A Modern Odissey to Ares

Team 892: Problem A

Abstract

In this paper we study the viability of sending a 2000 Kg spacecraft with the aid of a radiation pressure sail. In particular we study the optimal conditions for traveling to Mars with this system. In this work a model based on solving a system of second order ordinary differential equations by Runge Kutta 4 method is presented. Starting from the papers published by Heller et. al. [6] and Rozhcov et.al. [9] in 2017, we have focused on searching the optimal path respect to time and exploring the parameters that perturb the desired situation.

Once we have obtained the solution that enables to travel smoothly to Mars in 97.29 days, we focused on exploring different enhancements of the model so that it can become more realistic and physically well-described, but as well providing powerful results with paths close to optimum with two different methods.

Trough the understanding of every parameter in the problem, we have been able to observe a clear tendency between the smoothness of the paths with their tendency to approach Mars. Furthermore, we can affirm the dependency with the area of the sail is much greater than with the initial launching angle. Furthermore, we have found a clear example of how powerful this tool can be, because it enables to reach paths that were very expensive energetically.

In order to conclude, we would like to remind that although this study shows how much control can you have of this system, when working with the correct set of coordinates; we have studied the downsides off the system and its vulnerabilities, that are crucial to better understand the system and to therefore search a better solution for the problems that occur.

I Introduction

The effect of the Radiation Pressure it has been known for years, it is widely used in optical systems such as Optical Traps, but it was recently when researchers started to seek for applications on Spacecraft Building. The idea of a sailing spacecraft is discussed in several papers such as [6], and [1] to travel to distant stars; and even to travel to Mars in [9].

In this paper we discuss the optimal path and sail dimensions to be able to send a solar sail spacecraft to Mars such that its entry velocity is upper bounded.

We start by presenting the situation that we have been given, presenting the variables that will be used in the paper and detailing the geometric properties of the problem. Once the problem scenario has been presented, we introduce the simplified version of the main model that will be used on the paper, explaining the different physical terms considered, the approximations and assumptions made, and the optimized parameters in order to obtain the optimized path and sail shape.

Using the model, the system is solved numerically and the obtained data is provided by plot representations. We have also considered slight enhancements of the physical model which we discuss in the next section by explaining their theoretical reasoning and comparing its obtained results with the main solution. Once all results are provided we discuss the strengths and weaknesses of the principal model and the supplementary enhancements, by comparing the quality of the obtained data with the computational cost to obtain it and the elegance of the theoretical equations obtained.

I.1 Used variables and initial scenario

The used variables on this paper are specified in table 1, and in case to be known constants its value and definition it is shown at the table 2 that is displayed in the Appendix A.

Symbol	Variable
t	Time elapsed since the initial launch
$\vec{r}(t)$	Spacecraft position respect to the Sun 1
$\vec{r}_T(t)$	Earth's position respect to the Sun 1
$\vec{r}_M(t)$	Mars's position respect to the Sun 1
$\theta(t)$	Sail's orientation angle
$\Psi_S(t)$	Sail CM angle with the Sun
$\Psi_T(t)$	Sail CM angle with the Earth
$\Psi_M(t)$	Sail CM angle with Mars
α	Reflection angle
η	Angle of Mars exposure
φ	Angle of launch

Table 1: Specification of all used variables

We have considered that the main goal of the project is to optimize the path and the sail's mass of a spacecraft propelled by pressure of radiation from outer bright bodies, in order to minimize the time and the sail's mass under the restrictions to arrive to the planet Mars with no more relative velocity with the planet than an upper bound.

In the set of the problem we have been told that a rocket will launch for us the sail with Earth's escape velocity. From this statement we make the first assumption that we do not need to wonder about the specific maneuvers that the launcher will do in order to let the sailed spacecraft on its own. We also assumed that the direction of launching could be adjusted, so that it can be thrown (from the earth frame of reference) with its escape velocity to any direction wanted.

Since in the explanation of the problem it is stated that the launching would take place where the planets Earth and Mars are the closest together we have seen that this situation corresponds, theoretically, to when the Earth is at its aphelion and Mars at its perihelion [5].

It is a reasonable assumption to consider the movement in the 2D plane described by the orbit of the Earth, due to the fact that the deviation angle of the rest of orbits is negligible, in particular, the tilt of the Martian orbit is 1.3.

With this knowledge, we define the first of the three frames of reference that will be mentioned in this work. We define that the position of the Sun it is fixed at the origin and that the x-axis is the one that aligns Earth's aphelion with Mars' perihelion, and the y-axis the perpendicular direction so that both planets orbit counterclockwise. A scheme of this frame it is shown in the figure 1

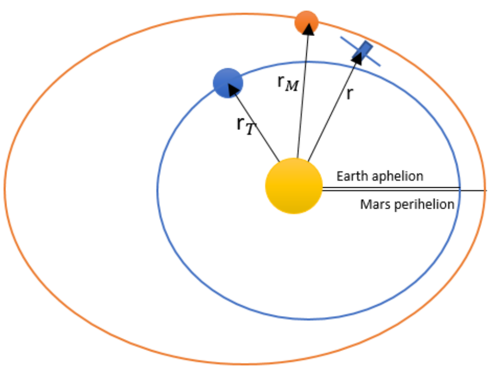

Figure 1: Description of the Sun's inertial frame of reference

We did assume that this frame of reference is an inertial frame of reference, and so that the Newton's equations of motion can be used.

We have also considered that the launch will be done in a nearby Earth orbit, more precisely in a Low Earth Orbit (LEO) at 2000 km from Earth's surface. For simplicity, and for reasons that will be mentioned in the discussion of the results, we have assumed that the launching point will be aligned with the x-axis. Using the values mentioned in the table 2, we can calculate the initial position of the sailed spacecraft, the Earth and Mars. We assume that the value $t = 0$ it corresponds to when the rocket is launched and the Earth, Mars and Sun are aligned in the two planets semi-major axis[1].

$$\vec{r}(0) = (r_{T_{max}} + r_{Earth} + r_{LEO}, 0)$$
$$\vec{r}_T(0) = (r_{max,T}, 0) \quad \vec{r}_M(0) = (r_{min,M}, 0) \quad (1)$$

Once the initial positions of these two planets, the Sun and the sail are known; it is natural to ask what it will be the value of the initial velocity of the spacecraft. For the two planets, we will use known data, mentioned in the table 2, and the impositions of having a counterclockwise rotation to affirm that the initial velocities of these two planets are $\dot{\vec{r_T}}(0) = (0, 29780)m/s$ and $\dot{\vec{r_T}}(0) = (0, 26500)m/s$.

To calculate the initial velocity of the sail spacecraft, we need to first calculate the scape velocity from Earth. In order to do so, we assume the situation where just the Earth exists as a celestial body, and it is at rest, thus considering an inertial frame of reference. This frame will be called Earth frame of reference and it will be used just in this part of the paper. Since it is assumed to be inertial, it can be used the known results from gravitation theory and educe that the Earth escape velocity referred at the paper will be:

$$v_{esc} = \sqrt{\frac{2M_T G}{r}} \quad (2)$$

where r is the radius of the orbit that the body to escape is. In order to be consistent with our previous assumptions and our model, this r corresponds to $r = r_{LEO} + R_T$, and thus it becomes that the escape velocity that we have

[1]It is assumed that when a vector is expressed with coordinates (x,y), it corresponds to the coordinates in the Sun frame of reference.

considered is: $v_{esc} = 9.764 km/s$.

As it has been already pointed out, from a Solar perspective, this velocity can be oriented towards any direction pleased. To model this situation we have set the angle φ as the indicator of this initial direction of the term of the v_{esc}. But since the Earth indeed is orbiting the Sun, it has a velocity described before. Assuming that the non-inertial terms are negligible as well as the relativistic terms, we conclude that the initial velocity of the spacecraft would be

$$\dot{\vec{r}}(0) = \dot{\vec{r}}_T(0) + (v_{esc}\cos(\varphi), v_{esc}\sin(\varphi)) = \\ = (v_{max,T} + v_{esc}\cos(\varphi), v_{esc}\sin(\varphi)) \quad (3)$$

So we have the initial velocity as a function of the parameter φ. In order to define some quantities that will be relevant to the development of the main model involving radiation pressure (explained in the next section), we introduce the following parameters as angular relations between the spacecraft and the main bright bodies of the system.

In the figure **??** one can see a simple scheme representation of the different defined angles. The picture illustrates the definition of the angle θ, which is a parameter that holds no direct relation with the bodies that used to control the orientation of the sail.

Furthermore, for each relevant body of the system we have defined the angle ψ_I where I denotes the index for each body. These angles fulfill the equation:

$$\tan(\Psi_I) = \frac{y - y_I}{x - x_I} \quad (4)$$

The angle α will be referred as the incidence angle, and it can be seen in the picture that is the complementary of the usual reflection angle. It will be important at the time where equations from external sources may have to be invoked.

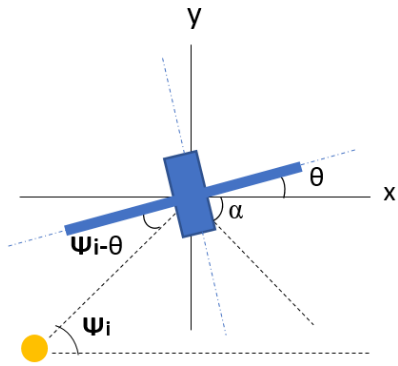

Figure 2: Graphical description of the angle Ψ_i in the Sun frame of reference.

II Main Model

Assumptions

The dynamics of the whole system are described by Newton's gravitational law and the radiative pressure that will receive the sail during the travel. We have simplified the problem by considering the Earth-Mars-Sun system, which are the bodies that will play fundamental role in the problem. Therefore any interaction with other celestial bodies has been neglected.

In addition, we neglect the spin of the planets. This way we finally obtain the evolution equations for the positions of the planets by using Newton's law:

$$\ddot{\vec{r}}_T = -GM_T\Big[\frac{M_M(\vec{r}_T - \vec{r}_M)}{|\vec{r}_T^2 - \vec{r}_M^2|^{3/2}} + \frac{M_S(\vec{r}_T)}{|\vec{r}_T|^3}\Big] \quad (5)$$

$$\ddot{\vec{r}}_M = -GM_M\Big[\frac{M_T(\vec{r}_M - \vec{r}_T)}{|\vec{r}_M^2 - \vec{r}_T^2|^{3/2}} + \frac{M_S(\vec{r}_M)}{|\vec{r}_M|^3}\Big] \quad (6)$$

Once the positions of the main bodies can be described, the next step is considering the movement of the ship. We have made the same assumptions in its gravitational interactions (Sun fixed, Earth-Mars-Sun system only and non-rotational sailing).

The proposed design of the ship will be a capsule with a certain mass attached with the sail. This sail will be considered as a one-sided perfect mirror in such a way that reflects all the incident light to one face (specially in the visible/UV-regime were the Sun emits more energy). The other face is assumed to be transparent. back Furthermore, our model takes into account the radiation pressure emitted by Mars, because of the reflection of the Sun radiation.

For simplifying the model we will neglect the thermal radiation of the planets due to their own temperature (Plank's Law) and also the deterioration of the mirror. Therefore, we will consider that the mirror is a perfect conductor during all the travel.

The radiative force is modeled depending on the proximity of the source: the point source approximation regime and the close source regime. For the first one, as in can be seen proved in [**princeton**], the radiation force for a mirrored surface is:

$$m\ddot{\vec{r}} = P_r A cos(\alpha) = \frac{L_s}{4\pi|\vec{r}|^2}\frac{2A}{c}sin^3(\Psi_s - \theta) \quad (7)$$

where A is the area of the sail, $L_s$we have used that sin(α)=cos($\Psi_s - \theta$). This force is always in the direction of the sail and in the Sun's reference frame will depend on its orientation (angle θ). However, this is only true under the punctual source approximation.

When the ship approaches a planet, the punctual method will not work for the reflected radiation on its surface. In more detail, we do the following assumptions in the planets radiation.

Earth's radiation will not be considered because the ship is launched from the non illuminated side. Also we assume that when the ship is far enough for receiving the reflected radiation from earth will be negligible compared to the Sun's. In summary, Earth will not have any contribution to the radiative pressure force component.

In the case of Mars we will consider the radiation reflected in the surface taking into account the albedo. Therefore, we will integrate around the solid angle a uniform intensity (solution obtained from McInnes, C. R. and Brown, J. C. in [2]).

$$m\ddot{\vec{r}} = \frac{L_m A}{3\pi c R_m^2}(1-(1-\frac{R_m}{r})^2)^{3/2} \quad (8)$$

However, not all the reflected light arrives to the sail. For modeling this, we introduce the following correction $C(\eta)$ to Mars' luminosity. It is justified geometrically in figure 3.

$$C(\eta) = \frac{1+cos(\eta)}{2} = \frac{1}{2}\Big(1 + \frac{(\vec{r}-\vec{r}_M)}{|\vec{r}-\vec{r}_M|}\cdot\frac{\vec{r}}{|\vec{r}|}\Big) \quad (9)$$

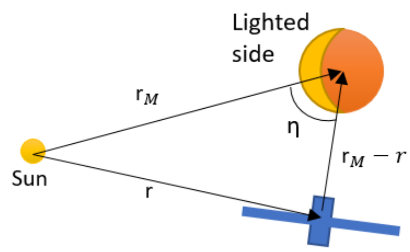

Figure 3: Geometrical interpretation of correction C(η) presented in equation 9.

This way, if the ship is in front of the Mars' lighted face will not have any correction, and zero if its in the obscure one. Also, the luminosity of Mars is going to be related with Ls and $\vec{r}_M$ (we will neglect the shadow produced by the ship)

$$L_m = \frac{\pi R_m^2}{4\pi|\vec{r}_m|^2}L_s\sigma \quad (10)$$

Where σ is the albedo of the mars Surface.

Numerical implementation

For proving the consistency of the model and understand the relation between some parameters we have done a numerical implementation (a scrip made in Python 2.7 which can be consulted at the end of this document).

According to the main Model, we need to solve a 2n order ODE system of 8 variables ($\vec{r}$,$\vec{r}_M$,$\vec{r}_T$, Area and ϕ). However, we can obtain a 1st order ODE system of 14 variables by introducing the velocities. We have used the Runge-Kutta method until fourth order (RK-4) for solving the system.

In the scrip, we set the planets into their initial conditions explained before. Then we place our ship in the LEO distance and in the not illuminated side (right side) with a certain launching angle ϕ. After that, we run the simulation and study if Mars' orbit crosses the ship trajectory.

Also, because the sail of the ship can move a certain angle, we will set two regimes: a first one of acceleration (making the sail perpendicular to $\vec{r}_s$ and using Sun's light for increasing velocity) and another of deceleration (moving the sail since its perpendicular to $\vec{r} - \vec{r}_M$).

Furthermore, we have used an Adaptative Step Method for the iterations. This way, we could obtain a higher accuracy in the gravitational interaction (especially in situations where the bodies are close) without heavily increasing the computation time.

III Principal results

Our main goal was obtaining the fastest trajectory for the ship that fulfill the proposed $\leq$9 km/s final velocity. The first step was obtaining a raw trajectory without taking into account any radiation pressure. This way, we could tell which were the initial conditions that approach better the position of Mars.

After trying different launching angles (ϕ),[2] any initial condition lead to an intersection between Mars and the ship in the first months, being the ship too fast. However, we saw that the most optimal configuration was having a $\phi \in [\frac{\pi}{2}, \frac{3\pi}{2}]$.

The next step was adding the radiation pressure for obtaining a more smooth approach. We considered an area of $A = 2,57142 \cdot 10^5 m^2$ (spending 1800 kg of the total available in the sail).The first idea was starting with a big ϕ (launch the ship near the opposite direction from Earth) and then use the sail as a parachute because of Mars' reflected radiation. One example of this type of trajectories was obtained in figure 4.

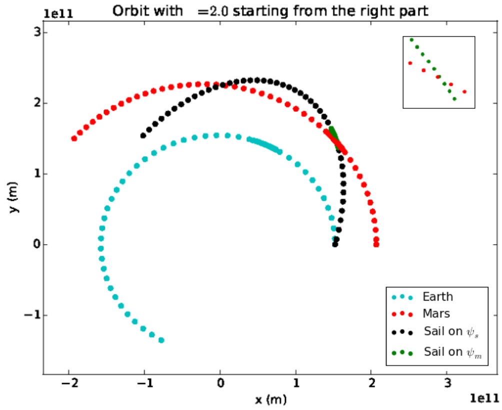

Figure 4: Trajectory that do not fulfill the conditions. Parachute approach opened at 10 times the radius of Mars. The ship is arrives first at the intersection point.

The problem with this trajectories was that opening the sail when the ship is at few times the radius of Mars is not enough for slowing down the ship. For this reason, we tried a different approach: instead of opening the parachute near Mars, the ship would have the sail extended during all the journey (full sailing approach). Then, using some engines or small gas propulsion, it would orientate the reflective part of the sail towards the Sun or Mars.

This way, the ship initially used the Sun for a small acceleration (for compensating the loss of velocity due to mars) and then decelerate using the Mars' reflected radiation. The contribu-

[2] The definition of ϕ is such ϕ=0 gives the same direction as Earth's velocity and ϕ=$\frac{\pi}{2}$ the opposite one.

tion of solar radiation while orientating the sail towards mass has also been considered. One example of a successful (distance from Mars smaller that 3 times its radius and relative velocity of 6.5 km/s) trajectory is given in figure 5 by a launching angle $\Phi = 2.690$.

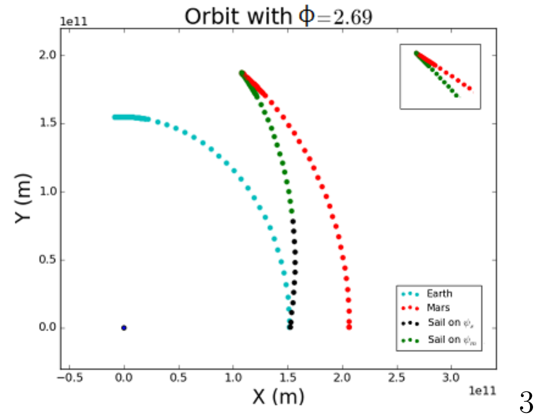

Figure 5: Trajectory that fulfills the conditions. Full sailing approach achieves a more smooth trajectory for an balanced proportion of acceleration and

Once found a solution, it is important to scan on the proximities of that particular point. In figure (6) one can find the distribution of the minimum distance between the rocket and Mars. The figure also contains the modulus of the relative velocity between them at their closest points.

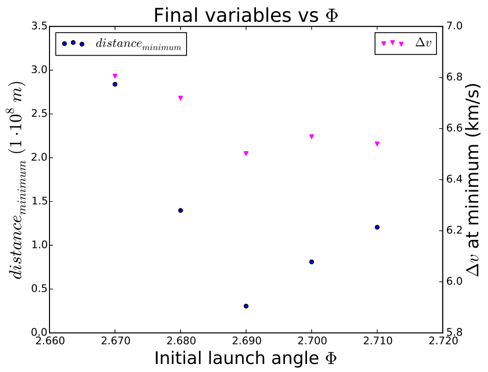

Figure 6: Minimum distance and relative velocity achieved for different values.

The figure 6 shows that in fact $\Phi = 2.690$ is the best angle to reach mars. Moreover, one can identify a linear relation between the minimum distance and its velocity at that point. The closest from Mars, the slower you go relative to him. It is important to remark that this relationship is only valid when using initial conditions similar to ours.

The solution relies on the value of many different parameters. We decided to study how the surface of the rocket and its orientation affect the solution on the particular solution $\Phi = 2.690$. The parameters area (A) and the parameter "Distance to flip the sail" were modified homogeneously.

The minimum distance to Mars was computed using all pairs of parameters and are shown in figure 7. This figure has given us the possibility to find the darker linear zone where the values of the area and "Distance to flip the sail" generate a better solution. This algorithm can be extended with no further work to a larger range of variables. However, due to the lack of computational time, this method was only

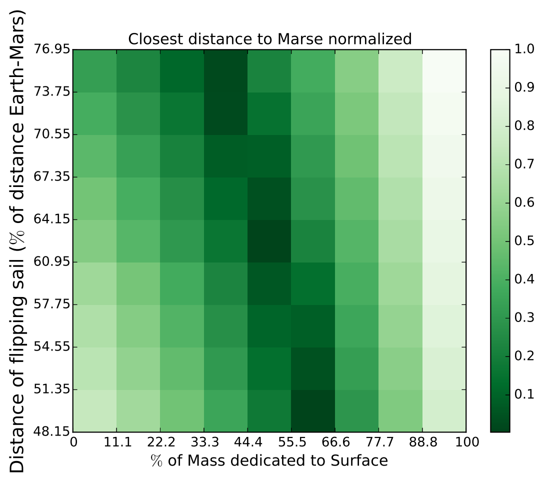

Figure 7: configurations near the successful initial conditions.

IV Model enhancements

Once the principal model has been presented and the principal obtained results displayed, we could be able to identify some interesting enhancements of the model, in order to be more complete, precise and realistic. We have also been able to develop some implementations of the model, that, by the lack of time and computational power we could not be able to present.

In this section, a first list of possible dim improvements will be provided. This improvements rely on the idea to physically better determine the model, to have a more complete description of the problem, in an attempt to obtain a more realistic description of the situation.

Light enhancements

- We could consider the gravitational effects of the other bodies in the solar system.
- We could not make the approximation that the system is 2-dimensional and take into account the tilts in the orbits of the body.
- Consider the effect of the action of solar winds, and other type of particle interactions, that will make compulsory to protect the sail.
- We have seen that in different publications on the subject, the researchers took into account the contributions of the "limb darkening effect"[3] [7], commenting on the luminosity depending on the radius of the star, and thus of the planet that reflects this light.
- A simple model has been made in order to calculate the amount of luminous energy that Mars reflected towards the spaceship has been made. It would be very interesting to compare the obtained results with the actual value, that will be obtained through the correct integration of the solid angle of the radiation reflected on Mars that arrives to the ship; taking into account also the direction of the reflection produced in this surface, and the possible scattering effect of this "mirror".
- A very curious and important aspect that would be necessary to explore is the improvement of the mechanism that enables the reorientation of the sail to the desired angles. It would have to be evaluated its energetic costs, and also its implications in terms of conservation of angular momentum, that are of huge importance in order to correctly orient the sail.
- The fact to include a factor of impurity on the mirror, that would be enlarged with time, due to the interaction of the surface with hits of other particles that would cause deformations and impurities in the surface.
- Comment towards the idea to design a specific shape of the sail, in order to take full advantage of the light received in order to collect it and focus it towards the desirable direction.
- Another aspect to consider is the fact that we assumed that just one side of the sail interacts with the electromagnetic radiation as a perfect mirror. We think that it will be interesting to search for materials with similar properties that also fulfill the astonishing property that $\rho_s = 7\frac{g}{m^2}$.

Improvements in optimizing

A second type of enhancements that we would like to discuss would be the ones that are implementation enhancements. With these improvements we expect that we would be able to optimize the path of the sail in such a way that we could find nice sub-optimal results.

We will mainly discuss two of these improvements. The main method that we used in order to find satisfactory paths to Mars with the sail, we have to say that although it is well motivated and makes sense in the regime of this competition, due to the fact of the limited time and computation power; it is rather rudimentary.

But as we gained confidence and understanding of the problem, we have seen that there are two reasonable ways that will lead to some optimization.

We think that a good method that would provide results it will be the Monte-Carlo simulation, where, given the known failing path of sailing with Ψ_S for the hole trip, calculating at every n steps if it is beneficial or not to add a $\Delta\theta$ to the orientation angle, subtract it, or remain at the same angle. The "benefit" function will be calculating some steps further and seeing whether which of the three paths is more beneficial in terms of direction to Mars.

Then, given some predetermined mid-distance to Mars related to the velocity of the ship and the amount of force that Mars can provide, the analysis of the situation would require a higher frequency of evaluation, and the condition of minimizing the velocity, in order to arrive to Mars at the desired less than $9Km/s$.

And a second method of implementing the model is to do a smart change of coordinates. This method it was seen in the paper of Rne Heller, and Michael Hippke [6]. We obtained the code that Mr. Hippke provided in its personal GitHub, it has been modified in order to adapt it to this particular problem, by substituting the obtained model of the radiation pressure from Mars and the Sun to the sail.

In order to do so, what it has been done it is to move the system into the Mars frame of reference. The change it is not trivial, due to the fact that it is a non-inertial frame of reference. And the transformations in order to achieve this change it takes the usual form of the non-inertial frame transformations.

V Discussion

Thanks to the main results presented in the sections above, we are in disposition to comment and relate all the information gathered during this project.

In our principal approach, we have focused on trying to find a path that smoothly lead to Mars. We did so by engineering the situation. First we have chosen the most extreme situation that lead to a decrease of time, that it was to do the full travel orienting maximally with the Sun, that was the body which illumination most lift produced towards Mars at the beginning, and as a counterpart we studied the situation where the sail is oriented towards Mars, in order to decrease its velocity provided by the rotation of the Earth around the Sun and the escape velocity provided by the launching rocket.

By combining these two situations we have obtained the satisfactory final result.

From the principal approach an important result has emerged once studied the relationship between the path and the initial variables such as the launching angle or the surface of the sail. The initial launching angle was thought to be the most important variable for a successful trip towards Mars. However, using the initial conditions written before, there has not been seen many changes on the trip path when changing Φ. In fact, all orbits with this exact initial velocities are very similar to (4) and (5). The main reason behind this invariance is the large orbital velocity that the Earth gives to our rocket.

VI Conclusions

In this work we have been able to completely study a very interesting system. In order to do so, we have taken several approaches. A first approach it has been to find the best quickest smooth path towards Mars that we could think of. By doing so we obtained a result that has enabled us to study the different behaviors to different perturbations added to the system, so the different parameters that described and conformed the orbit have been tested to whether be crucial to having a successful orbit or not.

From this analysis we could determine that the most relevant parameters of the orbit are the surface of the sail and its orientation. The latter it is for obvious reasons, but the dependency is greater than expected, because a slight change at the sails orientation, when the sail has important velocity can lead to very different paths.

With this particular problem, we concluded that the most determining fact is the angular velocity that Earth has, which is difficult to overcome.

We have been able to determine a orbit that smoothly travels to Mars, and the approach that we have taken to find it, it was to study the behavior of the system in simple circumstances, and from there define a simple strategy, consisting of just one reorientation of the sail, in order to try to not move apart from the extreme regimes that we know that the.

We have determined the influence that small parameters changes have on the solution by considering the particular cases of all small variations of the rocket's area and changing

VII Strengths and Weaknesses

In the following section some strengths and weaknesses of the model are commented in order to lead to further conclusions or even further research.

Acknowledgments

The authors would like to thank the Department of Physics for their livelihoods and also the Organization Committee of UPhysicsC 2016.

Appendix A: Used values

In this appendix we attach the table 2 that contains all the used values to obtain the provided results of this paper.

Symbol	Definition	Value
M_S	Sun's mass	1.988510^{30}Kg [10]
M_T	Earth's mass [4]	5.972410^{24}Kg [4]
M_M	Mars's mass [8]	0.6417110^{24}Kg [8]
m	Spacecraft's mass	2000Kg
ρ_s	Surface density	$7.0\frac{g}{m^2}$
$r_{max,T}$	Earth's aphelion [4]	1.521810^11 m
$r_{min,M}$	Mars's perihelion [8]	2.06610^11 m
$v_{min,T}$	Earth's aphelion velocity	29780 m/s
σ	Mars Albedo [8]	0.250
$v_{max,M}$	Mars's perihelion velocity	26500 m/s
R_T	Earth radius [4]	$6.378 \cdot 10^6$ m
R_{LEO}	Earth Low Orbit radius	$2 \cdot 10^6$ m
ϵ	Mars eccentricity [8]	0.0935
R_M	Mars radius [8]	3.396210^6 m
G	Gravitational constant	$6.6740810^{-11}\frac{Nm^2}{Kg^2}$
L	Sun's luminosity [10]	3.84610^26 W
c	Velocity of light	2.997910^8 m/s

Table 2: Values of used constants

Appendix C: Main Kernel and Optimization code

```
from numpy.linalg import norm as norm
import numpy as np
import matplotlib.pyplot as plt

##################################################
#                    Functions                   #
##################################################

def arctan(y,x):
    if x==0:
        a=np.pi/2*float(y)/np.abs(y)
    else:
        a=np.arctan(float(y)/x)+np.pi*(np.abs(x)-x)/(2*np.abs(x))
    return a

def accelerationEarth(rt,rm):
    a=-1*G*(Ms*rt/(norm(rt)**3)+MM*(rt-rm)/(norm(rt-rm)**3))
    return a

def accelerationMars(rt,rm):
    a=-G*(Ms*rm/(norm(rm)**3)+Mt*(rm-rt)/(norm(rm-rt)**3))
    return a

def accelerationShip(rt,rm,r,Psi_s,Psi_m,Theta,A):
    a_g=-G*(Ms*r/(norm(rm)**3)+MM*(rm-r)/(norm(rm-r)**3)+Mt*(rt-r)/(norm(rt-r))**3)
    ax_l=0.#acceleration due to radiation pressure in x direction
    ay_l=0.#acceleration due to radiation pressure in y direction
    CE=np.dot(r-rm/norm(r-rm),-r/norm(r))

    if -np.pi/2 <= -Psi_s+Theta <= np.pi/2:
        ax_l=ax_l+A/m/(np.pi*c*2)*Ls/(norm(r))**2*(np.sin(Psi_s-Theta))**3*((np.cos(Psi_s-Theta))*r[0]-(np.sin(Psi_s-Theta))*r[1])/norm(r)#Sun is very far
        ay_l=ay_l+A/m/(np.pi*c*2)*Ls/(norm(r))**2*(np.sin(Psi_s-Theta))**3*((np.cos(Psi_s-Theta))*r[1]+(np.sin(Psi_s-Theta))*r[0])/norm(r)#Sun is very far


    if -np.pi/2 <= -Psi_m+Theta <= np.pi/2:

        CE=np.dot((r-rm),r)/norm(r)/norm(r-rm)
        ax_l=ax_l+A/m/(12*np.pi*c*norm(rm)**2)*albedo*Ls*(1.-(1.-norm(Rm)/norm(r))**(3/2.))*(1.+CE)/2*((np.cos(Psi_m-Theta))*r[0]-(np.sin(Psi_m-Theta))*r[1])/norm(r)
        ay_l=ay_l+A/m/(12*np.pi*c*norm(rm)**2)*albedo*Ls*(1.-(1.-norm(Rm)/norm(r))**(3/2.))*(1.+CE)/2*((np.cos(Psi_m-Theta))*r[1]+(np.sin(Psi_m-Theta))*r[0])/norm(r)

    a=np.array([0.,0.])
    a[0]=a_g[0]+ax_l
    a[1]=a_g[1]+ay_l
    return a

#Kernel of the code
def kernel(numberiterations,rti,rmi,ri,vti,vmi,vi,A,a):
```

```
    #to store
    rearth=[]
    rmars=[]
    vearth=[]
    vmars=[]
    rship=[]
    vship=[]
    breakconditionearth=[]
    breakconditionmars=[]

    #initial conditions
    rt=np.array(rti)
    rm=np.array(rmi)
    r=np.array(ri)
    vt=np.array(vti)
    vm=np.array(vmi)
    v=vi
    h=float(10)
    Psi_s=0.
    Psi_m=np.pi
    Theta=0.2

    #main loop
    for i in range(numberiterations):

        #RK functions
        k1velearth=accelerationEarth(rt,rm)
        k1velmars=accelerationMars(rt,rm)
        k1velship=accelerationShip(rt,rm,r,Psi_s,Psi_m,Theta,A)

        k2velearth=accelerationEarth(rt+h*k1velearth/2,rm+h*k1velmars/2)
        k2velmars=accelerationMars(rt+h*k1velearth/2,rm+h*k1velmars/2)
        k2velship=accelerationShip(rt+h*k1velearth/2,rm+h*k1velmars/2,r+h*k1velship
    /2,Psi_s,Psi_m,Theta,A)

        k3velearth=accelerationEarth(rt+h*k2velearth/2,rm+h*k2velmars/2)
        k3velmars=accelerationMars(rt+h*k2velearth/2,rm+h*k2velmars/2)
        k3velship=accelerationShip(rt+h*k2velearth/2,rm+h*k2velmars/2,r+h*k2velship
    /2,Psi_s,Psi_m,Theta,A)

        k4velearth=accelerationEarth(rt+h*k3velearth,rm+h*k3velmars)
        k4velmars=accelerationMars(rt+h*k3velearth,rm+h*k3velmars)
        k4velship=accelerationShip(rt+h*k3velearth,rm+h*k3velmars,r+h*k3velship,
    Psi_s,Psi_m,Theta,A)

        k1posearth=vt
        k1posmars=vm
        k1posship=v

        k2posearth=k1velearth*h/2+vt
        k2posmars=k1velmars*h/2+vm
        k2posship=k1velship*h/2+v

        k3posearth=vt+k2velearth*h/2
        k3posmars=vm+k2velmars*h/2
        k3posship=v+k2velship*h/2

```

```
        k4posearth=vt+k3velearth*h
        k4posmars=vm+k3velmars*h
        k4posship=v+k3velship*h

        #update variables
        vt=vt+h*(k1velearth+2*k2velearth+2*k3velearth+k4velearth)/6
        vm=vm+h*(k1velmars+2*k2velmars+2*k3velmars+k4velmars)/6
        v=v+h*(k1velship+2*k2velship+2*k3velship+k4velship)/6

        rt=rt+h*(k1posearth+2*k2posearth+2*k3posearth+k4posearth)/6
        rm=rm+h*(k1posmars+2*k2posmars+2*k3posmars+k4posmars)/6
        r=r+h*(k1posship+2*k2posship+2*k3posship+k4posship)/6

        distancetoearth=norm(r-rt)
        distancetomars=norm(r-rm)
        breakconditionearth.append(distancetoearth)
        breakconditionmars.append(distancetomars)

        rearth.append(rt)
        rmars.append(rm)
        rship.append(r)
        vearth.append(vt)
        vmars.append(vm)
        vship.append(v)

        Psi_s=float(arctan(r[1],r[0]))
        Psi_m=float(arctan((r[1]-rm[1]),(r[0]-rm[0])))

        #arbitrary numbers, must be set in order to avoid errors
        #12 is arbitrary, so we know it wont be valid
        vsolution=12.
        rsolution=12.

        #postion of sail
        if((rm[0]-r[0])**2+(rm[1]-r[1])**2)**0.5<a:
            Theta=Psi_m-Psi_s

        else:
            Theta=Psi_s+np.pi/2

        #to make scatter faster, one each 500
        if i%500 ==0:
                #LIVE SCATTER
                plt.scatter(rt[0],rt[1],color='c')
                plt.scatter(rm[0],rm[1],color='red')
                plt.scatter(r[0],r[1],color='black')
                #print(norm(v-vm))
                plt.pause(0.005)

        #break if it enters the planet
        if distancetoearth<Rt:
            print('Collided with earth :(')
            break
        if distancetomars<10*Rm:
            print('Entered MARS')
            print('on iteration')
            print(i)
            print('with velocity')
```

```
                print(vm-v)
                vsolution=vm-v
                rsolution=norm(r-Rm)

                break

        #ADAPTATIVE STEP
        if distancetoearth<specialstepdistanceearth or distancetomars<0.1*
    specialstepdistancemars:
            h=float(10)

        elif distancetomars<specialstepdistancemars:
            h=float(100)

        else:
            h=float(1000)

        #end for

    minvalue=min(breakconditionmars)
    print(minvalue)
    indexminvalue=breakconditionmars.index(min(breakconditionmars))
    print(indexminvalue)
    velocityatmin=vship[indexminvalue]
    marsvelocity=vmars[indexminvalue]

    rearth=np.matrix(rearth)
    rship=np.matrix(rship)
    rmars=np.matrix(rmars)

    plt.figure()
    plt.scatter(rearth[:,0],rearth[:,1],color='red')
    plt.scatter(rmars[:,0],rmars[:,1],color='blue')
    plt.scatter(rship[:,0],rship[:,1],color='black')

    return minvalue,velocityatmin,marsvelocity,rmars,indexminvalue,rship,vsolution,
    rsolution

###########################################
#                 SCRIPT                  #
###########################################

plt.close('all')

G=float(6.67428e-11)
c=float(299792458)
Au=float(149597870700)

#Planets conditions
Ms=float(1.989e30)
Mt=float(5.9723e24)
MM=float(0.64171e24)
m=2000

leo=float(2e7)
Rt=float(6.371e6)
Rm=float(3.390e6)

```

```
albedo=0.25
Ls=float(3.846e26)

#numberiterations=35000
numberiterations=30000
divisions=1
#lpha=np.linspace(2.0995,2.0995,divisions)
alpha=2.69

vesc=(2*G*Mt/(Rt+leo))**0.5
vti=np.array([0,float(29780)])
vmi=np.array([0,float(26500)])
vi=np.array([vesc*np.sin(alpha),vti[1]+vesc*np.cos(alpha)])
rti=np.array([float(152.098e9),0])
rmi=np.array([float(206.655e9),0])
ri=np.array([rti[0]+Rt+leo,0])

specialstepdistanceearth=norm(rti-rmi)/500
specialstepdistancemars=norm(rti-rmi)/8

Avec=np.array([1700,1800,1900])/7e-03
acond=[norm(rti-rmi)*0.6215,norm(rti-rmi)*0.6415,norm(rti-rmi)*0.6615]

minimumdistance=np.zeros([3,3])
velocitiatminimum=np.zeros([3,3])
marsvelatmin=np.zeros([3,3])
solution=np.zeros([3,3])

#Different conditions optimization.
#Change 3 for any N arbitrary natural
for j in range(3):
    for k in range(3):
            print('la iteracio es')

            if j==1 and k==1:
                minimumdistance[j][k]=8.997*Rm #what we get with this initial value
            else:
                A=Avec[j]
                a=acond[k]
                print(a,A)
                plt.figure()
                [auxdist,auxveloc,auxmarsvelocity,rmars,indexminvalue,rship,
    vsolution,rsolution]=kernel(numberiterations,rti,rmi,ri,vti,vmi,vi,A,a)
                minimumdistance[j][k]=auxdist

velocitiatminimum=np.array(velocitiatminimum)
minimumdistance=np.array(minimumdistance)
marsvelatmin=np.array(marsvelatmin)
```

Appendix D: Mars Reference Frame code

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
```

```
UA = 149597870700.


# Sun

#( mass, radius )
sun_data = [1988500.*10**24,  695700.*10**3/AU ]


mars_data = np.array([ 0.00487279, 3.21250817e-6, 249.23*10**9/AU, 0.64171*10**24,
    3396.2*10**3/AU ,1.   ,0.   ])

# eccentricity, periheli, afeli, mass, radius(equator), [x_0,y_0]
# (    , m, m, kg, m )

a = mars_data[1] / (1.- mars_data[0])
b = sqrt( (1 + mars_data[0]) / (1- mars_data[0]) ) * mars_data[1]

unit1=np.array([0,1])
unit2=np.array([1,0])
e=mars_data[0]
G=6.674*10**(-11)/AU**3
T=665.*24*3600
m=2000.
c=2.9979e8
albedo=0.25
mM=mars_data[3]
Ms=sun_data[0]
RM=mars_data[4]
A=1800./(7e-3)
RS=sun_data[1]
Ls=3.846e26
mini=2.*RM
def Rs(t):
    '''
    t : array of the times we want

    It creates an array for the planet chosen
    '''

    parenthesis = sqrt( G*(mars_data[3] + sun_data[0] )  ) / a**(3./2.)

    x = a * ( mars_data[0] + cos( parenthesis * t ) )
    y = b*sin( parenthesis * t )

    x = -sqrt(x**2+y**2)
    y = 0.

    return np.array([x,y])

def w(t,Rs):
    return a*b*2*np.pi/(T*norm(Rs(t)))

def wp(t,Rs):
    return (-a*b*2*np.pi/T)*(-w(t,Rs)*e*a**2*sin(w(t,Rs)*t)+(b**2-a**2)*sin(w(t,Rs)*
    t)*cos(w(t,Rs)*t))/norm(Rs(t))**(3./2.)

```

```python
def Lm(t,Ls,Rs):
    return Ls*RM**2/(4.*Rs(t)**2)

def Rp(t):
    '''
    t : array of the times we want

    It creates an array for the planet chosen
    '''
    a = mars_data[1] / (1.- mars_data[0])
    b = sqrt( (1 + mars_data[0]) / (1- mars_data[0]) ) * mars_data[1]

    parenthesis = sqrt( G*(mars_data[3] + sun_data[0] )  ) / a**(3./2.)

    x = a * ( mars_data[0] + cos( parenthesis * t ) )
    y = b*sin( parenthesis * t )

    x = -sqrt(x**2+y**2)
    y = 0.

    return np.array([x,y])


def Rpp(t,Rp):
    return np.array([-Rp(t)[1],Rp(t)[0]])

def f(t,vx,vy,r,Rs):#forces de gravitaci
    return -G*m*(mM*r/norm(r)**3-(Ms/norm(Rs(t)-r)**3)*(Rs(t)-r))-w(t,Rs)*w(t,Rs)*a*e*(np.dot(unit1,Rp(t)/norm(Rp(t))))*unit2-(np.dot(unit2,Rpp(t,Rs)/norm(Rp(t)))*unit1)+2*w(t,Rs)*(vy*unit2-vx*unit1)+wp(t,Rs)*(r[1]*unit2-r[0]*unit1)+r*w(t,Rs)*w(t,Rs)

def F(x,y,vx,vy,t):
    d = sqrt((x**2) + (y**2))  # distance between sail and x,y=(0,0) in m

    if x != 0:  # prevent division by zero if float x == 0
        phi = arctan2(y, x)  # angle in radians
    else:
        phi = 0.+np.pi*(np.abs(y)-y)/np.abs(y)

    def photon_function(alpha):
        F_alpha_r = Ls * A / (12 *m* np.pi * c * norm(Rs(t))**2) * (1 - (1 - (RM / d)**2)**(3. / 2.)) * cos(alpha) * albedo * (1.-x/d)
        F_x = F_alpha_r * cos(alpha + phi)
        F_y = F_alpha_r * sin(alpha + phi)
        nu = np.arccos((F_x * vx + F_y * vy) / \
            (sqrt(F_x**2 + F_y**2) * sqrt(vx**2 + vy**2)))
        F_x_final = F_alpha_r * cos(nu)
        return float(F_x_final)

    alphamin= sp.optimize.minimize(photon_function, x0=0., args=(), method='SLSQP', jac=None, bounds=[(-0.5*np.pi,0.5*np.pi)], constraints=(), tol=None, callback=None, options={'disp': False, 'iprint': 1, 'eps': 1.4901161193847656e-08, 'maxiter': 100, 'ftol': 1e-06})
    # Calculate F_x and F_y for best alpha_min
    asd= (Ls*A/(12.*m*np.pi*c * norm(Rs(t))**2)) * (1. - (1. - (RM / d)**2)**(3/2))*cos(alphamin)*albedo*(1.-x/d)
    F_x = asd * cos(alphamin + phi)
```

```
    F_y = asd * sin(alphamin + phi)
    return F_x[0], F_y[0], alphamin

steps=30000
dt=100.
t=0
def fly(
    x,
    y,
    vx,
    vy,
    mini,
    t,
    steps):
    """Loops through the simulation, returns result array"""
    r=np.array([x,y])
    # Data return array
    result_array = np.zeros((steps), dtype=[
                                ('step', 'int32'),
                                ('time', 'int32'),
                                ('px', 'f8'),
                                ('py', 'f8'),
                                ('ship_speed', 'f8'),
                                ('alpha', 'f8'),
                                ])
    result_array[:] = np.NAN
    for i in range(steps):
        t=t+dt
        vx=vx+f(t,vx,vy,r,Rs)[0]*t/m
        vy=vy+f(t,vx,vy,r,Rs)[1]*t/m



        photon_F_x, photon_F_y, current_alpha = F(
            x, y, vx, vy, t)


        if px > 0:
            vx += +photon_F_x / m * t
            vy += +photon_F_y / m * t
        else:
            vx += -photon_F_x / m * t
            vy += -photon_F_y / m * t

        # Update positions
        x += vx * timestep
        y += vy * timestep

                # Forces
        probe_velocity = sqrt(np.abs(vx)**2 + np.abs(vy)**2)

        if (x**2+y**2)<mini:
            print("Ha arribat a Mart")
            print(probe_velocity)
            break
        if (x**2+y**2)>20.:
            print("Nicetu, NOOOB!!!")
            break
```

```
        # Write interesting values into return array
        result_array['step'][step] = i
        result_array['time'][step] = i * dt
        result_array['px'][step] = px
        result_array['py'][step] = py
        result_array['ship_speed'][step] = probe_velocity
        result_array['alpha'][step] = current_alpha

    return result_array

Q=fly(-0.5*UA,-0.5*UA,10.,10.*sqrt(3),mini,t,steps)
print(Q)
```