Master of Aerospace Engineering
Research Project

1MAE400

2nd semester Report

Mass estimating and structure sizing of a reusable launcher

Author:
Arthur GUY

Responsibles:
Laurent BEAUREGARD
Joseph MORLIER

July 1, 2019

Abstract

This research project aims to provide a tool computing the mass of a lunar launcher and sizing its structure. The study first focus on design choices to be made and then proposes an architecture and a tool started from scratch implementing several studies made in the field of launchers and landers. At the end of this study results are compared to design already existing, from old ones to brand new ones in order to validate these results. In conclusion, for the first part of a study that will be done in two parts, basic computations using only open source softwares appears to be sufficient for the preliminary design of reusable launcher. The first part, found in this report, is about the fuel mass estimation and the FEM design using scripting and batch mode. The second part will be about running analysis over the FEM design and implementing optimization loops for thickness and geometry of each element of the design.

Keyword : Multidisciplinary Design Optimization, Reusable lunar launcher, Mass estimation, Structure sizing, Open source

Contents

Glossary

I_{sp} **Specific Impulse.** 11

FPR Flight Performance Reserve. 12

LH2 Liquid Hydrogen. 6

LLO Low Lunar Orbit. 11

LOP-G Lunar Orbital Platform-Gateway. 3

LOX Liquid Oxygen. 6

MDO Multidisciplinary Design Optimization. 3

NRO Near Rectilinear Orbit. 11

SoA State of the Art. 5

Introduction

The U.S. (with NASA) wants to go back to the moon and create a Lunar Orbital Platform-Gateway (LOP-G). Which can be considered as a new International Space Station but around the Moon. In this context, supplies will have to be transferred by a vehicle acting as a shuttle between the surface of the Moon and this Space Station. The objective of the global project is to design a reusable launcher capable of doing this round trip with a given payload. All the subsystems; trajectory, propulsion, structure, etc, of the vehicle must be optimized to obtain the best performances. Such a problem is best handled by the formalism of multidisciplinary design optimization (MDO) for which OpenMDAO has been designed for. The research project you are currently reading the report from is focused on the structural sizing and mass estimation of this reusable launcher

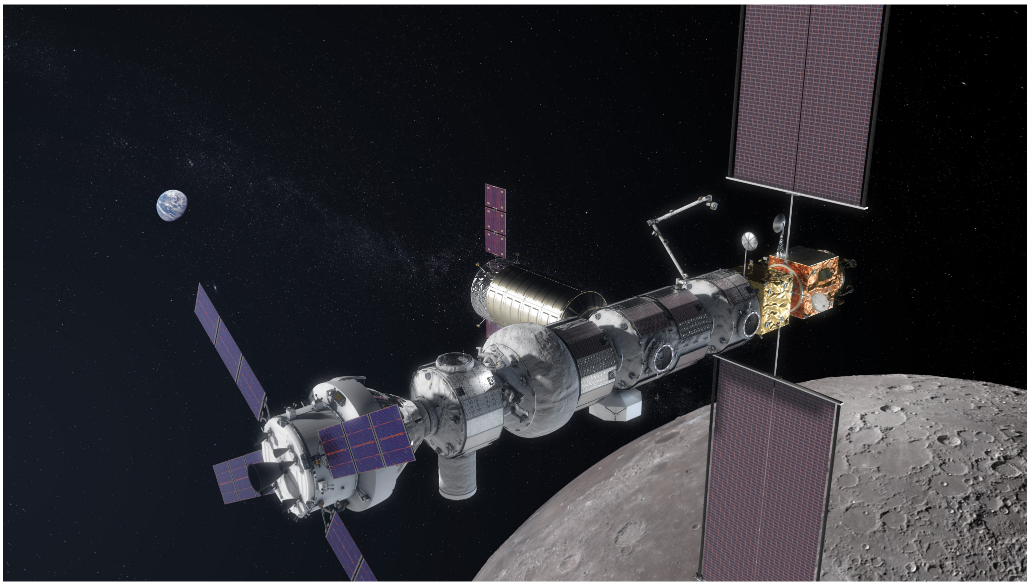

Figure 1: Lunar Orbital Platform-Gateway

1 Project's goal

The global project aims to provide a tool which will give the specifications and the design of a reusable launcher depending on inputs provided by a "client". This tool will be based on a MDO structure. Multidisciplinary Design Optimisation allows to optimise a whole system thanks to connection and feedback loops between the subsystems and their components. The connections and the structure of the MDO tool for the reusable launcher is found in the figure below.

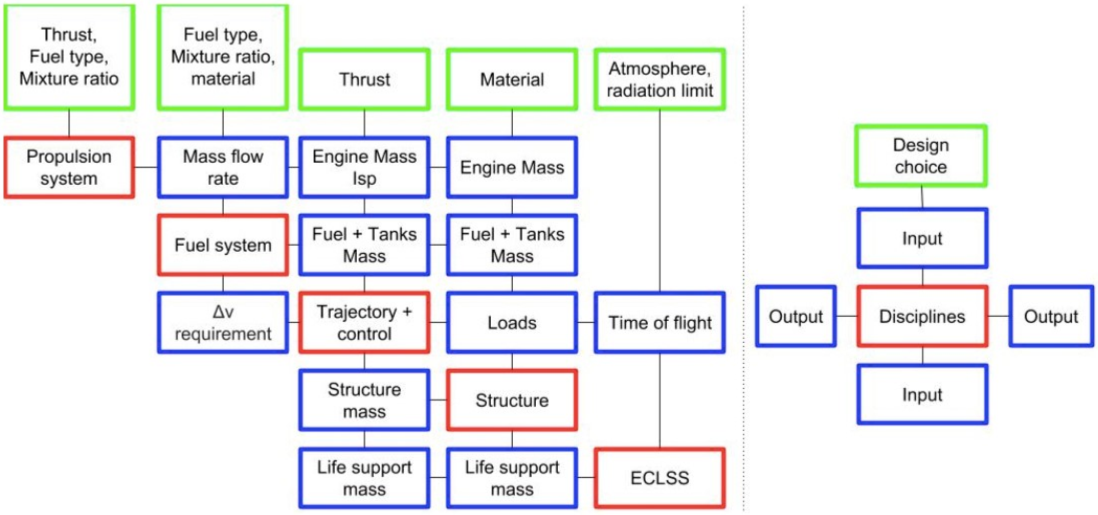

Figure 2: MDO structure of the whole project

In this project, my work will be focused on providing a tool relative to the structure "block". That is to say my tool will have to compute the sizing and the mass estimation of the structure. Furthermore, as you can see on Figure 2, the structure "block" has 4 inputs (inputs are on the vertical axis). Engine mass, life support mass (as well as the payload) will all be taken as a blackbox with one mass value so I won't have to make any calculation about it. However the fuel and tanks mass computation will be part of my work and these calculation will be performed with my tool. The loads will come from the trajectory block, but while my tool is being built these data will be input by hand as constants.

2 State of the Art

As for now, the project will focus on the mass estimation of tanks and propellant and the landing structure. Everything else will be considered as a black box, meaning : embedded system, electronic components, instruments, habitable environment and shielding will be considered as a single mass, chosen by Mr. BEAUREGARD regarding the mass of the ORION spacecraft. This black box mass is about 10t and is considered as a cylinder at the top of the Reusable Launcher. You can see on Figure 3[1] the basic shape of a lunar lander, this lander is a conceptual model designed by NASA. It will be seen in this State of the Art (SoA) that every lander looks quite the same, except from some specificities. This SoA will help us in the design choices that will need to be made.

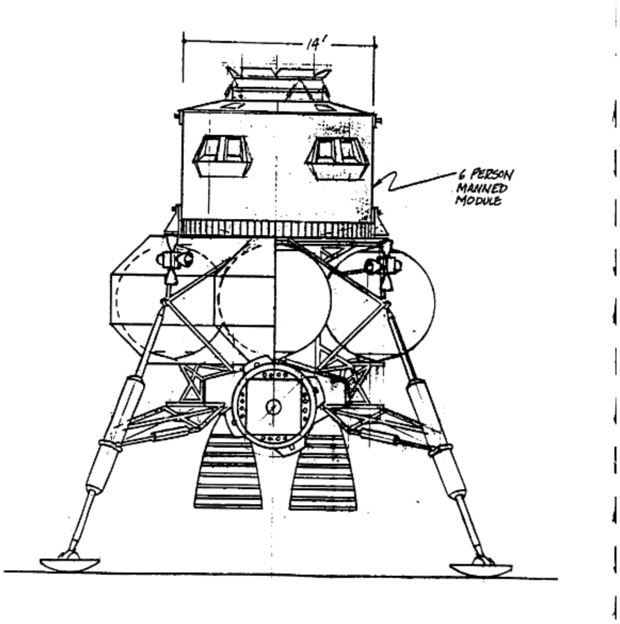

Figure 3: Reusable Lunar Lander Concept from NASA

2.1 NASA - Apollo Lander Module

The Apollo module, used by the USA to land astronauts on the moon, was the first of its kind and paved the way for future planetary and lunar lander. Indeed the lander from NASA was really basic and the landers designed today are really similar. The LEM used 4 landing legs, each made of 1 main strut and a truss configuration. The damper where made of honeycomb crushable absorbers, as it was landing only once. The lander/launcher was made of two separates stages, one landing the whole structure and staying there, the other launching the astronauts back to their spacecraft. The fuel used was the couple N2O4/Aerozine 50. Speaking of the mass, the module had a dry mass of 2000 kg and a total mass of 15000 kg.

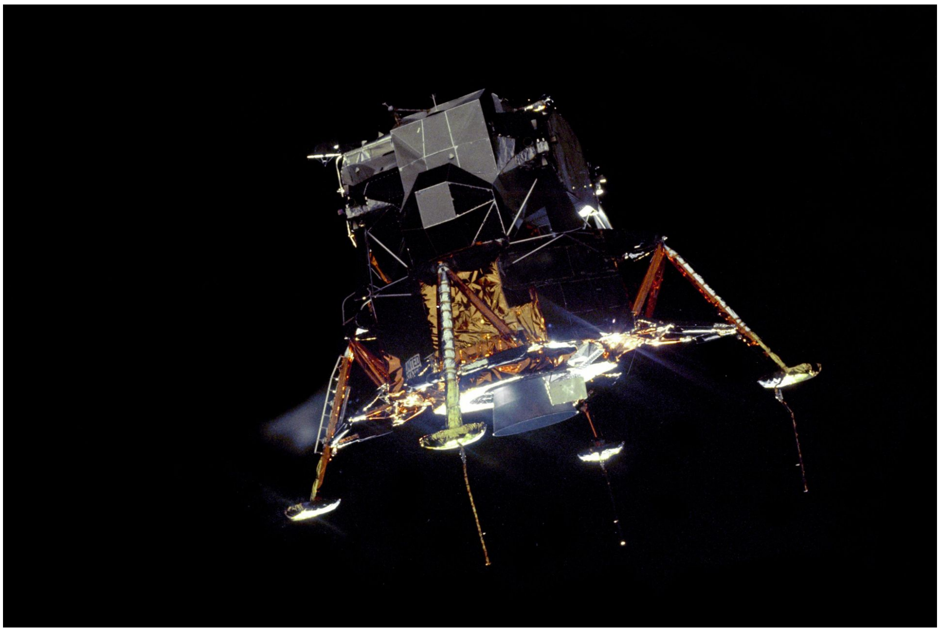

Figure 4: Apollo lander module

2.2 Lockheed Martin - Lunar lander

2.2.1 2018

The lander, presented by Lockheed Martin during the 69th International Astronautical Congress[2], is one of the most recent reusable lander design existing. With a dry mass of 22 metric tons and a launch mass of 62 metric tons this module is capable of transporting a 1000 kg payload. The couple Liquid Oxygen/Liquid Hydrogen (LOX/LH2) has been chosen as propellant. For two main reasons. According to them it is the only fuel couple that allows a round trip for a single stage lander/launcher operating from the gateway orbit thanks to its high I_{sp}. Indeed the round trip ΔV from the gateway to the surface of the moon is around 5000 m/s. And a single stage lander means full reusability so it is a big criteria. Although

LOX/CH4 could also work for a single stage lander, it will imply an increase in technology as ”the mass fraction required implies very lightweight system”, which is not preferable. The second reason to choose LOX/LH2 is that it can be made out of water, and water is known to be present on the moon. Therefore it would allow to in situ production of fuel. Talking about the structure, it has as well 4 legs but each made of one main strut and one secondary one. 4 thrusters are needed to propulse this heavy lander.

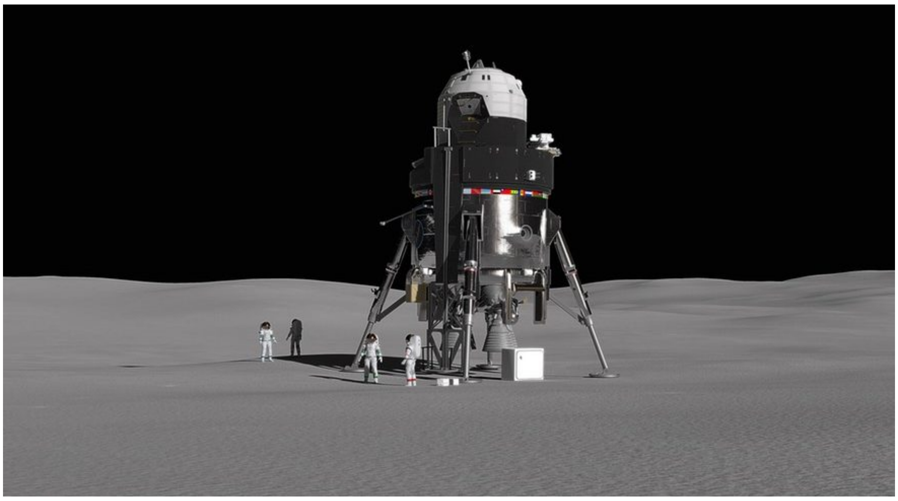

Figure 5: Lockheed Martin’s lunar lander

2.2.2 2019

In 2019, Lockheed Martin has presented a new design. This new lander is made of 2 stages, one lander and one launcher on top of it, as the Apollo module design. And most of the module is derived from the Orion spacecraft. This surprising new trajectory made in only one year is explained by political matters. Indeed NASA as been asked by Mike Pence, Vice President of the U.S., to move up the deadline for Americans return to the moon to 2024. Four years earlier than NASA’s previous target of 2028. Hence, Lockheed Martin endeavored to design a lander that would be easy and quick to design and build. The reusability constraint is then no longer considered.

2.3 Blue Origin - Blue Moon

The main particularity of this lander, designed by Blue Origin (founded by Jeff Bezos), is its adaptability. Indeed, the lander is strictly a lander it does not have a pressurized module or anything like this. It has a platform where customers can put whatever payload they want on. By doing that Blue Origin made its lander really versatile. As the previous landers from Lockheed Martin, Blue Origin chose to use LOX/LH2 fuel and designed its lander with 4 landing legs. However Blue Moon is not reusable and will only land once.

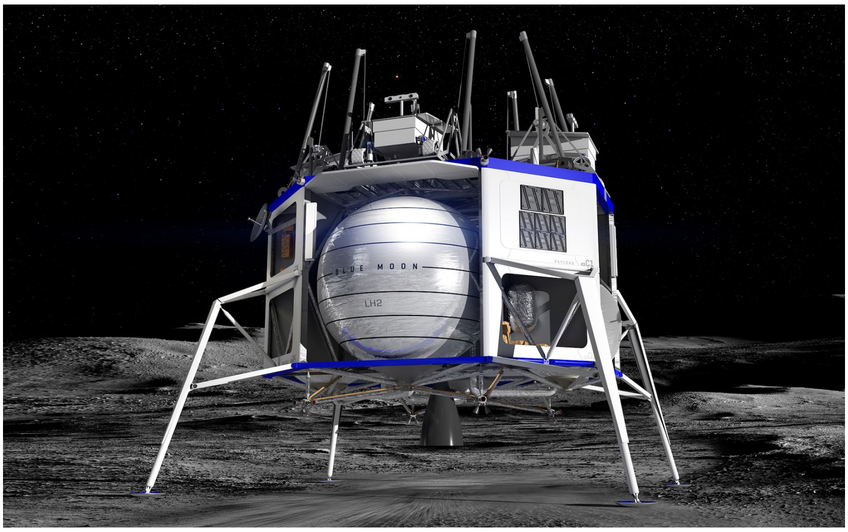

Figure 6: Blue Origin's lunar lander

2.4 New space

On Figure 7 can be seen several models of landers from different startups. All of these startups come from all over the world. ispace from Japan, SpaceIL from Israel, Astrobotic and Moon Express from the U.S.. For now they only designed little landers capable of transporting up to 100kg of payload. But in the future they plan to be the main transportation choice for an Earth-Moon system. It shows us how this field of lunar landers is active nowadays and how important it could be for a country to have access to the moon.

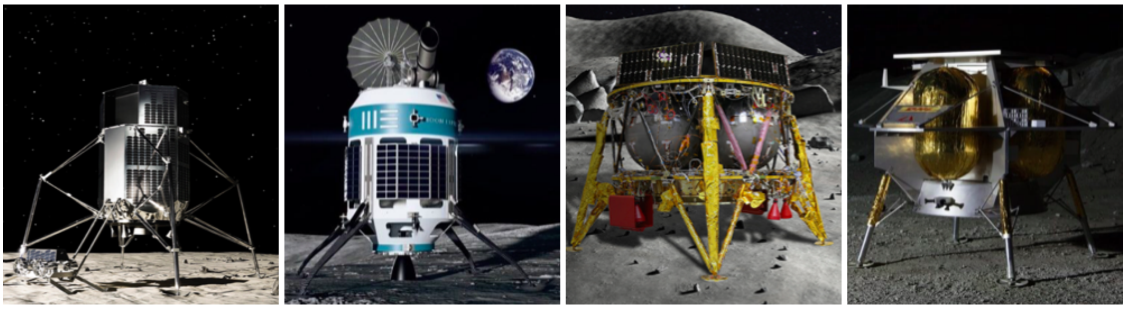

Figure 7: From left to right landers from ispace, Moon Express, SpaceIL and Astrobotic

2.5 Design choices

Thanks to this SoA we can already make several design choices. For instance it is clear that LOX/LH2 is the best choice for fuel. As Lockheed Martin explained for their lander, it's one of the only propellant allowing for a single-stage lander/launcher for our type of mission. It also proved its worth and as it has been widely used, the design of the propulsion system won't be costly. Talking about the structure 4 legs with a truss configuration seems to be the best choice, it offers a great stability for the landing phase.

3 Semester 2 work

At the beginning of this project, I have been asked only one thing : Provide a tool which gives the mass of the reusable launcher and its structure. Nothing less, nothing more. Hence my work started from scratch, the first goal was to find what tool was possible to build in order to do what I was asked. After a month or two of state of the art on landers, and looking for the key points of a mass estimation and the structure sizing, it was clear that a main code doing computations and asking for softwares to run analysis in a loop was the best solution. As I had never done that lots of research on the net about batch mode were needed while implementing the computations loops. On the figure below an overview of the tool I am creating.

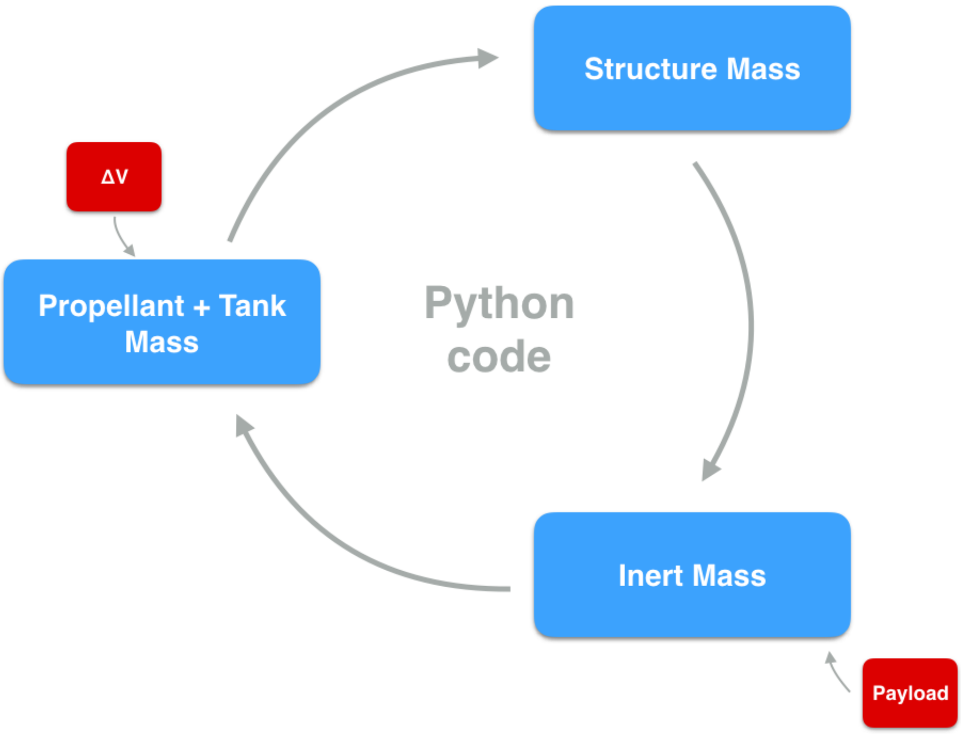

Figure 8: Scheme of how the tool operates

Thanks to a python code, I am creating a loop computing the fuel mass and the structure mass needed for a specific ΔV and a specific payload as inputs. At each loop a new inert mass (corresponding to the total mass without fuel in this study) is obtained and therefore additional fuel and then structure have to be computed. After a few iterations the total mass converges to a certain value. The loop stops when the new total mass value is less than 1 kg different from the previous one.

In the subsection below will be explained the operations inside the blocks. That is to say the computation of the fuel and tanks mass, and the structure sizing and mass estimation.

3.1 Fuel and tanks mass computation

3.1.1 Rocket equation

Propellants are the biggest part of launchers in terms of mass. It is crucial to have a good estimation. By using the rocket equation (1), we can retrieve the propellant mass of the spacecraft, depending on mission maneuvers.

$$\Delta v = I_{sp} g_0 \ln\left(\frac{M_{tot}}{M_{inert}}\right) \longrightarrow \frac{M_{tot}}{M_{inert}} = \exp\left(\frac{\Delta v}{I_{sp} g_0}\right) \tag{1}$$

As you can see in these equations the input would be I_{sp} and Δv, the first depends on the propulsion chosen, and the second one depends on the maneuvers the spacecraft needs to achieve. The values can be found in Table 1. The launcher will have to go from the moon surface to the Low Lunar Orbit and then to a Near Rectilinear Orbit (NRO) where the LOP-G is. This corresponds to a ΔV of 5100 m/s.

Table 1: Δv for lunar mission [3]

Mission phase	Δv km/s
LLO to moon surface	1.9
Moon surface to LLO	1.8
To/from LLO from/to NRO	0.7

With this we retrieved the ratio $\frac{M_{tot}}{M_{inert}}$. In the first computation loop the inert mass consists of the payload (blackbox) only. Therefore, by multiplying the inert mass to this ratio we obtain the total mass, and by subtracting this mass to the inert one we have the fuel needed.

This first computations are as follow in the python code.

```
# rocket equation
R = exp(deltav / (I_sp * g))

M_tot = M_inert * R
M_prop = M_tot - M_inert

M_prop = M_prop * 1.07 # 4% for FPR Propellant, 3% for Unusable Propellant
```

Listing 1: Rocket equation code

The last line adds propellant for Flight Performance Reserve (FPR) and for unusable propellant. The reason is because you can't fly with only the theoretical amount of fuel. You need more in order to be sure you won't run dry and to have precise changes in trajectory.

3.1.2 Tanks sizing and mass estimation

In this part equations from a University of Maryland's paper[4] have been applied.

With these equations are found the mass of LH2 and LOX tanks, as well as the mass of their insulation and their diameter.

```
# mixture ratio LOX/LH2, 6:1
M_LH2 = M_prop / 7
M_LOX = 6 * (M_prop / 7)
M_LH2_1 = M_LH2 / 2  # 2 tanks for mass distribution
M_LOX_1 = M_LOX / 2  # 2 tanks for mass distribution

# LOX Tank
M_LOX_Tank = 0.00152 * M_LOX_1 + 318
V_LOX_Tank = M_LOX_1 / rho_LOX
r_LOX_Tank = (V_LOX_Tank / (4 * pi /3))**(1 / 3)  # radius of LOX tank
A_LOX_Tank = 4 * pi * (r_LOX_Tank**2)  # Area LOX Tank
M_LOX_Insu = 1.123 * A_LOX_Tank  # Mass LOX insulation
M_LOX_Tanks = 2 * M_LOX_Tank
M_LOX_Insus = 2 * M_LOX_Insu

# LH2 Tank
M_LH2_Tank = 0.0694 * M_LH2_1 + 363
V_LH2_Tank = M_LH2_1 / rho_LH2
r_LH2_Tank = (V_LH2_Tank / (4 * pi  /3))**(1 / 3)  # radius of LH2 tank
```

```
    A_LH2_Tank = 4 * pi * (r_LH2_Tank**2)  # Area LH2 Tank
    M_LH2_Insu = 2.88 * A_LH2_Tank  # Mass LH2 insulation
    M_LH2_Tanks = 2 * M_LH2_Tank
    M_LH2_Insus = 2 * M_LH2_Insu
```

Listing 2: Tanks mass code

3.2 Structure sizing and mass estimation

As for now, the structure mass is retrieved thanks to a simple percentage which decreases while the total mass increases. For instance the percentage is about 5% for a launcher of 40t and 4% for a launcher of 50t. This come from the fact that as a system gets bigger it becomes more efficient, hence its structural mass gets smaller relative to its total mass. Nevertheless, the goal is to design the gross structure and provide a specific sizing as well as a mass coming directly from the design.

The idea is to use gmsh[5] (a geometry and meshing software) and Nastran95 (a FEM solver software) in order to do that. After the computation of fuel and tanks are done, the python code begin the structure part. It writes the radius of the tanks found before in a .geo file and then open this file in gmsh. This .geo file has already all the geometry of the launcher apart from the tanks radius. gmsh creates the launcher thanks to that and is asked by the python code to mesh it.
After the meshing is done, a .bdf file is created and some lines (about loads, materials, etc...) are added in it by the python code. This .bdf is then opened in Nastran which will run the analysis of the launcher's structure. After the analysis, the thicknesses found thanks to optimization loops, are used to know the structure's mass and then used as input for the next calculation of fuel mass.

During the second semester, only the gmsh part has been done. The python code for this part can be seen below. The .geo file can be found in the Legacy at the end of this report.

```
    def structure(M_inter):

    y = -M_tot/2500000 + 0.067

    M_Struct = M_tot * y

    # opens the .geo file and writes the new radius
    with open("geo_launcher.geo", "r+") as f:
        f.seek(0) # rewind
        f.write("r_LOX_Tank = %s;\nr_LH2_Tank = %s; \n" % (round(
   r_LOX_Tank,2),round(r_LH2_Tank,2))) # write the new line before
```

```
        f.close()

    # meshing
    sp.call([GMSH, File, "-1", "-2"], shell=True)

    return M_Struct
```

Listing 3: Structure mass and gmsh code

4 Results

4.1 Mass estimation

Delta V, Ascent	0	*2.28	*2.28
Payload, Ascent	0	6,000	0, Inert mass returned to LLO
Delta V, Descent	2.10	2.10	2.10
Payload, Descent	25,000	6,000	14,000
Total Inert Mass	9,823	9,823	9,823
Structure	1,681	1,681	1,681
Engines	822	822	822
RCS Dry	411	411	411
Landing Syst.	784	784	784
Thermal Prot.	2,017	2,017	2,017
Tanks	3,025	3,025	3,025
DMS (GN&C)	150	150	150
** Elect. Power	478	478	478
Airlock/Tunnel	455	455	455
Total Prop. Mass	25,251	32,395	30,638
Ascent Prop.	0	11,334	7,240
Descent Prop.	22,597	18,137	20,486
Unusable Prop.(3%)	678	884	832
FPR Prop. (4%)	904	1,179	1,109
Usable RCS	858	689	778
Unusable RCS (5%)	43	34	39
FPR (20%)	172	138	156
Deorbit or Gross Mass (less payload)	35,074	42,218	40,461
Deorbit or Gross	60,074	48,218	54,461

Figure 9: Mass Budget of a conceptual design from NASA[1]

On Figure 9 is found a table showing the budget mass of one of their conceptual reusable lander. In order to validate my code I used the 2nd column of this table in comparison of the results from my code.
To do so I had to compute the blackbox to take as input in my code, as well as the ΔV. For the ΔV I just had to sum the two ΔV, which is 4,380 m/s. For the blackbox, I did this : $Blackbox = Payload + Inert\ Mass - Structure - Landing\ Syst. - Tanks$. Which gives a blackbox of 10,333 kg.
After making my code running these values were found

As it can be seen all the values from my code are slightly lighter, which in the end gives a total mass 12% lighter from the NASA one.

Table 2: Validation results

	NASA concept (kg)	Code results (kg)
Inert Mass	9,823	9,062
Structure	2,465	2,122
Tanks Mass	3,025	1,906
Prop. Mass	32,395	27,352
Total Mass	48,218	42,415

It can be explained by several things. First, the tanks in my model are strictly sized for the amount of fuel needed. Which is not the case in real life where the tanks are often bigger than needed in order to be filled more or less depending on the mission.
Also, the concept from NASA was made in 1988, we can assume that in over 30 years some improvements have been made in the field. And then we have a snowball effect : because the tanks are lighter, the structure is lighter and because the structure is lighter we need less fuel, etc... In the end, the results are coherent.

I took another example, the Lockheed Martin one, more recent this time but less detailed because from a private company. To compare results this time, as only the inert and total mass are known, I inserted arbitrarily chosen blackboxes until I had the same Inert mass as a result to compare the total masses. In result when the inert mass from my code was about 22,000 kg like the Lockheed Martin's one, the total mass was about 66,000 kg, which is 4,000 kg more than the Lockheed Martin lander. This result is also coherent and acceptable

To conclude on this part, we do not have precise results with differences of less than a ton. But the work is about an estimation of the mass. We do not know all the specificities of all the launchers we can compare with. Hence, differences of 12% and 6% with two really different landers are more than acceptable for an estimation.

4.2 Geometry and Meshing

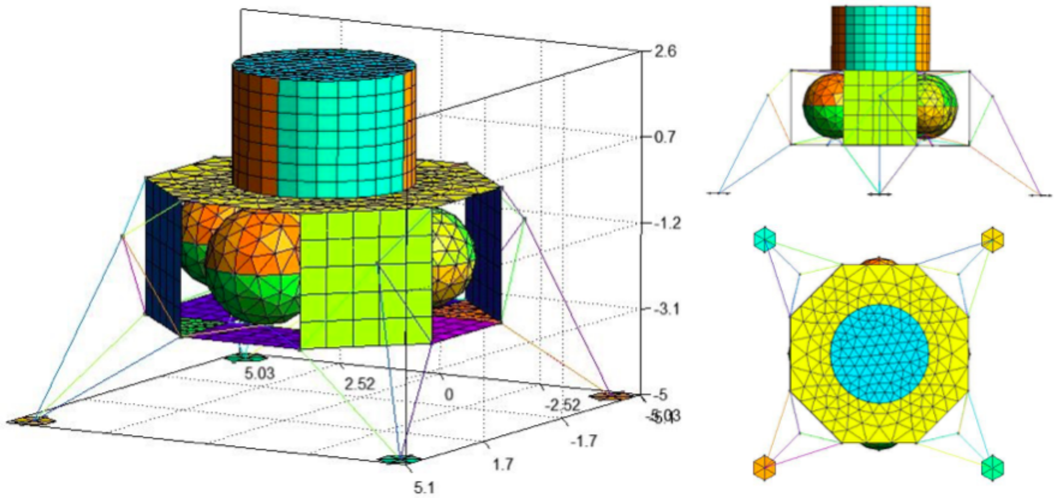

Figure 10: Geometry with mesh from gmsh

On Figure 10 different views of the lunar model with its mesh are seen. This model is a really basic one, using all the most common features of lander. It is made of 2D and 1D elements: bars and surfaces. The goal is to have a preliminary design, hence higher complexity is not desirable. For the mesh, quads have been set wherever it was possible, when not possible triangles are in place.
From this model a .bdf file is created and will be implemented in Nastran. A little change about the tank on the model will be made for the analysis. Indeed tanks are a really complex structure, not only spheres, there are a whole structure inside it. Since we already have there mass and only want a preliminary design, the tanks are going to be replaced by simple points having mass. But still attached the same way in the lander.

5 Future work in semester 3

5.1 Nastran

The work relative to the structural analysis by nastran is still to be done.
To begin with, the .bdf file obtained with gmsh only has data about the type of element (bars, surfaces, quadriangles, triangles, etc...) and their coordinates. Therefore several lines need to be added. For instance the materials, the properties in terms of inertia, the boundary conditions, etc... Also all the lines relative to the initialization of the analysis are to be written. After the analysis is run, the mass of the structure will be computed and then taken as input for the next calculation of propellant mass.
When this will be done, run tests of the whole code will be executed. In order to see if the structure of the code works well.
However an important thing will be missing and will be made afterward : optimization loops. Until then the thicknesses will be constants but we want an optimized structure at each iteration of the global loop.
Furthermore, the structural analysis will have to be run for different phases of flight. Launch and landing seem to be the worst cases in terms of stresses, so the best ones to size the launcher.

5.2 Further research

When everything will be running well, further research could be done. With Nastran we will have the total mass, its distribution and therefore the center of gravity of the launcher. Thanks to that possible analysis of the stability relative to the legs angle can be made.

References

[1] NASA. *Lunar Lander Conceptual Design*, 1988.

[2] Adam Burch Nickolas W. Kirby Timothy Cichana, Stephen A. Bailey. Concept for a crewed lunar lander operating from the lunar orbiting platform-gateway. In *69th International Astronautical Congress(IAC)*, oct 2018.

[3] Ryan Whitley and Roland Martinez. *Options for Staging Orbits in Cis-Lunar Space*, 2015.

[4] David L. Akin. *Mass Estimating Relationships*, 2005.

[5] C. Geuzaine and J.-F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. author. *International Journal for Numerical Methods in Engineering*, 2009.

[6] NASA. *NASA-STD-3000, Volume I, Section 8*, 1995.

[7] NASA. *Human Integration Design Handbook, Revision 1*, 2014.

[8] William Moore. *Treatise on the Motion of Rockets*. G. and S. Robinson, 1813.

[9] P. Perczynski and B. Zandbergen. Mass estimating relationships for manned lunar lander and ascent vehicle concept exploration. In *To Moon and Beyond*, sept 2008.

[10] Charles D. Brown. *Elements of Spacecraft Design*. AIAA, 2002.

[11] Walter E. Hammond. *Design Methodologies for Space Systems Transportation*. AIAA, 2001.

[12] Jean Pierre Aubry. *exploring the world of beams, plates, rods and cables structures in a linear and non linear fashion with Gmsh, Code Aster and Salome-Meca*, 2011.

A Legacy

A.1 python code

```
   # -*- coding: utf-8 -*-
"""
Created on Wed Apr 17 09:58:23 2019
s
@author: arthur guy
"""


from math import exp, pi
import subprocess as sp
#import time


###### path to the .geo file and to gmsh.exe
File = "C:/Users/arthu/OneDrive/Bureau/Nouveau dossier/test.geo"   #
    warning : put "/" instead of "\"
Path = "C:/Users/arthu/OneDrive/Bureau/Nouveau dossier"
GMSH = r'C:\Program Files\gmsh 4.3.0 Windows64\\gmsh.exe'


###### constants
g = 9.81 # m/s2
Epsilon = 2
rho_LH2 = 112 # kg/m3
rho_LOX = 1140 # kg/m3
I_sp = 450 # s
RCS = 700 # kg, Reaction Conctrol System propellant


###### input data
deltav_input = input('enter Deltav (m/s) : ')
Payload_input = input('enter payload (kg) : ') # black box

deltav = int(deltav_input)
Payload = int(Payload_input)


###### Initialization
M_inert = Payload
M_tot = Payload
i = 0


```

```
###### function


# computation of propellant and tank mass
def propellant_and_tank(deltav, M_tot, M_inert):

    # rocket equation
    R = exp(deltav / (I_sp * g))

    M_tot = M_inert * R
    M_prop = M_tot - M_inert

    M_prop = M_prop * 1.07 # 4% for FPR Propellant, 3% for Unusable Propellant

    # mixture ratio LOX/LH2, 6:1
    M_LH2 = M_prop / 7
    M_LOX = 6 * (M_prop / 7)
    M_LH2_1 = M_LH2 / 2  # 2 tanks for mass distribution
    M_LOX_1 = M_LOX / 2  # 2 tanks for mass distribution

    # LOX Tank
    M_LOX_Tank = 0.00152 * M_LOX_1 + 318
    V_LOX_Tank = M_LOX_1 / rho_LOX
    r_LOX_Tank = (V_LOX_Tank / (4 * pi /3))**(1 / 3)  # radius of LOX tank
    A_LOX_Tank = 4 * pi * (r_LOX_Tank**2)  # Area LOX Tank
    M_LOX_Insu = 1.123 * A_LOX_Tank  # Mass LOX insulation
    M_LOX_Tanks = 2 * M_LOX_Tank
    M_LOX_Insus = 2 * M_LOX_Insu

    # LH2 Tank
    M_LH2_Tank = 0.0694 * M_LH2_1 + 363
    V_LH2_Tank = M_LH2_1 / rho_LH2
    r_LH2_Tank = (V_LH2_Tank / (4 * pi  /3))**(1 / 3)  # radius of LH2 tank
    A_LH2_Tank = 4 * pi * (r_LH2_Tank**2)  # Area LH2 Tank
    M_LH2_Insu = 2.88 * A_LH2_Tank  # Mass LH2 insulation
    M_LH2_Tanks = 2 * M_LH2_Tank
    M_LH2_Insus = 2 * M_LH2_Insu



    return M_prop, M_LOX_Tanks, M_LH2_Tanks, M_LOX_Insus, M_LH2_Insus, r_LOX_Tank, r_LH2_Tank


# computation of structure mass
def structure(M_inter):
```

```
    y = -M_tot/2500000 + 0.067

    M_Struct = M_tot * y

    # opens the .geo file and writes the new radius
    with open("test.geo", "r+") as f:
        f.seek(0) # rewind
        f.write("r_LOX_Tank = %s;\nr_LH2_Tank = %s; \n" % (round(
    r_LOX_Tank,2),round(r_LH2_Tank,2))) # write the new line before
        f.close()

    # mesh
    sp.call([GMSH, File, "-1", "-2"], shell=True)


    return M_Struct



##############################################################

###### main


# Open and read original .geo file in order to keep the original
with open("test.geo", "r+") as f:
     f.read() # read everything in the file
     f.close()


# Masses computation loop
while Epsilon > 1 :

    i = i + 1

    save = M_tot

    # calling function propellant_and_tank
    M_prop, M_LOX_Tanks, M_LH2_Tanks, M_LOX_Insus, M_LH2_Insus,
    r_LOX_Tank, r_LH2_Tank = propellant_and_tank(deltav, M_tot, M_inert
    )

    # Intermediary mass for structure mass computation
    M_inter = RCS + Payload + M_prop + M_LOX_Tanks + M_LOX_Insus +
    M_LH2_Tanks + M_LH2_Insus

    # calling function structure
    M_Struct = structure(M_inter)
```

```

    # New inert mass
    M_inert = RCS + Payload + M_Struct + M_LOX_Tanks + M_LOX_Insus +
   M_LH2_Tanks + M_LH2_Insus

    # Total Mass
    M_tot = M_inert + M_prop

    Epsilon = M_tot - save




# Printing all the values
print('\n')
print('number of iterations :', i)
print('\n')
print('Inert mass :', M_inert)
print('Structure mass :', M_Struct)
print('Propellant mass :', M_prop)
print('LOX tanks mass :', M_LOX_Tanks)
print('LOX insulation mass :', M_LOX_Insus)
print('LH2 tanks mass :', M_LH2_Tanks)
print('LH2 insulation mass :', M_LH2_Insus)
print('Total mass :', M_tot)
print('\n')
print('LOX tank radius', r_LOX_Tank)
print('LH2 tank radius', r_LH2_Tank)
```

A.2 .geo code

```
// ——— modeling ———

Delete Model;

x = 0;
y = 0;
z = 0;
j = 0;



// ——— Tanks ———


Mesh.Algorithm = 6;

For ( 1 : 4 )

j = j + 1;


If ( j == 1 )  // 1st Tank
r = r_LOX_Tank;
sh1 = 2.55; // shift on x axis
sh2 = 0;     // shift on y axis
sh3 = - 0.3;
EndIf

If ( j == 2 )   // 2nd Tank
r = r_LOX_Tank;
sh1 = -2.55;
sh2 = 0;
sh3 = - 0.3;
EndIf

If ( j == 3 )  // 3rd Tank
r = r_LH2_Tank;
sh1 = 0;
sh2 = 2.55;
sh3 = - 0.1;
EndIf

If ( j == 4 )  // 4th Tank
r = r_LH2_Tank;
sh1 = 0;
sh2 = -2.55;
sh3 = - 0.1;
```

```
EndIf

lc = .55;
Point(1+x) = {sh1,sh2,sh3 - r,lc};
Point(2+x) = {r+sh1,sh2,sh3 - r,lc};
Point(3+x) = {sh1,r+sh2,sh3 - r,lc};
Circle(1+y) = {2+x,1+x,3+x};
Point(4+x) = {-r+sh1,sh2,sh3- r,lc};
Point(5+x) = {sh1,-r+sh2,sh3 - r,lc};
Circle(2+y) = {3+x,1+x,4+x};
Circle(3+y) = {4+x,1+x,5+x};
Circle(4+y) = {5+x,1+x,2+x};
Point(6+x) = {sh1,sh2,sh3,lc};
Point(7+x) = {sh1,sh2,-2*r + sh3,lc};
Circle(5+y) = {3+x,1+x,6+x};
Circle(6+y) = {6+x,1+x,5+x};
Circle(7+y) = {5+x,1+x,7+x};
Circle(8+y) = {7+x,1+x,3+x};
Circle(9+y) = {2+x,1+x,7+x};
Circle(10+y) = {7+x,1+x,4+x};
Circle(11+y) = {4+x,1+x,6+x};
Circle(12+y) = {6+x,1+x,2+x};
Curve Loop(13+y) = {2+y,8+y,-10-y};
Surface(14+y) = {13+y};
Curve Loop(15+y) = {10+y,3+y,7+y};
Surface(16+y) = {15+y};
Curve Loop(17+y) = {-8-y,-9-y,1+y};
Surface(18+y) = {17+y};
Curve Loop(19+y) = {-11-y,-2-y,5+y};
Surface(20+y) = {19+y};
Curve Loop(21+y) = {-5-y,-12-y,-1-y};
Surface(22+y) = {21+y};
Curve Loop(23+y) = {-3-y,11+y,6+y};
Surface(24+y) = {23+y};
Curve Loop(25+y) = {-7-y,4+y,9+y};
Surface(26+y) = {25+y};
Curve Loop(27+y) = {-4-y,12+y,-6-y};
Surface(28+y) = {27+y};


x = 7*j;
y = 30*j;
z = j;

EndFor


// -------- Cylinder --------

```

```
Point(37) = {2., 0, 0, 1.0};  Point(38) = {2., 0, 2.6, 1.0};
Point(39) = {-2., 0, 0, 1.0}; Point(40) = {0, -2., 2.6, 1.0};
Point(41) = {0, 2., 2.6, 1.0};  Point(42) = {0, 2., 0, 1.0};
Point(43) = {0, -2., 0, 1.0}; Point(44) = {-2., 0, 2.6, 1.0};

Line(113) = {38, 37}; Line(114) = {40, 43};
Line(115) = {41, 42}; Line(116) = {44, 39};

Point(45) = {0, 0, 0, 1.0};
Point(46) = {0, 0, 2.6, 1.0};
Circle(117) = {40, 46, 38}; Circle(118) = {38, 46, 41};
Circle(119) = {41, 46, 44}; Circle(120) = {44, 46, 40};
Circle(121) = {37, 45, 42}; Circle(122) = {42, 45, 39};
Circle(123) = {39, 45, 43}; Circle(124) = {43, 45, 37};

Curve Loop(120) = {118, 115, -121, -113};
Surface(120) = {120};
Curve Loop(121) = {117, 113, -124, -114};
Surface(121) = {121};
Curve Loop(122) = {123, -114, -120, 116};
Surface(122) = {122};
Curve Loop(123) = {122, -116, -119, 115};
Surface(123) = {123};
Curve Loop(125) = {119, 120, 117, 118};
Plane Surface(125) = {125};

// Meshing Cylinder

Transfinite Line{113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123,
    124} = 8;
Transfinite Surface{120, 121, 122, 123};
Recombine Surface{120, 121, 122, 123};


// ------- Platform -------------------

Point(29) = {1.5, 3.6, 0, 1.0};      Point(30) = {-1.5, 3.6, 0, 1.0};
Point(31) = {-1.5, -3.6, 0, 1.0};    Point(32) = {1.5, -3.6, 0, 1.0};
Point(33) = {3.6, -1.5, 0, 1.0};     Point(34) = {3.6, 1.5, 0, 1.0};
Point(35) = {-3.6, 1.5, 0, 1.0};     Point(36) = {-3.6, -1.5, 0, 1.0};

Line(141) = {36, 31}; Line(142) = {32, 31};
Line(143) = {32, 33}; Line(144) = {33, 34};
Line(145) = {34, 29}; Line(146) = {29, 30};
Line(147) = {30, 35}; Line(148) = {35, 36};

// middle reinforcement
Point(59) = {0, 1, 0, 1.0}; Point(60) = {0, -1, 0, 1.0};
Point(61) = {1, 0, 0, 1.0}; Point(62) = {-1, 0, 0, 1.0};
```

```
Circle(125) = {59, 45, 61}; Circle(126) = {61, 45, 60};
Circle(127) = {60, 45, 62}; Circle(128) = {62, 45, 59};

Curve Loop(126) = {145, 146, 147, 148, 141, -142, 143, 144};
Curve Loop(127) = {121, 122, 123, 124};
Plane Surface(126) = {126, 127};
Curve Loop(128) = {127, 128, 125, 126};
Plane Surface(127) = {127, 128};
Plane Surface(128) = {128};

// ---------- structure ----------

Point(51) = { 4.6 , 4.6, -5, 1.0};
Point(52) = { 4.6 , -4.6 , -5, 1.0};
Point(53) = { -4.6 , 4.6, -5, 1.0};
Point(54) = { -4.6 , -4.6 , -5, 1.0};

Point(55) = { 3.2, 3.2, - 1, 1.0};
Point(56) = { -3.2, 3.2, - 1, 1.0};
Point(57) = { 3.2, -3.2, - 1, 1.0};
Point(58) = { -3.2, -3.2, - 1, 1.0};

Point(67) = { 3.6, -1.5, -3., 1.0}; Point(71) = { -3.6, -1.5, -3.,
    1.0};
Point(68) = { 3.6, 1.5, -3., 1.0};  Point(72) = { -3.6, 1.5, -3., 1.0};
Point(69) = { 1.5, 3.6, -3., 1.0};  Point(73) = { -1.5, 3.6, -3., 1.0};
Point(70) = { 1.5, -3.6, -3., 1.0}; Point(74) = { -1.5, -3.6, -3.,
    1.0};

Point(75) = { 1.5, 1.5, -3, 1.0}; Point(76) = { -1.5, 1.5, -3, 1.0};
Point(77) = { 1.5, -1.5, -3, 1.0};  Point(78) = { -1.5, -1.5, -3, 1.0};

// Legs

Line(149) = {35, 56}; Line(150) = {30, 56};
Line(151) = {56, 53}; Line(153) = {53, 72};
Line(154) = {53, 73}; Line(173) = {56, 72};
Line(174) = {56, 73};

Line(155) = {36, 58}; Line(156) = {58, 31};
Line(157) = {58, 54}; Line(159) = {54, 74};
Line(160) = {54, 71}; Line(175) = {58, 74};
Line(176) = {58, 71};

Line(161) = {32, 57}; Line(162) = {57, 33};
Line(163) = {57, 52}; Line(165) = {52, 67};
Line(166) = {52, 70}; Line(177) = {57, 70};
Line(178) = {57, 67};
```

```
Line(167) = {34, 55}; Line(168) = {55, 29};
Line(169) = {55, 51}; Line(171) = {51, 69};
Line(172) = {51, 68}; Line(179) = {55, 68};
Line(180) = {55, 69};

// Substructure

Line(181) = {69, 73}; Line(182) = {73, 72};
Line(183) = {72, 71}; Line(184) = {71, 74};
Line(185) = {74, 70}; Line(186) = {70, 67};
Line(187) = {67, 68}; Line(188) = {68, 69};
Line(189) = {59, 75}; Line(190) = {61, 75};
Line(191) = {59, 76}; Line(192) = {76, 62};
Line(193) = {62, 78}; Line(194) = {78, 60};
Line(195) = {77, 60}; Line(196) = {77, 61};
Line(197) = {72, 14}; Line(198) = {76, 14};
Line(199) = {14, 78}; Line(200) = {14, 71};
Line(201) = {71, 78}; Line(202) = {76, 72};
Line(203) = {76, 78}; Line(204) = {78, 77};
Line(205) = {77, 75}; Line(206) = {75, 76};
Line(207) = {75, 21}; Line(208) = {21, 69};
Line(209) = {21, 73}; Line(210) = {76, 21};
Line(211) = {75, 69}; Line(212) = {76, 73};
Line(213) = {75, 68}; Line(214) = {77, 67};
Line(215) = {67, 7};  Line(216) = {7, 68};
Line(217) = {7, 75};  Line(218) = {7, 77};
Line(219) = {77, 70}; Line(220) = {78, 74};
Line(221) = {74, 28}; Line(222) = {28, 78};
Line(223) = {28, 70}; Line(224) = {28, 77};
Line(225) = {31, 74}; Line(226) = {70, 32};
Line(227) = {33, 67}; Line(228) = {68, 34};
Line(229) = {29, 69}; Line(230) = {30, 73};
Line(231) = {35, 72}; Line(232) = {36, 71};

Line(233) = {29, 20}; Line(234) = {20, 42};
Line(235) = {20, 30};

Line(236) = {39, 13}; Line(237) = {13, 35};
Line(238) = {36, 13};

Line(239) = {43, 27}; Line(240) = {27, 31};
Line(241) = {32, 27};

Line(242) = {37, 6};  Line(243) = {6, 33};
Line(244) = {34, 6};


Curve Loop(129) = {145, 229, -188, 228};
```

```
Plane Surface(129) = {129};
Curve Loop(130) = {147, 231, -182, -230};
Plane Surface(130) = {130};
Curve Loop(131) = {141, 225, -184, -232};
Plane Surface(131) = {131};
Curve Loop(132) = {226, 143, 227, -186};
Plane Surface(132) = {132};
Curve Loop(133) = {185, -219, -204, 220};
Plane Surface(133) = {133};
Curve Loop(134) = {186, -214, 219};
Plane Surface(134) = {134};
Curve Loop(135) = {187, -213, -205, 214};
Plane Surface(135) = {135};
Curve Loop(136) = {188, -211, 213};
Plane Surface(136) = {136};
Curve Loop(137) = {206, 212, -181, -211};
Plane Surface(137) = {137};
Curve Loop(138) = {202, -182, -212};
Plane Surface(138) = {138};
Curve Loop(139) = {203, -201, -183, -202};
Plane Surface(139) = {139};
Curve Loop(140) = {220, -184, 201};
Plane Surface(140) = {140};

Transfinite Line{145, 229, -188, 228,147, 231, -182, -230,141, 225,
    -184, -232,226, 143, 227, -186} = 6;
Transfinite Surface{129, 130, 131, 132};
Recombine Surface{129, 130, 131, 132};

Transfinite Line{185, -219, -204, 220, 187, -213, -205, 214, 206, 212,
    -181, -211, 203, -201, -183, -202} = 6;
Transfinite Surface{133, 135, 137, 139};
Recombine Surface{133, 135, 137, 139};

// Pads

Point(79) = {4.6, - 5.1, -5}; Point(80) = {4.6, - 4.1, -5};
Point(81) = {-4.6, - 5.1, -5};  Point(82) = {-4.6, - 4.1, -5};
Point(83) = {4.6,  5.1, -5};  Point(84) = {4.6,  4.1, -5};
Point(85) = {-4.6,  5.1, -5}; Point(86) = {-4.6,  4.1, -5};

Circle(245) = {79, 52, 80}; Circle(246) = {80, 52, 79};
Circle(247) = {81, 54, 82}; Circle(248) = {82, 54, 81};
Circle(249) = {83, 51, 84}; Circle(250) = {84, 51, 83};
Circle(251) = {85, 53, 86}; Circle(252) = {86, 53, 85};

Curve Loop(141) = {246, 245};
Plane Surface(141) = {141};
Curve Loop(142) = {247, 248};
```

```
Plane Surface(142) = {142};
Curve Loop(143) = {252, 251};
Plane Surface(143) = {143};
Curve Loop(144) = {249, 250};
Plane Surface(144) = {144};
//+
Physical Curve("MStruts") = {157, 163, 169, 151};
//+
Physical Curve("SStruts") = {165, 166, 178, 162, 161, 177, 167, 179,
    168, 180, 172, 171, 154, 153, 173, 174, 150, 149, 160, 159, 175,
    176, 155, 156};
//+
Physical Surface("Pads") = {142, 141, 144, 143};
//+
Physical Surface("Fuse") = {123, 120, 121, 122, 125};
//+
Physical Surface("Plat1") = {126};
//+
Physical Surface("Plat2") = {127};
//+
Physical Surface("Plat3") = {128};
//+
Physical Surface("Platside") = {132, 129, 130, 131};
//+
Physical Surface("Plattr") = {134, 140, 138, 136};
//+
Physical Surface("Platbot") = {133, 139, 137, 135};
//+
Physical Curve("Struct") = {191, 189, 192, 193, 194, 195, 196, 190,
    233, 235, 234, 244, 243, 242, 241, 240, 239, 237, 236, 238, 197,
    198, 200, 199, 221, 222, 223, 224, 218, 215, 216, 217, 207, 210,
    208, 209};
//+
Physical Surface("LOX") = {52, 44, 50, 54, 46, 56, 58, 48, 28, 14, 24,
    18, 26, 22, 20, 16};
//+
Physical Surface("LH2") = {78, 86, 76, 74, 80, 84, 82, 88, 118, 106,
    116, 108, 112, 110, 104, 114};
```