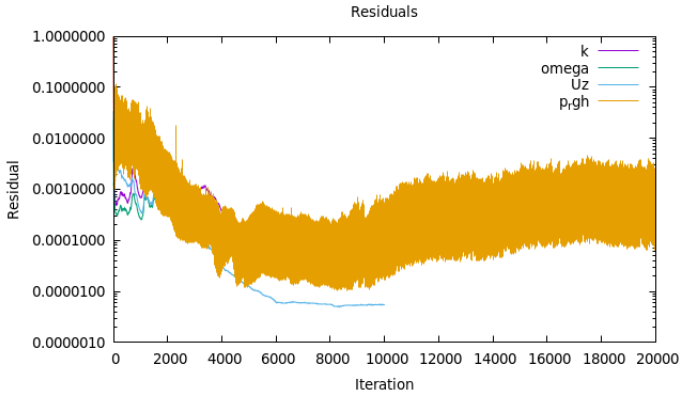


Figure 6: MPLIC, Local Time Step (LocalEuler)

case of MPLIC converges better.

4 Results and Discussions

After the simulation we can now proceed to postProcessing and verify the hull force calculated by the *ITTC 1987* formula

$$C_F = \frac{0.075}{(\log_{10}(R_n) - 2)^2} \tag{6}$$

where R_n is Reynold's number (Equation (5))

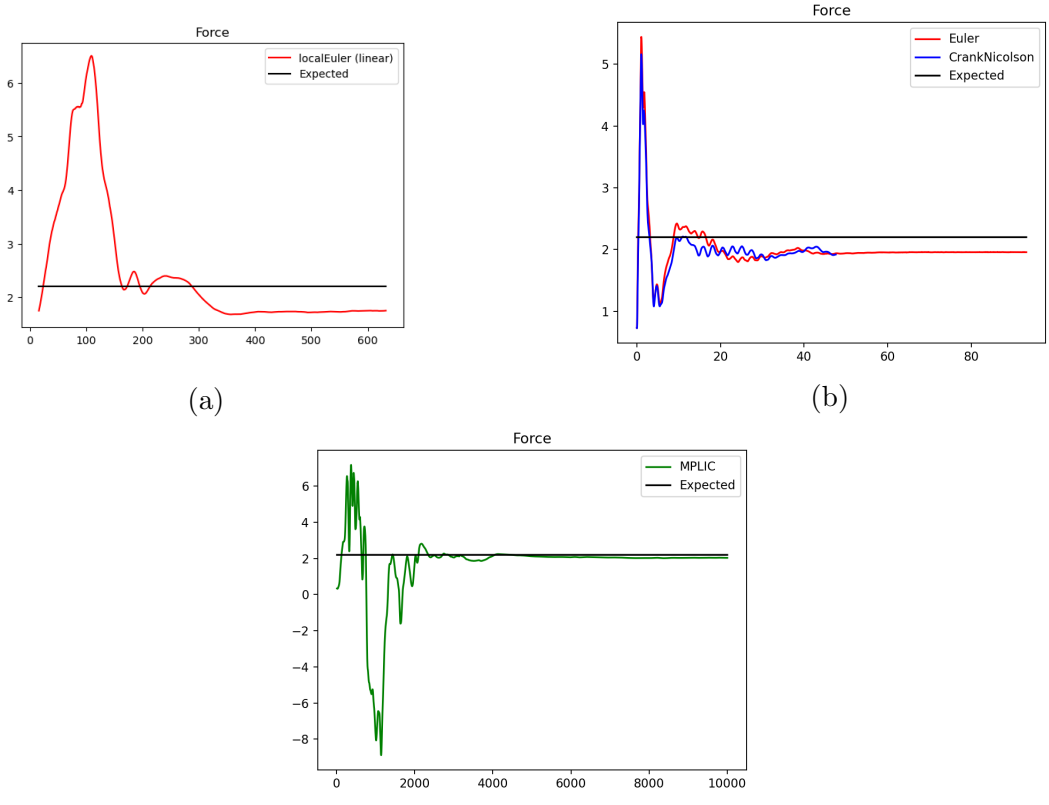
To give a brief idea, a ship's resistance can be divided into different components like air resistance, frictional reistance, wave-making resistance etc. Mainly we consider the Frictional Resistance (R_F) and the Residuary (also known as wave-making R_w) resistance to be the main contributors of the total resistance R_T.

The ITTC 1987 formula gives a rough idea about a flat plate's (in our case ship) frictional resistance.Taking velocity (U) as 1 m/s, the surface area as $1.08m^2$ and the Length of waterline (L_w) as 2m, we get the empirical Frictional resistance based on equation (6) to be **2.2N**.

The plots below (Fig:7 a,b,c)are of the viscous resistance calculated by OpenFOAM during solving. As we can clearly see MPLIC predicts the most accurate result to the expected empirical value. (Table 3).

OpenFOAM	ITTC Formula	% Error
1.5	2.2	**20**
1.94	2.2	**13**
2.035	2.2	**7.5**

Table 3: Result comparison

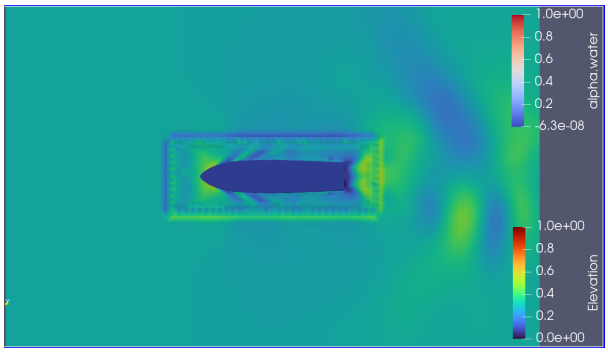


(a) (b)

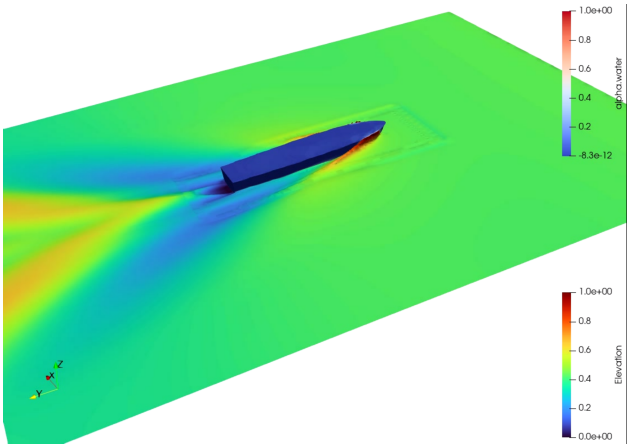
(c)

Figure 7: Force plots

The result above is quite satisfactory since ITTC formula is itself empirical and does not give the accurate practical result. The results can be further improved by using different turbulent modelling techniques currently under research.The visuals of the simulation can also be seen in the Figures below, the kelvin waves which is formed due to the motion of a ship/boat is also vividly seen.



(a) U = 1 m/s



(b) U= 3 m/s (high F_r)

Figure 8: Simulation

References

[1] "Turbulent calculator." `https://www.cfd-online.com/Tools/turbulence.php`.

[2] "Piso algorithm and momentum predictor." `https://doc.cfd.direct/openfoam/user-guide-v6/fvsolution`.

[3] M. T. Shahzad *et al.*, "Ship resistance computations using openfoam," 2017.

[4] L. Axner, J. Gong, A. Chiarini, and L. Mascellaro, "Shape pilot monotricat srl: Hull resistance simulations for an innovative hull using openfoam," *PRACE Partnership for Advanced Computing in Europe*, pp. 1–8, 2014.

[5] P. Voxakis, *Ship hull resistance calculations using CFD methods*. PhD thesis, Massachusetts Institute of Technology, 2012.

[6] O. e. Moctar, V. Shigunov, and T. Zorn, "Duisburg test case: Post-panamax container ship for benchmarking," *Ship Technology Research*, vol. 59, no. 3, pp. 50–64, 2012.

[7] D. S. H. Bustos and R. J. P. Alvarado, "Numerical hull resistance calculation of a catamarán using openfoam," *Ship Science and Technology*, vol. 11, no. 21, pp. 29–39, 2017.