TECHNICAL REPORT

PROJECT : LAUNCHER CONCEPTION

IPSA ROCKET COMPAGNY

Audrey Duthil
Lucie Michelet
Kim Le Rabo

I. Introduction

As a member of the IPSA Rocket Company (IRC), within the Launcher Conception Department, we are specialized in the development of launcher for space agencies. We were very enthusiast to work on this mission for NASA, and we did our best to answer technical requirements. In this report you will find the description of the mission, all the parameters we used and the technical specifications of the launcher.

The design consists in determining the optimal staging of the launcher to make the mission a success:

- the number of staging;
- the appropriate propellant type;
- the distribution of masses in each stage;
- the azimuth.

To facilitate a thorough understanding, the IRC Numerical Department has engineered a sophisticated tool known as the Launcher Simulator 23. This tool will be harnessed to vividly demonstrate how our design translates into a successful mission. Through the following sections, we aim to provide a transparent and insightful exploration of our design methodology, presenting a compelling case for the effectiveness and viability of our launcher configuration.

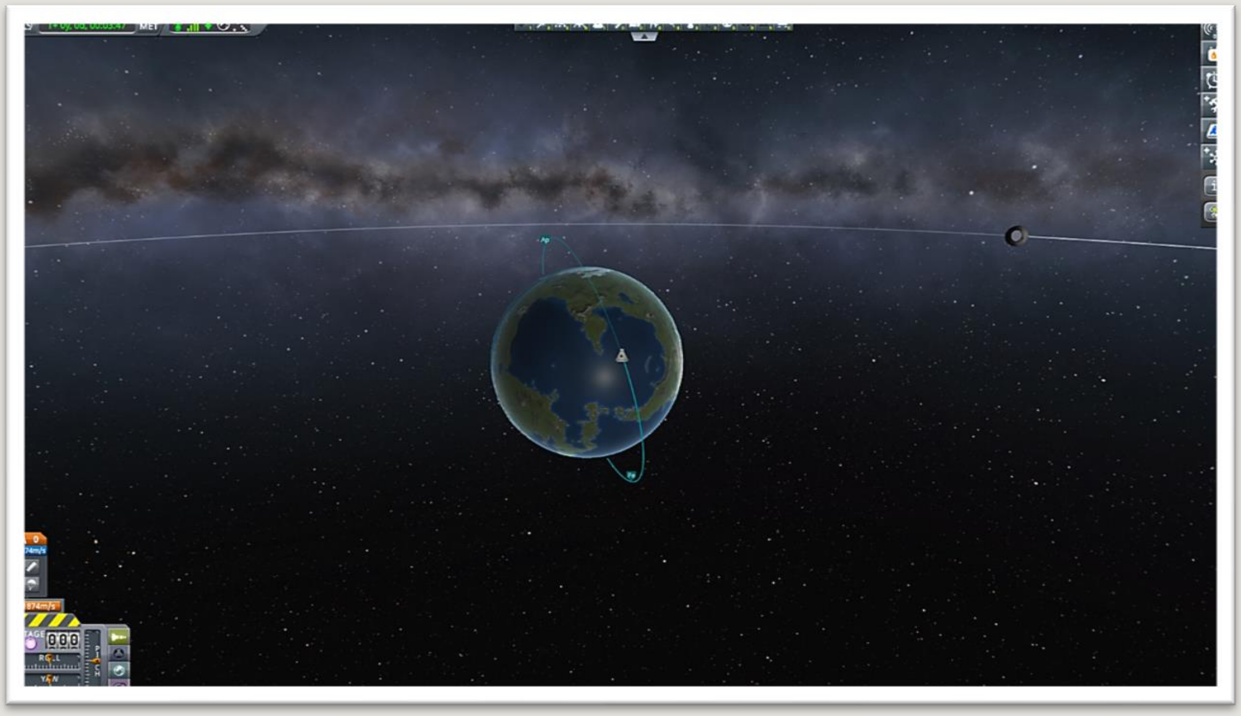
II. Space Launcher Design

Mission description

Mission:
NASA requires a Sun synchronous (SSO) satellite at constant altitude. Here you can find all the parameters of the orbit described by the SSO satellite:

Client	**NASA**
Injection/Perigee altitude (km)	1681
Apogee altitude (km)	1681
Orbit inclination (deg)	103.0
Mass of the payload (kg)	170
Launchpad	Cap Canaveral
Launchpad latitude (deg)	28.5

Visualization of the orbit on Kerbal:



Our goal is to reach a quasi-polar retrograde orbit. As a rocket company, our mission is to design an optimized launcher, design to inject the satellite in its orbit, by using the minimum amount of propellant.

Technical requirements

Propellant
IRC has developed several types of stages depending on the propellant. The characteristics are described in Table 1.

Table 1 – Characteristics of stages/propellant of IRC

Name	Code	possible stages	Isp (vaccum)	Mean Isp	structural index
			s	s	
Solid-Liquid	Solid	1	295	266	0.12
Refined Petroleum 1	LOX-RP1	1,2,3	320	285	0.16
Liquid oxygen - liquid hydrogen	LOX-LK2	2,3	450	-	0.25

Solid propellant can only be used in first stage, whereas LOX-LK2 is only usable for a second or third stage.

Masses
Stage specifications are given in Table 2. Security department does not allow a total mass bigger than 1000 tons.

Table 2 – Stages specifications

	minimum structural mass	maximum structural mass
	kg	kg
Stage 1	500	100 000
Stage 2	200	80 000
Stage 3	200	50 000

To respect equilibrium of the launcher, the total mass of stage 1 should be bigger than the mass of the rest of the launcher (stage 2 + stage 3 + payload), and the total mass of stage 2 should be bigger than the mass of the rest of the launcher (stage 3 + payload). The fairing mass is neglected in the study.

Losses
The losses department of IRC has studied the losses for different combinations of stages and altitude injection. They finally found that the losses mainly depend on the altitude injection. Fig.1 shows the total losses in relation to the injection altitude. They found a relation giving the losses depending on altitude z (in km) valid for the range [200, 1800]:

$$\Delta V_{losses}(m/s) = 2.452 \times 10^{-3} z^2 + 1.051 z + 1387.50$$

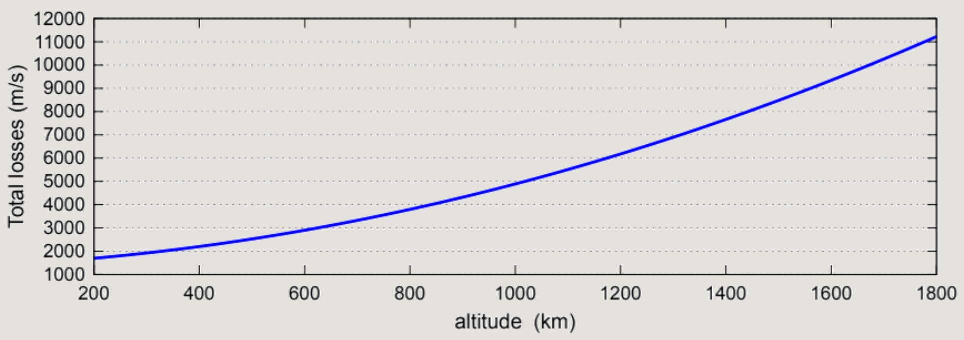


Figure 1 – Total losses related to the altitude injection

III. Simulation

Step One: Calculation of the injection requirement

For this project, we used a Python program to automate and facilitate the calculation. First, we have imported useful libraries. Space Mechanics is a personal library with many constants and functions, related to space mechanics.

```
import webbrowser                            #To open Web pages
import spacemechanics as sp                  #Personal library with useful functions
import numpy as np                           #Numpy library
```

We can compute the initialization of the given data, the calculation of the velocity at injection and azimuth:

```
zp = 1681
rp = sp.Re+zp
za = 1681
ra = sp.Re+za
i = sp.rad(103)
mu = 170
lat = sp.rad(28.5)

#Velocity at injection : vp = np.sqrt(mueE/(z+R))
vp = sp.v_circular(zp)
print("vp :",vp*1000,"m/s")

#launcher azimuth is given by cos(i) = sin(az)*cos(lat)
az = np.arcsin(np.cos(i)/np.cos(lat))
print("azimut :",sp.deg(az),"°")
```

Finally, we obtain:

Velocity at injection: 7032.74 m/s
Azimuth: -14.83°

We can notice that this azimuth corresponds to a launch in the opposite direction of earth rotation.

Step Two: Losses, initial velocity and ΔV requirement

To determine the ΔV required to reach the desired orbit (zp = 1681 km), we calculate the losses encountered and the earth velocity gain Vi, in m/s :

```
vsat = sp.vsat(sp.Re+zp,sp.mueE)*1000

loss = losses(zp)
vi = sp.omgE*sp.Re*np.cos(i)*1000 #m/s
dv = vsat - vi + loss #m/s

print(loss,"m/s\t",vi,"m/s\t",dv,"m/s")
```

We obtain:

Losses = 10082.996972 m/s
Vi = -104.62329663337815 m/s
ΔV = 17220.362573180275 m/s

Losses are high because the altitude of the desired orbit is important. Because we launch against earth rotation direction, the initial earth velocity is a loss for us and induces a highest ΔV. Finally, the ΔV needed is really high and makes this mission challenging!

Step Three: Optimal Stagging using Lagrange Method

For this mission we had the choice between three different propellants, with their own particularities. The ISP-mean is the first value in each list, and the ISP-vacuum is the two last ones. The list K corresponds to the structural index of the propellant in the following order: Solid-Liquid, Refined Petroleum, Liquid Oxygen/Hydrogen.

```
ISP_solid = [266,295,295]     #s      stage 1 only
ISP_petrol = [285,320,320]    #s      stage 1,2,3
ISP_liquid = [0,450,450]      #s      stage 2 and 3

K = [0.12,0.16,0.25]
```

Choose of the propellant:

This mission requires a high ΔV, because of orbit's parameters (retrograde and important losses). That's why, for the first stage, solid propellant is the most suitable. It allows high propellant density and important propellant mass, with a low structural index.

After first stage separation, we need to give thrust during a long duration. That's why, we want the highest ISP for second and third stage. We finally chose liquid Oxygen/Hydrogen propellant for these two stages.

In conclusion, we have the following propellant configuration:

Stage 1 = Solid
Stage 2 = LOX-LK2
Stage 3 = LOX-LK2

Then we applied the Lagrange Method, with different values of b, until the difference between the ΔV required and the ΔV reached, tends to zero.

```
Isp = [ISP_solid[0],ISP_liquid[1],ISP_liquid[2]]
k = [K[0],K[2],K[2]]
n_stage = len(k)

omg = []

#Lagrange multiplier method
for i in range(n_stage):
    omg.append(k[i]/(k[i]+1))
```

```
b3 = 4.0674
b2 = (1/omg[1])*(1-Isp[2]/Isp[1]*(1-omg[2]*b3))
b1 = (1/omg[0])*(1-Isp[1]/Isp[0]*(1-omg[1]*b2))
b = [b1,b2,b3]

Dv = np.zeros(n_stage)
a = np.zeros(n_stage)
dvf = 0

for i in range(n_stage):
    Dv[i] = sp.g*Isp[i]*np.log(b[i])
    a[i] = (1+k[i])/b[i]-k[i]
    dvf += Dv[i]


print("dv :",dv,"\ndvf :",dvf)
```

This method gives us the parameters to compute the optimized propellant and structural mass of each stage of the launcher. Then, we check mass constraints and repartition along stages, to ensure that our launcher is balanced and reliable.

```
Mi,me,ms = mass_calcul(k,mu,a)

mass_check(mu,ms,n_stage)

print("\nMass propellant stage 1 : ",me[0],"kg\nMass propellant stage 2 :
",me[1],"kg\nMass propellant stage 3 : ",me[2],"kg")
```

We used our Python program to compute all the possible combinations of propellant. The one described before is the only one successful for this mission.

RECAPITULATIVE TABLE

STAGE 1: SOLID STAGE 2: LOX-LK2 STAGE 3: LOX-LK2

	Structural mass ms (kg)	Propellant mass me (kg)	Total mass M (kg)	Equilibrium condition (kg)	
STAGE 1	94 660,5	788 837,9	883 498,4	500 < ms1 < 100 000 M2+M3+Mu < M1	✓
STAGE 2	9 754,5	39 018	48 772,5	200 < ms2 < 80 000 M3+Mu < M2	✓
STAGE 3	559,1	2236,6	2 795,7	200 < ms2 < 50 000	✓
TOTAL LAUNCHER	104 974,1	830 092,5	935 236,7 (with payload)	Mtot < 1 000 000	✓

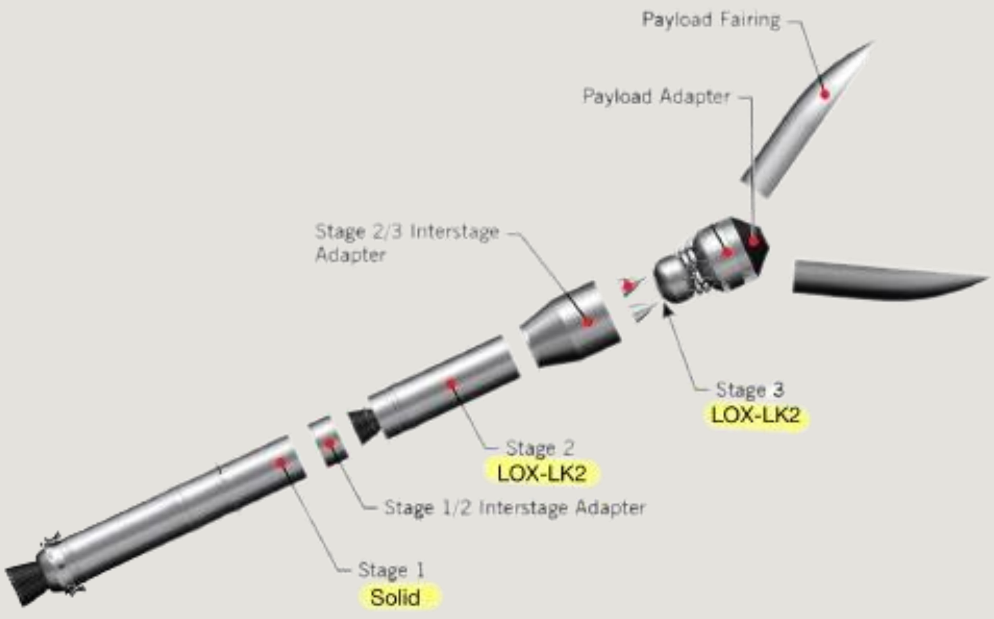


Figure 1:Schematic representation of our launcher

Step Four: Expected Orbit parameters

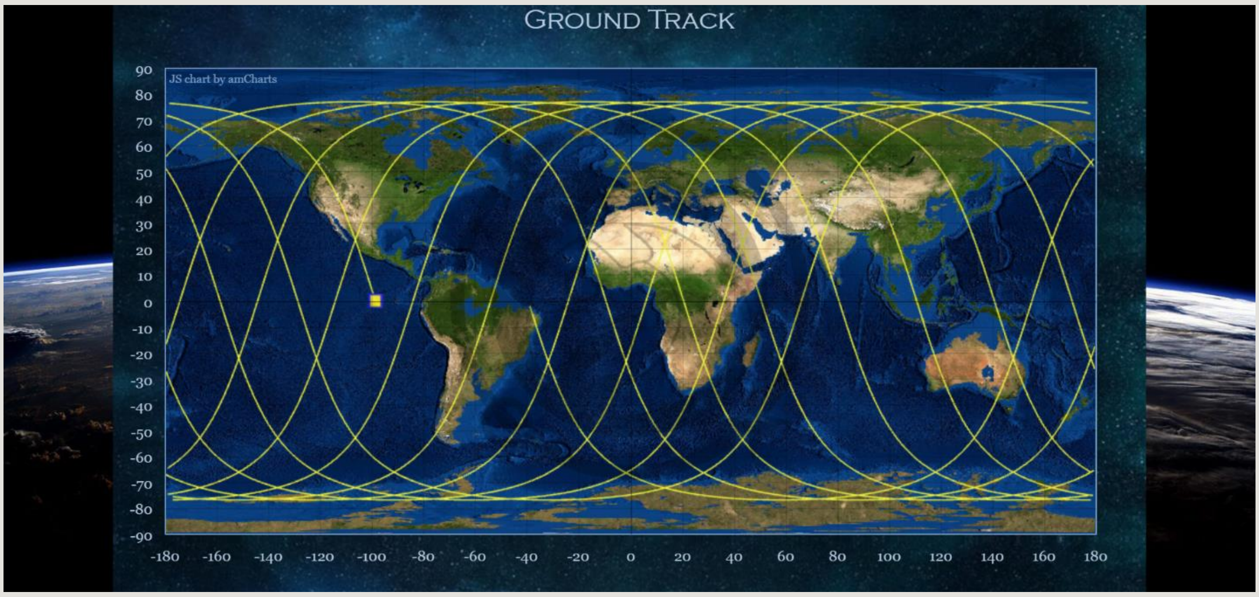
Now that our launcher is design and optimized, we can compute the expected orbit main parameters.

```
semi-major axis : ax
ax = sp.a_z(zp,za)
#eccentricity : circular
e = sp.e_r(rp,ra)
#inclination : i
#apogee et perigee : za and z^p
#Period
t = sp.T(ax)
print("semi-major axis : ",ax,"\neccentricity : ",e,"\nApogee alt :
",za,"km\nPerigee alt : ",zp,"km\nInclination : ",sp.deg(i),"°\nPeriod :
",sp.T_format(t))
```

Finally, we obtain:

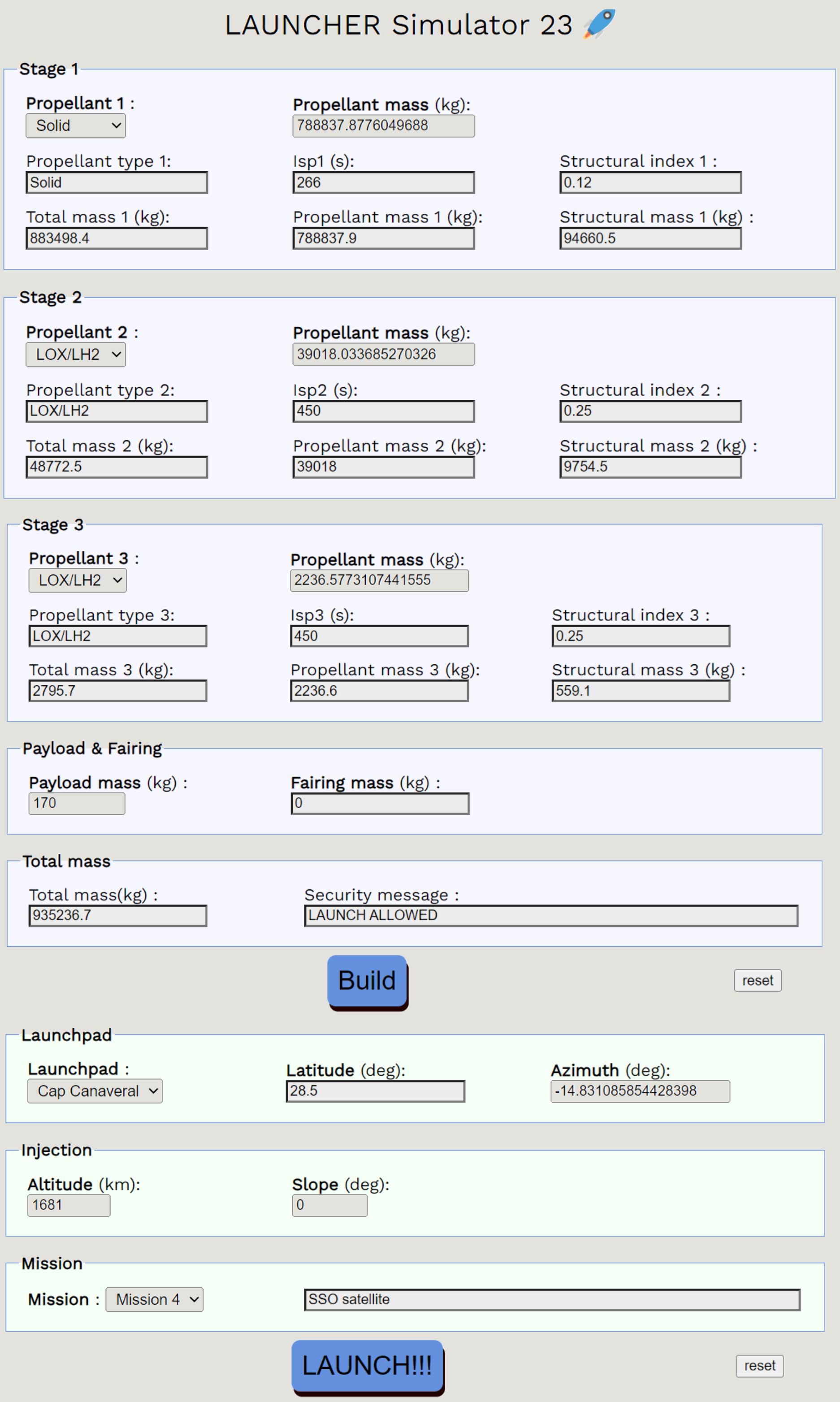
```
Semi-major axis:  8059.135 km
   Eccentricity:  0.0
Apogee altitude:  1681 km
Perigee altitude:  1681 km
   Inclination:  103.0 °
  Period:  02:00:0.184000
```

The launch date is unknown, but we can assume that right ascending node and the argument of perigee are equal to 0°. These assumptions allow us to plot a ground track simulation, which will vary according to launch date, but gives us a good preliminary visualization:



Verification using Simulation Software

We can check our results by using a launcher simulator software. It will ensure our client that our launcher is reliable, and the mission a success.



LAUNCHER Simulator 23

Stage 1

Propellant 1 : Solid	**Propellant mass** (kg): 788837.8776049688	
Propellant type 1: Solid	Isp1 (s): 266	Structural index 1 : 0.12
Total mass 1 (kg): 883498.4	Propellant mass 1 (kg): 788837.9	Structural mass 1 (kg) : 94660.5

Stage 2

Propellant 2 : LOX/LH2	**Propellant mass** (kg): 39018.033685270326	
Propellant type 2: LOX/LH2	Isp2 (s): 450	Structural index 2 : 0.25
Total mass 2 (kg): 48772.5	Propellant mass 2 (kg): 39018	Structural mass 2 (kg) : 9754.5

Stage 3

Propellant 3 : LOX/LH2	**Propellant mass** (kg): 2236.5773107441555	
Propellant type 3: LOX/LH2	Isp3 (s): 450	Structural index 3 : 0.25
Total mass 3 (kg): 2795.7	Propellant mass 3 (kg): 2236.6	Structural mass 3 (kg) : 559.1

Payload & Fairing

Payload mass (kg) : 170	**Fairing mass** (kg) : 0

Total mass

Total mass(kg) : 935236.7	Security message : LAUNCH ALLOWED

Launchpad

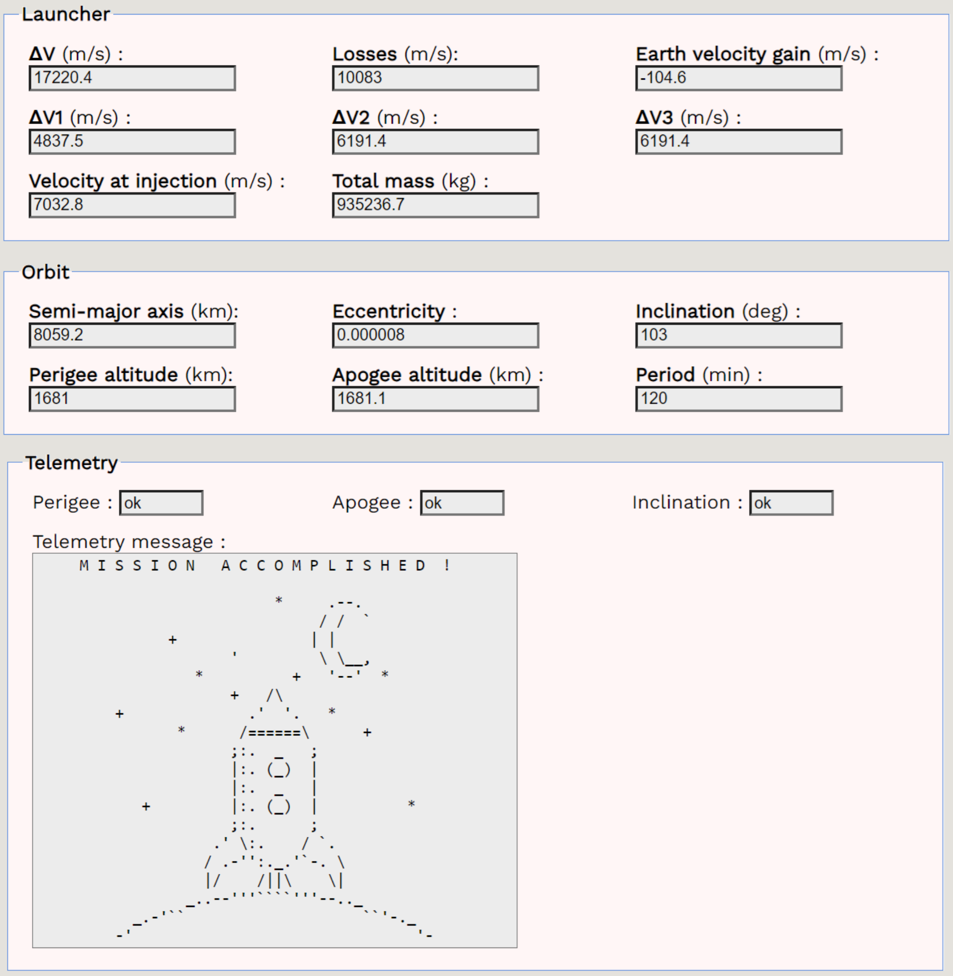
Launchpad : Cap Canaveral	**Latitude** (deg): 28.5	**Azimuth** (deg): -14.831085854428398

Injection

Altitude (km): 1681	**Slope** (deg): 0

Mission

Mission : Mission 4	SSO satellite



Launcher

ΔV (m/s) : 17220.4
Losses (m/s): 10083
Earth velocity gain (m/s) : -104.6
ΔV1 (m/s) : 4837.5
ΔV2 (m/s) : 6191.4
ΔV3 (m/s) : 6191.4
Velocity at injection (m/s) : 7032.8
Total mass (kg) : 935236.7

Orbit

Semi-major axis (km): 8059.2
Eccentricity : 0.000008
Inclination (deg) : 103
Perigee altitude (km): 1681
Apogee altitude (km) : 1681.1
Period (min) : 120

Telemetry

Perigee : ok
Apogee : ok
Inclination : ok

Telemetry message :

```
     M I S S I O N   A C C O M P L I S H E D  !

                           *     .--.
                                / /  `
               +               | |
                      '         \ \__,
                  *          +   '--'  *
                      +   /\
         +              .'  '.   *
                *      /======\      +
                      ;:.  _   ;
                      |:. (_)  |
                      |:.  _   |
            +         |:. (_)  |          *
                      ;:.      ;
                    .' \:.    / `.
                   / .-'':._.'`-. \
                   |/    /||\    \|
                 _..--'''````'''--.._
           _.-'``                    ``'-._
         -'                                '-
```

The software confirms that the mass repartition of the launcher and the choice of propellant are suitable for the mission. The satellite can reach its orbit, with the good parameters.

IV. Conclusion

In conclusion, we have designed a launcher to reach a quasi-polar retrograde orbit for NASA. Our launcher is **efficient, reliable, and optimized**. We chose to use **three stages**, with **solid propellant** for the first one and **liquid oxygen / hydrogen** for the others. The optimization process uses the Lagrange Method and returns the optimal mass configuration. Finally, it allows us to reach the delta V required while ensuring launcher equilibrium. The payload is well released in its orbit, to accomplish its mission.

Our launcher is your solution for your mission, let's launch together!

Annexes

```
#%% Bibliothèque
def mass_check(mu,m,nb_stage):
    #Structural mass per stage (kg)
    min_ms = [500,200,200]
    max_ms = [100000,80000,50000]
    res = True

    #Mass in intervals per stage
    for i in range(nb_stage):
        if m[i] > min_ms[i] and m[i]<max_ms[i]:
            res = True
        else :
            res = False
            print("Stage ",i+1,"mass not in the correct interval :
",m[i],"kg")
            break

    if res == True :
        #tot mass > tot mass - stage(i)
        for i in range(1,nb_stage):
            if m[i-1]>m[i]:
                #print("Stage ",i+1,"to",i+1,":",m[0],">",m[1])
                res = True
            else :
                res = False

        #tOT MASS > 1000 tons
        if m[0] > 1000000:
            res = False
            print("Total mass to heavy : ",m[0],"kg > 1000 000 kg")

        if res == True:
            print("Mass validated !")
    else :
        print("Mass not validated.")

    #Ajouter le check pour un stage = 1 ou 3

def losses(z):
    #m/s
    if z > 200 and z < 1800:
        result = 2.452*(10**-3)*z**2+1.051*z+1387.5
        return result
    else :
        print("The altitude z is not in the range [200,800]. Please try
again.")


def mass_calcul(k,mu,a):
    if len(k)== 3 :
        Mi = [0,0,mu/a[2]]
        Mi[1] = Mi[2]/a[1]
        Mi[0] = Mi[1]/a[0]

    if len(k) == 2:
        Mi = [0,mu/a[1]]
        Mi[0] = Mi[1]/a[0]
    me = np.zeros(n_stage)
    ms = np.zeros(n_stage)

    for j in range(n_stage):
       me[j] = ((1-a[j])/(1+k[j]))*Mi[j]
       ms[j] = k[j]*me[j]

    return Mi,me,ms
```