PHYS30009: Intro to Computational Physics — Exercise 4

The deadline for this exercise is **Monday 20th March 2023** at 12:30 p.m. Your report and program (*.py) files should be uploaded into Blackboard at the appropriate point in the *PHYS_20032_30009_2022_TB-4: Introduction to Computational Physics (PHYS20032 + PHYS30009) 2022* Course.

Further information related to this exercise and the Runge-Kutta method will be given in the final computing lecture in Week 18. *S. Hanna*

The aims of this exercise are:

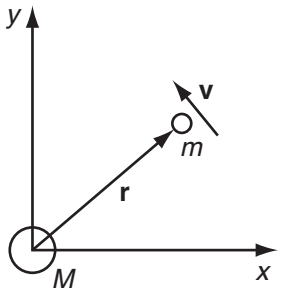
- To use some of the computational tools and knowledge gained thus far in the Computational Physics course to solve real physical problems;
- To use the Runge-Kutta method to explore the motion of rockets and orbits.

Problem 1 (60% of marks): Calculation of rocket orbits – one moving body

In both the situations studied here, you will consider a single moving body (the rocket) moving in a fixed gravitational potential provided by the earth and the moon. First of all, you will simulate orbits of the rocket around the earth (or other planet of your choice). This is most easily achieved by treating the planet as being very massive and stationary at the origin, while the rocket moves in orbit around it. Solving the problem entails solving the equation of motion:

$$m\ddot{\mathbf{r}} = -\frac{mMG}{|\mathbf{r}|^2}\hat{\mathbf{r}} = -\frac{mMG}{|\mathbf{r}|^3}\mathbf{r} \tag{1}$$

where M is the planetary mass, m the mass of the rocket and G the gravitational constant. $\mathbf{r}$ is the position of the rocket relative to the centre of the planet, as indicated in the figure.



a) Using the 4th order Runge-Kutta approach discussed in Lecture 6, solve Eq. (1) for $\mathbf{r}$ and $\dot{\mathbf{r}}$. The motion will be in the xy plane, so it is sufficient to restrict your program to 2 dimensions. It is recommended that you write functions $f_1(t,x,y,v_x,v_y)$, $f_2(t,x,y,v_x,v_y)$, etc., for the derivatives given in Eq. (12.4) of Section 12.4 of the lecture notes, as this significantly simplifies the programming required (although, as also noted, you will not need the full list of arguments shown here). As an alternative it is permissible to use the `scipy.integrate.solve_ivp()` function that was demonstrated in the lecture. If you take this approach, you will need to write a single function that returns the values of $f_1(t,x,y,v_x,v_y)$, $f_2(t,x,y,v_x,v_y)$, etc., arranged as a vector, and call the function using the optional parameter `method='RK45'`.

As in previous exercises, please include a simple menu in your code so you can choose between running this part of the exercise, and the part that follows. The following code achieves this in a simple way – you may adapt it to your needs.

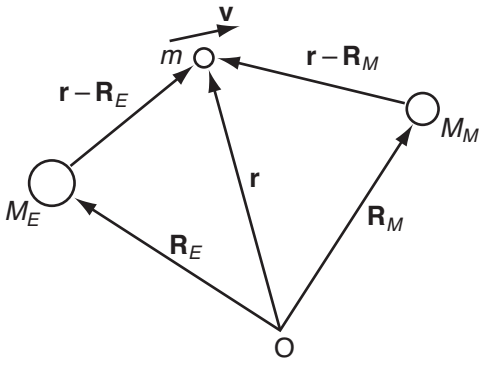
```
MyInput = '0'
while MyInput != 'q':
    MyInput = input('Enter a choice, "a", "b", "c" or "q" to quit: ')
    print('You entered the choice: ',MyInput)
    if MyInput == 'a':
        print('You have chosen part (a): simulation of an orbit')
        #
        # sample code for getting initial coords
        #
        Input_x = input('Enter a value for x (floating point position in meters): ')
        x0 = float(Input_x)
        Input_vx = input('Enter a value for vx (floating point component of velocity in m/s): ')
        vx0 = float(Input_vx)
        #
        # Simulation code here
        #
    elif MyInput == 'b':
        print('You have chosen part (b): shooting the moon')
        #
        # put your code for part (b) here
        #
    elif MyInput != 'q':
        print('This is not a valid choice')
print('You have chosen to finish - goodbye.')
```

Your code should generate plots of your orbital trajectories which you can include in your report.

In your report you should simulate both circular orbits and eccentric, comet-like, orbits. Are the orbits stable and repeatable? Examine the total energy of the rocket. Is it accurately conserved? What time-step do you need to ensure this accuracy?

Now for a 'real world' problem that uses what you have learnt so far. Your goal is to launch a rocket from low Earth orbit (orbital radius, say, between 6500 and 7000 km) such that it passes over the Moon's surface to take photographs, 'slingshots' around the moon, and then passes close to Earth once more to send back the information by radio. You have sufficient fuel for only one rocket 'burn' to leave Earth orbit, and the rest of the flight is simply coasting.

This is another example of numerical integration of a differential equation; the equation of motion results from the combined influence of the Earth and Moon's gravitational fields. To simplify the problem, treat the Earth and Moon as fixed in position for the duration of your flight.[1] Then the equation of motion becomes (see figure):



$$m\ddot{\mathbf{r}} = -\frac{mM_E G}{|\mathbf{r}-\mathbf{R}_E|^3}(\mathbf{r}-\mathbf{R}_E) - \frac{mM_M G}{|\mathbf{r}-\mathbf{R}_M|^3}(\mathbf{r}-\mathbf{R}_M) \qquad (2)$$

You can make the problem look simpler, by placing the Earth at the origin, so $\mathbf{R}_E = \mathbf{0}$. Then, it is also easier to place the moon on one of the axes.

b) Extend your Python script to simulate the motion of the rocket as it flies from the Earth to the moon and back, using the 4th order Runge-Kutta method, as before. Although the gravitational field is more complicated than previously, only the rocket is moving, so you can work with the same variables as in your first program.

Use your program to find the correct launch position (the point on the orbit where the rocket is fired) and velocity for the rocket to fulfil the requirements. You will need to use trial and error to establish these: produce plots of your trajectories, and devise tests to tell you if you have crashed into either Earth or moon.

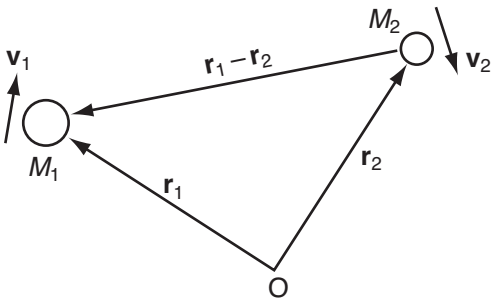
In your report, you should include discussion of the following points:

- How did you established reasonable starting conditions?
- How close can you get to the moon without crashing?
- How long does the flight to the moon and back take?
- How will you know when to halt your calculation?

As always, check that your numerical solution has sufficient accuracy. Think about ways to check this, using boundary conditions that should give known behaviour, and thinking about conservation of energy. Remember that your solution should be independent of step size.

Problem 2 (40% of marks): Earth and Moon and Sun

Now, you are going to make the problem more challenging still, by simulating a 2-body problem, where both bodies are moving under their combined gravitational forces. The equations of motion will become:



$$M_1\ddot{\mathbf{r}}_1 = -\frac{M_1 M_2 G}{|\mathbf{r}_1-\mathbf{r}_2|^3}(\mathbf{r}_1-\mathbf{r}_2) \qquad (3a)$$

$$M_2\ddot{\mathbf{r}}_2 = -\frac{M_1 M_2 G}{|\mathbf{r}_2-\mathbf{r}_1|^3}(\mathbf{r}_2-\mathbf{r}_1) \qquad (3b)$$

c) Write a Python script to simulate two similar bodies, each moving under the influence of the other, again using the 4th order Runge-Kutta method. Note that, in 2-dimensions, you will need to consider the components of the

[1] Note: This is unrealistic. The Apollo moon shots of the late 1960's and 1970's used to take around 5 days for the round trip from Earth to moon (around 500,000 miles), during which time the moon moves an appreciable distance in its orbit.

position vectors and velocities of both particles i.e. 8 variables, and 4 k's for each in the Runge-Kutta method. Test your program by simulating the motion of two equal bodies, and then the moon and the earth. If you take too large a time step, the motion of the two bodies can become unstable and unpredictable – chaos can ensue. Can you find evidence for this? Discuss your findings in your report.

d) Finally, as in Problem 1, you are going to add a stationary mass at the origin i.e. the Sun. Try to reproduce the orbit of the moon around the Earth as the Earth travels around the Sun.

Report

As previously, you should prepare a concise report outlining your methods and highlighting your findings with suitable graphs or tables. Credit will be given for a reasoned discussion of your findings.

Submitting your work

You should submit the following to Blackboard:

1. A brief report, in MS Word or pdf format;
2. Final version of your program for both the single body and the two body orbits.

As before, please note:

- Only upload your “prog.py" files.
- Blackboard anti-plagiarism software won't accept a “.py" extension so please rename your file with a “.txt" extension.
- **Please note, there is only a single upload point for your programs, so please only upload a single *.txt file to Turnitin.**
- Please also give your programs sensible distinguishing names, including your name or userid e.g. “my_userid_ex4_prob_1_and_2.txt".

If you have any problems submitting your work, please contact Dr. Hanna (s.hanna@bristol.ac.uk) or ask a demonstrator.