A Tool for the Automated Development of a Boundary Representation for the Generation of Volume Grids for CFD Calculations of Ships Flows

Bachelor Thesis

Vasil Yordanov

Technical University Varna
Faculty of Shipbuilding

University of Rostock
Chair Shipbuilding

01 July 2011

Version	Date	Comment
1.0	01 July 2011	Final version

Statement of Originality

This thesis contains no material that has been accepted for the award of any other degree or diploma in any other university. To the best of my knowledge, this thesis contains no material previously published or written by another person, except where reference is made in the text.

Place, Date Signature

Vasil Yordanov
University of Rostock
Albert-Einstein-Straße 2
D-18059 Rostock

Tel: + 49 381 498 9270
FAX: + 49 381 498 9272
email: `vasil.yordanov@uni-rostock.de`

This paper was created with the text formatting system LaTeX.

A Tool for the Automated Development of a Boundary Representation for the Generation of Volume Grids for CFD Calculations of Ships Flows

Bachelor thesis

Mr cand.ing. Vasil Yordanov
Matriculation Number: 211100040

The efficient generation of volume grids for RANS calculations highly depends on the defintion of the boundary domain. The mesh generator HEXPRESS uses geometries based on the Surface Tesselation Format (STL) as a bounding box for the discretisation of the fluid domain. Therefore the ship hull and the additional boundary faces (e.g. inlet, outlet and symmetry plane) have to be created as tesselated surfaces. Often the ships hull is provided in STL-Format, so there is the need to create the remaining boundaries of the domain in a tesselated form.

The aim of this thesis is to implement algorithms for the automated generation of boundary faces of a fluid domain, which can be used by HEXPRESS for the discretisation of the corresponding volume.

The following workpackages should be processed:

1. Literature survey about volume-mesh generation, computational geometry modelling, tesselated surfaces, the implementation of HEXPRESS and OpenFOAM into the ship design process.
2. Implementation of algorithms for the creation of tesselated boundary faces
3. Implementation of algorithms for boolean operations with tesselated surfaces (e.g. for cutting the ships profile out of the symmetry plane), adaptation of the refinement of the tesselated surface.
4. Implementation of algorithms for checking the quality of the boundary representation of the domain (e.g. checking that the domain is closed)
5. Testing the developed methods by creating boundary representation of the domain for some example ships.

Supervisor: Prof. Dr-Ing Robert Bronsart

Begin: 15. April 2011 Submission deadline: 15. August 2011

Abstract

For the realization of Computational Fluid Dynamic calculation is required a properly defined fluid domain to be provided. A fluid domain with no overlaps or holes in its boundary surfaces.

Usually in ship hydrodynamics the fluid domain boundaries include complex geometric shapes which normally are the object of interest for the Computational Fluid Dynamics calculation.

The manual definition of the boundary surfaces geometry and their meshing depends on using of Computer-Aided Design software where comprehensive user intervention is required.

This Bachelor Thesis consists of a C++ implementation of algorithms for geometry definition and preparation of the boundary surfaces for further Computational Fluid Dynamics calculations in order to make this process more used-independent.

Keywords: Computational Fluid Dynamics, Mesh Generation, Mesh Boolean Operations, Computational Geometry

Acknowledgments

I want to thank to Martins Brics and Oskars Rubenis for their help in finding memory leaks in the C++ code. I also want to express my sincere gratitude to Iliya Hadzhidimov for all the provided information and for his further assistance about the ERASMUS program. I would also want to thank to Pencho Penchev, Stefan Kyulevcheliev and Petyr Georgiev for inspired talks and invigorating discussions. Their enthusiasm and encouragement sustained me through the final stages.

Thank you all.

Contents

List of Figures

List of Tables

Abbreviations

CAM	Computer-Aided Manufacturing: the usage of computer software for controlling machine tools and related machinery in the manufacturing of work-pieces.
STL	Stereo Triangulation Lithography: geometry file format created by 3D Systems, which is widely used for rapid prototyping and CAM
ASCII	American Standard Code for Information Interchange: character-encoding scheme based on the ordering of the English alphabet
CFD	Computational fluid dynamics: is a branch of fluid mechanics that uses numerical methods and algorithms to solve and analyze problems that involve fluid flows. Computers are used to perform the calculations required to simulate the interaction of liquids and gases with surfaces defined by boundary conditions.
RANS	Reynolds-averaged Navier-Stokes equations: are time-averaged equations of motion for fluid flow.

Chapter 1

Introduction

1.1 Relevance of the topic

Very wide range of software packages for geometry modeling, geometry manipulation, mesh generation and mesh manipulation exists. These are so called CAD software packages. Most of them can be applied in almost all fields of science. They have unlimited capabilities for working with geometry shapes so only user skills and experience can affect the final results.

In some fields of applied science and technology are implemented special CAD systems in order to deal with the native for that filed problems and specifics. Because of the big competition in industrial software developing, all the released products are at once very specialized and with wide range of capabilities. Working with such products is assumed as a serious craft. There is not a option for single click solution to be made.

In the field of Naval Architecture a lot of more specific software solutions exist. Most of them are compound software packages which include even more specialized modules. Because of this they provide modules for Stability Analysis, Hull Design, Seakeeping, Statistical Resistance Prediction, Production Management etc. But when it comes to physical simulation the use of another type of software is required.

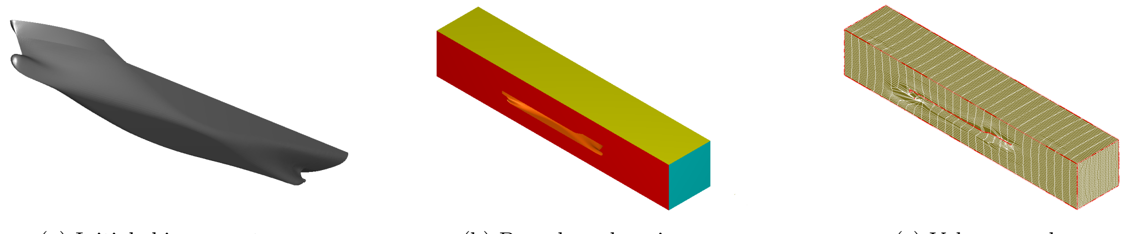

(a) Initial ship geometry (b) Boundary domain (c) Volume mesh

Figure 1.1: Process of CFD preprocessing

For doing so it is necessary to transfer the hull, see Figure 1.1a, of the ship from the software specialized in Naval Architecture to another one specialized in physics simulation. For the realization of the simulation itself, it is necessary the geometry of the domain of the investigation (the ship and surrounding water) to be specified, see Figure 1.1b. But this operation can not be executed in the simulation software. That is why again a CAD software has to be involved. In most cases this domain has shape of a “box” which should contain all the water and the part of the ship which has to be tested. And finally another specialized in mesh generation software has to be used in order to produce the actual computational mesh of finite element for the further CFD simulation, see Figure 1.1c

Due to all these different softwares which are used for the preparation of one CFD simulation, the need for automation or at least simplification of this process is felt.

1.2 Research topics

Undoubtedly the field of this thesis is entirely in the scope of Applied Linear Algebra and Analytical Geometry. This is so because of the nature of meshes themselves. They are built only from planar triangles or other planar polygons where no curved elements exist. The basic research topics which appear are:

- An investigation over all common techniques for working with surface meshes.
- An investigation about the common techniques for making a CFD simulation.
- Implementation of geometry algorithms and structures as a C++ functions or classes for achieving the wanted results.

1.3 Structure of the thesis

To obtain the best possible coherence in the thesis it is structured into seven parts. The first part is introductory and sets forth the frame and field of study for the thesis.

Following, to specify the concrete problems in achieving the aim of this thesis, the second part consists of the specific methodology in building the fluid domain which has to be implemented. Also there are presented all entry data needed for appropriate boundary definition. In addition, in that part are introduced the goals which have to be achieved. Lastly, all the possible test cases which have to be implemented are discussed according to their specifics.

With all the problems and requirements introduced, the third part concentrates on the algorithms which are suitable to be implemented for dealing the different problems. This is a kind of a discussion and overview about the possibilities which are available in the presents state of art. In addition a short comparison between these algorithms is made due to the external requirements.

Knowing all the milestones and some of their possible solutions, the fourth part comes with a detail explanation for their implementation. Here the solutions are presented in a practical point of view, as they will be used for solving this specific problems.

Following, to show the achieved goals, the fifth part introduces the test cases which are used for investigating the accuracy and the behavior of implemented algorithms. The test cases are selected with a view to represent realistic scenarios. Each of the cases has its own features which are possible to appear in a real situation. Lastly a evaluation of all results is made in order to be established what part of the work is done.

With all the results obtained, the sixth part comes to explain all possible improvements which have to be done. As well as here are presented all new ideas and additions which have appeared during the development of this work.

Technical specifications about the file formats with which the application works are provided in Appendix A on page 27.

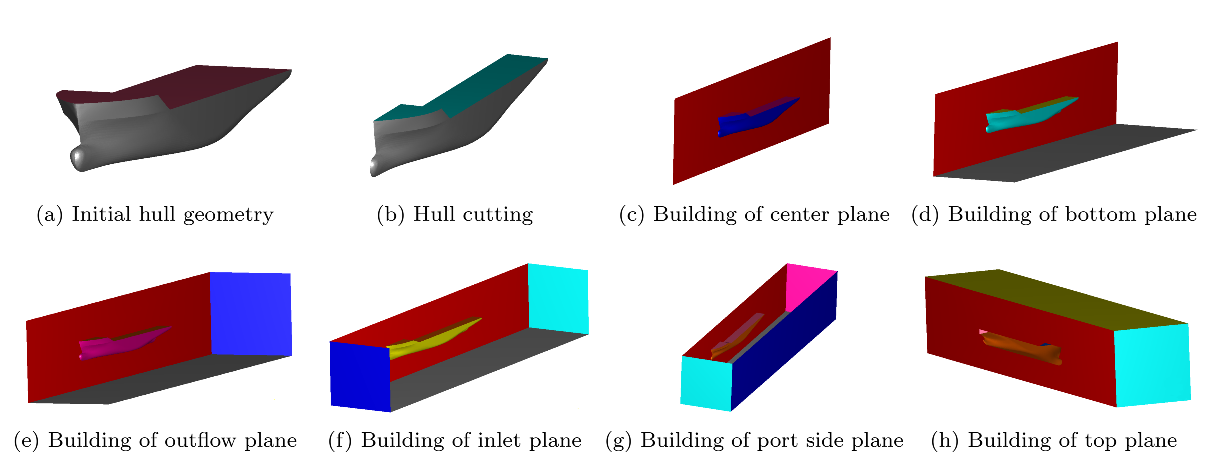
Guidances and specifics in working with the developed application can be found in Appendix B on page 29.

Due to the big number of graphical material which is obtained as a results from this work, an appendix with all of them is attached in the very end of this paper, see Appendix C on page 31.

Chapter 2

Problem Definition

First let consider some of the technical aspects of the problems. In HEXPRESS meshing software it is possible the geometry of the boundary domain to be imported in form of *STL*, see A.1, file format. Because of this it is suitable to present the initial hull geometry also in *STL* file format. This required that the geometry in the boundary domain have to be presented as a triangulation, in order to be written in the result *STL* file.


(a) Initial hull geometry (b) Hull cutting (c) Building of center plane (d) Building of bottom plane

(e) Building of outflow plane (f) Building of inlet plane (g) Building of port side plane (h) Building of top plane

Figure 2.1: Process of boundary domain generation

Figure 2.1 shows all the stages of boundary domain generation. It is necessary to consider the different problems which appear during this process according to the different types of ships. This is so because even though that the domain should be build around the floating body depending of its draft, the conditions in different cases vary.

There are a lot of common practices and rules in domain definition when it comes to a ship CFD simulation. These practices and rules will be assumed as a external restrictions for the application which is to be developed.

Admittedly in all cases no matter what is the type of the ship there are a common elementary operations, but their sequence and number of execution may vary. In the next sections are presented all the specifics in building the fluid domain, depending on the different type of ships and the external restrictions as well.

2.1 Specifics for monohulls

As monohulls are assumed ships having a single symmetrical hull. These are ships such as Container Ships, Bulk Carriers, Aircraft Carriers etc. In these cases the single hull is the object of interest of the CFD calculation.

2.1.1 Goals to be accomplished

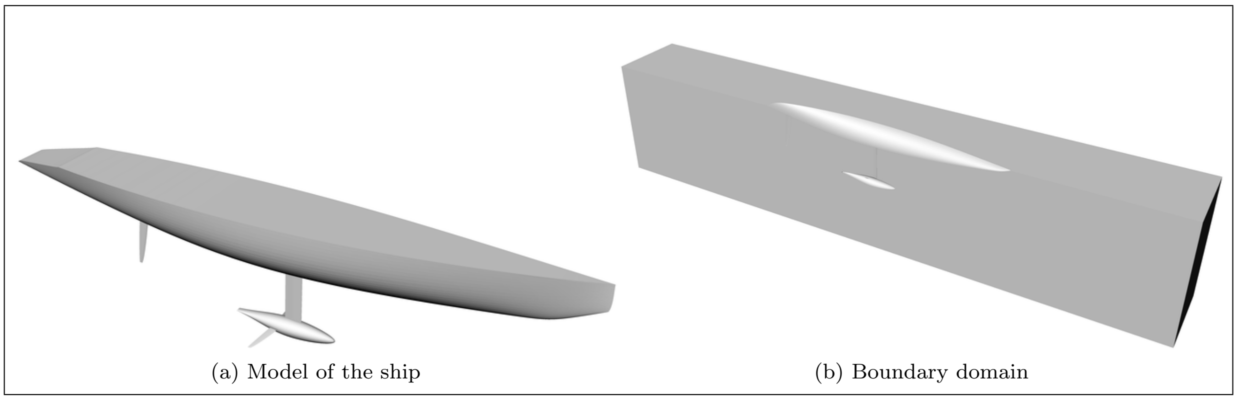

(a) Model of the ship (b) Boundary domain

Figure 2.2: Procedure in domain generation for monohull

In order to make all the further explanations more clear a graphical example is presented on Figure 2.2. A couple of modification are made to this model. First the hull has to be cut at the waterline because the remaining upper part is not a object of interest anymore. The second manipulation is made because of the symmetry of the hull so the starboard side is cut and removed too. The second step is made because CFD solvers has an option to define a boundary as symmetry and by this the amount of calculations is reduced significantly.

After these two steps are applied the generation of the boundary surfaces for the domain remains. Which means to define an array of triangles which lie in one single plane and are positioned in such a way to satisfy the dimensions of the fluid domain.

2.1.2 Entry Data

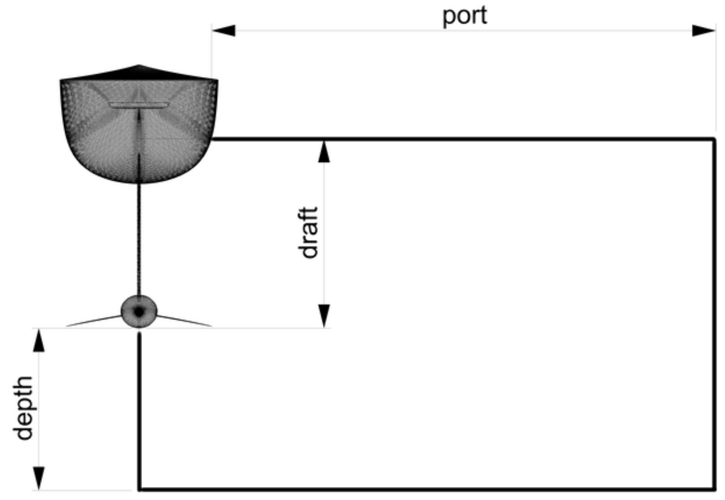
Figure 2.3 on page 5 presents all needed dimensions required for building a proper fluid domain. These dimensions are specified in advance and can be entered manually from the user. In addition there is one more dimension which is not presented on this figure. This is the position of the plane of symmetry. This issue will be discussed in later chapters.

2.2 Specifics for catamarans

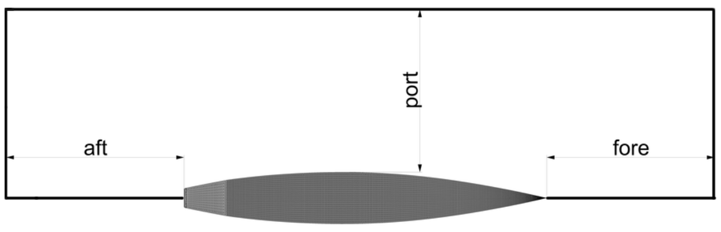
Catamarans are assumed having two connected hulls. In this case the object of interest are both hulls at the same time.

2.2.1 Goals to be accomplished

In order to make all the further explanations more clear a graphical example is presented on Figure 2.4 on page 5. Again as in the case of monohull it is assumed that a symmetry between the hulls exist. That

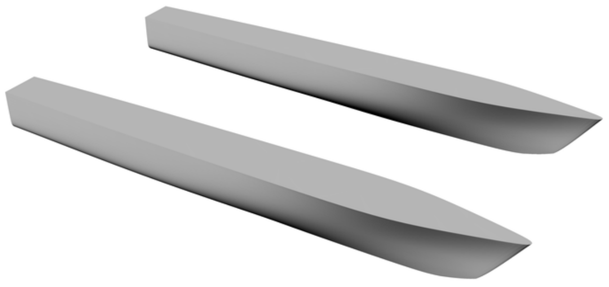


(a) Dimensions in front view

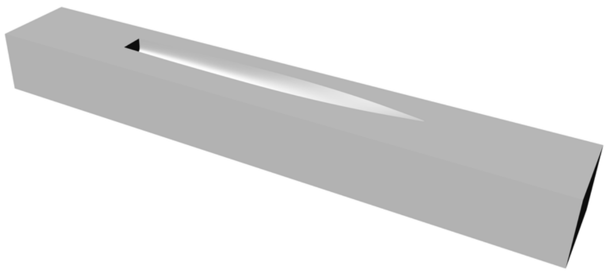


(b) Dimensions in top view

Figure 2.3: Domain dimensions required for monohull



(a) Model of the ship



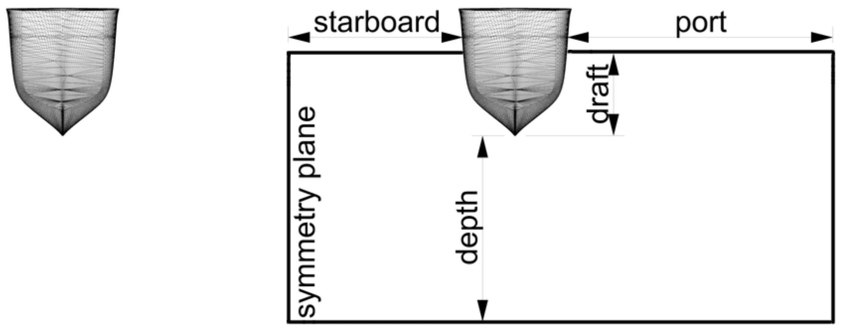
(b) Boundary domain

Figure 2.4: Procedure in domain generation for catamaran

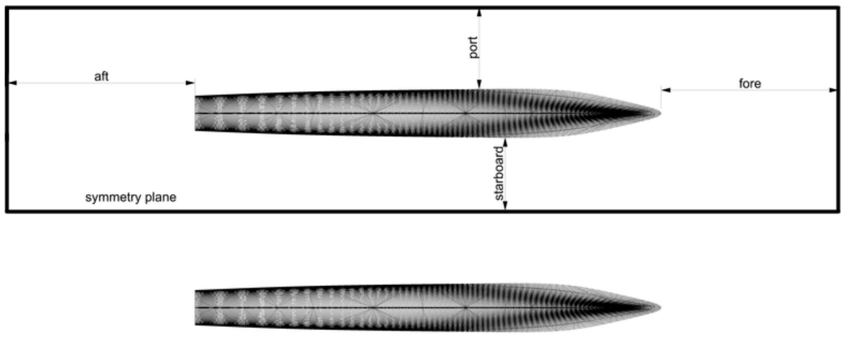
is why one of the hulls is removed entirely. But unlike the monohull case the remaining hull is cut only at the waterline. This is so because the plane of symmetry lies between the two hulls of the catamaran.

2.2.2 Entry Data

On Figure 2.5 on page 6 is presented the dimensions which have to be specified in advance. By defining these dimensions the user has a full capability to control the domain geometry. As a remark, it is good to know that the dimension called "starboard" represents the half of the distance between the two hulls. All other dimensions are identical with the monohull case.



(a) Dimensions in front view



(b) Dimensions in top view

Figure 2.5: Domain dimensions required for catamaran

2.3 Specifics for underwater vessels

As an underwater vessels it is assumed vessels which are fully immersed. Their draft is bigger than their height.

2.3.1 Goals to be accomplished

This case is completely analogical with monohull case, see 2.1, with the difference, that the draft of the vessel is greater than its height.

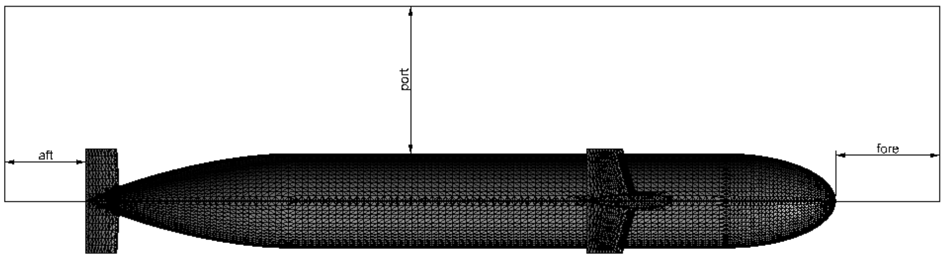
2.3.2 Entry Data

Figure 2.6 on page 7 presents all needed dimensions required for building a proper fluid domain in case of underwater vessel. Unlike the cases of monohull and catamaran in this case the draft has to be greater than the height of the underwater vessel. All other dimensions are identical with the previous cases.

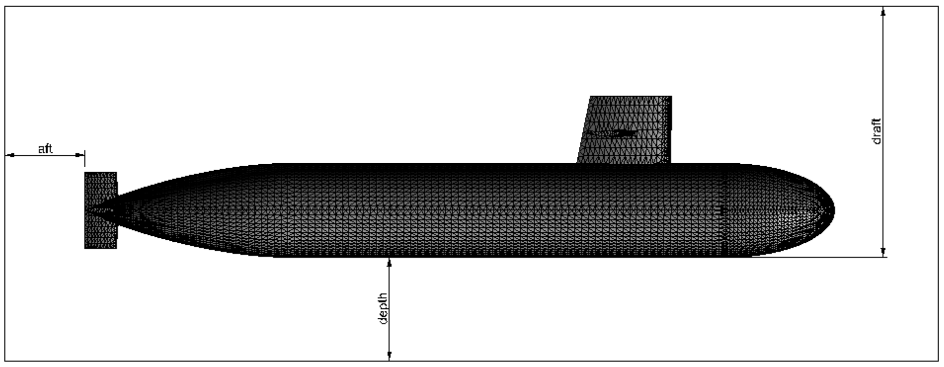
2.4 Implementation issues

Knowing the initial data and how the results should look like, it is possible to define the arising implementation problems correctly.

First of all algorithm for cutting the triangulation is needed. This algorithm is required to be capable in cutting in both horizontal and vertical planes, see Figure 2.7 on page 8.



(a) Dimensions in top view



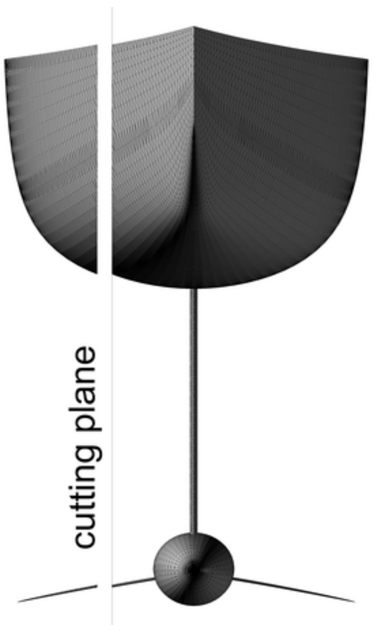
(b) Dimensions in side view

Figure 2.6: Domain dimensions required for underwater vessels

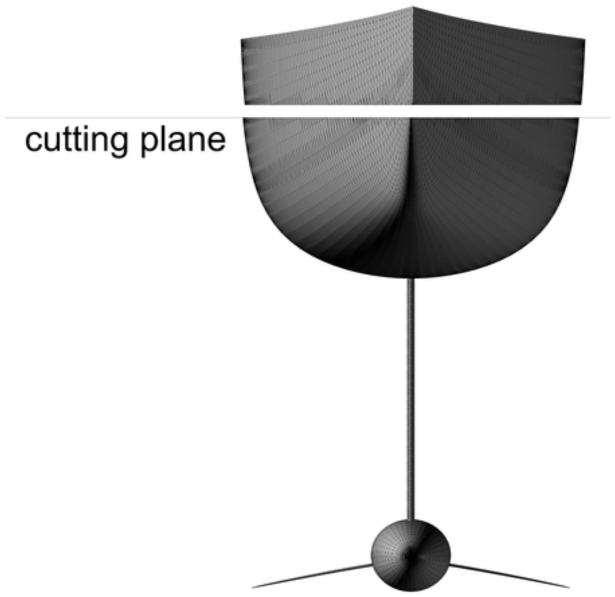
Next, an algorithm for building the simple bottom, inlet, outflow and side planes is required. This algorithms is assumed and implemented as a stand-alone procedure ,which does not affect the ship hull in any way, and does not even interact with it.

Following, as undoubtedly the most important part of the domain generation, comes the algorithm which has to build the complex waterline and centerline planes. The complex contour of waterline and centerline planes, see Figure 3.2a and 3.2b on page 10 requires the use of a procedure which is capable to represent the polygon bounded by it as a proper triangulation.

Lastly, as an addition, an algorithm for detecting sharp edges in the original ship geometry is needed. This is so because for the further volume meshing of the already defined domain, it is a must this edges to be defined. Such a sharp edges in the ship geometry are located in the area where the transom meets the sides and the bottom.



(a) Cutting by a vertical plane



(b) cutting by a horizontal plane

Figure 2.7: Ship cutting

Chapter 3

Possible approaches

3.1 Boundary surface generation

3.1.1 Simple boundary surfaces

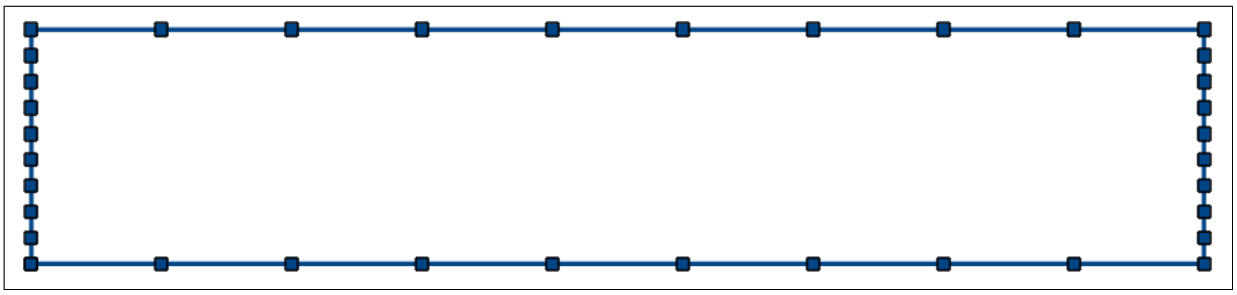


Figure 3.1: Simple surfaces contour

On Figure 3.1 is shown the geometry of the simple planes, such as bottom, side, inlet and the outflow plane. For the successful definition of these planes, it is necessary to triangulate the surfaces. Undoubtedly every grid generation algorithm can do this job, because the geometry is as simple as possible.

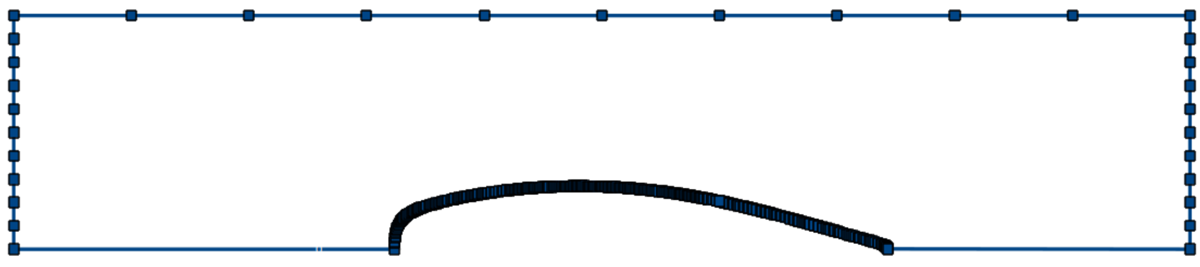
The simplest way to define a proper triangulation is to build and intersect all the segments between the corresponding nodes, and after that to split the received rectangles in triangles.

3.1.2 Complex boundary surfaces

On Figure 3.2 is shown the geometry of the waterline and centerline contours. This geometry depends only on the exact mesh of the ship hull, as well as on the number of nodes on the perpendicular simple planes. As an external requirement it will be assumed that the position of the nodes can not be changed.

Knowing the actual geometry of the domains which have to be triangulated makes the searching for a proper algorithm easier. A lot of possible solutions are presented in [5], which all can be applied in this situation. Solutions as:

- Delaunay triangulation
- Block-Structured grids
- Different variations for Structured grid
- Quadtree and Octree grids


(a) waterline contour

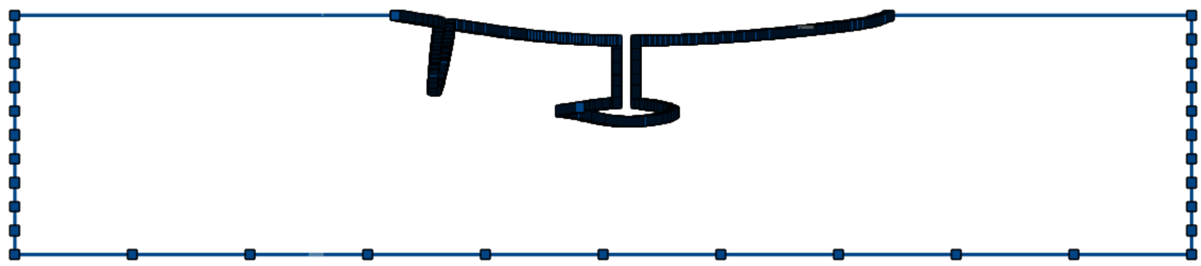

(b) centerline contour

Figure 3.2: Complex surfaces contours

The structured grid methods seems to be appropriate in this case, whilst the unstructured grid algorithms usage is not warranted for such a simple domain, see domain classification on Figure 4.2 on page 13.

Finding the simplest and easiest algorithm for implementation is important in this situation. This is so because there are no additional external requirements for the exact geometry of the elementary triangles. The only demand is that the whole domain to be tessellated with no overlapping, holes or any gapping. Therefore implementation of any algorithm which uses some axillary function for the grid generation may be assumed as a waste of resources.

The appropriate solution should only uses a simple triangulation built by adding diagonals to this polygon. This is so because no additional points or edges have to be built in the process of grid generation. Such algorithms are presented in [7, p. 1 - p. 58], [6, p. 491 - p. 519] and [8, p. 1 - p. 30].

As an addition there are couple of C++ geometric libraries which include a lot of geometric algorithms. Such libraries are CGAL and LEDA. There are a lot of materials about them, but unfortunately there is only one published in paper format. The capabilities of these external libraries are literally unlimited, but their usage makes the source code considerably more complicated, because of their multidimensional data structures.

3.2 Hull cutting

Figure 2.7 describes the requirements for the cutting algorithm very well. What the algorithm should do is to remove the “unwanted” triangles from the existing ship mesh, and after that it has to reconfirm the position of the triangles in the boundary layer, in order to guarantee the integrity of the result mesh. This is so because no gaps, overlapping and holes are allowed.

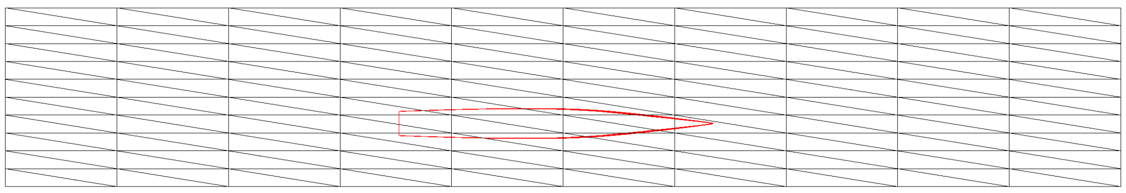
Very suitable for this job contouring algorithm is presented in [9], which has the capability to cut objects with really complex shapes. Because of the fact that in this case, one relatively complex geometry has to be cut by a simple plane, this algorithm can be significantly simplified before its implementation.

3.3 Transom detection

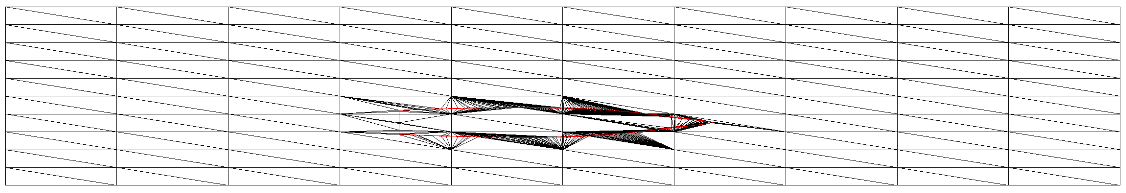
For the detection of the transom a topology analysis of the geometry is needed. Useful techniques and principles are provided in [2, p. 44 - p. 67]. Unfortunately implementing an entire topology analysis procedure would require the implementation of a big number of algorithms and would go far beyond the aims of this thesis.

Instead of that a simple technique for the detection of the transom can be applied, assuming that the transom plane is parallel to the midsection plane of the vessel. This can be done by a short algorithm which sorts the facets according to the components of their normal vectors.

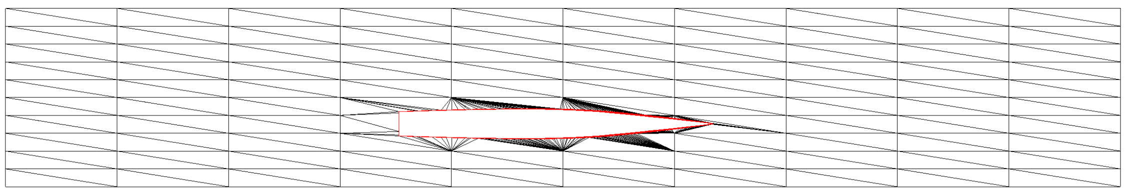
3.4 Boolean Intersection



(a) Initial triangulation and the ship contour



(b) Adaptation of the initial triangulation with the contour polygon



(c) Result triangulation after boolean intersection

Figure 3.3: Boolean Intersection example

On Figure 3.3 is shown another approach for dealing with the triangulation of the complex boundary surfaces. In this case all the boundary surfaces are being built as a simple boundary surfaces, and after that the parts, which are not included in the boundary domain are being removed. For this operation it is only needed all the edges from the different contours to be sorted in a native order in advance. This procedure can be applied for more complex contours for ships such as trimarans or even for special spatial floating structures.

Chapter 4

Implemented algorithms

4.1 Boundary Surface Generation

4.1.1 Simple boundary surfaces

For the generation of the flat boundary surfaces is implemented one relatively simple algorithm. All boundary surfaces are generated by one same algorithm, which just uses different boundary conditions in the different cases. In this section is presented an explanation for the bottom surface only, but all other surfaces are generated in analogical way.

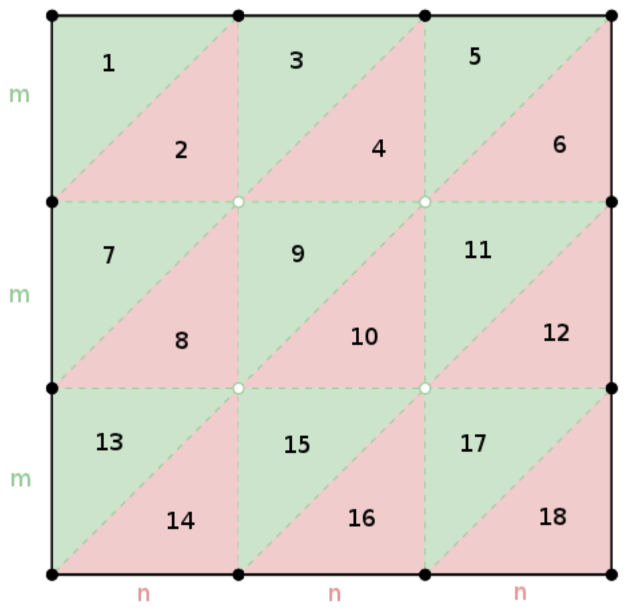


Figure 4.1: Sequence in generation of a boundary surface

For the tessellation of one surface it is discretized as an array of identical rectangles. Then these rectangles are split by one of their diagonals in order triangles to be produced. If all the points of these triangles are represented as a matrix it looks like:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \dots & a_{1j} \\ a_{21} & a_{22} & a_{23} & a_{24} & \dots & a_{2j} \\ a_{31} & a_{32} & a_{33} & a_{34} & \dots & a_{3j} \\ a_{41} & a_{42} & a_{43} & a_{44} & \dots & a_{4j} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & a_{i3} & a_{i4} & \dots & a_{ij} \end{pmatrix} \tag{4.1}$$

It is easy to see that all the triangles are defined by the triples:

$$\left[a_{ij},\, a_{(i+1)j},\, a_{i(j+1)}\right] \;\; and \;\; \left[a_{(i+1)(j+1)},\, a_{(i+1)j},\, a_{i(j+1)}\right] \tag{4.2}$$

Exactly this relationship is used for the tessellation of any of the boundary surfaces of the virtual fluid domain.

4.1.2 Complex boundary surfaces

Four basic types of planar domains can be distinguished [6, p. 294].

1. simple polygon - includes both boundary and interior, see Figure 4.10a
2. polygon with holes - it is a simple polygon minus the interiors of some other simple polygons and its boundary has more than one connected components, see Figure 4.10b
3. multiple domain - complex domain which contains internal boundaries, each of them can contain holes or another internal boundary, see Figure 4.2c
4. curved domain - planar domain which boundaries can be algebraic curves such as splines, Bezier and etc. see Figure 4.2d

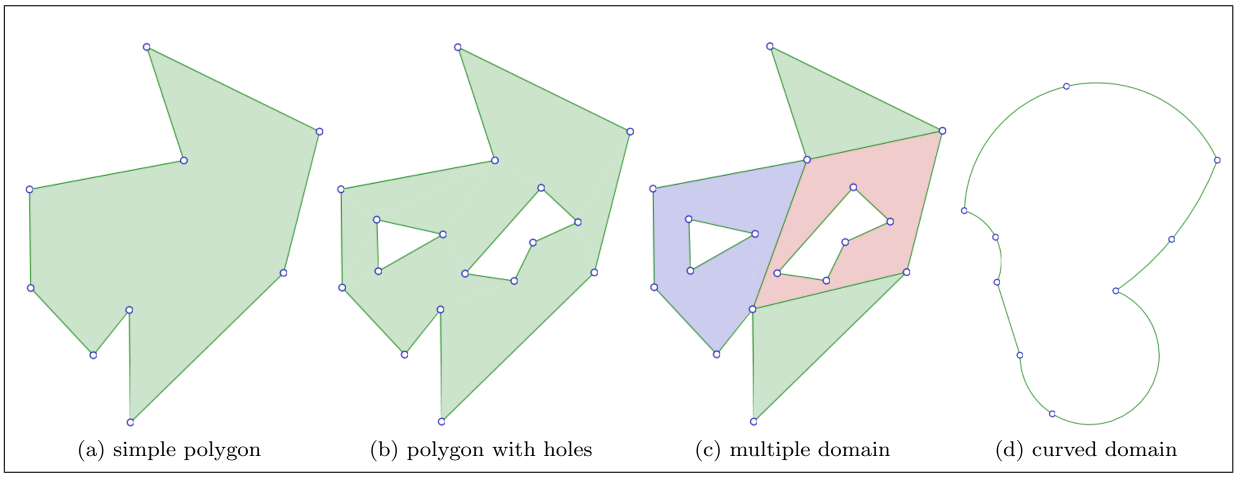


Figure 4.2: Basic Types of Polygons

It's possible to divide any non-simple polygon to two or more simple polygons by adding new edges or redefining the boundary rules. That's why only an algorithm for triangulating a simple polygon will be presented.

The triangulating algorithm which is used in this work is presented in [7, p. 1 - p. 43], [8, p. 1 - p. 30] and [6, p. 973 - p. 1023]. According to that algorithm the triangulation of any simple polygon can be presented as a dual connected graph with no cycles with nodes associated with each triangle, see Figure 4.3 . It's also true if the dual T of a triangulation is being assumed as a tree, with each node of degree at most three.

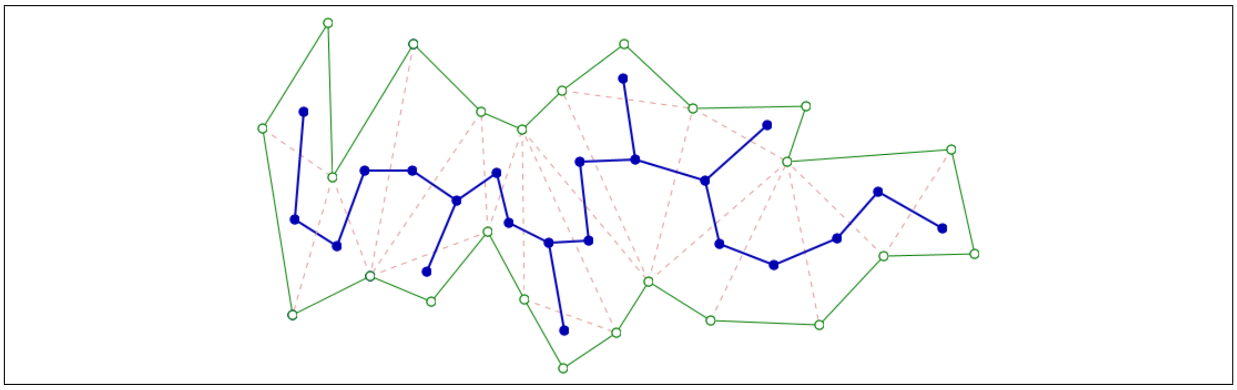


Figure 4.3: Triangulation Dual

For finding this tree it's necessary all non-overlapping diagonals to be found first. But finding all the diagonals and after that grouping them in triangles is not so easy task. That's why a different approach is needed. The simplest solution which uses minimum amount of topological information is to check for every three neighbor vertices if they define a leaf from the triangulation tree. In other words if v_0, v_1, v_2 are three ordered neighbor vertices and v_0v_2 is a diagonal then they define a leaf. It's possible to remove that leaf from the tree without changing the tree in any way. The simplest way to find all triangles of the triangulation is to loop through all ordered triples of points until only three points remain, see Figure 4.4.

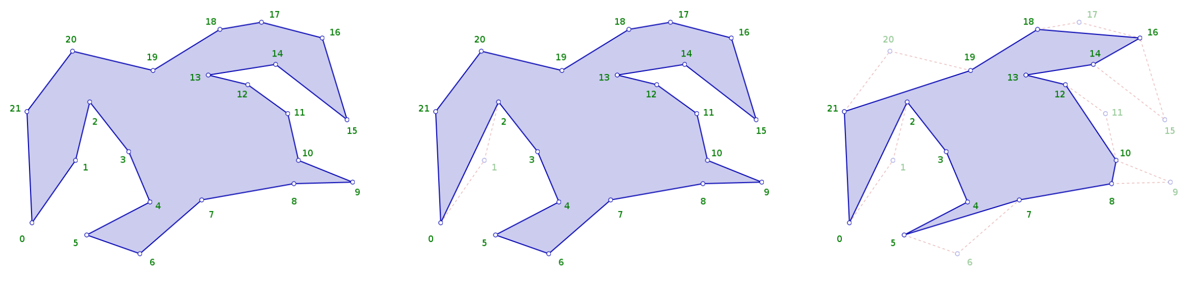

(a) Initial polygon (b) Polygon after removing one leaf (c) Polygon after removing seven leafs

Figure 4.4: Leaf removal

4.1.3 Domain simplification for catamarans

The algorithm presented in 4.1.2 works only with simple domains. And for its implementation in case of catamarans some manipulations should be made to their contours. On Figure 4.5 on page 15 are shown the original contour of the waterline and the same contour after adding two edges. In this way the polygon with holes transforms to two simple domains, see the classification of domains in 4.1.2 on page 13.

The two added edges define a line which splits the hull on the halves, depending on the position of the foremost point of waterline. This line is especially chosen because of the specific geometry of ship waterline.

4.1.4 Domain simplification for underwater vessels

In this case the geometry of the buttock contour is more "nonstandard" comparing to the case for catamaran, see Figure 4.6 on page 15. In this case the upper-foremost and upper-aftmost points are picked

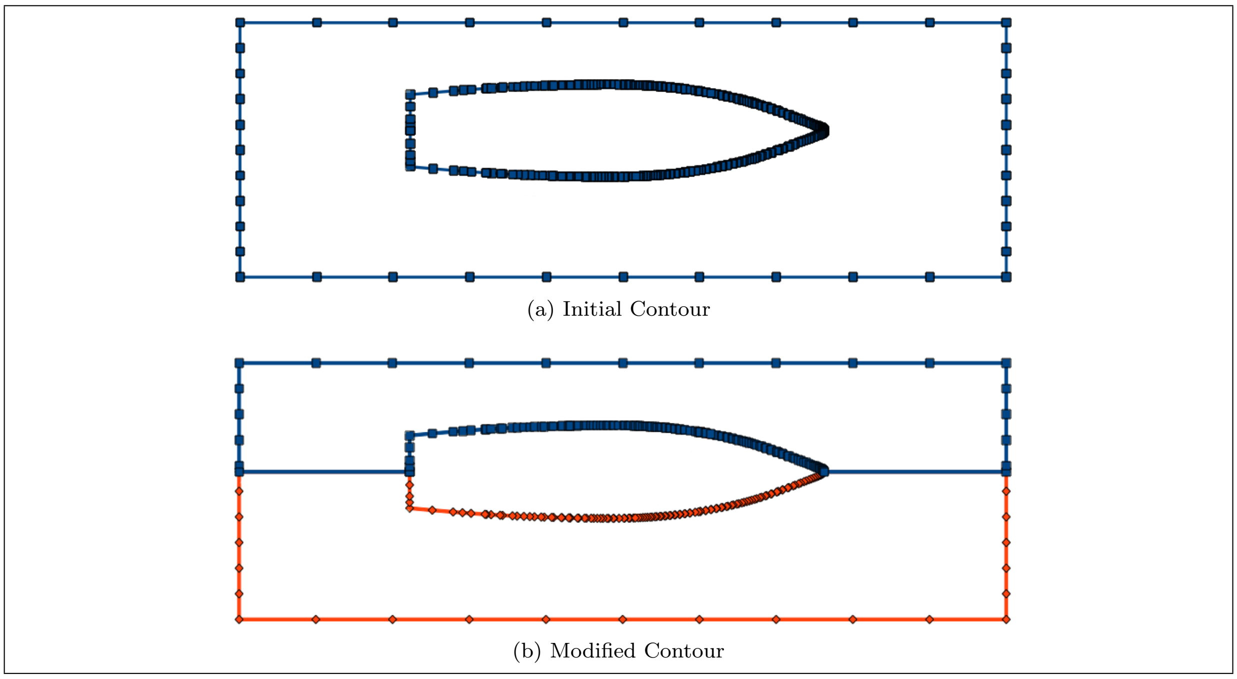


Figure 4.5: Catamaran boundary contour

for the building of the additional edges. For the working of this simplification is needed the initial ship mesh to be provided with a deck plane, with other words it has to be completely watertight, otherwise the algorithm will fall in endless loop, in searching the second path from the foremost to aftmost points.

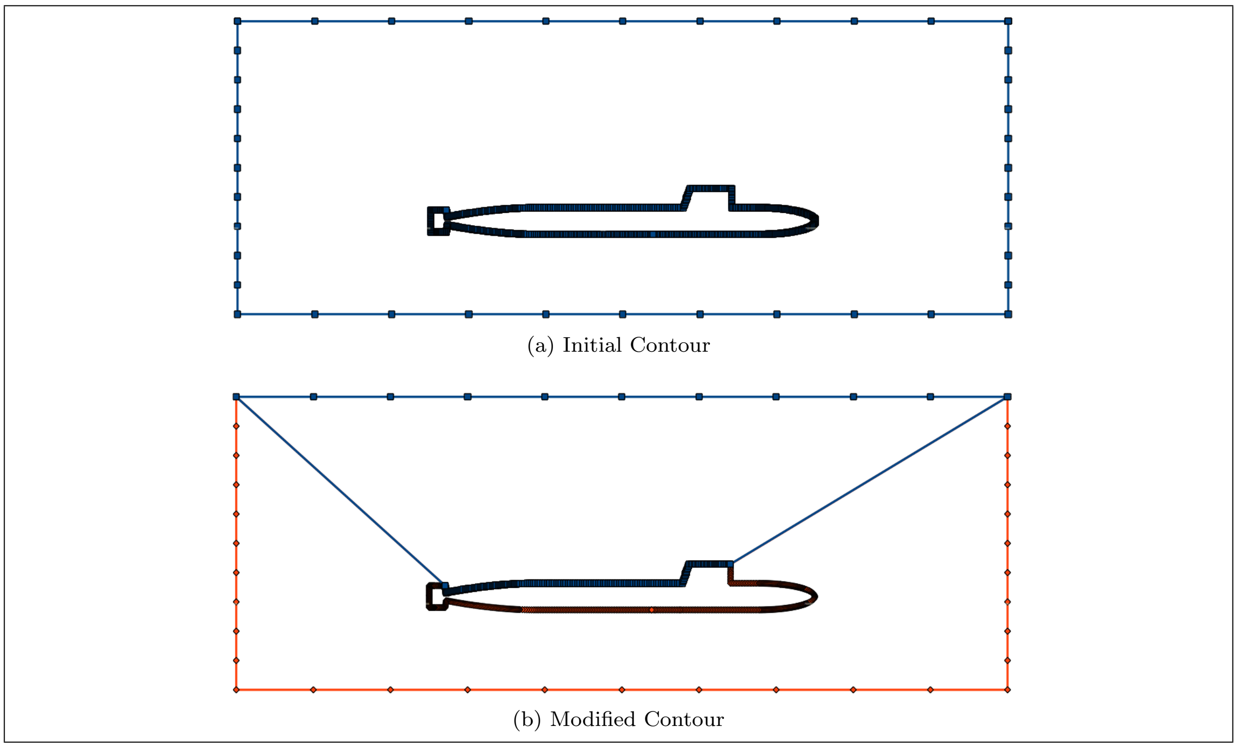
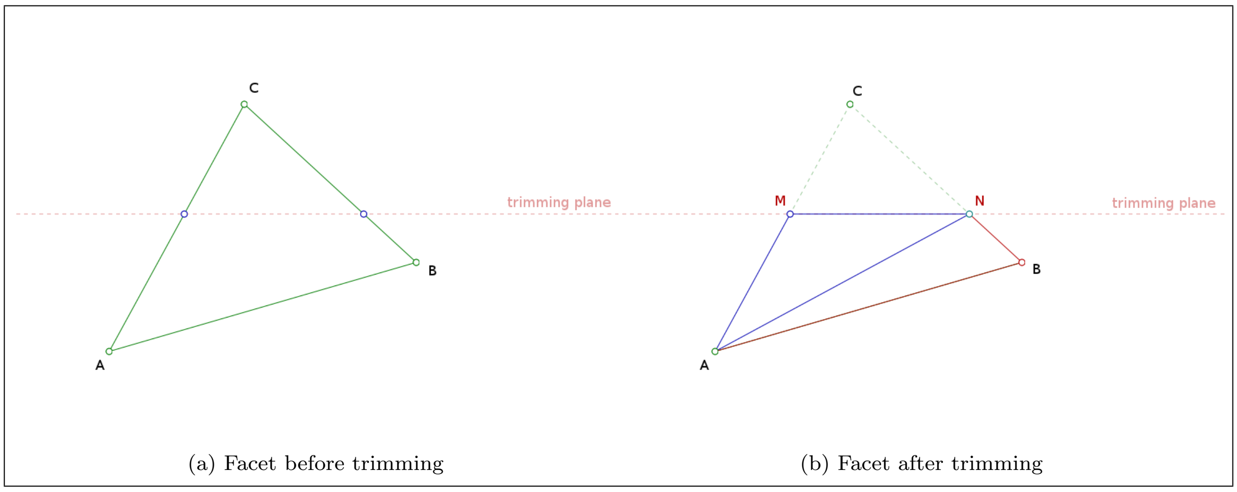


Figure 4.6: Submarine boundary contour

4.2 Hull cutting

The cutting algorithm checks the positions of the facet vertices and the trimming plane relative to each other for all facets. This algorithm is used for cutting the ship model by waterline and centerline planes.



(a) Facet before trimming (b) Facet after trimming

Figure 4.7: Facet trimming

On Figure 4.7 on page 16 is shown the logic of trimming algorithm. Looping through all the facets from the ship hull it performs the following steps:

1. Checks if any from the facet vertices lies outside the trimming boundaries. It is good to know that trimming boundaries are defined by the engineer in advance. Also it is possible that more than one vertices lie outside trimming boundaries. In that case the algorithm is analogical to the case shown on Figure 4.7, but it has to form only one triangle instead of two.
2. After identifying the external point it is necessary the original facet to be split it two separated parts. In order to do so first the points of intersection of trimming plane and the facet edges have to be found. This operation is done by linear interpolation.
3. After finding the points of intersection M and N it forms two separated triangles $\triangle ANM$ and $\triangle ABN$ and add them to the result array of hull facets. Finally the original facet $\triangle ABC$ is deleted.

Simultaneously with adding new facets to the result array it is necessary to calculate their normals too. There are two more "trivial" cases:

- When all vertices of the facet are outside the trimming domain - then the facet is just being deleted.
- When all vertices of the facet are inside the trimming domain - then the facet remains in the array in its original shape.

4.3 Transom detection

Normal Vector Components			
Region	x - component	y - component	z - component
Transom	$x \neq 0$	$y \approx 0$	$z \approx 0$
Flat Boards	$x \approx 0$	$y \neq 0$	$z \approx 0$
Flat Bottom	$x \approx 0$	$y \approx 0$	$z \neq 0$

Table 4.1: Topology groups and their normal vector components

This feature is implemented in order to make sure that the sharp edge between the transom and the rest of the hull geometry will be properly detected by HEXPRESS. This is very important that all the sharp edges have to be defined in advance for the building of proper volume mesh.

As it was mentioned in 3.3 for the detection of the transom is chosen a relatively simple algorithm. This algorithm strongly depends on the assumption that the transom lies only in one section plane. In Table 4.1 is presented the rule, which is used for sorting the different topology groups of facets.

There is nothing more especially about this algorithm. It has to be assumed as a sorting algorithm, which uses the criteria introduced in Table 4.1 for the sorting itself.

4.4 Boolean intersection

This algorithm can be separated in two parts. First part could be assumed as a procedure for adding of edges to existing mesh of triangles. This step is visualized on Figures 3.3a and 3.3b. The second part is removing of "unwanted" facets from the modified mesh, see Figure 3.3c.

4.4.1 Adding edges to existing mesh

This algorithm is implemented as a pattern recognition procedure, similar to the procedure presented in [9]. Having all contour edges sorted in a native way, and the initial mesh presented, it is necessary to add every single edge to the triangle in which it is situated or which it intersects. For this purpose all possible variations of the relative position of one edge and a triangle have to be covered, see Figure 4.8.

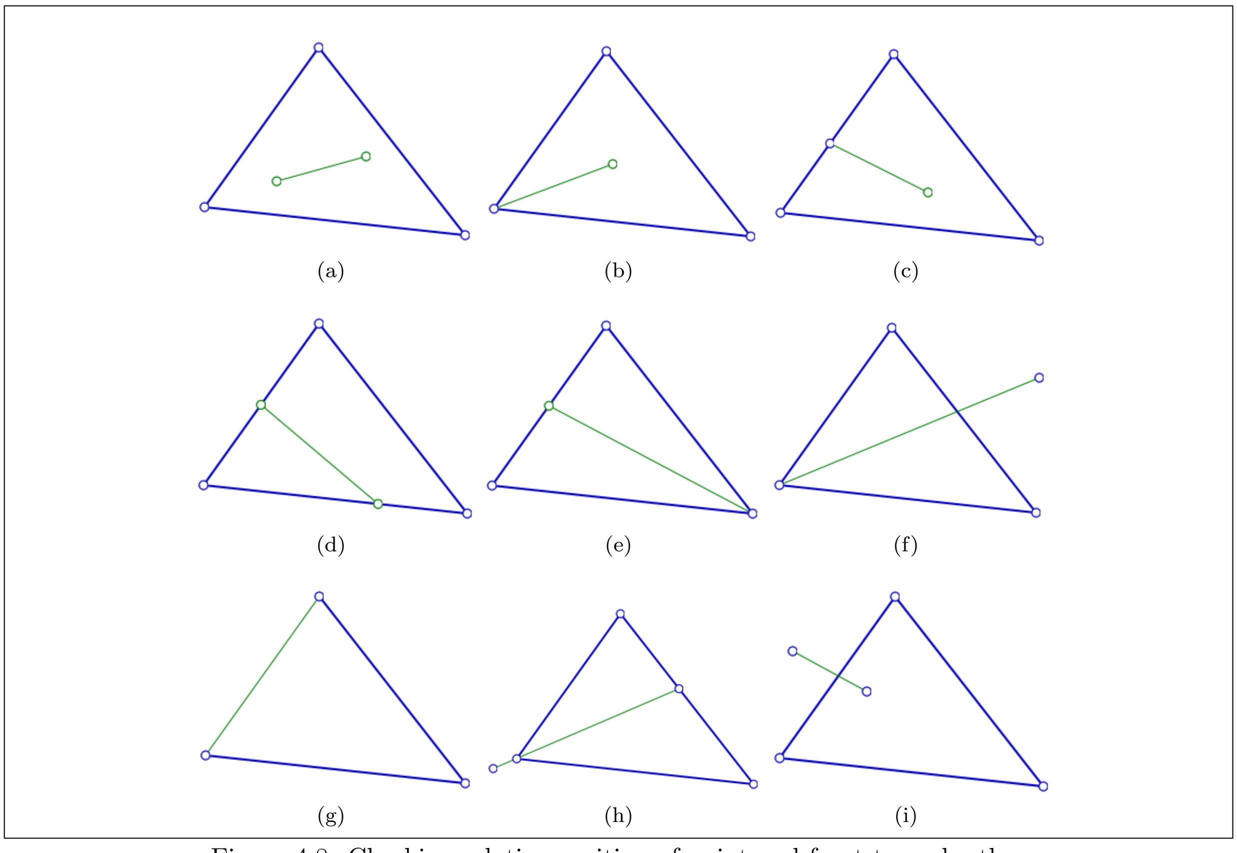


Figure 4.8: Checking relative position of point and facet to each other

Having all patterns defined, it is necessary to loop through all the edges in order to distinguish the current pattern. This is iterative search so if for any edge a pattern can not be found then after modification of the rest of the array of triangles on next iteration the situation should be different. When some of the patterns shown on Figure 4.8 is recognized, the topological space defined from the endpoints of the edge and the facet is being triangulated again. Finally the new facets are being added to the array of boundary surface facets and the original facet is being deleted, for example see Figure 4.9. This algorithm relies on the procedure of searching of triangle interior points. This is so because for both vertices of the current edge have to be checked if they lie inside/outside the facet, for such algorithm check [12].

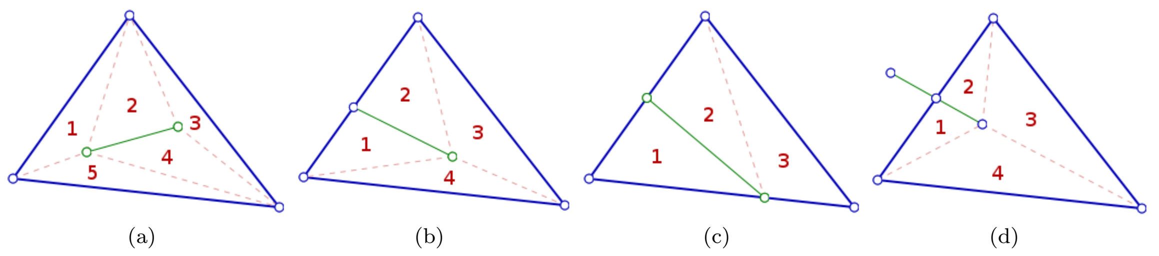


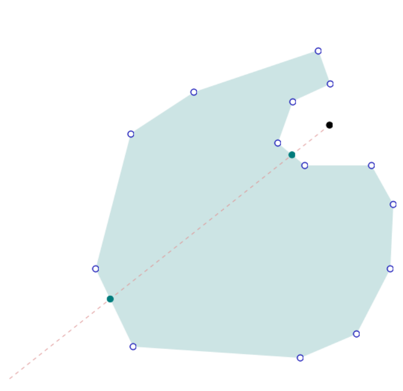
Figure 4.9: Examples for the triangulation of the new topological space

4.4.2 Removing of the "unwanted" facets

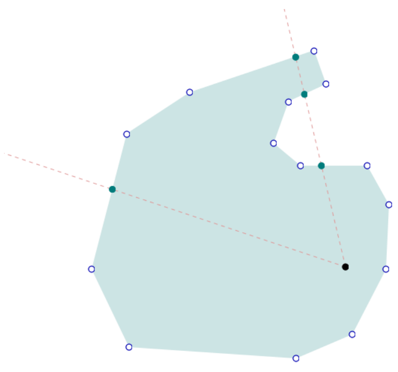
When the contour of the ship is presented and all the edges are added to the initial surface the only step which left is removing of "unwanted" facets. The criteria for removing one facet is if any of its vertices lies inside the polygon bounded by the ship contour. This step entirely depends on searching of polygon interior points and should be assumed as this. There are a lot of method for checking if one points lies inside or outside of a polygon, see [11], [7, p. 239 - p. 245]. For the needs of this thesis is implemented a modification of the so called "Ray Crossing" algorithm, presented in [7, p. 239 - p. 245].

Let assume that already one nonconvex polygon and a point P are presented. In order to check if P lies inside or outside this polygon the following method is introduced, see Figure 4.10 on page 19:

- A ray from P to the infinity is drawn. Each time that the ray crosses the boundary of the polygon, it will cross it from the interior to the exterior, or vice versa. Therefore the test point is on the interior if, and only if, the ray crosses the boundary an odd number of times.
- There is a possibility that this ray intersects the segment of the convex hull in where two edges segments meet or to go parallel to any segment. In these cases another ray will be drawn in another direction and the check will be made again.
- Another tricky case is when the point lies on any of the segments. For this case it is necessary to calculate the distance from P to the intersected contour edge.



(a) Point outside of a polygon



(b) Point inside in a polygon

Figure 4.10: Checking relative position of point and polygon to each other

Chapter 5

Investigating the results

5.1 Test Cases Definition

In this section are presented all ship models with all the entry data used for this survey. As a attempt to avoid the issue of floating-point round-off error it is recommended the models to be in their real dimensions, especially it's really undesirable if mesh edges are smaller than 1.10^{-5}.

5.1.1 NPL Hull Series

This model can be characterized as a round bilge hull with straight entrance waterlines, rounded afterbody sections and straight buttock lines terminating sharply at a transom [10]. The model hull form is presented on Figure 5.1 and its dimension are presented in Table 5.1 .

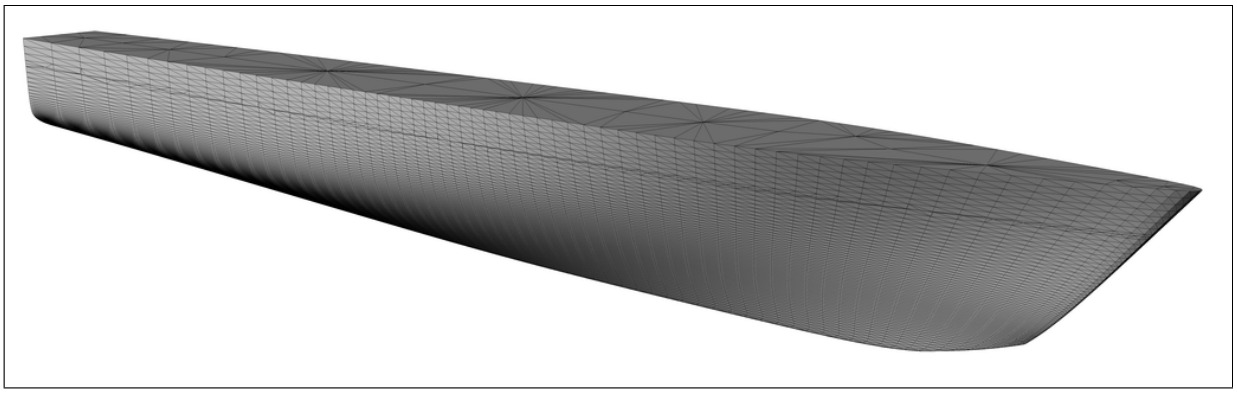


Figure 5.1: NPL Series Hull Mesh

NPL Hull			
Length	Beam	Depth	Number of facets
49.983	5.204	5.871	21312
units	units	units	facets

Table 5.1: NPL Hull Model Dimensions

5.1.2 MOERI KCS Container Ship

The KCS was conceived to provide data for both explication of flow physics and CFD validation for a modern container ship with bulb bow and stern. The model hull form is presented on Figure 5.2 and its dimension are presented in Table 5.2 .

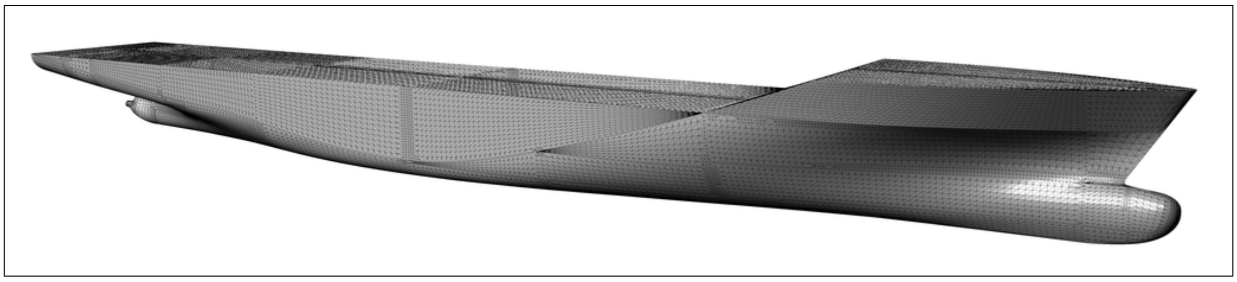


Figure 5.2: KCS Container Ship Hull Mesh

MOERI KCS Container Ship Hull			
Length	Beam	Depth	Number of facets
231.508	30.608	22.316	63236
units	units	units	facets

Table 5.2: MOERI KCS Container Ship Model Dimensions

5.1.3 US Navy Combatant, DTMB 5415

Model 5415 was conceived as a preliminary design for a Navy surface combatant. The hull geometry includes both a sonar dome and transom stern. The model hull form is presented on Figure 5.3 and its dimension are presented in Table 5.3 .

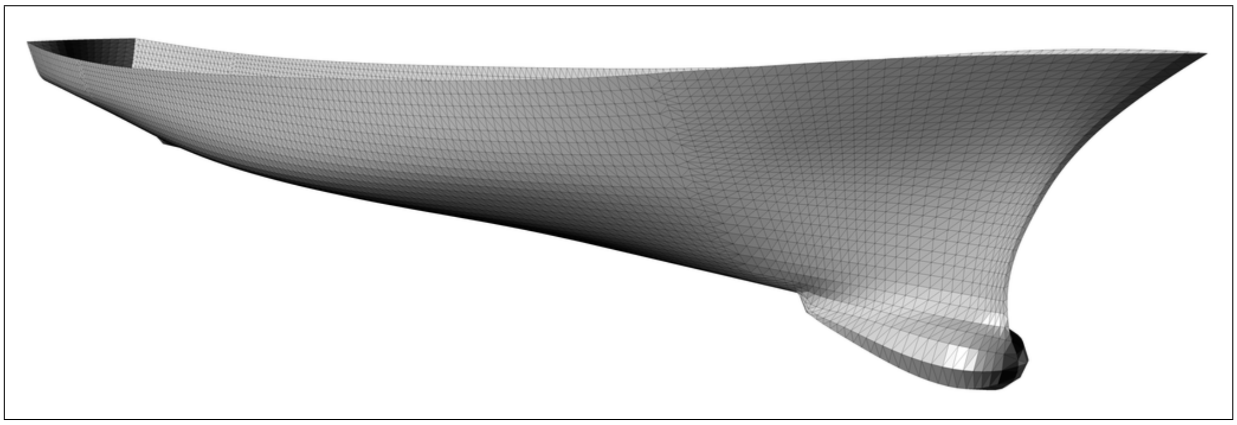


Figure 5.3: DTMB 5415 Hull Mesh

US Navy Combatant, DTMB 5415 Hull			
Length	Beam	Depth	Number of facets
328.841	44.089	41.200	17288
units	units	units	facets

Table 5.3: US Navy Combatant, DTMB 5415 Model Dimensions

5.1.4 Sample sailing Yacht

This model is provided with all appendages typical for sailing yachts. This hull is really suitable for testing the possibilities of the boolean operation algorithms, because of its complex underwater shape. The model hull form is presented on Figure 5.4 and its dimension are presented in Table 5.4 .

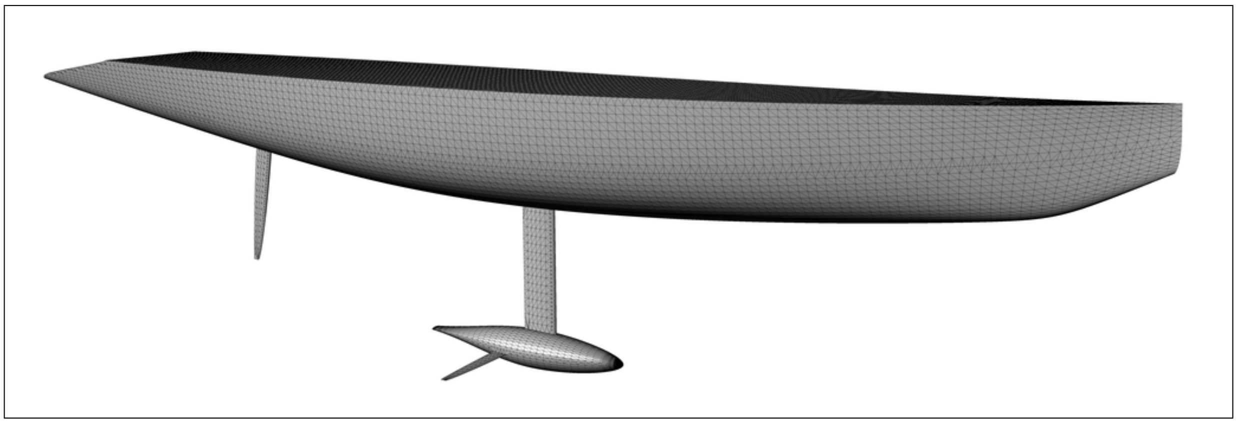


Figure 5.4: Sample sailing Yacht Hull

Sample sailing Yacht Hull			
Length	Beam	Depth	Number of facets
23.597	3.350	5.566	34994
units	units	units	facets

Table 5.4: Sample sailing Yacht Model Dimensions

5.1.5 Sample submarine

This model is provided with all appendages typical for sailing yachts. This hull is really suitable for testing the possibilities of the boolean operation algorithms, because of its complex underwater shape. The model hull form is presented on Figure 5.5 and its dimension are presented in Table 5.5 .

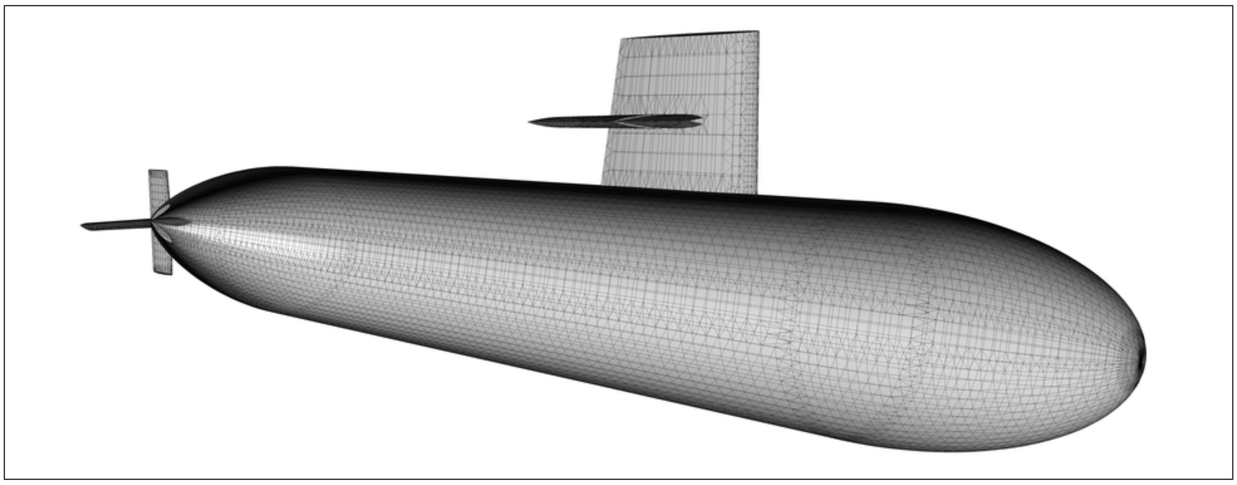


Figure 5.5: Sample submarine Hull

Sample submarine Hull			
Length	Beam	Depth	Number of facets
79.956	11.067	17.177	22768
units	units	units	facets

Table 5.5: Sample submarine Model Dimensions

5.2 Test Cases Results

Because of the simplicity of Boundary Surfaces Generation algorithm presented in 4.1.1 on page 12 no special investigation will be submitted about it. More interesting are the cases of Boolean Operations, performed on waterline and centerline planes. After manipulation with the developed application the new models will be directly loaded in HEXPRESS in order to check if they are compatible.

5.2.1 Testing complex boundary surfaces generation for monohulls

In this group of result will be submitted the hulls with centerline symmetry As monohulls are treated cases Sample sailing Yacht (5.1.4), US Navy Combatant, DTMB 5415 (5.1.3), MOERI KCS Container Ship (5.1.2). For these cases the hulls of the models have to be cut by the waterline and centerline.

Results for triangulation by leaf removal for monohulls						
Case	Perspective	Waterline	Centerline			Volume Mesh
5.1.2	Figure C.4	Figure C.5	Figure C.6	Figure C.7	Figure C.8	Figure C.9
5.1.3	Figure C.10	Figure C.11	Figure C.12	Figure C.13	Figure C.14	Figure C.15
5.1.4	Figure C.16	Figure C.17	Figure C.18	Figure C.19	–	Figure C.20

Table 5.6: Results for triangulation by leaf removal for monohulls

Table 5.7 presents a visualization of the triangulated domains for different monohulls. It can be noticed that one very accurate triangulation is made. No holes or gaps are detected even in areas of the bulbous bow and the bulbous stern for case MOERI KCS Container Ship. The triangulation is built successfully even in the case Sample sailing Yacht, where the geometry of the contouring domain is much more complex than in all other cases, see Figure C.18 on page 40. Due to these results it can be concluded that this triangulating algorithm can be successfully implemented for the needs of this thesis.

Here is the place to explain why a course example volume mesh is build only for case US Navy Combatant, DTMB 5415, see Figure C.15 on page 38. This is so because of the fact, that the meshing software HEXPRESS detects topology problems with cases MOERI KCS Container Ship and Sample sailing Yacht. This is an entirely expected behavior, due to the fact that no transom or Flat Bottom, or Flat Boards are defined and no edges are recognized in the ship hull.

The reason why case C.15 can be directly used for building an volume mesh is that not so sharp edges exist in the original geometry of the ship compared to the other two cases.

5.2.2 Testing complex boundary surface generation for catamarans

Here will be submitted only results for NPL Hull Series (5.1.1). This test is done in order to show that the application can work with ship hulls which don't lie on the line of symmetry or with asymmetric ship hulls.

From the visualization presented on Figures C.21 and C.22 can be noticed that the simplification of the initial domain is a successful approach for triangulating more complex domains.

Results for triangulation by leaf removal for catamarans				
Case	Perspective	Waterline	Centerline	Volume Mesh
5.1.1	Figure C.21	Figure C.22	–	–

Table 5.7: Results for triangulation by leaf removal for monohulls

5.2.3 Testing complex boundary surface generation for underwater vessels

Here will be submitted only results for Sample submarine (5.1.5), Sample sailing Yacht (5.1.4) , MOERI KCS Container Ship (5.1.2). This test is done in order to check if the application can work with vessels which entire hull is underwater and only centerline boolean is required.

Results for triangulation by leaf removal for underwater vessels					
Case	Perspective	Centerline			Volume Mesh
5.1.5	Figure C.23	Figure C.24	Figure C.25	Figure C.26	–
5.1.4	Figure C.27	Figure C.28	Figure C.29	–	Figure C.30
5.1.2	Figure C.31	Figure C.32	Figure C.33	Figure C.34	Figure C.35

Table 5.8: Results for triangulation by leaf removal for underwater vessels

From the figures presented in Table 5.8 can be concluded that the simplification of the centerline plane presented in 4.1.4 can be successfully applied for the triangulation of fully submersed vessels.

5.2.4 Testing transom detection

For testing the transom detection algorithm is chosen case MOERI KCS Container Ship (5.1.2) , because its transom meets the requirement to be parallel to the midsection.

Results transom detection			
Case	Transom	Flat Bottom	Flat Sides
5.1.2	Figure C.1	Figure C.2	Figure C.3

Table 5.9: Results transom detection

In Table 5.9 are presented visualizations of the ship hull after the sorting of the facets according to their normal vector components. On Figure C.1 on page 31 is seen that in this case the algorithm is working correctly, and no unpredictable behavior is observed. This algorithm is working correctly and for the detection of the Flat Bottom, see Figure C.2 on page 32. Unfortunately the results for the detection of the Flat Boards are not so good. On Figure C.3 on page 32 can be observed that some of the facets from the Flat Board are not recognized as such. This may be caused from incorrectly definition of the initial ship mesh, or due to tolerance issue. Because of this inconclusive performance, the detection of the topology is implemented as a optional feature.

5.2.5 Testing boolean intersection

Unfortunately the results for this algorithm couldn't been provided. This is so because of the unstable behavior of the algorithm during the tests. In most of the cases the algorithm falls in a infinite loop in splitting the remaining edges. Edge splitting is implemented for the cases shown on Figures 4.8f, 4.8h, 4.8i. In those cases the contour edge is being split by the intersection point with the facet edge. Further investigation is required in order the exact reason for this behavior to be localized.

Chapter 6

Conclusion

Received bounding box is entirely watertight and meets all the topological requirements to be compatible with HEXPRESS. The result domain is quite suitable to be a base for generation of the volume grid. The application can be used as a “single-click“ tool for generation of fluid domain boundaries without using any other CAD systems. Unfortunately due to implementation issues the boolean intersection algorithm could not obtain any results during its testing and further research is needed for its successful implementation.

The algorithm for topology detection has good capabilities for sorting the facets, according to their normal vector components, but no any further topological searching is implemented. Because of this fact it is not able to detect any aligned surfaces. This algorithm highly depends on the correctness of the initial model, which is its main disadvantage. Anyway this algorithm can be used as a first iteration of other topology search procedure.

Further research can be aimed in improving and developing topology detection, and in particular detection of all boundary edges of the ship hull e.g. deck edge, bottom edge for ship with V shaped sections.

Bibliography

[1] Stefan Krueger, Manuel Manzke, Thomas Rung, Hendrik Vorthoelter
"Introduction of RANS-CFD into the Initial Design Process", Hamburg University of Technology

[2] Herbert Edelsbrunner, "Geometry and Topology for Mesh Generation", Cambridge Monographs on Applied and Computational Mathematics

[3] Anthony D. Martin, "Admesh user manual", California State University - Long Beach

[4] "HEXPRESS User Manual v2.9", NUMECA International

[5] Thompson J, and Soni B, and Weatherill N "Handbook of grid generation.", CRC Press 1999

[6] J.-R. Sack, J. Urrutia "Handbook of Computational Geometry", Elsevier 2000

[7] Joseph O'Rourke "Computational geometry in C", Cambridge University Press 1998

[8] Joseph O'Rourke "Art Gallery and Algorithms", Oxford University Press 1987

[9] Paul Bourke "CONREC. A Contouring Subroutine", BYTE Magazine 1987
`http://paulbourke.net/papers/conrec/`

[10] D. Bailey "The NPL High Speed Round Bilge", The Royal Institution of Naval Architects 1976

[11] Paul Bourke "Determining if a point lies on the interior of a polygon", November 1987
`http://paulbourke.net/geometry/insidepoly/`

[12] Triangle Interior `http://mathworld.wolfram.com/TriangleInterior.html`.

Appendix A

Technical Specifications

A.1 .stl file format

The application developed in this thesis uses **.stl** file format as a native file format for geometry representation. Suitable software for visualization of **.stl** files under Linux OS are **NETGEN**, **MeshLab**, **Gmsh**. In case when any correction have to be made to the **.stl** file, **ADMesh** is very suitable for Filling Holes, Repairing nearby facets and Repairing normal vectors.

This file format is used for geometry representation as an array of triangles with theirs surface normals. The file is structured in different solids as each solid is a container for one triangle array. There are two types of STL files: ASCII and binary. For this work is used the ASCII version of the file, because of its human-friendly format.

```
solid name
    ...
    facet normal  $n_i$   $n_j$   $n_k$
      outer loop
        vertex  $x_1$   $y_1$   $z_1$
        vertex  $x_2$   $y_2$   $z_2$
        vertex  $x_3$   $y_3$   $z_3$
      endloop
    endfacet
    ...
endsolid name
```

A.2 .stl.prop file format

This file is created and associated to a STL file automatically by HEXPRESS^{TM} , when some topology modifications are made on the original STL file. In this file all triangles may be grouped in different topology facets, and when it's attached to the original geometry file (STL file) then HEXPRESS^{TM} knows which triangle belongs to which topology face. Using of this kind of file is really useful for accurate representation of geometry and for reducing of any modification work after loading of the geometry. This file format has very simple structure, which critically depends on the structure and the arrangement of the STL file to which it's attached.

ng
$attribute_0$
$\vdots$

$attribute_{n-1}$

where ng is the number of topology faces and n is the number of triangle in the *.stl* file to which the *.stl.prop* file is assosiated

Appendix B

The use of software

B.1 List of files

This is a list of the required files for running the application. All these files should be placed in a single folder.

settings.conf This file contains all the entry data for the current project. This file is read by **mesh** and the presented data is used for the building of the fluid domain.

admesh This application is automatically invoked by **mesh** in order to fix all tolerance problems that can exist. As a result of this application **model-v3.stl** is produced.

tri This is a stand alone triangulating application. It uses the algorithm presented in 4.1.2 on page 13. This application is exact copy of the original triangulating algorithm, but it is compiled with capability to work with *long long* data type in C++. In addition this application output is modified in order the triangulated facet to be available in a direct manner.

mesh This is the executable file of the developed application.

model.stl This is the initial mesh file of the ship. For the proper functioning of **mesh** application it is critically that all naked edges in the initial design to be joined before meshing, as well as all normal vectors should be unified. It is not recommended to make any further “operation” to be made to this mesh, e.g. Filling Holes with third party software, because there is no guarantee that the integrity of the mesh is preserved. In addition it is critical that in this file only one *solid* should be defined. If more than one solids exist, the application will not read the file correctly.

model-v2.stl This is the original output file. No tolerance corrections were made to it.

model-v3.stl This is the result mesh file of the domain after executing the **mesh** file.

model-v3.prop.stl This is the property file attached to the result domain mesh, after executing the **mesh** file.

B.2 Entry data

The entry data have to be written in **settings.conf** file in advance. It is a normal ASCII file with the structure presented in Table B.1. In case that a domain for a underwater vessel is wanted to be built, then *number of bodies* have to be equal to 1, and the *draft* have to be bigger than the height of the initial model. The arrangement of these dimensions is a must and any reordering will cause unwanted application behavior.

tag	data type	description
fore=	[double]	see Figure 2.3b on page 5
aft=	[double]	see Figure 2.3b on page 5
port=	[double]	see Figure 2.3b on page 5
starboard=	[double]	see Figure 2.4b on page 5
depth=	[double]	see Figure 2.3a on page 5
draft=	[double]	see Figure 2.3a on page 5
center=	[double]	see Figure 2.4a on page 5
N=	[int]	number of nodes produced the simple planes in one direction
tolerance=	[double]	tolerance used for edge sorting
number of bodies=	[int]	1 - monohull , 2 - catamaran
transom detection=	[int]	1 - yes, 0 - no
flat bottom detection=	[int]	1 - yes, 0 - no
flat sides detection=	[int]	1 - yes, 0 - no

Table B.1: settings.conf - strucutre

B.3 Running the application

After entering all dimensions and properties in **settings.conf** file, the application can be executed in terminal by *./mesh* and it will produce **model-v2.stl**, **model-v3.stl** and **model-v3.stl.prop** automatically. In any case of unexpected behavior the best solution is the *tolerance* and *center* to be considered again. For example by simply changing *center= 0* to *center= 0.01* can solve a lot of problems.

The normal execution time of the application may vary depending on this if any topology detection is turned on. If all detections are off, then the running time is about 2 seconds, but when all three are on then the running time may exceed 2 minutes.

Appendix C

Test Cases Results

C.1 Testing transom detection

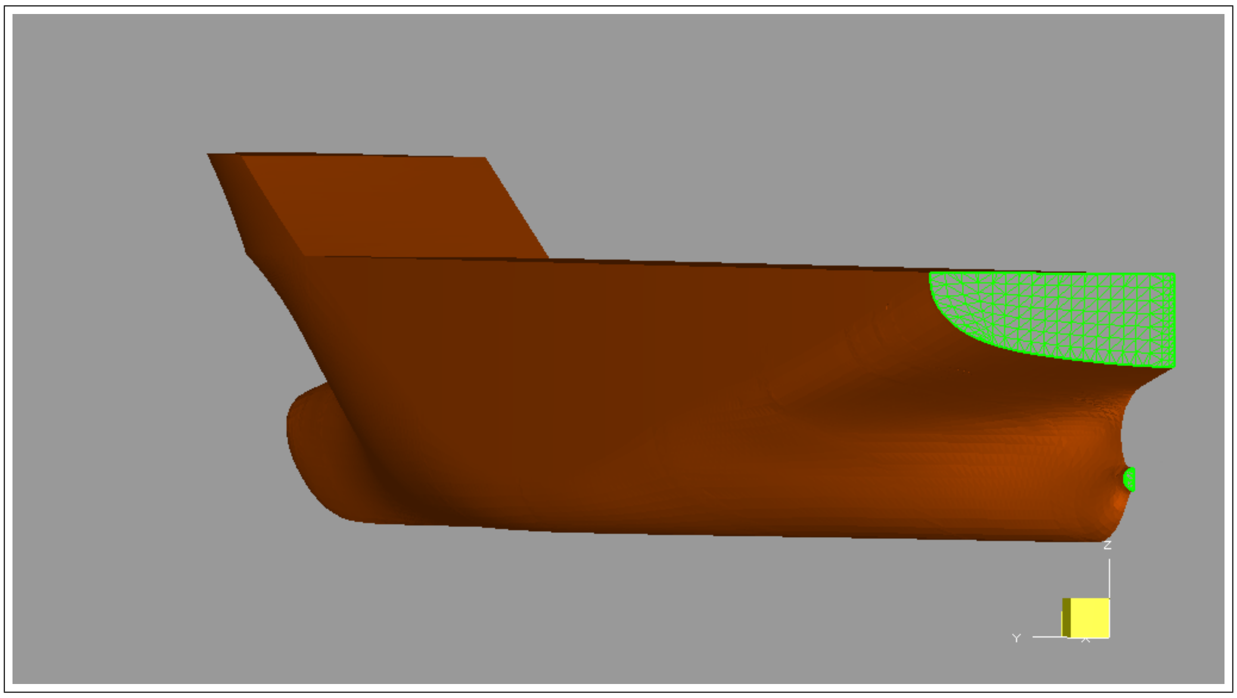


Figure C.1: Transom in HEXPRESS for MOERI KCS Container Ship

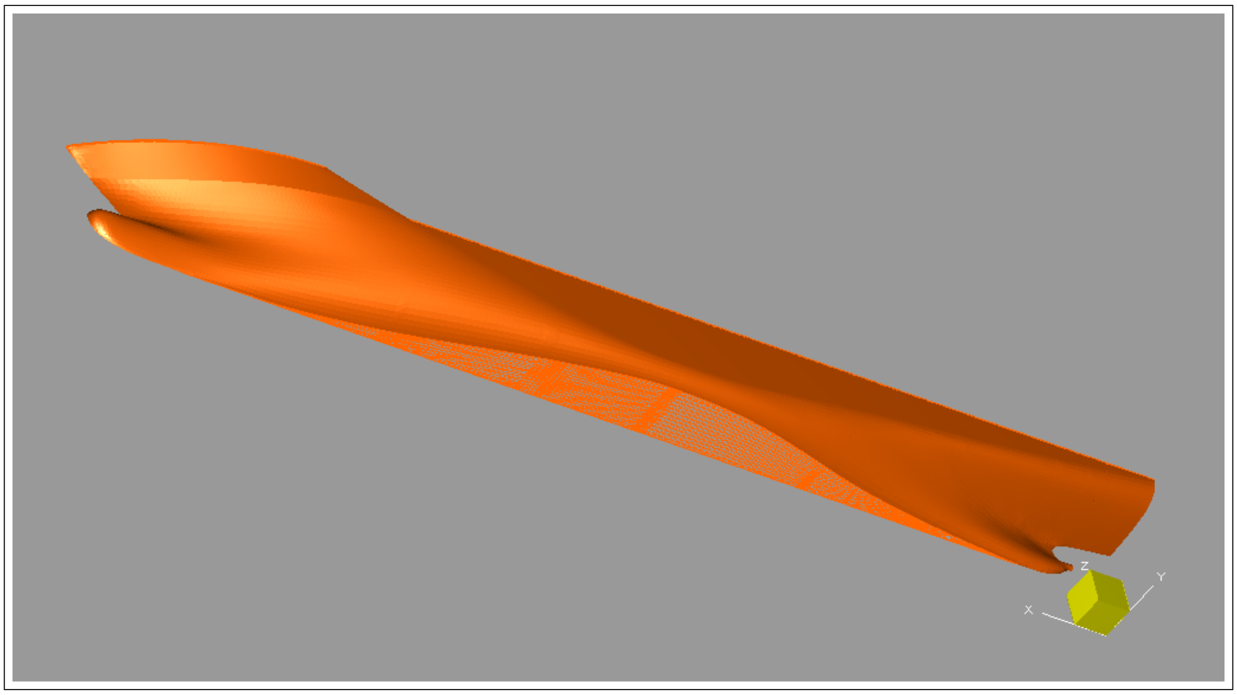


Figure C.2: Flat Bottom in HEXPRESS for MOERI KCS Container Ship

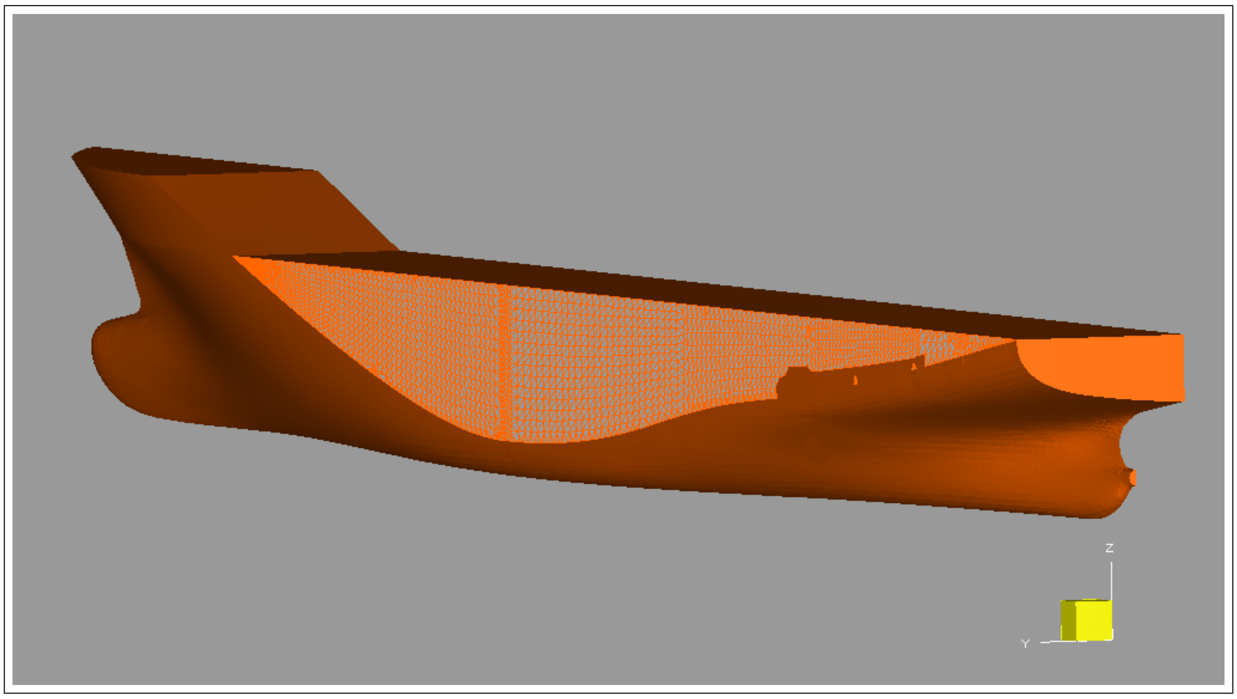


Figure C.3: Flat Board in HEXPRESS for MOERI KCS Container Ship

C.2 Testing triangulation by leaf removal for monohulls

C.2.1 MOERI KCS Container Ship

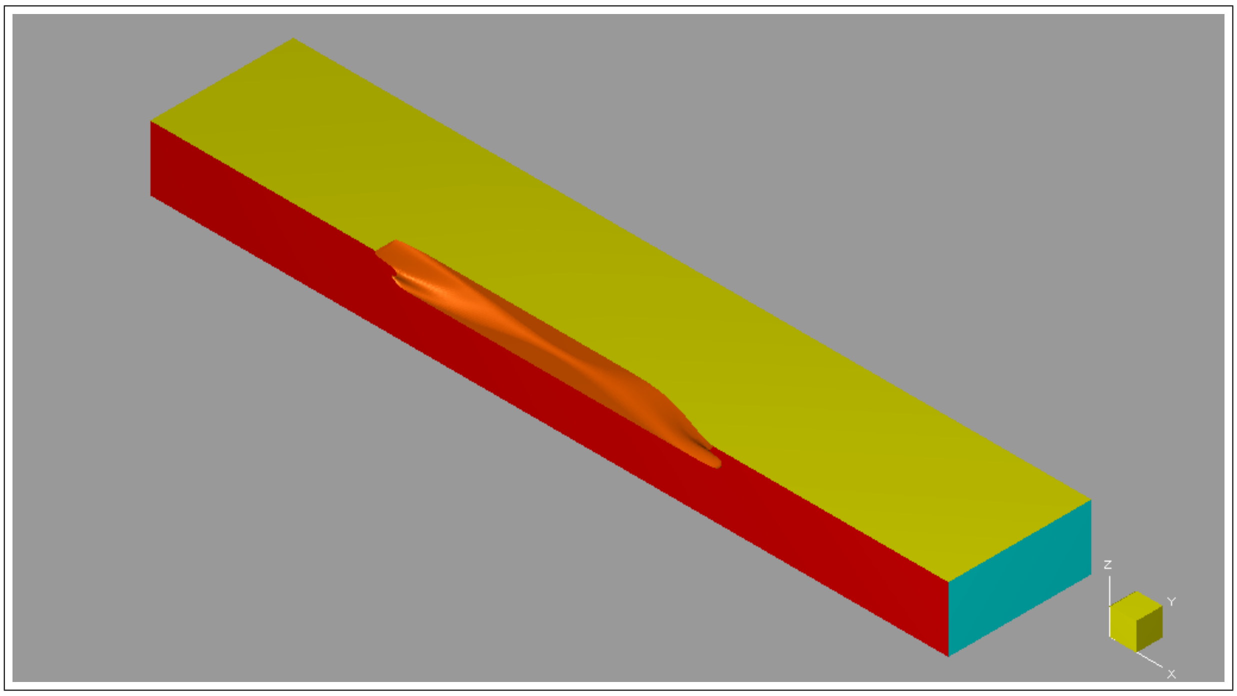


Figure C.4: Entire Fluid Domain in HEXPRESS for MOERI KCS Container Ship

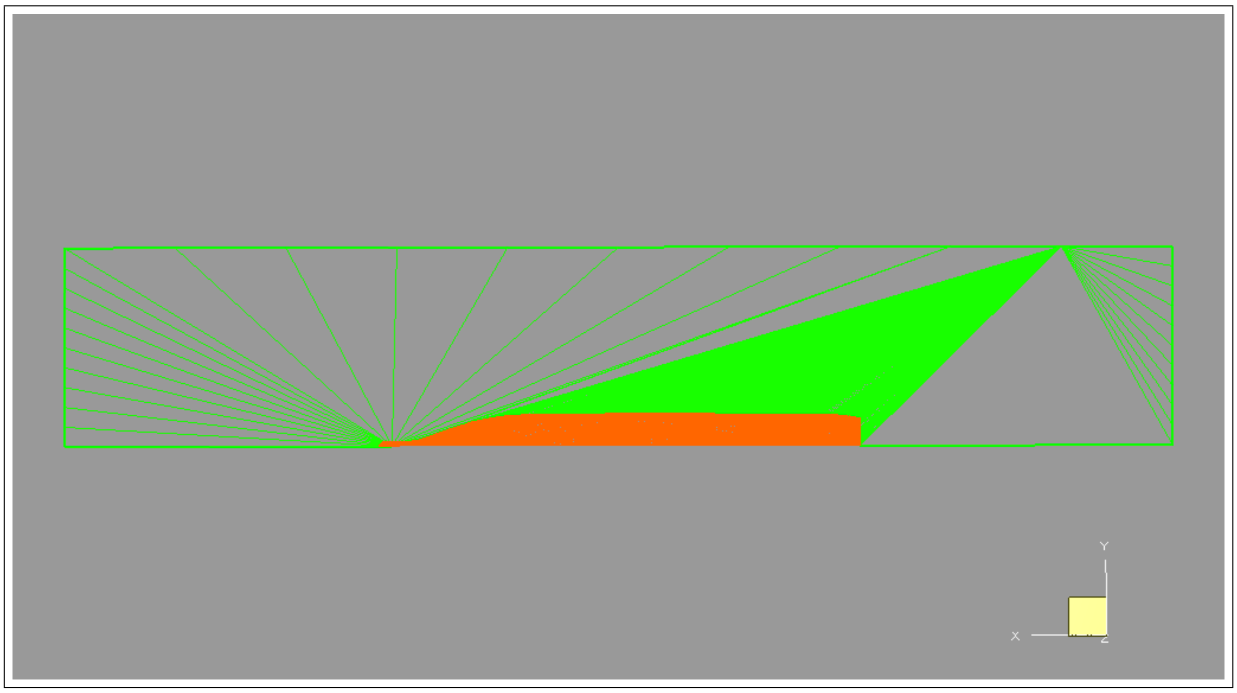


Figure C.5: Waterline Triangulation for MOERI KCS Container Ship

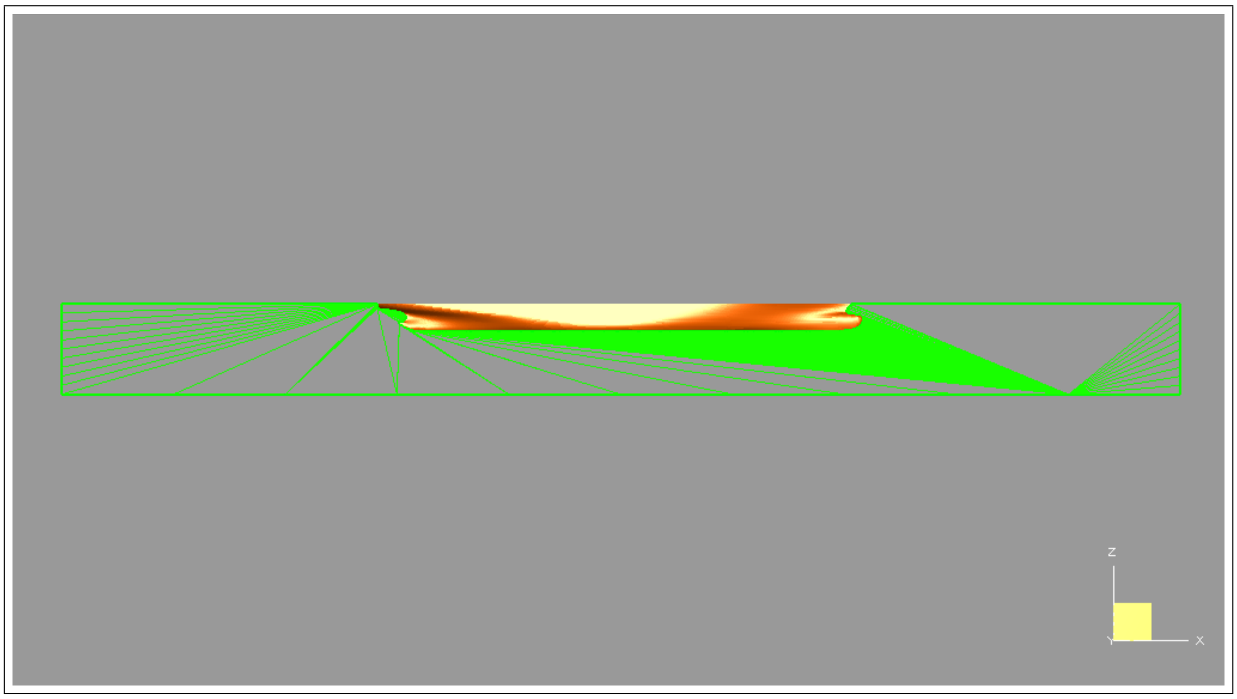


Figure C.6: Centerline Triangulation for MOERI KCS Container Ship

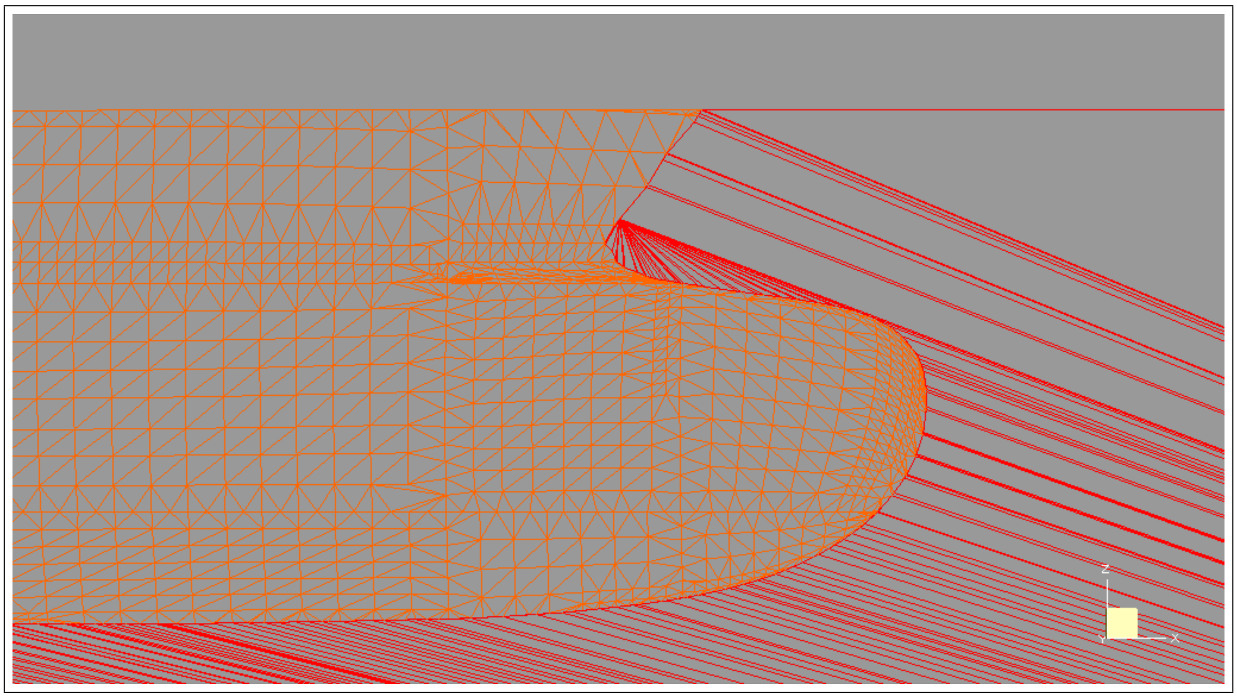


Figure C.7: Centerline Triangulation in bulbous bow region for MOERI KCS Container Ship

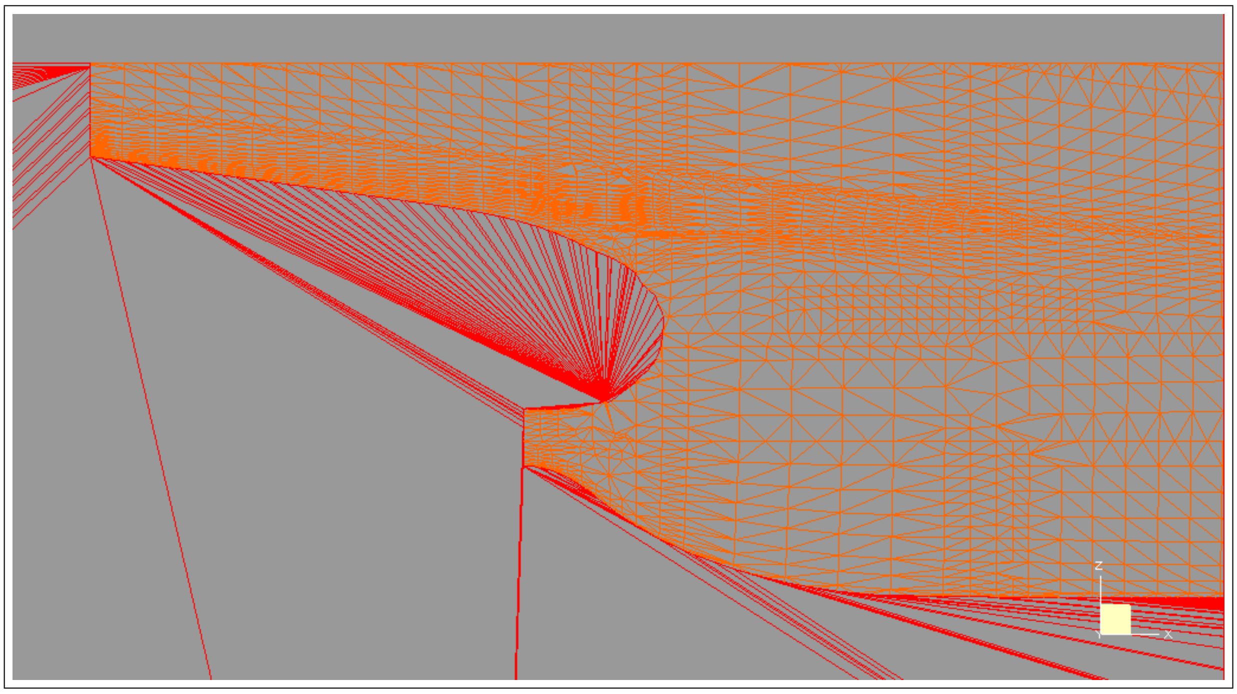


Figure C.8: Centerline Triangulation in transom region for MOERI KCS Container Ship

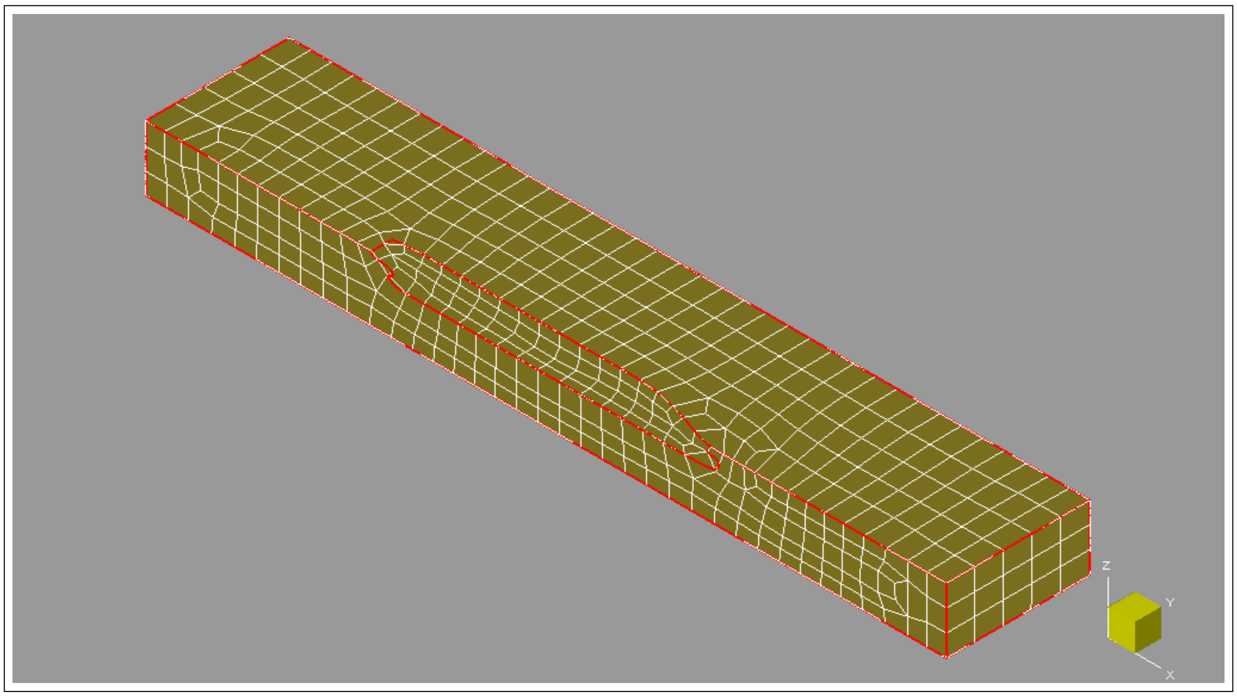


Figure C.9: Volume mesh for MOERI KCS Container Ship

C.2.2 US Navy Combatant, DTMB 5415

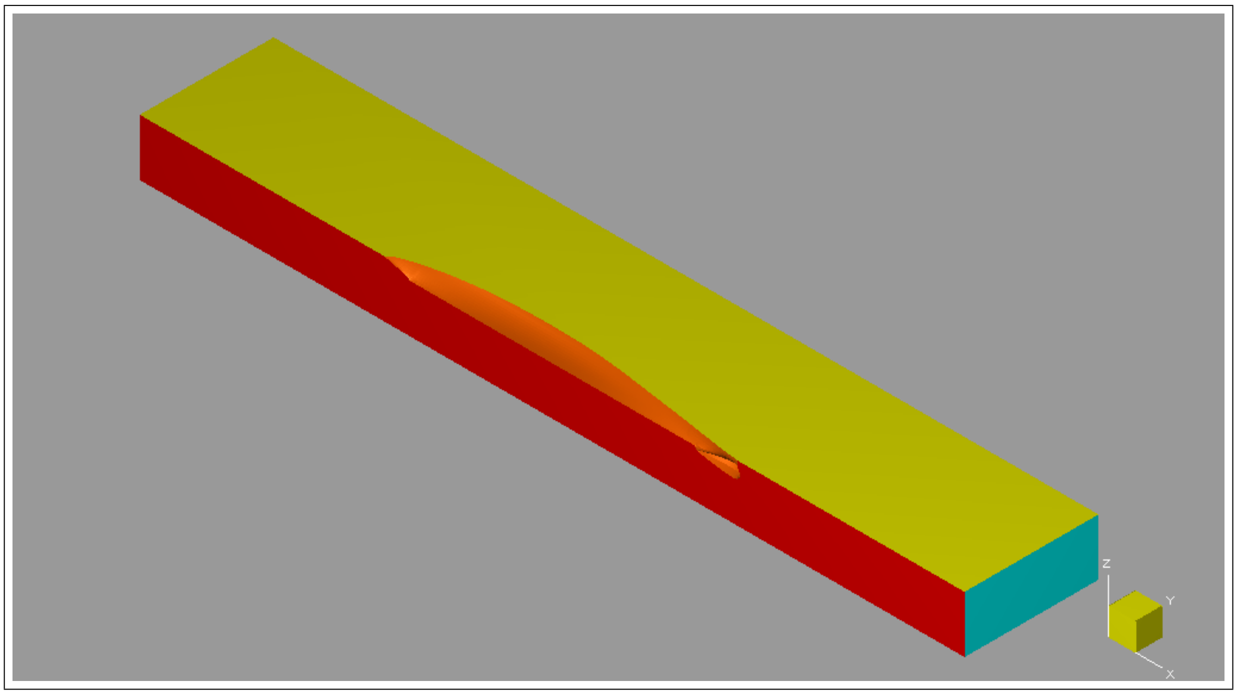


Figure C.10: Entire Fluid Domain in HEXPRESS for US Navy Combatant, DTMB 5415

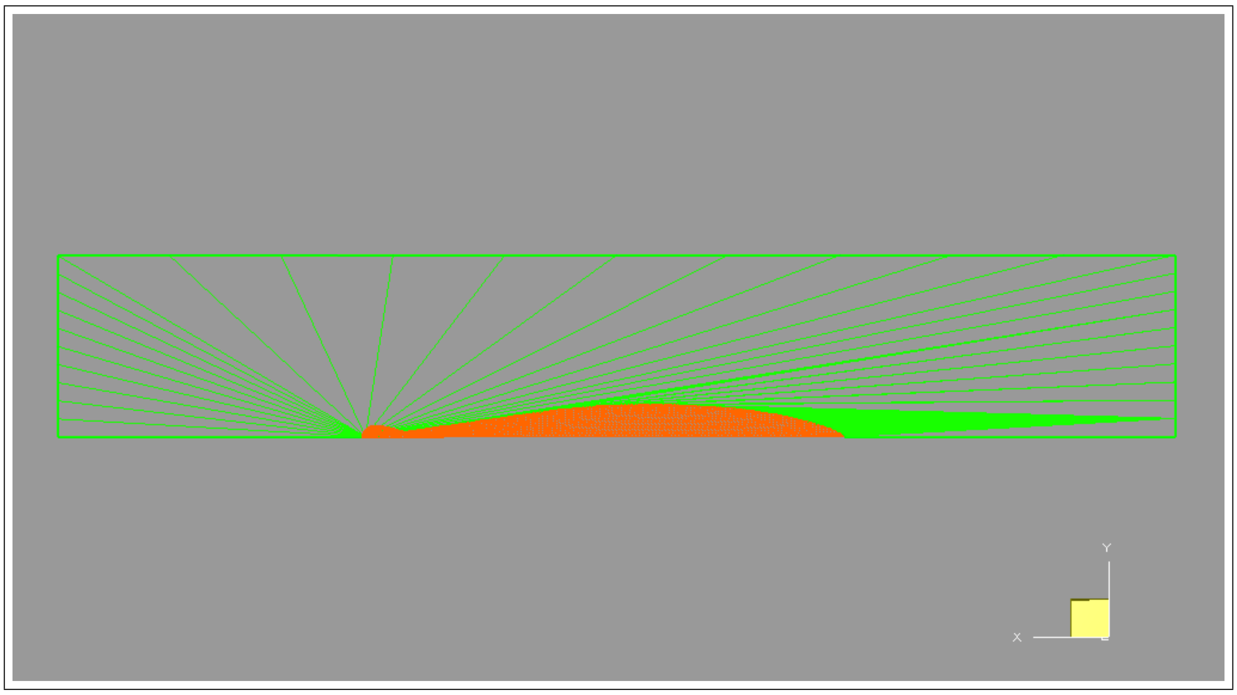


Figure C.11: Waterline Triangulation for US Navy Combatant, DTMB 5415

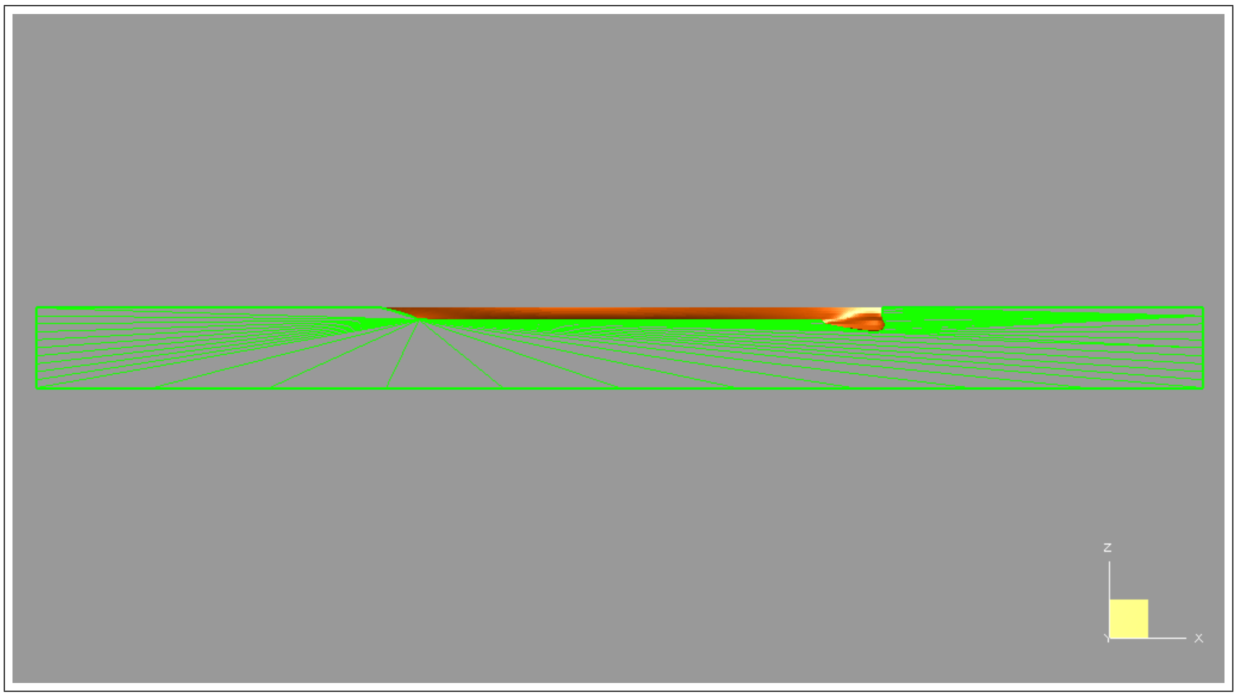


Figure C.12: Centerline Triangulation for US Navy Combatant, DTMB 5415

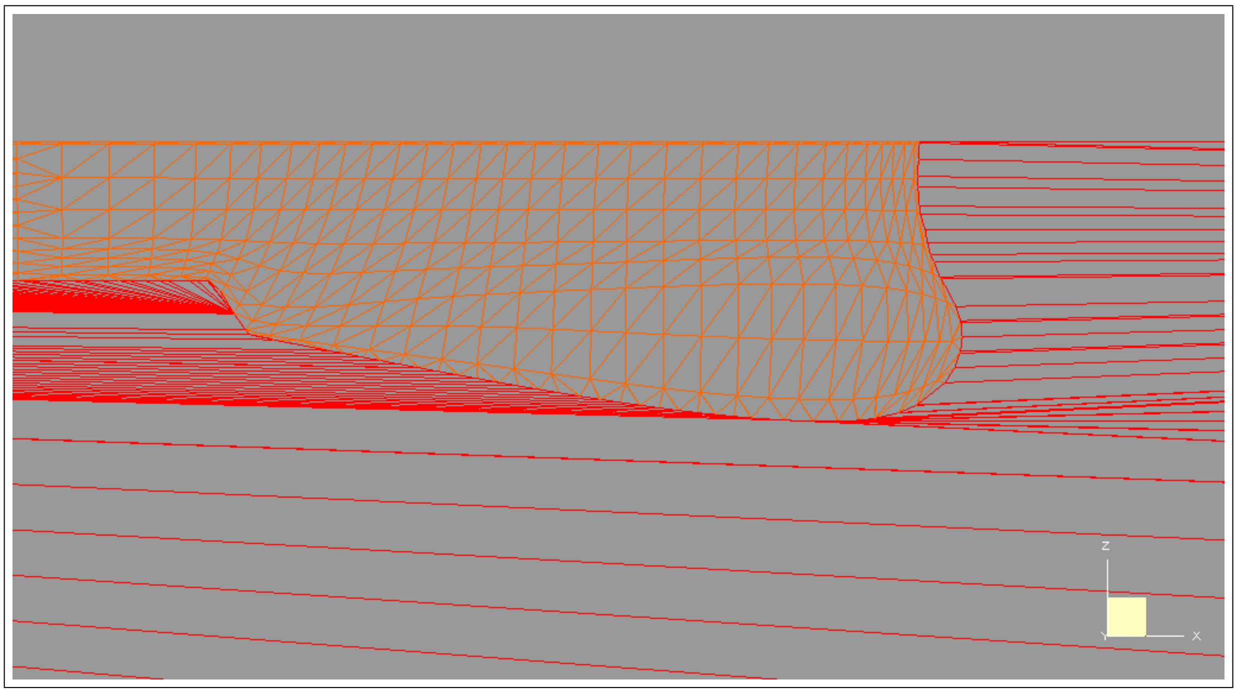


Figure C.13: Centerline Triangulation in Sonar Dome Region for US Navy Combatant, DTMB 5415

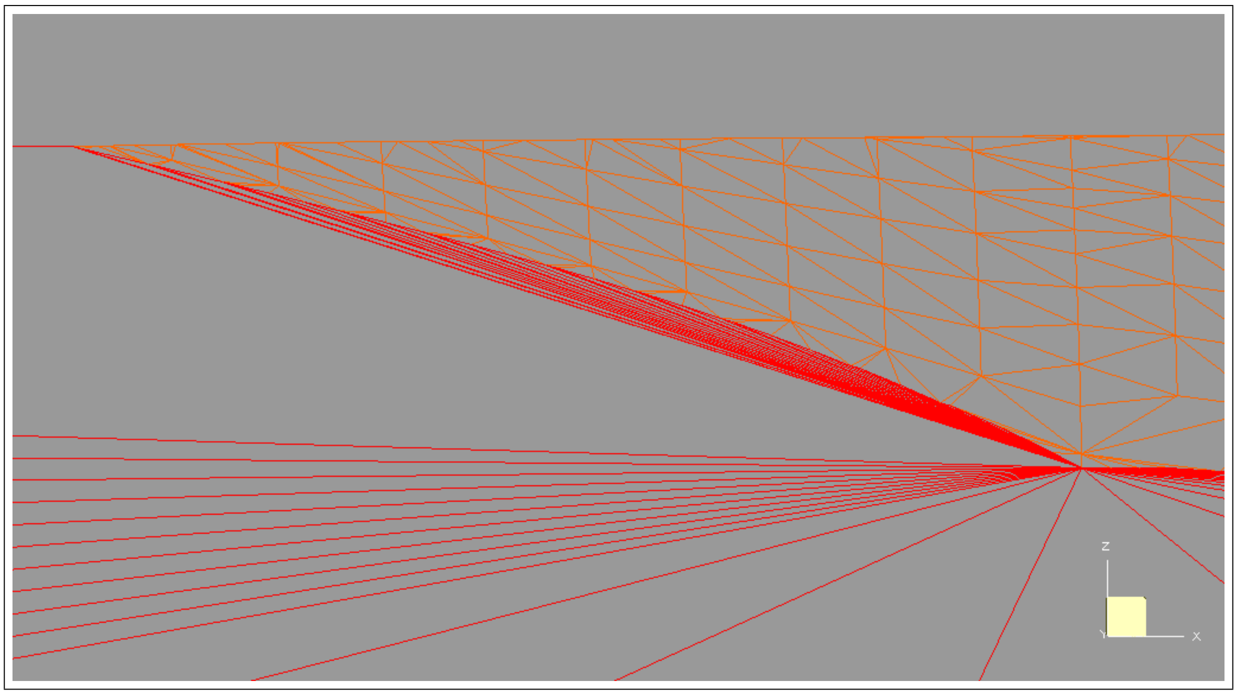


Figure C.14: Centerline Triangulation in Transom Stern Region for US Navy Combatant, DTMB 5415

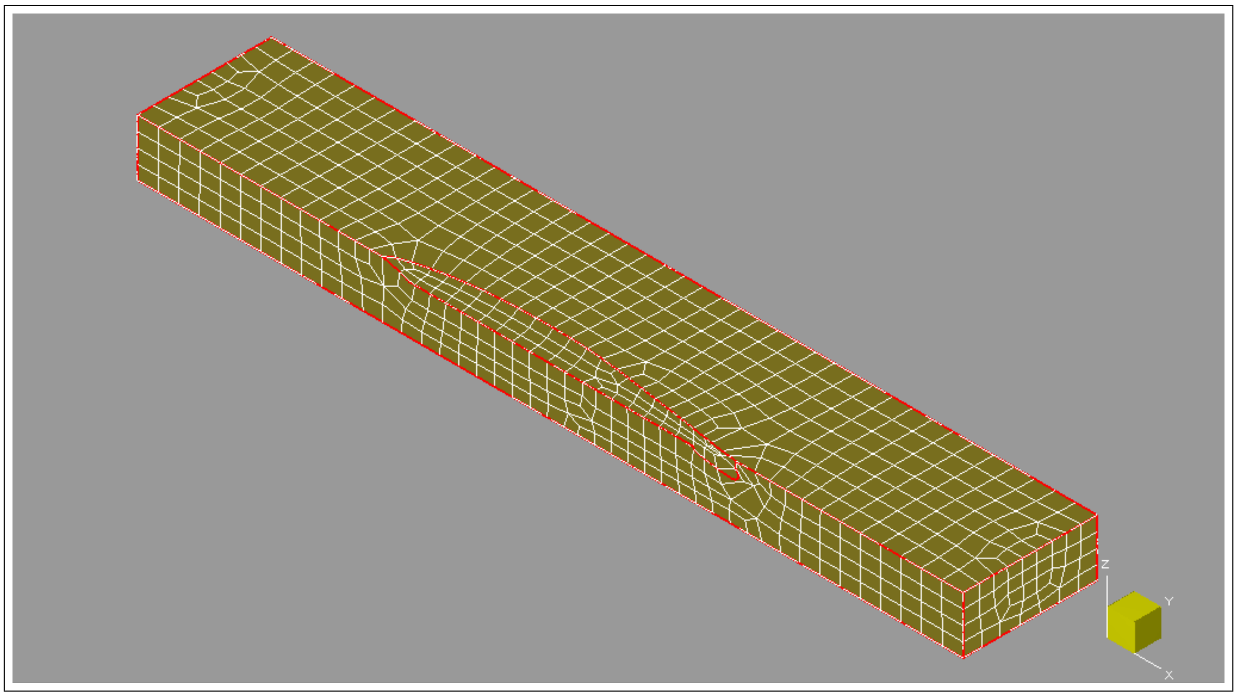


Figure C.15: Volume Mesh for US Navy Combatant, DTMB 5415

C.2.3 Sample Yacht Case

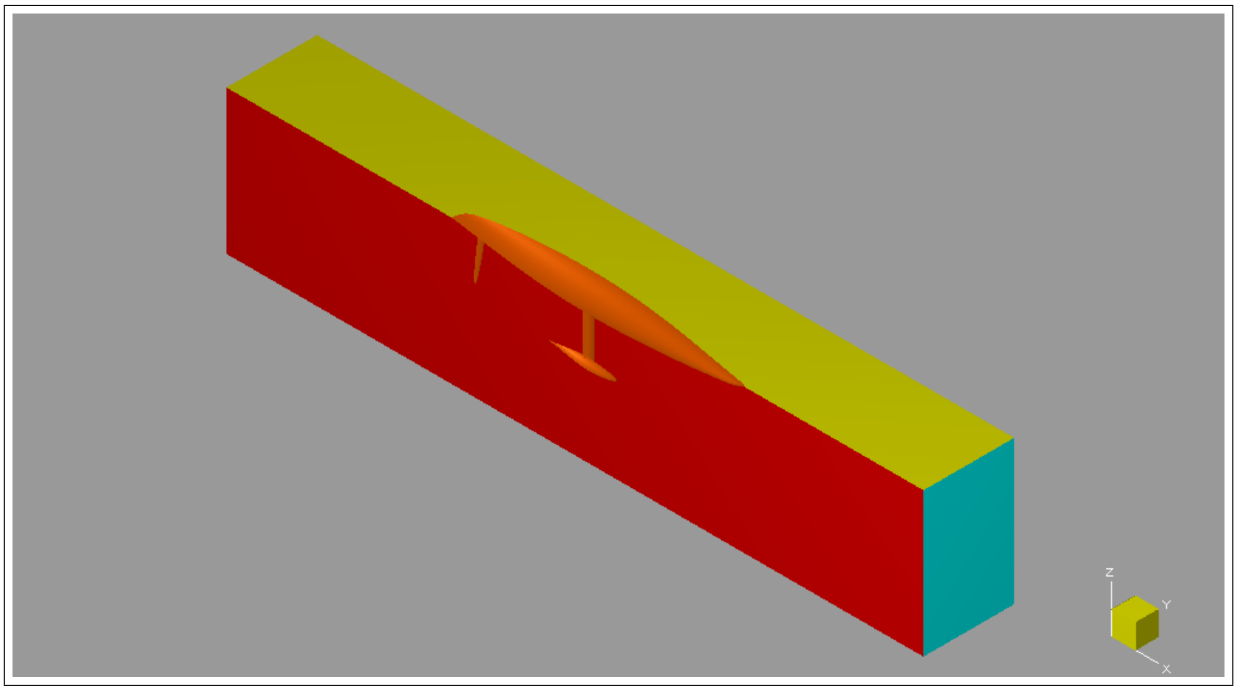


Figure C.16: Entire Fluid Domain in HEXPRESS for Sample Yacht

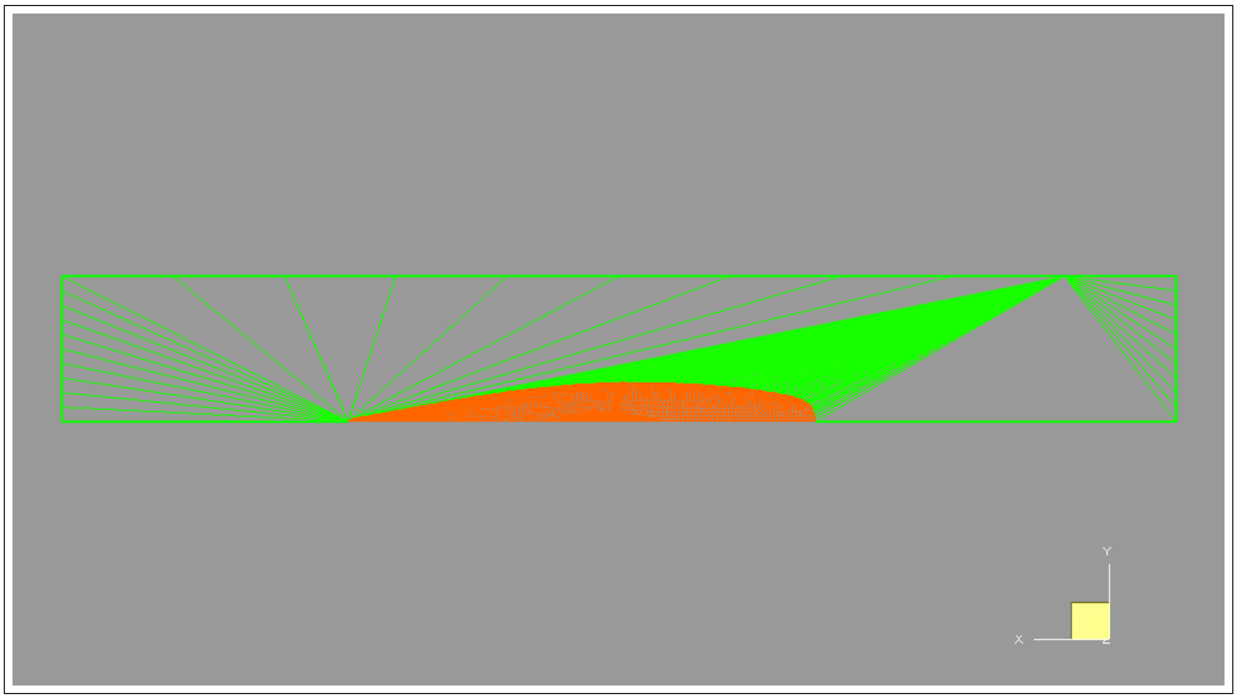


Figure C.17: Waterline Triangulation for Sample Yacht

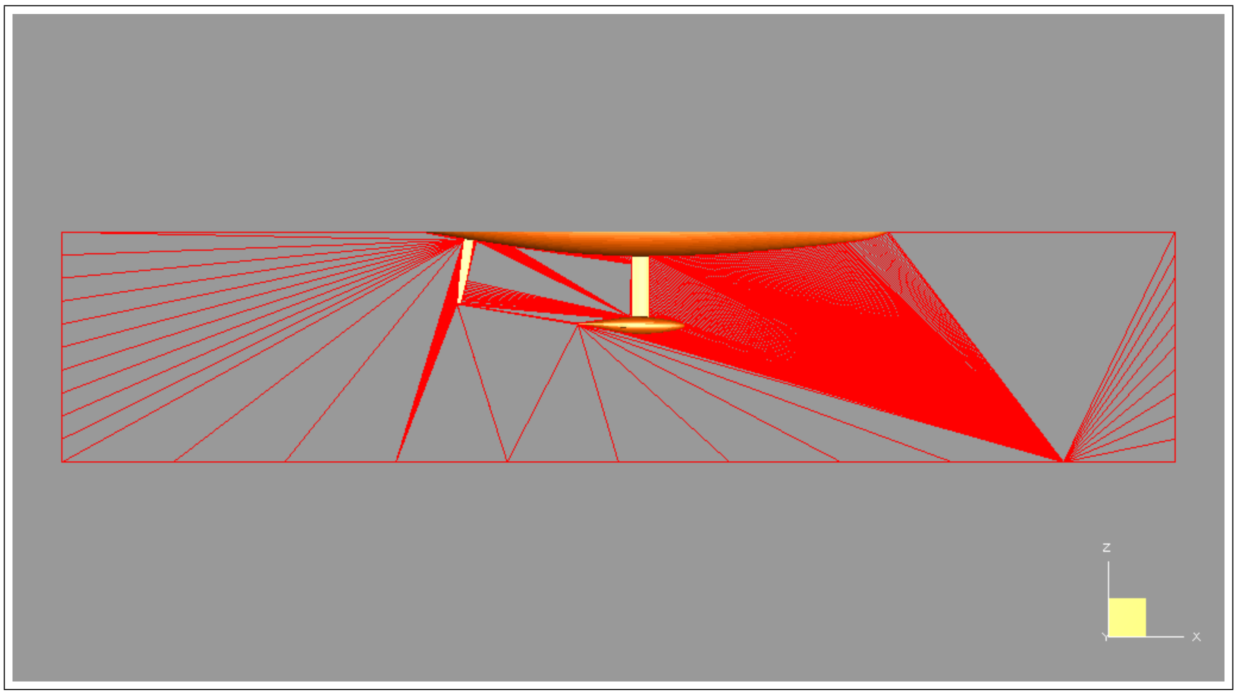


Figure C.18: Centerline Triangulation for Sample Yacht

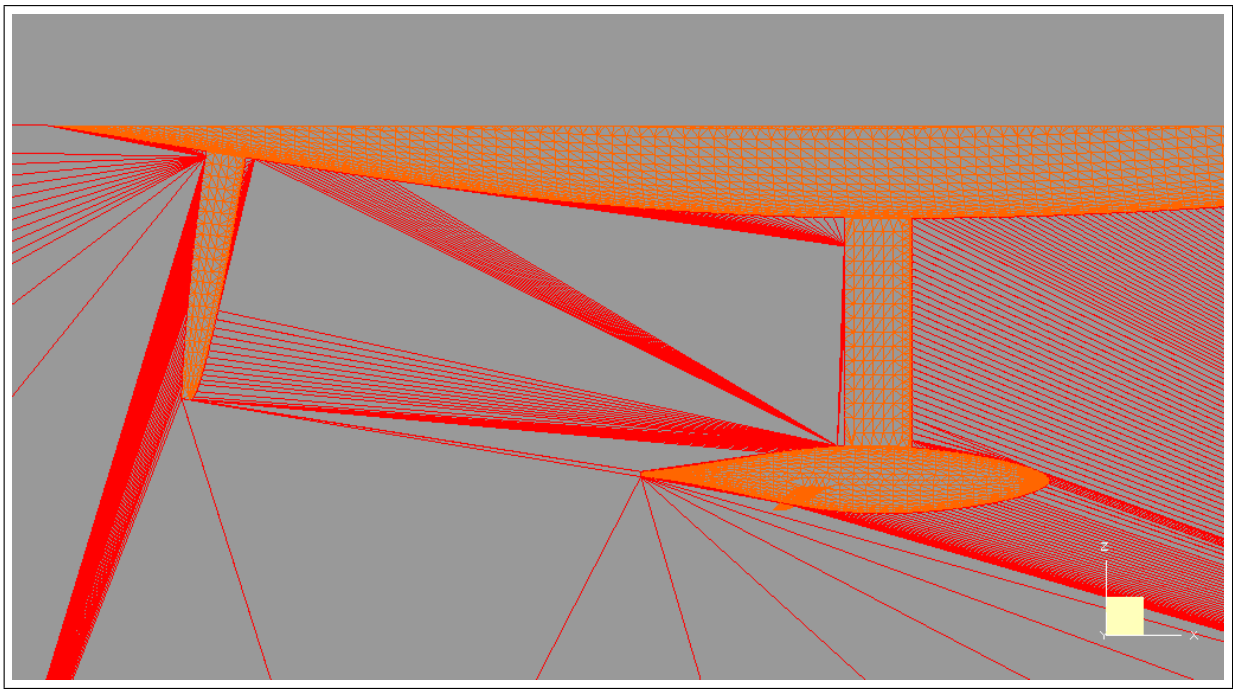


Figure C.19: Detailed Centerline Triangulation for Sample Yacht

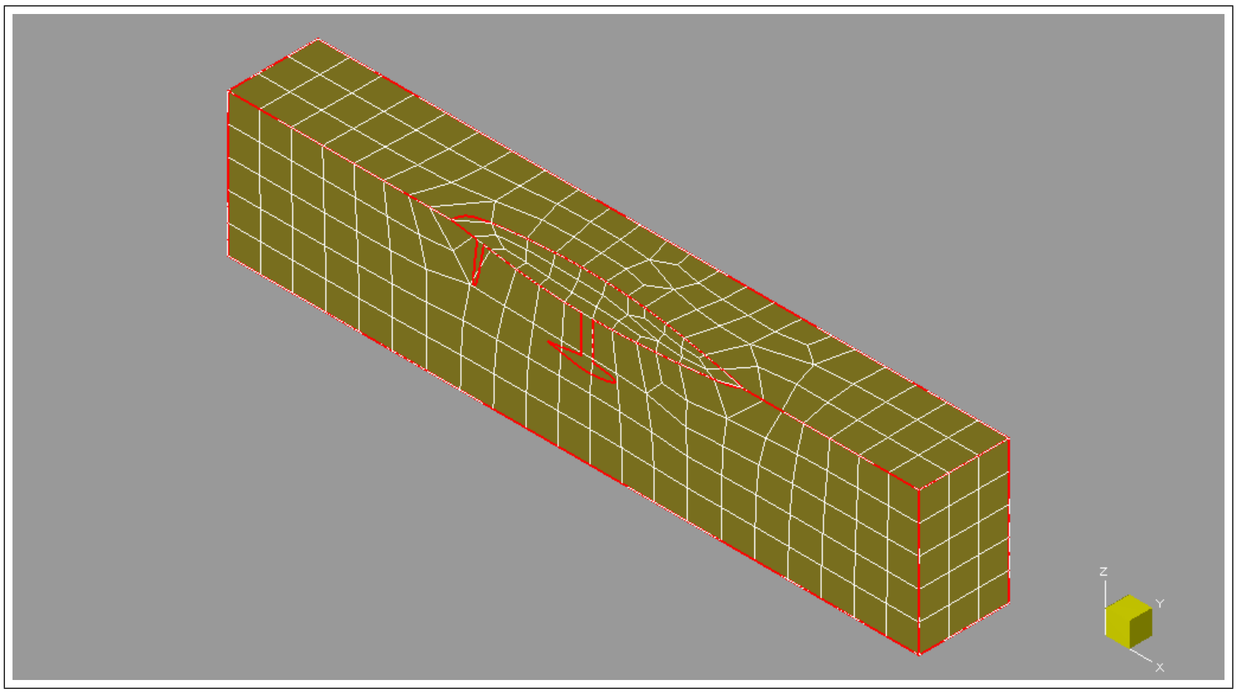


Figure C.20: Volume mesh for Sample Yacht

C.3 Testing triangulation by leaf removal for catamarans

C.3.1 NPL Hull Series

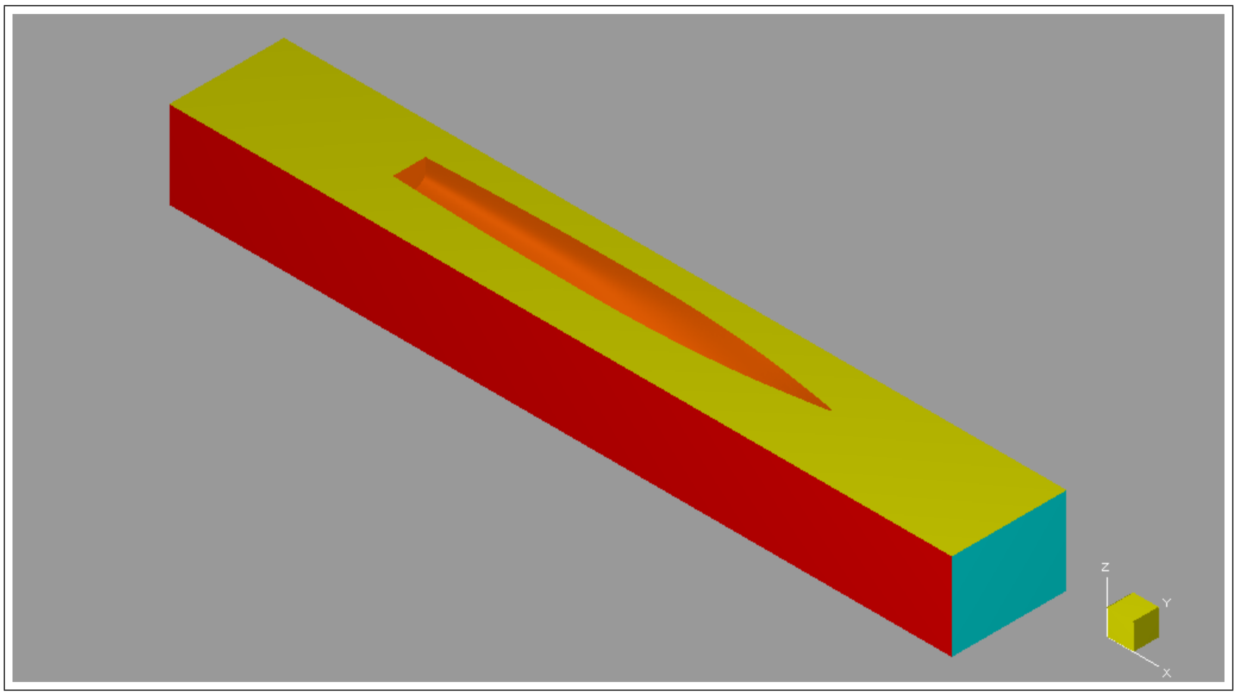


Figure C.21: Entire Fluid Domain in HEXPRESS for NPL Hull Series

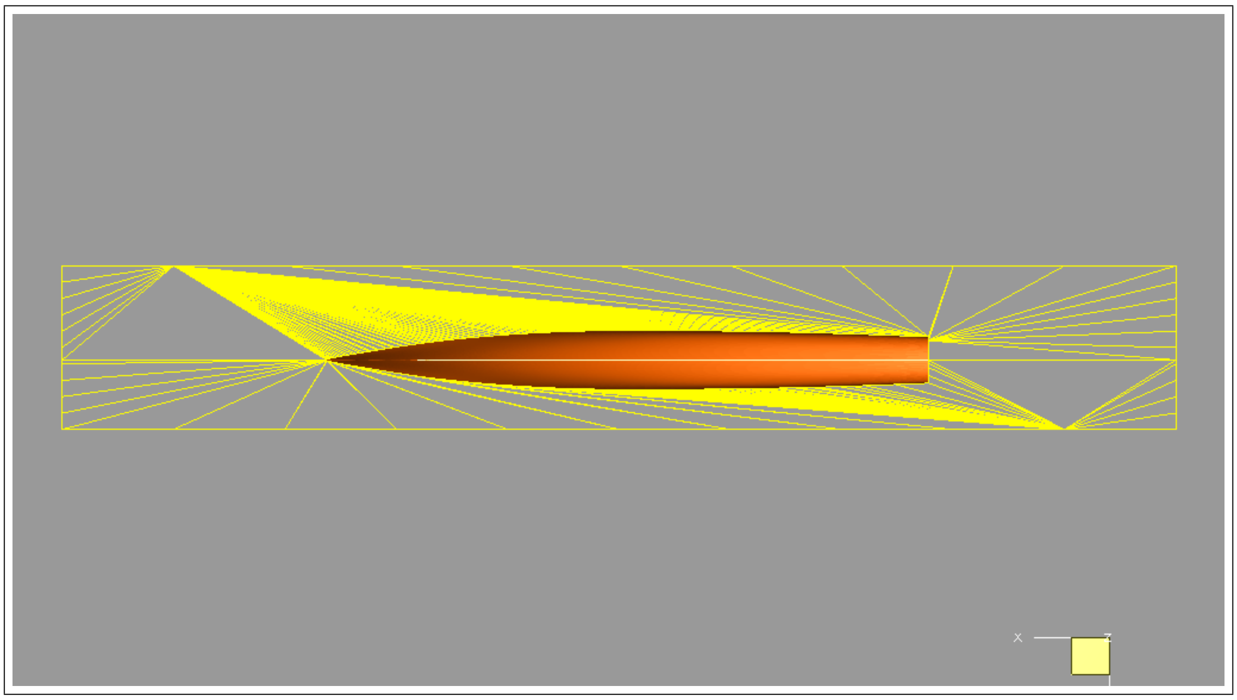


Figure C.22: Waterline Triangulation for NPL Hull Series

C.4 Testing triangulation by leaf removal for underwater vessels

C.4.1 Sample submarine

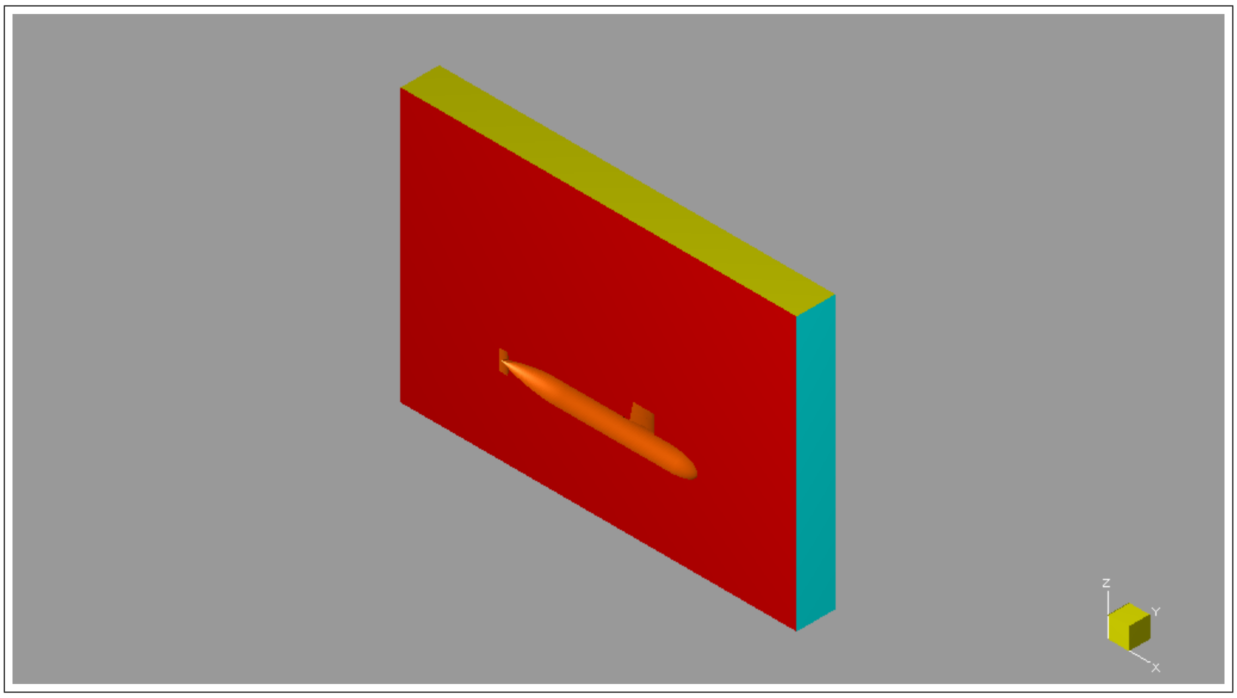


Figure C.23: Entire Fluid Domain in HEXPRESS for Sample submarine

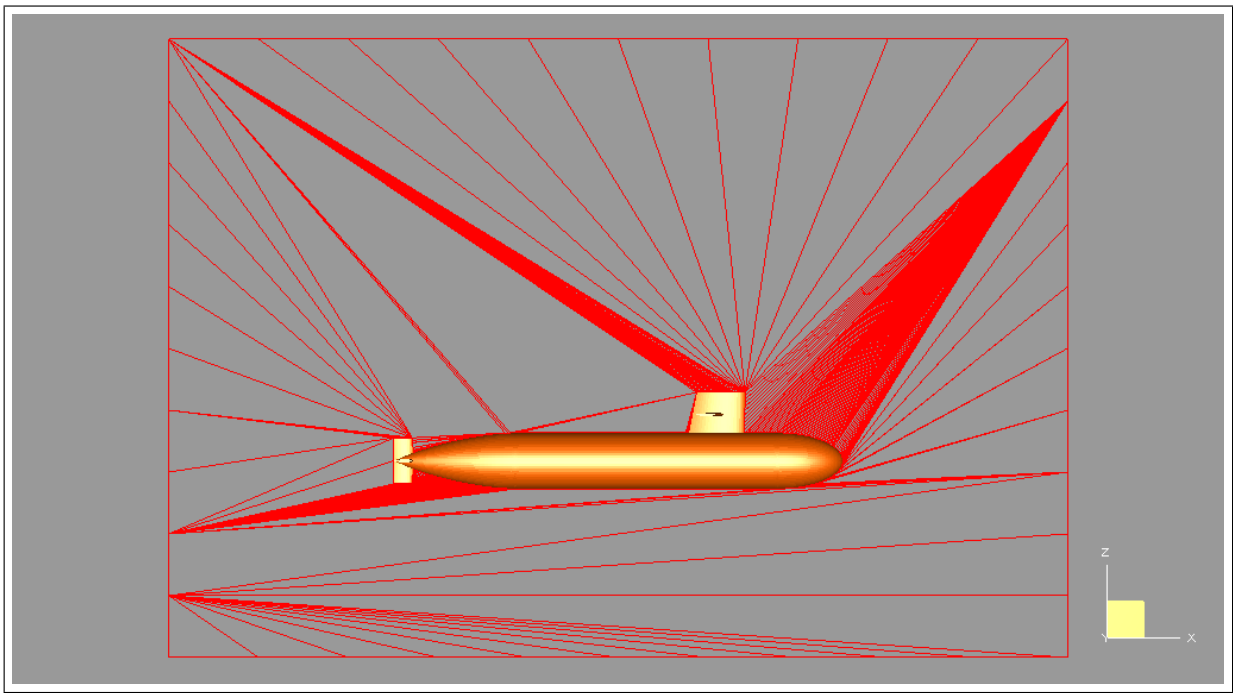


Figure C.24: Centerline Triangulation for Sample submarine

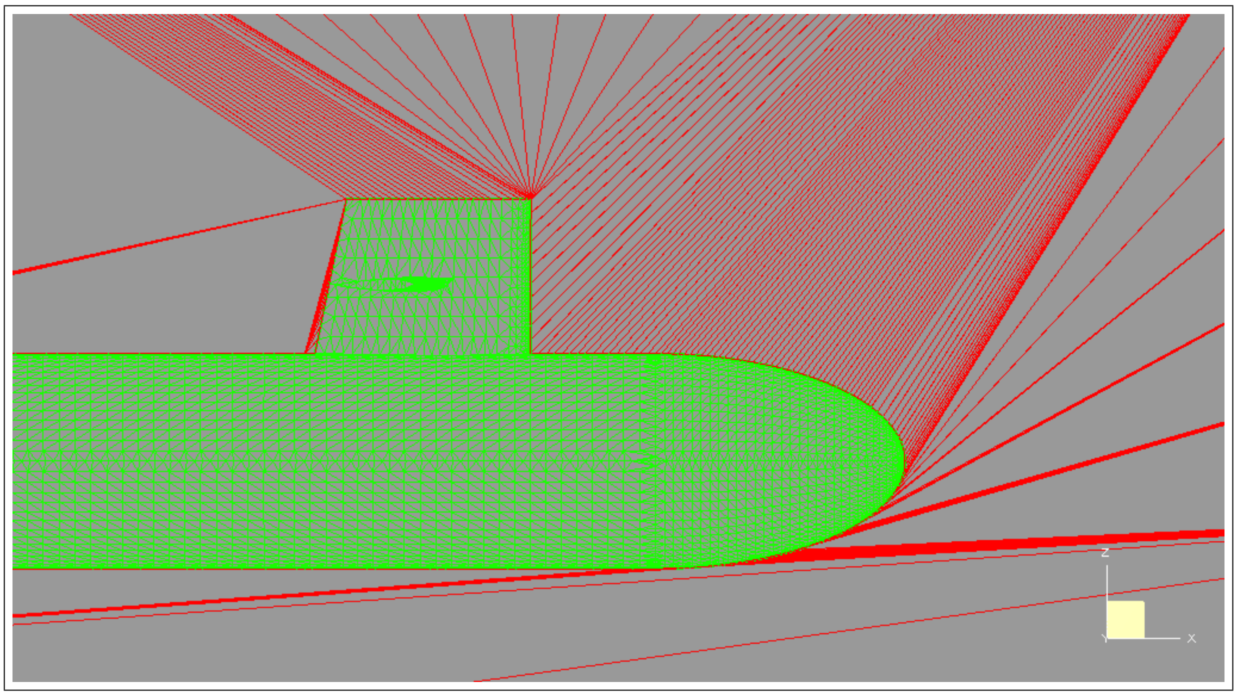


Figure C.25: Centerline Triangulation in Fore Region for Sample submarine

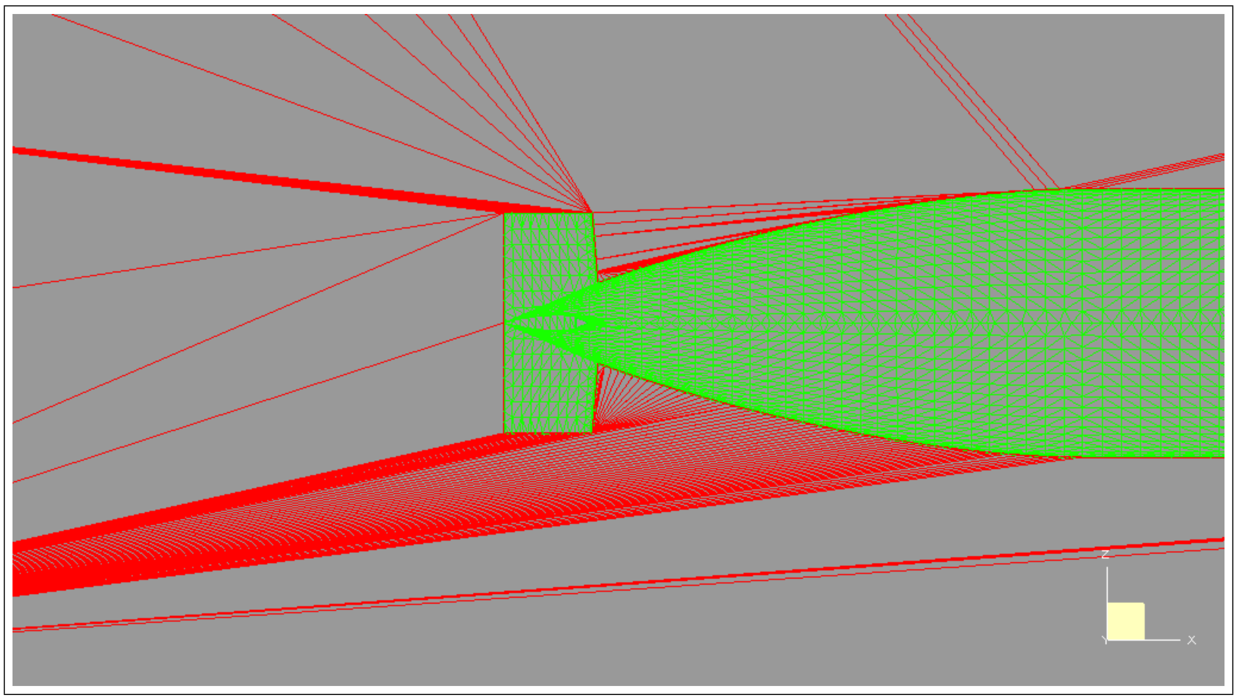


Figure C.26: Centerline Triangulation in Aft Region for Sample submarine

C.4.2 Sample yacht

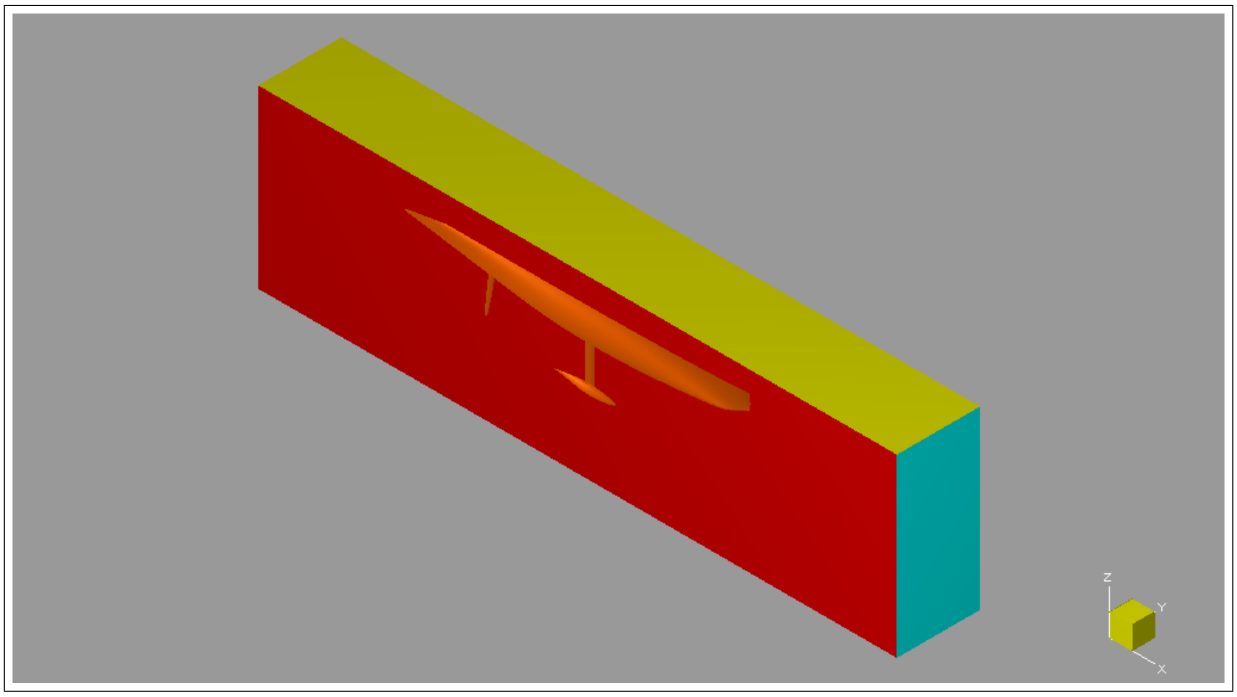


Figure C.27: Entire Fluid Domain in HEXPRESS for Sample yacht in case of underwater vessel

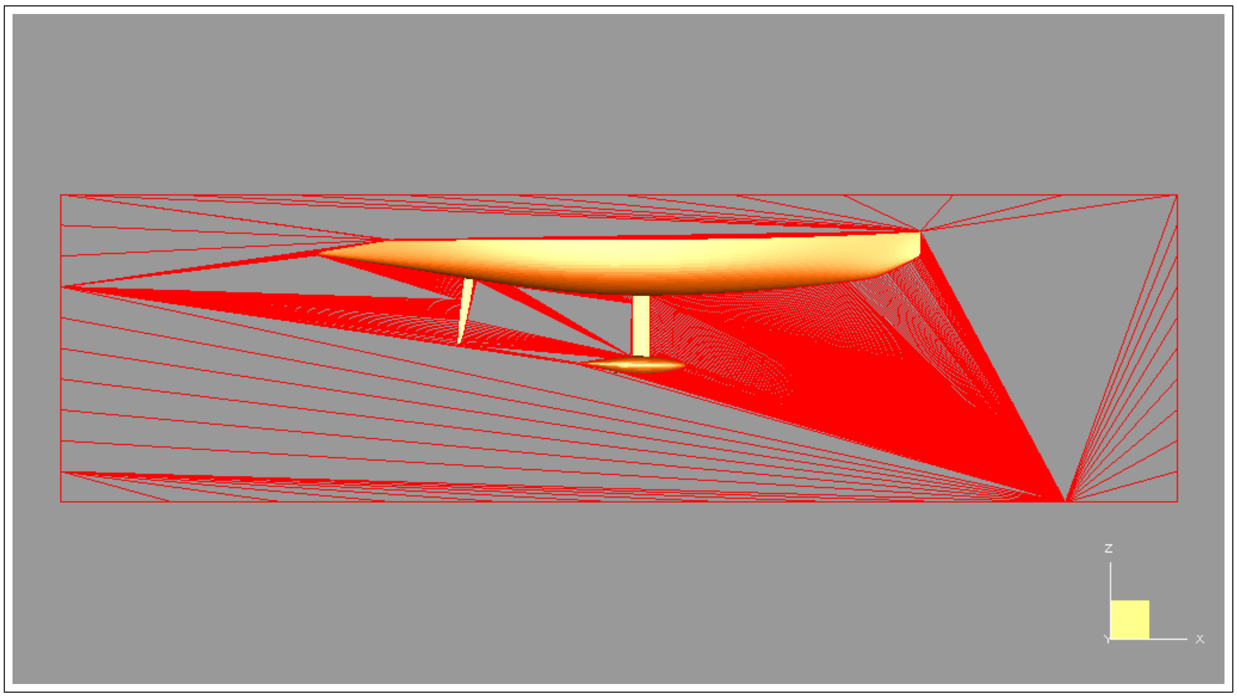


Figure C.28: Centerline Triangulation for Sample yacht in case of underwater vessel

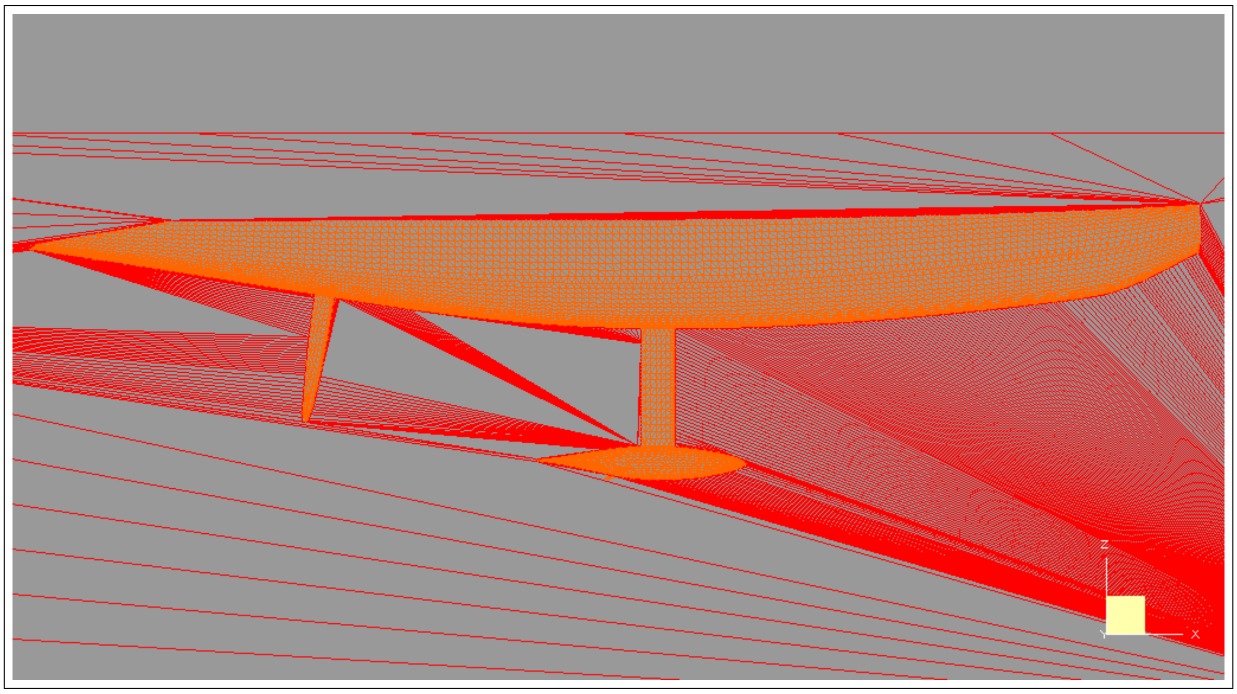


Figure C.29: Detailed Centerline Triangulation for Sample yacht in case of underwater vessel

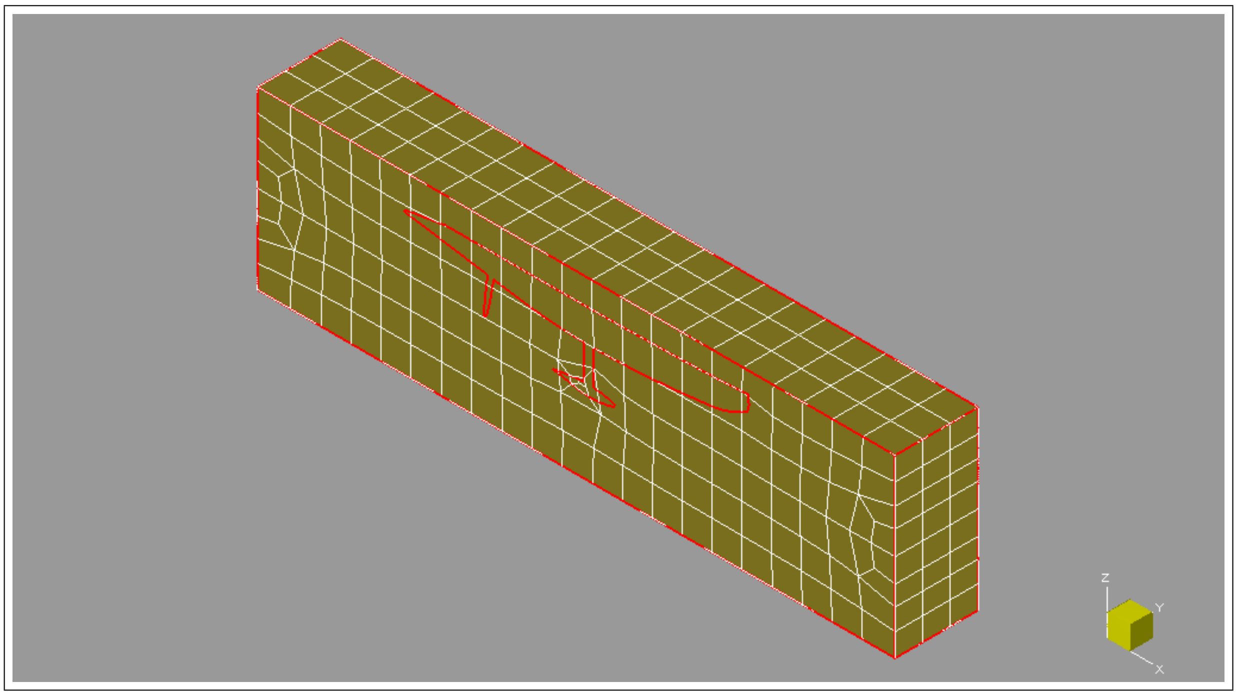


Figure C.30: Volume Mesh for Sample yacht in case of underwater vessel

C.4.3 MOERI KCS Container Ship

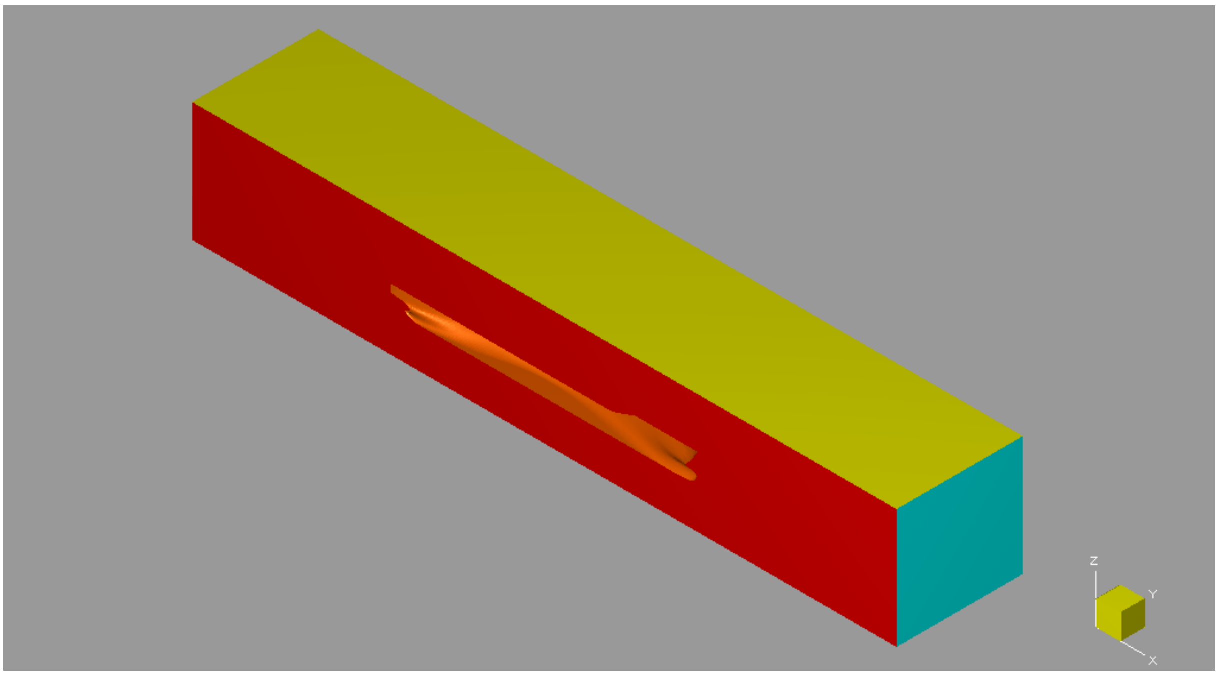


Figure C.31: Entire Fluid Domain in HEXPRESS for MOERI KCS Container Ship in case of underwater vessel

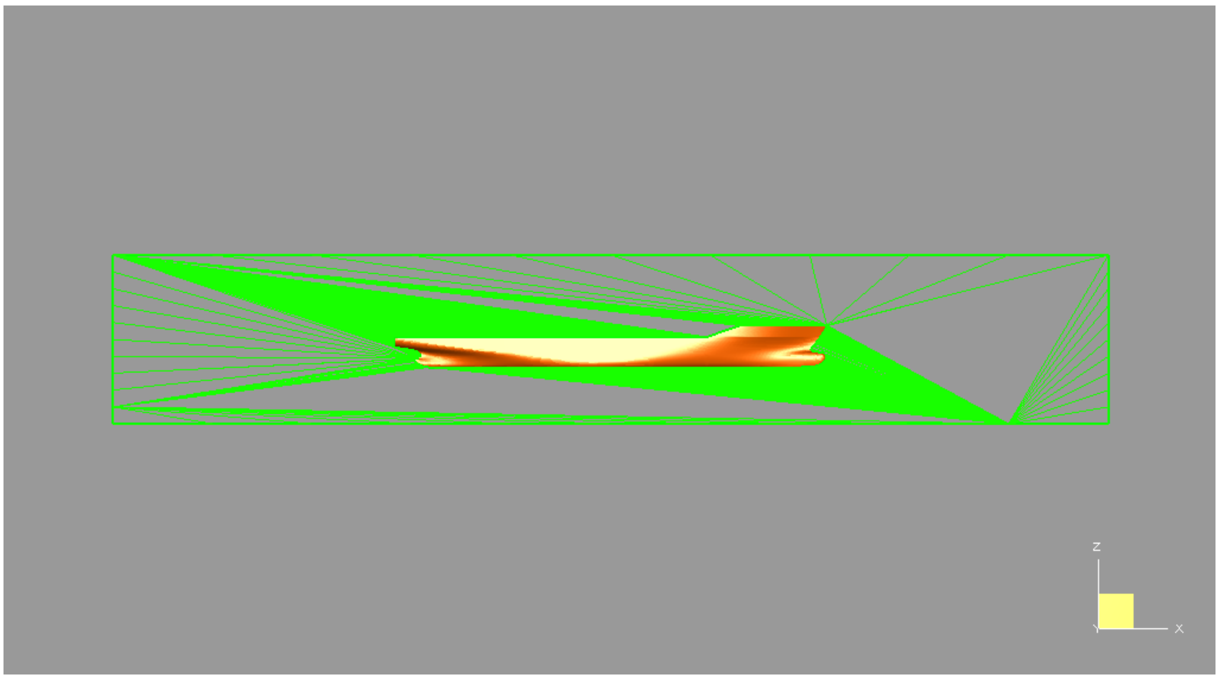


Figure C.32: Centerline Triangulation for MOERI KCS Container Ship in case of underwater vessel

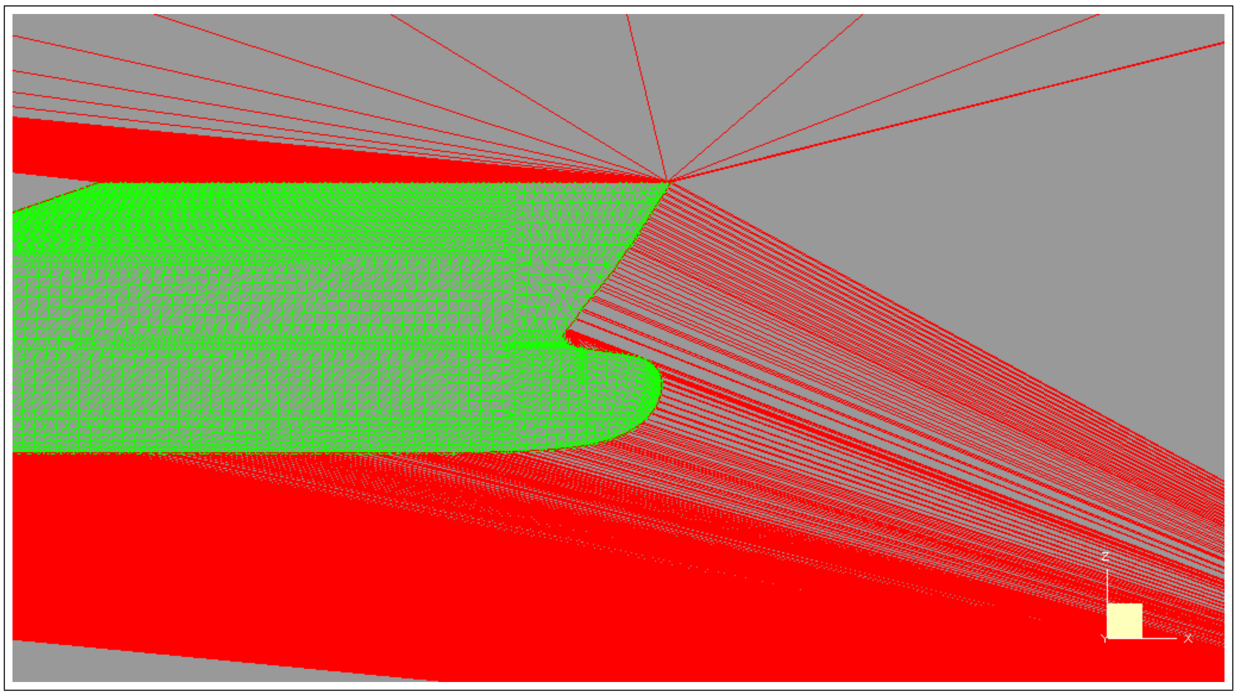


Figure C.33: Centerline Triangulation in bulbous bow region for MOERI KCS Container Ship in case of underwater vessel

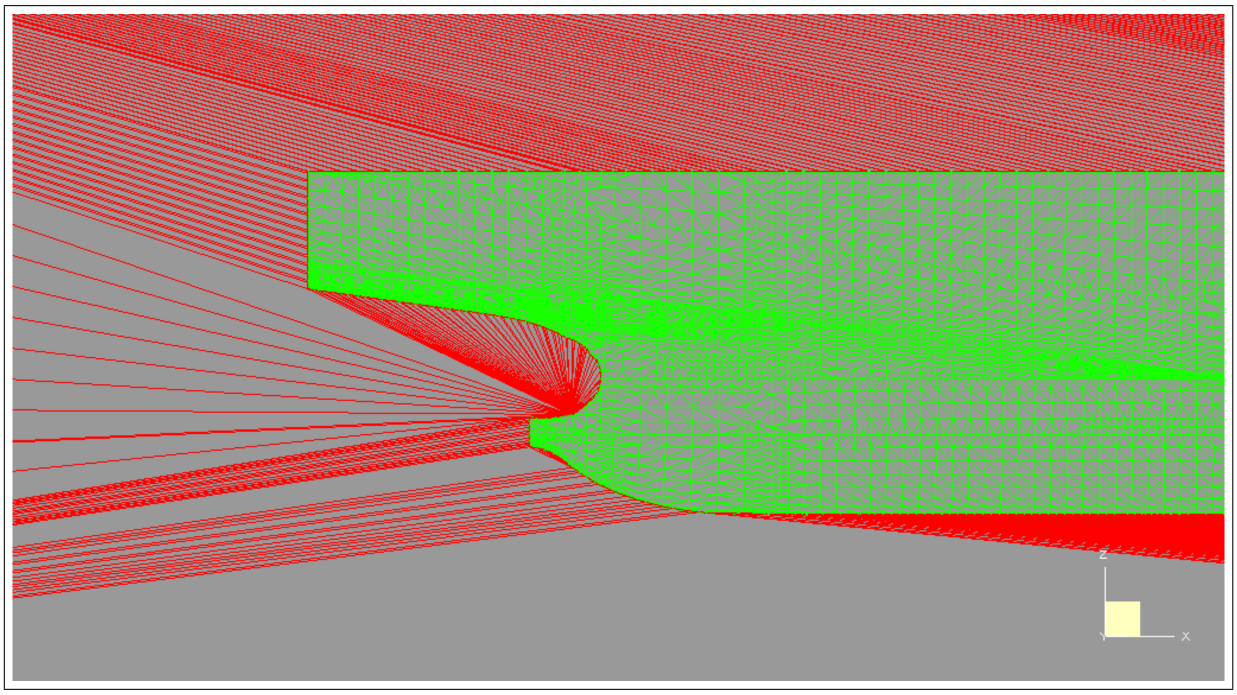


Figure C.34: Centerline Triangulation in transom region for MOERI KCS Container Ship in case of underwater vessel

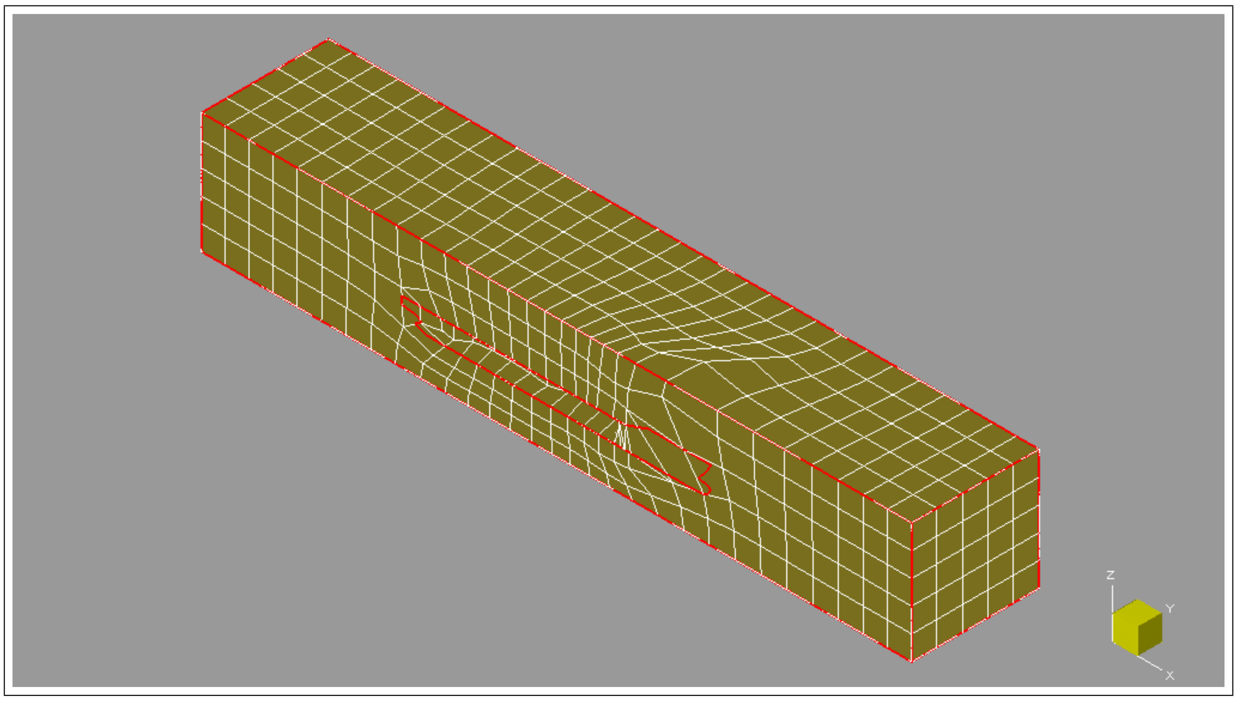


Figure C.35: Volume Mesh for MOERI KCS Container Ship in case of underwater vessel