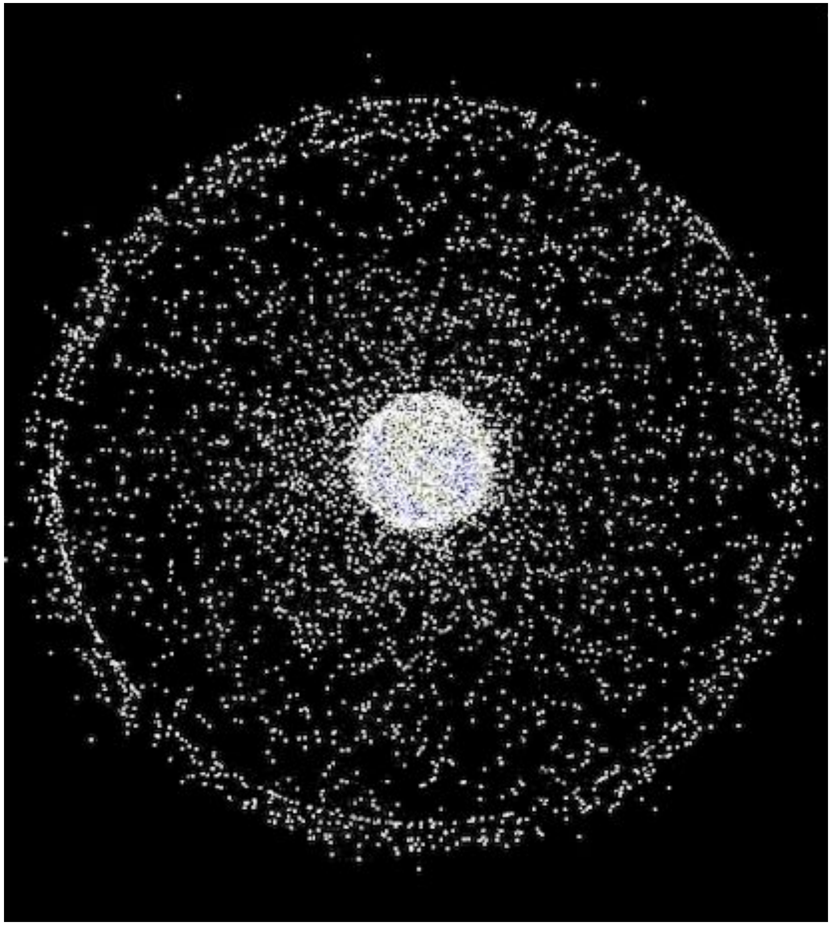
Using a solar sail to deorbit satellites in low-Earth orbit

Bouchaira Neirouz
August 2021

Introduction

- Space pollution
- Threat to scientific research
- Societal issues
- De-orbiting methods
- Solar sails



Problematic

How can solar sails be used to reduce the time satellites spend in the atmosphere?

Table of contents

Satellite trajectory

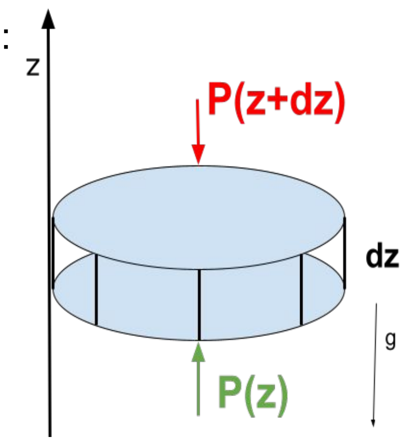
Model

- Geocentric reference frame assumed to be Galilean
- polar coordinates (plane trajectory)
- Assumptions:
  - ➢ Undeformable solid
  - ➢ Constant mass
  - ➢ Thermosphere 80 km-700 km
  - ➢ Average temperature: 0°C

<u>Exponential law of air density:</u>

Hydrostatic equilibrium :

$$\frac{dp}{dz} + \rho g = 0$$



$$\rho = \frac{P_0 M}{RT} e^{-\frac{z}{H_0}} \quad \text{with} \quad H_0 = \frac{R\,T_0}{Mg}$$

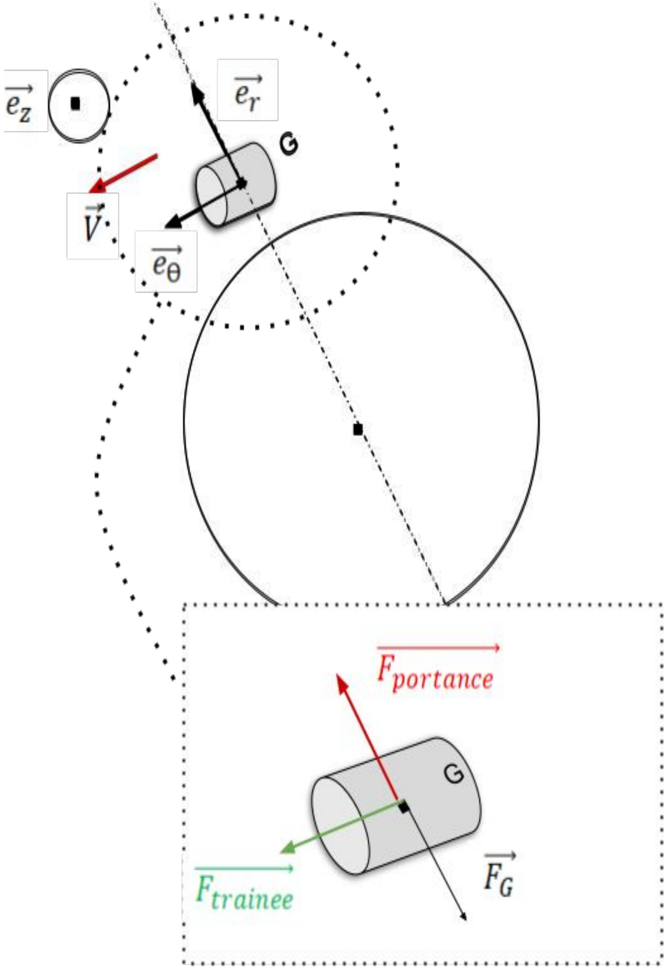
Equations

Force balance:

Lift force: $$\overrightarrow{F_{portance}} = \frac{1}{2}\,\rho V\,S\,C_p\overrightarrow{V^{\perp}}$$

Drag force: $$\overrightarrow{F_{trainee}} = -\,\frac{1}{2}\,\rho V\,S\,C_t\vec{V}$$

Gravitational attraction: $$\overrightarrow{F_G} = -\frac{G\,M_t m}{r^2}\overrightarrow{e_r}$$

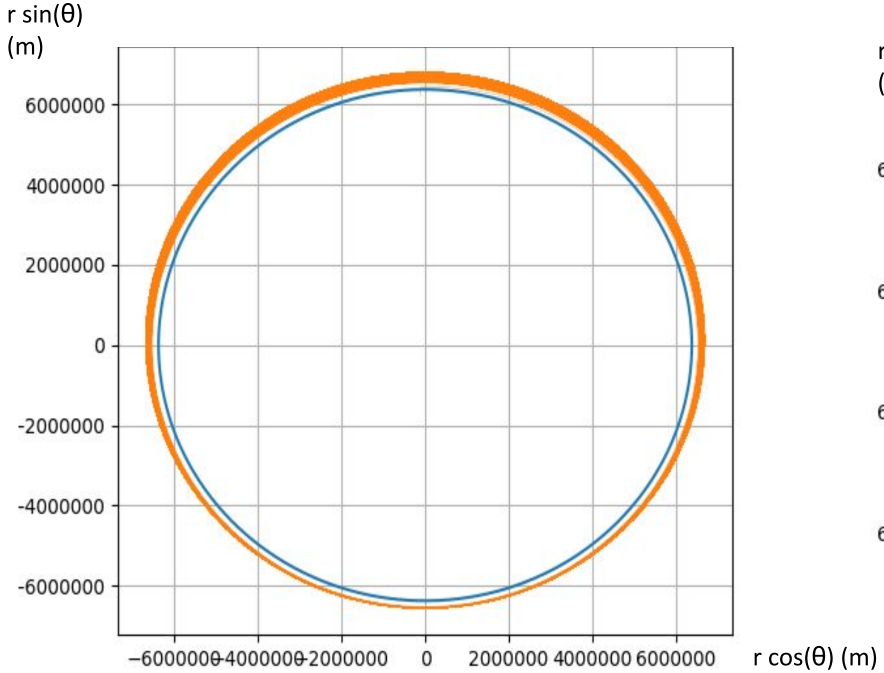


Fundamental relationship of dynamics:

$$m\,\frac{d\vec{V}}{dt} = \frac{1}{2}\,\rho V\,S\,C_p\overrightarrow{V^{\perp}} - \frac{1}{2}\,\rho V\,S\,C_t\vec{V} - \frac{G\,M_t m}{r^2}\overrightarrow{e_r}$$

$$\begin{cases} m\left(\ddot{r} - r\dot{\theta}^2\right) = \frac{1}{2}\,\rho V\,S\,(C_p r\dot{\theta} - C_t \dot{r}\,) - \frac{G\,M_t m}{r^2} \\ m\left(2\dot{r}\dot{\theta} + r\ddot{\theta}\right) = -\frac{1}{2}\,\rho V\,S\,(C_t r\dot{\theta} - C_p \dot{r}\,) \end{cases}$$

with $V = \sqrt{\dot{r}^2 + r^2\dot{\theta}^2}$



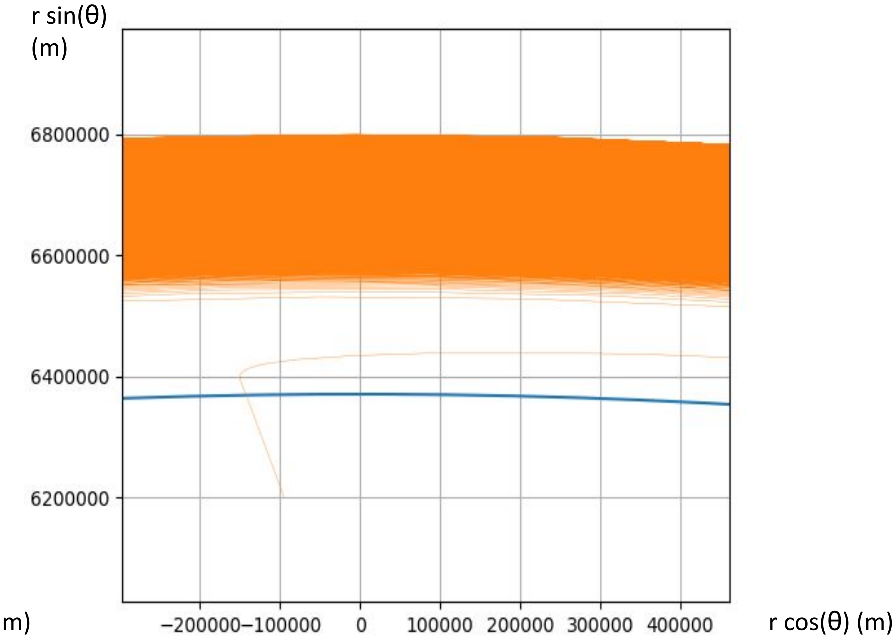


Figure 1: Satellite trajectory subjected to Python friction forces

Solar sails

Principle

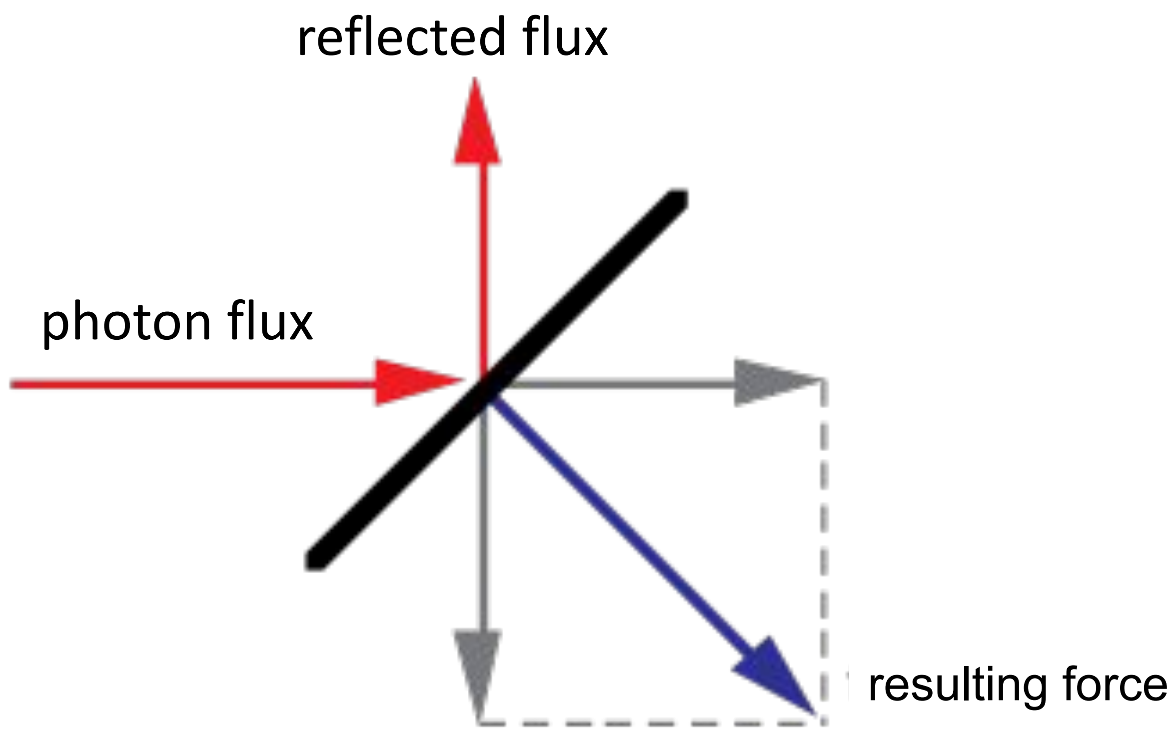
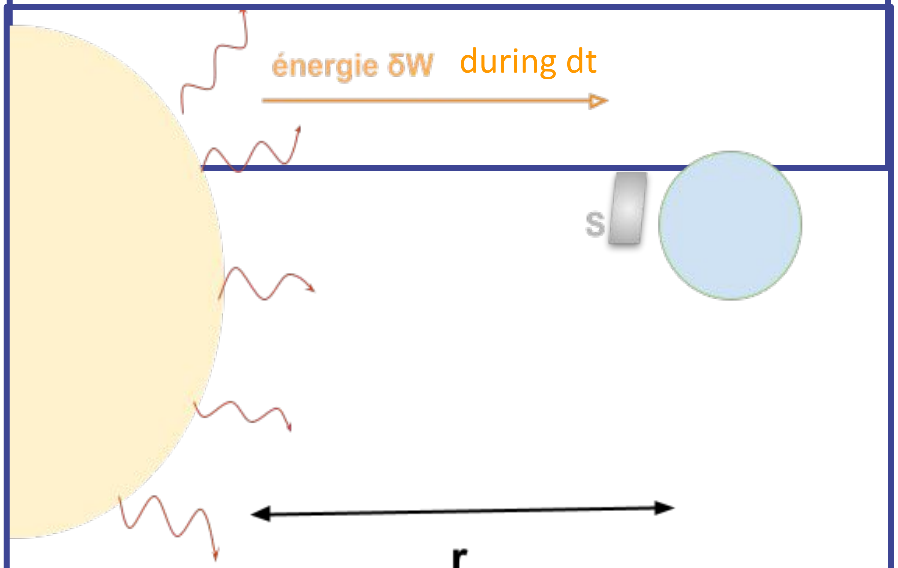


Figure 2: Thrust force of solar radiation

Modeling the radiation force

Where does the power emitted by the sun come from?

$P_0 = 4\pi R_s^2 \sigma T^4 \longrightarrow P_0 = 3{,}85.10^{26}\,\mathrm{W}$



F radial repulsion, due to radiation pressure

$$\vec{F} = \frac{1}{c}\frac{\partial W}{\partial t}\overrightarrow{e_r}$$

The power emitted by the sun is distributed over a sphere of radius r

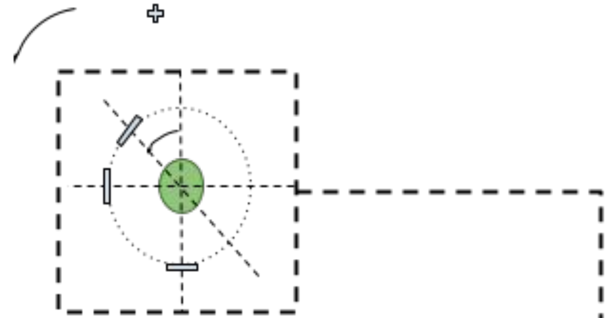
$$\delta W = P_0 \frac{S}{4\pi r^2}dt \quad \text{donc } F = P_0 \frac{S}{4\pi c r^2}$$

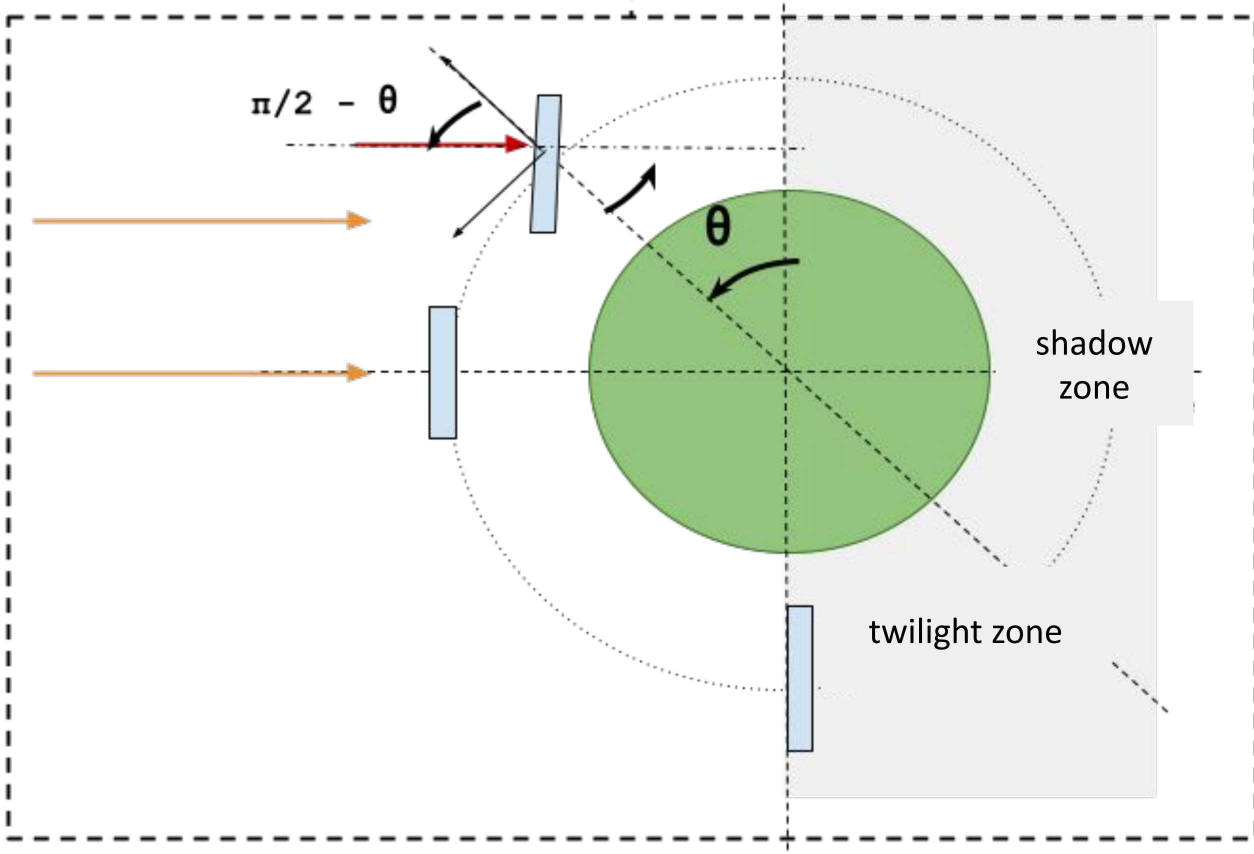
Numerical application:

for a sail model similar to Nasa's Sunjammer S=1200m² , we find: $F = 10^{-2}\,\mathrm{N}$

$$\vec{F}=-F\cos\left(\frac{\pi}{2}-\Theta\right)\overrightarrow{e_r}-F\sin\left(\frac{\pi}{2}-\Theta\right)\overrightarrow{e_\theta}$$

so: $$\vec{F}=-F\sin(\Theta)\,\overrightarrow{e_r}-F\cos(\Theta)\,\overrightarrow{e_\theta}$$





Application

Application to a concrete case

Material:: NanoSail-D2

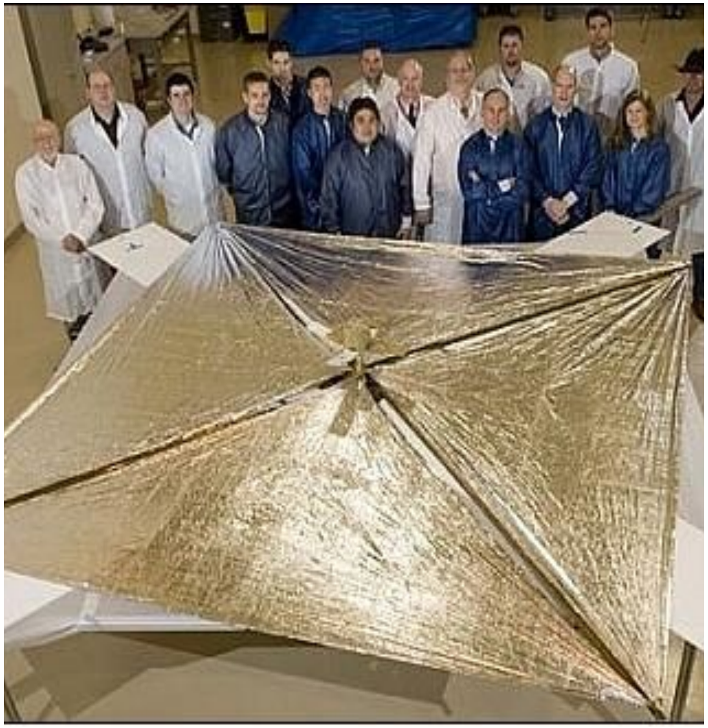


Figure 3:NanoSail-D2 ground deployment test (Source: NASA/MSFC)

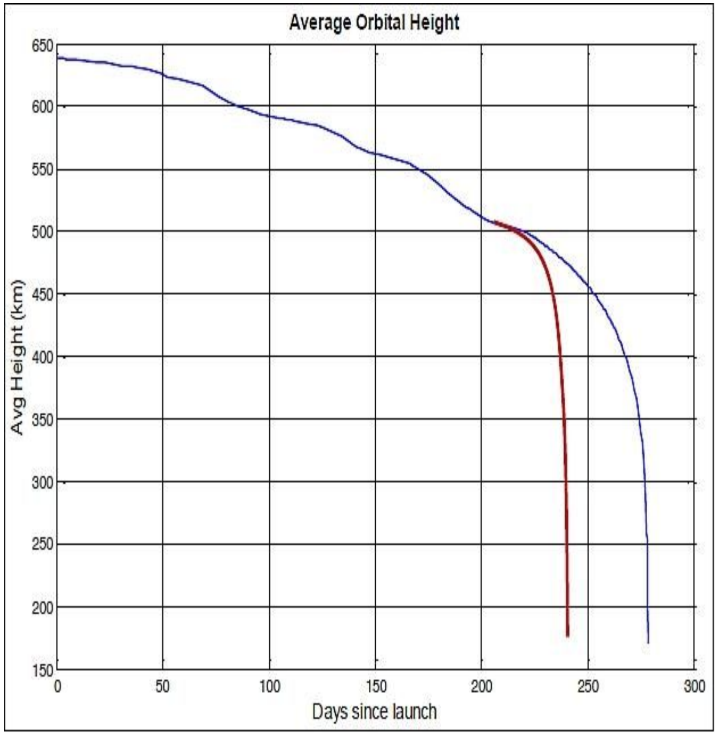


Figure 4: Orbital profile of the NanoSail-D2 mission (Source: NASA)

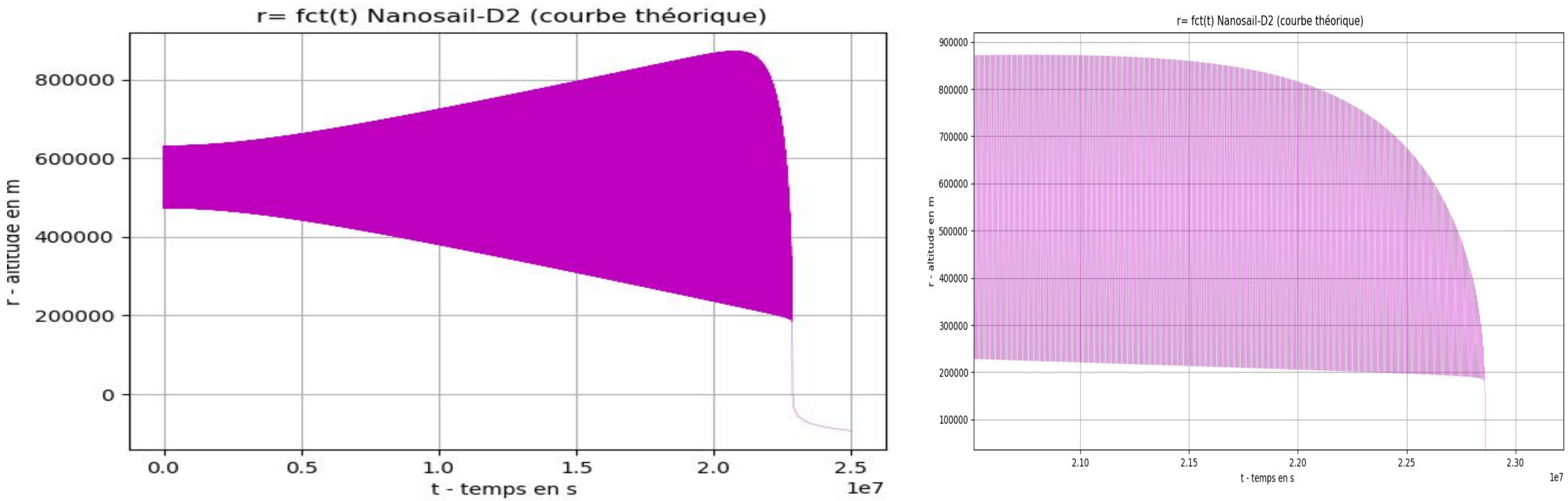


Figure 5: Altitude versus time for the Nanosail-D2 Python satellite

$T = 2{,}86 * 10^7 s \Rightarrow T = 264$ days

Difference of 24 days, i.e. an error of 10% for this example

Acceptable modeling → approved

Modelling and comparison

Satellite

Mass: m=2000 kg

V0=7635 m/s^2

Altitude= 429 km

Without solar sail

t= 4 years and 5 months (1620 days)

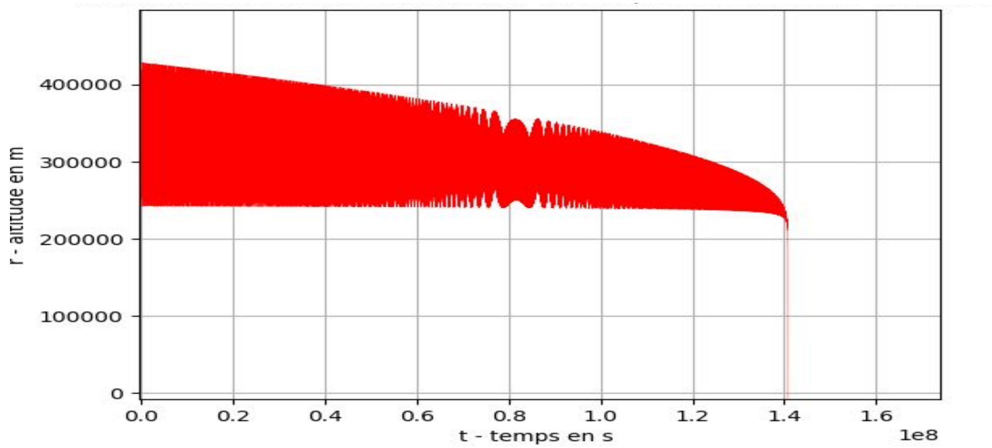


Figure 6: Altitude versus time without Python solar sail

N

With solar sail S=1200 m^2

t= 284 days

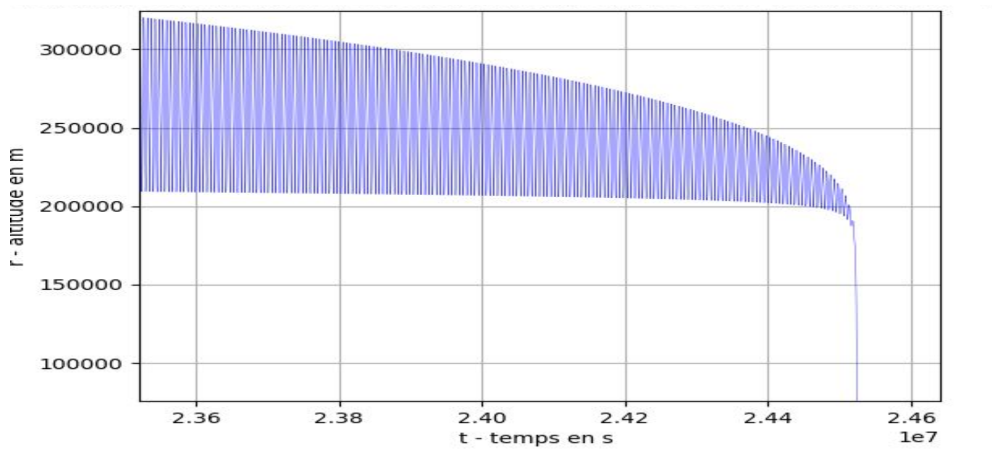


Figure 7: Altitude versus time for a sail with a surface area of 1200m2 Python

t (period in days)

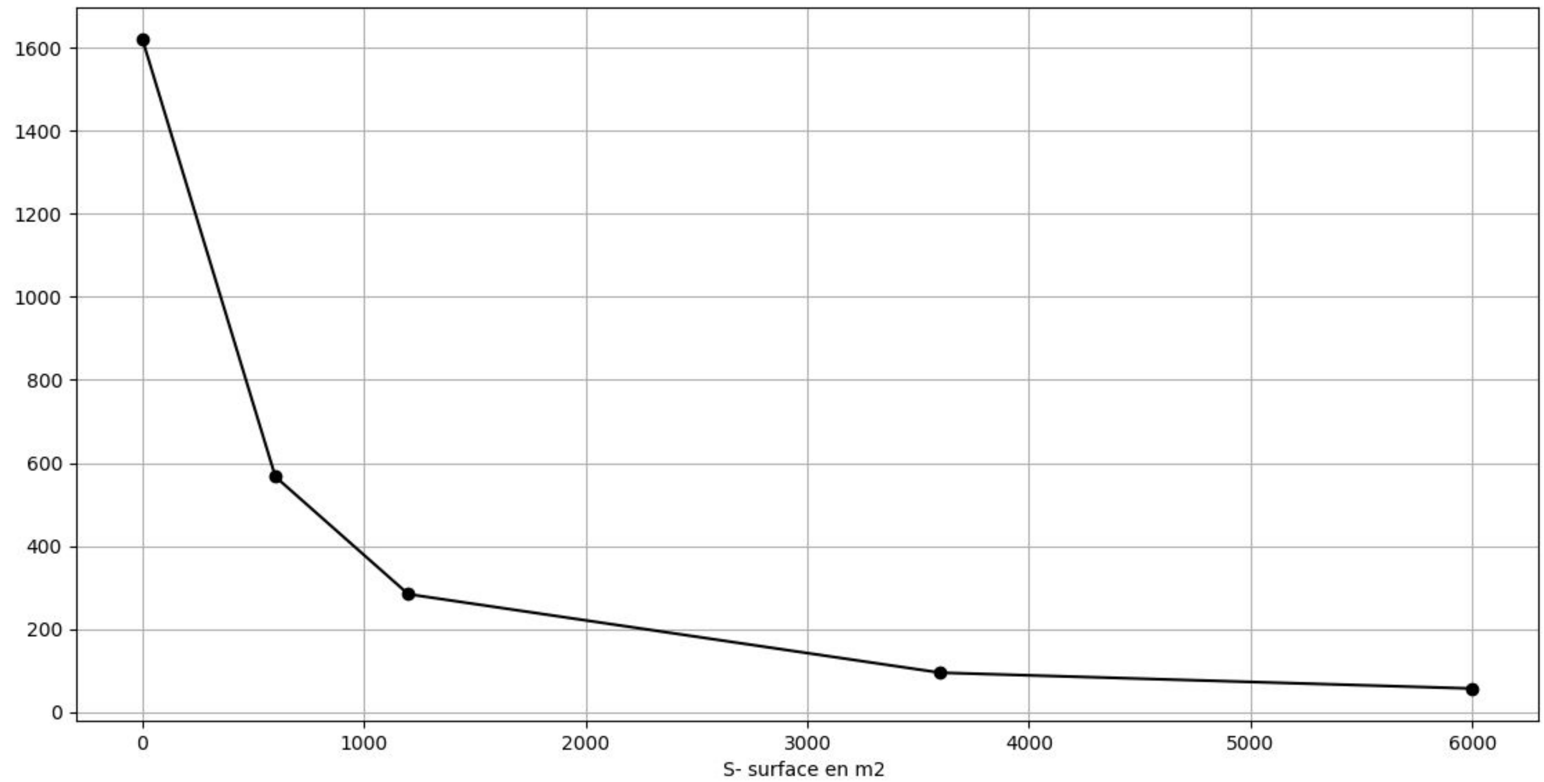


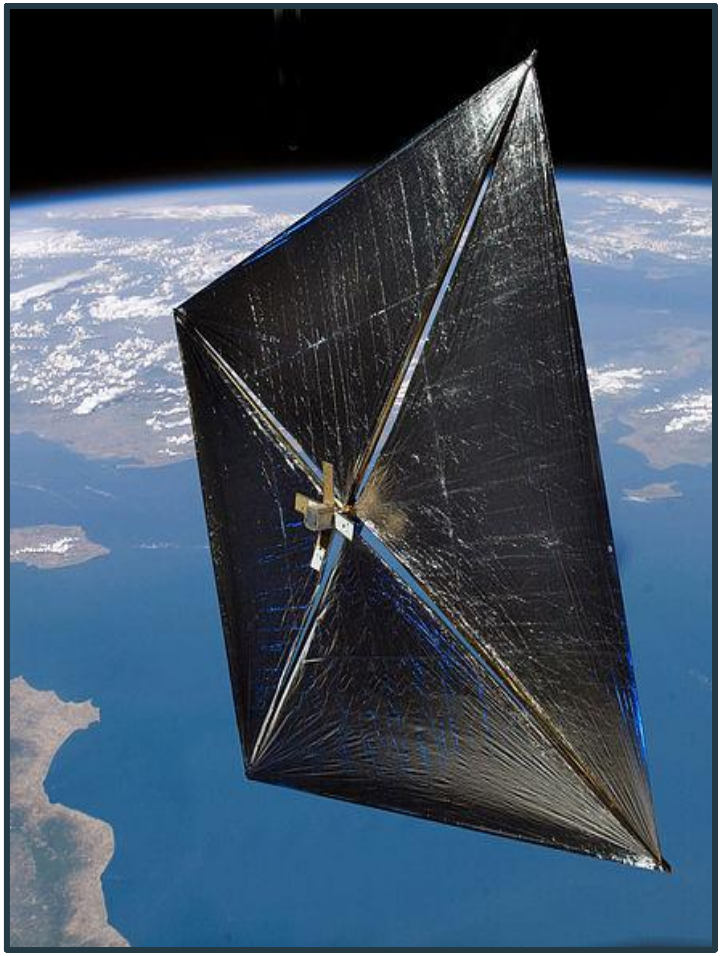
Figure 8: Orbit duration as a function of Python sail area

Regressi: $T = T_0 * \exp(\frac{-S}{A})$ with $T_0 = 1610 \pm 230$ days

et $A = 619 \pm 218\ m^2$

Limits

- Size
- Inexhaustible energy BUT impossible to concentrate the Sun's rays on a sail to obtain sufficient propulsion
- Impossible to unfold the sail without tearing it

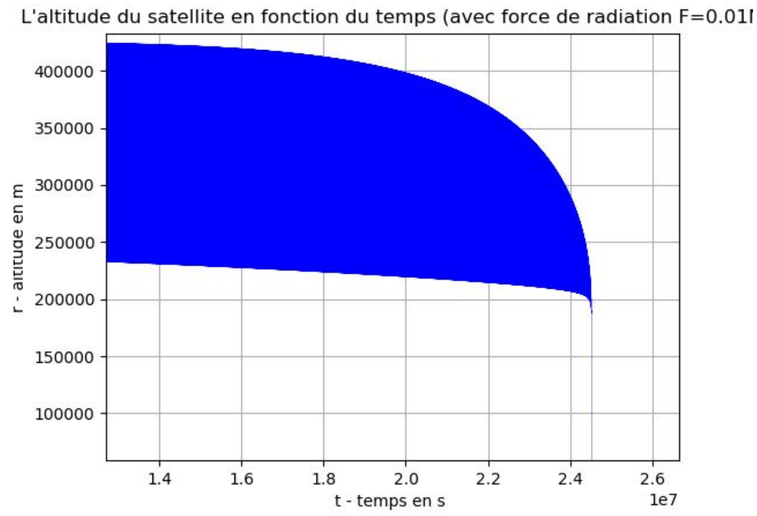


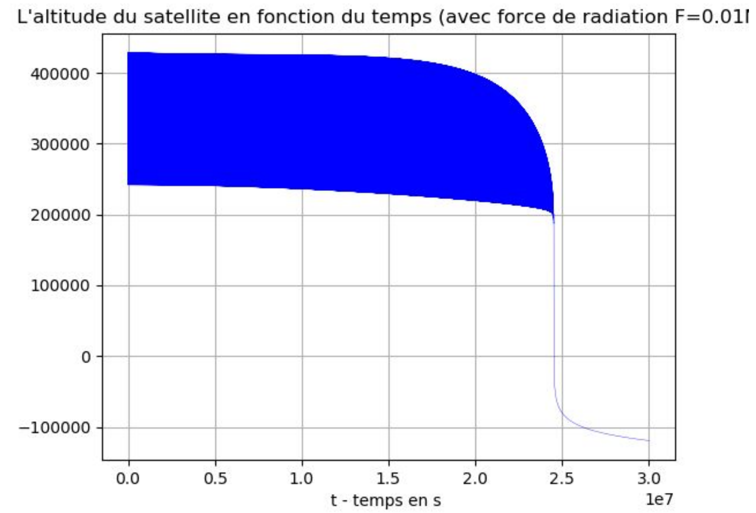
Thank you for your attention

Altitude versus time for a force F=0.01N


```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
from scipy import *
from scipy.integrate import odeint

#introduction des constantes
G = 6.7*(10**-11)
M = 6*(10**24)
m=2000
SSAT=1200 # modèle sunjammer
Cp=0.349
Ct= 1.34
Mair= 28.976* 10**(-3)
g=9.81
R=8.31
T=273.15
hs =(R*T)/(Mair*g)

def fonction(X,t):
    r = X[0]
    theta = X[1]
    v = X[2]
    omega = X[3]
    dr = v  # r point
    dtheta = omega #theta point

 # la force de radiation ne s'exerce pas sur la voile dans la zone d'ombre
    if theta % 2*pi > pi:
        F=0
    else:
        F=0.01

        # projection en considérant les ondes em parallèles entre elles/ voile perpendiculaire à la direction terre-satellite

 dv =-(G*M) /(r**2) + r*(dtheta**2)+( exp((6371000-r)/hs) *SSAT* sqrt(dr**2 +(r*dtheta)**2)*(Cp* r* dtheta - Ct *dr))/(2*m)-F*sin(theta)/m # r deux points

    dw = - 2*dr*dtheta/r - ( exp((6371000-r)/hs) *SSAT *sqrt(dr**2 +(r*dtheta)**2)*(Ct* r* dtheta - Cp *dr) )/(2*m*r)-(F*cos(theta))/(m*r) # theta deux points

    return [dr,dtheta,dv,dw]
```

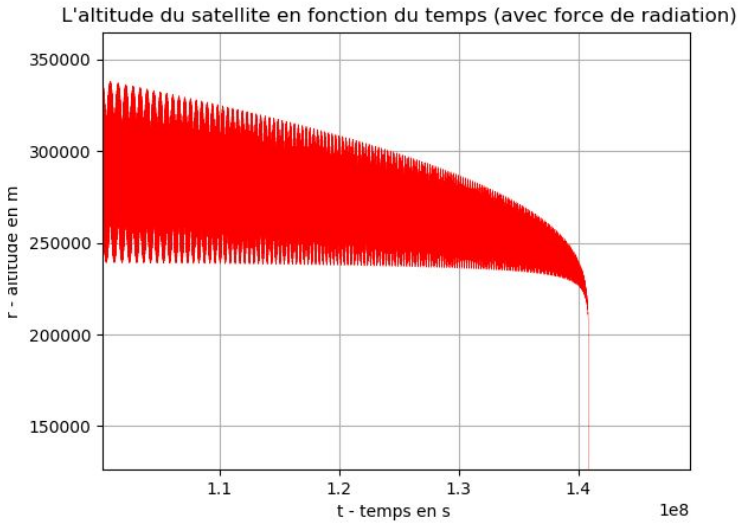
```
t = np.linspace(0, 4000000, 1000000) # intervalle de temps de l'étude

x0= 6800000  #altitude initiale
v0= 0
theta = pi/2
omega = 7635/6800000  # vitesse angulaire initiale en rad/s

# tableau des conditions initiales
CI=np.array([x0,theta,v0,omega])
Sols=odeint(fonction,CI,t)

r= Sols[:, 0]
u = Sols[:, 1]

# courbe de l'altitude par rapport au temps
plt.plot(t,-6371000+r, linewidth=0.25, color='m')
plt.grid()
plt.xlabel('t - temps en s ')
plt.ylabel('r - altitude en m')

plt.title(" L'altitude du satellite en fonction du temps (avec force de radiation)")
plt.show()
```





Part of the program: Altitude versus time without solar shading

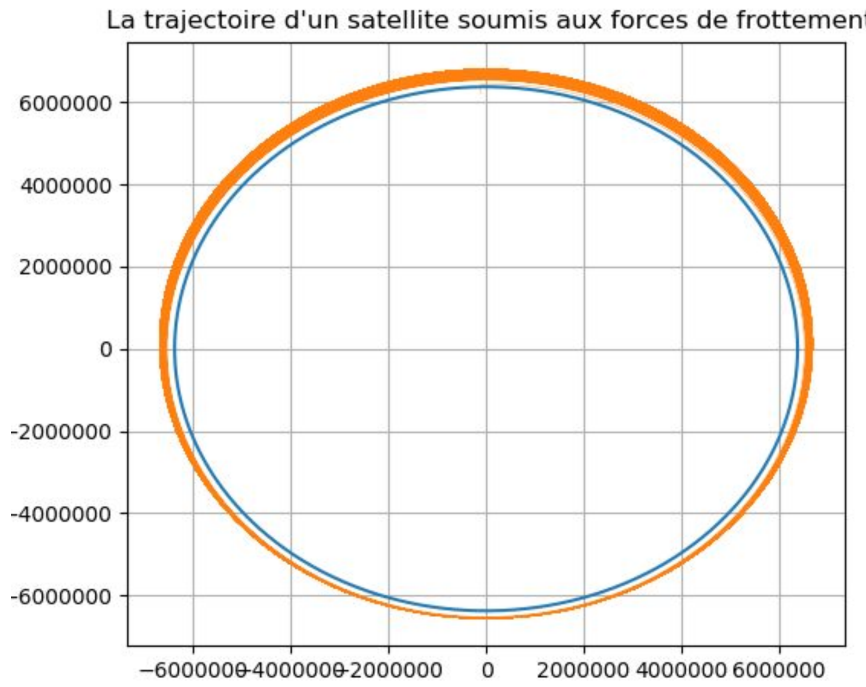

```
SSAT=50  # la surface change

def fonction(X,t):
    r = X[0]
    theta = X[1]
    v = X[2]
    omega = X[3]
    dr = v  # r point
    dtheta = omega #theta point
```




```
  dv =-(G*M) /(r**2) + r*(dtheta**2)+( exp((6371000-r)/hs) *SSAT* sqrt(dr**2 +(r*dtheta)**2)*(Cp* r*
dtheta - Ct *dr))/(2*m)-F*sin(theta)/m # r deux points

    dw = - 2*dr*dtheta/r - ( exp((6371000-r)/hs) *SSAT *sqrt(dr**2 +(r*dtheta)**2)*(Ct* r* dtheta - Cp
*dr) )/(2*m*r)-(F*cos(theta))/(m*r) # theta deux points

    return [dr,dtheta,dv,dw]
```

Part of the program: end-of-life satellite trajectory

```
t = np.linspace(-3000000, 30000000, 1000000)  # intervalle de temps de l'étude
x0= 6800000 #altitude initiale
v0= 0
theta = pi/2
omega = 7635/6800000
# tableau des conditions initiales
CI=np.array([x0,theta,v0,omega])
Sols=odeint(fonction,CI,t)
# traçage de la terre
theta = np.linspace(0, 2*np.pi, 1000)
A = 6371000*np.cos(theta)
B = 6371000*np.sin(theta)
plt.plot(A, B)

x = Sols[:, 0]
u = Sols[:, 1]
```

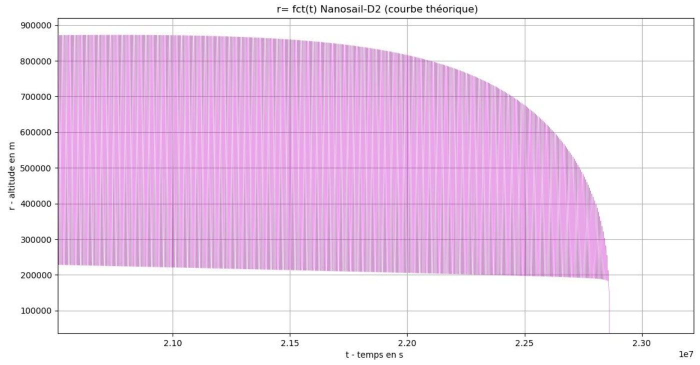


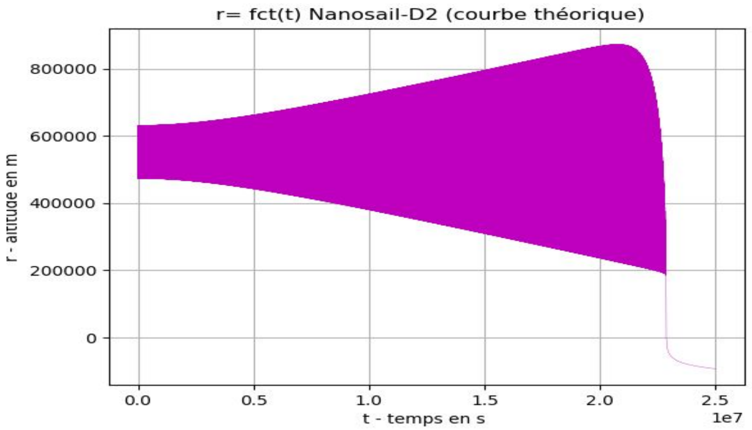

```
# courbe de la trajectoire
plt.plot(x*cos(u),x*sin(u), linewidth=0.25 )# projection sur x et y pour obtenir la
courbe finale
plt.grid()
plt.title( " La trajectoire d'un satellite soumis aux forces de frottement" )
plt.show()
```

Program section: Altitude versus time for Nanosail-D2


```
#introduction des constantes et les caractéristiques de ce satellite
m=4
SSAT=8
Cp=0.349
Ct= 1.34 # coefficients qui dépendent normalement de la forme du satellite
Mair= 28.976* 10**(-3)
g=9.81
R=8.31
T=273.15
hs =(R*T)/(Mair*g)

def fonction(X,t):
    r = X[0]
    theta = X[1]
    v = X[2]
    omega = X[3]
    dr = v
    dtheta = omega
    if theta % 2*pi > pi:      # la force de radiation ne s'exerce pas sur la voile dans la zone d'ombre
        F=0
    else:
        F=0.00009  # nouvelle force calculée pour S= 10 m2

    dv =-(G*M) /(r**2) + r*(dtheta**2)+( exp((6371000-r)/hs) *SSAT* sqrt(dr**2 +(r*dtheta)**2)*(Cp* r* dtheta - Ct *dr))/(2*m)-F*sin(theta)/m # r deux points

    dw = - 2*dr*dtheta/r - ( exp((6371000-r)/hs) *SSAT *sqrt(dr**2 +(r*dtheta)**2)*(Ct* r* dtheta - Cp *dr) )/(2*m*r)-(F*cos(theta))/(m*r) # theta deux points

    return [dr,dtheta,dv,dw]
```

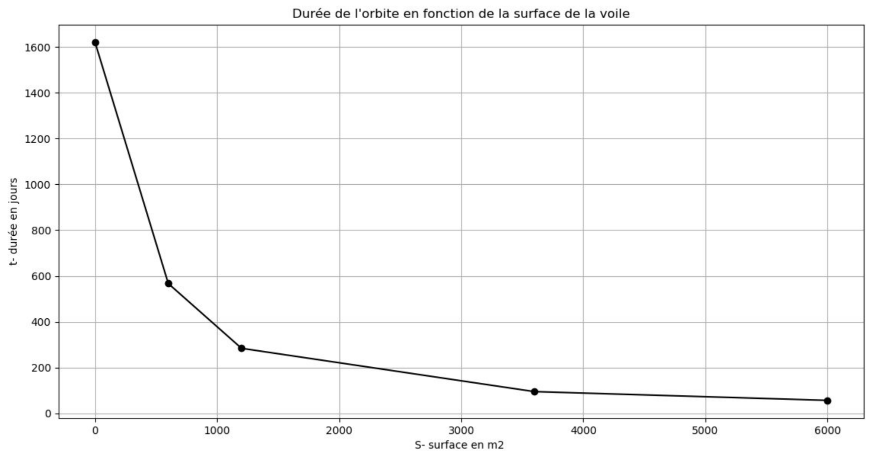
```
t = np.linspace(0, 25000000, 1000000)

x0= 7002500  #Moyenne du périgée 615km et l'apogée 648 km
v0= 0
theta = pi/2
omega =  2* pi /(97.34*60)  # période 97.34min
# tableau des conditions initiales
CI=np.array([x0,theta,v0,omega])
Sols=odeint(fonction,CI,t)


r= Sols[:, 0]
u = Sols[:, 1]


plt.plot(t,-6371000+r, linewidth=0.25, color='m')
plt.grid()
plt.xlabel('t - temps en s ')
plt.ylabel('r - altitude en m')

plt.title(" r= fct(t) Nanosail-D2 (courbe théorique)")
plt.show()
```





Orbit duration as a function of sail area


```
import numpy as np

import scipy as sp

import matplotlib.pyplot as plt


X=[0,600, 1200,3600,6000]

Y=[1620 , 566.8 , 284, 95 , 56.92]

# utilisation du programme “altitude en fonction du temps”  pour plusieurs valeurs de F

plt.plot(X,Y, color='k', marker='o')

plt.title("Durée de l'orbite en fonction de la surface de la voile")

plt.xlabel('S- surface en m2 ')

plt.ylabel('t- durée en jours')

plt.grid()

plt.show()
```



Approximation with Regressi: