university of groningen

faculty of science and engineering

FlexiWeight Weight Estimation

Bachelor Thesis

June 2019

Student: Philip Oetinger

Primary supervisor: J. Kosinka

Secondary supervisor: M. Florean (CADMATIC)

Abstract

A new intelligent general assembly for use within the HOLISHIP project requires a method for estimating the weight and center of gravity of a completed ship design, without needing to completely define the entire ship. A method was formulated and implemented that exploited the linearity of weighted vectors, object composition, and a proposed relation between actual ship construction and a virtual solid ship shape. The FlexiWeight Tool designed by CADMATIC is studied and extended to provide these estimations. Preliminary results indicate accurate estimations of incomplete ships; however, further testing will be conducted externally.

Contents

Chapter 1

Introduction

HOLISHIP is a large European R&D project that unites 40 different companies in a coordinated goal of designing and optimizing maritime assets considering many key performance aspects in a holistic manner (in other words, as a whole) [2]. The project encompasses many software systems and packages, and requires an integrated approach to ensure that the systems couple correctly. Each step in the ship design process (Figure 1.1) is essential for each subsequent step. One of these steps involves calculating the weight of a ship.

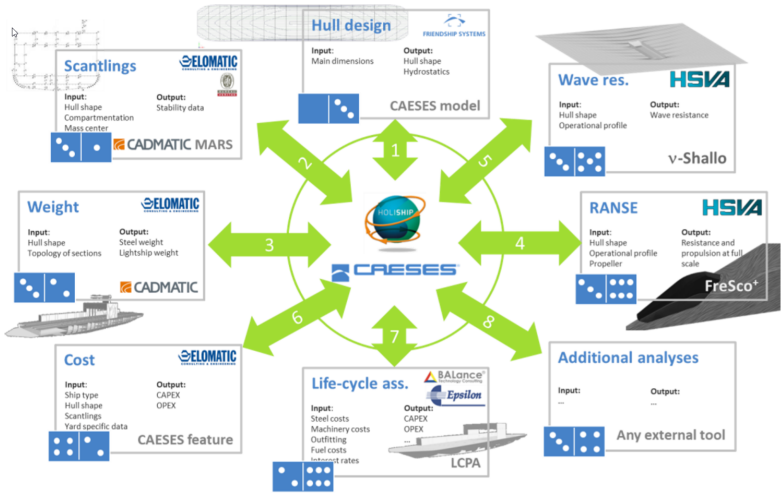

Figure 1.1: CAESES Connecting multiple systems, with the order in which a ship is designed. [2]

During the initial concept phase of designing a ship, called the general assembly (GA), the weight and center of gravity are key to ensuring that a

new ship design will be effective at sea. The weight of a ship determines how it sits in the water, and combined with center of buoyancy and center of gravity determines the three dimensional orientation in the water. This orientation effects how efficient the ship can travel through the water, and how stable it remains while maneuvering in many different oceanic conditions.

One of the options for estimating a weight of a ship during the GA would be to extend the FlexiWeight tool developed by CADMATIC. FlexiWeight is a tool that aims to assist engineers during the GA to help provide weights and centers of gravity for defined construction. The problem, however, is that defining this construction takes a long time, and can require too much fine detail to provide accurate weights and centers of gravity of each part of the ship. A solution to this problem would be an estimation that could accurately predict the weight and center of gravity (COG) of the completed ship without needing to actually define all of the construction.

Thus the aim of this thesis is to research and extend the FlexiWeight tool to estimate ships' weights and COG. I will cover required background knowledge, discuss theoretical examples, and discuss the tool. The FlexiWeight tool depends on several languages and technologies including C++, Fortran, Eagle Macro, and eControl. These technologies drive everything from the internal calculations in 3D space, construction and parsing of data structures, and creation of the user interface.

Chapter 2

Background

The development of the FlexiWeight Weight Estimation tool requires knowledge of a few mathematical concepts and existing software including weighted vectors, calculating center of gravity, a proposed relation of weights and COG to other construction through virtual blocks, and the FlexiWeight tool itself.

2.1 Center of Gravity

Center of gravity is "an imaginary point in a body of matter where, for convenience in certain calculations, the total weight of the body may be thought to be concentrated" [1]. It plays a key role in design, and is useful in prediction of the performance of vessels at sea. Knowing the ships dry weight and COG (empty, not loaded), and those of a fully loaded ship, will indicate to the architect whether the ship could have potentially unsafe or unwanted side effects.

2.1.1 Basic Calculation

A center of gravity can be in many dimensions, where each dimension is considered independently from the others. To better grasp what center of gravity is, and how to calculate it, we take a look at a one dimensional model pictured in Figure 2.1.

This model pictures three weights 15, 2, and 15 at arms A, B, and C. The center of gravity of this model is the average of these three moments. Thus $(15A + 2B + 15C)/3$ is the location of the center of gravity for this model. Basically, finding the center of gravity can be expressed as finding the average of all the weighted vectors of each part, if the weighted vectors represent a weight and arm (moment). This is very important later, when

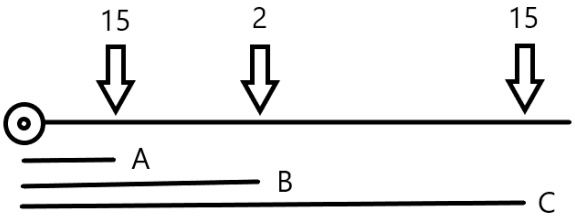

Figure 2.1: A one dimensional model for center of gravity.

we look at many objects composing larger objects in three dimensions.

2.1.2 Weighted Vectors in Center of Gravity

The moment is a magnitude of mass (weight) and a length from a specific datum. This is equivalent to stating that a moment is a weighted vector when all the vectors are projected onto a 1 dimensional axis that is being calculated. Thus when describing higher dimensional objects, it can be quicker to define them in terms of vectors; whereby, the vectors are broken down into their basic 3 dimensional components. So a 2D object can be measured using two different reference datums (example x and y), and a 3D object will have 3 (example length, breadth, height in ships).

This means that defining a primitive can boil down to the vectors that describe it. For example, a rectangle can be measured in its two dimensions by averaging the vectors in each dimension independently of the other. So for a rectangle defined by the vectors A, B, C, D, we simply average the sum of each. Thus, the center of gravity vector would be $[(A_x + B_x + C_x + D_x)/4, (A_y + B_y + C_y + D_y)/4]$.

2.2 FlexiWeight Tool

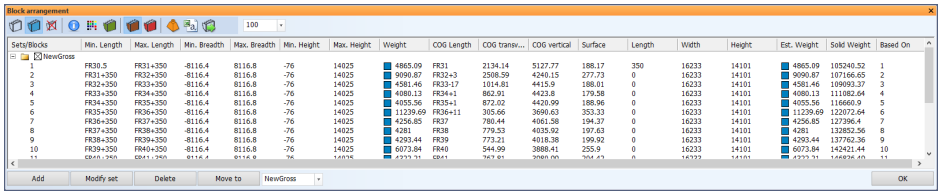

Figure 2.2: The FlexiWeight Tool UI

The FlexiWeight Tool (Figure 2.2) is built into Cadamtic's Hull application. It calculates the weight and COG of a defined construction (Figure 2.3) from Hull. It also exports the data in CSV for importing calculations into Excel or other applications.

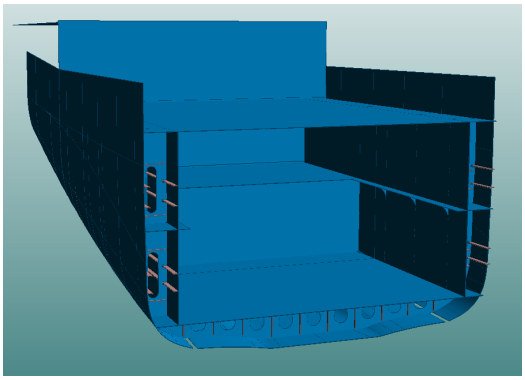

Figure 2.3: An example of an actual construction defined within base blocks from Hull. This construction is what will be estimated.

The FlexiWeight tool takes a ship with defined Base Blocks that contain the parts of the ship. Base Blocks allow the separation of parts into different sections, types of construction, and can allow calculations of specific regions of the ship. On top of these Base Blocks, there are user-defined virtual blocks (Figure 2.4) that can be defined for the specific calculation zones. For example, the virtual block(s) could be one large block covering the entire ship, $i * j * k$ number of blocks of size $m * n * o$, or slices of a certain breadth and height that cover exactly one frame in length. In our test cases, we use slices that are visible in Figure 2.4. The ship is clearly within the bounds of the blocks, as visible by the wire frame contours that mark the edges of the ship.

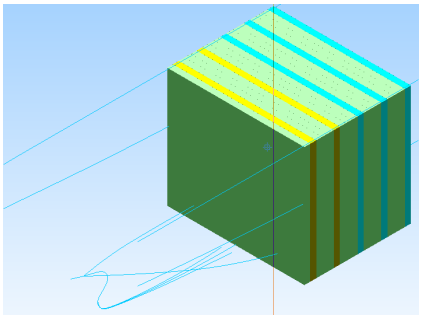

Figure 2.4: FlexiWeight Blocks defining area to be calculated. Parts are selected that are within these blocks, and trimmed if needed. The color is a user-defined range of weights, for easier recognition of potential areas of interest. Special curved wires give outlines of the ship shape, which can indicate any sections of the ship that is not contained within the blocks.

2.2.1 FlexiWeight Technologies

The user interface of the FlexiWeight tool utilizes two technologies that work hand in hand. Eagle Macro drives the main UI construction and event handling. Being an enriched Microsoft Foundation Library (MFC) library, it constructs the window, buttons, main panels, and prepares the spreadsheet for eControl. MFC is an Object Oriented Programming library for developing applications and user interfaces for Windows. eControl handles the data behind the scenes, and allows manipulation of the tree to add, remove, and change individual entries or blocks. Both are written in C++ for handling different parts of the user interface.

Once the user selects the blocks and base blocks that they wish to incorporate in the weight calculations, the FlexiWeight FORTRAN subroutines and C++ take over. These subroutines handle the fetching of parts from the database servers, and conduct the actual calculations.

Each part is sequentially fetched, checked for position and which block it lies within, and is pruned if any part(s) of it lie outside the boundary. Once pruned, they are added to the constructed virtual block. Once all parts are checked and pruned, the constructed virtual block is then queried for weight and the center of gravity (length, breadth, height).

Afterwards, the calculations are parsed by Eagle and are imported into eControl. Any changes may be made by hand, and the data can be saved for later, or exported into a CSV file for exporting to Excel.

Chapter 3

Method

Exploiting the linearity of the system of weighted vectors and object composition, a method of estimation was formulated and checked before implementation began. The first part of this dealt with how objects were composed of smaller objects, and could be made into a single vector once computed, and secondly the objects themselves being able to be related to each other via affine transformations between reference shapes. This method of first constructing a shape, and then relating it to different shapes, is used to estimate incomplete sections of the ship.

3.1 Object Composition

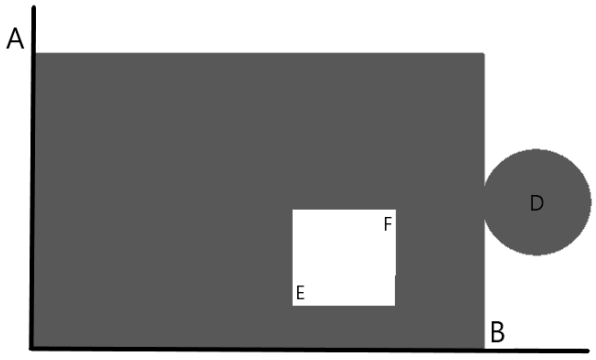

Figure 3.1: An object defined by smaller primitive objects.

The weight of a non-symmetrical object can be measured directly using physical tests, or found mathematically by breaking it down into smaller components. Take for example Figure 3.1. This object T (gray) of mass M

is composed of a rectangle (bounded by vectors A and B) and a disc centered on the point D with radius r. A smaller square defined by the vectors E and F is removed from the previously defined rectangle. We will use a reference datum set to the left bottom corner of the rectangle for measuring the arms in the x and y direction.

Since we can represent these components with vectors, calculating their center of mass is very straightforward. We simply find the centers of mass of each smaller component, add or subtract each, and divide by the total number of components. The center of mass of a disc (assuming uniform density as usual), lies at its center. Thus we have $[D_x, D_y] * M = D$. Next we find the *negative* mass of the area missing, which allows us to ignore the absent square later. This gives us $-1 * [(F_x - E_x)/2, (F_y - E_y)/2] * M = S$. Finally, we find the mass of the rectangle $[B/2, A/2] * M = R$. Now the entire object is equal to the average of these pieces $R + D - S = T$.

This method of defining an object by its parts is exploited heavily to produce accurate measurements of the centers of gravity of ships. Everything can be broken down into smaller primitives. However, this herein lies the cause of the problem, as we cannot derive the COG without having access to every part of the ship.

3.2 Construction Relation

Using what we know about object COG of constant densities and composition, we look into exploiting the linearity of COG to allow us to use existing construction to estimate unfinished construction. For this, a simple 1 dimensional model was created, pictured in Figure 3.2.

Take this simple model into consideration. A fully defined object A, has a COG of 5 units right of the datum. This object has an overall shape that we can create a virtual solid shape, of similar shape to A, that will refer to the actual construction of A. This virtual shape will be labeled AS (for A Solid). We also have a different object B that is not fully defined; however, it will be of the same composition of A but slightly larger. We wish to know the real COG of B, but to do so we would have to fully complete B, or use A to estimate B when complete.

This is where we relate the objects to find an estimate of an incomplete object. Since there are two virtual solid shapes that represent the finished objects, we relate them by finding the difference in moment between them. In this case, we know that BS is 120% of AS. So when B is fully complete, we expect that it will have a moment that is approximately 20% further to the right than that of A. Therefore, since A has an actual moment of 5,

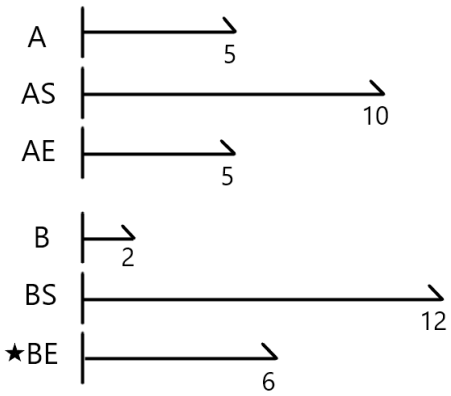

Figure 3.2: An actual COG "A", with a virtual solid reference COG "AS", giving the estimated COG AE. B Refers to A's solid, giving a better estimate of final COG

we estimate B to have a moment of $5 * 1.2 = 6$. This provides accurate estimations, as long as the relatable shapes are similar in composition.

In Figure 3.3 we can see examples of the actual construction (right) and virtual solids (left). Actual construction takes a long time to fully define, so using construction relation could allow a few highly detailed sections of the ship to allow accurate estimations of the rest of the ship. From the simple example in Figure 3.2, the actual construction (right) would be A or B, and the virtual solid blocks would be the AS or BS.

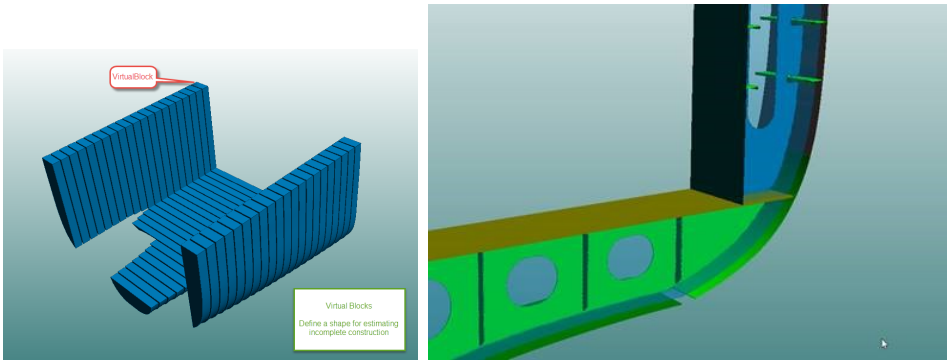

Figure 3.3: Virtual solid steel blocks define the shape of estimated construction, they are simple to define and can be quickly manipulated. These solids will define the shape of the ship, and will be used to reference the actual construction for calculations of weight and COG within a specific block's actual construction.

Chapter 4

Implementation

Implementing the proposed changes took place in 3 main phases. Phase 1 explored the relations between virtual blocks and actual construction by adding a separate calculation for the defined virtual solid block, and relating that to the actual construction. This first phase involved learning the existing tool written in C++, FORTRAN, Eagle, and eControl. This part took most of the time of Phase 1, followed by implementation of a basic weight estimation. This first phase was the most important because it provided the knowledge needed for extending the tool, and gave a good indication whether the proposed relations would be accurate enough. After the weight estimations were finished, the tool would be further extended to include the COG estimations. Finally, the user interface would be re-organized to satisfy the requirements of CADMATIC.

CADMATIC also made requests to keep the systems that have been created mostly as is and functionally intact. This meant that the implementation must adhere to CADMATIC's coding conventions and solutions made that incorporate their established systems. This meant that entirely new code is not allowed to be created from scratch; rather, the code-base must be studied and extended. Only minor issues within may be fixed as long as it does not change functionality of the systems.

4.1 Phase 1: Weight Estimation

During the first phase of implementation, time was spent learning the established systems of CADMATIC. As there are tens of thousands of lines of code, this phase took time. During this familiarization phase, a quick test was devised utilizing hard-coded discriminating conditions. Basically, virtual weights were identified and removed from the actual construction during cal-

culations, and summed separately.

Before conducting any major changes, some simple tests were required to test the theory for any major theoretical flaws. Therefore, the user interface was re-organized to add columns that would display simple weight estimations that can be gathered from the existing subroutines. Simple weight calculation changes were injected into the FlexiWeight C++ routine and allowed testing of the separation of actual construction from virtual solids. This data scrape also enabled the exploration of data export to Eagle and how each of the systems handled data exchange. While these export functions were explored and expanded, the UI code was updated, and more familiarization with the complex virtual construction within FlexiWeight was performed.

Once the conceptual UI changes and initial findings were demonstrated, and approval given, a more thorough and precise calculation method was planned for the expansion required for calculation of COG in 3D space.

4.2 Phase 2: Center of Gravity

Since weight calculations were only scalar numbers, and needed only to be duplicated from existing calculations, an entirely new method was needed to be created in order to find a three dimensional center of gravity within the virtual block construction. Thus the weight scraping method was replaced with a new virtual construction that assembles a virtual solid that can be then queried for a three dimensional COG as well as the weight. As discussed previously, an objects COG can be calculated recursively by computing each of the parts that comprise it. Calculating the weight and center of gravity of the actual and solid objects to create the ratios followed a short, but complex, algorithm. The specifics of the algorithm are property of CADMATIC, however the general algorithm follows:

```
for each block do:
  for each part do:
    if part lies (partially) within block
      trim part to block boundaries
      calculate part COG and weight
      add weighted vectors to actual or virtual object
  calculate actual and virtual object weight and COG
export block data to Eagle
```

While specific code example are not allowed to be used, in general the algorithm loops through the parts and takes each part that lies (partially) within the block. These parts are first trimmed to lie within the block, and

then their weight and COG is calculated and added to an "object" that just contains a set of weighted vectors. Once the loop ends, the weights and COGs are calculated from these sets of weighted vectors. This method allows any shape to be defined by the architects, and can be used to test any base block(s) that need to be estimated.

During implementation of the COG estimation, a problem was discovered with finding center of gravity from port to starboard (Figure 4.1). This was caused by the perfect symmetry of the virtual solid shapes. The tool used a datum of 0mm for each dimension, which previously had no impact or potential cause of concern. However, with the new changes to estimation, this gave a division by zero error when calculating ratios of actual over virtual (Figure 4.1 left). A solution was found by shifting the datum to the leftmost side of the ship, find the COG, and then shift the datum back. This keeps the system the same as before, but allows estimation without division by zero. This was possible due to the linear nature of affine transformations. If every vector is shifted by n units, they can be then shifted n units back without changing the object.

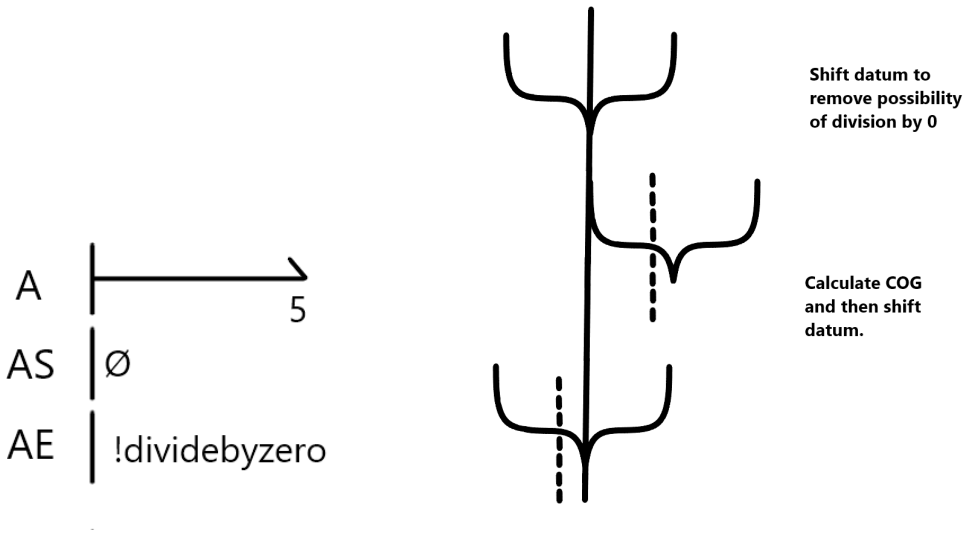

Figure 4.1: Division by zero caused errors in the ratio calculation, so a temporary shift of the datum fix was added

Eagle will receive the data, and will parse the data to be used within the UI and prepared for potential exportation to CSV or other tools.

4.3 Phase 3: User Interface

In keeping with the wishes of CADMATIC, the UI had to be re-organized, but still retain the same functionality. To accomplish this, the underlying data remains untouched, however the actual rendering and construction of the tree data structure is modified to allow the data to be rendered in a more convenient fashion. Columns are re-arranged to be in a logical order from most useful to least. The completed UI is shown in Figure 5.1.

Chapter 5

Results

After implementing the new estimations, a few simple tests were run on a basic test ship design. This ship was largely incomplete, but had a few detailed sections that could be used for testing base cases, and some basic estimations. More thorough testing was determined to be completed by a different company to ensure that the tool retained the existing functionality, and use more testing methods with different ship designs to determine the effectiveness of the extension. The results of these tests are not available at the conclusion of the thesis, but a follow up will occur in August 2019 to review the results of the tests.

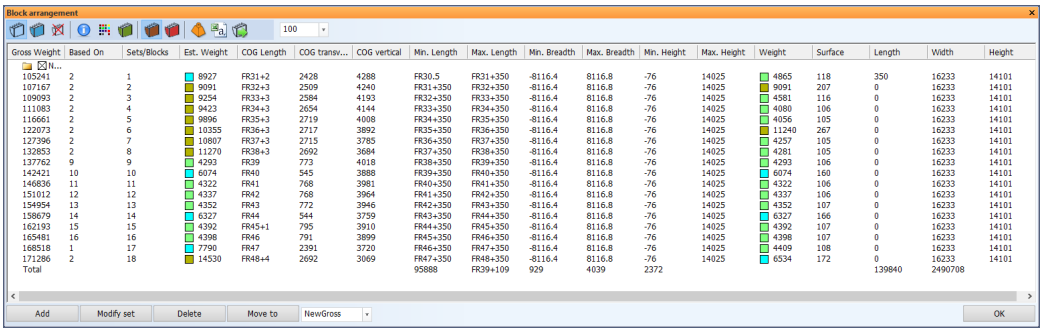

Figure 5.1: The Extended FlexiWeight Tool.

Figure 5.1 shows the results of the ship from Figure 2.5. It is broken into large slices that run from the tail towards the front. As we move higher in the block, the larger the ship gets in height and width. Each slice is slightly wider and taller than the widest section of the ship so that each slides includes every part of the ship within a single frame.

Let us consider the weight estimation first. Block 2 is set to use its own construction as the reference construction, (2 in second column). This means

that it will serve as a base case where we expect its estimated weight to be that of the actual construction. Which, according to the weight column, it is. Each block increases in size as the ship increases in breadth and height. Take for example, block 3 has a virtual increase of 1.79719% from block 2. The estimated weight of block 3 is 1.79298% higher than the weight of block 2. This is a strong indication that the weight estimations are accurate, but a concrete test is needed. These numbers are not concrete since this basic ship design is unknown when completed. However, we can find trends and conducted a separate finite test on construction that is known.

Block 1 has a slightly reduced estimated weight, which coincides with a somewhat smaller virtual construction, and blocks 3–8 have steadily increasing weights, of exactly the same proportions as the increase in virtual solid weights. This indicates that the estimations of weight are correctly taking the reference weights of their Based On block.

Next we examine the COG estimations. We can instantly see that the +300 millimeter offset is correctly derived from block 2 for each block that references it. We can also see that the asymmetrical construction is also found in all of the blocks that are set to use the actual construction of block 2. So far all of observations point towards an accurate estimation of weight and COG, but a test using known weights and COGs is required. So a new set of blocks was created, with specific construction that could be measured and checked against the estimated weights.

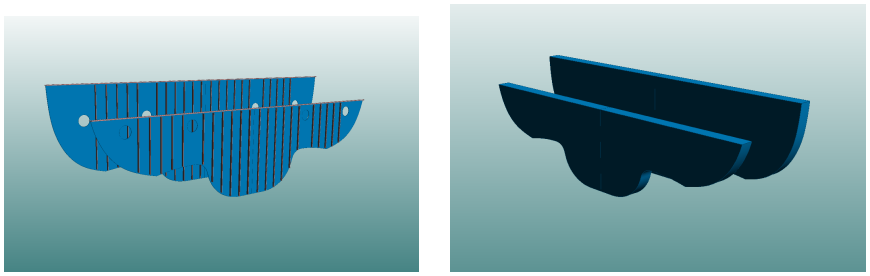

Figure 5.2: New construction (left) and virtual solid blocks (right) were created specifically for testing the estimation. Both actual construction sections are nearly identical.

Two new frames (Figure 5.2), with steel ribs and some cut away holes were created for testing the relation between known and unknown blocks. Both were measured for actual weight and COG before running estimations for verification. Using the actual construction of the smaller section, the larger section was estimated. The actual weight of the smaller section was 3137 kilograms, while the larger section was 4705 kilograms. Setting the smaller

block as the reference construction, our tool produced an estimated weight of 4686 kilograms, a COG length of FR36-35, breadth of 0mm, and height of 2696mm. Comparing these to the actual weight and COG, the errors were only 0.41%, 34mm, 0mm, 27mm respectively. This is clearly within acceptable error margins. Possible reasons for the higher than expected height of the COG was potentially due to a non-affine transformation of the four holes. These were not properly shifted when constructing the second frame. However, more thorough testing will be completed with many more ships and objects by an external company.

Chapter 6

Conclusion

In conclusion, the estimations of weight and center of gravity from the new extension to the FlexiWeight Tool delivers deliver accurate weight estimations and COG of incomplete ships with $< 0.5\%$ error in simple tests. These estimations are only accurate if the virtual solid blocks are changed through affine transformations, and will not provide accurate estimations of entirely different construction. As long as the related construction is similar in structure, the shapes determined by the virtual solid blocks will provide accurate estimates of weight and COG. This extension will require further testing which will be conducted by a separate company which will reduce bias and find possible fringe cases that require special calculations.

Chapter 7

Future Work

Future expansions to this extended FlexiWeight tool include the ability to parallelize the calculations, and incorporate automated testing to ensure functionality continues with every release. Currently, each block is calculated sequentially. Each block is completely independent of each other. This indicates that a very simple distribution of work across available cores could be achieved. This could be accomplished with openMP or possibly even openMPI if the resources are available to split work amongst a cluster. In projects with massive numbers of parts or blocks, this could result in a direct reduction in time required to compute related to the number of extra cores/nodes available.

Another extension would be the incorporation of automated testing. This would not effect the tool itself, but instead ensure that any changes to the code will not break the functionality of the tool in future releases. A test ship would be required where the frames are known, and the estimates are calculated within a certain error threshold.

Lastly, a re-work of the original algorithm could reduce the time complexity by switching the outer loop with the inner loop. This would iterate over massive list of parts only once instead of per block.

Chapter 8

Acknowledgments

I would like to thank Mădălina Florean and Jiri Kosinka for all of their help and supervision, this would never have been possible without them. I would also like to thank everybody at CADMATIC for all of their help, and the wonderful opportunity to experience software development in the real world. I would also like to thank Lisa for all of her motivation and support throughout my bachelor.

References

[1] Editors of Encyclopedia Britannica. *Center of Gravity*. Encyclopedia Britannica. https://www.britannica.com/science/centre-of-gravity. 2019.

[2] S. Harries, G. Dafermos, A. Kanellopoulou, M. Florean, S. Gatchell, E. Kahva, P. Macedo. *Approach to Holistic Ship Design - Methods and Examples*. COMPIT 2019, Tullamore

[3] A. Yrjänäinen, M. Florean. Intelligent General Arrangement, Marine Design XIII, Vol.1: 13th Int. Marine Design Conf. 2018, Helsinki

[4] The EC HOLISHIP (HOLIstic optimization of SHIP design and operation for life-cycle) project. http://www.holiship.eu/