Exercise 1 Report

Paddy Milner[1]

[1] *Department of Physics, University of Bristol*

01.03.2025

Abstract

The aim of this exercise was to use computational methods to model the gravitational interactions between the earth, the moon, and a lunar probe orbiting the moon. The model provided accurate results when provided with realistic starting conditions, and when the starting conditions were changed, the model responded as expected.

1 Introduction

In this exercise we will create a computational model to first simulate the orbit of the moon around the earth, and then the orbit of a lunar probe around the moon as well. Using these simulations, we will first use realistic starting conditions to verify that the simulations behave as expected, i.e. they behave how these objects do in real life. We will then alter these starting conditions and observe how these alterations affect the simulations result.

2 Theory and Methods

Our models will simply use Newton's equation for gravitational attraction,

$$F = \frac{GMm}{r^2}$$

Combined with Newton's second law of motion,

$$F = ma$$

to find the acceleration of the objects. For the moon's orbit around the earth, we find that

$$M_m \ddot{r}_m = -\frac{M_e M_m G}{|r_m|^2}\hat{r}_m = -\frac{M_e M_m G}{|r_m|^3} r_m$$

where M_e is the mass of the earth, M_m is the mass of the moon, G is the gravitational constant, and $r_m = (x_m, y_m)$, the coordinates of the moon relative to the fixed origin at the centre of the earth. We will assume that the mass of the moon is negligible compared to that of the earth, and therefore the motion of the earth can be ignored, hence the fixed origin at its centre. From this, we can obtain the following differential equations:

$$\begin{aligned} \frac{\mathrm{d}v_{m,x}}{\mathrm{d}t} &= -\frac{M_e G x_m}{(x_m^2 + y_m^2)^{3/2}}; & \frac{\mathrm{d}x_m}{\mathrm{d}t} &= v_{m,x} \\ \frac{\mathrm{d}v_{m,y}}{\mathrm{d}t} &= -\frac{M_e G y_m}{(x_m^2 + y_m^2)^{3/2}}; & \frac{\mathrm{d}y_m}{\mathrm{d}t} &= v_{m,y} \end{aligned} \tag{1}$$

This can then be extended to model the probe orbiting the moon as well. It feels attraction from both the moon and the earth, so needs terms for both:

$$\begin{aligned} \frac{\mathrm{d}v_{p,x}}{\mathrm{d}t} &= -\frac{M_e G x_p}{(x_p^2 + y_p^2)^{3/2}} - \frac{M_e G x_{pm}}{(x_{pm}^2 + y_{pm}^2)^{3/2}}; \\ \frac{\mathrm{d}x_p}{\mathrm{d}t} &= v_{p,x} \\ \frac{\mathrm{d}v_{p,y}}{\mathrm{d}t} &= -\frac{M_e G y_p}{(x_p^2 + y_p^2)^{3/2}} - \frac{M_e G y_{pm}}{(x_{pm}^2 + y_{pm}^2)^{3/2}}; \\ \frac{\mathrm{d}y_p}{\mathrm{d}t} &= v_{p,y} \end{aligned} \tag{2}$$

where r_p is the position of the probe relative to the earth, and $r_{pm} = r_p - r_m$ is the position of the probe relative to the moon. These will then be solved in python using the scipy.integrate.solve_ivp () method, after being supplied with initial condition for velocity and position.

3 Explanation of Code

3.1 Orbit of the Moon

```
def part1(velocityFactor, orbits):
    #Part one Diffeq
    def f_part1(t, state, Me, Mm, G):
        xm, ym, vx, vy = state #all input values

        dxmdt = vx #functions for each diffeq
        dymdt = vy
        dvxdt = -(Me*G*xm)/((xm**2+ym**2)**(3/2))
        dvydt = -(Me*G*ym)/((xm**2+ym**2)**(3/2))


        return (dxmdt, dymdt, dvxdt, dvydt) #return differentiated values
```

The function for part 1 takes 2 inputs, velocityFactor and orbits. These are scalar multipliers for the initial velocity and max time respectively, which can be altered as desired. The function for the derivatives is then defined, taking an input of time, the initial state, the masses of each object and the gravitaional constant. State is provided as a tuple with 4 objects, and each value in it is then assigned to xm, ym, vx and vy. The equations for the derivative of each of these values, relating to the equations outlined in equation 1 are then defined, and are returned in the same order they were initially given.

```
# Initial Conditions
Me = 5.97*(10**24) #Constants
Mm = 7.35*(10**22)
G = 6.67*(10**-11)

t_min = 0
t_max = 2360620*int(orbits) #Time for one orbit * number of orbits
numpoints = 2000*int(orbits) #To keep good clarity when plotting
t = np.linspace(t_min, t_max, numpoints)

r = 384400000 #Distance or orbit
v = sqrt((G*Me)/r)*float(velocityFactor)#Balancing centrepetal force and gravitational force,
                                        #then multiplying by a scalar to obtain an elliptical orbit

xm0 = r #Initial position and velocity of moon
ym0 = 0 #Set so moon is travelling only in the y direction initially
vx0 = 0
vy0 = v

rtol = 1e-5 #Tolerences set to balance computation speed with accuracy
atol = 1e-8
```

The initial starting conditions of the system are then defined. M_e, M_m, G and r are all known and fixed quantities. The standard value for v, the initial velocity of the moon, is set by balancing centripetal force with gravitational force to produce a stable, circular orbit, however it can be changed by altering velocityFactor in order to produce an elliptical orbit. t_max is set to the sidereal period of the moon multiplied by the number of orbits desired. The position and velocity are then split up into x and y values, and the simplest case is taken, with the moon starting on the x axis and moving directly upwards, allowing us to set y_m and $v_{x,m}$ to 0, and x_m and $v_{y.m}$ to r and v respectively.

```
#Solver
results = solve_ivp(f_part1, (t_min,t_max), (xm0, ym0, vx0, vy0), args=(Me, Mm, G), t_eval=t, atol=atol, rtol=rtol)


#Graph plotting
ax=plt.axes()    # This creates some axes, so that we
ax.set_aspect(1) # can set the aspect ratio to 1 i.e.
                 # x and y axes are scaled equally.
ax.set_xlabel("x coordinate (m)") # Must label axes (with
ax.set_ylabel("y coordinate (m)") # units) and give
ax.set_title("Orbit of moon around earth") # plot title.
#
ax.plot(results.y[0],results.y[1]) # Make the plot
ax.plot(0,0)
ax.legend(['Moon'])              # and add a key.
ax.axhline(y=0, color='k', linewidth=0.5)
ax.axvline(x=0, color='k', linewidth=0.5)
plt.show()
```

The solve_ivp() function is then called to solve the differential equations. The initial conditions are provided as a tuple, in the order defined in the differential equations function. The time range is also provided as a tuple, and the other arguments needed for the differential equations function are provided using args=. Matplotlib is then used to create a plot of the position of the moon at each time defined in t. x and y axes are then added for clarity.

3.2 Lunar Probe

```
def part2(velocityFactor, orbits):
    #Part two Diffeq
    def f_part2(t, state, Me, Mm, G):
        xm, ym, vx, vy, xp, yp, vpx, vpy = state #all input values

        xpm = xp-xm
        ypm = yp-ym

        dxmdt = vx #moon diffeqs
        dymdt = vy
        dvxdt = -(Me*G*xm)/((xm**2+ym**2)**(3/2))
        dvydt = -(Me*G*ym)/((xm**2+ym**2)**(3/2))

        dxpdt = vpx #probe diffeqs
        dypdt = vpy
        dvpxdt = -((Me*G*xp)/((xp**2+yp**2)**(3/2)))-((Mm*G*xpm)/((xpm**2+ypm**2)**(3/2)))
        dvpydt = -((Me*G*yp)/((xp**2+yp**2)**(3/2)))-((Mm*G*ypm)/((xpm**2+ypm**2)**(3/2)))

        return (dxmdt, dymdt, dvxdt, dvydt, dxpdt, dypdt, dvpxdt, dvpydt)
```

The initial configuration for section 2 is very similar to that of section 1. The function for the section also takes velocityFactor and orbits as inputs, which perform the same function as in section 1. The differential equations function has a similar form to the one in section 1, taking time, state and some constants as input. The differential equations that govern the movement of the moon and their associated state values are the same, but we now have the additional equations seen in equation 2, and the variables required for them. Each differential equation is then returned in the same order they were provided, as in the first part.

```
# Initial Conditions
Me = 5.97*(10**24)
Mm = 7.35*(10**22)
G = 6.67*(10**-11)

t_min = 0
t_max = 2360620*int(orbits)
numpoints = 2000*int(orbits)
t = np.linspace(t_min, t_max, numpoints)

rm = 384400000
vm = sqrt((G*Me)/rm)*float(velocityFactor)

rpm = 10000000
vpm = sqrt((G*Mm)/rpm)

xm0 = rm
ym0 = 0
vx0 = 0
vy0 = vm

xp0 = rpm+rm #as position is relative to the earth
yp0 = 0      #not to the moon
vpx0 = 0
vpy0 = vm+vpm

rtol = 1e-6
atol = 1e-9
```

We then set the initial conditions for both the moon and probe. The values for the moon are all the same as in section 1. The initial conditions for the probe are found by selecting an appropriate distance for the probe from the moons center, then again balancing centripetal and gravitational forces to find its velocity around the moon. However, as the coordinate system is relative to the earth and not the moon, both the initial position and velocity of the probe relative to the moon must be added to the position and velocity of the moon relative to the earth, in order to get the state of the probe relative to the earth. The tolerances for this part were set lower, as the motion of the probe is more precise than that of the moon, so this was required to achieve a good degree of accuracy.

```
#Solver
results = solve_ivp(f_part2, (t_min,t_max), (xm0, ym0, vx0, vy0, xp0, yp0, vpx0, vpy0), args=(Me, Mm, G), t_eval=t, atol=atol, rtol=rtol)


#Graph plotting
ax=plt.axes()    # This creates some axes, so that we
ax.set_aspect(1) # can set the aspect ratio to 1 i.e.
                 # x and y axes are scaled equally.
ax.set_xlabel("x coordinate (m)") # Must label axes (with
ax.set_ylabel("y coordinate (m)") # units) and give
ax.set_title("Orbit of moon around earth") # plot title.
ax.plot(results.y[0],results.y[1], label="Moon" ) # Make the plot
ax.plot(results.y[4],results.y[5], label="Probe")
ax.legend(loc='upper right')          # and add a key.
ax.axhline(y=0, color='k', linewidth=0.5)
ax.axvline(x=0, color='k', linewidth=0.5)
plt.show()
```

The results are then calculated using solve_ivp (). Again this is very similar to the procedure in part 1, only with more variables for the initial state. Matplotlib

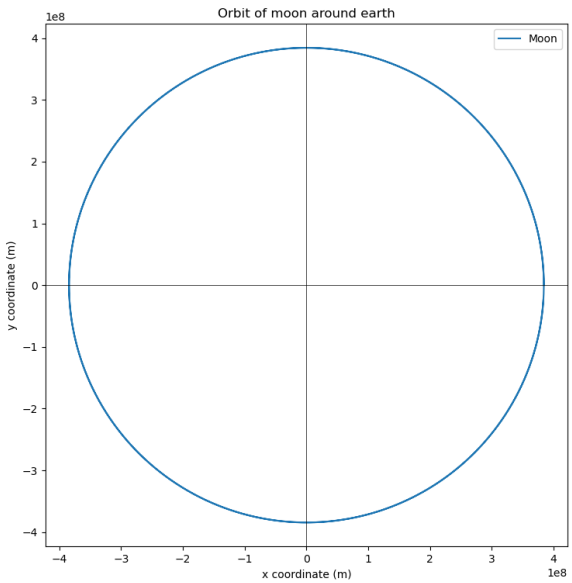

Figure 1: Circular orbit as a result of standard initial conditions

is again used to plot the position of both the moon and the probe in the same fashion as part 1.

4 Results and Discussion

4.1 Orbit of the Moon

When given standard starting conditions, the simulation provided a stable and circular orbit, as expected, shown in figure 1.

The simulation kept its shape for a large number of orbits, showing its stability. When the initial velocity was altered but the radius of the orbit was held, we observed a range of effects. For small alterations, approximately in the range of 0.1¡velocityFactor¡1.41, stable elliptical orbits were produced, as shown in figures 2 and 3.

For velocities less than 0.1*v, the simulation began to fail, producing what was effectively noise. At velocities greater than 1.41*v, the moon appears to escape the orbit of the earth, as shown in figure 4. These trajectories continued with increasing t_max, to the point where simulation took a very long time, suggesting that they are not simply very large orbits.

This can be compared with what we would expect

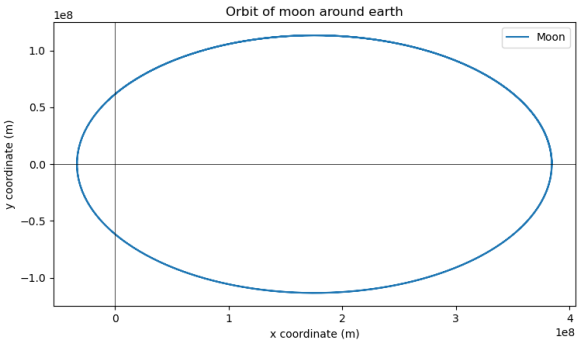

Figure 2: Elliptical orbit with a velocity of 0.4*v

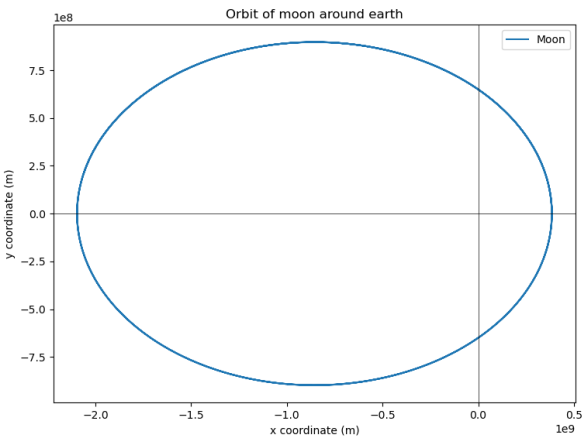

Figure 3: Elliptical orbit with a velocity of 1.3*v

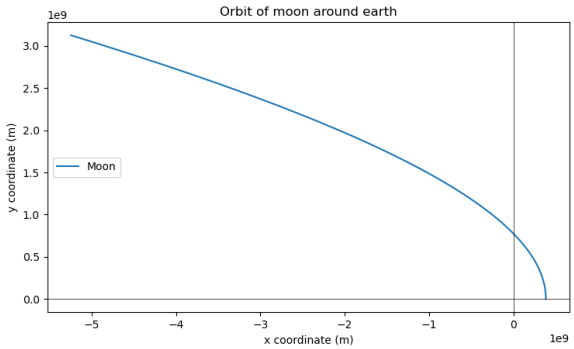

Figure 4: Moon escaping the earths orbit

using the escape velocity equation, which is given by

$$v_e = \sqrt{\frac{2GM}{r}} \tag{3}$$

As we find the velocity for a circular orbit by balancing centripetal and gravitational force, we use the equation

$$v = \sqrt{\frac{GM}{R}} \tag{4}$$

We can see that the difference between equation 3 and equation 4 is simply a factor of $\sqrt{2}$, or 1.414, so our model holds well with what we would expect.

4.2 Lunar Probe

For standard starting values for section 2, we again see the moon in a stable circular orbit around the earth, and the probe forms a stable orbit around the moon as well, as shown in figure 5 When altering the initial velocity of the moon, for small changes we again see stable elliptical orbits. With increases above 1.41*v, causing the moon to escape the earth, the probe continues to orbit the moon in a stable manner, as shown in figure 6 When we decrease the starting velocity of the moon, it continues in a stable orbit, however when reaching the periapsis of the moons orbit, the probe is accelerated enough to

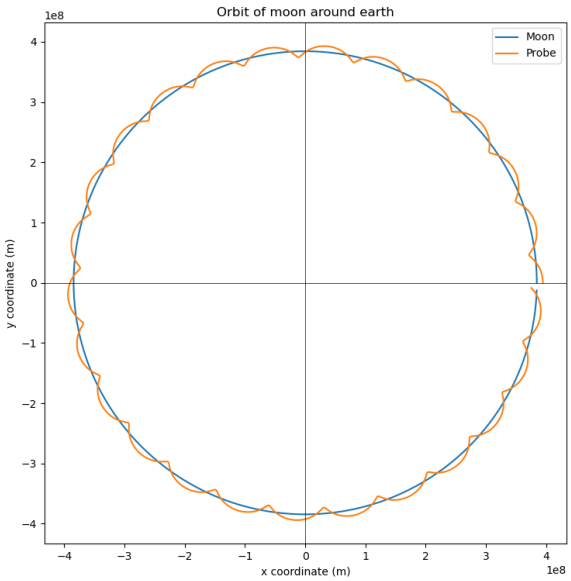

Figure 5: Circular orbit of moon and probe

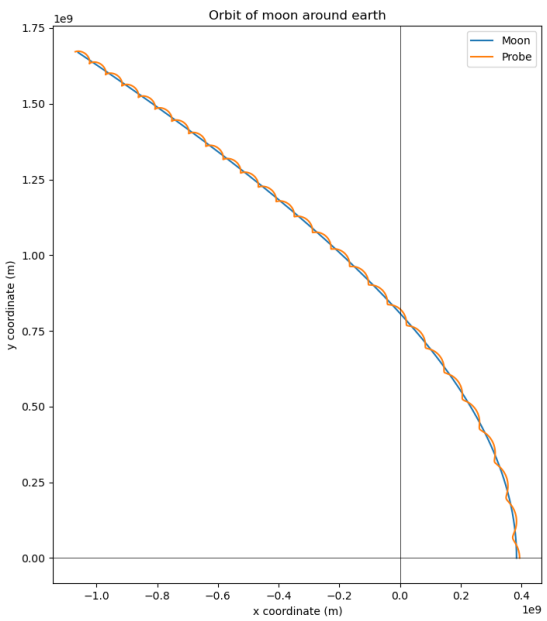

Figure 6: Moon and probe escaping the earth

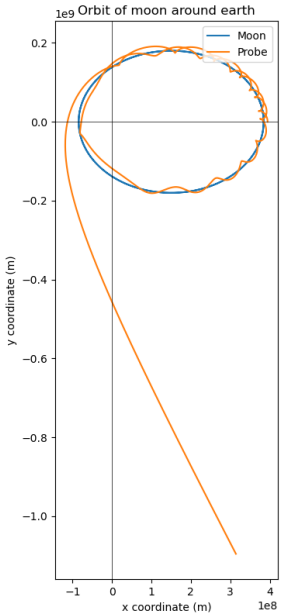

Figure 7: Probe escaping the moon

escape the orbit of both the moon and the earth, as shown in figure7 The point at which the probe escapes the moon can be increased by changing the direction of the probes orbit around the moon, simply by setting vpy0 = vm-vpm, rather than vm+vpm. Now, when the probe escapes the orbit of the moon, it is still gravitationally bound to the earth and enters an erratic orbit around it, shown in figure 8

5 Conclusion

Overall, the model produced mostly accurate results when given starting values somewhat close to realistic values, however began to lose clarity at extreme values, especially with low starting velocities. This could likely be improved by decreasing the tolerance of the model, however this would lead to larger calculation times. The model agreed with the value for the escape velocity of the moon, showing a good accuracy when provided with somewhat realistic data. Other improvements to the model could include adding a third dimension, as well as modelling the effects of the rotation of each of the objects, as these tidal forces do affect the path of the moons orbit in real life.

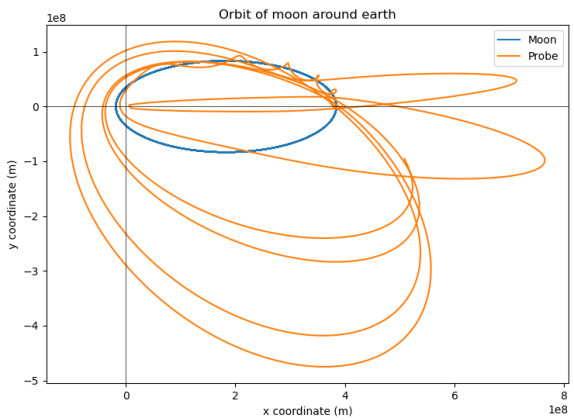

Figure 8: Probe escaping the moon and entering an orbit around the earth