Space Debris

1 Assignment Description

These assignment sheets describe the modelling of space debris.

The assignment consists in writing an essay that contains an answer to each question and which expand on the material included in these notes. The references are there to provide you with extra source of information and they are not exhaustive. When answering mathematical derivations, provide enough details so that we can asses that you fully understand them (They must be understandable by your peers without needing to do any derivation on the side).

The essay must be readable on its own without reference to the assignment sheets. The essay needs not necessarily to follow the same order as the notes and does not need to answer the questions in the same order either. As a matter of fact using a different structure and order is a bonus. Do describe the interpretation or consequences of the results that you obtain. We also expect you to focus more on some aspects of the material presented in these notes. This could be a more detailed discussion of the derivation of the equations or algorithm described in the notes or to better explain the interpretation of the results. You can also solve problems which are not set and which answer some further questions that you might have.

Some part of the assignment sheet will need to be included in your essay, but do not copy these section verbatim, use your own words. The equations do not need to be changed, but you can provide more detailed explanations or derivations when appropriate.

You have to submit 4 files: 1 pdf file for the essay and 3 python programs. We ask that you typeset your essay in LaTeX, using the provided template file `debris_template.tex` and LaTeX style file `mm2.cls`. We expect the essay to be approximately 3000 words long.

2 Introduction

Orbital debris are leftovers from old satellites or rockets and they orbit the Earth like all satellites. They are a major problem because when a piece of debris hits a satellite it is very likely to damage it, especially if the relative speed between the piece of debris and the satellite is large. They vary in size from micrometers to several centimetres[1].

The questions one needs to ask is how long it takes them to be absorbed by the Earth atmosphere and also if there is a way to clean space of debris. This project will attempt to answer these 2 questions, trying to collect debris by manoeuvring a dedicated spacecraft that will collect them.

Making a satellite reach an object located on another orbit is called a rendez-vous and is not as simple as one would think. NASA discovered this on June 3, 1965, when Gemini 4 astronaut Jim McDivitt tried to manoeuvre his craft close to the upper stage of the Titan II rocket that launched the space capsule. The astronaut was unable to get close enough to achieve station-keeping, mostly because NASA had not yet learned the orbital mechanics involved in the process. Naively, one would simply point the active vehicle's nose at the target and thrust, but this does not work. If the target is ahead in the orbit and the tracking vehicle increases speed, its altitude also increases and as a result moves away from the target. As it reaches a higher altitude the vehicle's velocity decreases and as a result the space capsule ends up above and behind its target.

3 Orbital Dynamics

3.1 Newton equations

The dynamic of an object orbiting around the Earth is described by Newton´s equations

$$\begin{aligned} m\ddot{x} &= -\frac{GM_E m}{x^2+y^2}\frac{x}{\sqrt{x^2+y^2}} + F_x(t) \\ m\ddot{y} &= -\frac{GM_E m}{x^2+y^2}\frac{y}{\sqrt{x^2+y^2}} + F_y(t), \end{aligned} \tag{1}$$

where $G = 6.673\,10^{-11} N m^2 kg^{-2}$ is Newton's constant, M_E the mass of the Earth, m the mass of the spacecraft. F_x and F_y are the forces exerted on the object along the x and y axis respectively. These forces can come from the thrusters of a spacecraft or the frictions forces due to the very thin atmosphere or even solar wind. These equations are easier to solve in polar coordinates

$$x = r\sin(\theta), \qquad y = r\cos(\theta), \tag{2}$$

as the equations then become:

$$\ddot{r} = -\frac{GM_E}{r^2} + r\dot{\theta}^2 + \frac{F_r}{m} \tag{3}$$

$$\ddot{\theta} = -2\frac{\dot{\theta}\dot{r}}{r} + \frac{F_\theta}{mr}, \tag{4}$$

where $F_r = F_x\sin\theta + F_y\cos\theta$ and $F_\theta = F_x\cos\theta - F_y\sin\theta$.

Question 1:

Derive (3) and (4) by inserting equations (2) into (1)).

We then notice that if $F_r = F_\theta = 0$, the simplest solution is given by

$$r = r_0, \qquad \theta = \omega_0 t + \theta_0 \tag{5}$$

which corresponds to a circular orbit of period

$$T_0 = \frac{2\pi}{\omega_0}, \qquad \omega_0 = \sqrt{\frac{GM_E}{r_0^3}}, \tag{6}$$

where r_0 and ω_0 are both constant.

Question 2:

a) Derive the expression for the speed of a piece of debris as a function of G, M_E and r_0, assuming it follows a circular orbit. (Hint, use (6)).

b) Using the data given in the section 6 , *parameter values*, below, compute the time it takes a piece of debris to go around the Earth when they are on circular orbits of 330km and 435km above the Earth surface (which corresponds to respectively the lower and higher orbits of the International Space Station (ISS)). Provide the answers both in seconds and in minutes. Does your answer look reasonable?

In an orbital rendez-vous, the spacecraft starts the manoeuvre when they are only a few kilometres apart. In equation (3) and (4) the radius r will be of several thousand kilometres and the angle θ will cover the full circle while the angular separation between the 2 spacecraft during the manoeuvre will be a small fraction of a degree. This could lead to numerical inaccuracies but also make it harder than necessary to track the relative position between the 2 spacecrafts.

To solve this problem, we will assume that the target piece of debris is on a circular orbit of radius $r_0 = R_E + h$ where h is the altitude of the orbit above the Earth surface. Moreover the piece of debris will be orbiting the Earth at the angular speed ω_0 given by (6) and its trajectory will thus be given by

$$r = r_0, \qquad \theta = \omega_0 t. \tag{7}$$

In what follows, we will refer to it as the reference trajectory.

For the active spacecraft we will then use the following relative coordinates

$$r = r_0 + z(t), \qquad \theta = \omega_0 t + \phi(t). \tag{8}$$

Inserting (8) into (3) and (4) we get

$$\ddot{z} = -\frac{GM_E}{(r_0+z)^2} + (r_0+z)(\omega_0+\dot{\phi})^2 + \frac{F_r}{m} \tag{9}$$

$$\ddot{\phi} = -2\frac{(\omega_0+\dot{\phi})\dot{z}}{(r_0+z)} + \frac{F_\theta}{m(r_0+z)}. \tag{10}$$

Question 3:

Convert the pair of second order ordinary differential equations (9), (10) into a system of 4 first order ordinary differential equations.

Question 4:

Derive the expression for the distance between the spacecraft (8) and the reference trajectory (7).

Check your answer by checking the following special cases for which d can be easily computed:

- $z = 0, \phi = 0.$
- $z = a, \phi = 0.$ (What is the expected distance?)
- $z = 0, \phi = a.$ Assume a very small so that you can compute a first order Taylor expansion to check the result. (What is the expected distance?)
- $z = 0, \phi = \pi/2.$
- $z = a, \phi = \pi$ where a is an arbitrary distance.

Show that your expression matches the five simple case described above.

3.2 Python code

Coding task 1:

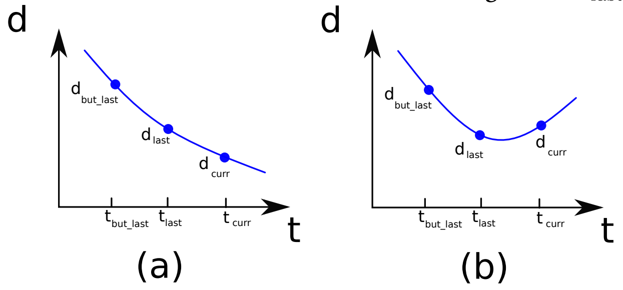
Write a python program, called `gravity.py`, to solve the system of 4 equations derived in question 3 using the class `ODE_RK4` from the module `ode_rk4.py`. The class `ode_rk4.py` is identical to the one used during the *Cyclotron* practical classes, except that we have added a class function called `post_integration_step()` which is executed after every integration step (more about this later).

Proceed as follows:

- Copy the supplied program `gravity_template.py` into `gravity.py`. It defines a class called `Gravity`, which is a subclass of the class `ODE_RK4`.
- Modify the method F so that it solves the 4 first order differential equations which you have derived in question 3. For the time being, assume that `F_r` and `F_theta` are constant. `self.V[0]` must correspond to z, `self.V[1]` to dz/dt, `self.V[2]` to ϕ and `self.V[3]` to $d\phi/dt$.
- You must complete the definition of the method `dist_2_reference` which computes the distance between the spacecraft and the reference trajectory that you have computed in question 4. The position of the spacecraft is to be taken from `self.V` .

- Complete the function `post_integration_step(self)`. [This function will only be used as part of coding task 2]. It must determine when the spacecraft has reached a local minimum distance, to the target.

  To find the minimum distance, one can use the distance evaluated at the current integration step , d_{curr}, as well as the values obtained for the previous 2 steps: d_{last} and d_{butlast}. When d_{last} is smaller than both d_{curr} and d_{butlast} we have a local minimum. This is illustrated on the figure below. On figure a, $d_{\text{butlast}} > d_{\text{last}} > d_{\text{curr}}$ and there is no local minimum. On figure b, $d_{\text{butlast}} > d_{\text{last}} < d_{\text{curr}}$ and d_{last} is an approximation of the local minimum occurring at $t = t_{\text{last}}$.



  `post_integration_step()` is automatically run after each integration step and it must first call `dist_2_reference` to compute the current distance between the spacecraft and the piece of debris. The values class variables `d_last` and `d_butlast` should be used to keep track of the past distances (They are both initialised in `__init__()`). After testing for local minima, the value of `d_last` must be copied into `d_butlast` and the distance you have just computed must be copied into `d_last`. The current time is the class variable `t` and the class variable `t_last` can be used to keep track of the time when the last distance was evaluated. Notice that you can only check for local minima after the 3rd integration step, once you have 3 values of the distance.

  When a local minimum is found, the list `[t , d]`, corresponding to the minimum distance and the time when it was reached, must be appended to the list `d_min`. Ultimately, this list will contain all the local minima and the time when they were reached: `[[t1,d1], [t2,d2], ...]`.

- Complete `min_min(self,t_after)`. [This function will only be used as part of coding task 2]. If must return the list `[t , d]` from the class variable `d_min` (set in by `post_integration_step`) corresponding to the smallest distance in the list, but ignoring all the minima occurring before `t_after`.

To test your program run the supplied program `run_gravity.py`. It imports the `gravity.py` module that you have just written to integrate the equation for over 2 full periods for the 3 cases described below. It outputs 3 figures for each of them: z as a function of t, ϕ as a function of t and the trajectory z as a function ϕ.

The 3 test cases considered, all using $h = 500km$ as well as $F_r = F_\theta = 0$, are

- a) The reference trajectory with initial conditions $z(0) = \dot{z}(0) = \phi(0) = \dot{\phi}(0) = 0$. z and ϕ should remain zero within the numerical accuracy. (Check the figure scales carefully. Numerical errors often result in very odd figures but they are on an very tiny scale.)

- b) A circular orbit at a different altitude, $z(0) = 1km$, than the reference trajectory. The initial conditions are $z(0) = 1000$, $\dot{\phi}(0) = \sqrt{\frac{GM_E}{(r_0+z_0)^3}} - \omega_0$, $\phi(0) = \dot{z}(0) = 0$. As this is a different circular orbit z should remain the same, within the numerical accuracy, and ϕ should decrease linearly as the period of that high orbit is larger.

- c) A non circular orbit with initial conditions $z(0) = \phi = \dot{z} = 0$, $\dot{\phi} = 100/(r_0 + z_0)$. This corresponds to an elliptical orbit where the initial condition is the perigee. We expect thus z to remain positive and to become 0 after every full orbit. ϕ on the other hand will drift like in the case ”b” above, but the drift will not be linear in that case.

4 Rendez-vous:

We will now go back to our original problem of a rendez-vous in space between a target, a piece of debris, and a cleaning spacecraft of mass $m = 4000kg$. The piece of debris will be on the reference circular orbit at an altitude $h = 400km$ above the Earth. The spacecraft has 4 small thrusters that it can use for manoeuvres (we consider that the spacecraft, the piece of debris and their orbits are all confined to the same plane). Two of the thrusters push the spacecraft parallel to its orbit; this corresponds to the term F_θ in eq (4). The other thrusters push the spacecraft radially and this corresponds to F_r in (3). A positive value for F_r pushes the spacecraft on a higher orbit while a negative one pushes it downwards, activating a different thruster. Similarly a positive F_θ pushes the spacecraft forward while a negative one pushes backwards. We also assume that the orientation of the spacecraft does not change and each thruster can exert a maximum force of $100N$.

The spacecraft will be initially on a circular orbit 1km lower than the piece of debris and about 2km behind:

$$\begin{aligned} z &= -1000, \\ \dot{z} &= 0, \\ \phi &= -\frac{2000.0}{r_0}, \\ \dot{\phi} &= \sqrt{\frac{GM_E}{(r_0+z)^3}} - \omega_0 \end{aligned} \tag{11}$$

The rendez-vous problem consists in determining how one can fire the thrusters to bring the spacecraft within 1 meter of the piece of debris (the reference orbit) so that it can catch it. To reduce the number of parameters we will assume that the thrusters will only be fired from $t = 0$ up to t_{thurst}. Both F_r and F_θ will each have a fixed, but not necessarily equal, value during that time and they will both start and stop at the same time. The total amount of fuel used will be equal to Fuel $= (|F_r| + |F_{theta}|)t_{trust}$ in units $kg\, m/s$.

Coding task 2:

Write a program called `spacecraft.py` that will compute the trajectory of the spacecraft. As we have to change the equation and add some extra functionality, we will create a subclass of the Gravity class. Proceed as follows:

- Copy the file `spacecraft_template.py` into `spacecraft.py`.
- Copy the function `F(t,v)` from your previous program, `gravity.py`, but make the force time dependant as described above: `F_r` and `F_theta` must be null after the time `t_thrust`. This is similar to what was done during the *cyclotron* practical except that here the force is being turned on or off based on the current time, `t`, rather than the position of the object.
- Write a method called `min_dist_to_target(F_r,F_theta,t_thrust,tmax,dt)` which computes the trajectory of the spacecraft for the specified values and duration of the thrusters and determines the closest distance to the piece of debris that the spacecraft achieves for these values, as well as the time it takes to reach that nearest point.
  - One must first store the value of `F_r`, `F_theta` `t_thrust` in class variable so that they are available from the method `F` you have just written
  - You must then fill in the initial condition corresponding to (11) using the `reset()` class function.
  - Use the `reset` class function to set the initial condition, `dt` and `t0`. Then call the `iterate` class function to integrate the equations.
  - Use the class function `min_min` to return the list `[t_min, d_min]` containing the minimum distance `d_min` between the spacecraft path and the piece of debris, and the time `t_min` needed to reach that point. Remember that the method

`min_min(t_after)` was written as part of coding task 1. The parameter `t_after` is useful to find minima that occurred after a full orbit, something which you might find useful to do during your investigations.

- The supplied program `run_spacecraft.py` uses the module `spacecraft` to integrate the equation for the spacecraft from $t = 0$ to $t = 4000$ using $F_r = 50$, $F_\theta = 100$ and $t\_thrust = 100$. It then outputs the shortest distance between the spacecraft and the piece of debris during its trajectory, the time when this distance is reached and the amount of fuel used.

  Expand the code so that it also computes and outputs the same quantities for $F_r = 25$, $F_\theta = 50$, $t\_thrust = 200$ and then $F_r = 10$, $F_\theta = 20$, $t\_thrust = 500$. Notice that the total amount of fuel is the same in the three case. Modify the code so that the figures of the trajectory are output on the same graph using the colour blue for the first values, green for the second and red for the last.

  The code that you must submit must output the data for the 3 scenario described above and generate a single figure showing the orbits for the 3 cases. (Add code at the end of `spacecraft.py` that is only executed when the program is run, *i.e.* not when it is loaded as a module. The code to generate the figure with the 3 trajectories is very similar to `run_spacecraft.py` except that it must generates data for 3 different trajectories.)

The code output should look like this

```
Fr=50 Ftheta=100.0 t_thrust=100
dmin=..., tmin=... Fuel=...
Fr=25 Ftheta=50.0 t_thrust=200
dmin=..., tmin=... Fuel=...
Fr=10 Ftheta=20.0 t_thrust=500
dmin=..., tmin=... Fuel=...
```

with the correct numbers instead of . . . and it must also generate one figure with the 3 trajectories.

If your code is correct, the time taken to reach the nearest point to the target should be between 8 and 12 minutes in each case (the times output by the program are expressed in seconds).

Once you have tested it, make a copy of the program `run_spacecraft.py` so that you can modify it to answer the questions below. We will now try to accelerate the spacecraft so that it reaches the piece of debris. We will not worry about the speed the spacecraft when it reaches the piece of debris as we assume it can *swallow it* at any speed. We just want to reach the piece of debris within a meter.

Question 5:

a) The problem we want to solve now is how to manoeuvre the spacecraft to reach the piece of debris by just giving it a steady push for a finite amount of time. In the 3 cases considered in the previous coding task, we have manoeuvred the spacecraft directly towards the piece of debris. If you keep $F_r = 50$, $F_\theta = 100$ and adjust t_thrust, how close can you get to the piece of debris within 15 minutes after thrusting?

b) Is it not possible to reach the piece of debris by aiming straight at it? Why is this so?

c) Try to reach the piece of debris by adjusting F_r and t_thrust , but keeping $F_\theta = 100$. Take `tmax=4000`. Can you find a value for which the spacecraft reaches the piece of debris? Here is a set of values that works: $F_r = 32N$ $F_\theta = 100N$, $t_{thrust} = 266s$ which requires $35112 kg\, m/s$ of fuel. Can you adjust these parameters to find a trajectory that requires less fuel (assuming the target must be reached within 90min or so)? Produce a figure showing the trajectory of the best orbits that you have found (the one requiring the least fuel).

Question 6:

a) Based on the results obtained in the question above, can you determine what the best strategy is to capture the piece of debris if one does not assume that the spacecraft activates its thrusters when it is 2km behind the piece of debris, but is free to chose the best time to do so? What is the amount of fuel needed in that case?

b) Assuming that the ejection speed of the thruster's gas is 1km/s, what is the mass of fuel needed to capture a piece of debris on an orbit 1km above that of the spacecraft (remember the fuel is expressed in kgm/s)? If the spacecraft has 500kg of fuel could one capture many pieces of debris in the range 300 to 500 km above the earth surface? Discuss the various limitations of the assumptions that were made.

5 Atmospheric Capture of Debris

The Earth atmosphere extends a long way into space, even if it is very thin beyond a few kilometres above the Earth surface [4]. Because they travel at very high speed, debris are progressively slowed down by the high atmosphere until they reach the denser part of it where they are eventually burned out. The question then is how long this process takes. This obviously depends on the orbit of the piece of debris and its shape and size. We will thus try to estimate what the friction force is and how the altitude of debris might decrease as a result. For simplicity we will assume that debris are on a circular orbit all the time and, as the friction force are expected to be very small, that the orbit radius will decrease very slowly. We will thus compute the energy of debris and the rate at which this energy is dissipated by the atmosphere.

The speed of a piece of debris on a circular orbit was computed in question 2. Calling $r_h = R_E + h$ the radius of the orbit of the piece of debris, its gravitational and kinetic energies are respectively

$$P_h = -\frac{GmM_E}{r_h}, \qquad K_h = \frac{GmM_E}{2r_h} \tag{12}$$

and its total energy

$$E_h = P_h + K_h = -\frac{GmM_E}{2r_h}. \tag{13}$$

(In the essay you should derive the expression for K_h.)

As derived in the Appendix 2, the speed of atmospheric particles at 300km above the Earth surface is smaller than the orbiting speed of a satellite. So to estimate the friction force on a piece of debris we can thus assume that the atmospheric molecules are at rest and that their relative speed with the piece of debris is the orbital speed of the piece of debris. As it travel through this low density atmosphere, debris will collide with $v_h A\rho_{At}$ kg of molecules (if ρ_{At} is expressed in kg/m^3) where A is the area cross-section of the piece of debris and ρ_{At} the atmosphere density. When a molecule hits a piece of debris it will bounce from it with a speed equal to the initial relative speed between the two objects but with the opposite sign (this is because the molecules are much lighter than debris). As it does so, the moment gained by the molecules is lost by debris and the resulting force on the pieces of debris is given by

$$F_{fr} = 2A\rho_{At}v_h^2. \tag{14}$$

The energy loss due to this force is equal to the force time the distance travelled and so the power dissipation is then given by the product of the force times the speed of the piece of debris

$$\frac{dE_h}{dt} = -2A\rho_{At}v_h^3. \tag{15}$$

The average density of the high atmosphere can be found in [4] and it can be fitted reasonably well with the function

$$\rho_{At}(h) = ae^{-hl} + B\left(\frac{h}{h_0}\right)^{-\sigma} \tag{16}$$

where h is expressed in km, ρ in kg/m^3 and

$$a = 1.946\,kg/m^3, \quad l = 0.15/km, \quad B = 8.82 \times 10^7 kg/m^3, \quad \sigma = 7.57, \quad h_0 = 1km. \tag{17}$$

Coding task 3:

The file `data/AthmDensity.csv` contains experimental data of the atmospheric density at very high altitude. The first column contains the altitude in kilometres and the second the density kg/m^3. Write a python program called `fit_high_atmosphere.py` which reads that file and fit the data with the function (16) for the parameter a, l and σ, but fixing $B = 8.82 \times 10^7$ (the python fitting function can't determine B).

The program must print on the screen the value of

```
a= ...
l= ...
s= ...
```

with the values of the parameter (which will differ slightly from the values given above). It must also generate a logarithmic plot of $\rho(h)$ showing the data point as blue stars and the fitted curve as a red line. (The figure scale must be logarithmic for ρ and linear for h. There is a `pyplot` function that lets you do this.) The label must be h for the x axis and $dens$ for the y axis.

The data file can be read with the following code:

```
data = np.genfromtxt("data/AthmDensity.csv", delimiter=",", skip_header
    =1)
```

This reads the file as an array, skipping the first line of the file which contains the names of the columns. `data` is now a list of the type `[[h1,d1], [h2, d2], ... ,[hn,dn]]` where the `hi` are the altitude and the `di` the corresponding atmosphere density.

We can then convert (17) to measure h in meters and we get

$$a = 1.946\,kg/m^3, \quad l = 1.5 \times 10^{-4}/m, \quad B = 4.63 \times 10^{30} kg/m^3, \ \sigma = 7.57, \ h_0 = 1m. \tag{18}$$

(For the questions below, use these values rather than converting the ones you have obtained for question 3).

Question 7:

Use (13) and the expression of v obtained for question 2 to express v as a function of E_h.

Use (13) to compute dE/dt as a function of dr_h/dt and then (15) as well as the expression that you have just derived for v and $h = r_h - R_E$, to express dE/dt as a function of G, M_E and r_h, as well as R_E, A and $\rho_{At}(h)$.

The use (13) to express r_h as a function of the energy E_h and compute $dr_h/dt = dh/dt$ as a function of $A, \rho_{At}(h), G, M_E, R_E, m$ and h. Notice that (13) is only valid for circular orbits, but has the orbits of debris decrease very slowly, this is a very good approximation.

This gives a 1st order separable differential equation for $h(t)$ which you can integrate from $h = h_0$, the initial altitude, to $h = 0$ the surface of the Earth. To be able to do this you can use $r_h \approx R_E$ and $\rho_{At}(h) \approx Bh^{-\sigma}$. This last approximation consists in underestimating the density of the atmosphere below $150km$ and so will lead to overestimating the time of re-entry by a very small amount.

The result will give you an estimate for the time of re-entry of a piece of debris in the atmosphere as a function of its orbital altitude and the ratio m/A.

Consider next the following 4 types of debris:

- A plain cube of density ρ and edge length L with cross-sectional area $A = L^2$. Consider an aluminium bolt $\rho = 2700 kg/m^3$ and $L \approx 1cm$.
- A rod of length L and square cross section l and density ρ. The average areas is $A \approx L \times l/2$, $m = l^2 L\rho$ and $m/A = 2\rho l$. So for $L = 10cm$, $l = 1cm$ and aluminium we have $m/A \approx 50$.
- A square aluminium plate of length L and thickness l has an average area $A \approx L^2/2$. Take $l = 1mm$.
- Gemini spacecraft with mass $m = 3850kg$ and a larger diameter of $3m$.

Create a figure of $t(h)$ for the 4 different values of m/A above using the colours black, green, blue and red in that order. The time $t(h)$ should be given in years on a logarithmic plot and h in km ranging from 300km to 1000km.

In [5] NASA quotes the following: *Debris left in orbits below 600 km normally fall back to Earth within several years. At altitudes of 800 km, the time for orbital decay is often measured in decades. Above 1,000 km, orbital debris will normally continue circling the Earth for a century or more.*,

How good is our prediction?

6 Parameters values:

- Gravitational constant $G = 6.673\,10^{-11} Nm^2kg^{-2}$
- Earth´s mass $M_E = 5.97\,10^{24} kg$
- Earth´s radius $R_E = 6370km$

6 References

[1] https://iaaweb.org/iaa/Studies/orbitaldebris.pdf

[2] https://www.nasa.gov/feature/facts-and-figures

[3] https://www.livescience.com/32583-how-big-is-the-international-space-station.html

[4] http://www.braeunig.us/space/atmos.htm

[5] https://www.nasa.gov/news/debris_faq.html

[6] http://wordpress.mrreid.org/2014/08/01/the-composition-of-earths-atmosphere-with-elevation

7 Appendix 1: Debris collision rate with a satellite

The probability of collision between a satellite and debris of density ρ_d can be estimate at follows: a satellite of cross sectional area A travelling at speed v sweeps a volume Av per unit of time and the number of collision per unit of time is thus

$$k = \rho_d AV. \tag{19}$$

If the speed is given in m/s, The number of collisions per year is thus given by

$$N = \rho_d AV 3.1536\,10^6. \tag{20}$$

As an example, the International Space Station is the largest artificial satellites and it has a cross sectional area of about $2500m^2$ [2],[3] while its orbit is approximately $400km$.

Its orbital speed is thus $V_{ISS} = 7670m/s$. NASA states [4] that the typical collision speed between satellites and debris is approximately $10km/s$. The debris density at $400km$ is approximately $\rho_d = 10^{-9}/km^3 = 10^{-18}m^{-3}$ [5].

The number of collisions per year is thus given by

$$N = 0.0006 \tag{21}$$

or approximately once every 15 years. Fortunately most of the ISS area consist in solar panels so the problem is not as critical as it might seem but this explains why NASA is carefully monitoring all the orbital debris using radars.

8 Appendix 2: Thermal speed versus satellite speed

At 300km, the atmosphere is mostly made out of Nitrogen, Helium and oxygen [6]. and the density of gas is approximately $\rho = 10^{-11}kg/m^3$ [4]. Moreover, the temperature at high altitude is approximately $1500K$ at 300km and drops to $300K$ around $100km$.

At such altitude, the average speed of a gas molecules in one direction can then be estimated using

$$\frac{k_B T}{2} = \frac{mv^2}{2}, \tag{22}$$

where $k_B = 1.38 \times 10^{-23}J/K$ and T is the temperature in Kelvin. As for Nitrogen $m_N = 2.32 \times 10^{-26}kg$ we have $< v_N > \approx \sqrt{k_B T m_N} \approx$ 950m/s. On the other hand, using (6) the orbital speed at $300km$ is $v_h \approx 7700m/s$, or in other words, much larger than the average thermal speed of gas molecule.

9 To submit:

- The file `essay.pdf` containing your essay.
- Python code 1: `gravity.py`
- Python code 2: `spacecraft.py`
- Python code 3: `fit_high_atmosphere.py`