Chapter 12

The Restricted Three-Body Problem (Rocket to the Moon)

12.1 Background

The goal of this project is to integrate very accurately the equations of motion that govern gravitating particles. You will tackle problems in which accuracy is of key importance. This will allow you to predict confidently the position of particles over many orbital periods.

Why would you want to be able to do this? While the motion of two point masses around each other can be solved exactly, gravitating systems of three or more particles have no known analytical solution and are inherently chaotic: small variations in the initial position or velocity result rapidly in increasingly large deviations as a function of time. We will concentrate on systems of three particles. In the simplest case, two very massive particles orbit each other, defining a time-evolving gravitational potential, and we are interested in the motion of the third, lighter body. This is called the 'restricted three-body problem' - restricted because we assume that the third body does not affect the orbit of the other two. This has many interesting applications, from the stability of satellite orbits, the formation of planetary rings, the stability of the solar system, to the destruction of stellar disks in galaxy mergers.

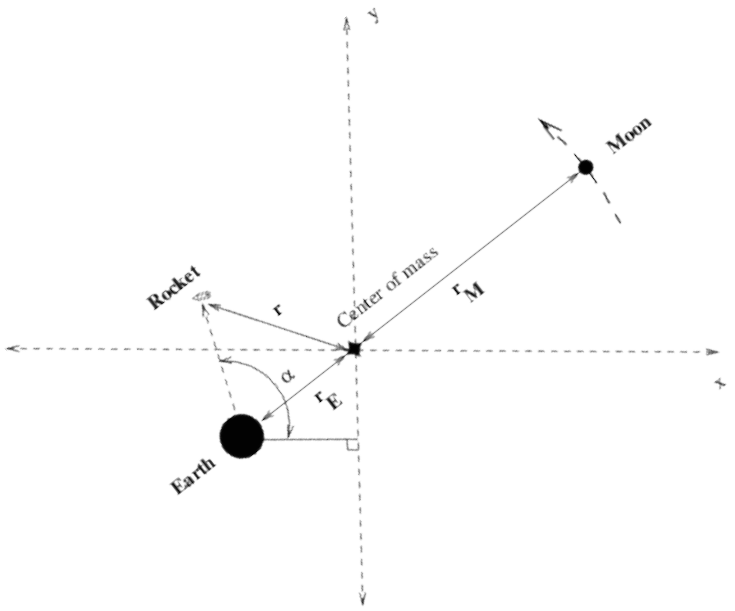
We will start with a simpler example, that of sending a rocket to the Moon. We will solve for the ballistic motion of the rocket, as it is attracted gravitationally by both Earth and Moon, treating all as point particles. The Earth and Moon follow elliptical orbits about a common centre of mass. Approximating this orbit as circular, the distance of each body from the centre of mass can be written in terms of d, the Earth-Moon distance, as

$$r_E = d\frac{m_M}{(m_M + m_E)}$$

and

$$r_M = d\frac{m_E}{(m_M + m_E)}.$$

Here, r_E and r_M are the distances of the Earth and Moon to the centre of mass, and m_E and m_M are the masses of the Earth and Moon, respectively. The period of the orbit, T, can be worked out using Kepler's third law (using the sum of the Earth and Moon masses as the gravitating mass).



Let us now consider the forces exerted on a spacecraft (whose mass, m, is negligible compared with the Earth or the Moon!), again with the centre of mass as the origin of the co-ordinate system. For simplicity we assume that the rocket moves in the orbital plane of the Earth-Moon system. All we need is Newton's law of gravity:

$$\mathbf{F} = -Gmm_E\frac{(\mathbf{r}-\mathbf{r_E})}{|\mathbf{r}-\mathbf{r_E}|^{\mathbf{3}}} - Gmm_M\frac{(\mathbf{r}-\mathbf{r_M})}{|\mathbf{r}-\mathbf{r_M}|^{\mathbf{3}}},,$$

and hence

$$\frac{d^2\mathbf{r}}{dt^2} = -Gm_E\frac{(\mathbf{r}-\mathbf{r_E})}{|\mathbf{r}-\mathbf{r_E}|^{\mathbf{3}}} - Gm_M\frac{(\mathbf{r}-\mathbf{r_M})}{|\mathbf{r}-\mathbf{r_M}|^{\mathbf{3}}}\,,$$

where variables in bold type now represent vector quantities. In terms of Cartesian coordinates x and y in the orbital plane,

$$\frac{d^2x}{dt^2} = -Gm_E\frac{(x-x_E)}{d_E^3} - Gm_M\frac{(x-x_M)}{d_M^3}\,,$$

where

$$d_E = \sqrt{(x-x_E)^2+(y-y_E)^2}$$

is the distance from the spacecraft to the Earth, and

$$d_M = \sqrt{(x-x_M)^2+(y-y_M)^2}$$

is that from the spacecraft to Moon.

Your first task is to write a programme to solve for the motion of the spacecraft as a function of time in the gravitational field of the Earth and Moon.

12.2 Numerical solution: Solving differential equations

There are many different ways solving the differential equations of motion numerically. The simplest is via a Taylor expansion. Given the position and velocity of the rocket at time t_n, the Taylor expansion in the small time-step $a = t_{n+1} - t_n$ is

$$\mathbf{x}_{n+1} = \mathbf{x}_n + a\dot{\mathbf{x}}_n + \frac{a^2}{2}\ddot{\mathbf{x}}_n + \mathcal{O}(a^3) \tag{12.1}$$

$$\dot{\mathbf{x}}_{n+1} = \dot{\mathbf{x}}_n + a\ddot{\mathbf{x}}_n + \mathcal{O}(a^2) \tag{12.2}$$

where $\mathcal{O}(a^n)$ indicates the error term resulting from truncating the expansion depend on the step size a to the n^{th} power, and $\dot{\mathbf{x}} \equiv \mathrm{d}\mathbf{x}/\mathrm{d}t$.

A more complicated but more accurate method is the fourth-order Runge-Kutta formula,

$$\mathbf{z}_1 = \mathbf{x}_n + \frac{a}{2}\dot{\mathbf{x}}_n \qquad \dot{\mathbf{z}}_1 = \dot{\mathbf{x}}_n + \frac{a}{2}\ddot{\mathbf{x}}_n \tag{12.3}$$

$$\mathbf{z}_2 = \mathbf{x}_n + \frac{a}{2}\dot{\mathbf{z}}_1 \qquad \dot{\mathbf{z}}_2 = \dot{\mathbf{x}}_n + \frac{a}{2}\ddot{\mathbf{z}}_1 \tag{12.4}$$

$$\mathbf{z}_3 = \mathbf{x}_n + a\dot{\mathbf{z}}_2 \qquad \dot{\mathbf{z}}_3 = \dot{\mathbf{x}}_n + a\ddot{\mathbf{z}}_2 \tag{12.5}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{a}{6}(\dot{\mathbf{x}}_n + 2\dot{\mathbf{z}}_1 + 2\dot{\mathbf{z}}_2 + \dot{\mathbf{z}}_3) + \mathcal{O}(a^5) \qquad \dot{\mathbf{x}}_{n+1} = \dot{\mathbf{x}}_n + \frac{a}{6}(\ddot{\mathbf{x}}_n + 2\ddot{\mathbf{z}}_1 + 2\ddot{\mathbf{z}}_2 + \ddot{\mathbf{z}}_3) + \mathcal{O}(a^5) \tag{12.6}$$

In this method the velocity and acceleration are evaluated at 3 test points $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3$ across the time-step, in addition to at the starting point. By taking a weighted average over these 4 points we greatly improve the accuracy of the position and velocity estimated at the next time-step. The error terms in this method are of order a^5. If carefully applied, this method will give us the accuracy to evolve the position of the 'rocket' over many orbital time periods, allowing you to tackle much more interesting problems.

For both the Taylor expansion and Runge-Kutta methods, the time-step size a should be 'small ' enough to track the orbit accurately. You should think carefully how to best limit a. Given initial values for $\mathbf{x} = (x, y)$ and $\dot{\mathbf{x}} = (dx/dt, dy/dt)$, with either the Taylor expansion or Runge-Kutta methods, we can march the orbit forward in time and calculate the position and velocity up to any given time. For each location along the orbit, we can calculate the acceleration from the position by using the equations of motion. Note that for the Runge-Kutta method we can only calculate $\mathbf{z}_2$ and $\dot{\mathbf{z}}_2$ after we know *both* $\mathbf{z}_1$ and $\dot{\mathbf{z}}_1$; this means we must apply the equations 12.3, 12.4, 12.5, and 12.6 in that order.

There is the added complication that the positions of the Earth and Moon are continually changing along their circular orbit, so their positions also need updating at each time-step. Since we assume that the Earth and Moon move on circular orbits about the centre of mass, it is straightforward to write down expressions for the x and y co-ordinates of these bodies as a function of time, using simple harmonic motion.

12.3 Implementation in Python

Although you should start to write your program using a simple implementation of these schemes, you are welcome to use the scipy library of routines that solve differential equations. For example, the routine scipy.integrate.ode will solve the differential equations for you. More help is given at the back of the book, or you can look at:

http://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html

However, these routines are a 'black box' and you should check that they work by comparing with your own simple code. You will need to understand and establish what limits the accuracy of these integrators.

This challenge is all about accuracy. If you are out by a few kilometre, the rocket will crash into the Moon's surface or escape into space. If you are integrating the 3-body system for a long period of time, small integration errors will rapidly build up and your results will be nonsense. As you write your program, think carefully about how to assess the accuracy of your results.

12.4 Your Work Plan

1. **Design your program.** First, design a code to follow the motion of a rocket in the gravitational field of the Earth and Moon. Write down a pseudo-code which sets out the steps you will need to go through to solve the equations of motion.

   Your outline of the code structure should include: (a) Names and descriptions of the functions needed along with the variables used by the function and the variables returned by the function. (b) What data will you need to input to the code and how will you show the results? (c) What data structures will you use to store information in the code?

   Some things to think about: We have assumed that the Earth and Moon follow circular orbits about their centre of mass. Therefore, at a given time, we can write down the x and y co-ordinates of the Earth and Moon using the solutions of simple harmonic motion and given the period of the orbit calculated using Kepler's third law. Write your program in such a way that it would be easy to adapt to another method for calculating the orbit.

   Since you will use many time steps, you may need to think carefully about data storage. Whilst all of the time steps are needed to calculate the motion of the rocket, not all of the steps need to be output to a file or shown on a plot. Also, note that to apply the Runge-Kutta algorithm, only a small number of variables need to be stored. We do not need to retain these variables for steps before the one we are advancing from. Hence, we can be selective about how many values of x and y we store in a vector (or 1-D array) to use in a plot of the orbit. Be aware that computational resources are limited. You should focus on extracting the maximum accuracy from the available resource.

2. **Solve the milestone problem.** Write a Python code to solve the equations of motion and follow the orbit of a rocket about the Earth-Moon system. You will need to solve for the position of the Earth and Moon along their orbits, and then us these to calculate the acceleration on the rocket.

   We will test the program by investigating the stability of the lunar L2 'Lagrangian' point. The L2 point lies on line connecting Earth and Moon, past the Moon as seen from Earth. The additional gravitational force exerted by the Moon compensates for the greater distance from the Earth and a satellite (or rocket) at this point will orbit Earth with the same period as the Moon, locked into the same relative position. An example of a satellite in the lunar L2 point is the Artemis spacecraft (see http://www.nasa.gov/mission_pages/themis/news/artemis-orbit.html). Satellites are often place at

the Sun-Earth's L2 point, an example is the *Planck* spacecraft. The lunar L2 point has also been proposed for radio SETI observations. Throughout this test, we will ignore (tidal) effects arising from the gradient in the Sun's gravitational field. This is a good approximation since the Earth-Sun distance is so much larger than the radius of the Moon's orbit, but see below for a possible improvements to the scheme. The distance of the lunar L2 Lagrangian point measured *from the Moon* is approximately

$$\Delta r = r_m \left(\frac{M_m}{3M_E} \right)^{1/3} . \tag{12.7}$$

To confirm that your program works correctly, you can solve for the motion of the rocket when it is initially placed at L2, and moving with speed such that its orbital period is that of the Moon. Solve for the position of the rocket over 1 lunar period. Plot the orbit of the Moon and the rocket: you will find that they initially orbit together but then separate. The expression for Δr is only approximate, and you can manually vary the initial position by a few percent to obtain better results. Now plot the position of the satellite and show that for a suitable choice of Δr it remains less than 100,000 km from the Moon over one period.

Hint: If you use the Taylor expansion algorithm, you should get reasonable results with a time-step of $a = 10$ seconds. The use of the more accurate Runge-Kutta method (or the method within SciPy) would allow you to use a larger time-step and trace out the motion of the rocket more quickly.

In your milestone interview, your program should solve for the orbit of a rocket at the L2 point to high accuracy over one lunar orbit. Use *both* the Taylor expansion *and* the Runge-Kutta (or SciPy) routine. Determine the optimal value of Δr (the distance of the L2 point from the Moon) to better than 1% accuracy, and plot the orbit of the Moon and the rocket. Determine the separation of the rocket relative to the Moon after 1 orbit.

3. **Research with your program.**

   Now you have a working code, you can investigate various aspects of the 3-body problem. High marks will only be given to reports that demonstrate initiative. Here are some ideas that you may wish to consider. You probably do not have space in your report to address more than one of these points: it is better to address one particular aspect in detail. Remember that you will need to be able to demonstrate the accuracy of your results.

   - You now have the tools to send a rocket to the Moon. Start with the rocket in a circular orbit around Earth, with radius of, say, 7000 km (note that this is *not* the distance from the rocket to the centre of mass!). Assume that the rocket is given an instantaneous velocity boost which sets it on a new orbit that takes it to the Moon. Use your code to estimate the required boost. Minimising the amount of fuel required is critical to the success of the mission. You should not be happy simply to reach the same orbital radius as the Moon or indeed crash into it. To extend your work, consider using the Lagrangian points, or travelling to Mars using a sling-shot around the Moon. Do not mix up units of miles and kilometres.
   - A class of asteroids called Trojans is associated with the Jupiter-Sun Lagrangian points, see for example http://astronomy.swin.edu.au/cosmos/T/Trojan+Asteroids. Investigate the stability of the orbits of these bodies.
   - Giant planets, such as Saturn and Jupiter, are surrounded by thin disks of rocks and dust debris. These rings have very detailed structure that results from resonances in the gravitational

potential created by the planet and its moons (Colombo et al. 1968). You could use your program to understand how such structures develop and investigate their long-term stability. How do your results compare to the observed rings?

- Gravitating systems of several massive bodies are inevitably chaotic. Yet the solar system has been stable for long enough for life to evolve on Earth. Use your program to investigate the long term stability of the solar system. This is a challenging problem because you will need to simultaneously evolve several planets over long timescales with extreme accuracy. Laskar, 2012, provides a recent review.
- In a seminal paper, Toomre & Toomre (1972) investigated galaxy collisions using the restricted 3 body method. A good starting point is to approximate the galaxies as point masses, and then calculate orbits of stars using your program. A better way is to represent the potential of each galaxy by an extended dark matter halo. By ignoring the gravitational interactions between individual stars, you can compute the orbits of stars independently, and so avoid needing to solve the full N-body problem). Initialise locations and velocities of the stars by placing them in a stable disk (check stability by evolving the galaxy in isolation), then compute their location as a function of time. 'When mice collide' is a spectacular 'Astronomy Picture of the day' image, located at http://apod.nasa.gov/apod/ap020506.html.

Constants

In order to avoid difficulties in comparing results for the Milestone problem, please adopt the following values of the physical and astronomical constants in your calculation.

The mass of the Earth is $m_E = 5.9742 \times 10^{24}\ kg$

The mass of the Moon is $m_M = 7.35 \times 10^{22}\ kg$

Newton's constant of gravitation is $G = 6.6726 \times 10^{-11}\ Nm^2kg^{-2}$

The Earth-Moon distance is $d = 3.844 \times 10^5\ km$

References

Toomre, A., & Toomre, J. 1972, ApJ, 178, 623

Colombo, G., Franklin, F. A., & Munford, C. M. 1968, AJ, 73, 111

Laskar, 2012, http://arxiv.org/abs/1209.5996

DeVries P.L.: A First Course in Computational Physics (Wiley)

Barger V.D. and Olsson, M.G.: Classical Mechanics, A Modern Perspective (McGraw-Hill)

A nice explanation of the Lagrangian points can be found at:
http://www.esa.int/esaSC/SEMM17XJD1E_index_0.html