Aero 5 ELSS - Project - IPSA

Launcher Conception Project

Authors :
M. Chris De Claverie
M. Alexis Rolin
M. Guillaume Truong-Allié

Teacher :
M. Josselin Desmars

Version 1.0.0, 8 décembre 2020

1 Introduction, Mission choice

You are part of IPSA Rocket Company (IRC) working on the launcher conception departement involvedin the development of launcher for space agencies. Your work is to design a launcher for one of specific missions described in 1. The design consists in determining the optimal staging of the launcher to make the mission a success :

- *the number of stages*
- *the appropriate propellant type*
- *the distribution of masses in each stage*
- *the azimuth*

The project needs to be approached for a single mission that must be chosen from the table below.

	Mission 1	**Mission 2**	**Mission 3**	**Mission 4**
Client	Roscosmos	Military	ESA	NASA
Injection/Perigee altitude (km)	410	340	200	567
Apogee altitude (km)	410	340	35786	567
Orbit inclination (deg)	51.6	90	5.2	97.6
Mass of the payload (kg)	32000	290	3800	1150
Launchpad	Baikonur	Vandenberg	Kourou	Cap Canaveral
Launchpad latitude (deg)	45.6	34.7	5.2	28.5
Difficulty	★★★★	★	★★★	★★

FIGURE 1 – Possible missions and their specifications

Difficulties vary widely, and as a group we decided to take the mission that gives a good balance between complexity and personal interest. As such, **we chose the third mission, ESA GEO satellite**, which has a medium complexity.

2 Approach and algorithm

To solve the problem at hand, we decided to leverage the computer's power by developing a small Python3 script. This script would use a smart brute-force approach to find and optimize all the possible combinations of staging and fuel combinations, in a reproducible and testable way.

The full open-source code can be found on the following GitHub Repository :
https://github.com/pwnorbitals/LauncherConceptionClass

2.1 Universal constants and Mission data

The following constants will be needed for the algorithm :

— g_0, earth gravitational acceleration at z=0 ($g_0 = 9.80665\ m/s^2$)

— R_e, earth radius ($R_e = 6378.137\ km$)

— Ω_e, earth rotation rate ($\Omega_e = \frac{2\pi}{Ts}\ rad/s$ with $Ts = 86164\ s$ the sidereal period)

— μ_e, earth gravitational parameter ($\mu_e = 398600.5\ km^3/s^2$)

The following mission parameters will be needed for the algorithm :

— $Z_p = 200\ km$, perigee altitude

— $Z_a = 35786\ km$, apogee altitude

— $\theta_{deg} = 5.2°$, inclination of the objective orbit (degrees)

— $m_p = 3800\ km$, mass of the payload,

— $\phi_{deg} = 5.2°$, latitude of the launch pad (degrees)

From this basic data, we can deduce the following parameters :

— $R_p = Z_p + R_e = 6578.137\ km$, perigee radius

— $R_a = Z_a + R_e = 42164.137\ km$, apogee radius

— $\theta_{rad} = 0.09\ rad$, inclination of the objective orbit (radians)

— $\phi_{rad} = 0.09\ rad$, latitude of the launch pad (radians)

We have all we need to proceed to the next steps.

2.2 Injection requirements

The injection requirements need the calculation of the following parameters :

— $\tau_{deg} = \arcsin \frac{\cos\theta}{\cos\Phi} = 90°$ and $\tau_{rad} = 1.57\ rad$, launch azimuth (degrees and radians)

— $a = \frac{R_a + R_p}{2} = 24371.14\ km$, semi-major axis
— $V_p = \sqrt{\mu(\frac{2}{R_p} - \frac{1}{a})} = 10238.85\ m/s$, perigee speed
— $V_i = \Omega_e . R_e . \cos\Phi . \sin a = 463.18\ m/s$, initial speed
— $L = 2.452.10^{-3} . Z_p^2 + 1.051 . Z_p + 1387.5 = 1695.78\ m/s$, losses
— $\Delta V_n = V_p - V_i + L = 11471.45\ m/s$, needed delta-v for the mission

2.3 Scenarii

Three different propellant types are available :
— LOX-RP1 : Refined Petroleum 1
— LOX-LK2 : Liquid oxygen - liquid hydrogen
— Solid : Solid-Liquid

With two stages, we have four possible combinations (Stage 1 ; Stage 2) :

— Scenario 1 : LOX-RP1 ; LOX-RP1
— Scenario 2 : LOX-RP1 ; LOX-LK2
— Scenario 3 : Solid ; LOX-RP1
— Scenario 4 : Solid ; LOX-LK2

With three stages, we have eight possible combinations (Stage 1 ; Stage 2 ; Stage 3) :

— Scenario 5 : LOX-RP1 ; LOX-RP1 ; LOX-RP1
— Scenario 6 : LOX-RP1 ; LOX-RP1 ; LOX-LK2
— Scenario 7 : LOX-RP1 ; LOX-LK2 ; LOX-RP1
— Scenario 8 : LOX-RP1 ; LOX-LK2 ; LOX-LK2
— Scenario 9 : Solid ; LOX-RP1 ; LOX-RP1
— Scenario 10 : Solid ; LOX-RP1 ; LOX-LK2
— Scenario 11 : Solid ; LOX-LK2 ; LOX-LK2
— Scenario 12 : Solid ; LOX-LK2 ; LOX-RP1

This gives us a 12-elements list of scenarii to try to optimize.

2.4 N-stages Optimization

The Lagrange multiplier method will be used for stage mass optimization. The method uses the following notations :

— a_i the interstage mass ratios

— b_i the initial/final mass ratios

— M_i the initial masses

— M_e the propellant masses

— M_s the structural masses

The algorithm is described using the following steps :

1. Find the optimal mass ratios by looping until $\Delta V_{obj} - \Delta V_{cur} > threshold$
   (a) Find the mass ratios b_i
   (b) Find the speed changes ΔV_i
   (c) Find total speed change $\Delta V_{tot} = \sum_i \Delta V_i$
   (d) Dichotomize the upper mass ratio :
   If $\Delta V_{obj} < \Delta V_{cur}$, $b_n = b_n + step$ else $b_n = b_n - step$
2. Find the mass ratios a_i
3. Find the initial masses M_i
4. Find the propellant masses M_e
5. Find the structural masses M_s and the total mass $M_{tot} = \sum_i M_i + M_{payload}$

We implemented a Python code to try this algorithm on all possible scenarii. The Python implementation is the following :

```python
def NStages(n, deltaV, Isp, k, bn=5, M_payload=3800, threshold = 0.1, step = 0.00001, itmax = 1000000):

    assert(len(Isp) == n)
    assert(len(k) == n)

    g0 = 9.80665
    Omega = k / (1 + k)
    deltaV_current = 0
    it = 0 # Counts Lagrange multiplier methods iterations

    while abs(deltaV_prop - deltaV_current) > threshold:
        b = np.zeros((n,)); b[n-1] = bn
        for j in range(n-2, -1, -1):
            b[j] =  (1 / Omega[j]) * (1 - ((Isp[j+1] / Isp[j]) * (1 - (Omega[j] * b[j+1]))))
            if b[j] < 0:
                raise RuntimeError("negative b")

        deltaVs = Isp * g0 * np.log(b)
        deltaV_current = np.sum(deltaVs)
        it += 1

        bn += step if deltaV_current < deltaV_prop else -step
        if it > itmax :
            raise RuntimeError("iterations limit")

    a = ((1 + k) / b) - k

    Mi = np.zeros((n+1,)); Mi[n] = M_payload
    for j in range(n-1, -1, -1):
        Mi[j] = Mi[j+1] / a[j]

    Me = ((1 - a) / (1 + k)) * Mi[:n]
    Ms = k * Me
    Mtot = np.sum(Mi)

    return deltaV_current, Mi, Me, Ms, Mtot, it, b, a, Omega
```

2.5 Preliminary checks

In order to have a first validation of the optimization we found, we apply a number of simplified heuristic checks which are listed below :

- First stage structural mass check : $500 < M_{s,0} < 100000$
- Second stage structural mass check : $200 < M_{s,1} < 80000$
- Third stage structural mass (if applicable) : $200 < M_{s,2} < 50000$
- Mass scaling check : $M_{i,0} > M_{i,1} + M_{i,2} + M_{payload}$ and $M_{i,1} > M_{i,2} + M_{payload}$

3 Possible solutions

When running the algorithm, a number of different combinations have been found to reach the desired mission parameters and pass the preliminary checks. We can show that every three-staged scenario has an optimum sizing that respects the specification but none of the two-staged launches is acceptable : the required structural and propellant masses for only two stages exceed the specifications.

The configurations are be detailed in the following table :

Scenario	**n**	**propellants**	b_n	M_{tot} **(kg)**	ΔV_{tot}	**it**
4	3	LOX-RP1 + LOX-RP1 + LOX-RP1	3.634	597 413	11471.36	31700
5	3	LOX-RP1 + LOX-RP1 + LOX-LK2	3.15498	384 650	11471.41	7749
6	3	LOX-RP1 + LOX-LK2 + LOX-RP1	3.26016	384 645	11471.39	13008
7	3	LOX-RP1 + LOX-LK2 + LOX-LK2	2.99998	306 096	11789.98	1
8	3	Solid + LOX-RP1 + LOX-RP1	3.57382	612 729	11471.35	28691
9	3	Solid + LOX-RP1 + LOX-LK2	3.14	397 077	11471.46	7005
10	3	Solid + LOX-LK2 + LOX-LK2	2.99998	313 354	11771.08	1
11	3	Solid + LOX-LK2 + LOX-RP1	3.23272	397 076	11471.40	11636

4 Optimal solution and verification

Among the 8 allowed launches, we identified the most optimized one, this is to say, the scenario that results in the lighter total mass.
This scenario is the scenario using :

— For the first stage : LOX-RP1
— For the second stage : LOX-RP1
— For the third stage : LOX-LH2

As explained earlier, this combination results on 3 Specific Impulses, $Isp_1 = 286\ s$, $Isp_2 = 360\ s$ and $Isp_3 = 440\ s$, and 3 structural index, $k_1 = 0.15$, $k_2 = 0.15$, $k_3 = 0.22$.

Therefore, the output of the Python algorithm after 7749 iterations is :

$$\Omega_1 = 0.1304,\ \Omega_2 = 0.1304,\ \Omega_3 = 0.1803$$

$$b_1 = 2.5999,\ b_2 = 3.2601,\ b_3 = 3.1549$$

$$a_1 = 0.2923,\ a_2 = 0.2027,\ a_3 = 0.1667$$

$$Mi_1 = 384650.4578kg,\ Mi_2 = 112440.5896kg,\ Mi_3 = 22796.7744kg$$

$$m_{e1} = 35505.6349kg,\ m_{e2} = 11692.6715kg,\ m_{e3} = 15571.1266kg$$

$$m_{s1} = 35505.6349kg,\ m_{s2} = 11692.6715kg,\ m_{s3} = 3425.6478kg$$

With a final $\Delta V = 11471.4124\ m/s$, and an initial mass at lift-off of $384650.45\ kg$.

In order to check if our optimal solution works, we used the "LAUNCHER Simulator".

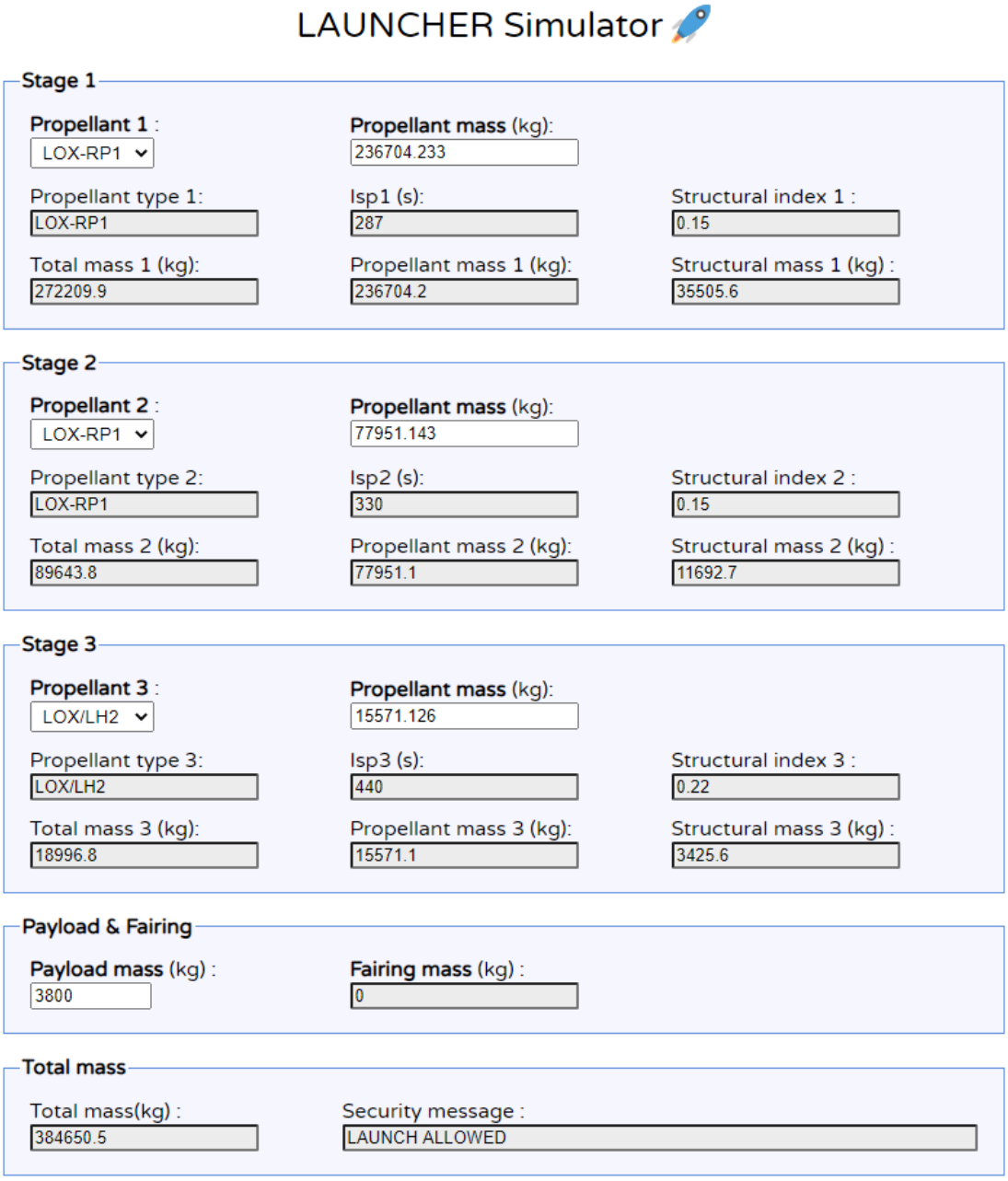

FIGURE 2 – Simulation of the propellant and masses

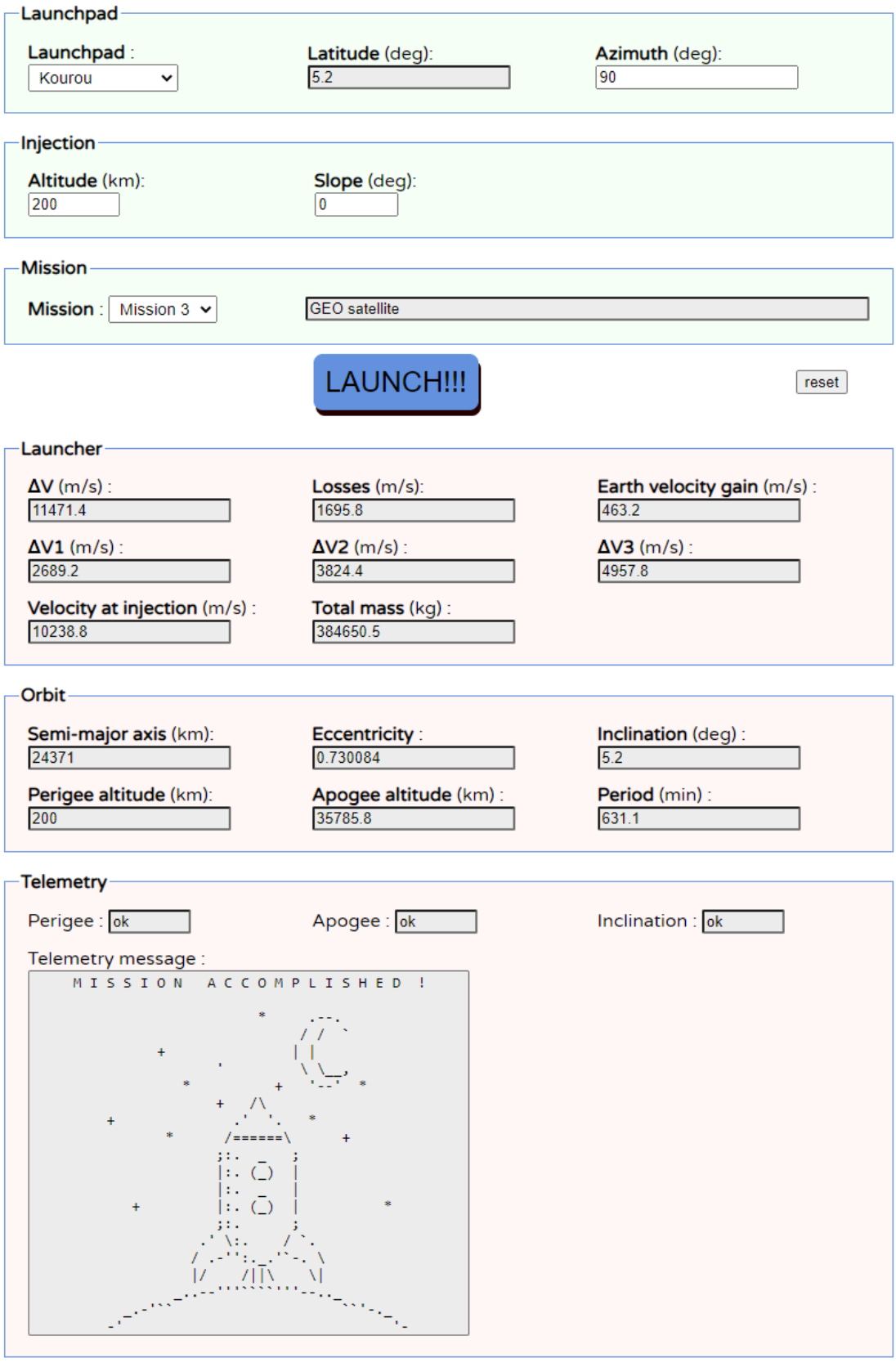

FIGURE 3 – Simulation of the insertion

From this simulation, we can say that our mass, propellant and ΔV parameters are correct.

From what we see and choose, our launcher composed of 3 stages (LOX-RP1, LOX-RP1 and LOX-LH2) can put a geostationary satellite in a GTO with success.

5 Encountered biases and issues

During the project, we encountered some biases which caused some issues.

The first one was the choice of the initial value of b_3. To cover the largest amount of possible situation we chose to start with a low value $b_3 = 1$ but this lead to several issues. Some scenarios were unable to make even the first step of the algorithm or some other were trapped in infinite loop of optimization and other gave negative value for the different masses. Over the iterations changing the initial value of b_3, we saw a slight improvement when b_3 became bigger. We stopped our iterations when we reached $b_3 = 3$ as initial value where almost all the 3-staged scenarios were successful.

Also, we encountered another issues, the step size for the main loop. Indeed, we started with a step of 0.1 but we rapidly saw that this step was clearly to big to be precise. As a first try, we put it a 0.01 and the ΔVs were around $\pm 50 m/s$ of the required ΔV. Then, when we started to check on the "LAUNCHER Simulator", none of our scenarios were able to fit the mission's specifications. This issue lead us to decrease a lot our step size to finally set it at 1.10^{-5} ($b = b \pm 10^{-5}$).

The last bias we faced is in the "LAUNCHER Simulator" itself. Indeed, even if this simulator works perfectly, it doesn't allow a greater ΔV than what is strictly necessary. By that, we mean that if a launcher has a greater propulsive ΔV than the required ΔV to reach the desired injection, it's not a problem. It's a fact, it's not optimum in this case but to have a margin of few tens of m/s of ΔV is a good thing to avoid miss injection running out of fuel. The simulator will burn all the fuel in the tank even it is not necessary, whereas most of the launchers keep a comfortable headroom of ΔV to ensure that the satellite will be correctly injected.

Keeping that in mind, some of our scenarios were rejected due to greater ΔV than necessary. As the scenarios 7 and 10 which have higher final ΔV than necessary and that are rejected by the simulator. Those two work, but are not optimum for our mission.