Sketched Parametric Modeling in CFD Optimization

Martin Alexander Barrios Gundelach

Master Thesis

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

developed at University of Rostock
in the framework of the

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

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

Supervisors:	M.Sc. Sebastian Greshake, University of Rostock
	Prof. Robert Bronsart, University of Rostock
Internship Supervisor:	Dr.-Ing. Stefan Harries, Friendship Systems
Reviewer:	Prof. Zhe Li, École Centrale de Nantes

Rostock, January 2017

ABSTRACT

In the field of shape optimization for CFD computations, users of geometry-modeling software normally have a wide variety of tools to choose from in order to parametrically design a hull from scratch and build a Fully Parametric Model (FPM). However, in practice most hull models are initially designed with no parametrization, and are then exported and imported between different software, normally forcing a subsequent required optimization to be performed in a Partially Parametric Model (PPM). These PPMs can sometimes be too complex for an average designer to build, since he may not be familiar with the mathematical constraints needed for the application of the required transformations, or the particular tools for given software to be used for the parametrization. In this context, a solution that can be introduced is sketched parametric modeling, which is the combination of complex geometry-modeling operations into intuitive, simple and user-friendly tools. By experience or input from adjoint/shape sensitivity analyses, designers may know where they would like their hull to be modified in order to improve CFD performance, and with the help of sketching, may parameterize those changes quickly to be input for a CFD optimization. This thesis presents an application of sketched parametric modeling in the parametric geometry-modeling and CFD integration software CAESES, and further use of it in a hull shape resistance optimization case using CFD viscous flow software FINE/Marine.

Keywords: sketched parametric modeling, parametric modeling, CFD, hull shape optimization, ship resistance.

DECLARATION OF AUTHORSHIP

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

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

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

I have acknowledged all main sources of help.

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

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

I cede copyright of the thesis in favour of the University of Rostock.

Date: _____ Signature: _____

ACKNOWLEDGEMENTS

Making this thesis would not be possible without the aid of my supervisors, Dr.-Ing. Stefan Harries from Friendship Systems, and M.Sc. Sebastian Greshake and Prof. Robert Bronsart from University of Rostock. I am thankful for their support. The help of Mattia Brenner, Carsten Feutterer, Claus Abt, Joerg Palluch, Carlo Pasquinucci, Willy Maschen, and my other colleagues at Friendship Systems during my internship period was also of great importance, and for that I am thankful. Thanks also to CPU 24/7 and NUMECA Ingenieurbüro Germany for providing me with support and resources to run the calculations needed for the optimization performed here.

I thank all of my EMship friends, who made this course a pleasant experience in my life. A big "thank you" also to Prof. Rigo, Ms. Emna Belaid and especially Ms. Christine Reynders, for their help and support to me and my colleagues during this course, going far and beyond to assist us with everything we needed and then some.

I thank here all the people that helped me walk the path that led to the conclusion of this chapter of my life, be them family, friends, or acquaintances. None of you are forgotten, rest assured. You are all part of who I am, and I will carry the lessons I learned from you for all my life. People are the reason I am here, and people are the reason I go on.

I know it may be a little silly to do it here, but I thank the European Union for creating and managing the Erasmus Programme. It is the largest student exchange program in the world, it takes a lot of effort and investment from the European people, and I do not take it for granted. This program has positively changed the lives of millions of people with the noble objective of extending and developing education in Europe and the world. I am proud to take a part on it.

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

"I know one thing: that I know nothing." – the Socratic Paradox

GLOSSARY AND CONVENTIONS

In general, symbols and abbreviations are given when they first show up in the text, and many others are commonly used in the Naval Architecture field; nonetheless, the following list shows some of the ones used in this thesis.

CFD: computational fluid dynamics;

CAD: computer-aided design;

R&D: research and development;

FPM: fully parametric model;

PPM: partially parametric model;

FFD: Free-Form Deformation technique;

DoF: degrees of freedom;

FV: finite volumes;

STL: stereolithography file format (a.k.a. standard triangle language);

TriMesh: triangular mesh.

Units throughout the thesis are all in the SI system, and the coordinate system used for ships has x+ to fore, y+ to portside, and z+ upwards.

TABLE OF CONTENTS

1. INTRODUCTION AND LITERATURE REVIEW

The field of Computational Fluid Dynamics (CFD) has grown considerably in the last few years, with the improvement and development of an abundance of methods and solvers for different fluid physical phenomena. Along with it, geometry manipulation Computer Aided Design (CAD) tools have had to increase their capabilities for modeling an ever growing amount of different scenarios for simulation. The high level of complexity quickly reached by each of these tools has led some designers in the industry to be successful at one of them while lacking practice at the other. This, together with the normal workflow specialization applied in the industry, has led to difficulties in analyses requiring input from both sides.

Such gaps in knowledge and communication between CAD designers of hull geometry and CFD engineers can be seen, for example, in the difficulties surrounding hull shape optimization processes for hydrodynamic purposes. In many industrial cases, hull geometries are, as tradition, made by CAD teams before they reach CFD engineers, who may then want to change the input geometry but might have their own problems doing that in CAD environments. An alternative to this traditional way has been the so-called Upfront CAD approach (Brenner et al., 2015 [1]), which proposes the hull geometry to be parametrically modeled as early and as readily as possible to be prepared for further simulations and required variations. The impact of this approach may be quite significant in a design process that requires model variants; the following schematic figure shows this expected impact by expected designer (user) effort.

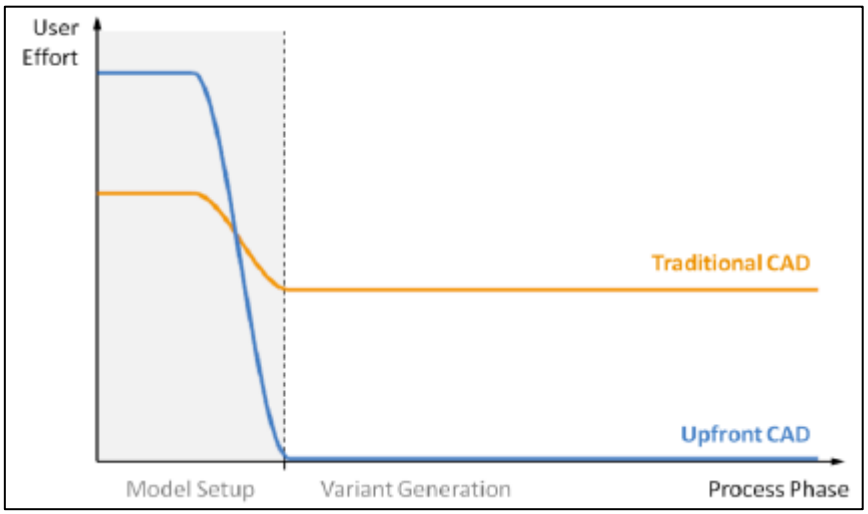

Figure 1-1 – Traditional vs. Upfront CAD in processes with model variants (taken from [1])

Although there is a current trend towards earlier employment of parametrization, most models are still built in the traditional way, leaving CFD engineers to try to work through the issues found in modeling, sometimes with the added difficulty of exporting/importing geometry in different formats between different programs. Some software, like Friendship Systems' CAESES, have the available tools for modifying geometry for simulation, connecting and running the model with CFD solvers, and post-processing results. But the complexity of the mathematics required for the transformations to be imposed may sometimes take too much time and effort for such users.

This thesis comes in this context presenting methods for sketching modifications in existing geometries and quickly making them available for simulation and variation processes such as CFD shape optimization.

1.1. Parametric Modeling and Shape Optimization in CFD

When building a geometry for CFD hydrodynamic analyses, designers traditionally model its shape by using sets of well-known CAD tools that are controlled by points in the model space. This method is flexible enough to allow producing any possible shape, providing designers with a model that is as realistic as desired. However, if variants of such geometry are meant to be produced as part of an optimization process at some (or various) points of the design, which is frequently the case, doing it can be extremely time-consuming due to the way its shape was modeled: each constructive vertex/node of the geometry would represent a degree of freedom (DoF) that would be a design variable. It is easy to see that the more complex the shape of the model, the higher the number of DoF and computational cost of the optimization.

As an alternative to traditional modeling, designers can parameterize the whole geometry into a small number of parameters that regulate all of its characteristics and take into account design constraints, thus building a Fully Parametric Model (FPM). This procedure can greatly reduce the number of DoF of a model to the number of design parameters and allow an efficient creation of geometry variants, and therefore an efficient optimization process.

Difficulties in applying parametric modeling can come mainly from high complexity in a geometry's definition. For the case of hydrodynamic analyses, however, model geometries frequently are deliberately simplified up to levels that can be captured by the solvers and mesh refinement levels to be used. Even when dealing with a complex geometry such as a ship hull,

a Partially Parametric Model (PPM) can be built to parametrically deform a specific region of the object, instead of the whole of it. This way, the use of parameter-based geometry modeling for optimization can be justified in many hydrodynamic problems.

Lackenby (1950 [2]) can be recognized as the first to build a meaningful systematic (parametric) geometric variation method on ship hull forms. His method consists in moving transversal sections of the ship in the longitudinal direction to reach a desired sectional area curve and longitudinal center of buoyancy (LCB). Goal-seeking techniques can be used to quickly find the deformations needed for each section. Since CFD was not well developed nor possible to be executed in that time, it is only natural that the objective functions were simply geometric ones. The method is still used to this day in the industry with improvements and variations.

The development of shape optimization in CFD has greatly grown with the advance of computer power in the last decades. Parametric shape optimizations for CFD improvement were performed initially for the aeronautics industry, such as in the work of Hicks et al. (1974 [3]). Reasons for the late use of parametric shape optimization in the shipbuilding industry can be speculated to come, for example, from the different design objects' bulk costs (plane designs are cheaper and are built in more numbers than ship designs), or the differences in their geometry complexities (plane wing profiles have long been represented in fully parametric forms).

The works of Nowacki et al. (1977 [4], 1990 [5]) and Nowacki & Lu (1994 [6]) in modeling ship hull form lines by the use of multiple form-parameters was investigated by Harries & Abt (1997 [7]), who proposed an optimization process for reaching desired geometric objective functions using such parameters. This led to the work of Harries (1998 [8]), who investigated the use of parametric design in shipbuilding, gave examples of tools to be used in transformations for fully- or partially-parametric model design, and made the combination of it with a CFD optimization.

The European Commission funded in 2000 the joint industry R&D project FANTASTIC (Functional Design and Optimization of Ship Hull Forms), with the objective of improving ship design by use of parametric shape modeling and state-of-the-art CFD analysis tools to predict ship hull performance (Maisonneuve et al., 2003 [9]). The project resulted in significant improvements in the tools available for parametric modeling of ships, while simultaneously bringing the spotlight to its potential combined use with CFD simulations and shape optimizations. The work of Hoekstra & Raven (2003 [10]) in the project applied some of those tools in a

successful hull shape optimization using a potential flow solver, highlighting its possible use in any CFD cases.

Since then, simulation-driven parametric design of ships has been the subject of multiple works on research, development, and innovation, such as the ones from Harries et al. (2004 [11]), Abt & Harries (2007 [12]), Harries (2008 [13]), Wilson et al. (2010 [14]), Biliotti et al. (2011 [15]), and Han et al. (2012 [16]).

More recently, the works of Weickgennant et al. (2014 [17]) and Brenner et al. (2015 [18]) have proposed the use of parametric modeling for optimizing shapes based on results from the Adjoint CFD method, in a so-called Parametric-Adjoint Method. The Adjoint method provides the normal sensitivities of points on the surface of an object to a given flow parameter result (e.g. drag forces on a hull); in other words, it gives the gradient of an objective flow function to geometric changes, from the base geometry. Results from parametric-adjoint solutions provide CFD engineers with the sensitivities required for the design parameters of a fully- or partially-parametric model, greatly facilitating the use of gradient-based optimization methods in problems with numerous design variables.

Harries et al. (2015 [19]) presented an updated overview of the different available methods and approaches for parametric modeling and their usefulness when combined with CFD optimization processes. The following schematic chart shows the relation of the parametric modeling approaches and their flexibility on creating desired geometric shapes.

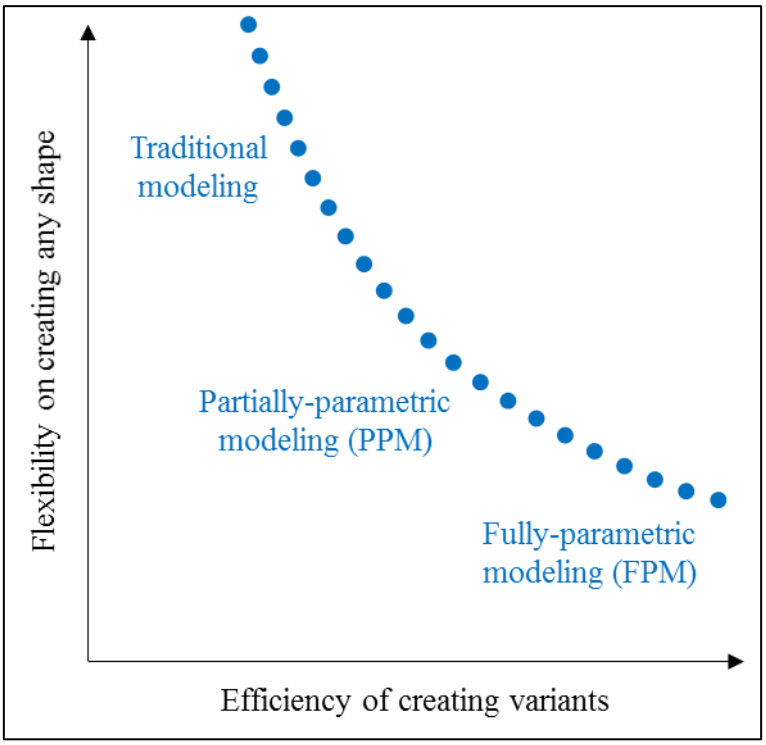

Figure 1-2 – Flexibility of parametric modeling (adapted from [19])

It is visible from the chart that FPM and PPM models require more effort from traditional designers; it may greatly pay off, though, if the process is expected to require many variants, such as in optimization processes for CFD.

1.2. Sketched Parametric Modeling

Although research has shown great potential in the Upfront CAD approach and simulation-driven design, and a general trend exists now towards earlier development of parametric models, the industry practice is still adapting and is still largely traditional in its methods. Observing this issue, this thesis investigated methods for sketching modifications in existing hull shapes such that users would have simpler ways of tackling desired transformations in existing geometries and running CFD shape optimizations based on them. However, regular partial parametric modeling requires users to create complex transformation geometries and/or mathematical definitions, which, as previously discussed, can bring difficulties to users. Therefore, new tools were idealized in a methodology which had the idea of giving users a direct access from the source geometry to the desired final outcome, hiding the most complex parts so as to bring intuitiveness to them. This user-oriented method was called *sketched parametric modeling*, or simply *sketched parametrics*.

The term "sketched parametric modeling" seems to not have been yet definitely coined by any science research field, but a few references to it or to sketched modeling have been found in some papers and articles in the field of computer graphics and animations, such as Kho & Garland (2005 [20]). In them, these terms have been normally associated with quick, easy, user-oriented and intuitive interactive deformations of computer model geometries/meshes for various uses. The concept comes from the fact that, in general, end-user designers of animations are not concerned with the transformations required on existing geometries, but only with their desired final shapes, which are already defined in their creative minds. Making this bridge is a challenge because some transformation tools have to be present, but not so many and not so complex, so as to avoid overwhelming the user. The majority of the underlying mathematical complexities of the transformations have to be hidden, and the visible ones have to make sense intuitively so the artist can focus on the art itself.

The parallel was drawn here between the goals of computer animations and CFD shape optimizations: to let the user easily change an existing geometry to a final desired shape. In CFD optimization, however, the user actually has to run iterations of the modifications, since he does not know exactly the final shape; which is why the implementation of parameterization is important in this case too. Also, it is important to say that many CFD engineers may not know exactly where to change an existing geometry so as to improve its results. In general, good experience on the part of the engineer is the main drive to tell where modifications can be beneficial. However, it is worth noting that the Adjoint CFD Method provides as result the flow properties' direct sensitivities to shape modifications; it would be an ideal application of sketched parametrics. For the scope of this thesis, however, it was not possible to perform it.

In the field of computer graphics and mesh modeling, the creation of tools for easily modifying existing geometries has had an important place. Among the breakthrough technologies that were created, we can highlight here the Free-Form Deformation (FFD) technique developed initially by Sederberg & Parry (1986, [21]) and its major improvements by Coquillart (1990, [22]), and MacCracken & Joy (1996, [23]), among others. The technique involves the application of deformable lattices ("boxes") of controllable vertices around an existing geometry which apply translations in space to the initial geometry's vertices based on their positions inside the lattices and a number of mathematical functions. The method is somewhat intuitive, but requires many user manipulations and has significant complexity when fine control is desired, due to increasing number of control vertices.

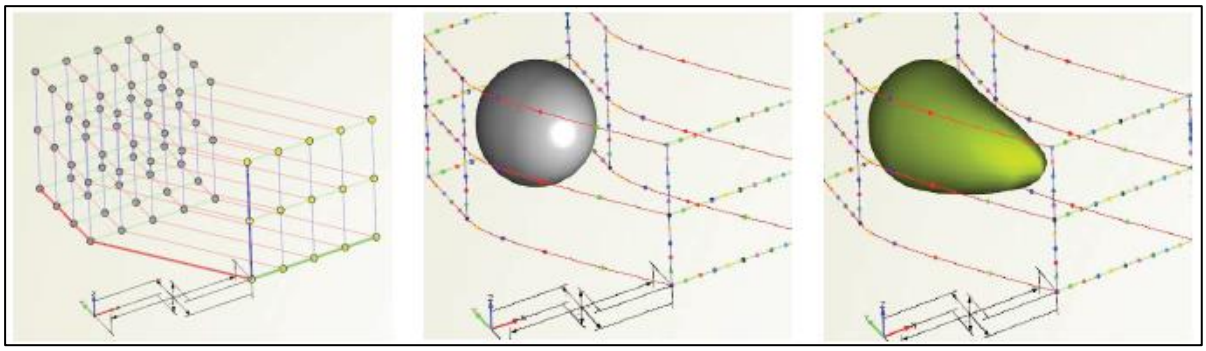

Figure 1-3 – Example of FFD performed on a sphere (figure taken from [29])

To reduce the complexity of FFD, various different tools have been built from it, hiding the complexities while giving more intuitive manners for sketching. We can point here to such examples in the works of Singh and Fiume (1998, [24]), Igarashi et al. (1999, [25]), Llamas et al. (2003, [26]), Botsch & Kobbelt (2004, [27]), Kho & Garland (2005, [20]), and Nealen et al. (2005, [28]). They vary from sketching simple lines or circling regions of influence to complex use of two-hand 3D-manipulating hardware in virtual reality space.

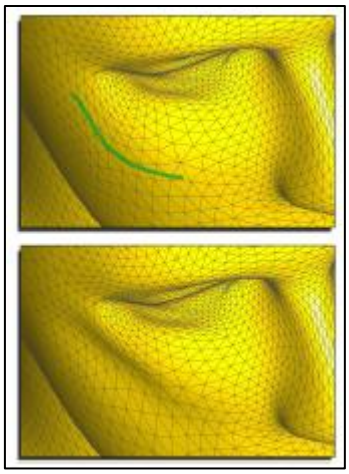

Figure 1-4 – Example of sketched parametric modification in TriMesh (figure taken from [28])

During the last few years, many geometry manipulation software companies have created or adapted tools for sketching modifications on meshes, like Autodesk's Fusion 360, Dassault Systèmes' SolidWorks, and Google's SketchUp, among others. The field has been mainly motivated by computer graphics advances and demands in the entertainment industry.

Regarding ship design, the work of Brizzolara et al. (2016 [29]) showed a resistance optimization procedure performed for a PPM model of a ship hull using FFD, and a comparison with the same procedure performed for an FPM model of the same ship. The final hull geometry result obtained from the PPM model was not a realistic one, took more time for converging, and still gave a worse resistance result than the optimized version of the FPM model. The source of the unrealistic geometry was concluded to be coming from the difficulty in implementing an

FFD transformation that would deform the hull in the natural ways that an experienced naval architect designer would. Pasquinucci (2016 [30]) worked on building a PPM model of a given ship hull with exhaustive manipulation of many control points in FFDs and imposition of constraints, and subsequent grouping of them into a smaller number of parameters for optimization. The final model was robust, and its optimization produced a realistic geometry with better hydrodynamic results. However, the handling of the FFD modeling was difficult and complex, even for an advanced user of the tools and software. This can be seen as another example of the challenge of developing sketched parametric tools for building a PPM model of a hull.

1.3. Thesis' Objectives

This thesis presents methods for sketching modifications in existing meshes and quickly making them available for CFD simulation in shape optimization. Complex geometry-modeling operations are combined into intuitive, simple and user-oriented tools within the so-called Sketched Parametric modeling method.

The method requires users to know where they want to impose changes in the hull. Applications are given for a hull shape resistance optimization case, connecting parametric-modeling software CAESES with CFD viscous flow software FINE/Marine. Optimization results are finally analyzed, advantages and disadvantages of the method are discussed, as well as ideas for future works in the area.

2. DESCRIPTION AND DEVELOPMENT OF TOOLS

In order to apply sketched parametric modeling to a CFD shape optimization, various tools and software had to be learned and implemented (CAESES and Fine/Marine) or developed (Features inside CAESES). This section describes in short the tools and methodology used.

The codes of the Features developed for this thesis are presented in Appendix I.

2.1. Parametric-Modeling Software CAESES

The software used for all CAD geometric manipulations and variations was Friendship System's CAESES program, which is a CFD-oriented 3D object-oriented modeler with a wide range of capabilities. It is able to create and modify geometries, pre-process hydrodynamic meshes, make direct connections to run CFD software, post-process results and perform shape optimization using a number of different algorithms.

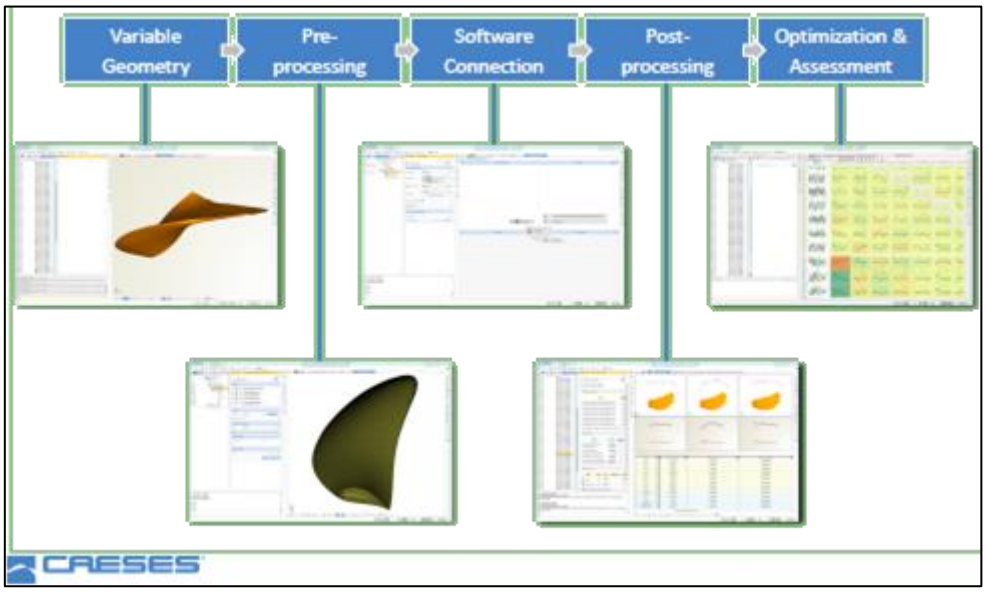

Figure 2-1 – Workflow of parametric-modeling software CAESES

In terms of creating partially parametric models from existing geometries, CAESES offers users tools that are based on the following process chain.

Figure 2-2 – Process chain for modifying a geometry in CAESES

The source and image are the input and output geometry/mesh, respectively. The transformation can be defined by simple vectors (such as a scaling on a principal axis), but for complex changes they are also defined by geometries that can implicitly introduce translation shifts, rotations, scaling, and other deformations of different kinds. One example of a 2D geometry modification by a geometrically-defined transformation is given in the following figure.

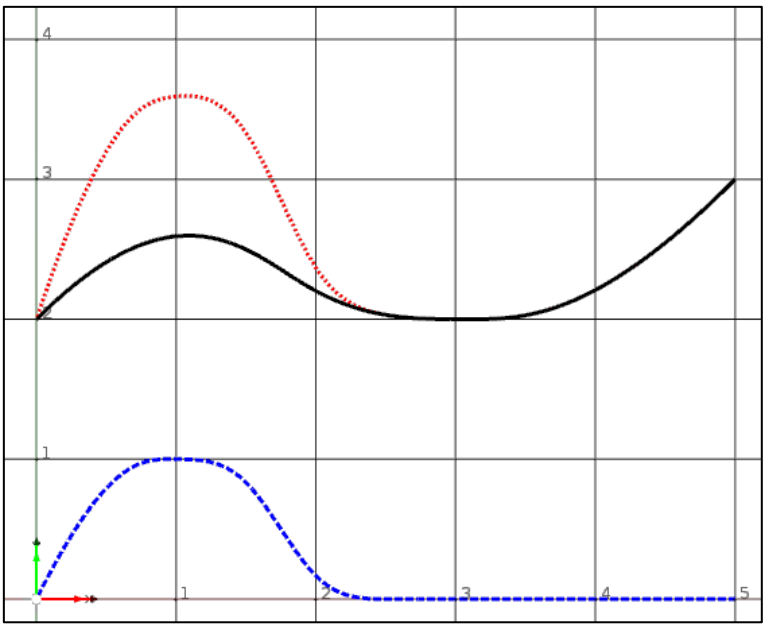

Figure 2-3 – Example of 2D geometry modification in CAESES

The solid black B-spline is an existing line used as input (source). The dashed blue line is a geometrically-defined transformation; in this case it is a so-called "delta curve" function: a curve that, by its ordinate values (vertical) along its abscissa (horizontal), imposes translations on the source object by the same values on an axis determined by the user – in this case, on the vertical direction as well. The pointed red line is the output curve (image). The first peak of the source curve, at (x,y) = (1,2.5), becomes (x_i,y_i) = (1,3.5) at the image, due to the transformation

value imposed by the blue dashed line at $(x_t,y_t) = (1,1)$. When the transformation line's ordinate values go to zero at $x \geq 2.5$, the transformation is null, making the image curve coincide with the source.

As the input geometry grows in dimensions and complexity, such as in the case of a ship hull, transformations can also grow in the same pace or even faster. Thus, as previously discussed, it should be easy to see why even experienced CFD engineers without much acquired ability in CAD would struggle to build a parametric model for shape optimization in such scenario. In a way, if a CFD-experienced user should want a parametric model based on an input hull, and he has an idea of where and how the geometry should be parameterized (he "knows what he wants"), he would rather take a shortcut in the process and not deal with the transformation geometry, as in the following process chain.

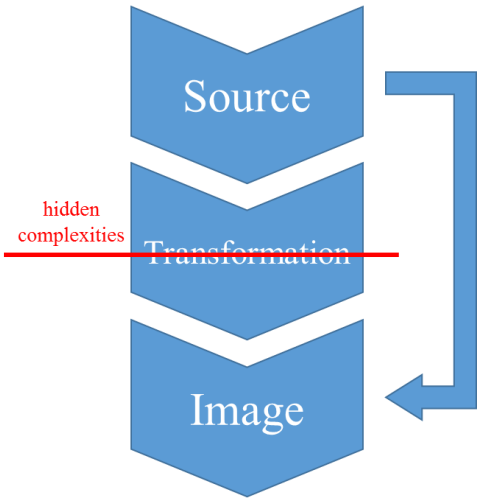

Figure 2-4 – Ideal process chain for less experienced CAD users who have idea of output

With this philosophy in mind, tools were developed inside CAESES by the author, using what are called "Features" in the software. These Features are no more than blocks of code developed in the API (Application Programming Interface) of CAESES, with its own programming language, enabling users to automate creation and modification of objects inside a model. With them, it was possible to achieve the objective of hiding the complexities from a user and asking from them only the absolute necessary to modify a geometry and create a partial parametric model, with the additional goal of making this necessary information as intuitive as possible. A bonus from using Features is that they are not only available to the users, but are also editable by them; this should at the same time help them understand a tool or transformation, and let them modify the transformation itself to achieve different custom goals or improve the existing ones.

Given that this thesis deals mostly with the method and its application, the attention was not focused on the number or completeness of the sketched parametrics Features to be developed; and due to time constraints, only two Features were developed to showcase examples and reveal some of the method's potential in CFD shape optimization.

For more information on CAESES and its Features, its Documentation Browser should be accessed directly in the software; more information is also available at [31].

2.2. Feature: Bulb Transformation

One of the most usual modifications performed on a ship hull for resistance optimization is on its bulbous bow. The whole 3D geometry of the bulb may be modified for this purpose: length, width, depth, and shape of sections and waterlines. For the tool developed for this thesis, only 3 bulb modifications were considered: changing its length, changing its nose height (the foremost point of the bulb), and inflation/deflation in the transversal (Y-axis) direction.

In the practical case of an imported triangular mesh, all of the nodes on the bulb are potentially enlisted for being design variables in optimization. We know, though, that this would mean an unfeasible computation time, possibly unrealistic solutions, and difficulty for a beginner designer to achieve. In the case of an experienced CFD user, he may also have a clue as to by how much he would want to change the bulb's length, nose height or volume. The tool to be developed had to take into consideration solutions for all these problems.

The following figure, adapted from Kracht (1978 [32]), indicates some of the bulb geometric properties cited in this thesis, in longitudinal (to the right of z axis) and transversal (left of z axis) section views.

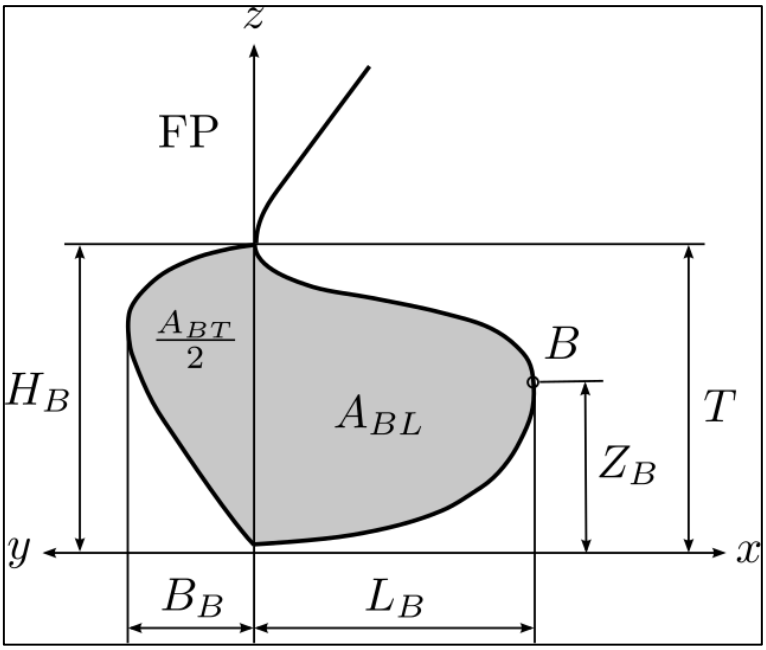

Figure 2-5 – Bulb properties, adapted from [32]

Legend:

- T = design draft;
- FP = forward perpendicular;
- B = foremost point of bulb;
- L_B = bulb length;
- Z_B = bulb nose height;
- A_{BL} = longitudinal bulb area;
- H_B = bulb height at FP section;
- B_B = bulb breadth at FP section;
- A_{BT} = transversal bulb area at FP section (only half is shown in figure);
- V_B = bulb volume (not shown in figure).

It is important to note that in the figure, the FP (defined normally at the foremost point of the design waterline) is coinciding with the aftmost point of the bulb, where the stem changes direction. In many cases, though, the design waterline is higher than this point, making the FP go forward; in such cases, for the purposes of this Feature, we continue naming the section at the aftmost point of the bulb "the FP section" and taking that point for building the bulb region.

The Bulb Transformation tool was developed by asking from the user the following input:

- Input 1: design draft for bulb definition;

- Input 2 (optional): shape of Aft Zone of Influence;
- Input 3 (optional): center of bulb inflation;
- Input 4: choose target bulb length L_B, target bulb nose height Z_B, and target bulb volume V_B (or target section area A_{BT}), and the % margin of tolerance.

These questions were made to be answered by simple number values, but with the option of changing the sketching by 3D point manipulation if advanced users so wish to. And, importantly, the user can come out of it with only three (or less) parameters for the optimization: the L_B, the Z_B, and the V_B (or A_{BT}).

It was not possible to simultaneously obtain a target V_B and a target A_{BT} with the tool because they are directly correlated in the transformations that were programmed.

The required inputs are explained in the following Items.

2.2.1. Input 1: Design Draft for Bulb Definition

It should be needless to say that the user first has to indicate to the tool what is the TriMesh of the hull; it would be an Input 0. This mesh should be as fine as desired for the shape quality.

Given the hull, the tool has to pick which part of it is considered to be the bulb, for the purpose of transforming it later. This is done simply by inputting the design draft of the ship. The Feature it responsible for finding the FP (or the aftmost part of the bulb, in case the FP point is higher than that) and creating the fore part of the bulb Zone of Influence for the transformations. The following figures show such cases (the hashed area is the Fore Zone of Influence).

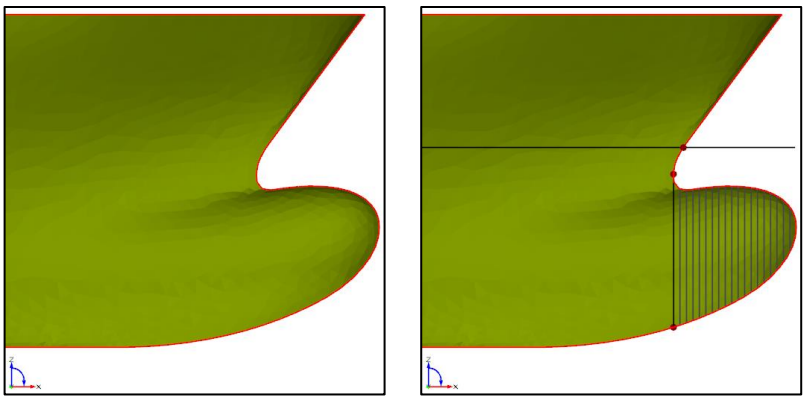

Figure 2-6 – Bow of a hull before and after introducing the draft: high FP point

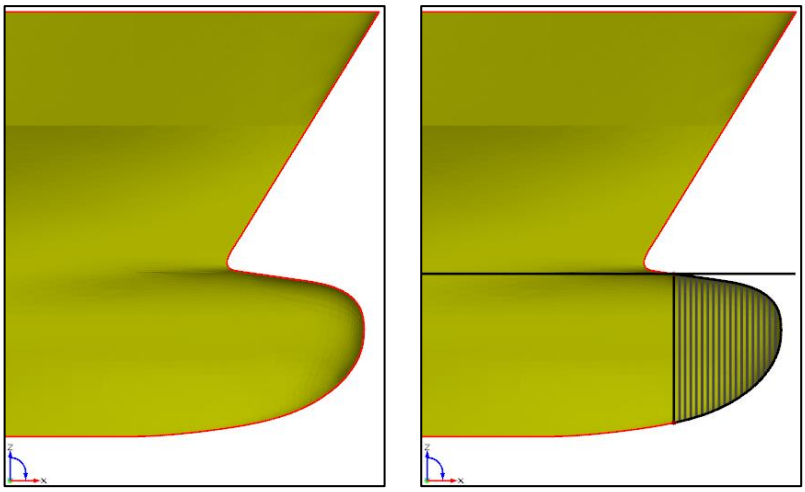

Figure 2-7 – Bow of a hull before and after introducing the draft: low FP point

2.2.2. Input 2 (Optional): Shape of Aft Zone of Influence

When deforming a bulbous bow, normally a significant portion of the hull to the aft of the bulb is also considered, so the resulting transformed geometry can fit in smoothly with the rest of the hull. The shape and size of this Aft Zone of Influence is controlled by 1 main parameter – the "Slider" – and 3 minor parameters – the Upper, Lower, and Central factors.

The Slider is a positive numerical value that defines the area of the Aft Zone in comparison to the Fore Zone of the bulb. When set to zero, the Aft Zone will be null, such as in the previous figures; when set to 1.0, the Aft Zone will have approximately the same longitudinal area as the Fore Zone (if the default values for the minor parameters are kept the same also). Fractions and multiples of this value will result in fractions and multiples of this area, respectively. The following figures show the shape of the Zone of Influence with Slider = 0.0, 0.5, 1.0, and 2.0, respectively, in clockwise direction starting from top left.

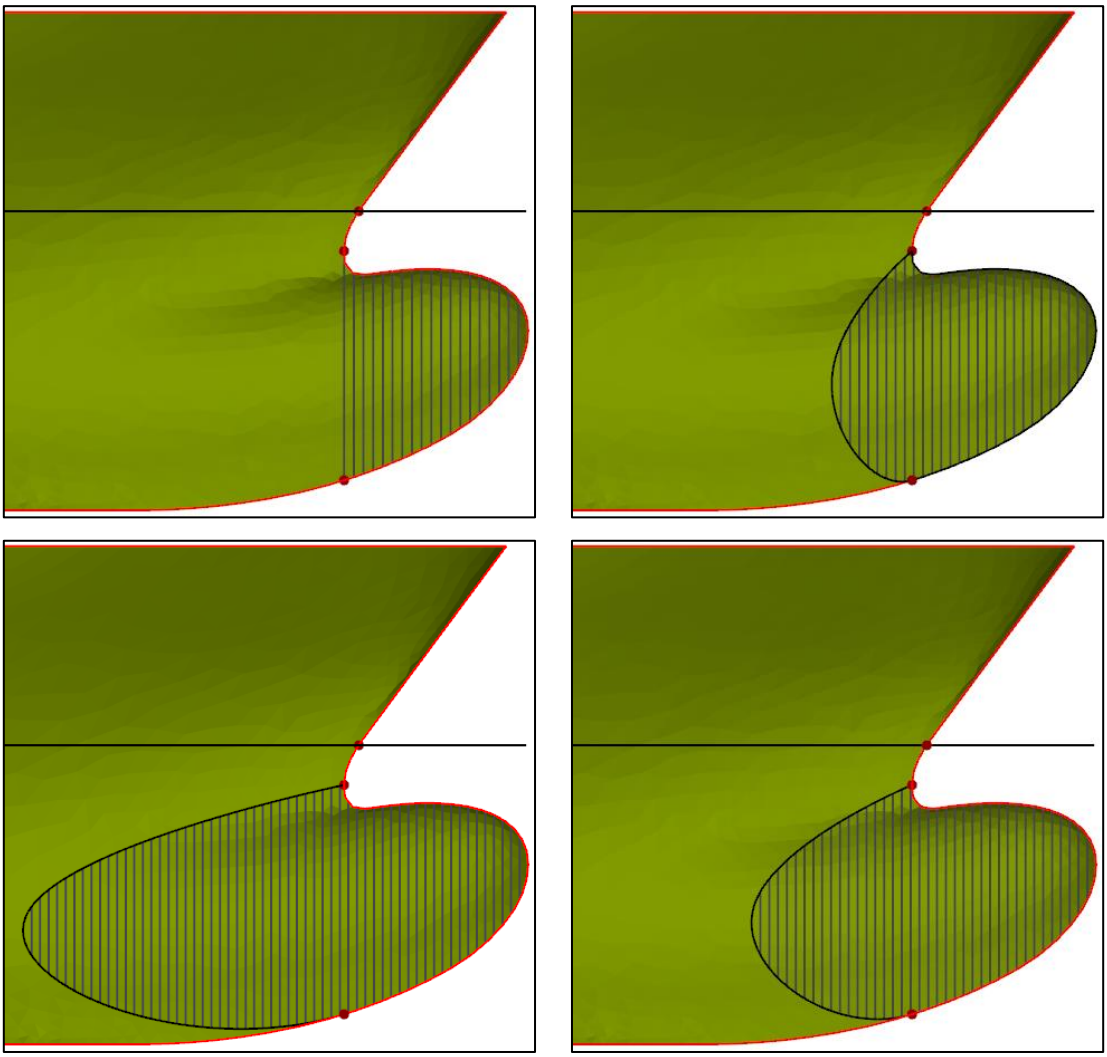

Figure 2-8 – Slider = 0, 0.5, 1, and 2 (clockwise from top left)

The Upper and Lower factors are minor shape parameters that control the edges of the outline of the zone by scaling their tangent vectors. The bigger their value, the bigger is the influence of the tangent on the shape of the Aft Zone. When set as zero, the point becomes a "hard" point, such as with the Upper factor on the previous figures. Any other positive value will make it "softly" join both zones, such as with the Lower factor of 5.0 used in the previous figures. The following figures show a few variations of these factors with Slider set to 1. The values of the factors are written in the figures.

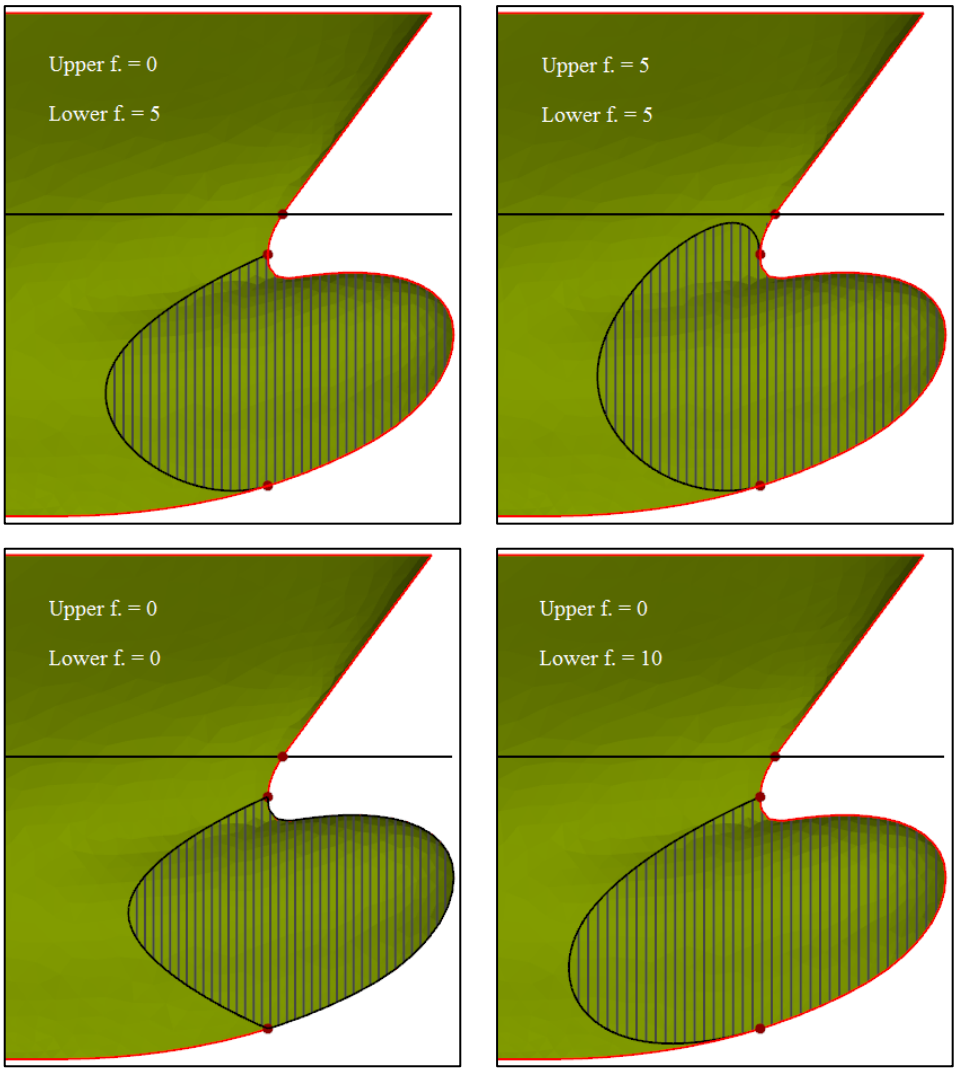

Figure 2-9 – Variations of the Aft Zone with minor shape parameters Upper and Lower factors

Finally, the Center factor is the minor shape parameter that controls the center of the Aft Zone outline. Increasing or decreasing it affects the influence and size of the center portion of the Aft Zone. The following figures show a few variations of this factor with Slider, Upper and Lower factors set to 1, 0 and 5, respectively. The values of the factor are written in the figures as well.

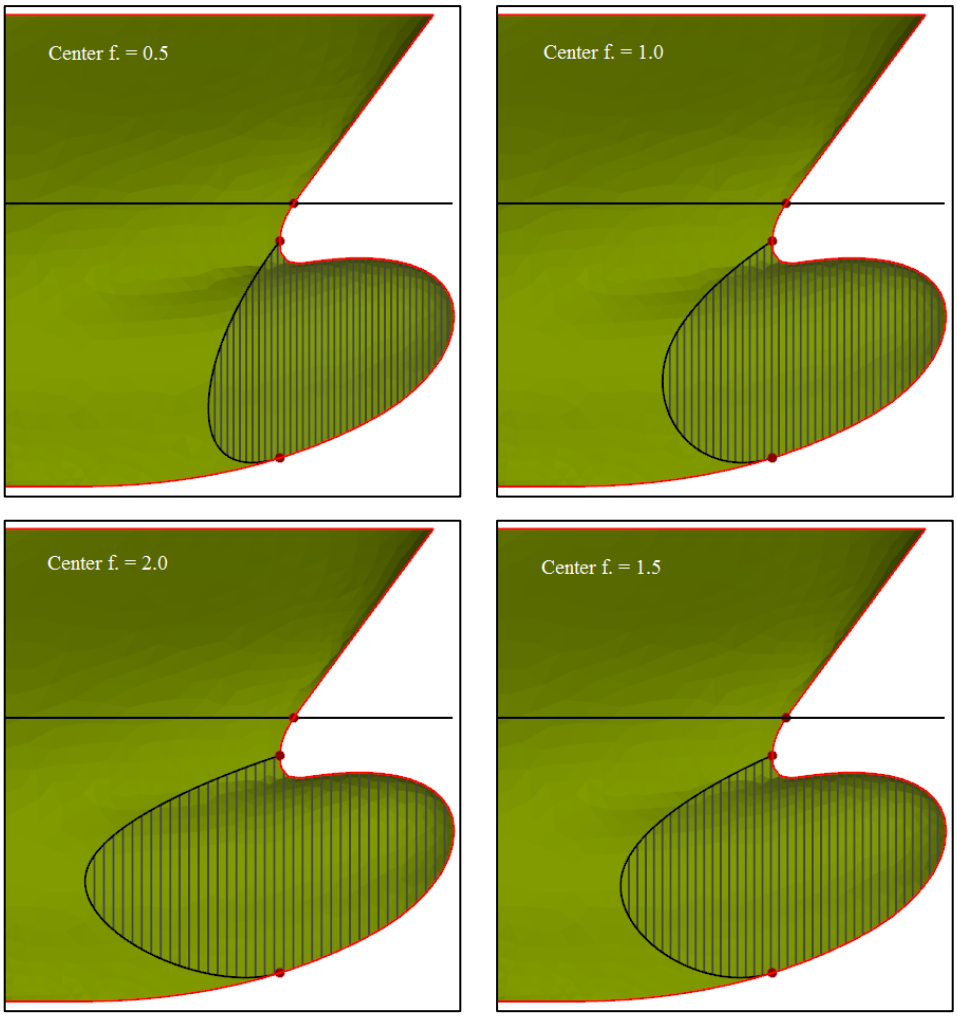

Figure 2-10 – Variations of the Aft Zone with minor shape parameter Center factor

With these parameters, the shape of the Aft Zone of Influence can be very flexible and accessible to any user. For advanced users more familiar with CAD, the Feature can also be executed so that the entire aft outline of the zone can be edited as a B-spline.

This shape of the Aft Zone is considered as optional in the Feature, since the default values should give a fair enough region for starting (default is Slider = 1, Upper factor = 0, Lower factor = 5, and Center factor = 1.5).

2.2.3. Input 3 (Optional): Center of Bulb Inflation

The default center for the inflation/deflation of the bulb volume is at the point of maximum breadth at the FP section. This is normally the point of maximum breadth of the entire bulb, and therefore a good place for having the maximum inflation be imposed. However, it was also made possible for the user to choose a different center of inflation. This is particularly

desired if the Slider is set to zero, and therefore no Aft Zone is existent. An inflation on the FP section in such case would provoke discontinuities on the mesh.

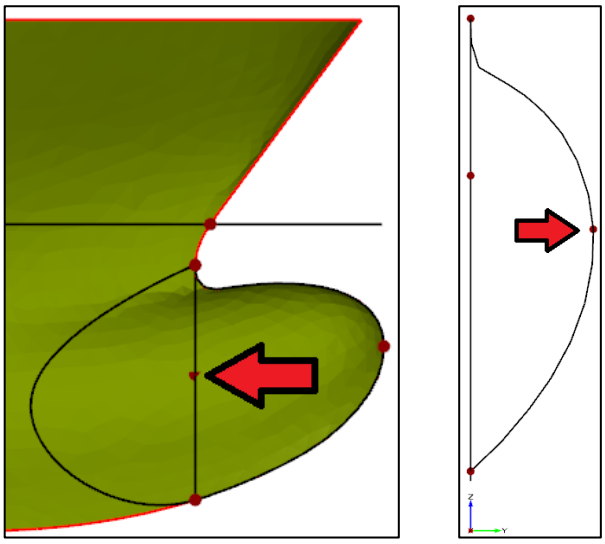

Figure 2-11 – Default center of inflation (max. breadth at FP), longitudinal and section views

2.2.4. Input 4: Targets L_B, Z_B, V_B (or A_{BT}), and Tolerance

The Feature calculates and informs the user of the original bulb properties: L_B, Z_B, H_B, B_B, A_{BT} and V_B. With that information, the user can provide target values for L_B, Z_B, and V_B (or A_{BT}), as well as a tolerance margin, and the tool will make the required transformations. It is important to note that V_B is calculated including the Aft Zone of Influence too, and using a set number of offsets that the user can control too.

For L_B and Z_B, the transformations used are delta shift curves, such as the example from Item 2.1, but applied to a full 3D geometry. The transformations were made to be as smooth as possible, keeping the connections between the bulb and the rest of the hull without discontinuities or change of tangents. This was done by imposing zero translation at the FP section and aftwards, and also keeping the tangent of the delta shift curve as zero at that point. The curves used were B-spline curves that smoothly change from zero to the maximum target ordinate at the tip of the bulb. The following figures show these curves and transformations for two examples of a target L_B and a target Z_B. The delta shift curves for L_B (in blue) and Z_B (in red) are shown below the bulb before the transformation (left figure). The transformed hull is shown on the figure to the right.

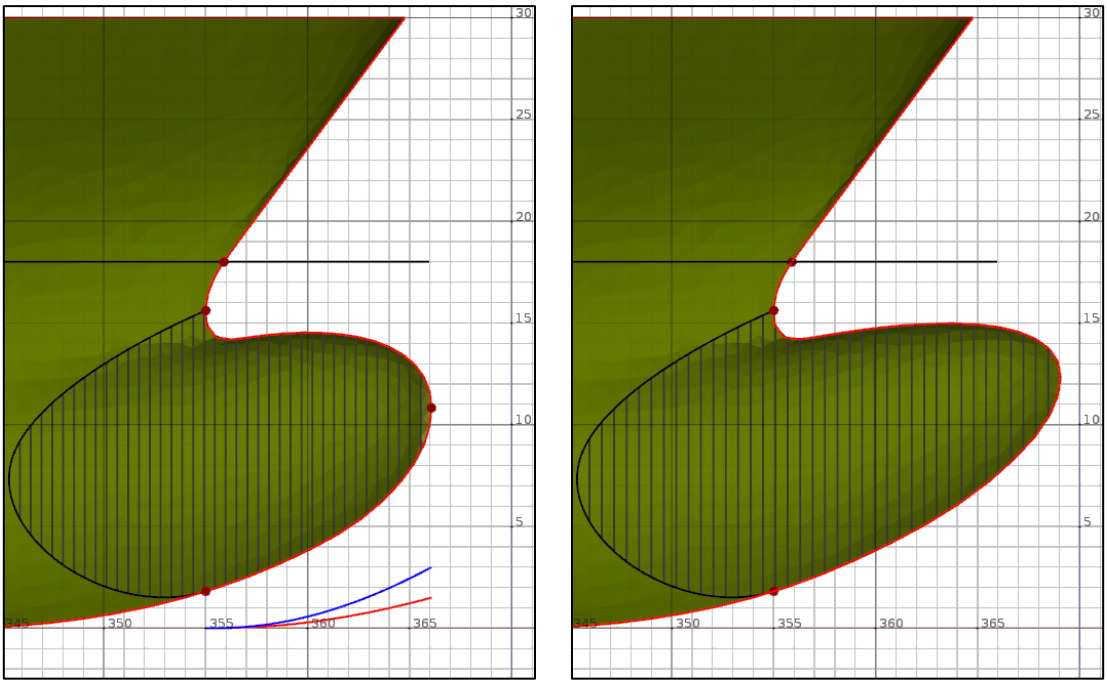

Figure 2-12 – Increasing both L_B and Z_B, by 30% and 15%, respectively

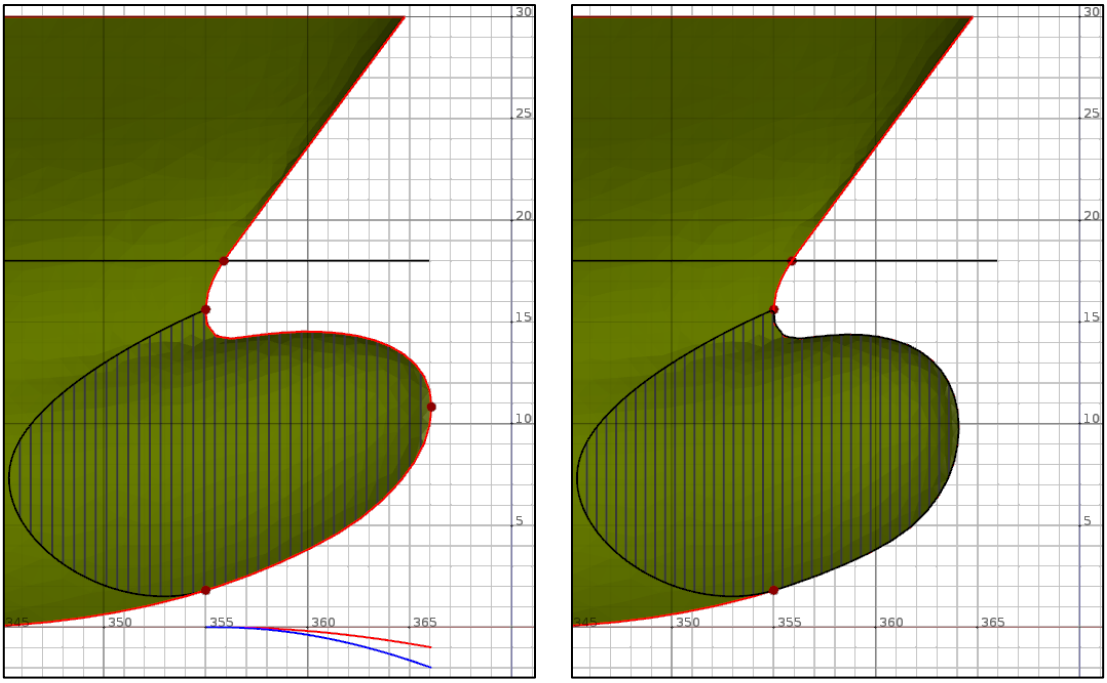

Figure 2-13 – Decreasing both L_B and Z_B, by 20% and 10%, respectively

The transversal sections are kept the same for these transformations, just shifted.

To inflate the bulb in its Zone of Influence, the so-called Surface Delta Shift Transformation (Deltasurface) was used. This transformation applies translation shifts to any geometry located on the projection of a surface to a chosen principal plane. In this case, the Deltasurface would be located completely inside the Zone of Influence, applying shift values in the Y-axis

direction of any point of the bulb located within the outline created when projected on the Centerline plane. It thus modifies the transversal sections without changing their positions.

The Deltasurface could have any kind of function for its elevation. It was decided that the best one for a bulb inflation would be with a derivative equal to 0 at its top (at the center of inflation, where maximum inflation occurs), and both derivatives and elevations equal to 0 all around the outline of the Zone of Influence. This way, the final deformation of the bulb would transition to the undeformed hull as smoothly as possible, avoiding discontinuities or tangent changes. A B-spline with controlled tangents at its ends was used as the generatrix. The following figure shows a schematic drawing of how the delta function (solid blue line) would be (to the Y-axis) from the center to an edge (on the fore of the outline of the Zone of Influence). A dashed red line indicates where the function would be located on the Zone of Influence.

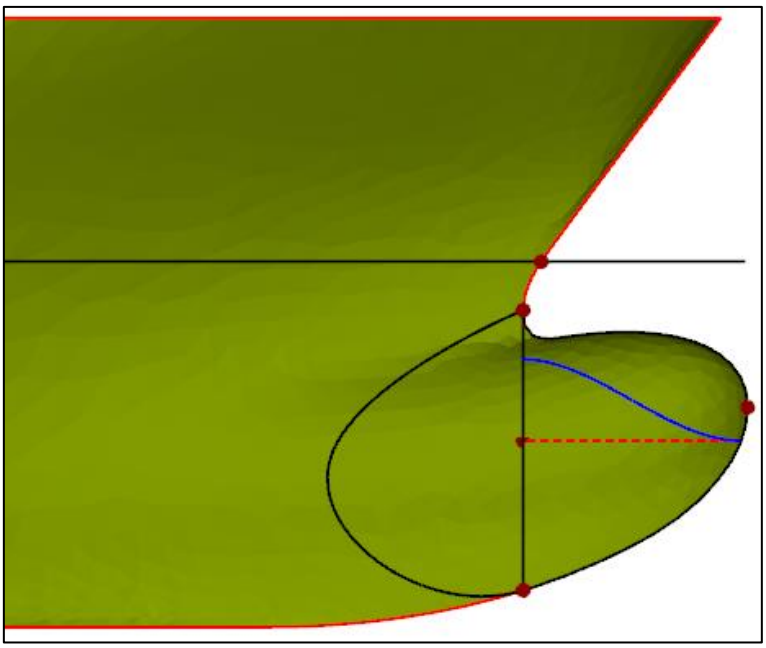

Figure 2-14 – Schematic drawing of delta function expected at center and edges

Thus, the idea would be to revolve such a delta function around the center of inflation, keeping the other end always on the outline of the Zone of Influence. Such a surface is not possible to be done with a normal revolution, because the outline is not a circle or an ellipse, but a generic curve. Thus, a so-called Metasurface was used; this Metasurface is a unique feature of CAESES to create parametric surfaces using parametric curves in virtually any scenario.

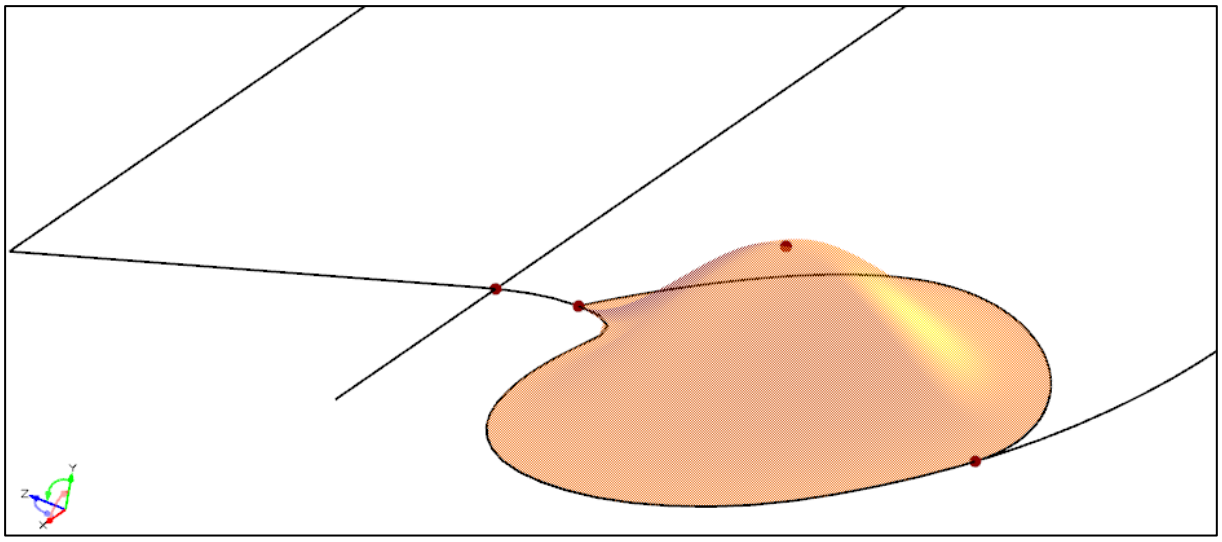

Figure 2-15 – Deltasurface for bulb inflation with a large height value in 3D view

The created Deltasurface is a smooth Metasurface with one main parameter: the height (y-axis shift) applied on its center. Optionally for advanced users, almost every aspect of it can be changed too, such as the delta function shape of its sections.

Finally, the user inputs a target bulb volume V_B (or a target section area A_{BT}) and a margin of tolerance (in percentage) for finding it. The Feature runs a goal-seek procedure to find the targets using a step-by-step variation of the Y-axis height of the Deltasurface and a bisection method until the result including margin of tolerance is achieved. Normally a bit less than 5 iterations is needed with a 2% tolerance, which represents a few seconds in a standard computer.

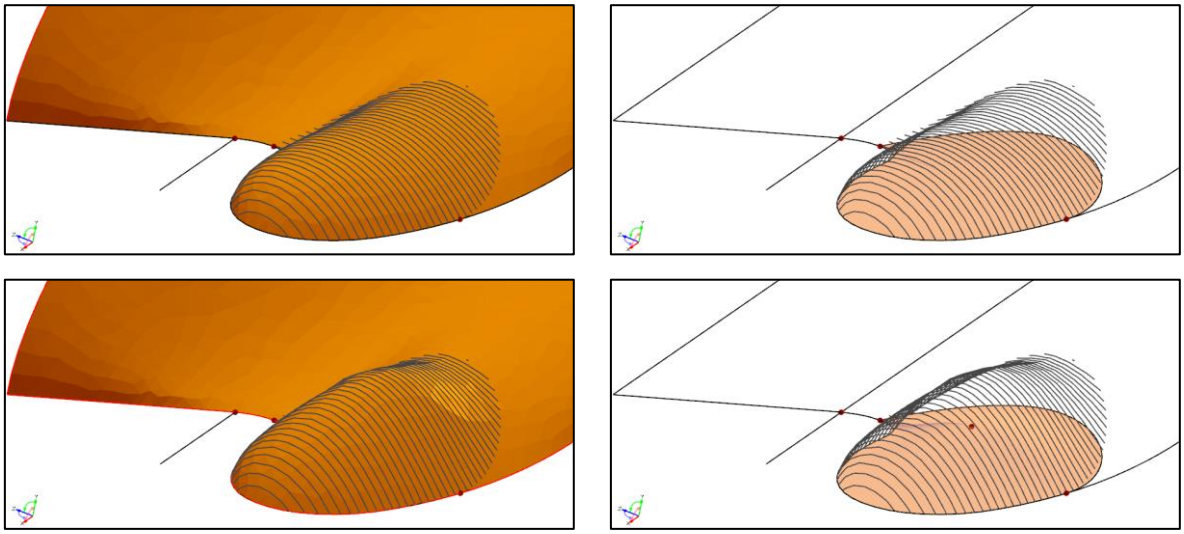

Figure 2-16 – Deltasurface inflating bulb (from top to bottom), by a large value

2.2.5. Summary of Inputs and Outputs

In summary, the user has to input only the design draft for the Bulb Transformation Feature to automatically calculate many bulb properties. The tool also automatically creates an adequate Zone of Influence for the bulb transformation, but the user can optionally change its shape with ease, just by changing 1 shape main parameter and/or 3 minor shape parameters. The user can optionally choose a center of inflation or let it be at the FP section. The user then just has to input target values for the bulb length, bulb nose height and bulb volume (or transversal section area at center), and the Feature will the requested bulb, and the output geometry will be formed.

Advanced users can also optionally manually change the outline of the zone of influence, the delta function used for the bulb inflation or bulb deformations, and almost any aspect of it.

Of course the user can parameterize anything in the input, but the simplest solution is to parameterize only one or more of the targets (L_B, Z_B, V_B or A_{BT}), thus leaving an easy output for a CFD optimization. The tool can be considered a good example of sketched parametric modeling due to its simplicity and intuitiveness.

The following figures show how the dialog window for the Feature in CAESES looks like, leaving default values.

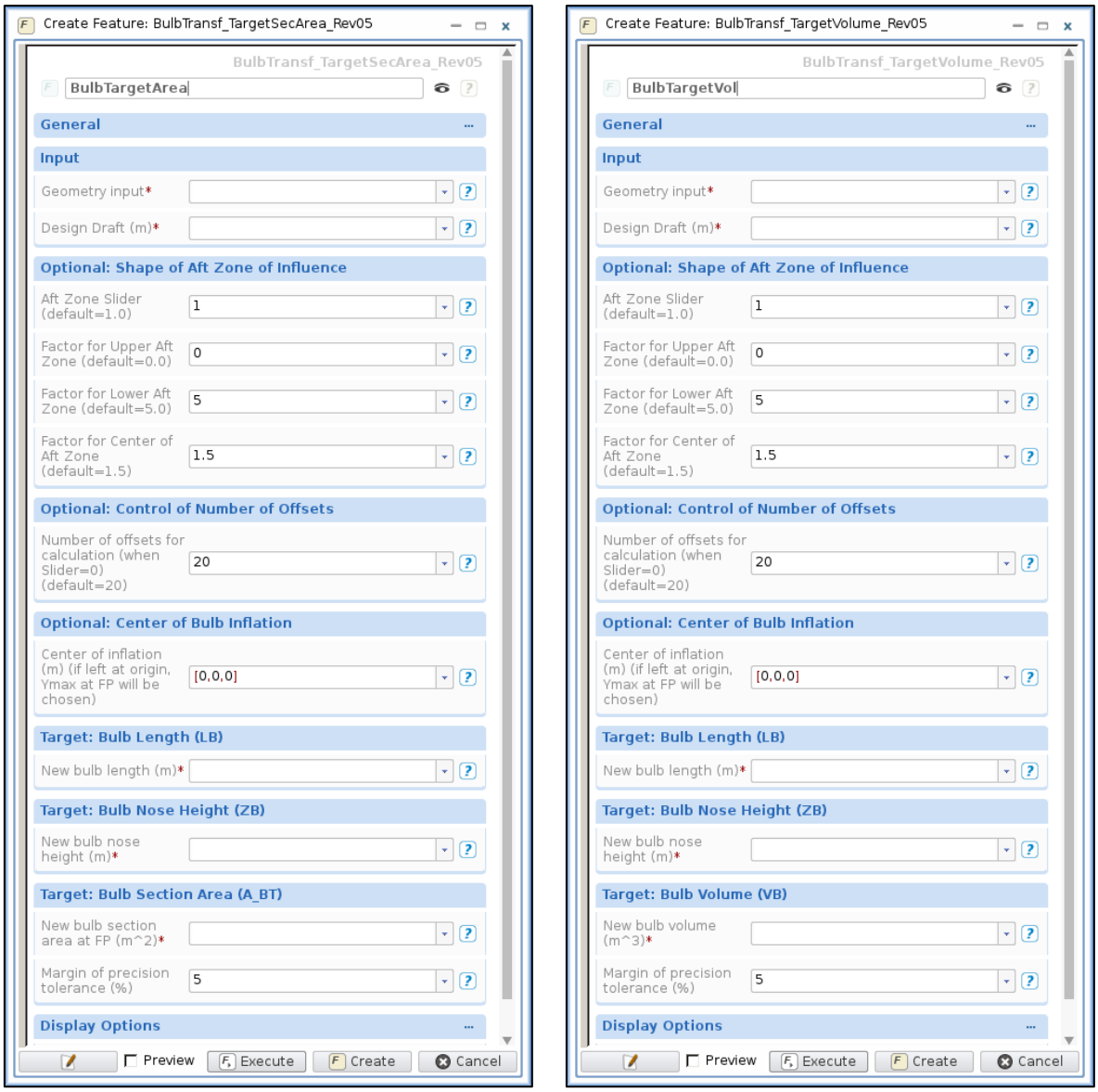

Figure 2-17 – Outlook of dialog window for Feature (for target section area (L) or volume(R))

2.2.6. Suggestions for Improvements

Many particulars of the Bulb Transformation Feature could be changed or improved. One issue that is apparent with the tool is the uncomfortable amount of time that it takes for the model to find target values for the transformation. It only takes a few seconds, but it is enough to make it less user-friendly. The reasons for this are the creation and application of the Metasurface as a Deltasurface transformation, and the goal-seek procedure used inside the Feature for finding the targets. The lethargy related to the Metasurface would be difficult to work around, but the goal-seek procedure could be done via many faster methods than the current.

One improvement would be to make both V_B and A_{BT} to be targets simultaneously. This could be done by changing the extension of the slopes in the generatrix curve that creates the Deltasurface. In practice, this would mean a wider "top" or wider "base" of the Deltasurface.

Finally, the code is not completely fool-proof, and would need more usage in different cases in order to find bugs and issues. It was only tested in 3 different hull meshes.

2.3. Feature: Aft Waterline/Diagonal

When performing hydrodynamic optimization of a ship hull, an area of great interest is the stern region. The shapes of the waterlines in this area greatly influence the efficiency of the propulsion system and the resistance drag. In the tool developed for this thesis, only transversal (Y-axis) transformations to the aft hull were considered in order to shape the aft waterline to be as the user wishes, along with a vertical Zone of Influence for the transformation to smoothly join with the rest of the undeformed hull. The edges of the deformed waterline also have matching tangents with the original one for the purposes of smooth transition. The transformation type used was the Surface Delta Shift (Deltasurface). However, the tool was made to also change waterlines in a diagonal direction, meaning a downward or upward angle being visible in transversal sections of the aft hull when showing the waterline cut. This gave flexibility to the tool, given the 3D characteristics of common stern geometries.

In theory, this Feature should work with any hull geometry that presents continuous aft waterlines. In practice, however, since it applies transformations on nodes of imported triangular meshes, the user should be careful about the refinement level and triangle tessellation of the hull where the transformations are to take place. Examples of a proper and an improper mesh for use of the Feature are shown in the following half-hull stern figures.

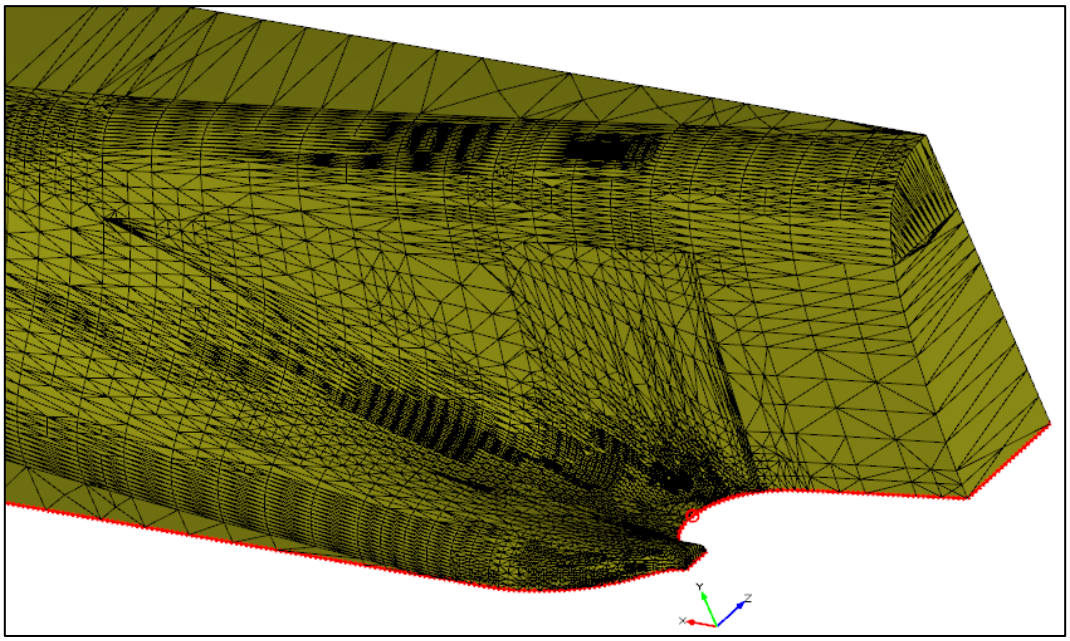

Figure 2-18 – Proper mesh for Aft Waterline/Diagonal Feature

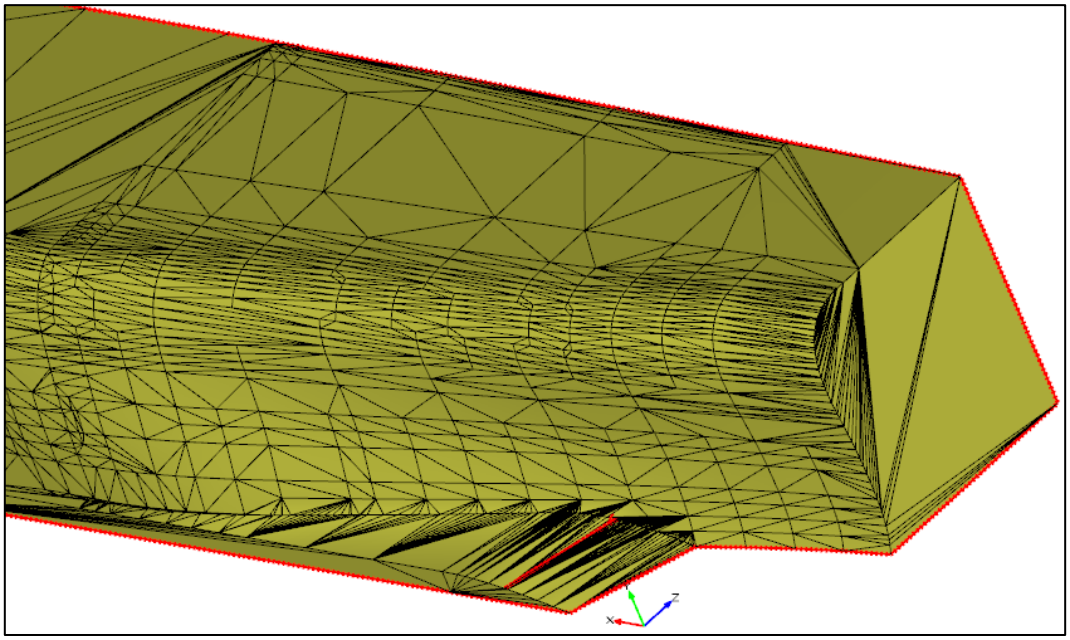

Figure 2-19 – Improper mesh for Aft Waterline/Diagonal Feature

Regardless of the different refinement levels of both meshes, the main problem with the second mesh is that its central skeg does not have nodes in places other than at its edges. This means that any transformation applied on the waterlines of the skeg would result in no geometric change whatsoever, since it can only apply modifications to nodes. The Feature would not be adequate to change other regions of this twin-propeller hull also, because almost any degree of change to the aft waterlines in Y-direction would cause bad deformations of the triangles that compose the mesh. In principle these problems could be fixed by proper remeshing of the hull, but this is not always possible in practice.

The Aft Waterline/Diagonal tool was made by asking from the user the following inputs:

- Input 1: depth of aft waterline to change, and longitudinal extension;
- Input 2: angle of waterline/diagonal in transversal view;
- Input 3 (optional): number of control points on new waterline;
- Input 4: vertical extension of Zone of Influence at waterline ends
- Input 5: maximum amplitude of waterline change in transversal direction.

These questions were made to be answered by simple number values, but with the option of changing the sketching by 3D point manipulation if advanced users so wish to. And, importantly, the user can come out of it with only one parameter for the optimization: the maximum amplitude of change.

The required inputs are explained in the following Items.

2.3.1. Input 1: Depth of Waterline and Longitudinal Extension

As with the Bulb Transformation Feature, it should be needless to say that the user first has to indicate to the tool what is the TriMesh of the hull. This mesh should be as fine as desired for the shape quality, and should avoid the problems discussed in the beginning of this Section.

The waterline depth has to be given so the tool can find the aftmost point of the hull at that height. The longitudinal extension gives the window where changes will be made to the waterline, by inputting the X values for its start and end.

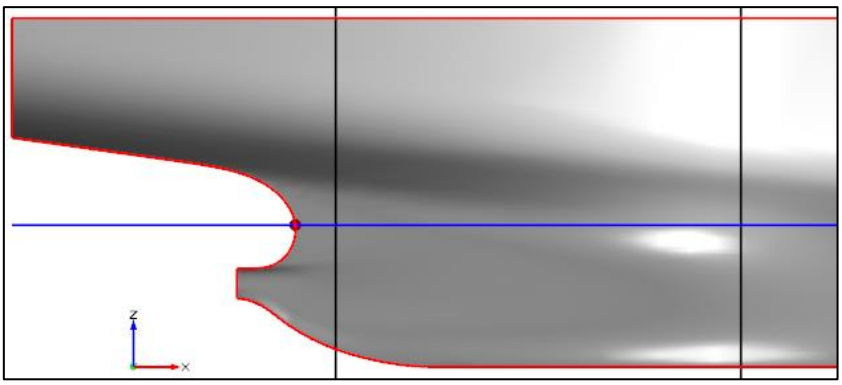

Figure 2-20 – Waterline (blue line) and longitudinal window (black lines) for changes

2.3.2. Input 2: Angle of Waterline/Diagonal in Transversal View

The waterline/diagonal angle is taken with reference to the transversal plane view (YZ plane), from the aftmost point at the waterline provided, as taken in the previous figure. For the purposes of this Feature, this angle is positive downwards, with zero being the natural waterline. The following figures show the aft view of the ship with the natural waterline (dashed blue line) and the waterline/diagonal (solid red) planes for the angles of 10° and 30°.

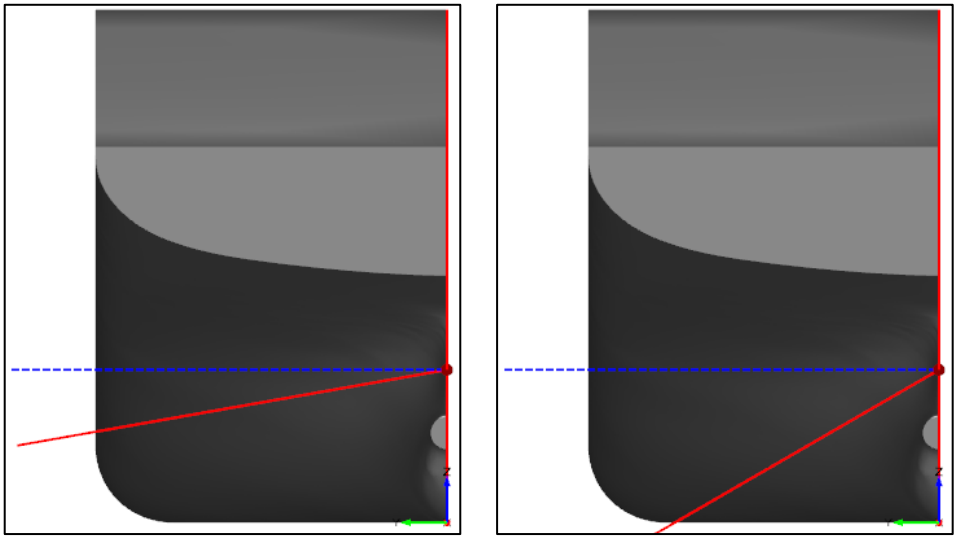

Figure 2-21 – Waterline/diagonals, 10° (left) and 30° (right)

The Feature then cuts the hull into sections and the waterline/diagonal.

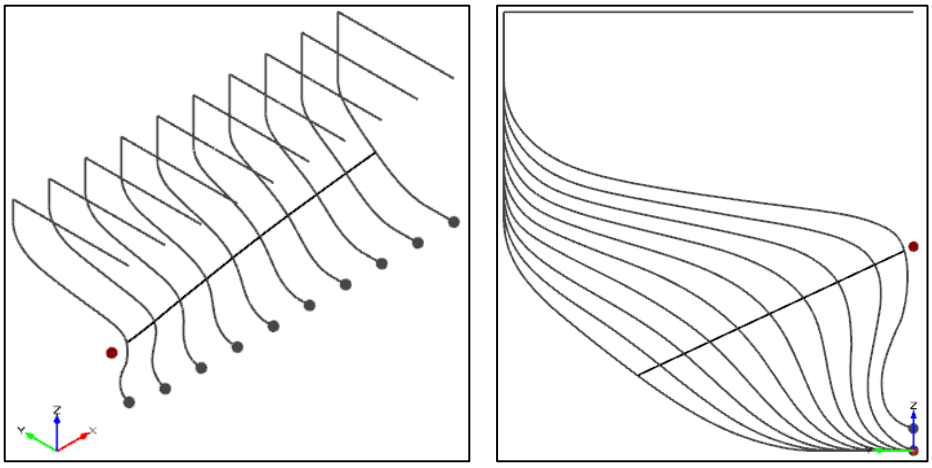

Figure 2-22 – Sections and waterline/diagonal (isometric and transversal view)

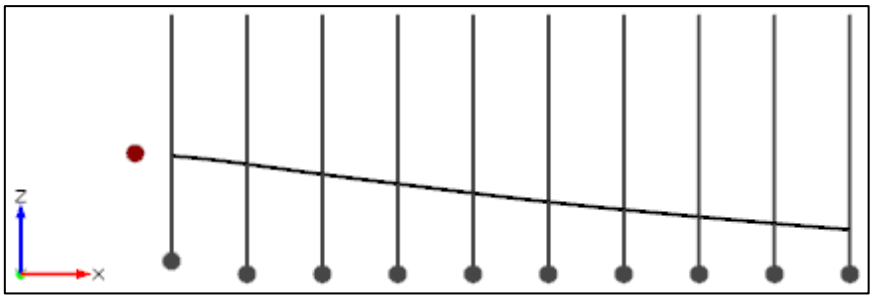

Figure 2-23 – Longitudinal view of the waterline/diagonal and the sections

2.3.3. Input 3 (Optional): Number of Control Points on New Waterline

By default, 9 control points along the waterline/diagonal projection on the XY plane are created, but the user can choose how many to use. This is mainly directed to advanced users who wish to have a fine control of the shape of the new deformed waterline.

The control points give the user the ability to define how the final waterline/diagonal will look like after the transformation. They control its projection on the XY plane. The following figure shows an original (blue line) and desired (red line) waterline/diagonal created by the Feature, with 6 control points that can be manipulated by the user. The edges of the waterline/diagonal are forced to have the same tangents as the original for purposes of smoothness of the final geometry, as described later in this Section.

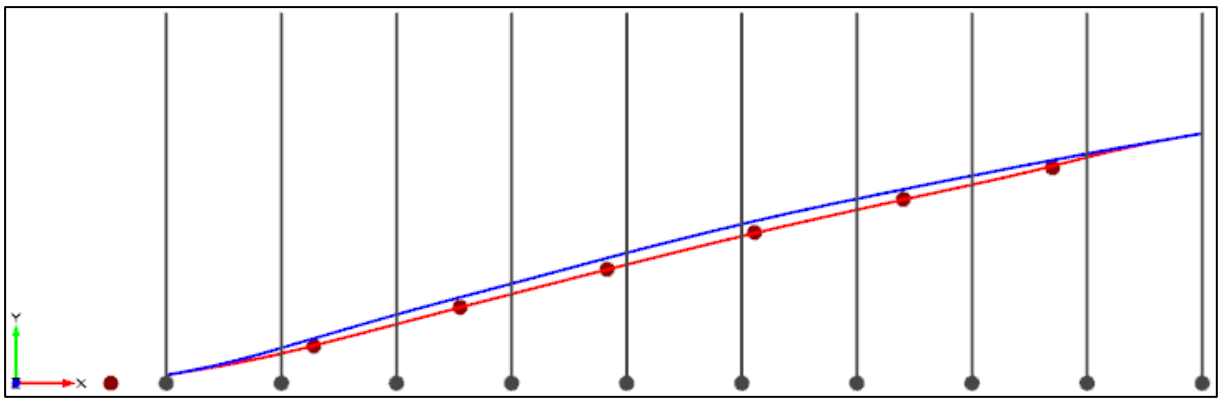

Figure 2-24 – Original (blue) and modified (red) waterline/diagonal, top view (XY plane)

2.3.4. Input 4: Vertical Extension of Zone of Influence at Waterline Ends

The Zone of Influence for the waterline/diagonal change surrounds it, making sure the vertical transition between the modified hull and the undeformed one is smooth. A vertical function is applied by default for the transformation, centered in the waterline/diagonal at all

transversal sections, and having tangents equal to zero at its edges. The following schematic figure shows that vertical function (solid red line) along random 3 longitudinal positions of the waterline/diagonal, with values for the Y-axis variation peaking at the waterline/diagonal and reaching zero (and derivative zero too) outside of its vertical limits.

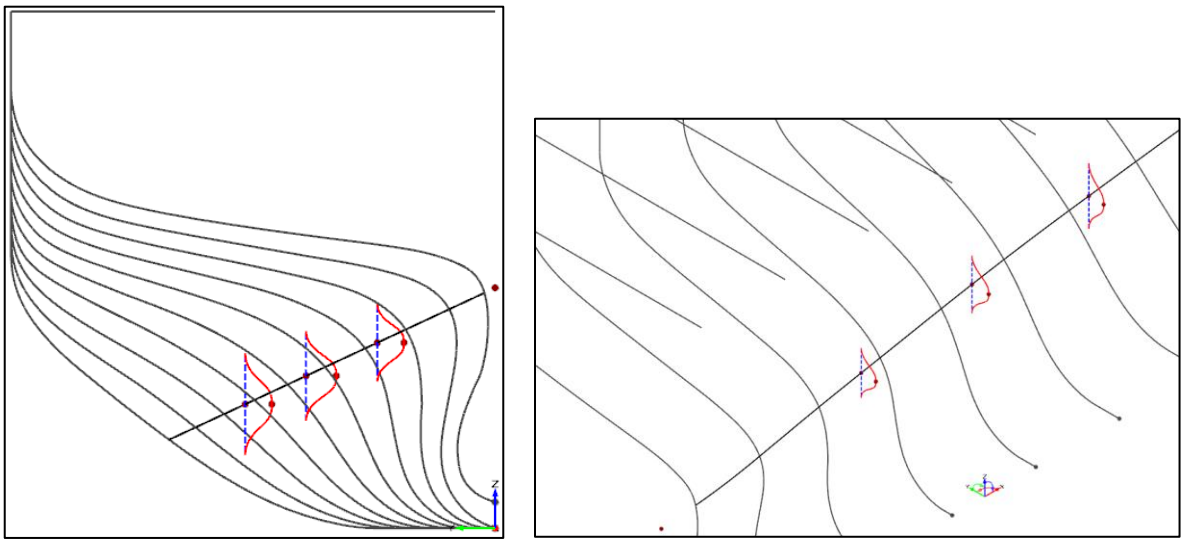

Figure 2-25 – Schematic drawing of the vertical Zone of Influence at 3 positions

It is possible to see in the previous figure that the vertical limits at each longitudinal position are somewhat different. This was done in purpose to show that the Zone of Influence for the transformation can vary vertically at any position. For the sake of simplicity, the Feature was programmed so the user just has to input the vertical semi-extension (half up, half down) of the Zone of Influence at the start and at the end of the waterline/diagonal; a simple linear variation is performed between them. The following figure shows the vertical limits (solid blue lines) of a Zone of Influence that has the extension of 2m at the aft part and 4m at the fore part.

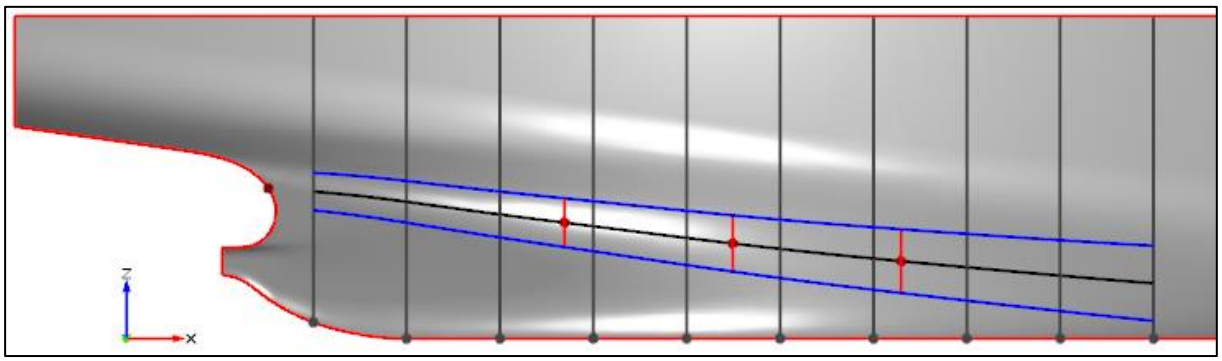

Figure 2-26 – Vertical limits of Zone of Influence for transformation

It is possible for advanced users to change the shape of this vertical function.

The peak amplitude of the variations in Y-direction along the Zone of Influence is explained in the next Item.

2.3.5. Input 5: Maximum Amplitude of Waterline Change

The Deltasurface for the transformation is created by moving the vertical function shown in the previous Item along the waterline/diagonal projection on the XZ plane, thus imposing translations on the Y-direction of all nodes of the mesh located in it.

If the peak values of the vertical function were kept the same value at all longitudinal positions of the waterline/diagonal, the start and end edges of the Zone of Influence would produce a longitudinal discontinuity (so-called G^0 discontinuity) between the transformed and the undeformed hull. To make the transition smooth, a longitudinal function was programmed in the Feature to describe the peak (amplitude) values for the transformation along the waterline/diagonal, making sure that the Deltasurface had zero tangent values at the edges – and therefore eliminating discontinuities and tangent changes of the output mesh at the transition. The normalized longitudinal function used was the following:

$$f(x) = (1 - x^2)^3 \quad , \quad x \in [-1, +1]$$

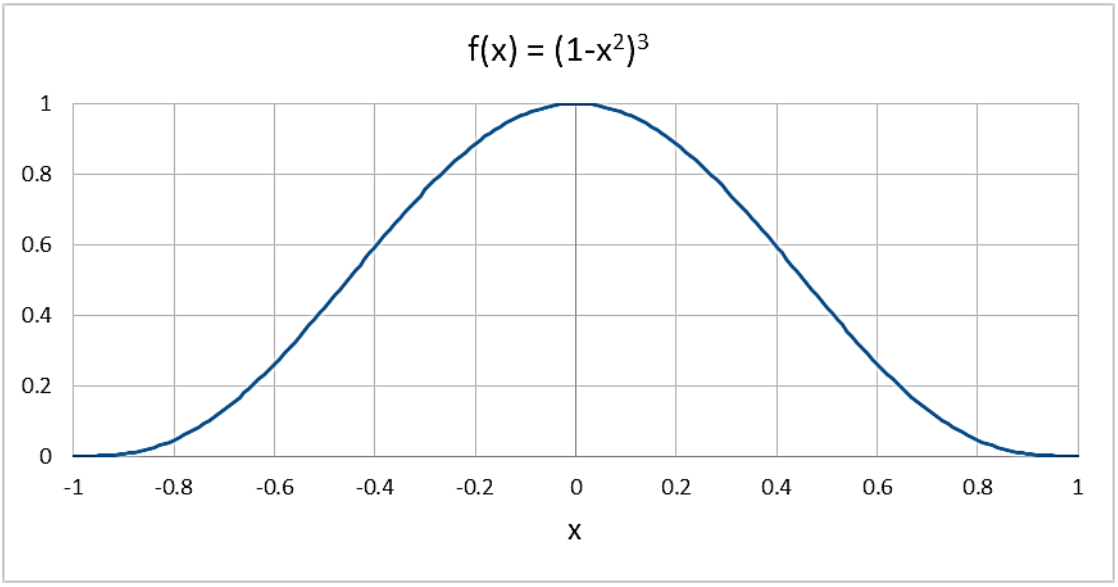

Figure 2-27 – Normalized longitudinal function for application of transformation

The domain from -1 to +1 is replaced by the longitudinal extension of the waterline/diagonal and centered on its middle, and the maximum amplitude is given the value input by the user, which can also be a negative number. The characteristic of the function is that it has zero tangents at its middle and edges and smooth slopes. The following figure shows an example of longitudinal function (solid red line) applied in the example hull used here, from a top view (XY plane).

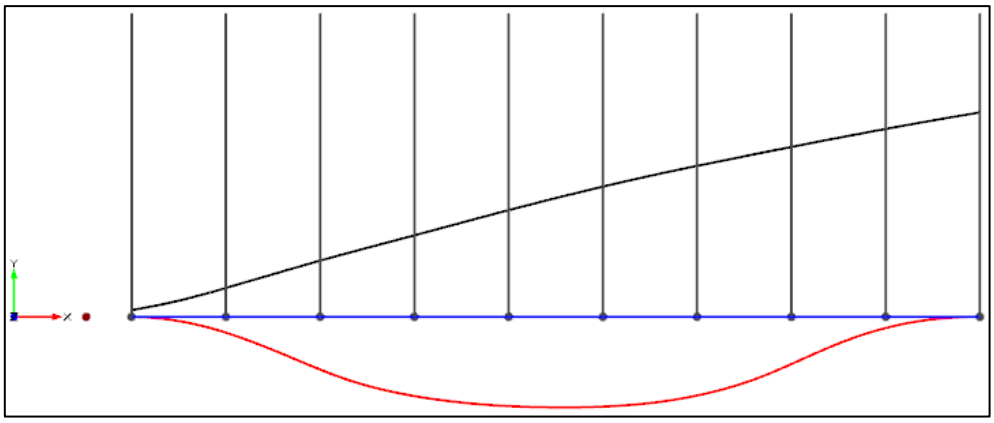

Figure 2-28 – Longitudinal function (red) used for amplitude of transformations in Y-direction

It is possible for advanced users to change the shape of this function by manipulating the optional control points created for the shape of the new waterline, as described in Item 2.3.3. Doing that will override this default longitudinal function. Smoothness is still guaranteed, since the deformed waterline/diagonal is forced to have the same tangents as the original.

Finally, the Deltasurface is created and imposed as a transformation to the hull. The following figures show an example of Deltasurface with very large deformation.

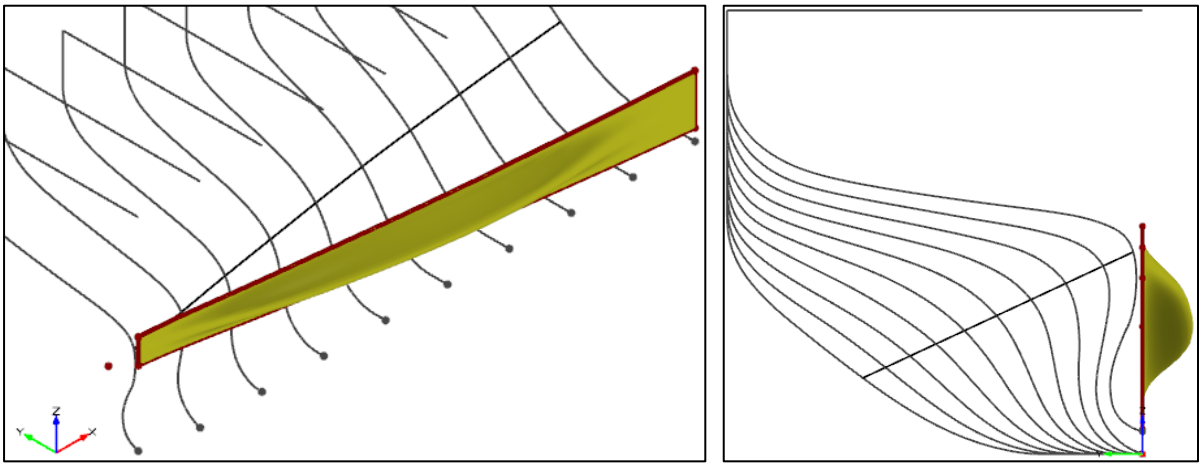

Figure 2-29 – 3D and section view of Deltasurface

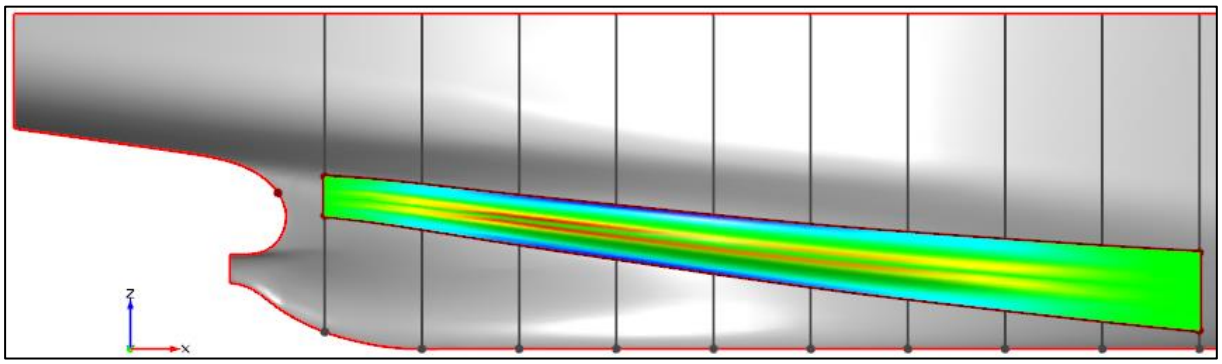

Figure 2-30 – Longitudinal view of Deltasurface with curvature color map

The following figure shows the resulting offsets of a hull deformed by the Aft Waterline/Diagonal Feature by -0.12m transversally at its maximum amplitude. The black sections are the original ones, and the red ones are the deformed.

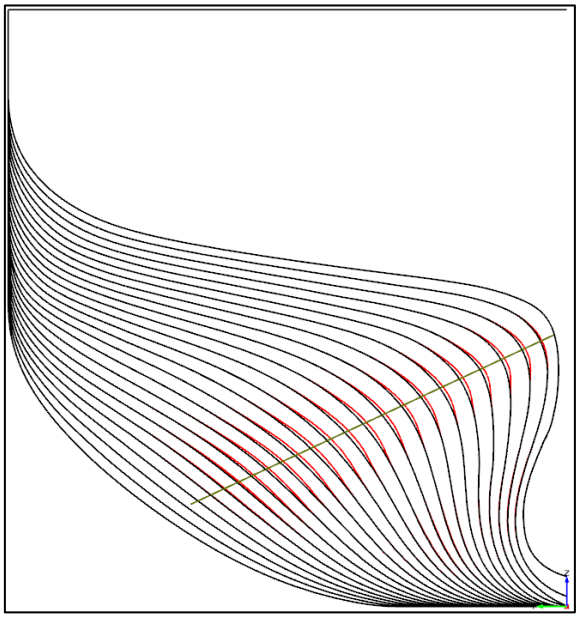

Figure 2-31 – Deformed (red) and original (black) sections of aft of ship

The transformations can also be combined for different waterlines/diagonals, such as in the following example figures. The pink lines indicate a negative deformation on a high diagonal waterline, and the blue lines indicate a positive deformation on a lower diagonal. The longitudinal view also shows the vertical limits of the Zone of Influence (dashed lines).

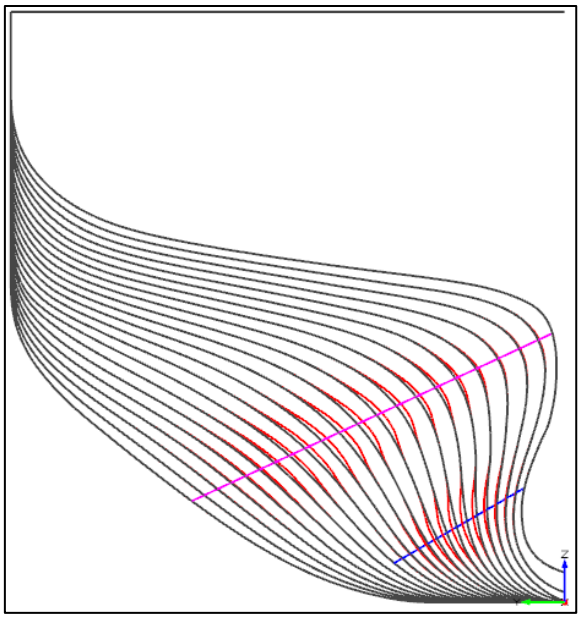

Figure 2-32 – Combined transformations at stern of ship (section view)

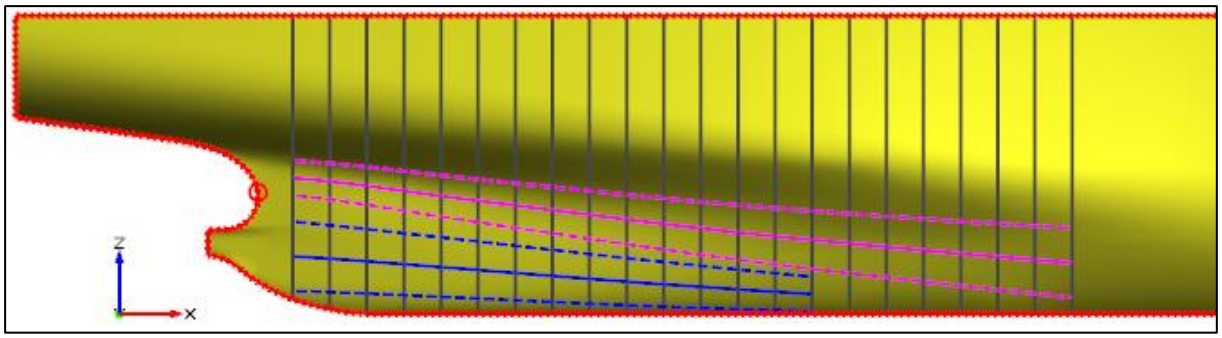

Figure 2-33 – Combined transformations at stern of ship (longitudinal view)

2.3.6. Summary of Inputs and Outputs

In summary, the user has to input the vertical location longitudinal window and diagonal angle for the Aft Waterline/Diagonal Feature to automatically create a Deltasurface ready to transform the aft portion of the hull. The user then should define the vertical Zone of Influence and the maximum amplitude for the transversal variation. Optionally, advanced users can manipulate almost any characteristic of the waterline/diagonal change, including the vertical function, the longitudinal function, and the shape of the desired final waterline/diagonal. Finally, the Feature returns the deformed hull and the transformation, which can be combined with several others to produce an ideal partially parametric model of the stern of a ship.

The user can parameterize anything in the input, but the simplest solution is to parameterize only one: the maximum amplitude of waterline/diagonal change, thus leaving an easy output for a CFD optimization. The tool can be considered a good example of sketched parametric modeling due to its simplicity and intuitiveness.

The following figure shows how the dialog window for the Feature in CAESES looks like, leaving default values.

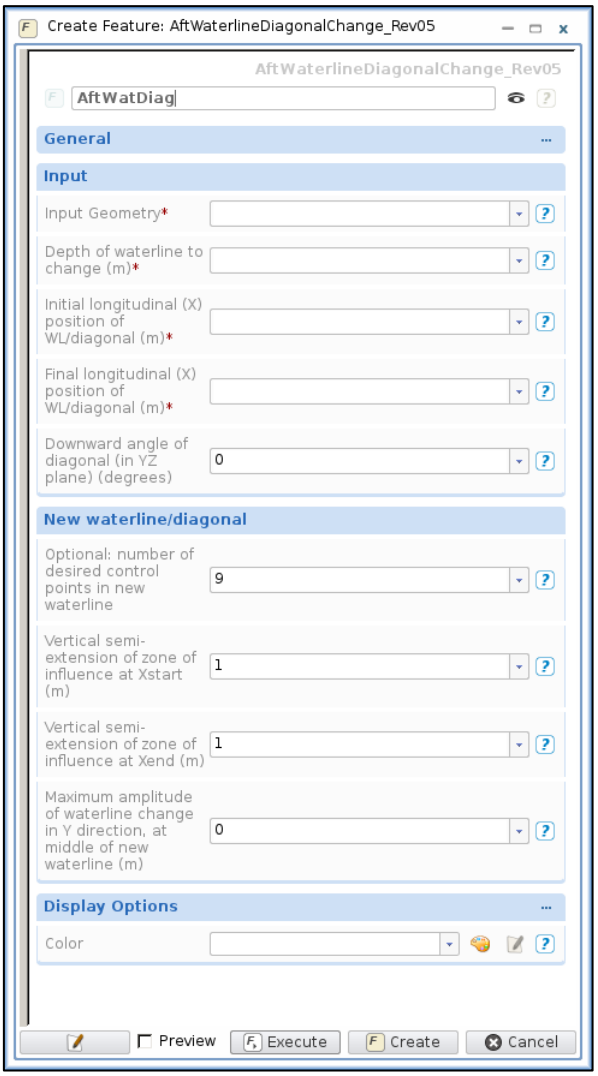

Figure 2-34 – Outlook of dialog window for the Feature

2.3.7. Suggestions for Improvements

A great improvement to the Feature would be regarding its applicability. As stated in the beginning of this Section, inappropriate meshes can make the tool unusable. It seems difficult for the Feature to work around original meshing problems, but some improvements could be tried, or at least a warning message could be given in such cases, so that at least users would not waste their efforts in trying to use a tool where it is not applicable.

Similar to the Bulb Transformation Feature, this tool also struggles a bit with the time it takes for the transformation to be applied. It is also only a few seconds in a standard computer, but this delay should be addressed as well.

Finally, as with the Bulb Transformation, this Feature should be tested with more hulls and cases for robustness of the code.

2.4. CFD Viscous Flow Software FINE/Marine

For the calculation of the hydrodynamic resistance of the ship hull, NUMECA's FINE/Marine software was used; it combines grid generator HEXPRESS and free-surface viscous flow solver ISIS-CFD (developed jointly with École Centrale de Nantes). Its main properties are depicted in the following table.

Table 2-1 – Properties of Fine/Marine's architecture [33]

Grid type	Unstructured
Free surface method	Volume of fluid
Domain discretization	Finite volumes
Turbulence models	RANS (1, 2 eqs.)

FINE/Marine is able to generate an FV mesh from an input domain geometry. The unsteady hydrodynamic resistance problem is solved in the time domain, step by step, while also finding the ship's equilibrium in sinkage and trim for a half-body and the wave elevation along the hull and free surface. An acceleration period is input for the first time steps. The default turbulence model used is the 2-equation k-ω SST model, but it also supports several others.

3. SETUP AND APPLICATION OF TOOLS

In the case of this thesis, an optimization of the resistance of a ship model was desired. The hull geometry was an input, taken from other studies. A partially-parametric model from sketched parametrics was created; a VOF mesh was created on each run and used in the resistance calculation; and an optimization algorithm was used for varying the shape parameters of the model. The following shows a flowchart of the steps taken and the software used for each.

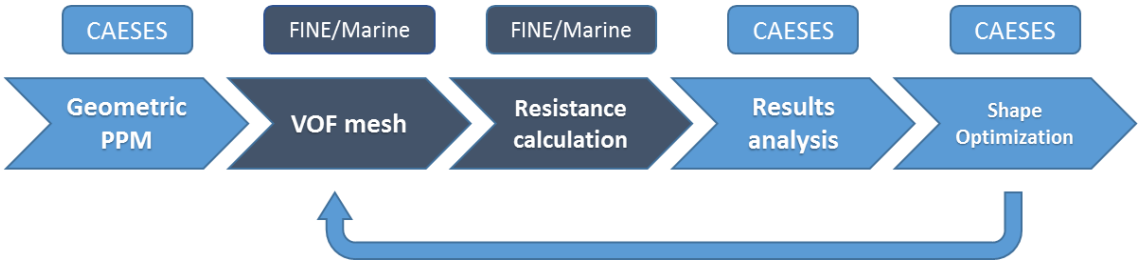

Figure 3-1 – Flowchart of the CFD shape optimization procedure used

3.1. The Model

The ship hull model used was the KCS container ship, a 3600TEU-capacity ship model used in the Gothenburg 2000, 2005 and 2010 workshops on numerical ship hydrodynamics (Larsson et al. 2002 [34], Hino 2005 [35], Larsson et al. 2011 [36]) for both computational and experimental investigations. The scaling factor used was $\lambda = 31.6$. The following information gives its main particulars.

Table 3-1 – Main particulars of hull model used (KCS container ship)

Main particulars	Full scale	Model scale ($\lambda = 31.6$)
Length between perpendiculars L_{PP} (m)	230.0	7.278
Length on waterline L_{WL} (m)	232.5	7.358
Length overall LOA (m)	243.8	7.715
Maximum waterline beam B_{WL} (m)	32.2	1.019
Depth D (m)	19.0	0.601
Draft T (m)	10.8	0.342
Displacement volume ∇ (m^3)	52030	1.6489
Wetted surface area (w/o rudder) S_W (m^2)	9424	9.438
Wetted surface area (with rudder) S_{WR} (m^2)	9539	9.553
Block coefficient CB ($\nabla/L_{PP}B_{WL}T$)	0.6505	0.6505
LCB (% L_{PP}), fwd+	-1.48	-1.48
Service speed V_S (kn), [F_R]	24, [0.26]	2.196, [0.26]

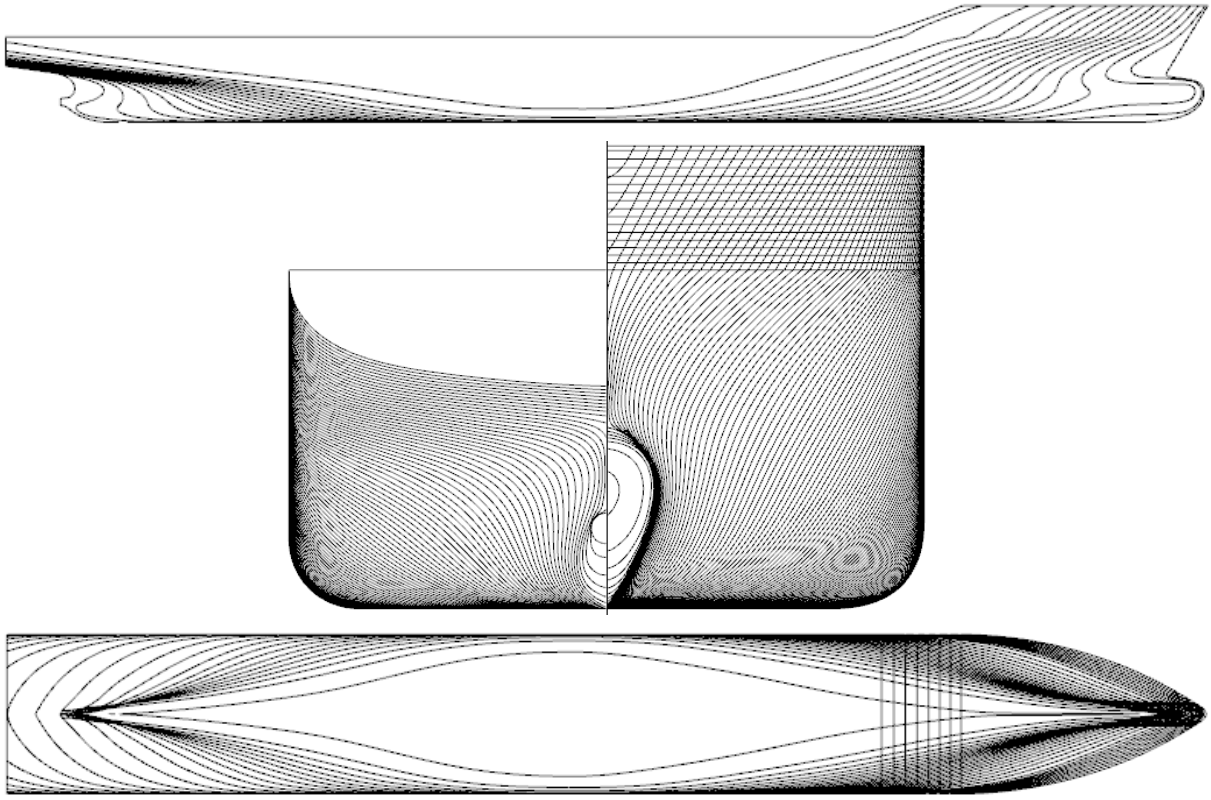

Figure 3-2 – 2D section views of the KCS container ship (transversal view not in scale)

The full scale model was provided in a triangular mesh file (STL), one of the most basic and widespread geometry file types that can be used between different engineering software. The reason was to simulate as well as possible a real situation found in the industry where CFD engineers have to import geometry from other parties.

The service speed of the vessel for the optimization (V_S) was set as 24 knots. However, a small study of the speed of 21 knots (V_{21} = 1.922 m/s in model scale) was also investigated for this study, by comparing its resistance results between the original and the optimized mesh.

Naturally, the water density ρ at full scale is taken as sea water (1025 kg/m^3), while at model scale, it is taken as fresh water (1000 kg/m^3).

The rudder was not present in the model.

The full scale hull model was scaled (reduced) to the model scale just before being used in all CFD simulations.

3.2. Resistance Results from Experiments

The KCS was subjected to several resistance and seakeeping studies. Experimental resistance data were obtained from Kleinsorge et al. (2016 [37]), for comparison with the results obtained here, for a free (2 DoF: trim and sinkage) towed scaled model. The following table gives the original values for the dimensionless total resistance coefficients C_T for the two speeds used in the thesis (in knots and in Froude number), as well as the calculated dimensionalized resistances R_T (in model scale, with $\lambda = 31.6$).

Table 3-2 – Resistance for free (2 DoF) model test experimental results [37]

Speed (kn), [F_R]	$C_T \cdot 10^3$	R_T (N) ($\lambda = 31.6$)
21.0, [0.227]	3.47	61.2
24.0, [0.260]	3.71	85.5

Where:

$$C_T = \frac{R_T}{0.5\,\rho_{water}\,S_{WR}\,V^2} \quad , \quad V = [m/s]$$

It is worth mentioning that the S_{WR} (wetted surface including rudder) was used for dimensionalizing the model test results here, since a rudder-equipped model was used for the free (2 DoF) model tests [38]. The ship model used in the CFD computations for this thesis, however, did not have the rudder present, so the results were not to be directly compared. Nonetheless, the experimental results were to give a good idea of the general accuracy of the CFD computations and the consistency of the results.

3.3. CFD Simulation Setup

The simulation setup for the KCS was mostly the same used by NUMECA (the developer of FINE/Marine) in a different study for the same hull and same speeds [39]. This decision was made because the hull variants were to be similar to the original one, and a full mesh convergence analysis had already been made by NUMECA.

The mesh domain for the simulation in FINE/Marine was created from a box around the model scale hull to be meshed, as a function of the total hull length. The dimensions of the box are summarized in the following figure.

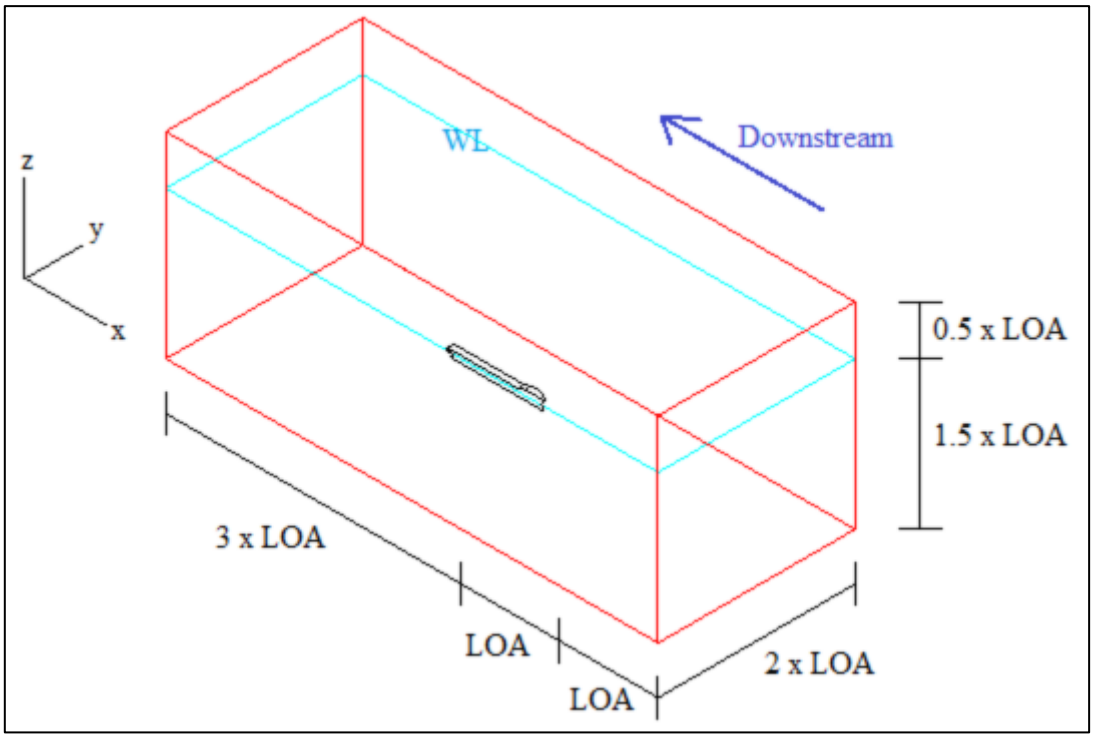

Figure 3-3 – Summary of domain box for CFD calculation

The half-model of the hull was used, with an XZ plane symmetry on the centerline. The fore of the box was defined as a velocity inlet, and the remaining limits were pressure outlets. The initial waterline delimited the separation between air and water.

The domain geometry was exported to HEXPRESS, the automatic mesh-generating module inside FINE/Marine. It generated a mesh of approximately 0.4 million cells while using a y^+ value equal to 30.

The total number of time steps for the simulation was 2000, taking the default step size of $0.005 \cdot L_{PP}/V$. An acceleration period of 500 time steps was used, as well as a relaxation function to impose it.

The number of nonlinear iterations for finding the ship's equilibrium (sinkage and trim) at each time step was set as 5. The water depth was set as infinite. The number of cores used for the optimization procedure was 16. One complete simulation with these configurations took about 1h of real time.

The input setup file for the FINE/Marine simulations is presented in Appendix II.

In order to try to optimize the computation time for the whole optimization procedure, a small convergence analysis was made for the original hull by changing the total time steps and

number of nonlinear iterations to 2500 and 4, instead of 2000 and 5, respectively, and not simultaneously. The conclusion reached was to use the same values as initially, since no significant difference in results nor computation time were detected.

3.4. Shape Optimization Setup

The Sketched Parametric tools created for this thesis were applied in the KCS hull, and a few design variables were chosen, so that geometric variants (new hulls) would be created by CAESES using a design engine. This is detailed in the next paragraphs. Each hull was run in FINE/Marine's CFD solver, and the results were collected and ranked in terms of lowest resistance. A few geometric constraints were also used. The total number of hull variants for the optimization was chosen as 200, considering the limited amount of computer power and time available for this thesis.

The Bulb Transformation tool described in item 2.2 was applied to the KCS full scale model in CAESES using the input parameters described in the following table and figure. The default values are shown for reference. The bulb volume V_B was taken as a target, instead of sectional bulb area A_{BT}. The values set as design variables for the optimization are highlighted in bold font ("dxBulb", "dzBulb", "BulbVolumeFactor"). The variables with subscript "-original" refer to the original hull values for such, which were calculated using the tool before running the simulations.

Table 3-3 – Bulb Transformation tool: parameters applied for KCS Optimization

Input parameter	Value	Default
Design draft (m)	10.8	-
Aft Zone Slider	2.0	1.0
Factor for Upper Aft Zone	0.0	0.0
Factor for Lower Aft Zone	5.0	5.0
Factor for Center of Aft Zone	1.5	1.5
Center of inflation	y_{max} at FP	y_{max} at FP
Target: bulb length (L_B)	$L_{B\text{-original}}$ + **dxBulb**	-
Target: bulb nose height (Z_B)	$Z_{B\text{-original}}$ + **dzBulb**	-
Target: bulb volume (V_B)	$V_{B\text{-original}}$ * **BulbVolumeFactor**	-
Precision tolerance for finding V_B	2%	1% - 10%

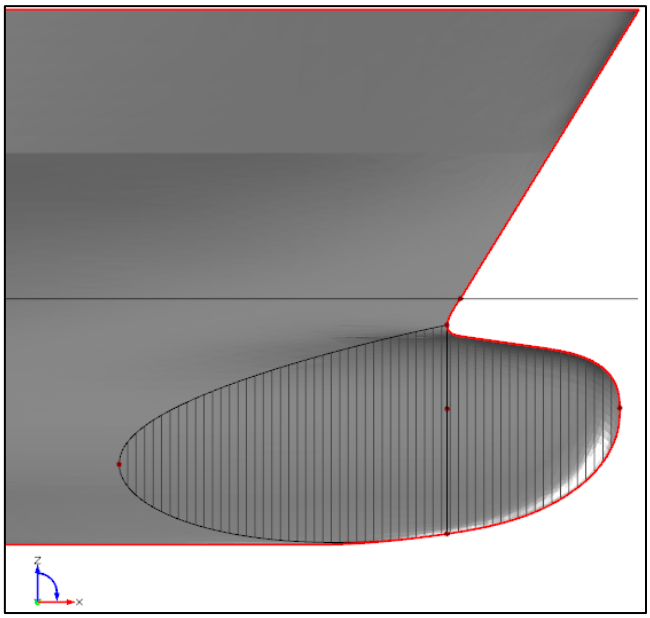

Figure 3-4 – Bulb Transformation tool applied for KCS Optimization

The original hull values of the target parameters are shown in the following table.

Table 3-4 – Original hull values for target parameters

Parameter	Value
Original bulb length ($L_{B\text{-original}}$)	7.589 m
Original bulb nose height ($Z_{B\text{-original}}$)	5.996 m
Original bulb volume ($V_{B\text{-original}}$)	538.5 m^3

The Aft Waterline/Diagonal tool described in item 2.3 was applied to the KCS full scale model too, using the input parameters described in the following table and figure. A combination of two diagonals was used: a high and a low one.

Table 3-5 – Aft Waterline/Diagonal tool: parameters applied for KCS Optimization (high diag.)

	Input parameter	Value	Default
Higher Diagonal	Waterline depth (m)	8.0	-
	x_{start} (m)	10.0	-
	x_{end} (m)	55.0	-
	Downward angle of diagonal (deg)	25	0
	Vertical semi-extension of zone of influence at x_{start} (m)	1.0	-
	Vertical semi-extension of zone of influence at x_{end} (m)	2.0	-
	Maximum amplitude of change in Y direction, at middle of waterline/diagonal (m)	**dyDiagHigh**	-
	Number of control points	9	-

Table 3-6 – Aft Waterline/Diagonal tool: parameters applied for KCS Optimization (low diag.)

	Input parameter	Value	Default
Lower Diagonal	Waterline depth (m)	4.0	-
	x_{start} (m)	10.0	-
	x_{end} (m)	40.0	-
	Downward angle of diagonal (deg)	30	0
	Vertical semi-extension of zone of influence at x_{start} (m)	2.0	-
	Vertical semi-extension of zone of influence at x_{end} (m)	1.0	-
	Maximum amplitude of change in Y direction, at middle of waterline/diagonal (m)	**dyDiagLow**	**-**
	Number of control points	9	-

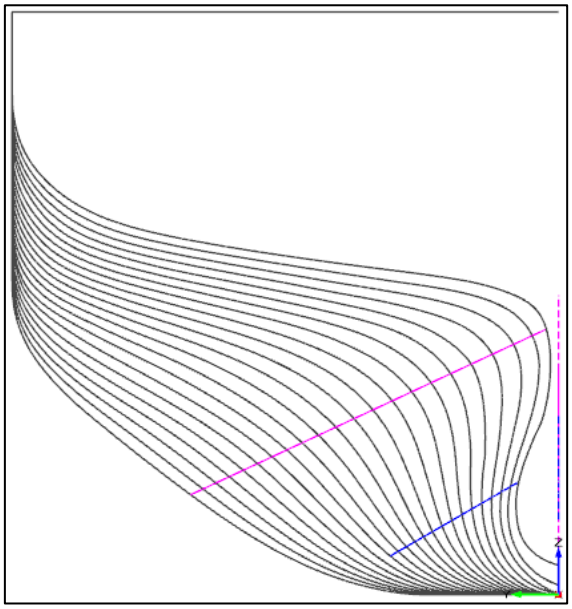

Figure 3-5 – Aft Waterline/Diagonal tool applied for KCS Optimization: high and low diagonals

A few values for the design variables were manually tested and the hull form was analyzed to verify possible ranges for these variables where the hull geometry would still be smooth, realistic, and not radically different from the original one. These ranges were chosen in a conservative way, meaning that the hull geometry variants were not very different from the original hull. It would have been possible to strain these ranges a bit further, but the focus of the study was more on the method used (sketched parametric modeling) than how different the final optimized hull resistance could be.

The following table shows the ranges selected for the design variables. It is important to remember that the values for such parameters are real numbers, meaning that there are infinite possible values for them in the chosen intervals (ranges).

Table 3-7 – Ranges for design variables for KCS Optimization

Design variable	Lower limit	Original value	Upper limit
dxBulb (m)	-1.4	0.0	1.0
dzBulb (m)	-0.9	0.0	1.1
BulbVolumeFactor	1.0	1.0	1.1
dyDiagHigh (m)	-0.3	0.0	0.1
dyDiagLow (m)	0.0	0.0	0.25

The design engine chosen for varying the design variables and creating the hull variants in CAESES was the Sobol sequence algorithm, which is already input in CAESES as an option. It is recognized as a good design engine for "the first stage of an optimization process" [31], populating the design space more or less evenly.

Although only 5 design variables were picked, they already generate a design space of significant complexity. For a more thorough and complete optimization, more design variables should be chosen, as well as wider intervals, which would imply the need for more calculations and computation time to get good results.

Three simple geometric constraints were used for the optimization: a minimum displacement volume ∇, a minimum LCB, and a maximum LCB. They were set to avoid very radical hull variants to be tested. Their values in full scale are shown in the following table.

Table 3-8 – Constraints for KCS Optimization

Parameter	Constraint	Original hull
Min. ∇ (m^3)	$\geq$ 51830	52030
Min. LCB (% L_{PP})	$\geq$ -2.0	-1.48
Max. LCB (% L_{PP})	$\leq$ -1.0	-1.48

In the optimization procedure, the resistance in only one vessel speed was calculated by CFD: the service speed V_S = 24 knots. This was done due to time limitations and due to the fact that the operational profile of the vessel was not known. For better hull optimizations, the expected vessel speed profile should be known and taken into account so that the fuel consumption would be minimized.

3.5. Analysis of Results

Resistance results were obtained in the time domain.

Considering the symmetry plane and coordinate system used, resistance result values from FINE/Marine were multiplied by -2 in the end to get the correct absolute values.

3.5.1. Original Hull: Resistance Results

The dynamic wetted surface S_{Wdyn} of the original hull (in model scale) was calculated by FINE/Marine for both speeds, and is shown in the following table. They were used for converting resistance results to nondimensional coefficients.

Table 3-9 – Dynamic wetted surface results for original hull

V (kn)	S_{Wdyn} (m^2) (model scale)	S_{Wdyn} (m^2) (full scale)
21.0	9.83	9819
24.0	9.90	9890

Dynamic wetted surface results were about 5% higher than the static wetted surface area S_W from Table 3-1.

The following chart shows the full time series results for the (model scale) resistance drag R_T (or F_X) of the original hull in the service speed of 24 knots (2.2m/s in model scale). The final 200 time steps from which the average is taken are highlighted in the chart, as well as the average (Fx_{AVG}), the dimensionless resistance coefficient calculated from it (Ct_{AVG}), and the standard deviation σ in relation to the value of Fx_{AVG}.

Figure 3-6 – Full resistance results for original hull, at service speed

The following figure is zoomed in on the highlighted area: the last 200 time steps.

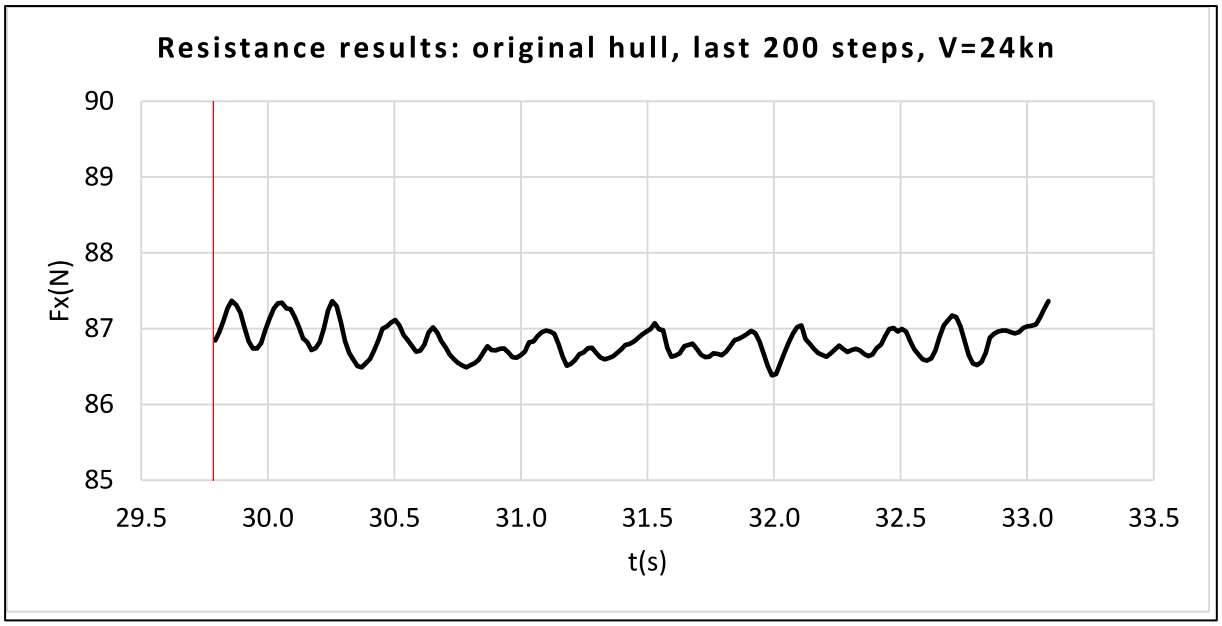

Figure 3-7 – Resistance results for original hull at service speed: last 200 steps

A convergence analysis was performed, increasing the total time steps by 25% (to 2500), but the oscillations did not decrease (or increase). As the standard deviation was found to be less than 1%, results were considered satisfactory.

The following chart shows the full time series for the original hull at the speed of 21 knots (1.9m/s in model scale).

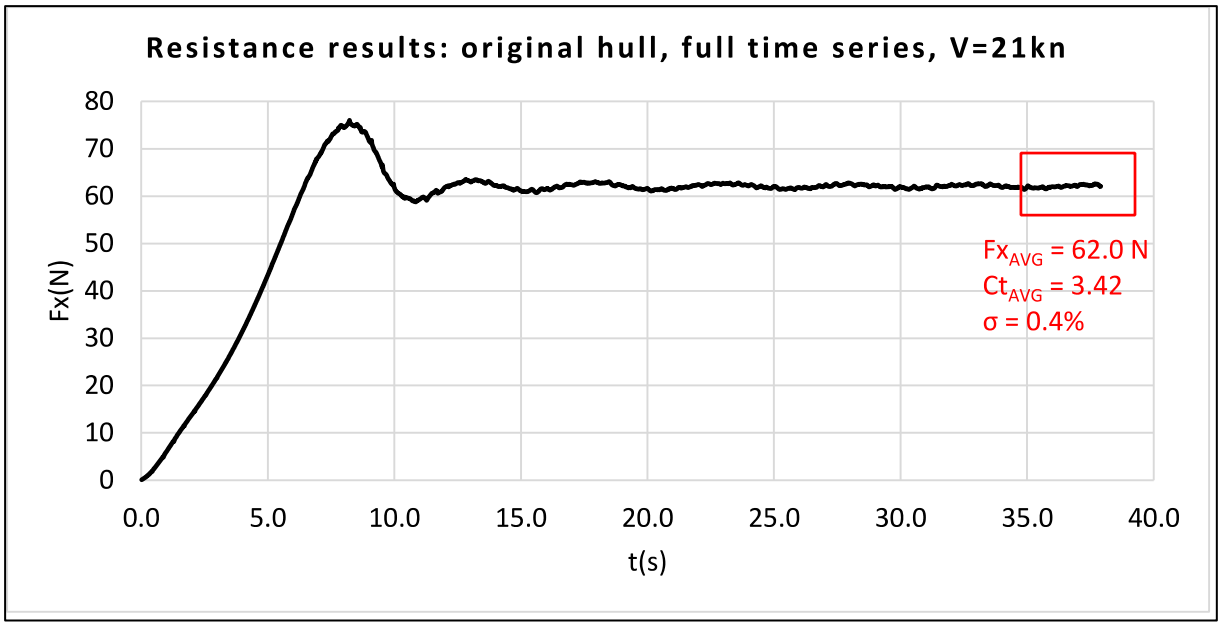

Figure 3-8 – Full resistance results for original hull, at speed of 21kn

The following shows the last 200 time steps for the same speed.

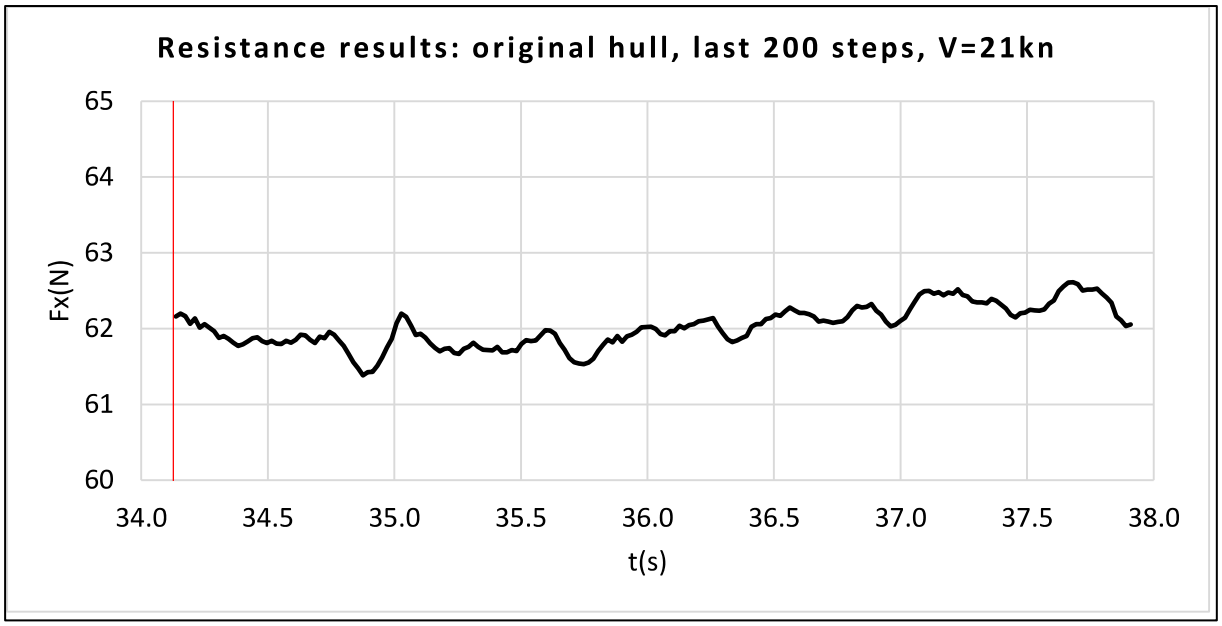

Figure 3-9 – Resistance results for original hull at speed 21kn: last 200 steps

A comparison of the CFD results obtained and the experimental ones from Table 3-2 is shown in the following table.

Table 3-10 – Dimensionless resistance results comparison: experiments vs. CFD

Speed (kn), [F_R]	$C_T \cdot 10^3$		
	Exp.	CFD	Diff.
21.0, [0.227]	3.47	3.42	1%
24.0, [0.260]	3.71	3.63	2%

The results for both speeds were compatible, indicating that the CFD setup in FINE/Marine was adequate.

3.5.2. Shape Optimization Results

With a cost of 1h for each CFD simulation in FINE/Marine, a total of around 200 hours was needed for completion of all simulations in the service speed (24 knots in full scale). Convergence was reached for all of cases in the final time steps, with some small noise/oscillations.

The resistance results R_T (or F_X) were taken as the average results from the final 200 time steps. Considering all the 200 simulations, the standard deviations reached a minimum of 0.2% and a maximum of 0.4%, which were considered satisfactory.

The following charts give the average resistance R_T over the last 200 time steps of each hull variant, in relation to the design variables used. The large red point indicates the original hull's results.

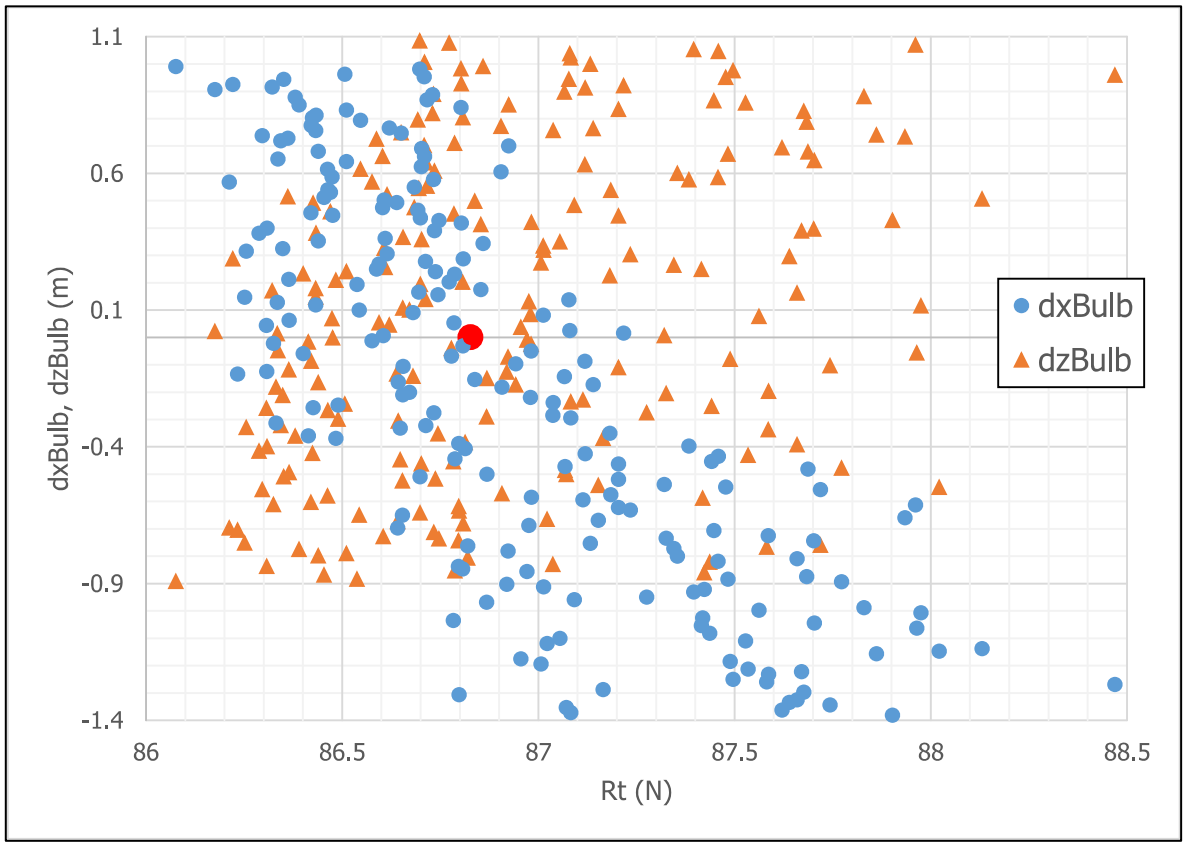

Figure 3-10 – R_T results vs. design variables "dxBulb" and "dzBulb"

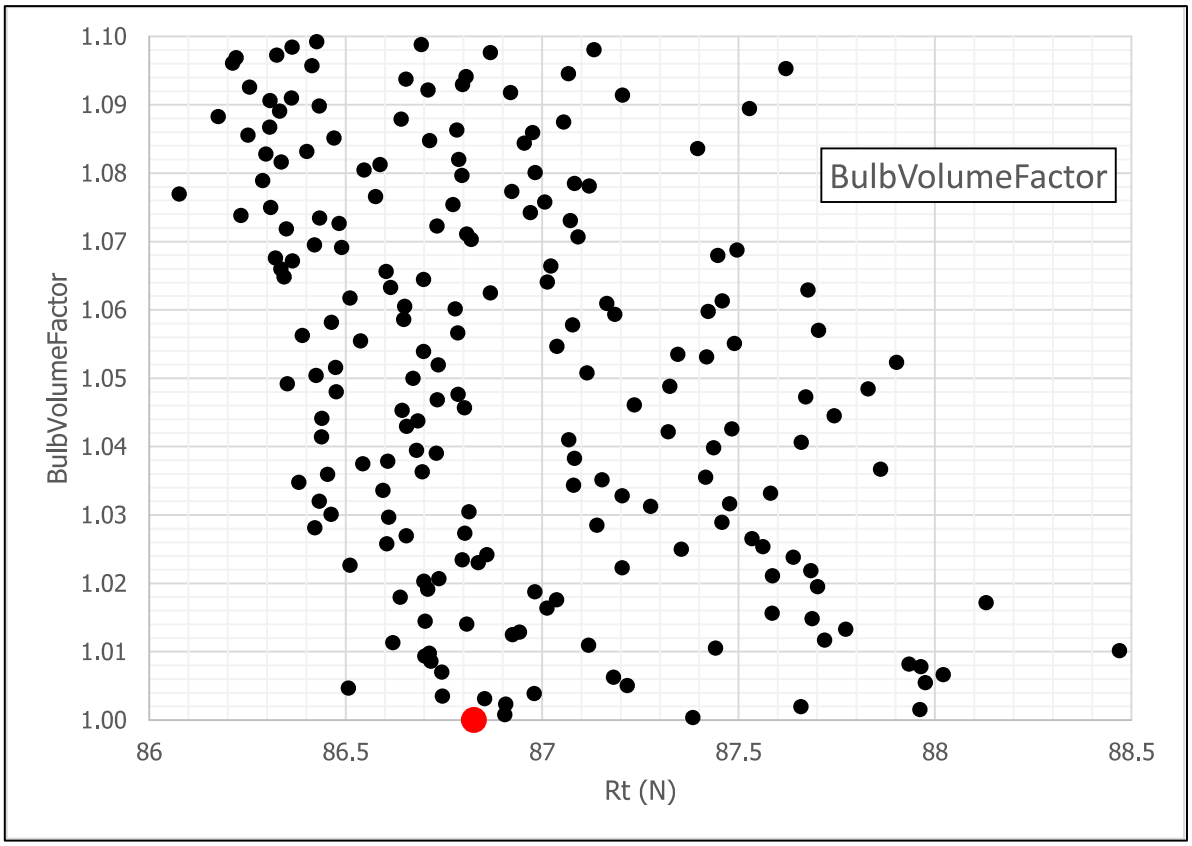

Figure 3-11 – R_T results vs. design variable "BulbVolumeFactor"

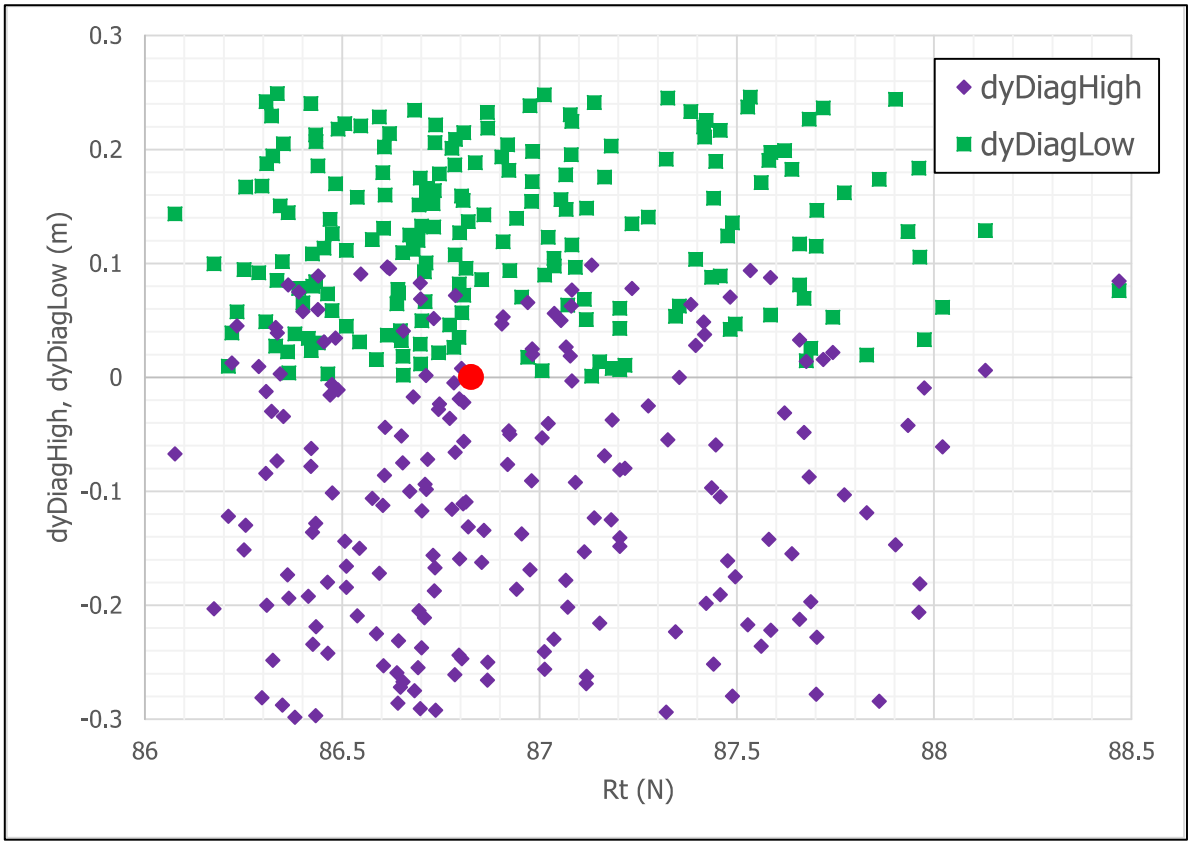

Figure 3-12 – R_T results vs. design variables "dyDiagHigh" and "dyDiagLow"

There were no strong correlations found between the resistance results and the design variables. However, weak correlations could be seen for dxBulb and BulbVolumeFactor, indicating that longer and bigger bulbs were generating more favorable resistance results for the KCS model.

The results tables in Appendix III give the average resistance R_T over the last 200 time steps of each series, as well as their standard deviation σ, for each hull variant tested.

3.5.3. Optimum Hull: Geometry Result

The following table gives the values for the design variables for the optimized hull shape.

Table 3-11 – Design variables' values found for optimum hull

Design variable	Optimum hull	Original hull	Interval
dxBulb (m)	0.990625	0.0	[-1.4 , 1.0]
dzBulb (m)	-0.892188	0.0	[-0.9 , 1.1]
BulbVolumeFactor	1.07695	1.0	[1.0 , 1.1]
dyDiagHigh (m)	-0.0671875	0.0	[-0.3 , 0.1]
dyDiagLow (m)	0.143555	0.0	[0.0 , 0.25]

Summarizing: the optimum hull has a longer bulb, with lower bulb nose height, and an inflated bulb volume. It also has a slightly thinner higher diagonal in the aft zone chosen (Table 3-5), and a wider lower diagonal in the aft zone.

The following figures attempt to show the visible differences between the original and the optimum hull by means of sections (buttocks, offsets and waterlines) equally spaced. Some differences are quite soft, while others are more pronounced. The black lines are from the original hull, and the red lines are from the optimum hull.

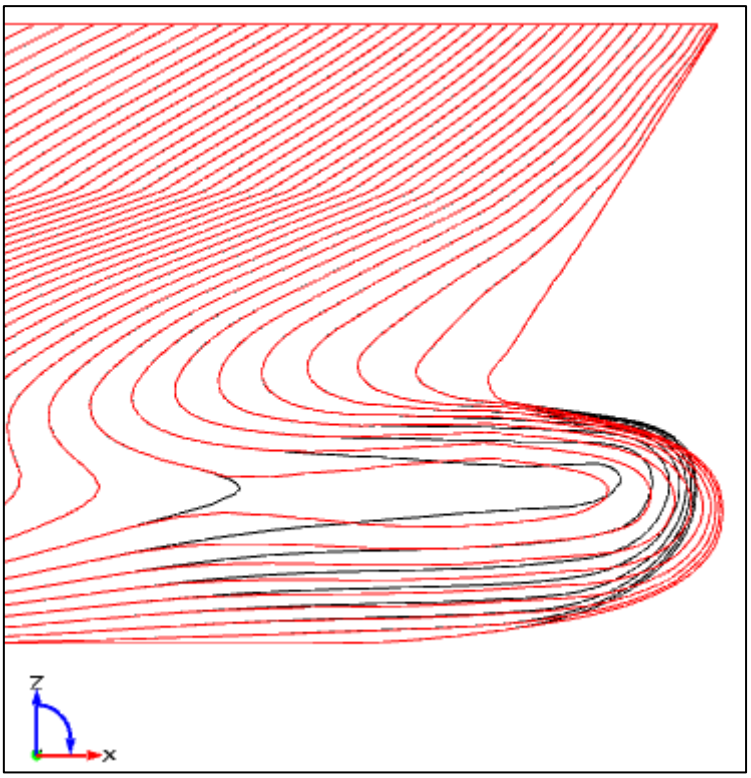

Figure 3-13 – Bulb region buttocks (black = original, red = optimum)

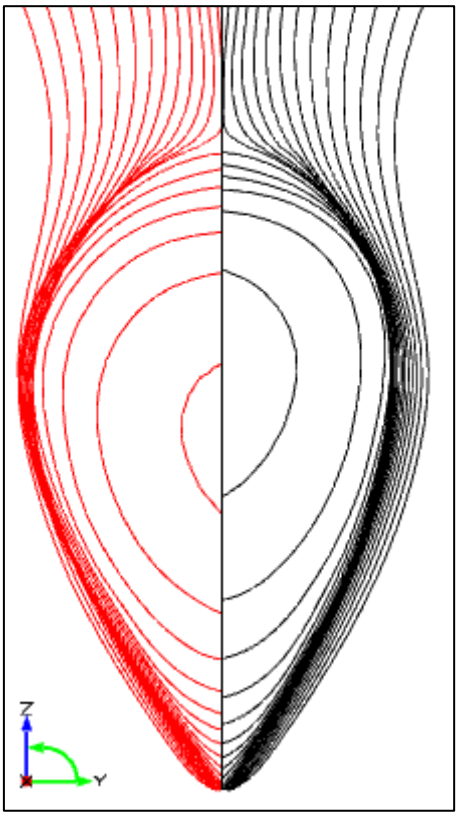

Figure 3-14 – Bulb region offsets (black = original, red = optimum)

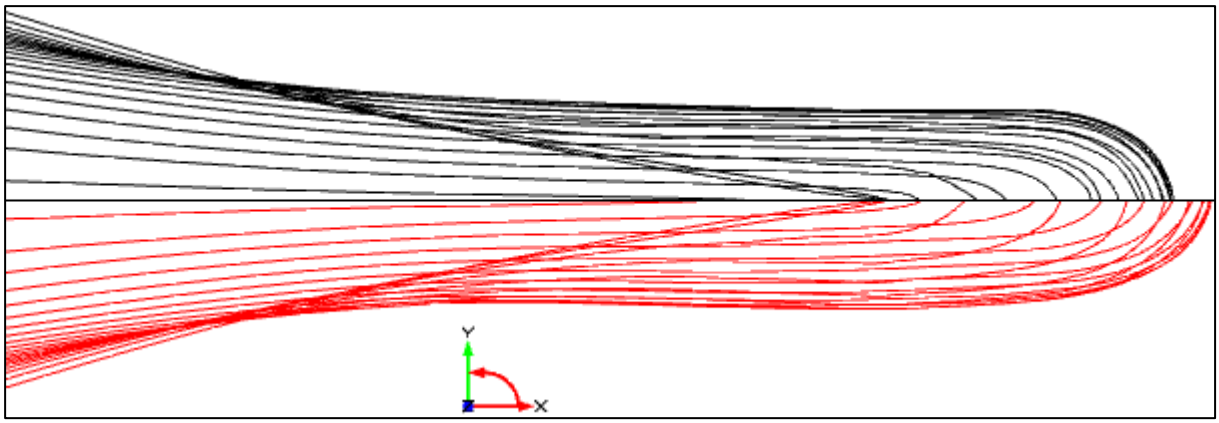

Figure 3-15 – Bulb region waterlines (black = original, red = optimum)

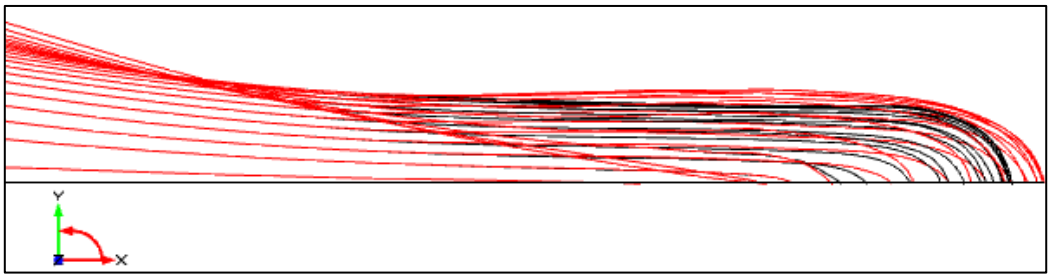

Figure 3-16 – Bulb region waterlines, same side (black = original, red = optimum)

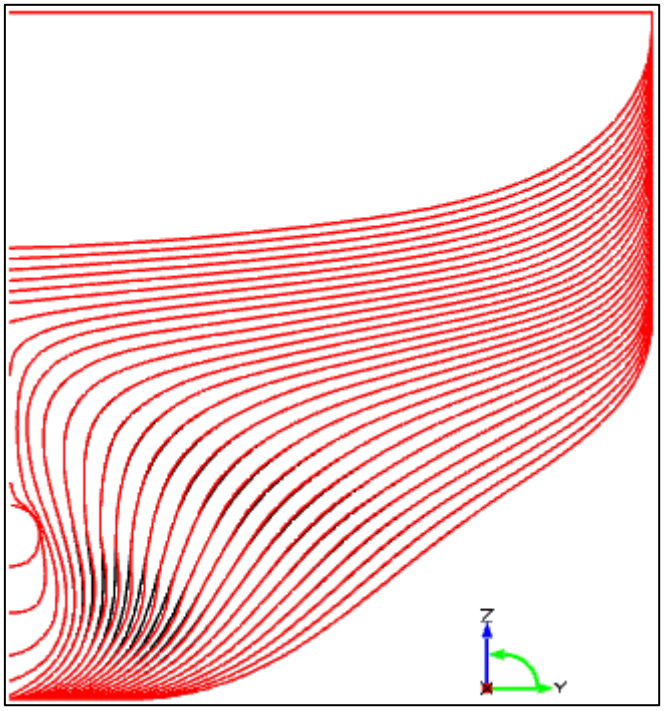

Figure 3-17 – Aft region offsets (black = original, red = optimum)

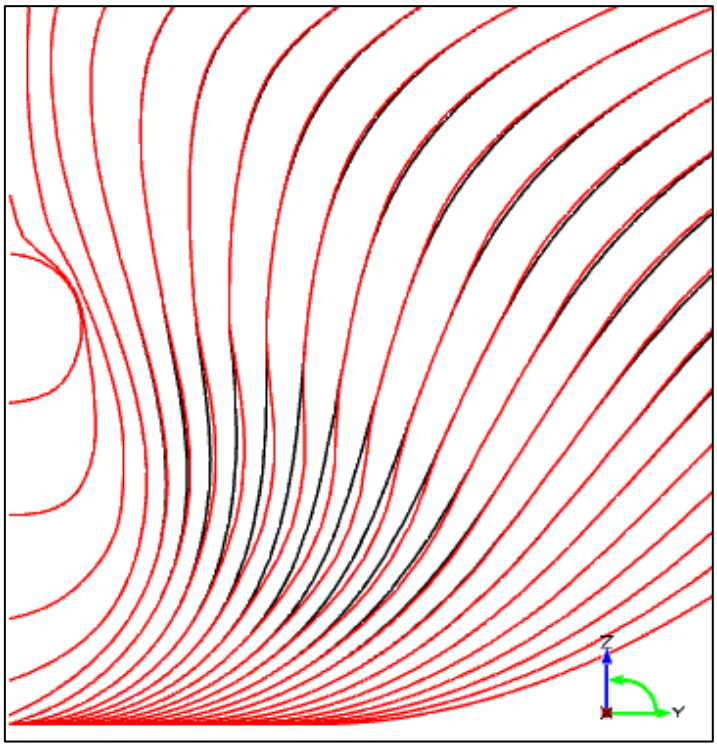

Figure 3-18 – Aft region offsets, detail (black = original, red = optimum)

Following is a table showing the values of the bulb parameters for the original and the optimum hull (in full scale).

Table 3-12 – Bulb parameters comparison

Bulb parameter	Optimum hull	Original hull	Difference
Bulb Length L_B (m)	8.579	7.589	13%
Bulb Nose Height Z_B (m)	5.104	5.996	-15%
Bulb Volume V_B (m^3)	604.9	538.5	12%

For illustration purposes, the following figures show a bulb profile comparison of the 5 best hulls, in terms of resistance results from the CFD calculations, as well as the original one.

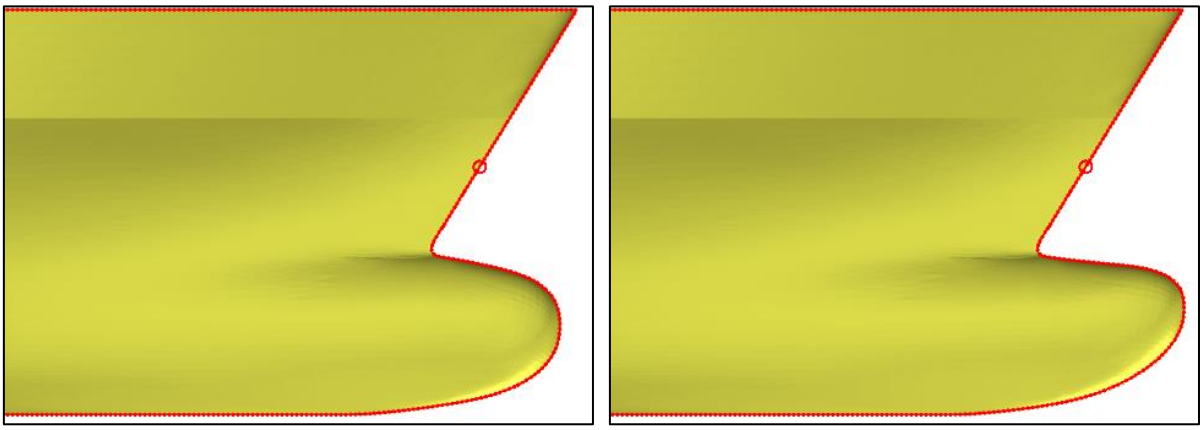

Figure 3-19 – 5th (left) and 4th (right) best hulls in terms of R_T

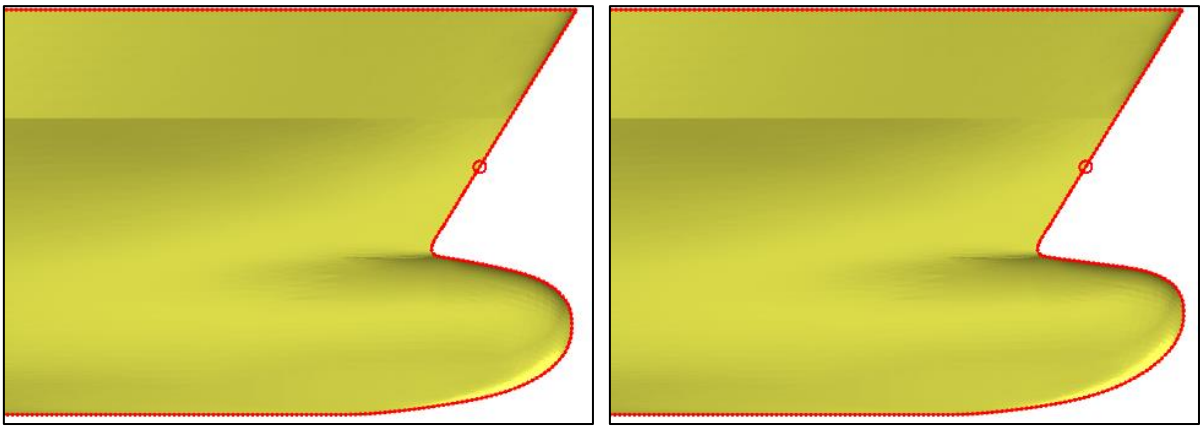

Figure 3-20 – 3rd (left) and 2nd (right) best hulls in terms of R_T

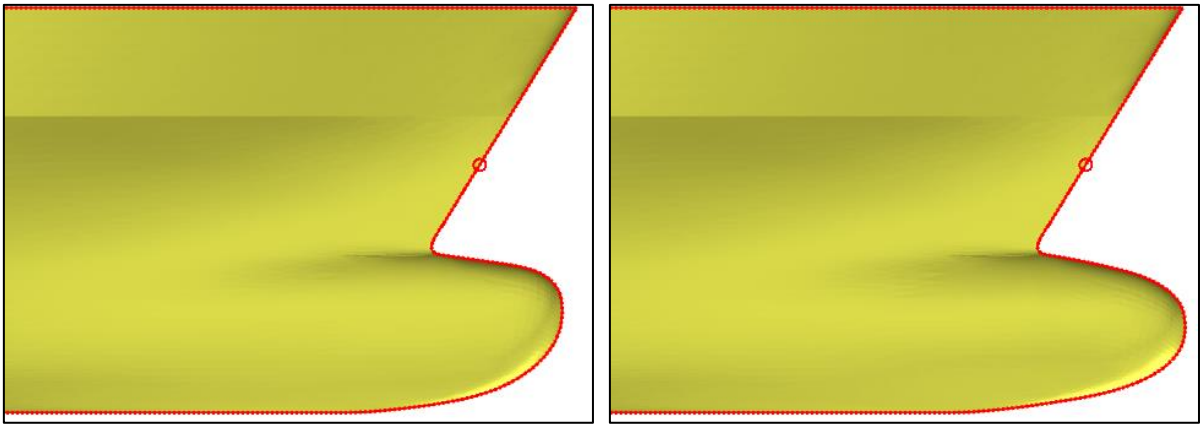

Figure 3-21 – Original (left) and Optimum/best (right) hulls in terms of R_T

3.5.4. Optimum Hull: Resistance Results

The dynamic wetted surface S_{Wdyn} of the optimum hull (in model scale) was calculated by FINE/Marine for both speeds, and is shown in the following table.

Table 3-13 – Dynamic wetted surface results for optimum hull

V (kn)	S_{Wdyn} (m^2) (model scale)	S_{Wdyn} (m^2) (full scale)
21.0	9.86	9845
24.0	9.94	9923

The S_{Wdyn} of the optimum hull is only slightly higher than the original by ~0.3%.

The following chart gives the full time series results for the resistance of the optimum hull in the service speed of 24 knots. The final 200 time steps taken are highlighted.

Figure 3-22 – Full resistance results for optimum hull, at service speed

The last 200 time steps are shown in the following chart.

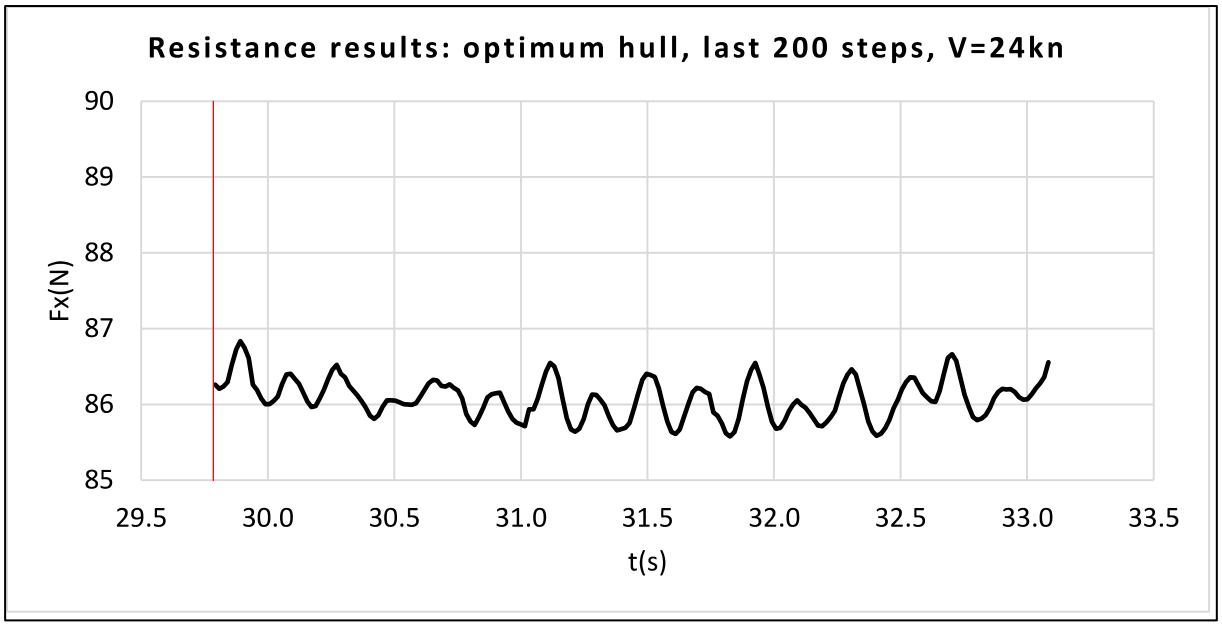

Figure 3-23 – Resistance results for optimum hull at service speed: last 200 steps

The resistance for the speed of 21 knots was calculated for the optimum hull as well, and the results charts were as follows.

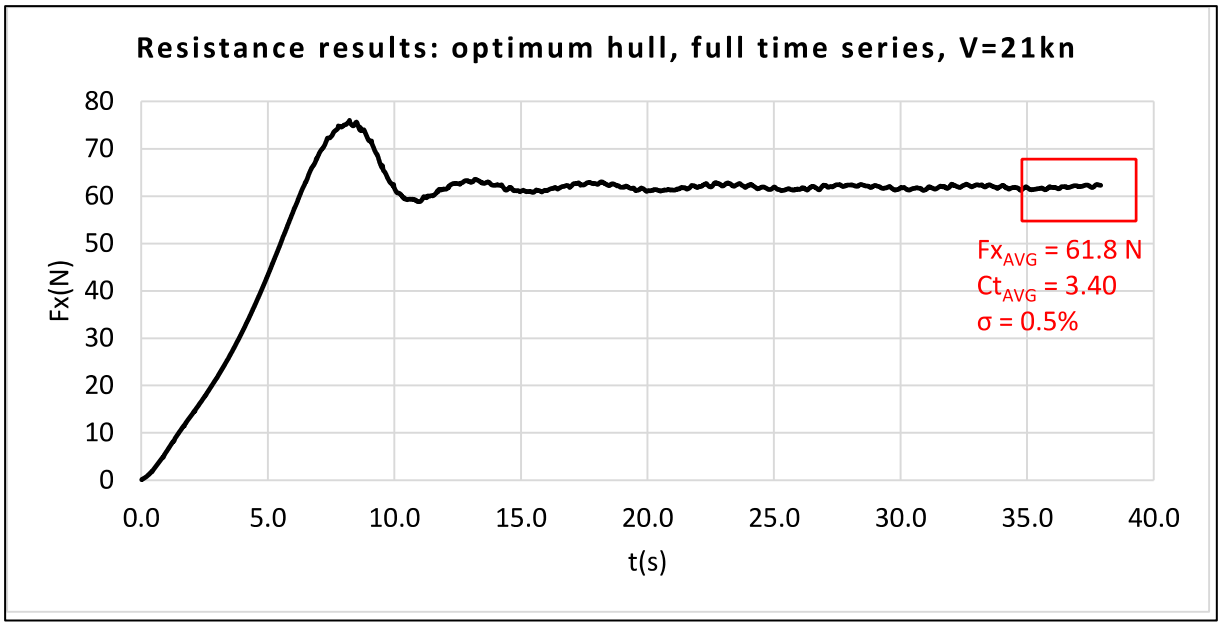

Figure 3-24 – Full resistance results for optimum hull, at speed of 21kn

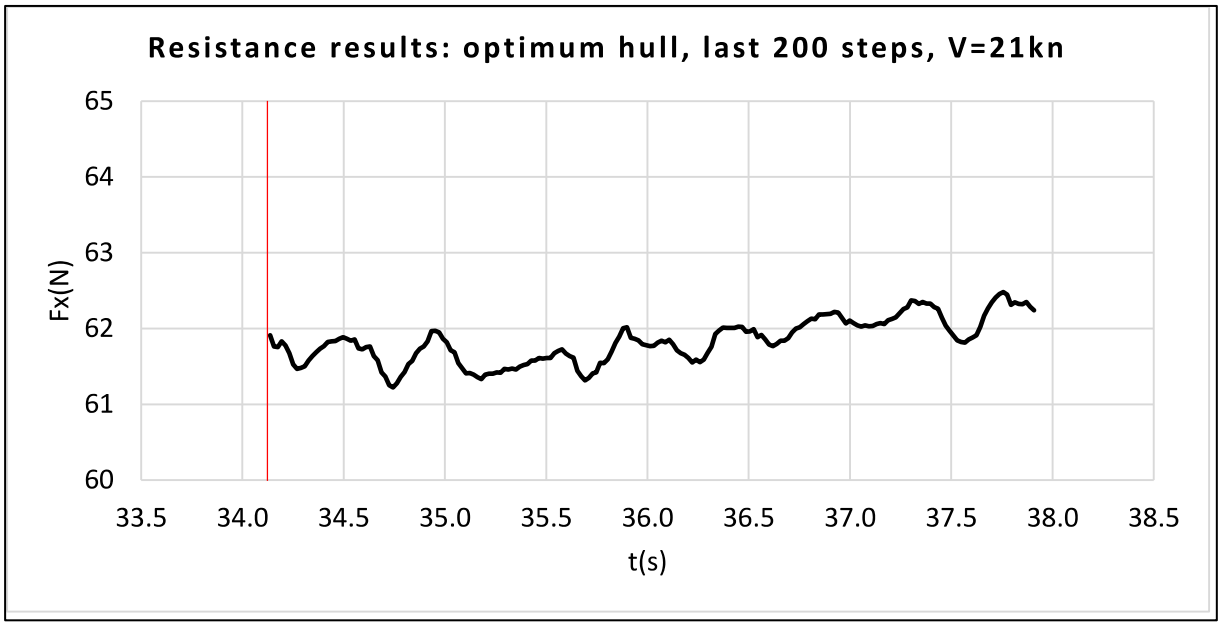

Figure 3-25 – Resistance results for optimum hull at speed of 21kn: last 200 steps

CFD results for the optimum hull in the speed of 21 knots had a slightly poorer quality than for the service speed of 24 knots, presenting a standard deviation of 0.5%.

Summarizing the resistance results, the following table and chart give a comparison of the resistance results from the original and optimized hull, with dimensionalized and dimensionless values, as well as the standard deviation σ.

Table 3-14 – Resistance results comparison: original and optimized hulls

Speed (kn)	$C_T \cdot 10^3$		R_T (N)		Diff.	σ	
	Original	Optimum	Original	Optimum		*Original*	*Optimum*
21.0	3.42	3.40	62.0	61.8	0.3%	*0.4%*	*0.5%*
24.0	**3.63**	**3.59**	**86.8**	**86.1**	**0.9%**	*0.2%*	*0.3%*

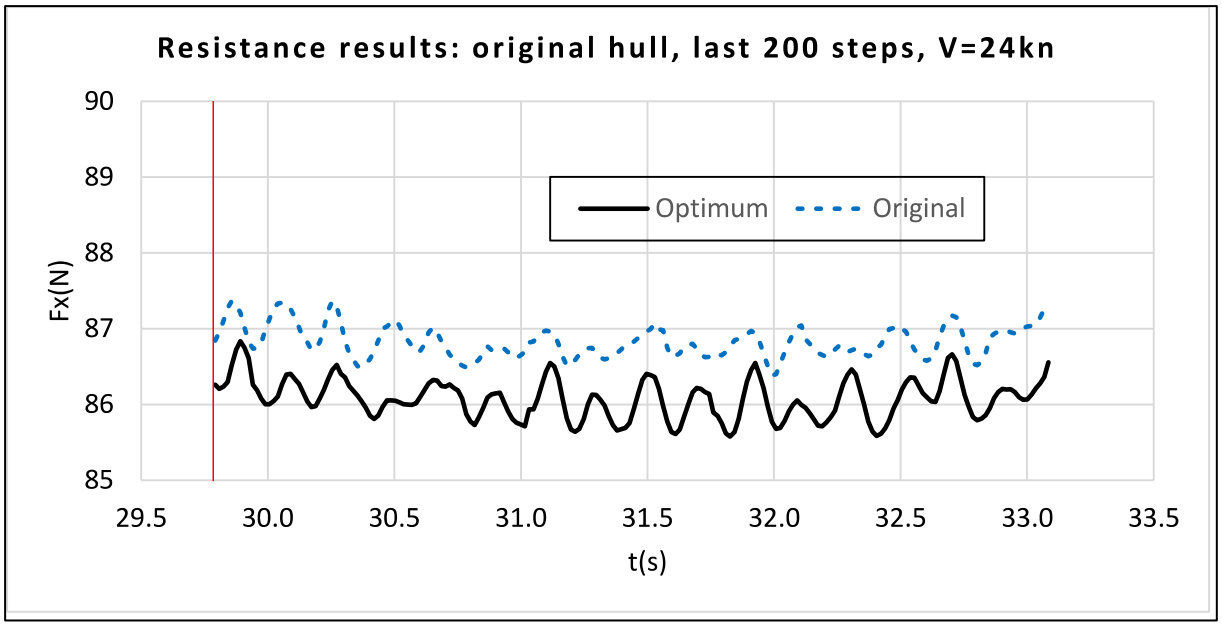

Figure 3-26 – Resistance results for optimum and original hull at service speed: comparison

The R_T for the optimum hull is lower than for the original one by a margin of 0.9%. The value of σ hovers around one third of this, so we can safely conclude that the optimum hull does present a better geometry in terms of hydrodynamic total drag resistance.

The following figures, taken from FINE/Marine's postprocessor, show the KCS original and optimum hulls' free surface elevation pattern when in the service speed of 24 knots. An option to take the 10% last steps of the simulation (200) was checked in FINE/Marine, to be consistent with the results collected. The same color scales are used for both speeds. Values are in model scale, and isolines are used for indication of regions of same free surface elevation. The mean free surface is located at z = draft of the hull, which is 10.8m in full scale and 0.341m at model scale.

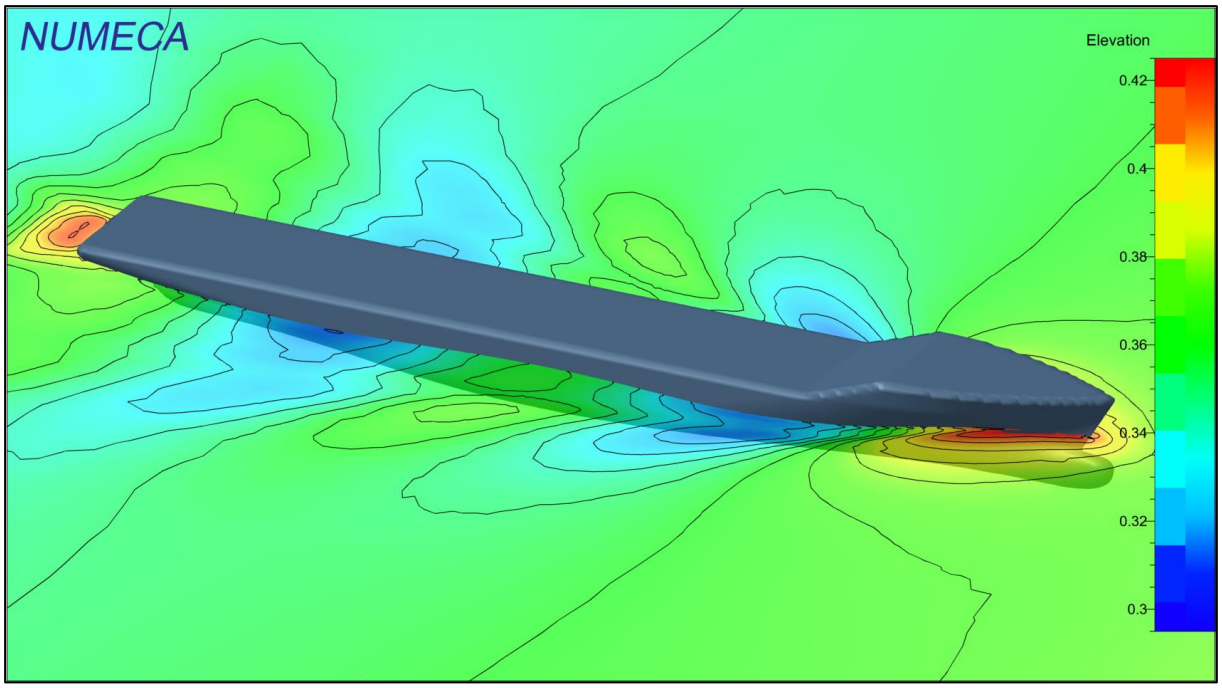

Figure 3-27 – Original hull: free surface elevation at service speed of 24kn

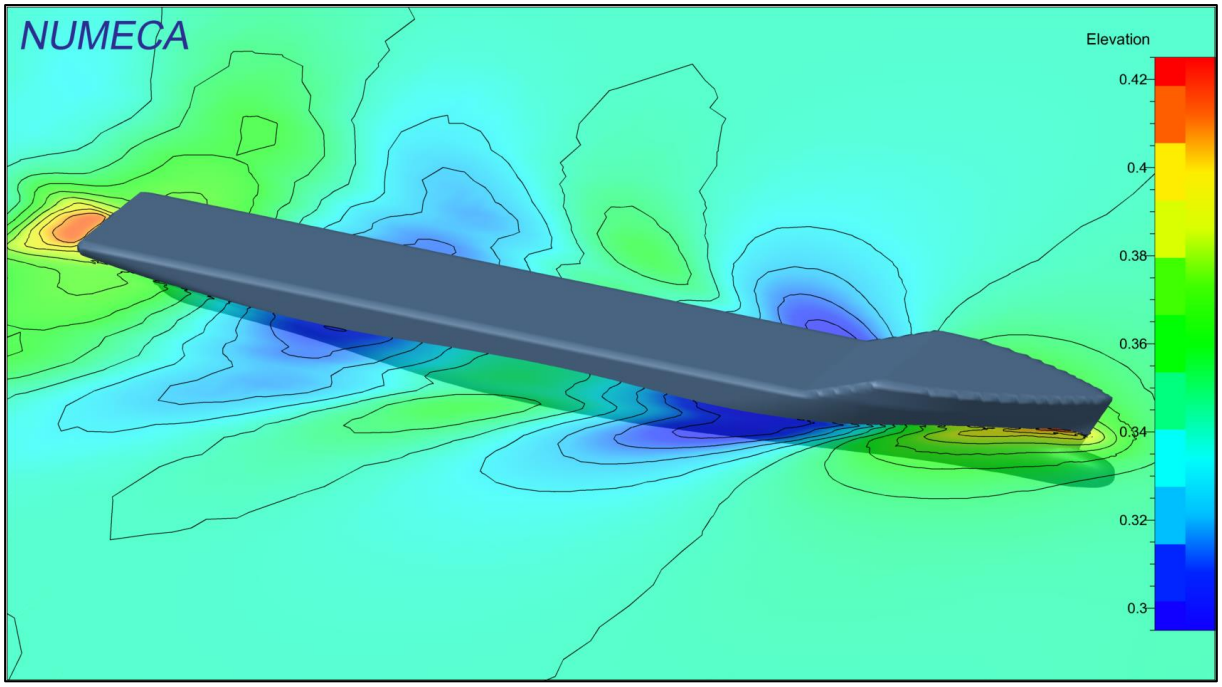

Figure 3-28 – Optimum hull: free surface elevation at service speed of 24kn

It is possible to see that the optimum hull produces significantly lower elevations at the bulb region, but also lower depressions at the suction regions at the shoulders of the vessel, at aft and fore. It is also visible that the general waterline level around the ship is slightly more elevated for the original hull.

The following figures show the wave pattern for the speed of 21kn, for reference.

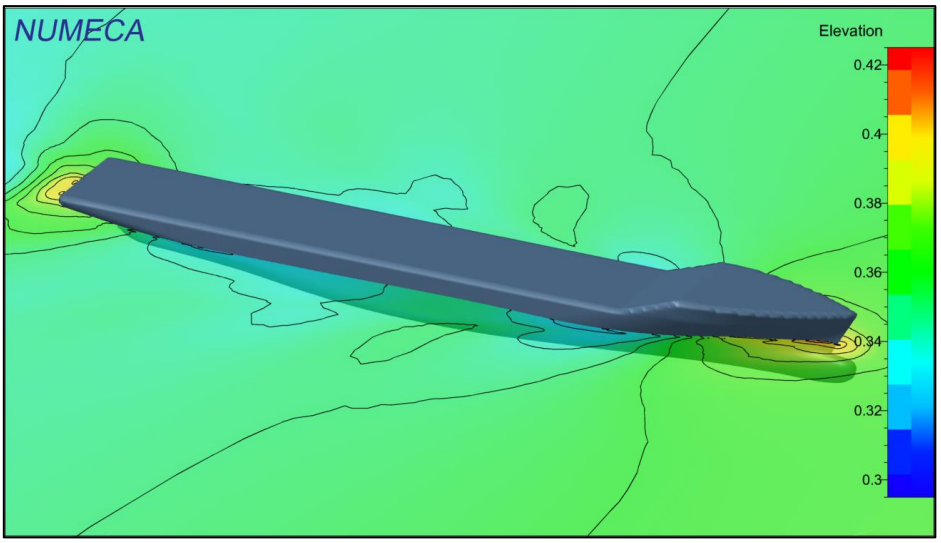

Figure 3-29 – Original hull: free surface elevation at speed of 21kn

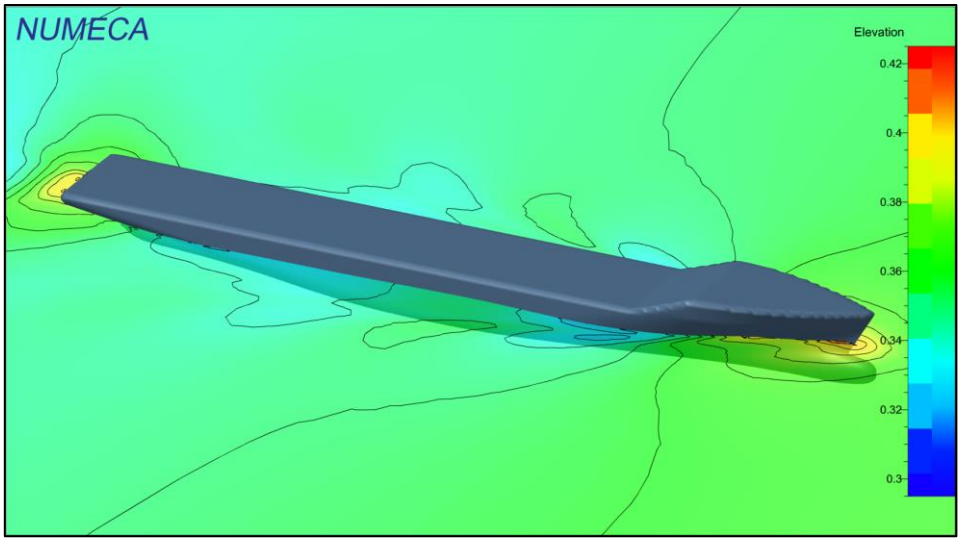

Figure 3-30 – Optimum hull: free surface elevation at speed of 21kn

Regarding constraints, the following results were obtained for the optimum hull.

Table 3-15 – Constraints for KCS: comparison between optimum and original hulls

Parameter	Constraint	Optimum hull	Original hull
Min. ∇ (m^3)	$\geq$ 51830	52102	52030
Min. LCB (% L_{PP})	$\geq$ 2.0	-1.41	-1.48
Max. LCB (% L_{PP})	$\leq$ 1.0	-1.41	-1.48

The optimum hull presented a slightly higher displacement value than the original one while at the same time having a smaller drag resistance, which is an optimum scenario in practical terms.

It would have been more interesting for this study if the optimization process produced a hull with a higher optimization gain, instead of only ~1%. In terms of CFD calculations this result can be seen as OK, but in model test experiments this might not be distinguishable enough, as a range of accuracy of ±1% on the resistance results can be expected. However, since the method was the main focus of the study instead of how different optimized the final hull would be, and taking into account also the limitations of time and computer power that were present, the results obtained can be considered a success of the Sketched Parametric tools and methods developed in this thesis and applied for a CFD shape optimization procedure.

It should also be mentioned that the optimization procedure could have been performed in a smarter, more efficient way. Instead of computing all of the 200 hull variants in a row starting from the original design, a smaller sample (e.g. 50 variants) could have been run, and then local searches could be performed around the best 5 of those, for example. There are many paths for optimizing a target function, and it is difficult to know for sure if the final resistance results would have been better using one path or another. However, it is statistically more likely that a combined global-local search would have been more successful than the simple global search performed here. More time would be needed to prepare and evaluate such a procedure, though, and this was a very limited resource along the study.

4. CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

The purpose of this thesis was to present a method for sketching geometry variations in hull meshes, making them ready for simulation and for creation of variants, and applying it to an industrial case such as a CFD shape optimization. These goals have been achieved by developing two Sketched Parametric tools within the framework of parametric-modeling software CAESES – the Bulb Transformation and Aft Waterline/Diagonal modification Features –, and applying them to a resistance optimization of the KCS containership hull with the use of CFD software FINE/Marine. Discussions of the usability, applications, recommendations and suggestions for the tools developed have been made as well. Additionally, a small literature review of the subjects of parametric modeling in CFD optimization and sketching mesh transformations has been made.

The Sketched Parametric tools introduced here have shown good potential for further applications by generating realistic hull variants with ease from a given baseline geometry. The fact that it was possible to reduce the drag resistance of the KCS hull with only a few design variables input in the tools shows the power that it can offer to CFD engineers in need of simple user-oriented ways of optimizing geometries. Despite being very easy to use, developing such Sketched Parametric tools requires a good understanding level of CAD, parametric modeling and geometry manipulation. The tools' and the method's usefulness, practicability and frequency of use in a working environment have to be evaluated before deciding for taking them.

A disadvantage of the method – and of creating any partially parametric model – is that the user must have an idea of where the hull geometry should be changed to achieve an objective flow function, such as minimization of resistance. CFD engineers may have a good idea in most cases, but it is not always guaranteed to work. The use of an Adjoint CFD analysis before applying Sketched Parametric tools would be ideal, since it would give locations for achieving the objective. It is suggested that future works could investigate this relation.

5. REFERENCES

[1] BRENNER, M., HARRIES, S., PALLUCH, J. & RATZ, J., 2015. "Simulation-Ready CAD for Fast Turn-Around Time in CFD and Optimization". *NAFEMS European Conference: Computational Fluid Dynamics (CFD) – Beyond the Solve*, December 2015, Germany.

[2] LACKENBY, H., 1950. "On The Systematic Geometrical Variation of Ship Forms". *RINA Transactions, Vol. 92*, 1950, UK.

[3] HICKS, R.M., MURMAN, E.M. & VANDERPLAATS, G.N., 1974. "An Assessment of Airfoil Design by Numerical Optimization". *NASA Technical Memorandum*, TM X-3092, July 1974, USA.

[4] NOWACKI, H., CREUTZ, G. & MUNCHMEYER, F.C., 1977. "Ship Lines Creation by Computer – Objectives, Methods, and Results". *1st International Symposium on Computer-Aided Hull Surface Definition*, September 1977, USA.

[5] NOWACKI, H., LIU, D. & LÜ, X., 1990. "Fairing Bézier Curves with Constraints". *Computer Aided Geometric Design, Vol. 7*, June 1990, Netherlands.

[6] NOWACKI, H. & LÜ, X., 1994. "Fairing Composite Polynomial Curves with Constraints". *Computer Aided Geometric Design, Vol. 11*, February 1994, Netherlands.

[7] HARRIES, S. & ABT, C., 1997. "Parametric Curve Design Applying Fairness Criteria". *International Workshop on Creating Fair and Shape-Preserving Curves and Surfaces*, September 1997, Germany.

[8] HARRIES, S., 1998. "Parametric Design and Hydrodynamic Optimization of Ship Hull Forms". *PhD Thesis, Institut für Schiffs-und Meerestechnik, Technical University Berlin*, 1998, Germany.

[9] MAISONNEUVE, J.J., HARRIES, S., MARZI, J., RAVEN, H.C., VIVIANI, U. & PIIPO, H., 2003. "Towards Optimal Design of Ship Hull Shapes". *8th International Marine Design Conference*, May 2003, Greece.

[10] HOEKSTRA, M. & RAVEN, H.C., 2003. "A Practical Approach to Constrained Hydrodynamic Optimization of Ships". *NAV 2003: International Conference on Ship and Shipping Research*, June 2003, Italy.

[11] HARRIES, S., ABT, C. & HOCHKIRCH, K., 2004. "Modeling Meets Simulation – Process Integration to Improve Design". *Sonderkolloquium zu Ehren der Professoren Hagen, Schlüter und Thiel, Universität Duisburg-Essen*, July 2004, Germany.

[12] ABT, C. & HARRIES, S., 2007. "A New Approach to Integration of CAD and CFD for Naval Architects". *6th International Conference on Computer Applications and Information Technology in the Maritime Industries*, April 2007, Italy.

[13] HARRIES, S., 2008. "Serious Play in Ship Design". Tradition and Future of Ship Design in Berlin, Colloquium, Technical University Berlin, February 2008, Germany.

[14] WILSON, W., HENDRIX, D. & GORSKI, J., 2010. "Hull Form Optimization for Early Stage Ship Design". *Naval Engineers Journal, Vol. 122*, June 2010, USA.

[15] BILIOTTI, I., BRIZZOLARA, S., VIVIANI, M., VERNENGO, G., RUSCELLI, D., GALLIUSSI, M., GUADALUPI, D. & MANFREDINI, A., 2011. "Automatic Parametric Hull Form Optimization of Fast Naval Vessels". *11th International Conference on Fast Sea Transportation*, September 2011, USA.

[16] HAN, S., LEE, Y.-S. & CHOI, Y.B., 2012. "Hydrodynamic Hull Form Optimization Using Parametric Models". *Journal of Marine Science and Technology, Vol. 17*, February 2012, Japan.

[17] WEICKGENNANT, S., ZIMMER, A. & GINES, J., 2014. "Parametric Adjoint Optimization: Using Adjoint Sensitivities to Quantify the Effect of Design Variables in a Parametric CFD Model". *6th European Conference on Computational Fluid Dynamics*, July 2014, Spain.

[18] BRENNER, M., HARRIES, S., KROGER, J. & RUNG, T., 2015. "Parametric-Adjoint Approach for the Efficient Optimization of Flow-Exposed Geometries". *6th International Conference on Computational Methods in Marine Engineering*, June 2015, Italy.

[19] HARRIES, S., ABT, C. & BRENNER, M., 2015. "Upfront CAD – Parametric Modeling Techniques for Shape Optimization". EUROGEN: International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems, September 2015, UK.

[20] KHO, Y. & GARLAND, M., 2005. "Sketching Mesh Deformations". ACM SIGGRAPH 2005: Annual Conference on Computer Graphics and Interactive Techniques, April 2005, USA.

[21] SEDERBERG, T.W. & PARRY, S.R., 1986. "Free-Form Deformation of Solid Geometric Models". *ACM SIGGRAPH 1986: Annual Conference on Computer Graphics and Interactive Techniques*, January 1986, USA.

[22] COQUILLART, S., 1990. "Extended Free-form Deformation: A Sculpturing Tool for 3D Geometric modeling". *ACM SIGGRAPH 1990: Annual Conference on Computer Graphics and Interactive Techniques*, January 1990, USA.

[23] MACCRACKEN, R. & JOY, K.I., 1996. "Free-form Deformations with Lattices of Arbitrary Topology". *ACM SIGGRAPH 1996: Annual Conference on Computer Graphics and Interactive Techniques*, August 1996, USA.

[24] SINGH, K. & FIUME, E., 1998. "Wires: A Geometric Deformation Technique". *ACM SIGGRAPH 1998: Annual Conference on Computer Graphics and Interactive Techniques*, July 1998, USA.

[25] IGARASHI, T., MATSUOKA, S. & TANAKA, H., 1999. "Teddy: A Sketching Interface for 3D Freeform Design". *ACM SIGGRAPH 1999: Annual Conference on Computer Graphics and Interactive Techniques*, August 1999, USA.

[26] LLAMAS, I., KIM, B., GARGUS, J., ROSSIGNAC, J. & SHAW, C.D., 2003. "Twister: A Space-Warp Operator for the Two-Handed Editing of 3D Shapes". *ACM SIGGRAPH 2003: Annual Conference on Computer Graphics and Interactive Techniques*, July 2003, USA.

[27] BOTSCH, M. & KOBBELT, L., 2004. "An Intuitive Framework for Real-Time Freeform Modeling". *ACM SIGGRAPH 2004: Annual Conference on Computer Graphics and Interactive Techniques*, August 2004, USA.

[28] NEALEN, A., SORKINE, O., ALEXA, M. & COHEN-OR, D., 2005. "A Sketch-Based Interface for Detail-Preserving Mesh Editing". *ACM SIGGRAPH 2005: Annual Conference on Computer Graphics and Interactive Techniques*, July 2005, USA.

[29] BRIZZOLARA, S., VERNENGO, G., PASQUINUCCI, C.A. & HARRIES, S., 2015. "Significance of Parametric Hull Form Definition on Hydrodynamic Performance Optimization". *Marine 2015, VI International Conference on Computational Methods in Marine Engineering*, June 2015, Italy.

[30] PASQUINUCCI, C.A., 2016. "Free Form Deformation and Surrogate Models for Exploration, Sensitivity and Optimization of Complex Hull Forms". *Master Thesis, Dipartimento di Ingegneria Navale, Elettrica, Elettronica e delle Telecomunicazioni, University of Genova*, September 2016, Italy.

[31] CAESES' website, available at: <https://www.caeses.com/>. Accessed on October 2016.

[32] KRACHT, A.M., 1978. "Design of Bulbous Bows". *SNAME Transactions, Vol. 86*, January 1978, USA.

[33] NUMECA INTERNATIONAL, 2016. "NUMECA Online Documentation Platform". *Version: FINE™/Marine 5.1*, April 2016, Belgium.

[34] LARSSON, L., STERN, F. & BERTRAM, V., 2002. "Gothenburg 2000 – A Workshop on Numerical Hydrodynamics". *Department of Naval Architecture and Ocean Engineering, Chalmers University of Technology, Report No. R-10:122*, 2002, Sweden.

[35] HINO, T., 2005. "Proceedings of CFD Workshop Tokyo 2005". *National Maritime Research Institute, Report*, 2005, Japan.

[36] LARSSON, L., STERN, F. & VISONNEAU, M., 2011. "CFD in Ship Hydrodynamics – Results of the Gothenburg 2010 Workshop". *Marine 2011, IV International Conference on Computational Methods in Marine Engineering*, September 2011, Portugal.

[37] KLEINSORGE, L., LINDNER, H., & BRONSART, R., 2016. "A Computational Environment for Rapid CFD Ship Resistance Analyses". *Proceedings of PRADS2016, 13th International Symposium on Practical Design of Ships and Other Floating Structures*, September 2016, Denmark.

[38] INTERNATIONAL TOWING TANK COMMITTEE (ITTC), 2011. "Final Report and Recommendations to the 26th ITTC". *Proceedings of 26th ITTC – Volume II, The Specialist Committee on Computational Fluid Dynamics*, September 2011, Brazil.

[39] KERNER, L., 2014. "KCS Resistance Calculation". *Presentation on Benchmark Report*, September 2014.

APPENDIX I: CODE OF THE FEATURES CREATED

The following codes are from the Features developed in CAESES' environment for this thesis. It uses the programming language of CAESES' own API.

_____

Bulb Transformation (Target Volume) Feature:

```
// Sketched parametrics: bulb transformation - target volume (bulb length, bulb nose height, inflation based on target volume)
// By Martin Gundelach, October 2016
// Objective: after importing a hull trimesh, easily transform the bulb.


//-------------------------------------------------------------------------------------------------------//
// Creation of objects used in feature

// Output trimesh geometry
imageTriMesh GeoOut()

// Forward Perpendicular and hull cut
line CutWL()
curveintersectionpoint pIntFP()
point FPpoint()
line FP()
curveintersectionpoint pIntFPdown()
imagecurve CutCL()

// Finding bulb profile
point pUp()
point pDown()
double precision_tolerance(5E-3)
line CutBulb()
curveintersectionpoint pIntBulbDown()
imagecurve BulbProf()

// Finding furthest point of bulb and some bulb parameters
point pB()
double LB()
double ZB()
double HB()
double BB()

// Creating Aft Zone of Influence of bulb for Volume calculation
point pAftZoneUp()
point pAftZoneDown()
```

point pAftZoneCenter()
bsplinecurve AftZoneOutline()
polycurve ZoneOutline()
sectionGroup SecBulb()
offsetGroup OffBulb()
offsetGroupAssembly OffAssBulb()
point pBaft()
double pBaftPos()
double VB()

// Finding section at Bulb aftmost point, plus some parameters
sectionGroup A_BTsecGrp()
bsplinecurve A_BTsec()
line CloseSec()
polycurve WholeSec()
double A_BT()
point pA_BT()

// Bulb length transformation
bsplinecurve LineDshiftLB()
bsplinecurve LineDshiftLBZBaux()
deltashift DshiftLB()
deltashift DshiftLBaux()
deltaproduct DshiftLBproduct()

// Bulb nose height transformation
bsplinecurve LineDshiftZB()
deltashift DshiftZB()
deltashift DshiftZBaux()
deltaproduct DshiftZBproduct()

// Uniting LB and ZB transformations
deltasum Dsum()

// Auxiliary curve and points for curve engine/metasurface creation
point pAux1()
point pAux2()
line cAux()

// Metasurface and curve engine for volume transformation
curveEngine Cengine()
metasurface MetaSurf()
surfacedeltashift SurfDelta()
point pA_BTproj()

// Bisection method parameters for finding target volume
unsigned i(0) //counter for finding yhigh and ylow
unsigned j(0) //counter for finding ymid, in case yhigh or ylow does not provide close enough result
unsigned imax(10) //max iterations for i

```
unsigned jmax(10) //max iterations for j
double yhigh(0)
double ylow(0)
double ymid(0)


//-------------------------------------------------------------------------------------------------------//
// Execution of feature: setting up and finding bulb properties

GeoOut.setSource(GeoIn)
GeoOut.setShowTriangles(false)

// Getting the CenterLine (CL) curve from the input geometry
bsplinecurve CL(GeoOut.getEdge(0))
FObjectList points(CL.getPointList())
foreach (FVector3 pt : points)
  pt.setY(0)
endfor

CutWL.setStartPos([CL.getMax(0),0,CL.getMin(2)+Draft])
CutWL.setEndPos([CL.getMin(0),0,CL.getMin(2)+Draft])
pIntFP.setCurveA(CutWL)
pIntFP.setCurveB(CL)

FPpoint.setVector(pIntFP)
FP.setStartPos(FPpoint-[0,0,Draft])
FP.setEndPos(FPpoint)

pIntFPdown.setCurveA(FP)
pIntFPdown.setCurveB(CL)

CutCL.setCurve(CL)
CutCL.setDomain([pIntFP.getParameterOnCurveB(),pIntFPdown.getParameterOn-
CurveB()])

if (CutCL.getMin(0) < (FPpoint:x-precision_tolerance))
  pUp.setVector(CutCL.getpos(CutCL.getMin(0,true)))
  CutBulb.setStartPos(pUp-[0,0,Draft])
  CutBulb.setEndPos(pUp)
  pIntBulbDown.setCurveA(CutBulb)
  pIntBulbDown.setCurveB(CL)
  pDown.setVector(pIntBulbDown)
else
  pUp.setVector(FPpoint)
  pDown.setVector(pIntFPdown)
endif

HB = pUp.getZ()-CL.getMin(2)

BulbProf.setCurve(CL)
```

```
BulbProf.set-
Do-
main([CL.getParameterShortestDistanceSquared(pUp),CL.getParameterShortestDistanceSq
uared(pDown)])
pB.setVector(BulbProf.getPos(BulbProf.getMax(0,true)))

LB = pB.getX()-pUp.getX()
ZB = pB.getZ()-CL.getMin(2)

pAftZoneCenter.setVector([pUp:x-Slider*LB*ZCmultiplier,0,(pUp:z+pDown:z)/2])
pAftZoneUp.setVector([pUp:x-Slider*ZUmultiplier*BulbProf.getTanVec(0).getNormal-
ized():x,0,pUp:z-Slider*ZUmultiplier*BulbProf.getTanVec(0).getNormalized():z])
pAftZoneDown.setVector([pDown:x+Slider*ZDmultiplier*BulbProf.getTanVec(1).getNor-
malized():x,0,pDown:z+Slider*ZDmultiplier*BulbProf.getTanVec(1).getNormalized():z])
AftZoneOutline.setPointList([pUp,pAftZoneUp,pAftZoneCenter,pAftZoneDown,pDown])
ZoneOutline.setCurveList([AftZoneOutline,BulbProf])
ZoneOutline.setAutomaticOrientation(true)
ZoneOutline.setParametrization("unit speed")

SecBulb.setSurfaces([GeoOut])
SecBulb.setPositions([AftZoneOutline.getMin(0)+0.001,  BulbProf.getMax(0)-0.001  :
round((Slider+1)*Noffsets)])
OffBulb.setOffsets(SecBulb.getSections())
OffBulb.cutMinMax(2,ZoneOutline.getMin(2),ZoneOutline.getMax(2))
OffAssBulb.setGroups([OffBulb])

pBaftPos = AftZoneOutline.getMin(0,true)
pBaft.setVector(AftZoneOutline.getPos(pBaftPos))
FObjectList offsets(OffBulb.getObjects(FOffset))
foreach (FOffset off : offsets)
  off.setShowOrientation(false)
  if (off.at(0):x < pUp:x+precision_tolerance)
    double UpCut(AftZoneOutline.ft(0,off.at(0):x,0,pBaftPos))
    double DownCut(AftZoneOutline.ft(0,off.at(0):x,pBaftPos,1))
    off.cutMinMax(2,AftZoneOutline.getPos(DownCut):z,AftZoneOutline.getPos(UpCut):z)
  endif
endfor

VB = OffAssBulb.calcDisplacement(Draft).at(0).castTo(FDouble)

A_BTsecGrp.setSurfaces([GeoOut])
A_BTsecGrp.setPositions([pUp:x+precision_tolerance])

if (A_BTsecGrp.getSections().at(0).getMin(2) < pUp:z+precision_tolerance)
  A_BTsec.setPointList(A_BTsecGrp.getSections().at(0).getData())
else
  A_BTsec.setPointList(A_BTsecGrp.getSections().at(1).getData())
endif
A_BTsec.setDegree(1)
```

```
BB = A_BTsec.getMax(1)

CloseSec.setStartPos(pUp)
CloseSec.setEndPos(pDown)
CloseSec.setParametrization("unit speed")
WholeSec.setCurveList([CloseSec,A_BTsec])
WholeSec.setAutomaticOrientation(true)

A_BT = abs(2*WholeSec.getArea(2,0))

pA_BT.setVector(A_BTsec.getPos(A_BTsec.getMax(1,true)))


//-------------------------------------------------------------------------------------------------//
// Execution of feature: transformations

LineDshiftLB.setPoint-
List([[pUp:x,0,0],[pUp:x+LB/6,0,0],[pUp:x+LB/2,0,0],[pUp:x+LB+precision_tolerance,0,N
ewLB-LB]])
DshiftLB.setDeltaX(true)
DshiftLB.setDeltaCurve(LineDshiftLB)
LineDshiftLBZBaux.setPointList([pDown,[pDown:x,1,pDown:z],[pUp:x,1,pUp:z],pUp])
LineDshiftLBZBaux.setDegree(1)
DshiftLBaux.setDeltaX(true)
DshiftLBaux.setDeltaCurve(LineDshiftLBZBaux)
DshiftLBaux.setAbscissaZ(true)
DshiftLBaux.setOrdinateY(true)
DshiftLBproduct.setFunctions([DshiftLB,DshiftLBaux])

LineDshiftZB.setPoint-
List([[pUp:x,0,0],[pUp:x+LB/6,0,0],[pUp:x+LB/2,0,0],[pUp:x+LB+precision_tolerance,0,N
ewZB-ZB]])
DshiftZB.setDeltaZ(true)
DshiftZB.setDeltaCurve(LineDshiftZB)
DshiftZBaux.setDeltaZ(true)
DshiftZBaux.setDeltaCurve(LineDshiftLBZBaux)
DshiftZBaux.setAbscissaZ(true)
DshiftZBaux.setOrdinateY(true)
DshiftZBproduct.setFunctions([DshiftZB,DshiftZBaux])

Dsum.setFunctions([DshiftLBproduct,DshiftZBproduct])

GeoOut.setImageTransformation(Dsum)

// Begin - Updating values for bulb profile change
bsplinecurve CL(GeoOut.getEdge(0))
FObjectList points(CL.getPointList())
foreach (FVector3 pt : points)
  pt.setY(0)
endfor
```

```
BulbProf.setCurve(CL)
LB = pB.getX()-pUp.getX()
ZB = pB.getZ()-CL.getMin(2)
OffBulb.setOffsets(SecBulb.getSections())
OffBulb.cutMinMax(2,ZoneOutline.getMin(2),ZoneOutline.getMax(2))
FObjectList offsets(OffBulb.getObjects(FOffset))
foreach (FOffset off : offsets)
  off.setShowOrientation(false)
  if (off.at(0):x < pUp:x+precision_tolerance)
    double UpCut(AftZoneOutline.ft(0,off.at(0):x,0,pBaftPos))
    double DownCut(AftZoneOutline.ft(0,off.at(0):x,pBaftPos,1))
    off.cutMinMax(2,AftZoneOutline.getPos(DownCut):z,AftZoneOutline.getPos(UpCut):z)
  endif
endfor
VB = OffAssBulb.calcDisplacement(Draft).at(0).castTo(FDouble)
// End - Updating values for bulb profile change

// Goal-seek for desired volume
if (abs(VB-NewBulbVol)/NewBulbVol*100 > VolMarg)
  pAux1.setVector([0,0,0])
  pAux2.setVector([1,1,0])
  cAux.setStartPos(pAux1)
  cAux.setEndPos(pAux2)

  //if pCenter was not given (default is at origin=[0,0,0]), we set a new center of inflation
  if (and([abs(pCenter:x)<0+precision_tolerance,abs(pCenter:y)<0+precision_tolerance,abs(pCenter:z)<0+precision_tolerance]))
    pA_BTproj.setVector([pA_BT:x,0,pA_BT:z]).detach() //breaking dependency to avoid infinite recurrence error
  else
    pA_BTproj.setVector([pCenter:x,0,pCenter:z]).detach() //breaking dependency to avoid infinite recurrence error
  endif

  Cengine.setDefinition(SmoothBspline)
  Cengine.setBaseCurve("|bmeta")
  MetaSurf.setUResolution(50)
  MetaSurf.setVResolution(50)
  MetaSurf.setRepresentation("cubic")
  MetaSurf.setNumberOfPointsForNURBSSurfaceDirection(50)
  MetaSurf.setTransparent(true)
  SurfDelta.setDeltaY(true)
  Dsum.add(SurfDelta)

  if (VB-NewBulbVol<0)
    while (AND([VB < NewBulbVol, i<imax]))
      yhigh+=0.5
      i+=1
      Cengine.setValues([pA_BTproj,yhigh,ZoneOutline,[cAux,1]])
      MetaSurf.setEngineAtBegin(Cengine)
```

```
    SurfDelta.setDeltaSurface(MetaSurf)
    // Begin - Updating values for bulb volume change
    GeoOut.setImageTransformation(Dsum)
    OffBulb.setOffsets(SecBulb.getSections())
    OffBulb.cutMinMax(2,ZoneOutline.getMin(2),ZoneOutline.getMax(2))
    FObjectList offsets(OffBulb.getObjects(FOffset))
    foreach (FOffset off : offsets)
      off.setShowOrientation(false)
      if (off.at(0):x < pUp:x+precision_tolerance)
        double UpCut(AftZoneOutline.ft(0,off.at(0):x,0,pBaftPos))
        double DownCut(AftZoneOutline.ft(0,off.at(0):x,pBaftPos,1))
        off.cutMinMax(2,AftZoneOutline.getPos(DownCut):z,AftZoneOutline.getPos(UpCut):z)
      endif
    endfor
    VB = OffAssBulb.calcDisplacement(Draft).at(0).castTo(FDouble)
    // End - Updating values for bulb volume change
  endwhile
  ylow = yhigh-0.5

else
  while (AND([VB > NewBulbVol, i<imax]))
   yhigh-=0.5
   i+=1
   Cengine.setValues([pA_BTproj,yhigh,ZoneOutline,[cAux,1]])
   MetaSurf.setEngineAtBegin(Cengine)
   SurfDelta.setDeltaSurface(MetaSurf)
   // Begin - Updating values for bulb volume change
    GeoOut.setImageTransformation(Dsum)
    OffBulb.setOffsets(SecBulb.getSections())
    OffBulb.cutMinMax(2,ZoneOutline.getMin(2),ZoneOutline.getMax(2))
    FObjectList offsets(OffBulb.getObjects(FOffset))
    foreach (FOffset off : offsets)
      off.setShowOrientation(false)
      if (off.at(0):x < pUp:x+precision_tolerance)
        double UpCut(AftZoneOutline.ft(0,off.at(0):x,0,pBaftPos))
        double DownCut(AftZoneOutline.ft(0,off.at(0):x,pBaftPos,1))
        off.cutMinMax(2,AftZoneOutline.getPos(DownCut):z,AftZoneOutline.getPos(UpCut):z)
      endif
    endfor
    VB = OffAssBulb.calcDisplacement(Draft).at(0).castTo(FDouble)
    // End - Updating values for bulb volume change
  endwhile
  ylow = yhigh+0.5

endif

while (AND([abs(VB-NewBulbVol)/NewBulbVol*100 > VolMarg, j<jmax]))
  ymid = (yhigh+ylow)/2
  Cengine.setValues([pA_BTproj,ymid,ZoneOutline,[cAux,1]])
  MetaSurf.setEngineAtBegin(Cengine)
```

```
  SurfDelta.setDeltaSurface(MetaSurf)
  // Begin - Updating values for bulb volume change
  GeoOut.setImageTransformation(Dsum)
  OffBulb.setOffsets(SecBulb.getSections())
  OffBulb.cutMinMax(2,ZoneOutline.getMin(2),ZoneOutline.getMax(2))
  FObjectList offsets(OffBulb.getObjects(FOffset))
  foreach (FOffset off : offsets)
    off.setShowOrientation(false)
    if (off.at(0):x < pUp:x+precision_tolerance)
      double UpCut(AftZoneOutline.ft(0,off.at(0):x,0,pBaftPos))
      double DownCut(AftZoneOutline.ft(0,off.at(0):x,pBaftPos,1))
      off.cutMinMax(2,AftZoneOutline.getPos(DownCut):z,AftZoneOutline.getPos(UpCut):z)
    endif
  endfor
  VB = OffAssBulb.calcDisplacement(Draft).at(0).castTo(FDouble)
  // End - Updating values for bulb volume change

  if (VB-NewBulbVol < 0)
    ylow = ymid
  else
    yhigh = ymid
  endif

  j+=1
 endwhile

 BB = A_BTsec.getMax(1)
 A_BT = abs(2*WholeSec.getArea(2,0))
endif
```

Bulb Transformation (Target Section Area) Feature:

```
// Sketched parametrics: bulb transformation - target section area (bulb length, bulb nose height, inflation based on target section area at FP)
// By Martin Gundelach, October 2016
// Objective: after importing a hull trimesh, easily transform the bulb.


//-------------------------------------------------------------------------------------------------------//
// Creation of objects used in feature

// Output trimesh geometry
imageTriMesh GeoOut()

// Forward Perpendicular and hull cut
line CutWL()
curveintersectionpoint pIntFP()
```

point FPpoint()
line FP()
curveintersectionpoint pIntFPdown()
imagecurve CutCL()

// Finding bulb profile
point pUp()
point pDown()
double precision_tolerance(5E-3)
line CutBulb()
curveintersectionpoint pIntBulbDown()
imagecurve BulbProf()

// Finding furthest point of bulb and some bulb parameters
point pB()
double LB()
double ZB()
double HB()
double BB()

// Creating Aft Zone of Influence of bulb for Volume calculation
point pAftZoneUp()
point pAftZoneDown()
point pAftZoneCenter()
bsplinecurve AftZoneOutline()
polycurve ZoneOutline()
sectionGroup SecBulb()
offsetGroup OffBulb()
offsetGroupAssembly OffAssBulb()
point pBaft()
double pBaftPos()
double VB()

// Finding section at Bulb aftmost point, plus some parameters
sectionGroup A_BTsecGrp()
bsplinecurve A_BTsec()
line CloseSec()
polycurve WholeSec()
double A_BT()
point pA_BT()

// Bulb length transformation
bsplinecurve LineDshiftLB()
bsplinecurve LineDshiftLBZBaux()
deltashift DshiftLB()
deltashift DshiftLBaux()
deltaproduct DshiftLBproduct()

// Bulb nose height transformation
bsplinecurve LineDshiftZB()

```
deltashift DshiftZB()
deltashift DshiftZBaux()
deltaproduct DshiftZBproduct()

// Uniting LB and ZB transformations
deltasum Dsum()

// Auxiliary curve and points for curve engine/metasurface creation
point pAux1()
point pAux2()
line cAux()

// Metasurface and curve engine for volume transformation
curveEngine Cengine()
metasurface MetaSurf()
surfacedeltashift SurfDelta()
point pA_BTproj()

// Bisection method parameters for finding target volume
unsigned i(0) //counter for finding yhigh and ylow
unsigned j(0) //counter for finding ymid, in case yhigh or ylow does not provide close enough
result
unsigned imax(10) //max iterations for i
unsigned jmax(10) //max iterations for j
double yhigh(0)
double ylow(0)
double ymid(0)


//-------------------------------------------------------------------------------------------------//
// Execution of feature: setting up and finding bulb properties

GeoOut.setSource(GeoIn)
GeoOut.setShowTriangles(false)

// Getting the CenterLine (CL) curve from the input geometry
bsplinecurve CL(GeoOut.getEdge(0))
FObjectList points(CL.getPointList())
foreach (FVector3 pt : points)
  pt.setY(0)
endfor

CutWL.setStartPos([CL.getMax(0),0,CL.getMin(2)+Draft])
CutWL.setEndPos([CL.getMin(0),0,CL.getMin(2)+Draft])
pIntFP.setCurveA(CutWL)
pIntFP.setCurveB(CL)

FPpoint.setVector(pIntFP)
FP.setStartPos(FPpoint-[0,0,Draft])
FP.setEndPos(FPpoint)
```

```
pIntFPdown.setCurveA(FP)
pIntFPdown.setCurveB(CL)

CutCL.setCurve(CL)
CutCL.setDomain([pIntFP.getParameterOnCurveB(),pIntFPdown.getParameterOn-
CurveB()])

if (CutCL.getMin(0) < (FPpoint:x-precision_tolerance))
  pUp.setVector(CutCL.getpos(CutCL.getMin(0,true)))
  CutBulb.setStartPos(pUp-[0,0,Draft])
  CutBulb.setEndPos(pUp)
  pIntBulbDown.setCurveA(CutBulb)
  pIntBulbDown.setCurveB(CL)
  pDown.setVector(pIntBulbDown)
else
  pUp.setVector(FPpoint)
  pDown.setVector(pIntFPdown)
endif

HB = pUp.getZ()-CL.getMin(2)

BulbProf.setCurve(CL)
BulbProf.set-
Do-
main([CL.getParameterShortestDistanceSquared(pUp),CL.getParameterShortestDistanceSq
uared(pDown)])
pB.setVector(BulbProf.getPos(BulbProf.getMax(0,true)))

LB = pB.getX()-pUp.getX()
ZB = pB.getZ()-CL.getMin(2)

pAftZoneCenter.setVector([pUp:x-Slider*LB*ZCmultiplier,0,(pUp:z+pDown:z)/2])
pAftZoneUp.setVector([pUp:x-Slider*ZUmultiplier*BulbProf.getTanVec(0).getNormal-
ized():x,0,pUp:z-Slider*ZUmultiplier*BulbProf.getTanVec(0).getNormalized():z])
pAftZoneDown.setVector([pDown:x+Slider*ZDmultiplier*BulbProf.getTanVec(1).getNor-
malized():x,0,pDown:z+Slider*ZDmultiplier*BulbProf.getTanVec(1).getNormalized():z])
AftZoneOutline.setPointList([pUp,pAftZoneUp,pAftZoneCenter,pAftZoneDown,pDown])
ZoneOutline.setCurveList([AftZoneOutline,BulbProf])
ZoneOutline.setAutomaticOrientation(true)
ZoneOutline.setParametrization("unit speed")

SecBulb.setSurfaces([GeoOut])
SecBulb.setPositions([AftZoneOutline.getMin(0)+0.001,      BulbProf.getMax(0)-0.001      :
round((Slider+1)*Noffsets)])
OffBulb.setOffsets(SecBulb.getSections())
OffBulb.cutMinMax(2,ZoneOutline.getMin(2),ZoneOutline.getMax(2))
OffAssBulb.setGroups([OffBulb])

pBaftPos = AftZoneOutline.getMin(0,true)
```

```
pBaft.setVector(AftZoneOutline.getPos(pBaftPos))
FObjectList offsets(OffBulb.getObjects(FOffset))
foreach (FOffset off : offsets)
  off.setShowOrientation(false)
  if (off.at(0):x < pUp:x+precision_tolerance)
    double UpCut(AftZoneOutline.ft(0,off.at(0):x,0,pBaftPos))
    double DownCut(AftZoneOutline.ft(0,off.at(0):x,pBaftPos,1))
    off.cutMinMax(2,AftZoneOutline.getPos(DownCut):z,AftZoneOutline.getPos(UpCut):z)
  endif
endfor

VB = OffAssBulb.calcDisplacement(Draft).at(0).castTo(FDouble)

A_BTsecGrp.setSurfaces([GeoOut])
A_BTsecGrp.setPositions([pUp:x+precision_tolerance])

if (A_BTsecGrp.getSections().at(0).getMin(2) < pUp:z+precision_tolerance)
  A_BTsec.setPointList(A_BTsecGrp.getSections().at(0).getData())
else
  A_BTsec.setPointList(A_BTsecGrp.getSections().at(1).getData())
endif
A_BTsec.setDegree(1)

BB = A_BTsec.getMax(1)

CloseSec.setStartPos(pUp)
CloseSec.setEndPos(pDown)
CloseSec.setParametrization("unit speed")
WholeSec.setCurveList([CloseSec,A_BTsec])
WholeSec.setAutomaticOrientation(true)

A_BT = abs(2*WholeSec.getArea(2,0))

pA_BT.setVector(A_BTsec.getPos(A_BTsec.getMax(1,true)))


//-------------------------------------------------------------------------------------------------//
// Execution of feature: transformations

LineDshiftLB.setPoint-
List([[pUp:x,0,0],[pUp:x+LB/6,0,0],[pUp:x+LB/2,0,0],[pUp:x+LB+precision_tolerance,0,N
ewLB-LB]])
DshiftLB.setDeltaX(true)
DshiftLB.setDeltaCurve(LineDshiftLB)
LineDshiftLBZBaux.setPointList([pDown,[pDown:x,1,pDown:z],[pUp:x,1,pUp:z],pUp])
LineDshiftLBZBaux.setDegree(1)
DshiftLBaux.setDeltaX(true)
DshiftLBaux.setDeltaCurve(LineDshiftLBZBaux)
DshiftLBaux.setAbscissaZ(true)
DshiftLBaux.setOrdinateY(true)
```

```
DshiftLBproduct.setFunctions([DshiftLB,DshiftLBaux])

LineDshiftZB.setPoint-
List([[pUp:x,0,0],[pUp:x+LB/6,0,0],[pUp:x+LB/2,0,0],[pUp:x+LB+precision_tolerance,0,N
ewZB-ZB]])
DshiftZB.setDeltaZ(true)
DshiftZB.setDeltaCurve(LineDshiftZB)
DshiftZBaux.setDeltaZ(true)
DshiftZBaux.setDeltaCurve(LineDshiftLBZBaux)
DshiftZBaux.setAbscissaZ(true)
DshiftZBaux.setOrdinateY(true)
DshiftZBproduct.setFunctions([DshiftZB,DshiftZBaux])

Dsum.setFunctions([DshiftLBproduct,DshiftZBproduct])

GeoOut.setImageTransformation(Dsum)

// Begin - Updating values for bulb profile change
bsplinecurve CL(GeoOut.getEdge(0))
FObjectList points(CL.getPointList())
foreach (FVector3 pt : points)
  pt.setY(0)
endfor
BulbProf.setCurve(CL)
LB = pB.getX()-pUp.getX()
ZB = pB.getZ()-CL.getMin(2)
OffBulb.setOffsets(SecBulb.getSections())
OffBulb.cutMinMax(2,ZoneOutline.getMin(2),ZoneOutline.getMax(2))
FObjectList offsets(OffBulb.getObjects(FOffset))
foreach (FOffset off : offsets)
  off.setShowOrientation(false)
  if (off.at(0):x < pUp:x+precision_tolerance)
    double UpCut(AftZoneOutline.ft(0,off.at(0):x,0,pBaftPos))
    double DownCut(AftZoneOutline.ft(0,off.at(0):x,pBaftPos,1))
    off.cutMinMax(2,AftZoneOutline.getPos(DownCut):z,AftZoneOutline.getPos(UpCut):z)
  endif
endfor
VB = OffAssBulb.calcDisplacement(Draft).at(0).castTo(FDouble)
// End - Updating values for bulb profile change

// Goal-seek for desired section area
if (abs(A_BT-NewBulbAbt)/NewBulbAbt*100 > AreaMarg)
  pAux1.setVector([0,0,0])
  pAux2.setVector([1,1,0])
  cAux.setStartPos(pAux1)
  cAux.setEndPos(pAux2)

  //if pCenter was not given (default is at origin=[0,0,0]), we set a new center of inflation
  if          (and([abs(pCenter:x)<0+precision_tolerance,abs(pCenter:y)<0+precision_toler-
ance,abs(pCenter:z)<0+precision_tolerance]))
```

```
  pA_BTproj.setVector([pA_BT:x,0,pA_BT:z]).detach() //breaking dependency to avoid infinite recurrence error
 else
  pA_BTproj.setVector([pCenter:x,0,pCenter:z]).detach() //breaking dependency to avoid infinite recurrence error
 endif

 Cengine.setDefinition(SmoothBspline)
 Cengine.setBaseCurve("|bmeta")
 MetaSurf.setUResolution(50)
 MetaSurf.setVResolution(50)
 MetaSurf.setNumberOfPointsForNURBSSurfaceDirection(50)
 MetaSurf.setTransparent(true)
 SurfDelta.setDeltaY(true)
 Dsum.add(SurfDelta)

 if (A_BT-NewBulbAbt<0)
  while (AND([A_BT < NewBulbAbt, i<imax]))
   yhigh+=0.5
   i+=1
   Cengine.setValues([pA_BTproj,yhigh,ZoneOutline,[cAux,1]])
   MetaSurf.setEngineAtBegin(Cengine)
   SurfDelta.setDeltaSurface(MetaSurf)
   // Begin - Updating values for bulb section area change
   GeoOut.setImageTransformation(Dsum)
   BB = A_BTsec.getMax(1)
   A_BT = abs(2*WholeSec.getArea(2,0))
   // End - Updating values for bulb section area change
  endwhile
  ylow = yhigh-0.5

 else
  while (AND([A_BT > NewBulbAbt, i<imax]))
   yhigh-=0.5
   i+=1
   Cengine.setValues([pA_BTproj,yhigh,ZoneOutline,[cAux,1]])
   MetaSurf.setEngineAtBegin(Cengine)
   SurfDelta.setDeltaSurface(MetaSurf)
   // Begin - Updating values for bulb section area change
   GeoOut.setImageTransformation(Dsum)
   BB = A_BTsec.getMax(1)
   A_BT = abs(2*WholeSec.getArea(2,0))
   // End - Updating values for bulb section area change
  endwhile
  ylow = yhigh+0.5

 endif

 while (AND([abs(A_BT-NewBulbAbt)/NewBulbAbt*100 > AreaMarg, j<jmax]))
  ymid = (yhigh+ylow)/2
```

```
    Cengine.setValues([pA_BTproj,ymid,ZoneOutline,[cAux,1]])
    MetaSurf.setEngineAtBegin(Cengine)
    SurfDelta.setDeltaSurface(MetaSurf)
    // Begin - Updating values for bulb section area change
    GeoOut.setImageTransformation(Dsum)
    BB = A_BTsec.getMax(1)
    A_BT = abs(2*WholeSec.getArea(2,0))
    // End - Updating values for bulb section area change
    if (VB-NewBulbAbt < 0)
      ylow = ymid
    else
      yhigh = ymid
    endif

    j+=1
  endwhile

  // updating bulb volume
  OffBulb.setOffsets(SecBulb.getSections())
  OffBulb.cutMinMax(2,ZoneOutline.getMin(2),ZoneOutline.getMax(2))
  FObjectList offsets(OffBulb.getObjects(FOffset))
  foreach (FOffset off : offsets)
    off.setShowOrientation(false)
    if (off.at(0):x < pUp:x+precision_tolerance)
      double UpCut(AftZoneOutline.ft(0,off.at(0):x,0,pBaftPos))
      double DownCut(AftZoneOutline.ft(0,off.at(0):x,pBaftPos,1))
      off.cutMinMax(2,AftZoneOutline.getPos(DownCut):z,AftZoneOutline.getPos(UpCut):z)
    endif
  endfor
  VB = OffAssBulb.calcDisplacement(Draft).at(0).castTo(FDouble)
endif
```

Aft Waterline/Diagonal Feature:

```
// Sketched Parametrics: aft waterline change
// By Martin Gundelach, October 2016
// Objective: after importing a hull mesh, easily change the waterline in the aft part of the hull.

//-------------------------------------------------------------------------------------------------------//
// Creation of objects used in feature

imageTriMesh GeoOut()

double precision_tolerance(1E-3)

sectionGroup OrigWLs()
sectionGroup OrigSecs()
```

plane Zplane()
double Zmax()
double Zmin()
double Xmin()
double Xmax()
unsigned Nwat(10)
unsigned Nsec(10)

plane WLplane()
sectionGroup WLorig()
bsplinecurve WLorigBspl()
double AuxWLdepth()
unsigned i(0)
unsigned imax(2)

line CutShip()
point pAft()
curveintersectionpoint pIntAft()

scaling XYscale()
scaling XZscale()
imagecurve projXY()
imagecurve projXZ()

bsplinecurve NewProjXY()
unsigned j(1)

deltashift DS_NewWL()
deltashift DS_OrigWL()
deltasum Dsum_WL()
imagecurve GeneratrixWL()

line AuxMeta()
curveEngine Cengine()
metasurface MetaSurf()
surfacedeltashift DeltaSurfY()

sectionGroup NewWLs()
sectionGroup NewSecs()

sectionGroup WLnew()
bsplinecurve WLnewBspl()

//----------//
// Execution of feature

GeoOut.setSource(GeoIn)
GeoOut.setShowTriangles(false)

```
// Getting the CenterLine (CL) curve from the input geometry
bsplinecurve CL(GeoOut.getEdge(0))
FObjectList points(CL.getPointList())
foreach (FVector3 pt : points)
  pt.setY(0)
endfor

CutShip.setStartPos([CL.getMin(0),0,(WLdepth-CL.getMin(2))])
CutShip.setEndPos([CL.getMax(0),0,(WLdepth-CL.getMin(2))])
pIntAft.setCurveA(CutShip)
pIntAft.setCurveB(CL)
pAft.setVector(pIntAft)

Zplane.setNormal([0,0,1])
OrigWLs.setPlane(Zplane)
Zmax = CL.getMax(2)-precision_tolerance
Zmin = CL.getMin(2)+precision_tolerance
OrigWLs.setPositions([Zmin,Zmax:Nwat])
OrigWLs.setSurfaces([GeoIn])
OrigWLs.setResolution(300)
OrigWLs.setAttachDistance(toggleDouble(true,0.01))

WLplane.setBase(pAft)
WLplane.setNormal([0,sin(DiagAngle),cos(DiagAngle)])
WLorig.setPlane(WLplane)
WLorig.setSurfaces([GeoIn])
WLorig.setPositions([0,0:1])
WLorig.setAttachDistance(toggleDouble(true,0.01))
WLorig.setResolution(300)

//I put this WHILE loop because in some cases, like for WLdepth=4.0 for KCS mesh, coincident
mesh nodes make the waterline look weird, breaking into many pieces that cannot be attached;
this little code tries to fix this bug by cutting a little higher
while (AND([WLorig.getSections().getSize() > 1 , i<imax]))
  i+=1
  AuxWLdepth = WLdepth+precision_tolerance*i
  CutShip.setStartPos([CL.getMin(0),0,(AuxWLdepth-CL.getMin(2))])
  CutShip.setEndPos([CL.getMax(0),0,(AuxWLdepth-CL.getMin(2))])
  // begin - updating wl
  pIntAft.setCurveA(CutShip)
  pAft.setVector(pIntAft)
  WLplane.setBase(pAft)
  WLorig.setPlane(WLplane)
  // end - updating wl
endwhile

Xmin = max([pAft:x , Xstart])
WLorigBspl.setPointList(WLorig.getSections().at(0).cutMinMax(0,Xmin,Xend).getData())
Xmax = min([WLorigBspl.getMax(0) , Xend])
```

```
OrigSecs.setSurfaces([GeoIn])
OrigSecs.setPositions([Xmin , Xmax : Nsec])

XYscale.setFactorZ(0)
XZscale.setFactorY(0)

projXY.setCurve(WLorigBspl)
projXY.setImageTransformation(XYscale)
projXY.setParametrization("unit speed")
projXZ.setCurve(WLorigBspl)
projXZ.setImageTransformation(XZscale)
projXZ.setParametrization("unit speed")

NewProjXY.setPointList([projXY:start , projXY:start+[projXY.getLength()/(2*(NpCon-
trol+1))*projXY.getTanVec(0):x , projXY.getLength()/(2*(NpControl+1))*projXY.getTan-
Vec(0):y , 0]])
beginPersistentSection
loop (NpControl)
  parameter DeltaY_00(MaxAmp*(1-((j-(NpControl+1)/2)/((NpControl+1)/2))^2)^3) //replace
these parameter values when using this feature, by executing instead of creating the feature; or
just let the equation as is: f(x)=(1-x^2)^3 ; x E [-1,1]
  point p_00([projXY.getPos(j/(NpControl+1)):x,projXY.getPos(j/(NpControl+1)):y+Del-
taY_00,0])
  NewProjXY.getPointList().add(p_00)
  p_00.setVisible(false)
  j+=1
endloop
endPersistentSection
NewProjXY.getPointList().add(projXY:end-[projXY.getLength()/(2*(NpControl+1))*pro-
jXY.getTanVec(1):x , projXY.getLength()/(2*(NpControl+1))*projXY.getTanVec(1):y , 0])
NewProjXY.getPointList().add(projXY:end)
NewProjXY.setParametrization("unit speed")

DS_NewWL.setDeltaY(true)
DS_NewWL.setDeltaCurve(NewProjXY)
DS_NewWL.setOrdinateY(true)
DS_OrigWL.setDeltaY(true)
DS_OrigWL.setDeltaFactor(-1)
DS_OrigWL.setDeltaCurve(projXY)
DS_OrigWL.setOrdinateY(true)
Dsum_WL.setFunctions([DS_OrigWL,DS_NewWL])
GeneratrixWL.setCurve(projXZ)
if (projXZ.getPos(0):x > projXZ.getPos(1):x)
  GeneratrixWL.setDomain([1,0])
endif
GeneratrixWL.setImageTransformation(Dsum_WL)
GeneratrixWL.setParametrization("unit speed")

AuxMeta.setStartPos([0,0,0])
```

```
AuxMeta.setEndPos([1,1,0])
AuxMeta.setParametrization("unit speed")
Cengine.setDefinition(VerticalBspline)
Cengine.setBaseCurve("|VertBspline")
Cengine.setValues([GeneratrixWL,[AuxMeta,1],VertStart,VertEnd])
MetaSurf.setEngineAtBegin(Cengine)
MetaSurf.setRepresentation("auto-cubic")
MetaSurf.setNumberOfPointsForNURBSSurfaceDirection(30)
MetaSurf.setUResolution(20)
MetaSurf.setVResolution(20)

DeltaSurfY.setDeltaY(true)
DeltaSurfY.setDeltaSurface(MetaSurf)

GeoOut.setImageTransformation(DeltaSurfY)

NewWLs.setPlane(Zplane)
NewWLs.setPositions([Zmin,Zmax:Nwat])
NewWLs.setSurfaces([GeoOut])
NewWLs.setResolution(300)
NewWLs.setAttachDistance(toggleDouble(true,0.01))
NewWLs.setColor(255,0,0)
NewSecs.setSurfaces([GeoOut])
NewSecs.setPositions([Xmin , Xmax : Nsec])
NewSecs.setColor(255,0,0)

WLnew.setPlane(WLplane)
WLnew.setSurfaces([GeoOut])
WLnew.setPositions([0,0:1])
WLnew.setAttachDistance(toggleDouble(true,0.01))
WLnew.setResolution(300)
WLnewBspl.setPointList(WLnew.getSections().at(0).cutMinMax(0,Xmin,Xend).getData())
WLnewBspl.setColor(255,0,0)
```

APPENDIX II: INPUT FILE FOR FINE/MARINE

```
************************************************************
*                                                          *
*      Expert template for resistance application          *
*                                                          *
************************************************************

********** PROJECT MANAGEMENT **********

*** MATRIX PROJECT
* 1 = YES /0 = NO
0
*
*** PATH & NAME
* Project directory
/home/u3a3727/Desktop/Thesis/KCS_SketchDeform_Rev38/
*
*** UNITS
* kt or m/s; deg or rad; m or ft
m/s deg m
*

********** MATRIX PARAMETERS **********

*
*** MATRIX PARAMETERS (IMPOSED ANGLES, DRAUGHT)
* Yaw (CAUTION: If the configuration is half body, yaw angles are not allowed)
0
* Pitch (CAUTION: if not 0, trimming motions won't be solved)
0
* Roll (CAUTION: If the configuration is half body, roll angles are not allowed)
0
* Draught (CAUTION: if not 0, sinking motions won't be solved)
0
*

********** BODY CONFIGURATION **********

*
*** PATH OF THE GEOMETRY FILE
* Complete path including file extension or directory
hull.stl
*
*** BODY CONFIGURATION
* 1 = half body/0 = entire body
1
*
*** BODY ORIENTATION FOR X AXIS
* 1 = positive X-axis/-1 = negative X-axis
1
*
*** BODY ORIENTATION FOR Y AXIS (if half body configuration)
* 1 = positive Y-axis/-1 = negative Y-axis
1
*
*** CARDAN ANGLES
* Yaw; Pitch; Roll (CAUTION : If the configuration is half body, only pitch angles are allowed)
0.0 0.0 0.0
*
*** BODY MOTIONS TO SOLVE
* Trim No=0/Yes=1; Sinkage No=0/Yes=1
1 1
*
```

```
********** FLOW DEFINITION **********

*
*** SPEED DEFINITION
* Vref (max speed if more than one speed computation)
2.2
* List of speeds
2.2
* Type of computation (independent computations=0/successive restarts=1)
0
*
*** INITIAL FREE SURFACE
* Z-coordinate
0.34177215
*
*** SCALING FACTOR
*
1.0
*
*** WATER PROPERTIES
* Name of the second fluid; Dynamic viscosity; Density; Name of the first fluid; Dynamic viscosity; Density
AIR 1.85e-05 1.2 WATER 0.00122 999.07
*
*** SHALLOW WATER
* Activate No=0/Yes=1; Depth
0 1.0
*

********** ADDITIONAL INPUTS **********

*
*** ACTUATOR DISK
* Activate No=0/Yes=1; Thrust; Thickness; Inner radius; Outer radius; Xcenter; Ycenter; Zcenter; Xdir; Ydir; Zdir;
0 0.0 1.0 0.1 1.0 0.0 0.0 0.0 0.0 0.0 0.0
* Activate tangential force No=0/Yes=1; Torque
0 0.0
* Self-update No=0/Yes=1; Frequency
0 10
*
*** DRAG BASED PROPELLER WRENCH
* Activate No=0/Yes=1; Xcenter; Ycenter; Zcenter; Xdir; Ydir; Zdir
0 0.0 0.0 0.0 0.0 0.0 0.0
*
*** ADAPTIVE GRID REFINEMENT
* No=0/Yes=1
0
*

********** MESH PARAMETERS **********

*
*** SELECT MESH DENSITY
* 0=Coarse/1=Medium/2=Fine
0
*
*** EXTRA REFINEMENT OF WAVES
* No=0/Yes=1
0
*
*** MERGE FACES WITH SAME NAME
* No=0/Yes=1
0
*
*** MERGE TANGENTIAL FACES
* No=0/Yes=1; Feature angle
0 160.0
```

```
*
*** USER DEFINED DOMAIN SIZE
* No=0/Yes=1; Nbr of LOA: Above, Before, Behind, Below, Side
0 0.5 1.0 3.0 1.5 1.5
*
*** SELECT TRIANGULATION DENSITY
* 0=Coarse/1=Medium/2=Fine
2
*
*** USER DEFINED Y+
* No=0/Yes=1; Value
1 30.0
*

********** ADVANCED PARAMETERS **********

*
*** BODY NAME
* Body name of the project
Vessel
*
*** STL PARAMETERS
* Tolerance, Feature angle (only for ASCII import)
1e-20 40
*
*** MOTIONS PARAMETERS
* Number of time steps for acceleration; Under relaxation for QS; Number of timesteps for predictions(dT2 & dT3)
500 0.05 22 22
*
*** CONTROL VARIABLES PARAMETERS
* Number of NL iter; Sub-cycling No=0/Yes=1
5 0
*
*** TIME STEP PARAMETERS
* Number of time steps; Number of time steps for successive restarts (type of computation = 1)
2000 750
*

**********  LAUNCH PARAMETERS  **********

*
*** MESH GENERATION RESSOURCES
* Number of cores
1
*

********** DOMAIN AND GEOMETRY **********

*
*** SELECT REFERENCE LENGTH
* 0 = User-defined/1 = Automatic (=Loa)
0
*
*** REFERENCE LENGTH VALUE
* Leave empty if computed from geometry file
7.27848
*
*** BODY MASS
* 0 = User-defined/1 = Automatic; Body mass
1 0
*
*** CENTER OF GRAVITY
* 0 = User-defined/1 = Automatic; xG; yG; zG
1 0 0 0
*
```

APPENDIX III: AVERAGE RESISTANCE RESULTS FOR ORIGINAL AND OPTIMUM HULLS

The following tables give the average resistance R_T over the last 10% time steps of each time series of results, as well as their standard deviation σ, for each hull variant tested, as well as the original hull's results.

Table 0-1 – Average R_T resistance results for different hull variants (part 1)

Hull ID	R_T (N)	σ (N)	Hull ID	R_T (N)	σ (N)	Hull ID	R_T (N)	σ (N)
Original	**86.8**	**0.21**	32	87.1	0.29	65	86.4	0.23
0	86.7	0.28	33	86.5	0.26	66	87.7	0.25
1	86.3	0.26	34	87.0	0.24	67	86.8	0.27
2	87.4	0.33	35	86.8	0.32	68	86.4	0.30
3	86.9	0.27	36	86.4	0.28	69	86.7	0.29
4	86.9	0.27	37	86.4	0.23	70	87.7	0.27
5	86.5	0.28	38	87.8	0.26	71	87.8	0.27
6	87.1	0.24	39	86.8	0.29	72	86.6	0.26
7	87.3	0.24	40	86.6	0.26	73	86.2	0.28
8	86.6	0.24	41	86.5	0.28	74	87.1	0.26
9	86.4	0.29	42	87.0	0.22	75	87.1	0.28
10	87.2	0.32	43	87.3	0.29	76	86.9	0.26
11	86.7	0.25	44	86.7	0.25	77	86.9	0.26
12	86.7	0.24	45	86.4	0.28	78	87.0	0.31
13	87.0	0.23	46	88.1	0.24	79	87.0	0.26
14	87.5	0.25	47	87.5	0.30	80	87.0	0.26
15	87.0	0.25	48	86.6	0.29	81	86.4	0.33
16	87.1	0.25	49	86.5	0.26	82	87.2	0.27
17	86.7	0.26	50	88.0	0.31	83	87.0	0.33
18	87.2	0.25	51	86.3	0.27	84	86.7	0.26
19	86.7	0.27	52	86.7	0.25	85	86.3	0.28
20	86.2	0.29	53	86.8	0.26	86	87.5	0.28
21	86.3	0.30	54	87.0	0.26	87	86.9	0.27
22	87.7	0.29	55	88.0	0.26	88	86.8	0.29
23	87.4	0.28	56	87.1	0.33	89	86.5	0.25
24	86.9	0.24	57	86.3	0.31	90	86.5	0.25
25	86.4	0.25	58	87.2	0.25	91	87.2	0.26
26	87.1	0.28	59	86.8	0.25	92	86.5	0.32
27	87.6	0.27	60	86.7	0.31	93	86.8	0.29
28	86.6	0.26	61	86.6	0.25	94	88.5	0.28
29	86.3	0.26	62	87.6	0.25	95	86.8	0.25
30	87.7	0.25	63	87.7	0.30	96	86.7	0.28
31	87.2	0.34	64	87.1	0.22	97	86.6	0.27

Table 0-2 – Average resistance results for different hull variants (part 2)

Hull ID	R_T (N)	σ (N)	Hull ID	R_T (N)	σ (N)	Hull ID	R_T (N)	σ (N)
98	87.4	0.30	132	86.4	0.28	166	87.1	0.26
99	86.8	0.31	133	86.3	0.24	167	87.7	0.31
100	86.5	0.28	134	88.5	0.25	168	86.7	0.28
101	86.5	0.28	135	87.4	0.32	**169**	**86.1**	**0.26**
102	88.0	0.33	136	86.7	0.19	170	86.7	0.29
103	87.0	0.28	137	86.4	0.25	171	86.7	0.28
104	86.9	0.31	138	86.7	0.29	172	86.7	0.29
105	86.4	0.28	139	87.2	0.26	173	86.7	0.28
106	86.4	0.27	140	86.7	0.28	174	87.5	0.28
107	87.7	0.31	141	86.3	0.28	175	87.5	0.26
108	86.5	0.25	142	87.7	0.26	176	87.2	0.31
109	86.3	0.29	143	88.0	0.31	177	86.5	0.23
110	87.9	0.25	144	86.8	0.23	178	87.0	0.19
111	87.6	0.28	145	86.3	0.20	179	87.0	0.30
112	86.8	0.26	146	87.5	0.28	180	86.3	0.24
113	86.2	0.29	147	86.5	0.26	181	86.3	0.28
114	87.2	0.24	148	86.7	0.30	182	87.5	0.25
115	86.6	0.25	149	86.4	0.27	183	86.8	0.23
116	86.7	0.28	150	86.8	0.26	184	86.7	0.28
117	86.6	0.28	151	87.6	0.25	185	86.6	0.30
118	87.4	0.33	152	86.8	0.28	186	87.5	0.24
119	87.4	0.28	153	86.4	0.26	187	87.3	0.25
120	86.4	0.24	154	87.4	0.35	188	86.7	0.34
121	86.3	0.26	155	86.6	0.24	189	86.2	0.20
122	87.7	0.25	156	86.6	0.32	190	87.6	0.25
123	86.9	0.28	157	86.9	0.27	191	87.1	0.34
124	86.8	0.22	158	87.7	0.33	192	86.8	0.30
125	86.9	0.27	159	87.6	0.24	193	86.5	0.27
126	87.9	0.26	160	86.4	0.29	194	87.1	0.29
127	87.1	0.24	161	86.5	0.25	195	87.4	0.30
128	87.1	0.30	162	87.9	0.30	196	86.6	0.25
129	86.7	0.23	163	86.4	0.25	197	86.3	0.24
130	87.3	0.32	164	86.8	0.25	198	87.4	0.31
131	87.1	0.30	165	86.7	0.18	199	86.9	0.32