Modelling Satellite/Lander Behaviour And Simulating The Rosetta Mission

Yaojia Huang& 01213992

December 3, 2017

Abstract

The purpose of this project is using Python to model how a satellite/lander behaves under the gravitational force of a single celestial body, which is Mars during the first stage. The final programme gives a general result for such systems of various initial conditions. In the simulation of the Rosetta Mission, the satellite model lands about $1152m$ away from the intended landing site after bouncing twice, which is quite close to the $1000m$ of the real case. Thus it is reasonable to assume that the results given by the simulation well represent the reality.

1 Introduction

Computer science has been an indispensable part in aero engineering since recent years; scientists must approximate the trajectory and/or the landing point of a satellite/lander with the help of computers to ensure a successful launching. Real cases could be extremely difficult because factors affecting the motion of satellites/landers are innumerable. For instance, in the Rosetta Mission, a mission in which a spacecraft firsts orbit around and then lands on Comet 67P [1], gravity-sources outside the system, the shape of the comet, etc., make the motion of the spacecraft hard to simulate. In this project, only the masses, initial positions and velocities of the targeted celestial body (Mars/67P) and the spacecraft are considered; therefore, the only things needed to be understood and calculated are how the gravitational force from a single source affects the spacecraft's motion and how the spacecraft behaves upon hitting the surface. Later simulation of the Rosetta Mission will be presented later and it gives a general idea about how accurate this simplified programme is.

2 Theory

When the motion of an object is considered, basic definitions are recalled that velocity as the time derivative of displacement, acceleration as the time derivative of velocity, and force as the product of mass and acceleration. The following equations can be listed:

$$\vec{v} = \frac{d\vec{x}}{dt} \tag{1}$$

$$\vec{a} = \frac{d\vec{v}}{dt} = \frac{d^2\vec{x}}{dt^2} \tag{2}$$

$$\vec{F} = m\vec{a} = m\frac{d^2\vec{x}}{dt^2} \tag{3}$$

In this system, it is clear that the gravitational force is the only external force exerted on the satellite. According to Sir Isaac Newton, gravitational force is given by this equation:

$$\vec{F} = -\frac{GMm}{|\vec{r}|^2}\hat{r} \tag{4}$$

where G is the gravitational constant, M is the mass of the object that exerts force, m is the mass of the object on which force is exerted and r is the distance between the two objects. By transforming Equation(3) and Equation(4) to the scalar form and combining, the equation of acceleration for this two-body system is generated:

$$a = \frac{GM}{r^2} \tag{5}$$

The above equations can explain all motions of the satellite as long as it does not hit the surface of Mars. Solving these sets of ordinary differential equations by hand could be tremendous work, but Python can use *odeint* method to solve them easily [2].

Equations for the kinetic energy and potential gravitational energy of an object are also useful in determining whether the satellite is captured by the planet or not. Remember that kinetic energy is one half of the product of mass and squared velocity and gravitational potential energy is the force integrated from the initial position to infinity:

$$KE = \frac{1}{2}mv^2 \tag{6}$$

$$PE = \int_r^\infty -\frac{GMm}{|\vec{r}|^2}\vec{r}dr = -\frac{GMm}{r} \tag{7}$$

In the cases when the satellite collides with Mars, this equation:

$$\vec{a}_{proj\vec{b}} = |\vec{a}|cos\theta \cdot \hat{b} = \frac{\vec{a}\cdot\vec{b}}{|a||b|} \cdot \vec{b} \tag{8}$$

gives the projection of $\vec{a}$ onto $\vec{b}$, where $\hat{b}$ is the unit vector of $\vec{b}$. It can help to find the velocity components after the collision.

3 Method

In the first simulation, Mars is placed at the origin with the radius set to be $3.4 \times 10^6 m$ and the mass $6.4 \times 10^{23} kg$; the mass of the satellite is $260kg$ ($R = 3.4 \times 10^6 m$, $M = 6.4 \times 10^{23} kg$, $m = 260kg$)[2]. In the beginning of the programme, the displacement, velocity and acceleration of the satellite need to be defined in a set of ordinary differential equations (ODE) for Python to solve. However, these physics quantities cannot be typed as vectors in the Python script, so their components in the x-axis and y-axis need to be expressed separately. Therefore, each axis has a set of ODE. After adding the time variable, and the initial position and velocity of the satellite, the *odeint* method in the scipy.integrate package is able to solve these equations; then the programme returns arrays of displacements and velocities of different time points for both axises. Figure(1) shows the trajectory

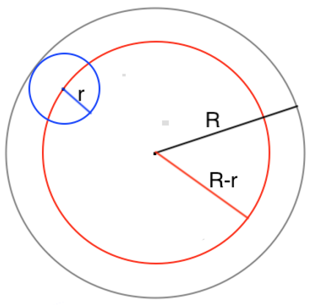


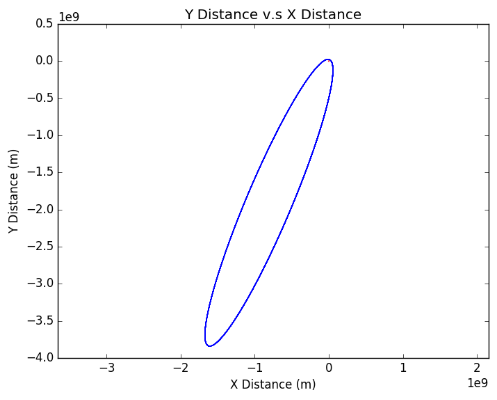
Figure 1: Trajectory of The Satellite with initial conditions: $x = 7R$, $y = 3R$, $v_x = -1000ms^{-1}$, $v_y = 1000ms^{-1}$

of the satellite that can be obtained by plotting the displacement in the y-axis versus that in the x-axis.

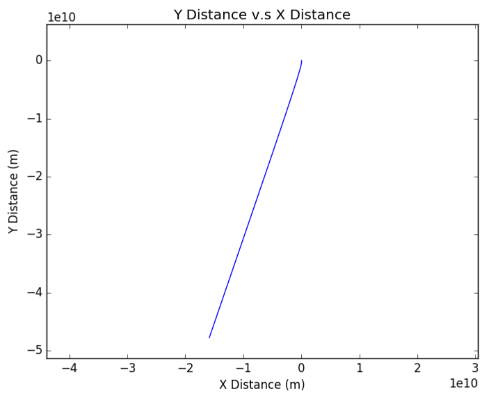
One of the three things could happen to the satellite as it is approaching Mars: it escapes, it orbits around or collides with Mars. Analysing the initial kinetic energy and potential energy of the satellite is crucial to determine how the satellite behaves. When the kinetic energy is larger than the gravitational potential energy, the satellite's velocity exceeds the escape velocity of its initial position; therefore it passes by Mars and never comes back. Otherwise it is captured by Mars (orbiting or colliding). By comparing Equation(6) and Equation(7), an equation of the escape velocity of a specific displacement can be generated :

$$v_{escape} = \sqrt{\frac{2GM}{r}} \tag{9}$$

Using this equation, the escape velocity of Mars at $(x = 7R,\ y = 3R)$ is found to be $1815.8ms^{-1}$. The result can be tested by adjusting the initial velocities in both axises to generate a net velocity close to $1815.8ms^{-1}$.



(a) $v_{net} = 1810.2ms^{-1}$



(b) $v_{net} = 1817.3ms^{-1}$

Figure 2: Trajectory of the satellite

The two trajectories in Figure(2a) and Figure(2b) shows that when the net velocity is a little bit below the escape velocity, the satellite orbits around Mars; when the net velocity is slightly higher than the escape velocity, the satellite escapes, which proves that the method used to find escape velocity is correct. The angular deviation can be easily obtained by calculating the arctangent of the ratio of the initial v_y to v_x and the ratio of the v_y to v_x at the last time point. The resultant plot of angular deviation versus net velocity is shown in Figure(3).

The plot shows that the angular deviation approaches 0 as the net velocity increases, which makes sense because the gravity has no enough time to change the direction of the satellite if it moves too fast.

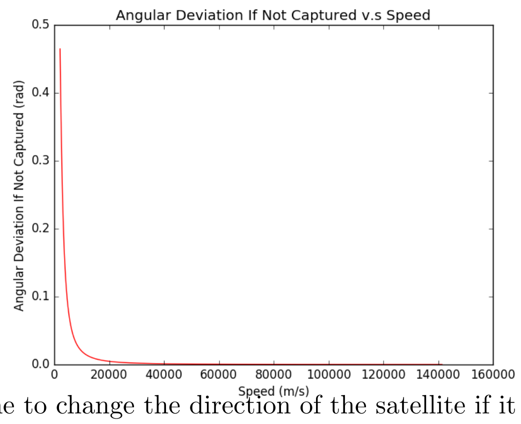
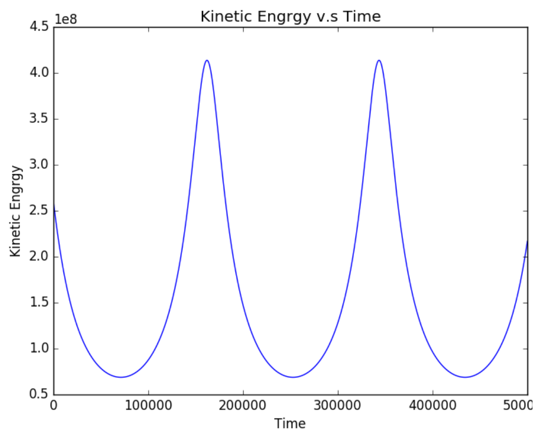


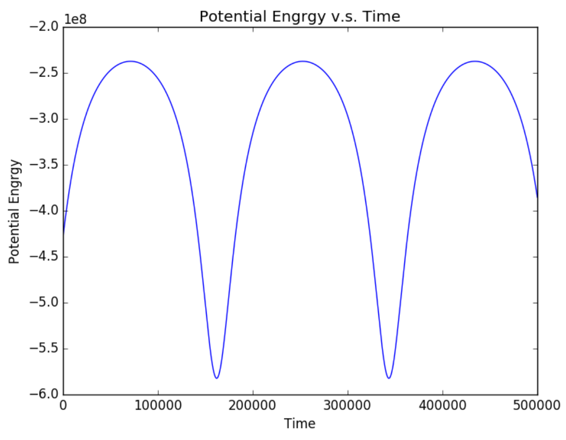
Figure 3: Angular Deviation v.s. Velocity

If the satellite is captured, it can either collide with or orbit around Mars. To determine which the case is, the minimum distance between the satellite and Mars has to be found. By applying the Pythagorean theorem to the displacement arrays of x and y-axis, an array of distance (r) is obtained. Using $min(r)$ to get the minimum distance to see whether it is less than the radius of Mars (R). If it is, then the satellite will collide with Mars.

By using Equation(4a) and Equation(4b, two arrays of kinetic energy and potential energy can be generated. The plots of these two energy and the total energy versus time are shown as follow:

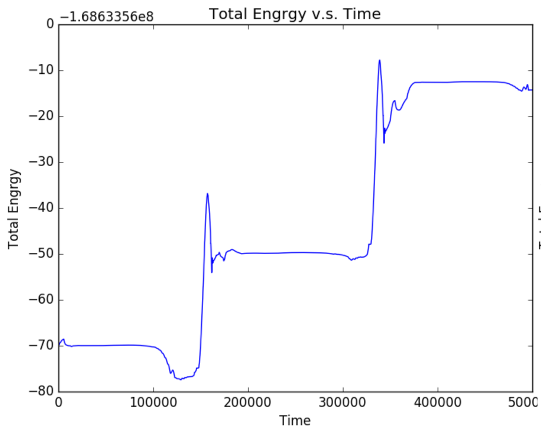


(a) Kinetic Energy v.s Time

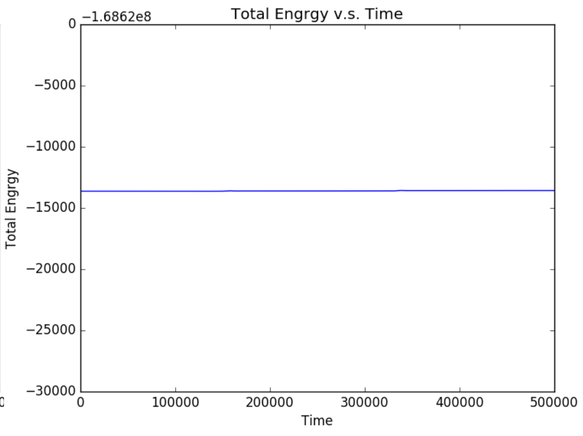


(b) Potential Energy v.s Time

Figure 4: Energy (Satellite Captured) v.s. Time



(a) At a small scale



(b) At a larger scale

Figure 5: Total Energy (Satellite Captured) v.s. Time

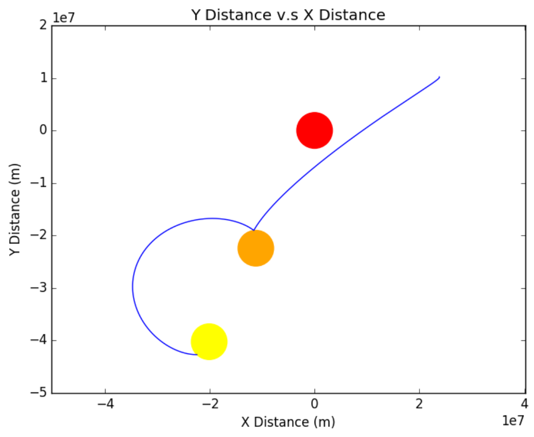
As the satellite orbits around Mars, its kinetic energy and potential energy fluctuate periodically as show in Figure(4) because it gains kinetic energy and loses potential energy when it is close to Mars. Figure(5b) shows that the total energy is conserved if being observed in a large scale; however, it fluctuates irregularly at a small scale, which is probably due to the error of the programme. The error is negligible considering the vale of the total energy itself is extremely large.

If a velocity is given to Mars, all the results obtained from the previous parts lose their accuracy because the moving gravity source changes the acceleration of the satellite. Therefore, the displacement of Mars must be added to all the displacements of the satellite and the velocity of Mars also needs to be added to the velocities of the satellite, which solve the problem.

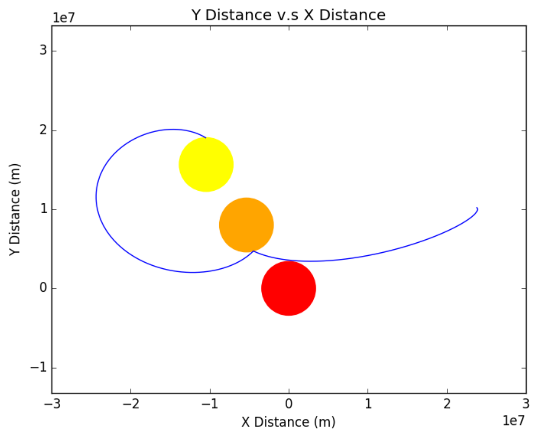
The last work is to make the satellite bounce like a lander after collision. A loop is used to check whether $r[i] > R$ until the index i of the collision point is found. This index can also be used to find the velocity and position of the satellite at the time it collides (e.g. $x[i]$, $y[i]$, $v_x[i]$, $v_y[i]$). The vector from the centre of mass of Mars to the collision point can be obtained by using this operation:

$$\vec{r} = (x[i] - v_{xmars} * t[i], y[i] - v_{ymars} * t[i]) \tag{10}$$

where v_x and v_y are the velocities of Mars in the direction of the x and y-axis. By applying Equation(8) to $\vec{r}$ and the velocity vector of the satellite at the time of collision, and multiplying with the restitution (0.7), the normal component of the reflected velocity is obtained. Using simple vector addition can get the reflected velocity's tangent component. Since the velocity and position of the satellite are known, a new ODE can be set up to calculate the trajectory after the bouncing. Taking the elements that are before the collision from the displacement arrays and combining them can give a continuous trajectory of the satellite that shows the bouncing process while does not goes into Mars. Two examples are shown in Figure(6), where the red circle is the initial position of Mars, orange one is the position of Mars when the first collision takes place and so on.



(a)



(b)

Figure 6: Satellite Bouncing on Moving Mars

In the simulation of the Rosetta Mission, the radius of the small lobe is set to be $1100m$ (r_{small}) and the radius of the large one is $1220m$ (r_{large}), which are determined by the ratio of their estimated volumes. The total mass of the Comet 67P also needs to be divided into two part by the ratio of the volumes of the two lobes, so two gravity sources are affecting the satellite. Two circles are put along the x-axis with the centre of mass at the origin. The small circles locates at $(1659, 0)$ and the large one locates at $(-660, 0)$. By adjusting the settings in the displacement and acceleration, and repeating the code for bouncing gives Figure(7)

The landing point is found to be $1152m$ along Mars' surface away from the original colliding point, which is very close to the reality.

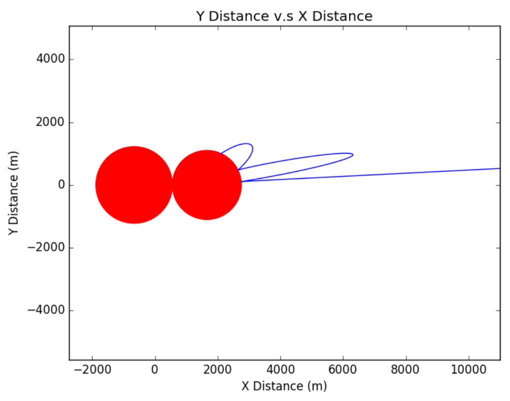


Figure 7: Simulation of The Rosetta Mission

4 Errors, Results and Discussion

Since the whole project is carried out in computer programme, the errors mainly exist in the inaccuracy of computer's calculation and the modeling process. First of all, as stated in the previous section, this simulation of satellites/landers' behaviour is over simplified. For example, the masses of Mars and the satellite are not precisely measured, and Mars is not a perfect sphere, nor is 67P made up of two balls. When bouncing from the surface, the incident angle is very important to determine the reflected velocity; using a circle to simulate the bouncing is not sufficient. The restitution is also estimated. Moreover, disturbance outside the system, such as cosmic dust and gravity of other planets, also affects the motion of the satellite. the computer also makes mistakes because it cannot calculate the position and velocity of every time point. The time must be divided into finite intervals, which loses much detail. Because of the time interval, the satellite cannot bounce back exactly from the surface and therefore the programme gives inaccurate result.

Most of the errors in this project are uncontrollable. If the project were to be done for the second time, better models of the satellite and the targeted celestial body will probably improve the result greatly; gravity-sources near this system could also be put into consideration.

5 Conclusion

The product gave a simulation of a satellite's behaviour under the gravity of a single source in a ideal situation. Although many details were omitted in the model, the simulation of the Rosetta Mission gave a satisfactory result; therefore it is reasonable to assume that the simulation is close to the reality to some degree.

References

[1] "Rosetta Rendezvous Mission with Comet 67P/Churyumov-Gerasimenko". Directory.eoportal.org. https://directory.eoportal.org/web/eoportal/satellite-missions/r/rosetta. Retrieved Nov 22, 2016.

[2] First Year Lab Manual 2016-17. Imperial College London.