Proceedings of PRADS2016
4^{th}-8^{th} September, 2016
Copenhagen, Denmark

A Computational Environment for Rapid CFD Ship Resistance Analyses

Lutz Kleinsorge[1)], Hannes Lindner[1)], and Robert Bronsart[1)]

[1)]University of Rostock, Germany

Abstract

A reliable estimation of the total resistance in the early design phase becomes of more importance due to the need of a ship design for higher efficiency. With continuously increasing computational power the use of modern CFD methods like RANS play a common role in the resistance prediction. Still these methods, even more precise, are more time-consuming in pre- and post-processing due to their complexity. Hence they have to be soundly integrated into the design and computational environment for an efficient usage and reduction of turnaround times. This paper discusses how RANS methods can be embedded into a design infrastructure. Possibilities to automatize the steps of CAD and computational case preparation including the mesh generation step are introduced and discussed. Special attention is given to the application of open-source software and the discrepancy between automatism and user interaction. As a conclusion an example implementation for CFD ship resistance analyses with OpenFOAM and Numeca Hexpress into an existing computational infrastructure is presented. Based on example CFD resistance calculations, this is discussed according to accuracy in results, user flexibility and turnaround times. The developed procedure proves to be efficient and flexible in performing CFD resistance predictions and is able to support the decision making process in early design.

Keywords

Resistance; RANS; CFD; CAD; KCS; early design; OpenFOAM; ship hull form.

Introduction

Estimating the total resistance in the early design phase becomes more important due to the need of a ship design for higher efficiency. Furthermore the knowledge about flow details is of increasing interest not only in hull form design, but also in the design of appendages like propellers, rudders or energy-saving-devices. Continuously increasing computational power leads to the use of Computational Fluid Dynamics (CFD) methods like Reynolds-Averages-Navier-Stokes-Equations (RANSE) as a common tool in the prediction of the hydrodynamic performance of a vessel. The CFD workshops of Gothenburg, 2010 (Larsson et al., 2010) and Tokio, 2015 show that developed CFD methods are able to achieve good results compared with experimental results. Hence RANS methods nowadays are a common used tool in design departments to support the design process. However their efficiency is limited by a time consuming preparation of the computational model and analysing the results. The International Towing Tank Conference conducted a questionnaire on the difficulties of using CFD, which was answered by 194 persons out of the shipbuilding sector from 30 countries (ITTC, 2011). It showed that besides the accuracy and confidence in results, the mesh generation and the turnaround times of calculations are main limitations and difficulties for a wider use and acceptance of CFD methods. Hence CFD methods can efficiently support the daily design process, if they are better integrated into the pre- and post-processing software environments.

General CFD-Process

The CFD process in ship hydrodynamics is shown in Figure 1. It can be subdivided into three main parts: preprocessing, solving and post-processing. The parts and its subtasks have to be sequentially processed, as they build up on each other.

Pre-Processing

Before starting a CFD-computation the necessary simulation model has to be derived with the help of the design parameters of the ship and transport properties, e.g. density and viscosity of fluid.

In the *Domain preparation* the fluid domain is defined. Therefore the ship hull geometry has to be adapted first. Appendages might be added or removed. Complex geometry features of the hull, that are not in the scope of investigation are simplified, depending on the experience of the engineer. Furthermore the geometry has to be scaled or transformed according to the floating position to be investigated. Additionally boundaries of the fluid domain, e.g. inlet and outlet, need to be generated. As it is essential, the geometry of the fluid domain should be watertight, discontinuities like holes, overlaps, etc. have to

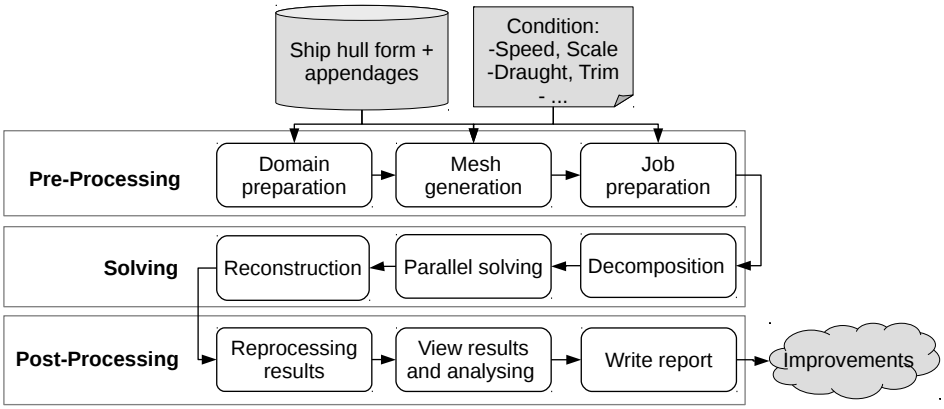


Figure 1. The general CFD Process and its subtasks.

be healed. Requirements by the mesh generation tools related to the geometry have to be also considered during this step. Therefore very often the geometry has to be converted into an appropriate file format to be soundly read and understood. Bronsart et al. (2013) show an automated approach for *domain preparation* with special attention on eliminating geometry inconsistencies. Abt et al. (2012) present a *domain preparation* procedure integrated in an existing CAD-software environment.

The next step *mesh generation* in which the fluid domain is discretised into finite cells finally resulting in a volume mesh. The cells in the generated grid must fulfil a number of quality criteria, otherwise the later computation might result in inaccurate results. Besides this general constraints, it is obvious, that the mesh as the spacial discretisation should be able to correctly resolve the fluid flow. Therefore grid convergence has to be checked and special attention has to be given to the composition of the boundary layer and the free water surface. ITTC (1999) and WS Atkins Consultants (2002) give recommendations on how to generate the mesh with special attention to ship hydrodynamics. A more general guideline is given by Casey and Wintergerste (2000). To conclude mesh generation strongly depends on the the kind of numerical simulation, the geometry itself and the flow phenomena to be investigated.

Finally in the *job preparation* step all boundary conditions, transport properties and solver parameters are defined in order to complete the information needed for calculating the fluid flow. Boundary conditions have to be selected in accordance with the physical behaviour of the flow. General boundary conditions for ship hydrodynamic problems are given in the mentioned guidelines. *Fluid properties* are given by the environmental conditions the ship is simulated in, usually by selecting density and viscosity of the fluid. The *solver parameters* are control properties, e.g. run time control, numerical solution control and schemes to be used. Also the turbulence model has to be selected.

After setting up all properties for the numerical solution successfully, the case is ready to be solved. Usually the *pre-processing* task is the most time consuming process step when performing CFD due to the extensive manual work during domain preparation and mesh generation.

Solving

In its simplest form the *solving* step represents the execution of the CFD-solver in order to compute the numerical solution. This step can be more extended with using High-Performance-Computing (HPC).

To solve the CFD-computation parallel on several CPUs the pre-processed job first has to be *decomposed*. Then it is parallely solved and finally the job has to be *reconstructed*.

When running on a HPC-cluster the described procedure is committed to a *workload manager*, which cares for the execution of the solving process. Of course it needs also job specific information, like execution command, resources to use, running time, etc. After completing the job the results have to be transferred back to the user in order to be analyses.

Post-Processing

In the following *post-processing* step a detailed analyses of the fluid flow is conducted. The residuals of the numerical solution and the forces acting on the hull are plotted over simulation time to assure convergence of the solution. Visualization tools are used to plot the dynamic pressure distribution, streamlines or wave elevation along the hull. As a result possible improvements of the shape of the hull can be derived.

Environment for Rapid Resistance Estimation

Based on the briefly described general CFD steps a procedure to perform rapid resistance estimation is developed.

As the computing environment should be independent of license cost, it was decided to use the open-source CFD-Code *OpenFOAM* Version 2.2.2 (OpenCFD, 2016) for the RANS resistance estimation. *OpenFOAM* beside the numerical solvers includes numerous mesh generation, pre-processing and post-processing tools. Kleinsorge et al. (2011) shows how the *OpenFOAM*-tools can

be used to generate grids for ship resistance estimation automatically. Unfortunately the algorithm cannot reliably generate cells in the boundary layer, which is of major importance for determining the viscous forces. Additionally difficulties in resolving knuckles in the ship hull form exist. Even though the code is continuously improved, the commercial software *Numeca Hexpress* (Numeca International, 2016) was chosen to build the meshes as it overcomes these problems (Bronsart and Kleinsorge, 2011).

For parallel solving of the RANS-equations the reference HPC-infrastructure consists of a Linux-Cluster, which uses the *SLURM* workload manager (SchedMD, 2016) as queue management.

In the post-processing step the visualization of the flow is made with the open-source tool *Paraview* (Kitware Inc., 2016), graphs are plotted with open-source application *gnuplot* (gnuplot, 2016).

To build an environment for rapid resistance estimations the presented applications have to be successfully integrated into the general CFD-process. Therefore interfaces are needed to control and manipulate them efficiently. As *Numeca Hexpress* and *Paraview* provide an application programming interface (API) based on the programming language *Python* (Python Software Foundation, 2016) it was decided to develop the whole environment in *Python*-Code. To read and write *OpenFOAM* files a parser from the python-library *PyFoam* (Gschaider, 2016) is used.

Figure 2 shows the developed environment for rapid ship resistance analyses. The domain preparation and input parameter definition for mesh-generation are strongly depending on each other, as it is essential to know the topology of the domain for defining meshing parameters. Therefore the pre-processing step of domain preparation is still left to the user.

The following mesh generation and job preparation runs automatically, controlled by the input parameters and is explained in the next subsections in detail. At the end a completely prepared job is generated. The submission of a job to the HPC workload manager is done manually by the user. This can be seen as a final quality check before starting the solving in order to eliminate unwanted jobs from allocating resources on the HPC.

After submitting the job to the SLURM workload manager, it takes care for decomposition, execution of solver, reconstruction and reprocessing of results. The plotting of residuals and forces is also executed after each computation. Thereby the convergence of solutions can be directly controlled after solving.

The reprocessed results are viewed with the help of the visualization tool *Paraview*. Analysing details of the flow with visualisation tools is performed manually as it is highly depending on the user's experience and needs. To save time in analysing the flow the API in *Paraview* routines are continuously developed.

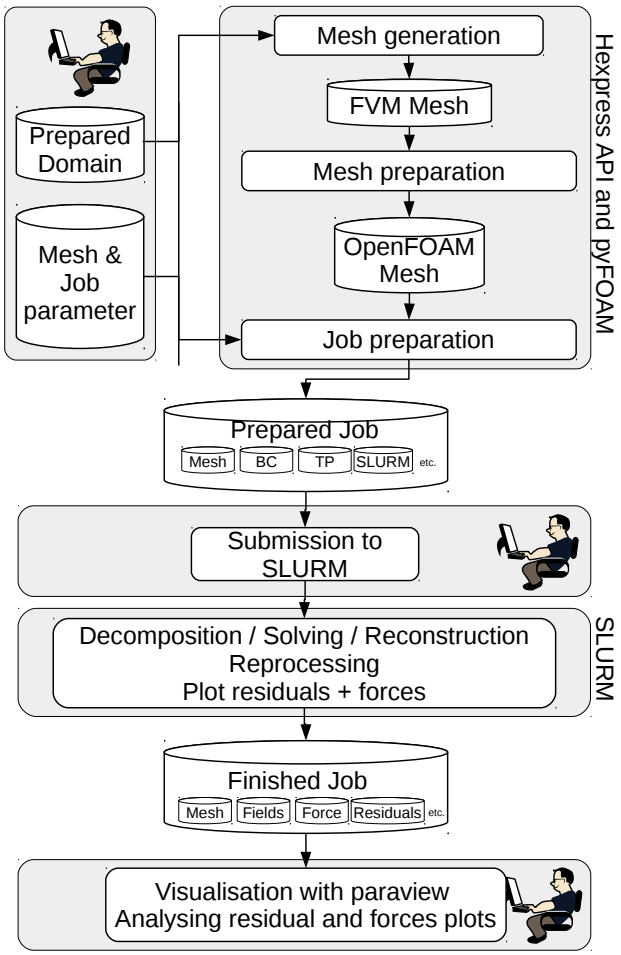


Figure 2. Derived environment for rapid ship resistance analyses

Domain Preparation and Input Parameters

As described above the domain preparation is left to the user, for an example see Figure 4. The domain has to be prepared in a special *Numeca Hexpress* file format called *.dom* which is essential for later meshing. It is generated from Standard Tesselation Language *.stl*-file-format and can handle the topology of the geometry. Details on domain preparation and how to use the *.dom*-format are described in Numeca (2013).

Input parameters to control the meshing process are defined by using the *Python* dictionary data structure. In total six dictionaries form the input for the rapid ship resistance estimation. The parameters are shown in Figure 5. The `GenPar` dictionary defines general information for meshing and job preparation. Five additional dictionaries exist with parameters defining the meshing, namely:

- refinement of faces: `faces`
- refinement of boxes inside the domain: `box`
- refinement of edges: `edges`
- control of snapping cells on edges: `snapp`
- viscous layer insertion: `viscLay`

Within the dictionaries meshing parameters are assigned to the faces or edges by its topology entity number, which is derived by a special developed tool. As the domain is

prepared by the user, he/she can easily define meshing parameters manually.

Once the domain is generated and meshing parameters are defined the process can be easily automated for investigation of hull form variations in an optimization, as long as the domain topology is constant and only the geometry of surfaces change.

Mesh generation

Mesh generation using *Numeca Hexpress* is following an octree-approach to create unstructured hexahedral grids. Briefly the methods first generates an initial coarse grid which in the following is refined at the boundaries by subdividing initial cells. Cells outside the domain are deleted and the remaining cells are snapped on the domain boundaries. Then a mesh optimization is started to improve the quality of the cells. Finally viscous layers are inserted on designated boundaries, see Numeca (2013)

An automated process for the meshing procedure is developed. As a result each meshing step is implemented in a method of a *python*-class to control the meshing. As the meshing steps have to run sequentially one after another, once an object as instance of the class is defined meshing is started.

Before preparing the initial mesh it is checked whether the mesh is generated for double body or free water surface flow. Due to the refinement of free water surface and the presence of parts of the body above the water, the latter task needs a special treatment. If the keyword `freeSurface` is not listed in general parameters dictionary the meshing will be performed for double body flow.

The initial mesh is then generated by subdividing the domain in vertical direction using the parameter `globRef`. The subdivision in longitudinal direction is calculated by using the domain dimensions and the parameter `globAspect` as well as `globRef` to assure the aspect ratio of cells is met. In lateral direction the subdivision of cells is calculated using an aspect ratio of one. Hence the initial mesh consists of cells that are stretched in longitudinal direction. As the subdivision routine accepts only integer values, the aspect ratio might slightly differ from the value defined in `globAspect`. For free surface meshing the parameter `globRef` defines the subdivision of the underwater domain, hence the subdivision for the complete domain in vertical direction is corrected due to the part above the water line. This is introduced in order to have the same underwater initial mesh cell sizes, independent of the kind of simulation.

Afterwards the initial mesh is adopted to the domain. Therefore the dictionaries for face, box and edges refinement are used. Nearly all meshing parameters available in *Numeca Hexpress* can be defined in the dictionaries. Anyhow, for most of them defaults are defined according to the authors experience. Only parameters to be explicitly changed have to be written into the dictionaries. The procedure is mainly similar to the procedure when using the GUI of *Numeca Hexpress*, but automated. Generating free surface meshes an internal surface is inserted into the domain. Contrary to boundary faces cells along internal surfaces can be anisotropically refined. Therefore refinement of cells in longitudinal direction is calculated based on radiated wave length of the ship, which is determined with speed and length of the ship. Refinement is calculated by division of wave length trough `CpWave`. For the height of cells the refinement is directly given by the value of keyword `CzWave`. In lateral direction the refinement is restrained by the global aspect ratio.

Snapping of refined cells on the domain is performed according the snapping dictionaries. Again, all snapping parameters available can be assigned to the edges of the domain. By default it is assigned in such a way that cells should be snapped to all available edges. Therefore the user only has to consider special treatment of edges when defining the dictionary. Afterwards a mesh optimization step is performed. Parameters are automatically defined based on the authors experience and build a compromise between computing time and quality.

In the last mesh generation step viscous layers have to be inserted along the faces representing the ship hull. Therefore from the viscous layer insertion dictionary the faces for inserting a layer, its y^+ value and the number of boundary layer cells are read. Based on ship's speed the height of the first cell next to the faces is calculated. If the number of boundary layer cells are not given the number of cells is computed using the cell thickness of refined cells outside the boundary layer. Again mesh optimization is performed to assure a good quality mesh.

The final mesh is exported into *OpenFOAM* native mesh format *polyMesh*. Additional files and folders are provided to set up an initial *OpenFOAM*-Job which is essential for the subsequent steps.

Before finally preparing the job for computation the generated *OpenFOAM*-mesh is further adjusted in order to meet requirements of the job preparation step. In the mesh by default each face of the domain builds a patch. To simplify the mesh, patches of the ship hull form are combined using the *OpenFOAM*-tool *createPatch*. Additionally the exported mesh topology is optimized for computation by using the tool *renumberMesh*. Finally the quality of the mesh is checked using the tool *checkMesh*. Its result is written together with a detailed description of meshing into a log-file, so that the procedure can be reproduced and analysed.

Job Preparation

The initial OpenFOAM job is handed over to the *job preparation* task. The following developments are combined in a *job preparation* class. The class consists of several methods to perform each sub task. Contrary to the mesh generation task, where the procedure is started with building an instance of the class, here the deloped methods of the class have to be called separately. This is more flexible as it provides the possibility to use the class also outside of the developed procedure for reconfiguration of already prepared jobs. The job preparation class is called at the end of meshing class within the API of *Numeca Hexpress*.

```
#load class for job preparation
from preProcess import preProcess

#instance of class
process=preProcess(solvername, cell-of-mesh,  v,  LPP,  transportProperties)
#call of methods
process.setFieldU()
process.setFieldp()
process.setFieldk(0.001)
process.setFieldomega(0.001)
process.setFieldnut('nutUWallFunction')
process.setFieldalpha1()
process.switchTurbulence('on')
process.setFvSchemes()
process.setControlDict(endTime, writeTime,  1, 1,  ['ship'])
process.writeSlurmJob(job-name, path-to-job-folder, email-of-user)
```

Figure 3. Script for job preparation task (*Python*)

The velocity and length of the ship, fluid properties, the specific *OpenFOAM* solver name and the mesh size are transferred. The job preparation can be performed for the following solvers:

- *simpleFoam*: for steady-state double body computation
- *LTSInterFoam*: for steady-state free surface computation
- *interFoam*: for transient free surface computation
- *interDyMFoam*: for transient free surface computation including free trim and sinkage of ship

If the generated mesh is generated for double body flow computations, the solver *simpleFoam* is used, otherwise *LTSInterFoam* is applied. The use of transient solvers is delegated to special methods that switch steady state job properties to transient properties as this procedure improves the stability of a transient run. Overall more than 20 methods are developed to manipulate and prepare the *OpenFOAM*-job with the help of this class.

Figure 3 shows how the methods of job preparation class are called within the developed environment for job configuration for a free surface computation. Italic written parameters are variables that have to be defined.

First boundary conditions for the velocity *U*, the pressure *p*, the volume fraction *alpha1* and the turbulence quantities *k* and *omega* are set depending on solver by calling the `setField`-methods. Next the numerical schemes and solution control are set according to the solver.

The properties to control the computation are also set solver specific. Additionally are defines properties for writing intermediate results, stopping the computation at a certain time step and logging forces on patches.

In the next steps the script for submission to the workload manager *SLURM* is written. A decomposition of the job is selected automatically based on mesh size and solver type. Therefore preliminary performance tests have been performed on the cluster. Besides resource management properties, the methods add the execution commands of *OpenFOAM*-tools for decomposition, solving, reconstruction and reprocessing the results to the submission script.

Running through the whole process the job is prepared for manual submission to the HPC-cluster. For running optimizations this manual step can be easily automated by adding commands for copying and submission of jobs to the HPC-cluster.

Post-Processing

Post-Processing is performed in two alternative ways: The resistance of the ship and the residuals of the computation are automatically supplied to the user as plotted graphs. Therefore a *Python* class is developed that analyses the log-file of the computation and filters for the initial residual of each field quantity. The extracted data is saved and then plotted with *gnuplot* as function of simulation time / iteration. Additionally the total resistance force and its components are filtered and saved in order to be plotted also. In addition a mean force and its maximum deviation is calculated to give a fast feedback on convergence of resistance data. For detail flow analyses scripted processes are developed to directly visualise the flow. Using the API of *Paraview* helps to automate this process. Within the developed environment automated scripts for comparison of wave elevation between two computations, for extracting wave elevation along the hull and for comparison of dynamic pressure distributions and wall shear stresses between two computations are developed. These pre-defined scripts can be directly selected from the GUI tool bar in *Paraview*.

Case Study: KCS-Modelscale Resistance

The developed procedure is used to calculate the calm water resistance of the KRISO Container Ship model (KCS). The KCS is a well known computational benchmark case for numerical resistance and propulsion predictions. KCS is a panamax container ship with $LPP = 230\,m$, $B = 32.2\,m$ and a draught at design condition of $T = 10.8\,m$. The ship is fitted with a bulbous bow and a transom stern. Model resistance data for an even keel condition at design speed $Fr = 0.26$ is given by Kim et al. (2001). Furthermore in the proceedings of the Gothenburg CFD-Workshop experimental resistance values for six speeds ranging from $Fr = 0.11$ to 0.282 with dynamic trim and sinkage are given. All experimental data are for model scale $\lambda = 31.6$, no full scale experimental data are available.

Figure 4 shows the overall domain for free surface computations. The ship is split into six patches: transom, deck, above water line, running, midship and entrance. Edges between the faces are depicted. Global, face and edge parameters are defined manually according to the described procedure in *Python*-dictionary data structure as shown in Figure 5. In order to perform the pre-processing for several speeds, the procedure of preprocessing is performed in a loop, for each run a new ship speed is transferred in the global parameter `GlobPar` dictionary. Jobs for computation of ship resistance are automatically generated and stored under the given relative path for each velocity. The prepared jobs can be viewed before manual submission to the HPC-cluster workload manager.

This procedure is used for a mesh refinement study. Therefore for each speed the refinement values of faces `running-entrance` and `midship` as well as box `bow` and `stern` is increased. This results in meshes of

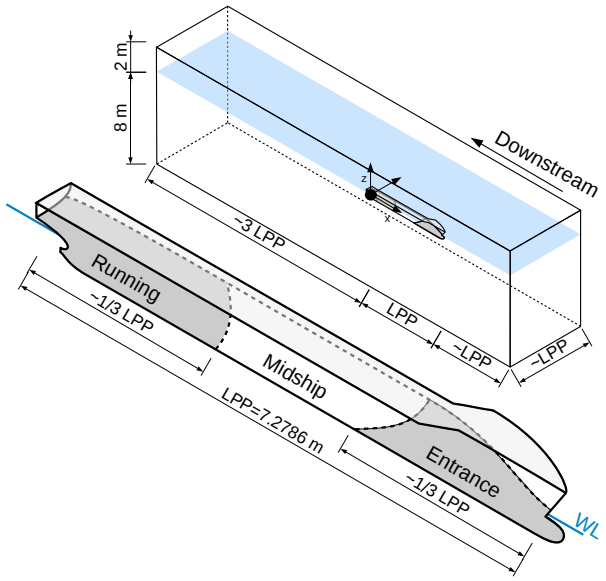


Figure 4. Free surface computation domain of KCS model

```
GenPar = {                              #Definition of global parameters dictionary
         'dom' : 'project/kcs.dom',     #Name and path to domain file
         'v': 2.3795,                   #Speed of ship [m/s]
         'LPP': 7.2786,                 #Length of ship [m]
         'globRef' : 5,                 #Subdivision of initial mesh in vertical direction
         'globAspect': 1.5,             #Aspect ration of cells in initial mesh

         'freeSurface':   {             #Free surface refinement (if necessary)
                          'CpWave': 5,      #Cells per wave length
                          'CzWater': 0.005  #Height of cell on free surface
                          },
         'transProp':     {             #Properties of Fluid
                          'visc': 1.1419e-06,   #Viskosity [m²/s]
                          'rho':1000            #Density [kg/m³]
                          },
         }

Box =    {
         'bow' : {'ref' : 4, 'dim' : [ 5.5, 0, -0.4, 8, 0.6, 0.05]},
         'stern': {'ref' : 4, 'dim' : [-0.3, 0, -0.4, 2, 0.6, 0.05]}
         }

Faces =  {
         'deck-overwater-transom' : {'faces' : [0, 7, 10], 'ref' : 3, ..., 'aspect' : 5, 'trim': True},
         'running-entrance' : {'faces' : [13, 14, 8], 'ref' : 4, ..., 'aspect' : 10, 'trim' : True},
         'midship' : {'faces' : [15], 'ref' : 3, 'aspect' : 20, ..., 'trim' : True},
         'in-out-top-bottom-farps-sym' : {'faces' : [1, 2, 3, 4, 5, 6], 'trim' : False},
         }
Edges =  {
         3 : [0, 18, 10],
         6: [6, 29, 5,7, 8, 25]
         }
Snapp =  {
         'snap': {0 : '', 1 : [33, 32, 31, 28, 27], 2 : [0, 10, 18]},
         'buffer' : {0 : '', 1 : [1, 29, 5, 8, 25]}
         }
ViscLay = {
         'faces' : [8,13, 14, 15],
         'yPlus' : 100,

         'bL' : ''}
```

Figure 5. Dictionaries for meshing and job preparation of free surface domain

approx 250 thousand to 4.5 million cells for design speed $Fr = 0.26$ for coarse and fine mesh respectively.

Figure 6 shows the generated coarse meshes for the lowest and design speed of KCS ship model. As can be seen in detail, the boundary layer is meshed according to the specifications, resulting in a thinner first cell thickness for higher speeds. The chosen wall distance for the generated meshes is $y^+ = 100$, so that a wall function implementation is used in boundary conditions of the ship. All created grids have a sufficient quality as only very few skewed and non-orthogonal cells exist which does not influence the stability and the result of the computation.

Figure 7 shows the residuals of finest mesh at $Fr = 0.26$. Figure 8 shows the forces after completing the computation for the same mesh. The mean forces are calculated based on the last 1000 performed iterations. The maximum deviation of mean forces is also displayed.

The total ship resistance coefficient c_T and its components viscous c_V and pressure coefficients c_P are computed. Using the developed environment meshing parameters are systematically varied. The converged mesh independent solutions are compared with measured ITTC '57 resistance coefficients in Figure 9. Additionally to the CFD results for even keel (fixed condition), results with dynamic trim and sinkage are plotted for $Fr = 0.11$, $Fr = 0.227$ and $Fr = 0.282$.

Even though the calculated total resistance c_T is at most speeds in good agreement with the experiments, the computed even keel fixed condition at design speed is not fully met and has about 6% error. Consequently for calculations with dynamic trim and sinkage the error is smaller compared to the experiments, especially with increasing speed. Comparing the viscous resistance c_V with common ship correlation lines it can be seen that the gradient of the computed $c_V = f(R_e)$ curve is not in agreement with experiments. For dynamic conditions this can be due to the increased wetted surface with sinkage and trim. The calculated pressure resistance c_P fits with the residuary resistance c_R of experiments.

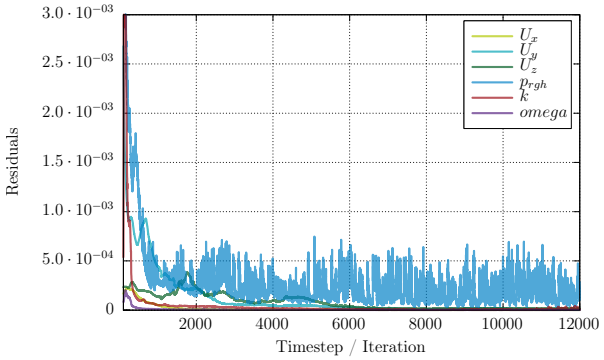
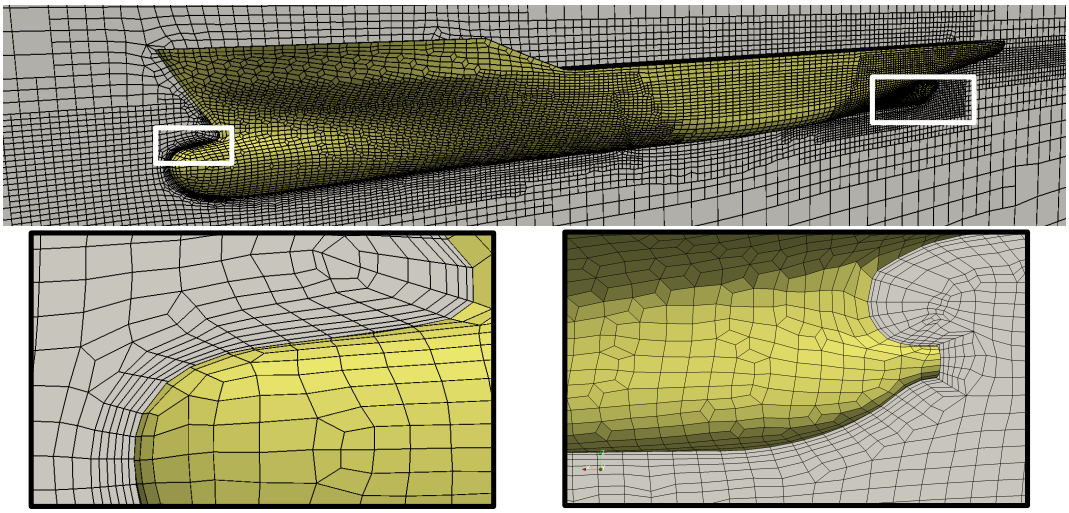
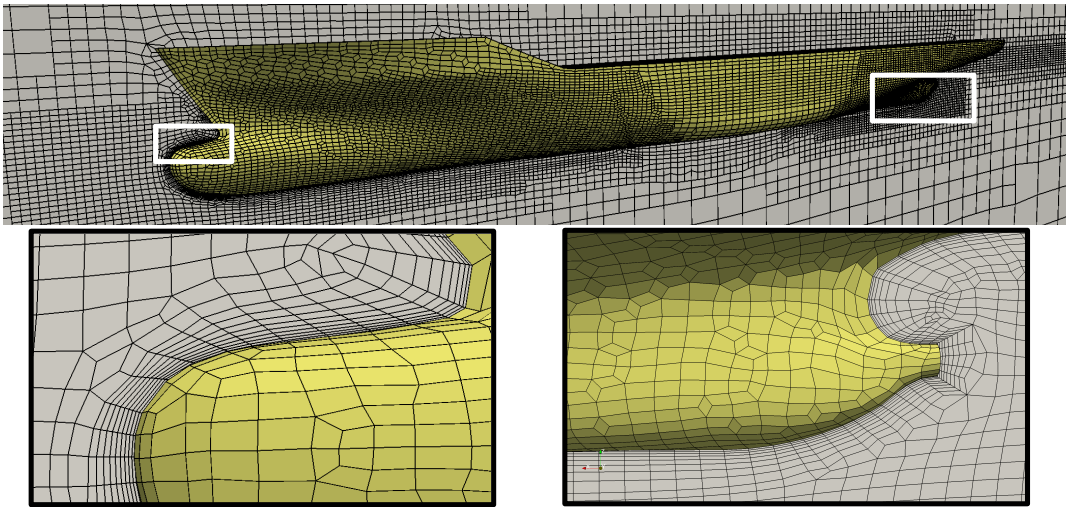


Figure 7. Automatically plotted residuals for fine mesh caluclations at $Fr = 0.26$

The case study shows that within the developed environment the automated mesh generation delivers good quality grids, also with varying mesh parameters. The job preparation task sets the appropriate boundary conditions and solver parameter for all the jobs, which reduces significantly editing errors compared to a manual preparation by the user. Therefore the quality of the whole process is substantially increased. Due to the automated process, the turnaround time of performing a resistance analyses is decreased, especially when a variation of conditions are to be investigated using the same domain. Nevertheless the environment is flexible in usage and easily adaptable due to its object oriented approach. It is shown, that the developed environment is able to produce sound resistance results for free surface RANS computations



(a) Lowest speed $Fr = 0.11$



(b) Design speed $Fr = 0.26$

Figure 6. Coarse meshes for free surface computations

with the KCS model at different speeds.

Conclusion

In this paper an environment for rapid ship resistance analyses based on the tools *Numeca Hexpress* and *OpenFOAM* is presented. The workbench is build with help of the programming language *Python* and supports the guided and automated handling of all CFD process steps. The developed automated procedure for pre-processing, solving and post-processing is described. Due to the complexity of preparing ship hull geometries, it was decided to exclude this step from the environment.

A case study is performed to show how the environment supports a fast and reliable resistance analysis. Therefore the KCS model is computed at six different speeds. For each speed a systematic variation of meshing parameters has been performed. The overall result of the study is presented. It is in good agreement with experimental ship resistance data, especially when computations with dynamic trim and sinkage are performed. The case study reveals that the developed environment significantly decreases turnaround times when performing many computations with the same input geometry. A faultless process of resistance analyses is guaranteed.

For a further improvement of turnaround times the environment should be extended to use input geometries that are prepared by CAD systems automatically. Also methods to used the environment a ship hull form optimisation should be added. Therefore also the integration of the HPC infrastructure has to be improved and further automated.

References

Abt, C., Bergmann, C., and Harries, S. (2012). Domain preparation for ranse simulations about hull and appendage assemblies. *Ship Technology Research*, 59(3):24–37.

Bronsart, R., Edessa, D., and Kleinsorge, L. (2013). Automatic Pre-Mesh CAD Data Repairing. *International*

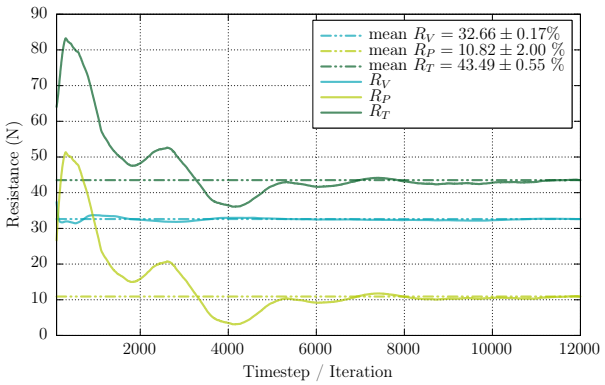


Figure 8. Automatically plotted forces for fine mesh calculations at $Fr = 0.26$

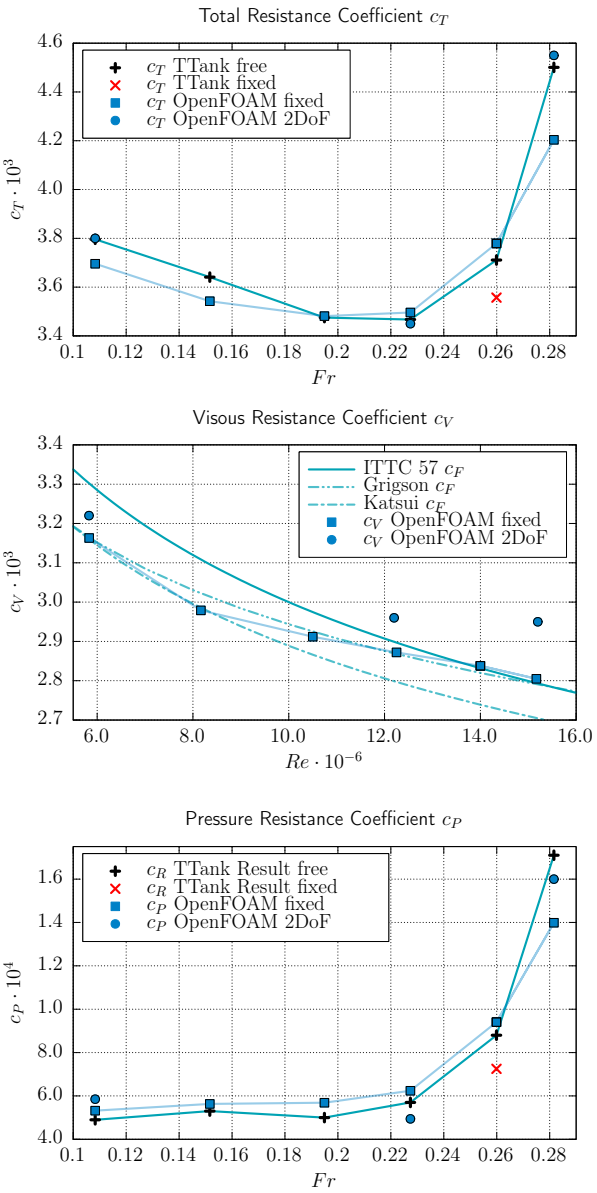


Figure 9. Calculated resistance coefficients c_T, c_V and c_P of KCS model

Journal of Mechanical Engineering and Applications, Vol. 1(No. 1).

Bronsart, R. and Kleinsorge, L. (2011). Integration of CFD Grid Generation Tools into the Ship Design Process. In *Proceedings of 1st IntNam*, pages 241–251, Istanbul, Turkey.

Casey, M. and Wintergerste, T. (2000). *ERCOFTAC Best Practice Guidelines*.

gnuplot (2016). gnuplot - command-line driven graphing utility. Website: `http://www.gnuplot.info`.

Gschaider, B. (2016). pyFoam - Python Utilities for OpenFOAM. Website: `https://pypi.python.org/pypi/PyFoam`.

ITTC (1999). Practical Guidelines for Ship CFD Applications (7.5-03-02-03).

ITTC (2011). The Specialist Committee on Computational Fluid Dynamics. In *Proceedings of 26th International Towing Tank Conference*, volume II, pages 337–377, Rio de Janeiro, Brazil.

Kim, W. J., Van, S. H., and Kim, D. H. (2001). Measurement of flows around modern commercial ship models. *Experiments in Fluids*, 31(5):567–578.

Kitware Inc. (2016). Paraview - visualization application. Website: `http://www.paraview.org`.

Kleinsorge, L., Kornev, N., Bronsart, R., and Hannker, B. (2011). Openfoams CFD Workbench and its Integration into the Initial Design Process. In *Proceedings of ICCAS*, pages 1–8, Trieste, Italy.

Larsson, L., Stern, F., and Visonneau, M. (2010). Gothenburg 2010, A Workshop on Numerical Ship Hydrodynamics. Report, Chalmers University of Technology.

Numeca (2013). User Manual Hexpress.

Numeca International (2016). Hexpress - Unstructured Full-Hexahedral Meshing. Website: `http://www.numeca.com`.

OpenCFD (2016). OpenFOAM - The open source CFD toolbox. Website: `http://www.openfoam.com`.

Python Software Foundation (2016). Python Programming Language. Website: `https://www.python.org`.

SchedMD (2016). SLURM - Slurm workload manager software. Website: `http://slurm.schedmd.com/`.

WS Atkins Consultants (2002). Best Practice Guidelines for Marine Applications of Computational Fluid Dynamics. `https://pronet.atkinsglobal.com/marnet/guidelines/guide.html`.