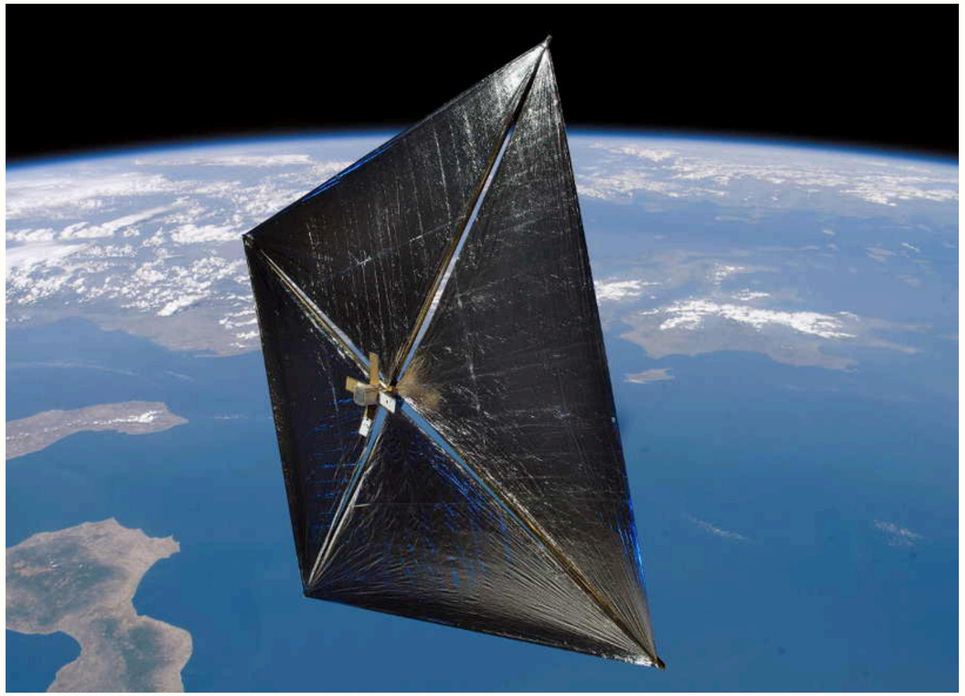
Attitude control of Payankeu solar sail

Preliminary study on the sail deployment



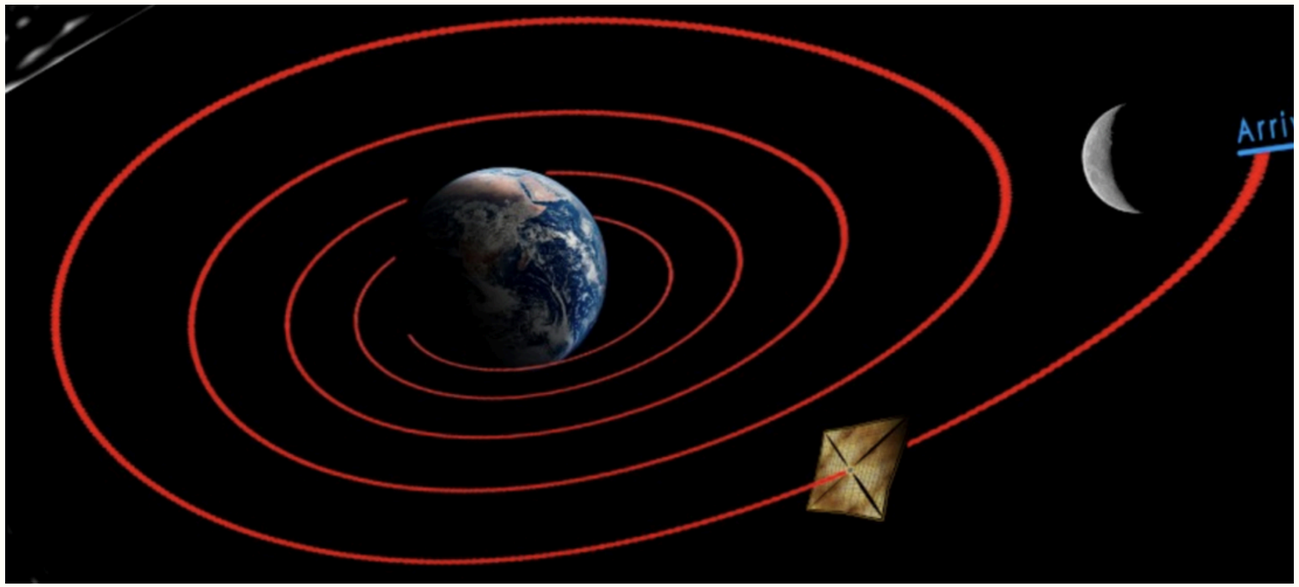
Owen STRUB University of La Réunion

1. Introduction

2. Study procedure

3. Summary

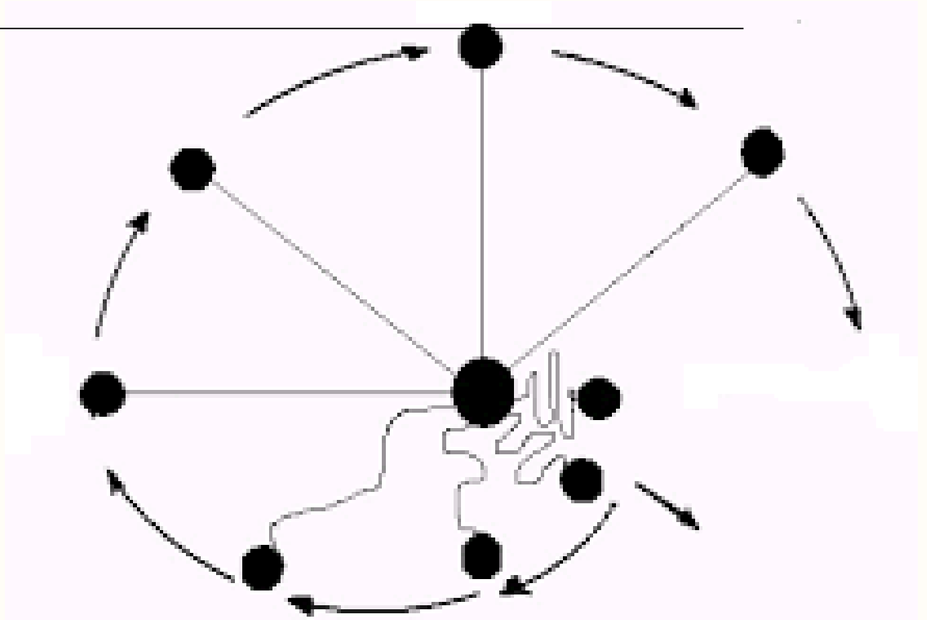
Inspired by IKAROS (JAXA), Payankeu must overcome many challenges:



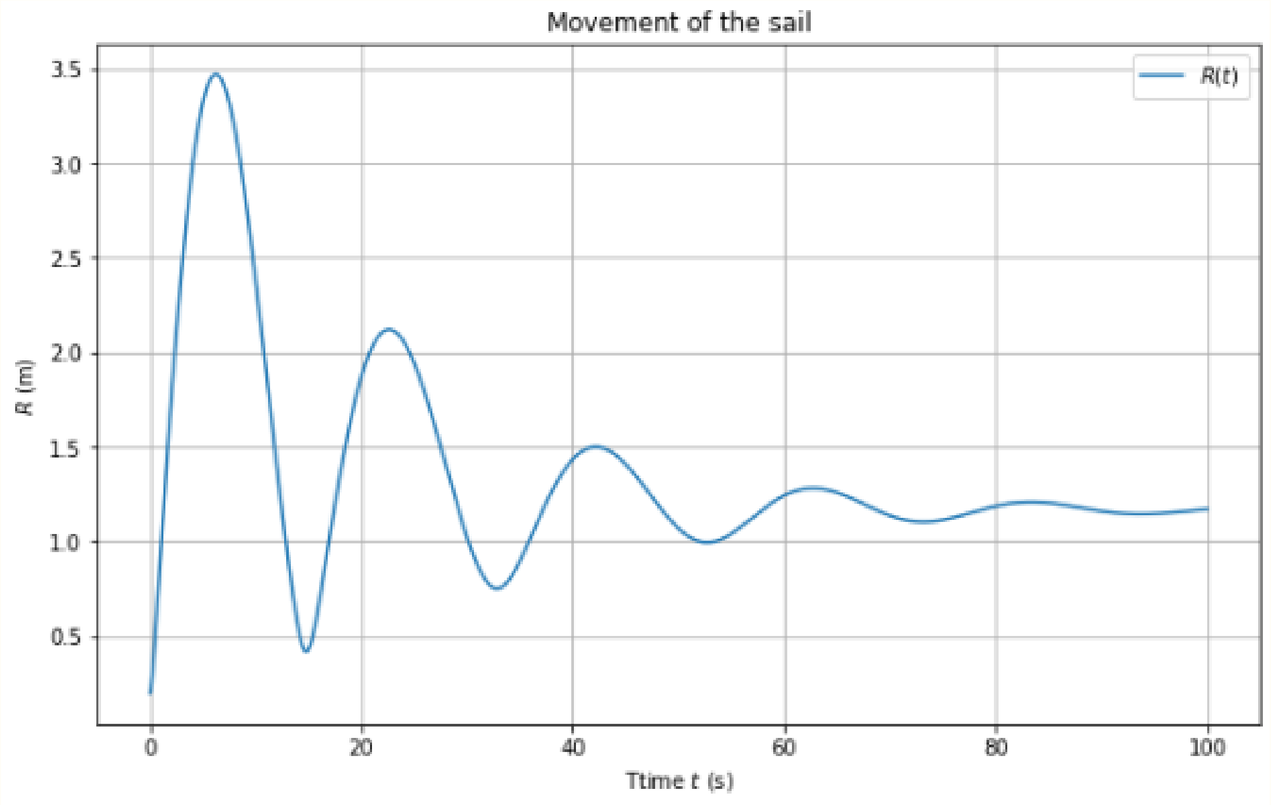
- Weights 3kg
- Has no power source other than the Sun
- Navigating in Space

Payankeu uses only centrifugal force to deploy its sails

As nothing holds the sails deployed, the system becomes complex and delicate



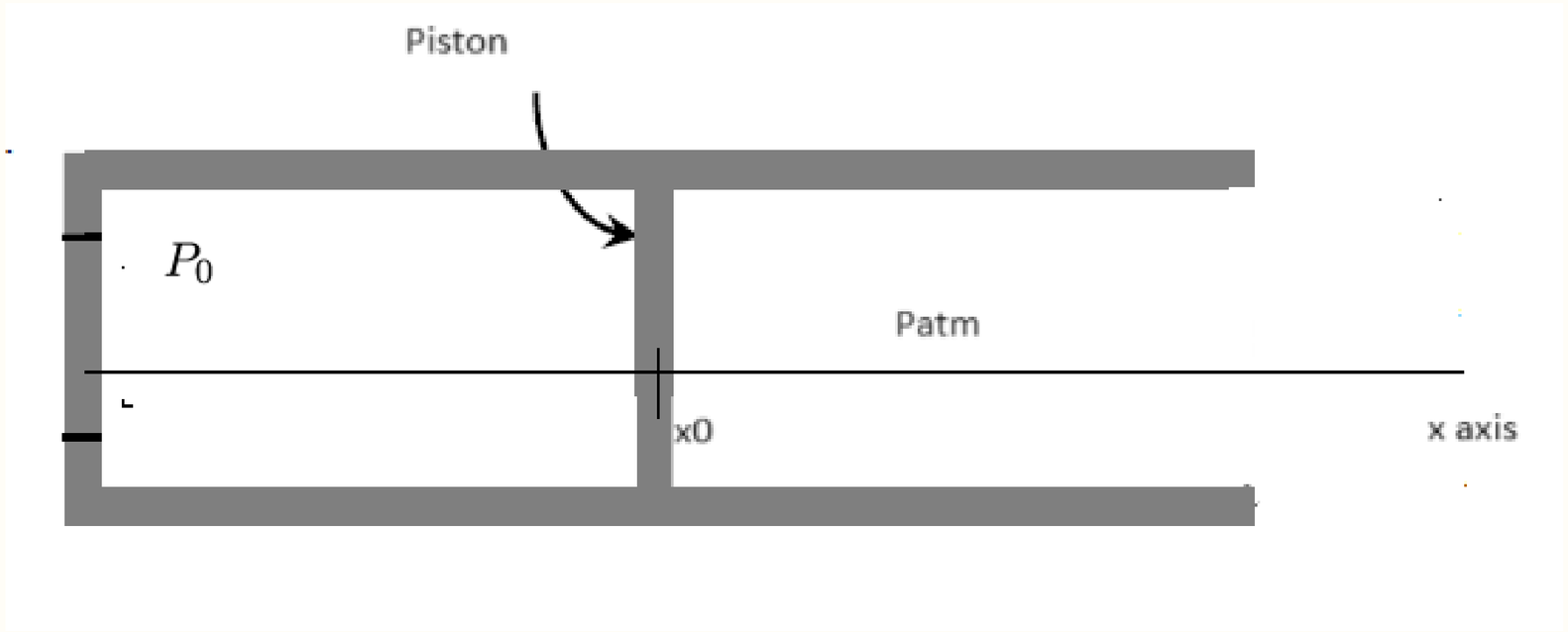
Simple representation of the sail opening



the movement corresponding to the schematic system

it satisfies a nonlinear differential equation

However, an analogy can be made with a system that can be modeled more efficiently :



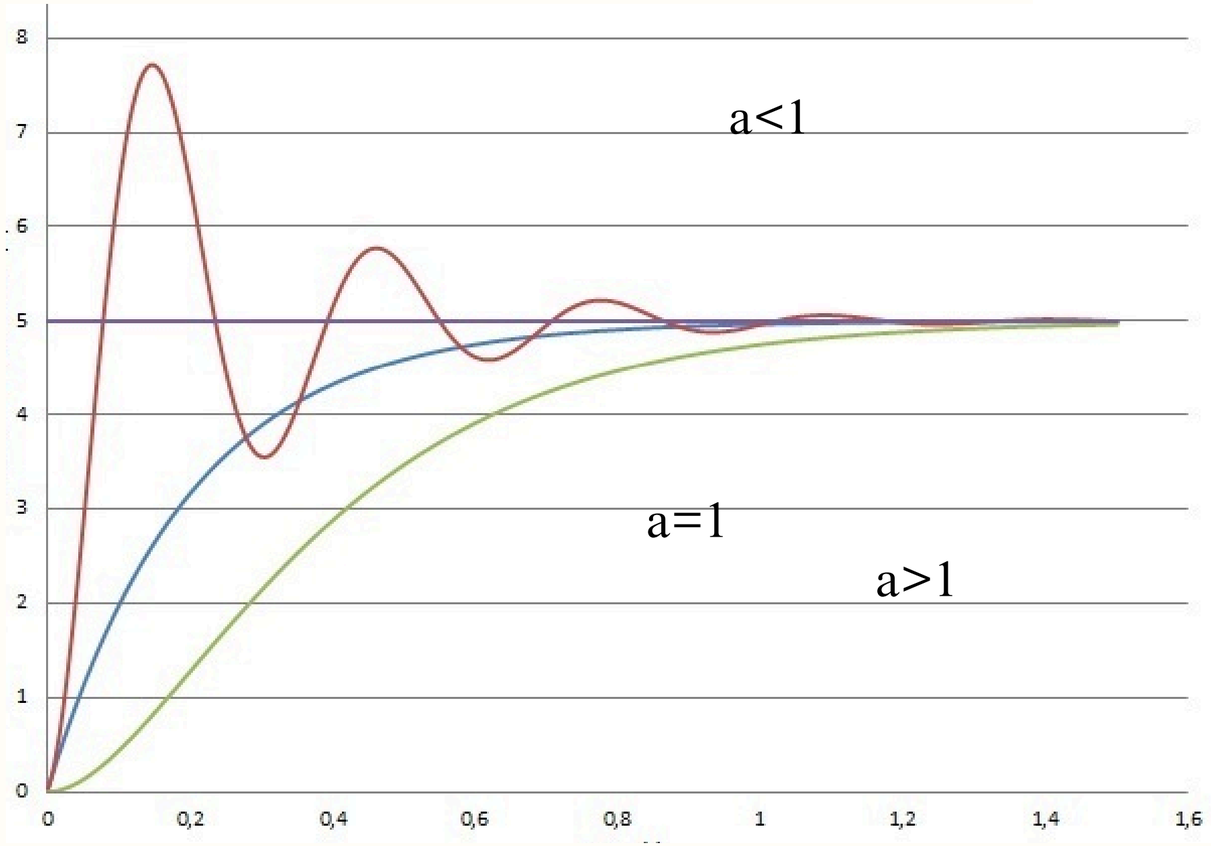
The implementation of this system for conducting tests is all the simpler

$$\frac{d^2x}{\mathrm{dt}^2} + \frac{\mathrm{AP_0}}{\mathrm{Mh}}x + \frac{b}{M}\frac{dx}{\mathrm{dt}} - \frac{P_{\mathrm{atm}}A}{M} = 0$$

P_0	initial pressure
h	height of the gaz tank
A	surface of the piston
M	mass of the piston
Patm	atmospheric pressure
b	drag coefficient

isothermal gas transformation, ideal gas

Transfer function in the Laplace domain: $$H(p) = \frac{\frac{P_{atm}h}{P_0}}{1 + \frac{bh}{AP_0}p + \frac{Mh}{AP_0}p^2}$$



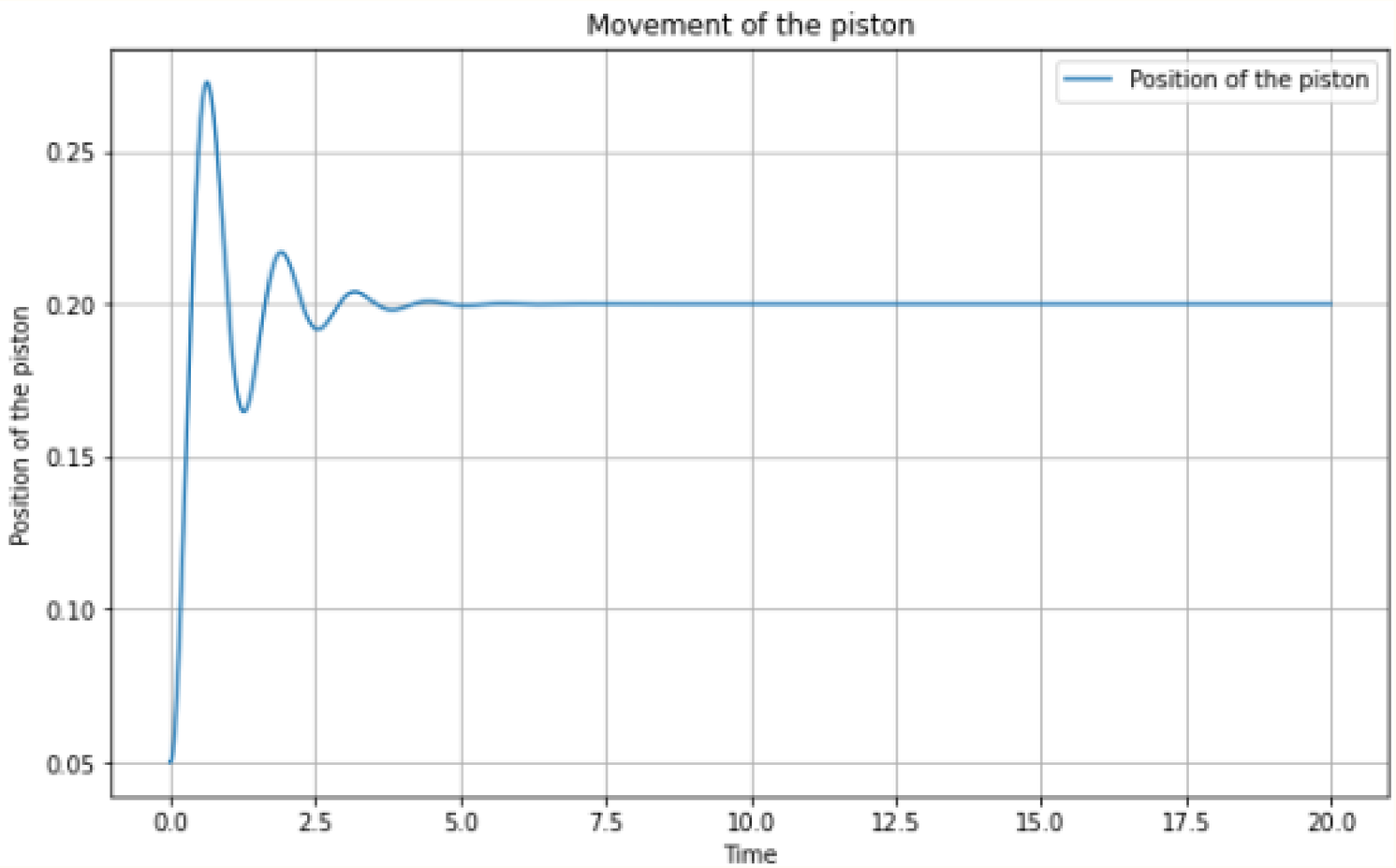
the damping ratio : $$a = \frac{b}{2}\sqrt{\frac{h}{AMP_0}}$$

Illustration of possible response types for the equation according to the value of the damping ratio

the function converges to the coefficient:

$$\frac{P_{\mathrm{atm}} h}{P_0}$$

the piston stroke can therefore be adjusted

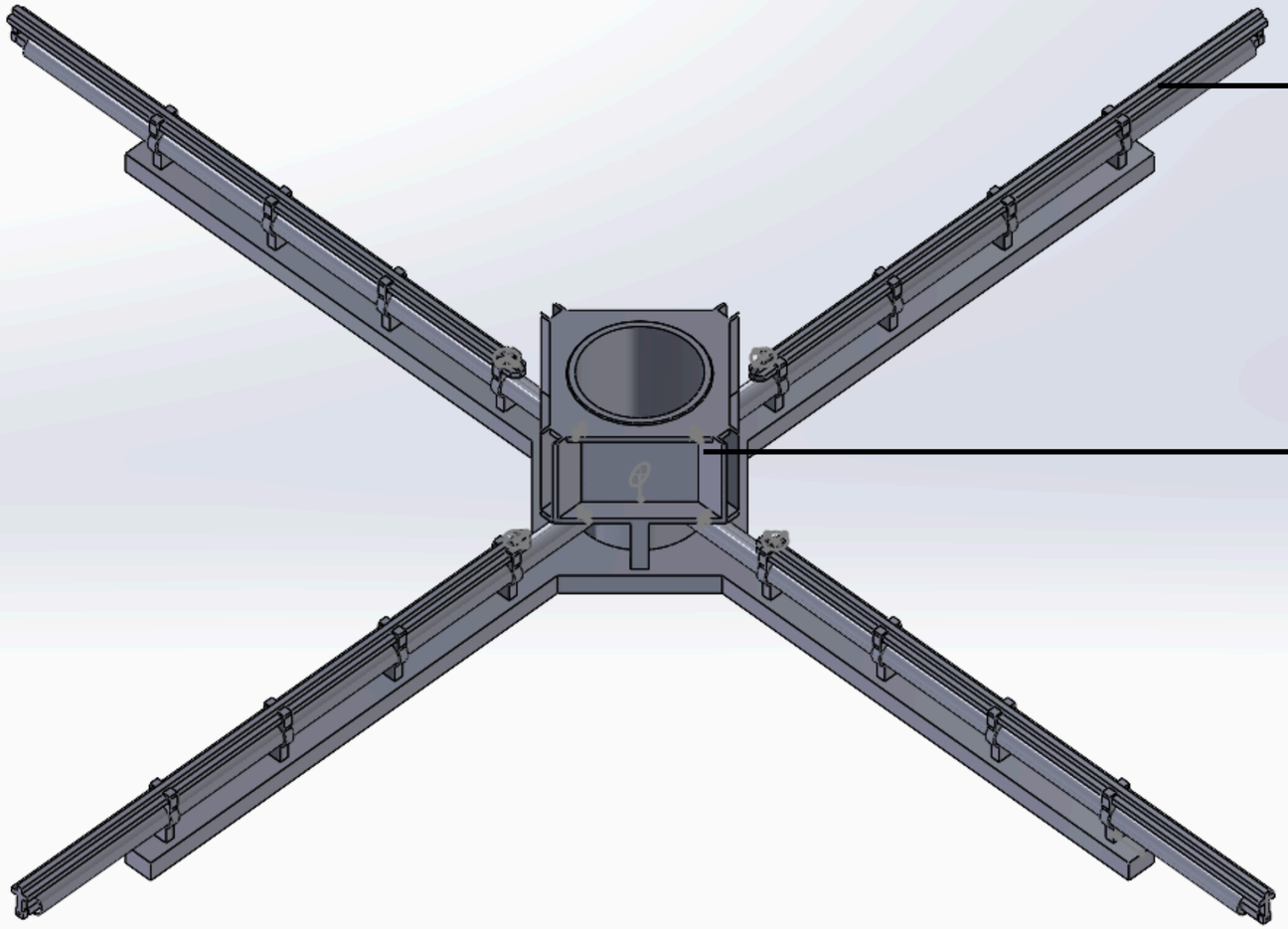


Now the overshoot needs to be reduced using the damping coefficient :

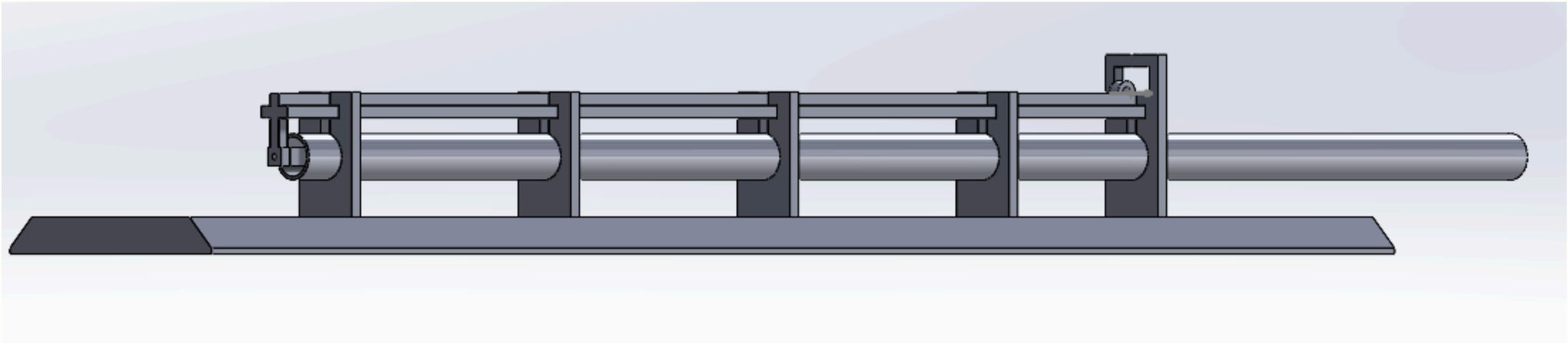
$$a = \frac{b}{2}\sqrt{\frac{h}{\mathrm{AMP}_0}}$$

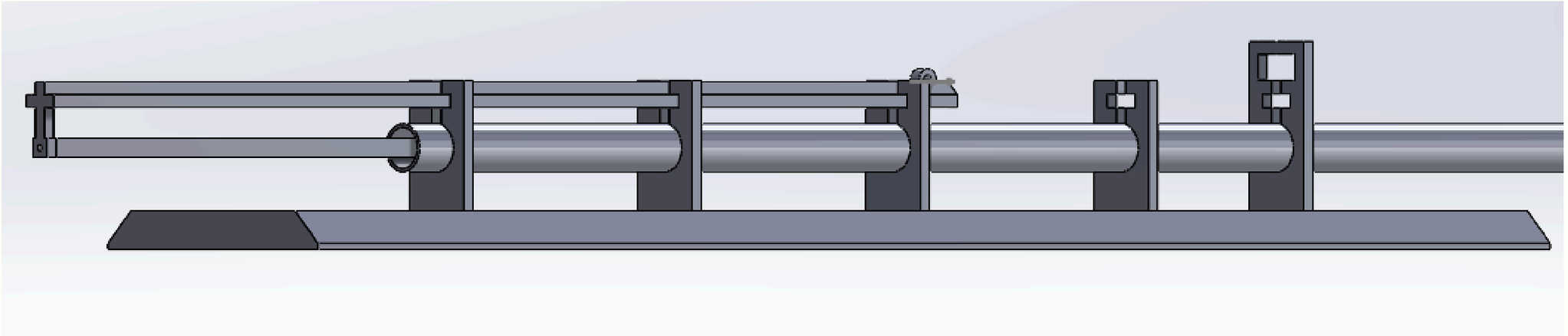
The coefficient of friction needs to be determined, and it must be verified if this modeling accurately corresponds to a real situation

Creating a model for conducting tests



Quarter of the model



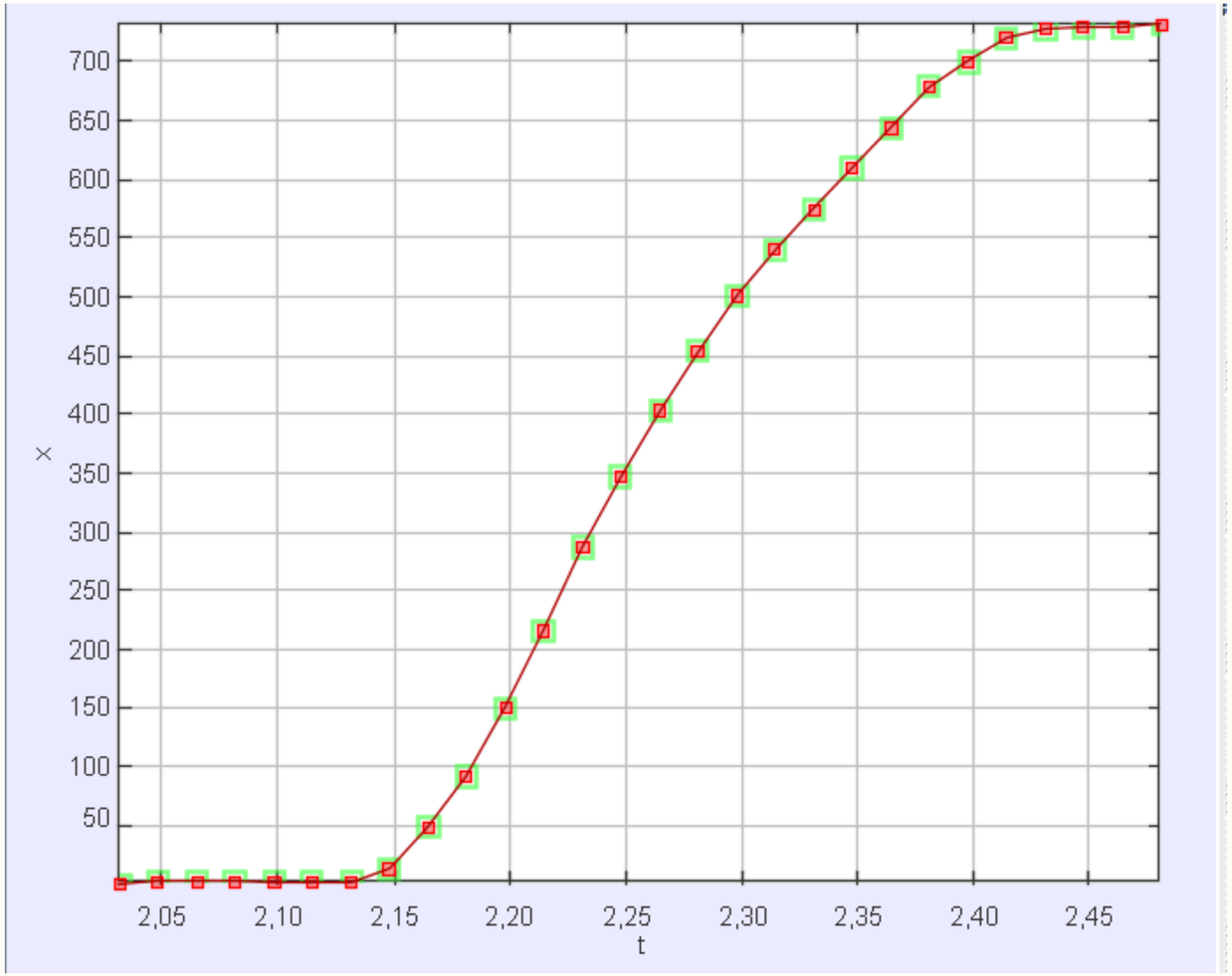


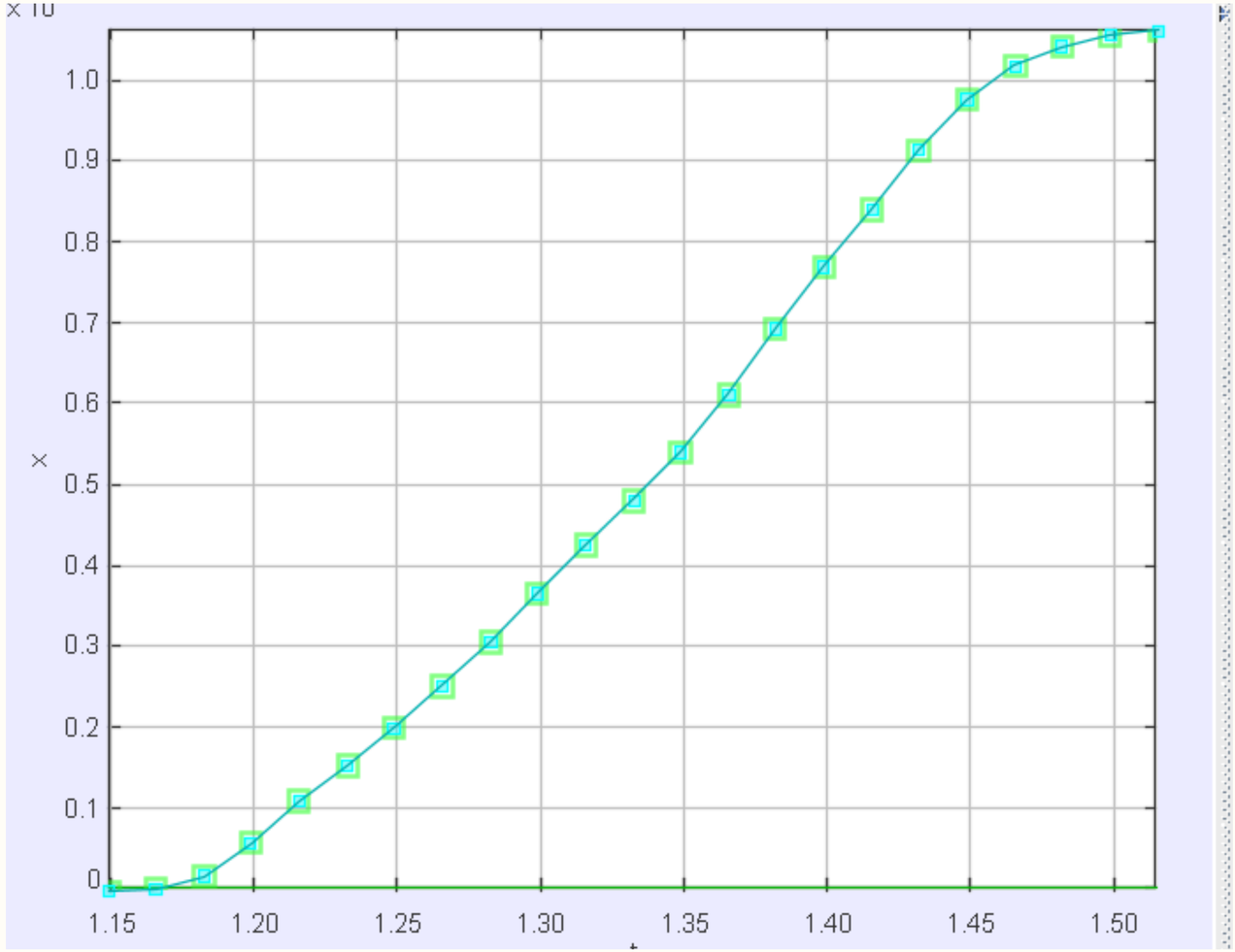
valve

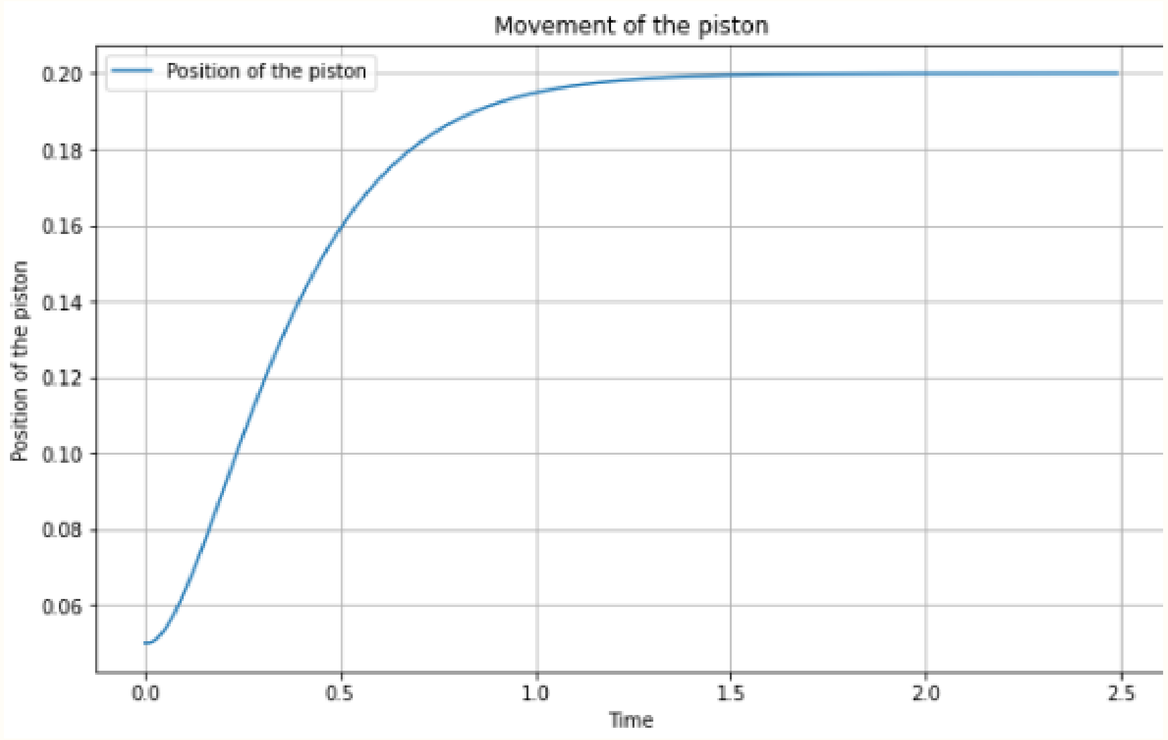
pressurized tank

pump

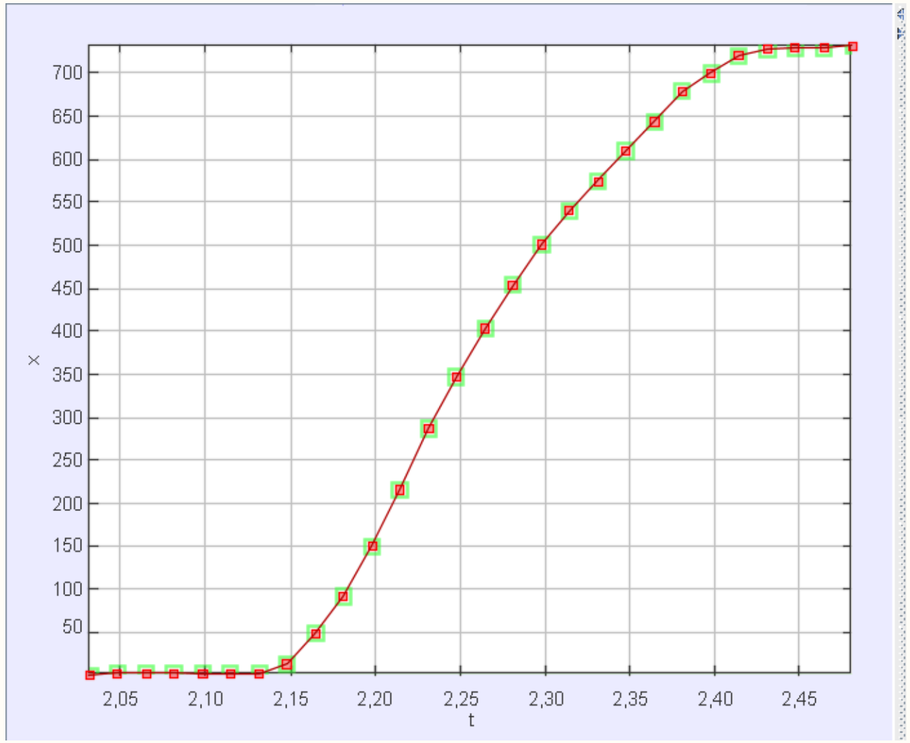
Position of the piston as a function of time







theoretical movement



experimental movement

- The deployment mode of the sail is complex;

  the study is limited to an empirical study and requires a test in low Earth orbit

- The analogy with the piston allows for adjusting the essential parameters necessary for proper deployment

- Concessions need to be made regarding the mass of the device to implement a solution capable of ensuring successful deployment

APPENDICES

Piston Code

```python
import numpy as np
import matplotlib.pyplot as plt


A = 0.02          # Surface
P0 = 1.1          # initiale pressure
M = 0.1           # weight
h = 0.1           # Lenght of gaz chamber
Patm = 1.0        # atm pressure
b = 0.5           # drag value


x0 = 0.05         # Position 0
v0 = 0.0          # Speed 0


dt = 0.01
t_max = 20.0


num_steps = int(t_max / dt)

t_values = np.zeros(num_steps)
x_values = np.zeros(num_steps)
v_values = np.zeros(num_steps)

x_values[0] = x0
v_values[0] = v0

for i in range(num_steps - 1):
    acceleration = - A * P0 * x_values[i] / (M * h) - b * v_values[i] + Patm * A / M
    v_values[i+1] = v_values[i] + acceleration * dt
    x_values[i+1] = x_values[i] + v_values[i] * dt
    t_values[i+1] = t_values[i] + dt


plt.figure(figsize=(10, 6))
plt.plot(t_values, x_values, label='Position of the piston')
plt.xlabel('Time')
plt.ylabel('Position of the piston')
plt.title('Movement of the piston')
plt.grid(True)
plt.legend()
plt.show()
```

Sail Code

```python
import numpy as np
import matplotlib.pyplot as plt


R0 = 0.2
omega0 = 0.1 * np.pi  # (rad/s)
tf = 50.0
dt = 0.01
R_max = 3.5
k = 0.01  # Coil strenght
gamma = 0.01  # drag
M = 0.75

R = [R0]
v = [0.0]
time = [0.0]
omega = [omega0]


while time[-1] < tf:
    R_current = R[-1]
    v_current = v[-1]
    t_current = time[-1]


    force_recall = k * (R_current - R0) /M


    force_dissipative = -gamma * v_current /M


    if R_current > R0:
        acceleration = (R0**4 / R_current**3 * omega0**2 / M - force_recall + force_dissipative)
    else:
        acceleration = R0**4 / R0**3 * omega0**2 / M - force_recall + force_dissipative
```

```
    v_next = v_current + acceleration * dt
    R_next = R_current + v_next * dt
    if R_next < R0:
        R_next = R0
        v_next = 0
    R.append(R_next)
    v.append(v_next)
    time.append(t_current + dt)
    if R_next > 0:
        omega.append(omega0 * (R0 / R_next)**2)
    else:
        omega.append(0)

R = np.array(R)
v = np.array(v)
time = np.array(time)
omega = np.array(omega)

plt.figure(figsize=(10, 6))
plt.plot(time, R, label='$R(t)$')
plt.xlabel('Ttime $t$ (s)')
plt.ylabel('$R$ (m)')
plt.title('Movement of the sail')
plt.grid(True)
plt.legend()
plt.show()
```