Universität Rostock Traditio et Innovatio

Information Technology in Ship Design and Production

Calculation of Hydrostatic Parameters from a Triangular Mesh Representation of a Hull

Submitted by

Manoj Kumar Suresh	217100293
Sreedhar Kokkarachedu	216100230

Supervised by

Prof. Dr.-Ing.Rober Bonsert
Dipl.-Ing. Hannes Lindner

January 19, 2018

Contents

List of Figures

1 Objective

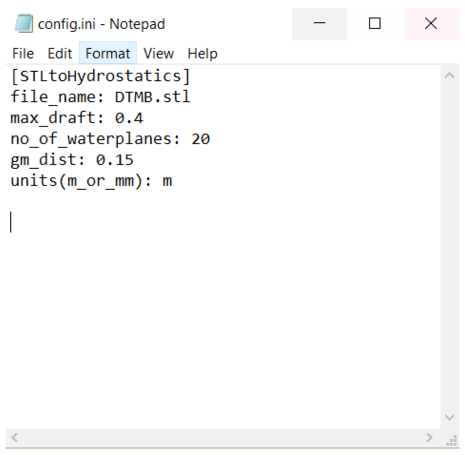
The objective of this project is to develop a tool to calculate the basic hydrostatic parameters of a ship's hull form, represented in the form of triangular mesh using any programming language. The hydrostatic parameters of a ship's hull are of utmost importance during the various stages of the ship design. Therefore, it is useful to have a handy tool available to calculate the basic hydrostatic parameters.

2 Motivation

Calculation of hydrostatic parameters even for a single draft consumes considerable amount of time, when calculated manually. When doing the same for various drafts, even with the help of design softwares leads to repeating same process manually and human errors. In order to reduce the computational time and to eliminate human errors, we are motivated to develop a tool for calculating the basic hydrostatic parameters of any hull form using python. Among many programming languages python proved to be useful as both efficient and open source, thus encouraged us to use it to develop the tool.

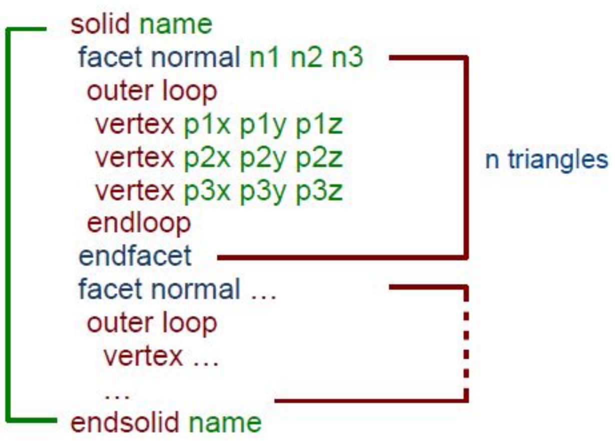
3 Introduction

This tool takes the input in the form of a config file ”config.ini”, which consist of the input parameters for the program to run as shown in Figure 1. The program accepts only the ”STL” file format. STL means Standard Triangulation Language, which specifies ASCII format representation. The STL file format has become standard data transmission format for the Rapid Prototyping and On Demand Manufacturing. This format approximates the surfaces of a solid model with triangles[1]. The first parameter in the config file is the name of the ”STL” file of the ship hull form. The user has to specify the name of the file here.



```
[STLtoHydrostatics]
file_name: DTMB.stl
max_draft: 0.4
no_of_waterplanes: 20
gm_dist: 0.15
units(m_or_mm): m
```

Figure 1: 'Config' file



```
solid name
 facet normal n1 n2 n3
  outer loop
   vertex p1x p1y p1z
   vertex p2x p2y p2z
   vertex p3x p3y p3z
  endloop
 endfacet
 facet normal ...
  outer loop
   vertex ...
   ...
endsolid name
```

Figure 2: 'STL' file format

The STL file consist of coordinate position of vertices of a surface. The file starts with "solid" followed by the name of the file and ends with "endsolid" as shown in the Figure 2. Each loop forms one triangle and each triangle represents the surface of the model. 'facet normal' in the file represents the normal vector of the triangle. The loop continues until all the coordinates are covered. The STL file only stores the value of the coordinate, but not its units. So, in order for the tool to be efficient all models must be created in meters.

The "config.ini" file as shown in Figure 1 consist of the input parameters. Apart from the STL filename, The user has to enter the total draft height and number of water planes to be created. Optimal number of water planes are required for near accurate results and less computation time. The "gm_dist" is the distance between the center of gravity and metacentric height of the hull. It can be altered by the user as per their requirements, but less than the present value is not advisable. Units of the STL must be mentioned in the last position.

4 Methodology

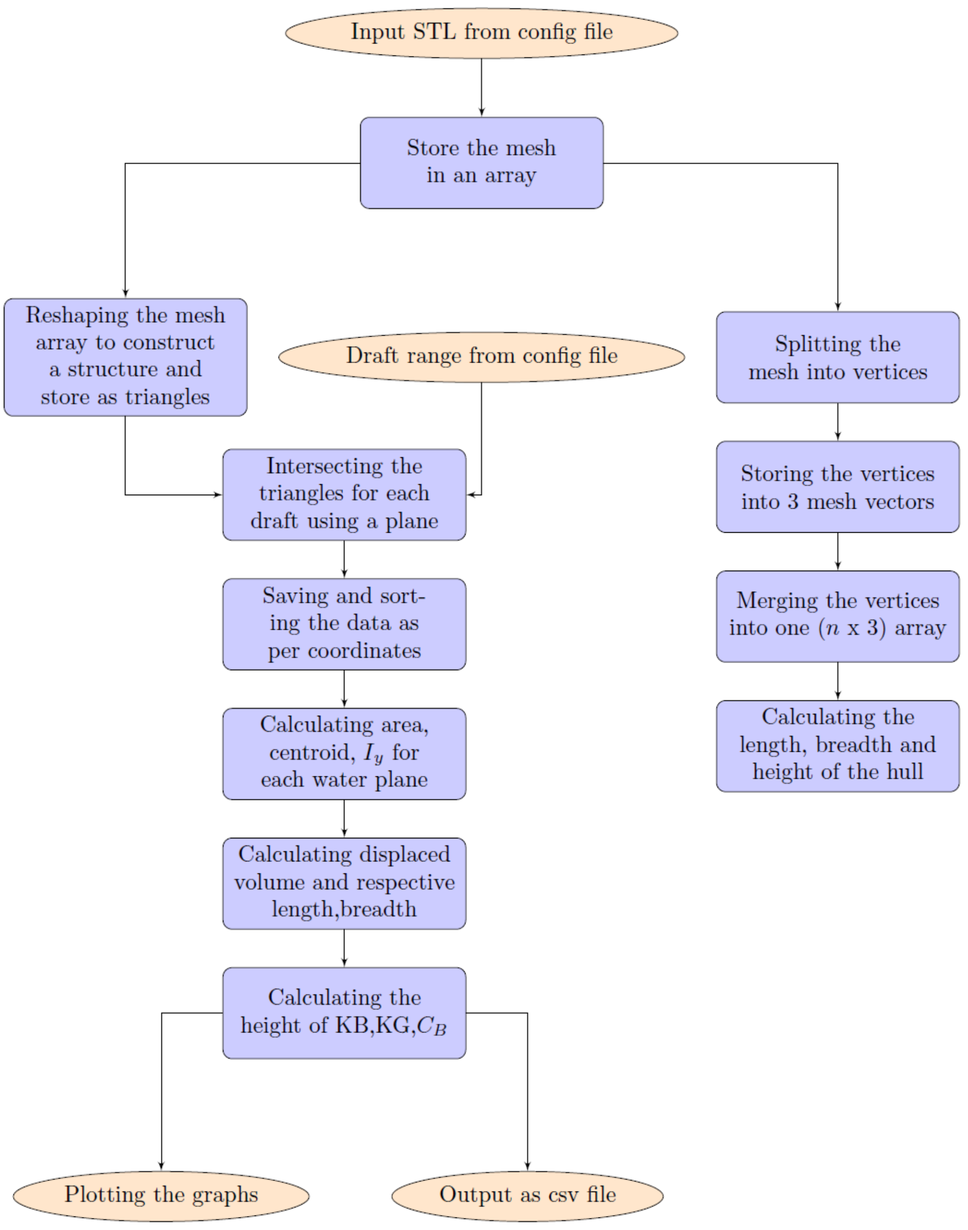

Input STL from config file
Store the mesh in an array
Reshaping the mesh array to construct a structure and store as triangles
Draft range from config file
Splitting the mesh into vertices
Intersecting the triangles for each draft using a plane
Storing the vertices into 3 mesh vectors
Saving and sorting the data as per coordinates
Merging the vertices into one (n x 3) array
Calculating area, centroid, I_y for each water plane
Calculating the length, breadth and height of the hull
Calculating displaced volume and respective length,breadth
Calculating the height of KB,KG,C_B
Plotting the graphs
Output as csv file

Figure 3: Flowchart

5 Implementation

The implementation of this project was done using python based on the flowchart as shown in Figure 3. The user can change the input in "config.ini" file present in the same folder. Assuming, the hull form in the STL file is in meters, the STL file name, total draft and the number of water plane sections are the major input and the "gm_dist" can be left default which is the minimum value. Advanced users can vary the "gm_dist" value as per their requirements. A step by step implementation is explained further.

5.1 Reading and storing the vertices

The program reads the ASCII STL file mentioned in the "config.ini" file. A function is created in the start to read the "config.ini" file[3]. This program does not read Binary STL files. To read the STL files, the program makes use of the "numpy-stl" library which is a part of the "numpy" a standard python library, which may present in the python package or can be downloaded from online python library[2]. After reading, all the vertices of the hull form will be stored in an $(n \times 9)$ numpy array, where the 9 columns represent the (x, y, z) coordinate of three vertices of a triangle and n represents the number of triangles. The next step is to split the mesh into vertices by assigning each (x, y, z) coordinate to their respective vertices (v_0, v_1, v_2) of the triangle in such a way that v_0 contains the first vertex of all triangles forming a $(n \times 3)$, the same is done with other two vertices to form the mesh vectors. Then the three mesh vectors are merged into one vector of $(n \times 3)$ array using "numpy.concatenation" command. Three functions are created to calculate the length, breadth and height of the hull form, from the newly created vector, by finding the difference between maximum and minimum values of each coordinate.

5.2 Creation of triangles

Since this vector does not possess the vertices of the triangle in an order, we need to create a structure for storing triangles. Which was created by reshaping the initial $(n \times 9)$ numpy array to construct the structure of a triangle and stored it in an array named "tri_array" using for loop and reshape command. The "tri_array" is created in such a way that the first three vertices will form the first triangle and the second three will form the second triangle and so on. In order to get the intersection point, a plane is created at the specified draft and the point of intersection is calculated for each line segment of the triangle using the line-plane intersection equation.

5.3 Triangle-Plane intersection

Each triangle is divided into three line segments and the line-plane intersection is calculated for each line using for loop for each draft. The intersection point of line and plane is calculated using the equation (5). Before calculating the intersection point, we need to make sure whether the plane cuts the line or not. In order to do that, the distance between the plane and each vertices of a line segment is calculated as d_0, d_1 respectively from the "tri_array". The distance is calculated only for z coordinate of each vertices v_{0z}, v_{1z}, v_{2z}. If the plane intersects a line segments, its corresponding product of d_0 and d_1 will be negative. Only these line segments are taken into account for the next step ignoring other lines which lies completely above or below the plane.

$$d_0 = plane - v_{0z}$$
$$d_1 = plane - v_{1z}$$

The above equation is used to calculate the distance between the plane and vertices of one line segment of a triangle in Figure 4

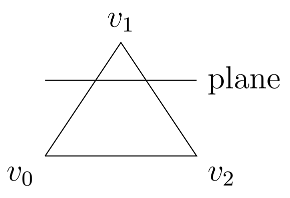


Figure 4: Intersection of triangle by a plane

Once the condition d_0 x d_1 ¡ 0 is satisfied the program takes the line segment to next step, else it will discard that line. Here, the intersection ratio p in equation (1) is calculated and substitute in the equation (5) to calculate the intersection point for one line segment[4].

$$p = \frac{d_0}{d_0 - d_1} \tag{1}$$

$$r_t = v_0(x_0, y_0, z_0) + p(v_1(x_1, y_1, z_1) - v_0(x_0, y_0, z_0)) \tag{2}$$

The (x, y, z) coordinates for each vertex is obtained from the "tri_array". Running this in a "for loop" will calculate the intersections points for single draft by following the same procedure for each line segment of all triangles.

5.4 Sorting the points to create water plane

Once the intersection points are calculated, they are stored in an array named "inter_pt" which is an $(n \times 3)$ numpy array. Since the points are not sorted, calculation of water plane area is not possible. Hence, they are sorted according to the xy coordinates in clockwise direction. Since the vertices have same z coordinate, we can assume it as a 2D array. The vertices are split into three arrays as per these conditions $y > 0$, $y = 0$, $y < 0$. Each array is sorted as per the x

coordinate in ascending order. Finally merging the three arrays using "numpy.concatenate" about x axis will sort all the vertices in clockwise direction.

5.5 Calculating area, centroid and momet of inertia for each water plane

The area of each water plane is split into number of small symmetric trapeziums. Each trapezium is formed from the intersection points. From the Figure 5, the h is the x axis distance between two consecutive points in the array "inter_pt" and a and b values are the twice the y axis distance for each point. Since we accept only half the hull as an input, twice the y axis distance will give us the total water plane area for the symmetric hull.

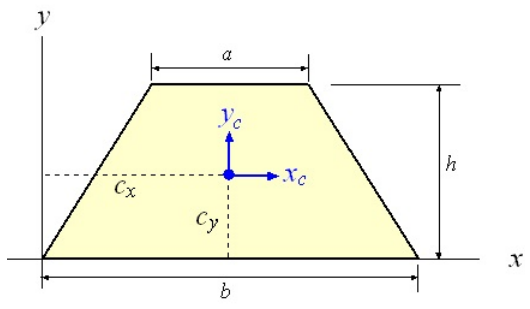


Figure 5: Symmetric Trapezium

$$h = |v_{1x} - v_{0x}|,\ a = 2v_{0y},\ b = 2v_{1y}$$

$$A = \frac{(a+b)h}{2}$$

where A is area of symmetric trapezium[5], v_0 and v_1 are the consecutive points in the array "inter_pt". The total water plane area A_{wp} can be calculated by adding the areas A of all the trapeziums. The centroid and moment of inertia for the water plane can also be calculated in the same way. The centroid for each trapezium x_c located at its center since it is symmetric. When all the obtained values are substituted in the equations (3) and (4)the centroid C_x and moment of inertia I_y about the x-axis for total water plane can be calculated[5].

$$C_x = \frac{\sum(A \times x_c)}{A_{wp}} \tag{3}$$

$$I_y = \sum \frac{h(a+b)(a^2+b^2)}{48} \tag{4}$$

5.6 Calculating Displacement, its length and breadth and C_B

The displaced volume ∇ can be calculated by integrating the water plane area A_{wp} curve over draft as shown in equation (5). This can be done numerically by plotting an A_{wp} curve over draft range as shown in Figure **??**. The area under that curve can be calculated by splitting the area into simple rectangles and triangles, and adding their corresponding areas using the equation (6) where $(a \times h)$ is the area of rectangle and $(0.5 \times h \times c)$ is the area of triangle. This will gives us the displacement until each draft.

$$\nabla = \int_T A \tag{5}$$

$$h = |v_{1x} - v_{0x}|,\ a = v_{0y},\ b = v_{1y},\ c = |b - c|$$

$$\nabla = \sum ((a \times h) + (0.5 \times h \times c)) \tag{6}$$

Its corresponding length L and breadth B can be calculated from the "tri_array" by storing the vertices below the draft range T in a new array and subtracting the minimum distance from maximum distance for both x and y coordinates respectively. With these values we can calculate the block coefficient C_B using the equation (7).

$$C_B = \frac{\nabla}{L \times B \times T} \tag{7}$$

5.7 Calculating height of center of buoyancy and center of gravity

The height of center of buoyancy $\overline{KB}$ can be calculated by using the same equation (3) but with the z coordinates or the draft values T and the area of water plane A_{wp} for each draft as shown in the equation (8)

$$\overline{KB} = \frac{\sum(A_{wp} \times T)}{\sum A_{wp}} \tag{8}$$

The height of center of gravity can be calculate by substituting all known values in the equation (9) with the minimum height of meta-center to center of gravity GM_{min} as $0.15m$. This value is given in the "config.ini" file which van be changed by the user as per their requirements.

$$KG = BM + \overline{KB} - GM \tag{9}$$

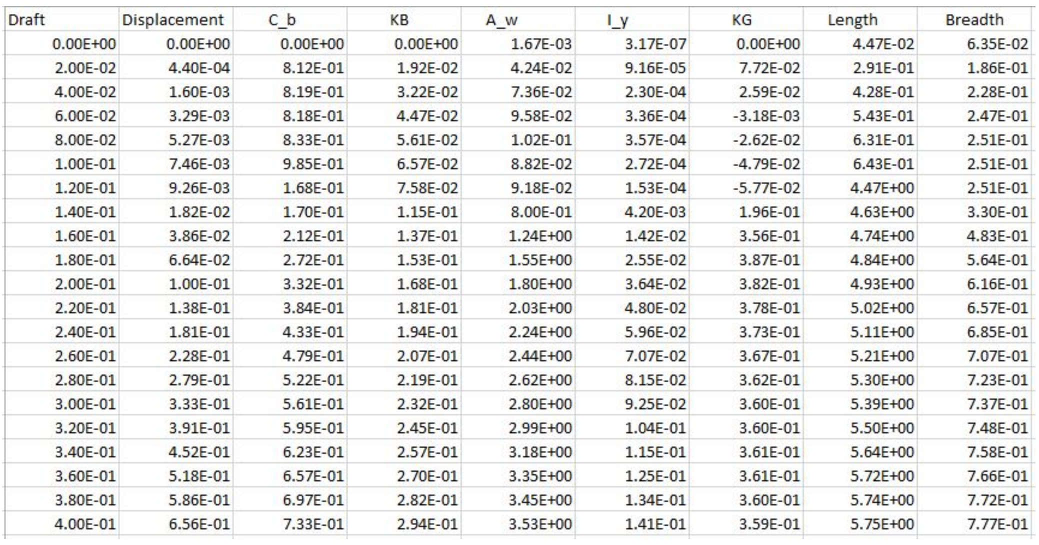
Where,

BM is the Meta-centric radius $= \frac{I_y}{\nabla}$

5.8 Saving the output and plotting the graph

All the above calculation will be carried out for each draft value T which is calculated from the number of sections entered by user in the "config.ini" from the base of the hull until the user specified draft T_{max}.

All the calculated values are stored in a comma separated file of format ".csv" as shown in the Figure 6.



Draft	Displacement	C_b	KB	A_w	I_y	KG	Length	Breadth
0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.67E-03	3.17E-07	0.00E+00	4.47E-02	6.35E-02
2.00E-02	4.40E-04	8.12E-01	1.92E-02	4.24E-02	9.16E-05	7.72E-02	2.91E-01	1.86E-01
4.00E-02	1.60E-03	8.19E-01	3.22E-02	7.36E-02	2.30E-04	2.59E-02	4.28E-01	2.28E-01
6.00E-02	3.29E-03	8.18E-01	4.47E-02	9.58E-02	3.36E-04	-3.18E-03	5.43E-01	2.47E-01
8.00E-02	5.27E-03	8.33E-01	5.61E-02	1.02E-01	3.57E-04	-2.62E-02	6.31E-01	2.51E-01
1.00E-01	7.46E-03	9.85E-01	6.57E-02	8.82E-02	2.72E-04	-4.79E-02	6.43E-01	2.51E-01
1.20E-01	9.26E-03	1.68E-01	7.58E-02	9.18E-02	1.53E-04	-5.77E-02	4.47E+00	2.51E-01
1.40E-01	1.82E-02	1.70E-01	1.15E-01	8.00E-01	4.20E-03	1.96E-01	4.63E+00	3.30E-01
1.60E-01	3.86E-02	2.12E-01	1.37E-01	1.24E+00	1.42E-02	3.56E-01	4.74E+00	4.83E-01
1.80E-01	6.64E-02	2.72E-01	1.53E-01	1.55E+00	2.55E-02	3.87E-01	4.84E+00	5.64E-01
2.00E-01	1.00E-01	3.32E-01	1.68E-01	1.80E+00	3.64E-02	3.82E-01	4.93E+00	6.16E-01
2.20E-01	1.38E-01	3.84E-01	1.81E-01	2.03E+00	4.80E-02	3.78E-01	5.02E+00	6.57E-01
2.40E-01	1.81E-01	4.33E-01	1.94E-01	2.24E+00	5.96E-02	3.73E-01	5.11E+00	6.85E-01
2.60E-01	2.28E-01	4.79E-01	2.07E-01	2.44E+00	7.07E-02	3.67E-01	5.21E+00	7.07E-01
2.80E-01	2.79E-01	5.22E-01	2.19E-01	2.62E+00	8.15E-02	3.62E-01	5.30E+00	7.23E-01
3.00E-01	3.33E-01	5.61E-01	2.32E-01	2.80E+00	9.25E-02	3.60E-01	5.39E+00	7.37E-01
3.20E-01	3.91E-01	5.95E-01	2.45E-01	2.99E+00	1.04E-01	3.60E-01	5.50E+00	7.48E-01
3.40E-01	4.52E-01	6.23E-01	2.57E-01	3.18E+00	1.15E-01	3.61E-01	5.64E+00	7.58E-01
3.60E-01	5.18E-01	6.57E-01	2.70E-01	3.35E+00	1.25E-01	3.61E-01	5.72E+00	7.66E-01
3.80E-01	5.86E-01	6.97E-01	2.82E-01	3.45E+00	1.34E-01	3.60E-01	5.74E+00	7.72E-01
4.00E-01	6.56E-01	7.33E-01	2.94E-01	3.53E+00	1.41E-01	3.59E-01	5.75E+00	7.77E-01

Figure 6: Output as ".csv" file

Each output parameter is plotted against the draft values in python and are saved in the same folder in ".pdf" format.

6 Guidelines for using the Tool

- If this program is running for the first time in the system which doesn't have the required libraries already installed in it. It will automatically install them from the online python library. It is advisable for the system to be connected to the internet.
- This program may not work properly in python ver. 2.7. It is advisable to use spyder (cross-platform for Python) which will have majority of the libraries installed in it.
- Please make sure the "STL" file and the python file is present in the same folder before executing the program.
- The output will be saved as 'output.csv' file and diagrams are saved in pdf format in the same folder.
- The user has to give the input in the 'config.ini' which also be placed in the same folder as the code. Default input has been given in the config file, further changes can be made by the user.
- Specify the unit of the stl file in the config file since stl files does not save the unit informations. 'Meter' is recommended.
- The Input STL file must contain only one symmetric half of the ship Hull, but the output will be calculated for complete Hull.
- Input Hull with positive coordinates would be advisable for easy visualization.
- Please ignore any error message regarding the STL file format.

7 Results and Discussion

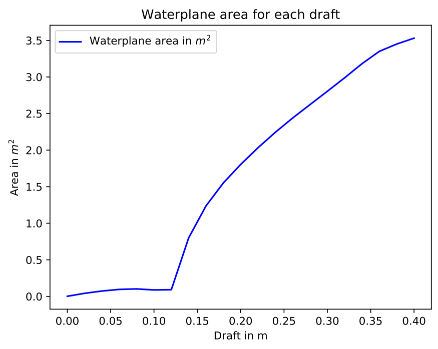


Figure 7: area of water plane vs draft

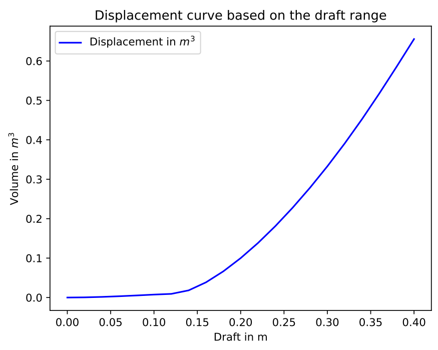


Figure 8: Displaced volume by ship hull vs draft

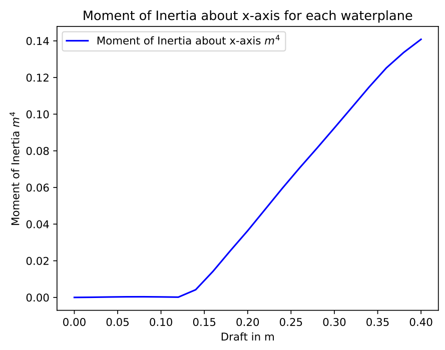


Figure 9: Moment of inertia vs draft

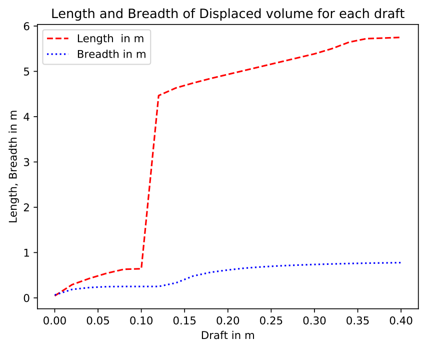


Figure 10: length,breadth of each displaced volume

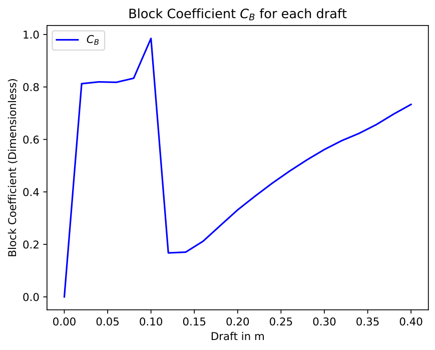


Figure 11: Block coefficient vs draft

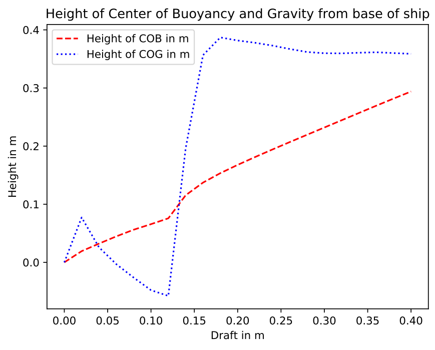


Figure 12: Height of centre of gravity ,centre of buoyancy vs draft

We studied that water plane area increasing while draft increases because ship area will increases from keel to above. From Figure 7.

we noticed from that Displacement increases with the draft range due to for the higher draft more volume of water will be displaced. We also observed that length is increased drastically while slight variation in breadth over along the draft range. Because length parameter has more variation from keel to above. From Figure 11, Block coefficient increases after severe fluctuation. This is because of irregular shape of hull.It will reach the maximum value of 1 if block fully occupies the displaced volume.

From figure 9 shown that moment of inertia is maintained zero and then increased linearly.This is due to increase in radius of gyration when draft increases. From Figure 12 Height of centre of buoyancy is almost linearly varying while height of centre of gravity is oscillating.The reason is centre of buoyancy depends of only centroid in z direction.So, draft in z direction increases height of centre of also increases.On contrast,the height of centre of gravity depends upon metacentric height and Metacentric radius.so,it fluctuates.

8 Conclusion

The program reads the STL file and based on the number of draft it creates slicing planes, finds the intersection points of triangle with section plane. Then we sorted and store the all the points as per coordinates. From that we calculated the each desired hydrostatic parameters using above equations and from STL file. Then all the output hydrostatic parameters shown in ".csv" format and graphs are plotted. All out Objectives are met it is fast and reliable. There are certain limitations such as not reading binary STL files and conversion of units which can be overcome by updating further.

References

[1] STL-file - Description
`https://www.3dsystems.com/quickparts/learning-center/what-is-stl-file`

[2] Numpy - A Python library
`https://pypi.python.org/pypi/numpy`

[3] Configparser - To create and read "config.ini" files
`https://wiki.python.org/moin/ConfigParserExamples`

[4] Intersection of Plane and triangle
`https://https://stackoverflow.com/questions/3142469/determining-the-intersection-of-a-triangle-and-a-plane`

[5] Trapezium - calculations
`http://www.efunda.com/math/areas/IsosTrapezoid.cfm`