# Open Cosmos Challenge Project.
# Mathematical Modelling & Simulation.

Alejandro de Miguel Mendiola.
alejandrode96@gmail.com

1st June 2021

## 1 Motivation

This paper tries to briefly summarise the thought process and to give an idea of how the challenge proposed by OpenCosmos has been approached.

Both this report and the code delivered are completely original and they have been personally and exclusively developed for this challenge.

## 2 Challenge project description

Considering a flat solar panel orbiting around the Earth, write a software that models the power generated by the panel over time. Assume that the solar panel has only one face covered by solar cells, and that this face is always pointing to the direction opposite to nadir (i.e. opposite to the satellite-Earth direction).

## 3 Assumptions and hypotheses

When developing this project, a series of assumptions have been made:

- An end-of-life (EOL) solar panel: which affects the performance and efficiency of the solar panel in terms of amount of energy that can be absorbed and converted. It is a worst-case scenario assumption just to be safe. Therefore, an EOL efficiency of 87% has been added on top of the beginning-of-life (BOL) efficiency of 29%, for an overall efficiency of 25% [1].
- Photovoltaic cells do not cover the entire panel area: it has been estimated that 90% of the area is covered by photovoltaic cells.
- Plane wavefront: because of the great separating the sun from the earth, it is safe to assume that all solar rays arriving to the solar panel are parallel to the sun-earth position vector. Even for GEO orbits, the angle between earth-sun/satellite-sun position vectors is less than $0.02[deg]$.

- Sun's radiation intensity is constant: with a value known as solar constant $S = 1367\ [W/m^2]$ [3], computed for a mean earth-to-sun distance of 1 AU. Differences in distance from the satellite orbit to earth are negligible, so it is safe to say that it is indeed pretty much constant.

- Angle of incidence of the solar rays over the solar panel affects its efficiency: this is mainly due to the effects of reflection. A first approach, and the one implemented in the simulation because of its simplicity, has been considering the following law: $P_{generated} = P_{max} \cos\theta$, where $\theta$ is the incident angle between the solar rays and the normal to the panel surface, and $P_{max}$ is the maximum obtainable power. It is easy to see that when the incident angle is $90[deg]$ the panel does not generate any power. On the other hand, for an angle of $0[deg]$ the efficiency of the solar panel is maximum. Doing some research, another law was proposed in [2] after carrying out some simulations of this effect. Figure 1 depicts the result of this study, in which it can be observed that the conversion efficiency of silicon solar cells is almost constant for angles $0 \leq \theta \leq 45[deg]$ and decreases by 1.7% at angles of approximately $60[deg]$. This law could be easily implemented by computing the function associated with these results by means of extrapolation.

- Orbit perturbations have not been considered: in real life there are many perturbations affecting the orbit of an object, such as the oblateness of the earth, air drag (when considering LEO orbits), solar radiation pressure (SRP) or third-body effects (such as the moon or the sun). It has been considered that implementing all of these perturbations are out of the scope of this challenge, and also the lack of information about the spacecraft size, dimensions, material or possible orbits are discouraging in order to carry out a proper study of these effects. Lastly, because the main goal of the challenge seems to be computing the power generated depending on a given orbit, all these effects that alter directly the orbit elements will surely complicate the study and the conclusions.

# 4 Code structure

The code has been designed to be clear, simple and flexible. Python has been the chosen programming language for this task because it is well known in the industry, flexible, simple and compatible. Its object-oriented capabilities have not been exploited in this case because it has been considered that they were not required in an assignment of this nature.

The code has been structured into several different files:

- *main_OC.py*: the file to be run. It imports all the files and packages necessary for the program to be executed and contains the logic of the program. It also plots the results of the simulation.

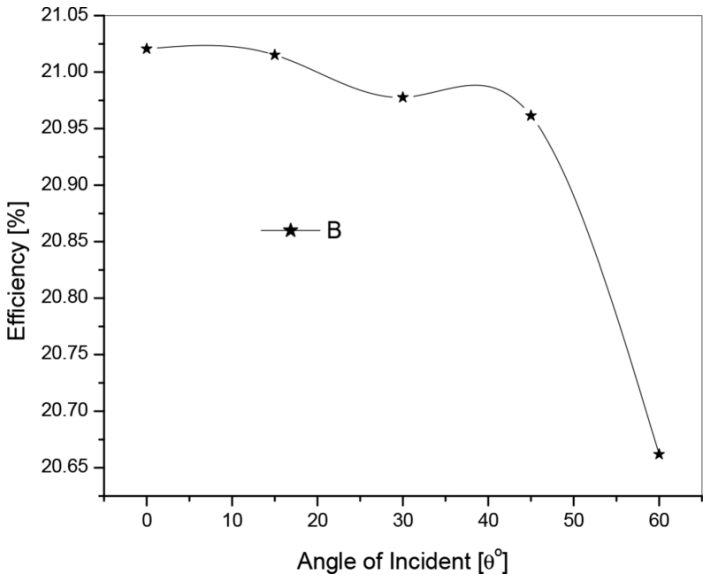


Figure 1: Variation of conversion efficiency of silicon solar cell as a function of angle of incidence of sunlight [2].

- *data_OC.py*: this file contains the data of the simulation. It is the key of the flexibility of the software, as it contains all the key parameters of the simulation and allows to easily change them. It is properly commented for easy understanding.
- *lineofsight.py*: it stores the function to compute if the satellite is eclipsed by earth (not in line of sight with the sun) or not. It returns a bool depending on the case and also the angle $\theta$ between the position vectors earth-sun and earth-satellite.
- *sun_position.py*: it contains a function to compute the sun geocentric position vector given a Julian date. The function is explained in [3]. It also computes the obliquity of the ecliptic and the apparent ecliptic longitude although they are not needed in this case.
- *JD.py*: defines the function to convert from a date in the format *year/month/day hour:minutes:seconds* to the Julian day equivalent.
- *kep2car.py*: file containing the function to convert from keplerian elements to cartesian coordinates. It is useful to be able to define an orbit according its keplerian elements and then convert this initial values to cartesian coordinates, more appropriate for propagating the orbit.
- *integrate_orbit.py*: to call for the function responsible for integrating the orbit given the initial state vector of the orbit in cartesian coordinates.

## 5 Results

The solution has been found with an easy procedure. First of all it is necessary to obtain the geocentric position vector of the solar panel for all times because as stated in the exercise, it indicates the direction of the solar panel normal surface vector. This is easily done by propagating the orbit around the earth for a specified time period (integrating equation 1). Figure 2 shows an example of a propagated orbit around the earth.

$$\ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}\boldsymbol{r} \tag{1}$$

The next step is implementing a function which returns if the satellite is in line of sight with the sun or if in the contrary it is being shadowed by the earth. In this way, we can completely define the instants the solar panel could be potentially generating power. From this function, the angle between the position vectors earth-sun and earth-satellite can be also returned. This angle will tell if, and in which way, the solar panel faces the solar rays. As the red rectangle in figure 3 depicts, it could happen that, despite the solar panel being illuminated by the sun, it is in fact unable to generate power because it is not oriented in the solar rays direction. Only when the angle becomes smaller than $90[deg]$ the solar array begins producing power. Notice the periodic trend in the graphs, as they have been plotted for 2 orbital periods. If this paper were considering perturbations, this would not be the case as the orbit would change over time depending on the predominant perturbations, which in turn would complicate the study and making conclusions.

## 6 Possible improvements and future work

This section will mention some ideas and possible improvements that could be implemented in order to increase the success of the mission (maximising the amount of power generated by the orbiting solar panel).

- Active sun pointing: even though it is a constraint of the challenge, adding this feature would allow the solar panel to directly face the sun rays more often. This improvement would be constrained by the characteristics and capabilities of the sun pointing system (e.g maximum rotational speed of the actuators, maximum rotational angle...)

- Orbit selection: the amount of time the solar panel is eclipsed by the earth depends on the orbit it is at. The key dependence orbital elements are $a$ and $i$, precisely the so called beta angle $\beta$ (the angle between the orbital plane of the satellite and the sun vector). Figure 4 shows the dependency of the sunlight received as a function of this angle.

  High inclination, high altitude orbits maximise the time a spacecraft spends in sunlight. Therefore, high polar orbits ($\beta = 90deg$) would be the best option. Another idea would be modify the shape of the orbit towards a high-elliptical orbit, in which the satellite spends a lot of time in a high altitude (close to apogee)

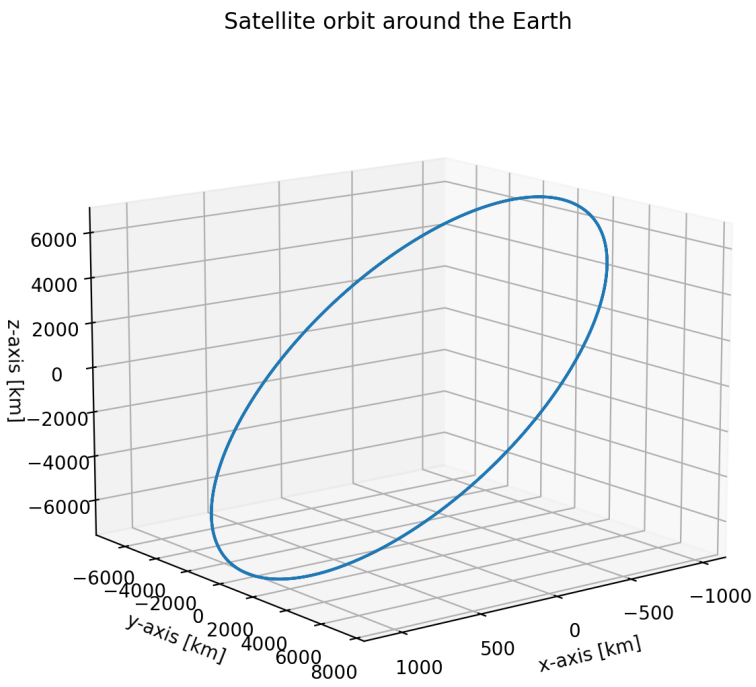


Figure 2: A propagated orbit.

and would travel fast through the perigee (inspired by Molniya orbits). However, a more in-depth study would need to be done as there are many factors that alter an orbit.

3 different types or orbit have been tested in the code, but with the idea of keeping this paper brief and simple they will not be discussed in depth. Anyway, the results support the reasoning explained above: higher and more inclined orbits usually enjoy of longer sunny periods.

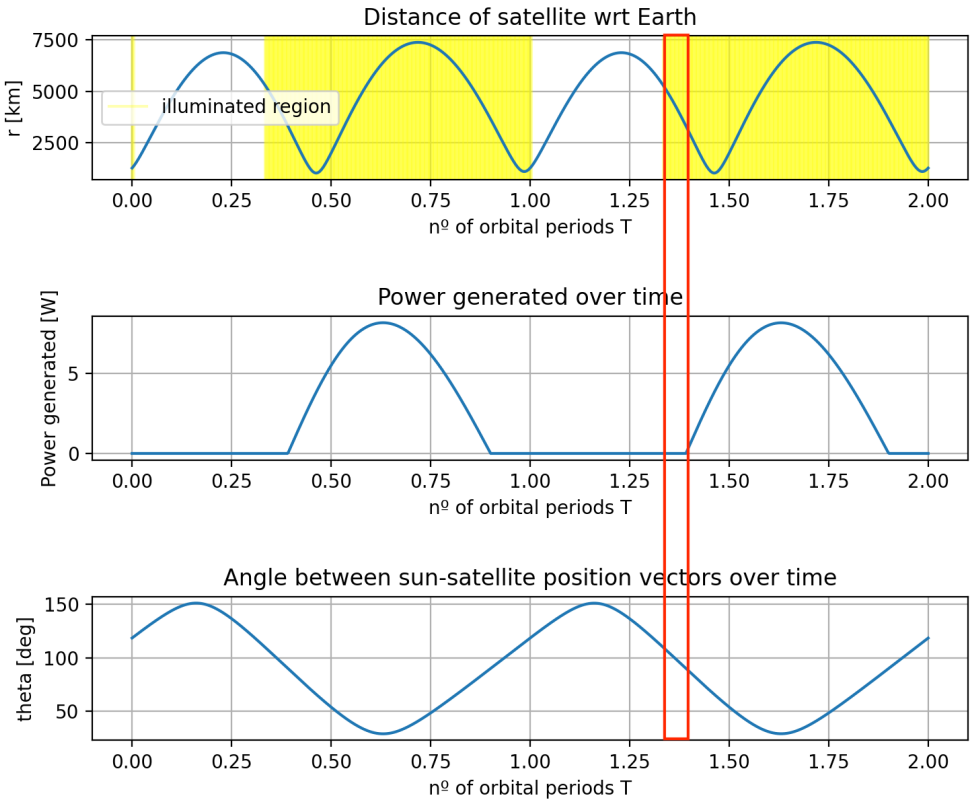


Figure 3: Plots depicting the illuminated periods, the power generated over time, and the angle between the solar rays and the solar panel.

## References


[1] *SatImaging*, information about space solar panels, `http://propagation.ece.gatech.edu/ECE6390/project/Sum2015/team3/PowerSystem.html`

[2] *Effects of obliquity of incident light on the performance of silicon solar cells.* R. Sharma, `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6611928/`

[3] *Orbital Mechanics for Engineering Students*, Howard D. Curtis,

[4] *Satellite orbital seasons*, Brown University, `https://blogs.brown.edu/umbricht/2018/10/09/satellite-orbit-seasons/`

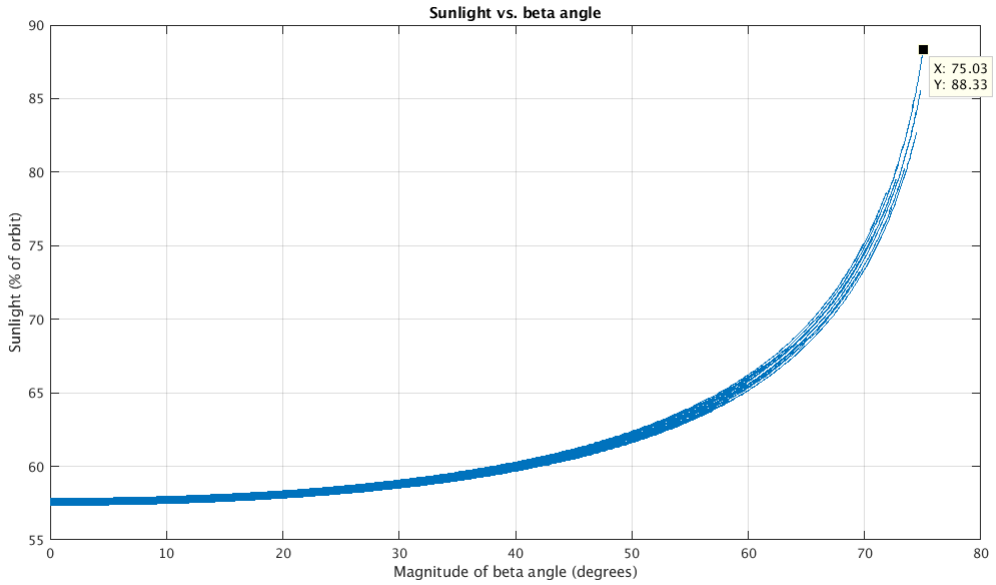


Figure 4: Relationship between the beta angle $\beta$ and the percentage of the orbit period spent in sunlight [4].