PHY407 Computational Lab 1
Python Programming Introduction

Computational Background

- The purpose of this lab is to review some basic concepts in programming using python.
- Concepts include: mathematical operations in python, loops, basic numerical integration and plotting.
- If you have little to no experience in programming or in using python (i.e. you didn't take PHY224 or PHY254 or a CSC course), then you need to carefully go through Chapters 2 and 3 of the text. Below I list some specific sections of the text that will be useful for this lab:
  - Assigning variables: Sections 2.1, 2.2.1-2.2.2
  - Mathematical operations: Section 2.2.4
  - Loops: Sections 2.3 and 2.5
  - Lists & Arrays: Section 2.4
  - Making basic graphs: Section 3.1
- Numerical integration review: For a general first order system:

$$\frac{d\vec{x}}{dt} = \vec{F}(\vec{x}) \tag{1}$$

the simplest way to numerically integrate the system is by approximating the derivative as:

$$\frac{d\vec{x}}{dt} \approx \frac{\Delta\vec{x}}{\Delta t} = \frac{\vec{x}_{i+1} - \vec{x}_i}{\Delta t} \tag{2}$$

and then rearranging the system to read:

$$\vec{x}_{i+1} = \vec{x}_i + \vec{F}(\vec{x}_i)\Delta t \tag{3}$$

We then start with an initial $\vec{x}$, pick a Δt and implement equation 3 in a loop to calculate the new value of $\vec{x}$ on each iteration. This is called the "Euler" method.

Physics Background

- **Newtonian orbits**: The Newtonian gravitational force keeping a planet in orbit can be approximated as:

$$\vec{F}_g = -\frac{GM_sM_p}{r^2}\hat{r} = -\frac{GM_sM_p}{r^3}\left(x\hat{x} + y\hat{y}\right) \tag{4}$$

where M_s is the mass of the Sun, M_p is the mass of the planet and r is the distance between them. Using Newton's law: $\vec{F} = m\vec{a}$ and numerical integration, we can solve for the velocity

and position of a planet in orbit as a function of time. (Note: I have assumed the planet is much less massive than the Sun and hence that the Sun stays fixed at the centre of mass of the system).

- **General relativity orbits**: The gravitational force law predicted by general relativity can be approximated as:

$$\vec{F}_g = -\frac{GM_sM_p}{r^3}\left(1+\frac{\alpha}{r^2}\right)(x\hat{x}+y\hat{y}) \tag{5}$$

where α is a constant depending on the scenario. Notice this is just the Newtonian formula plus a small correction term proportional to r^{-4}. Mercury is close enough to the Sun that the effects of this correction can be observed . It results in a precession of Mercury's elliptical orbit.

- **The Three Body Problem**: We can consider adding a second planet and see how the gravitational interactions between the Sun and the 2 planets affect the motions. This problem cannot be solved analytically. If we assume one of the planets is much more massive than the other, then we can just consider the gravitational influence of the larger planet on the smaller one and neglect the effect of the smaller object on the larger one (similar to how we treated the Sun in the 2 body problem above).

Lab Instructions

For grading purposes, you only need to hand in solutions to the following parts. Ensure that your codes are well commented and readable by an outsider:

- Q4: submit your pseudocode
- Q5: submit your python code and the 3 specified plots.
- Q6: submit your plot of the orbital precession.
- Q7: submit your python code and a plot showing both Jupiter's and Earth's orbits.
- Q8: submit the plot of the orbit in the x,y plane.

Modeling a planetary orbit

1. Using equation 4 and Newton's law, convince yourself that the equations governing the motion of the planet can be written as a set of first order equations (i.e. of the form of

equation 1) in cartesian coordinates as:

$$\frac{dv_x}{dt} = -\frac{GM_s x}{r^3} \tag{6a}$$
$$\frac{dv_y}{dt} = -\frac{GM_s y}{r^3} \tag{6b}$$
$$\frac{dx}{dt} = v_x \tag{6c}$$
$$\frac{dy}{dt} = v_y \tag{6d}$$

2. Rearrange equations 6 to put in a format similar to equation 3 so that they can be used for numerical integration. I.e., replace the derivatives on the L.H.S. of the equations with equation 2 forms, then isolate the "i+1" component of the variable on the L.H.S. Move everything else to the R.H.S. You should end up with 4 equations. One each for: $\mathrm{v}_{x,i+1}$, $\mathrm{v}_{y,i+1}$, x_{i+1}, y_{i+1}.

3. You could try and code this up, but it turns out that this system is a little temperamental when the Euler method is used. Instead, we will slightly tweak the Euler method into the "Euler-Cromer" method and use that. Later in the course you will understand why, but for now, just make the following slight change to your equations from question 2:

   When you update the position variables x and y, use the newly updated velocities instead of the old velocities. So these lines should look like:

$$x_{i+1} = x_i + v_{x,i+1}\Delta t \tag{7a}$$
$$y_{i+1} = y_i + v_{y,i+1}\Delta t \tag{7b}$$

   The updates for the velocities should remain as they are.

4. Write a "pseudocode" for a program that integrates your equations to calculate the position and velocity of the planet as a function of time under the Newtonian gravity force. The output of your code should include graphs of the components of velocity as a function of time and a plot of the orbit (x vs y) in space.

5. Now write a real python code based on your pseudocode. Constants you will need for the code can be found in Appendix A. We will assume the planet is Mercury. The initial conditions are:

$$x_M = 0.47\,\mathrm{AU}, \quad y_M = 0.0\,\mathrm{AU} \tag{8a}$$
$$v_{x,M} = 0.0\,\mathrm{AU/yr}, \quad v_{y,M} = 8.17\,\mathrm{AU/yr} \tag{8b}$$

   (Note: These aren't arbitrary. I derived the velocities by using conservation of angular momentum and energy for the orbit).

Use a time step $\Delta t = 0.0001$ yr and integrate for 1 year. (Note: our unit of year here is actually the Earth year, so your integration will cover several Mercury years). You should have found an elliptical orbit for Mercury.

Hints:

- Define all of your constants in easy to work with units. That means use solar masses as your mass unit, AU as your distance unit and year as your time unit. "G" in these units is provided in appendix A.
- If you are confused about how to go about this, follow the steps in lecture 1: (1) Discretize, (2) Initialize, (3) Evaluate. Time is your independent variable. Position and Velocity components are your dependent variables.
- Make sure to label all of your plot axes and include legends if you have more than one curve on a plot.

The precession of the perihelion of Mercury due to GR

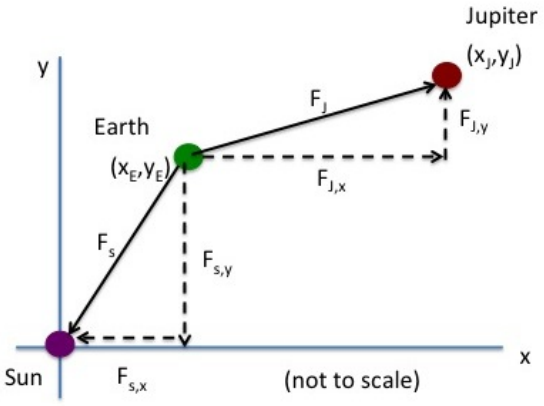
6. Now alter the gravitational force in your code to the general relativity form given in equation 5. The actual value of α for Mercury is given in Appendix A, but it will be too small for our computational framework here. Instead, try $\alpha = 0.01$ AU^2 which will exaggerate the effect. Demonstrate Mercury's orbital precession by plotting several orbits in the x, y plane that show the perihelion (furthest point) of the orbit moves around in time.

The 3 Body Problem

Now lets add another planet to our system. We will consider Earth as the small planet and Jupiter as the large planet. The orbit of Earth will be determined by the gravitational force of the Sun at the centre of the solar system and the gravitational force of Jupiter at an orbital radius of 5.2 AU. We will assume that Jupiter and the Sun are not affected by Earth's gravitational force.

7. Write a pseudocode to add Jupiter to the system. Then alter your code from Q5 (i.e. the Newtonian gravity code) to add Jupiter and integrate the orbits for 10 years. Relevant data for Jupiter are given in Appendix A and the figure below might be of some help. Some steps you will need to take:

   - Simulate Jupiter's orbit
   - Calculate the distance between Jupiter and Earth as a function of time.
   - Determine the net gravitational force on Earth due to both the Sun and Jupiter.



F_J: Gravitational force on Earth due to Jupiter
F_S: Gravitational force on Earth due to the Sun
(x_J,y_J): coordinates of Jupiter
(x_E,y_E): coordinates of Earth

Use the following initial conditions for Jupiter:

$$x_J = 5.2\,\text{AU}, \quad y_J = 0.0\,\text{AU} \tag{9a}$$
$$v_{x,J} = 0.0\,\text{AU/yr}, \quad v_{y,J} = 2.63\,\text{AU/yr} \tag{9b}$$

Use the following initial conditions for Earth:

$$x_E = 1.0\,\text{AU}, \quad y_E = 0.0\,\text{AU} \tag{10a}$$
$$v_{x,E} = 0.0\,\text{AU/yr}, \quad v_{y,E} = 6.18\,\text{AU/yr} \tag{10b}$$

Hints (I'm providing these because I know you didn't have a lot of time to prepare this week):

- To calculate the distance from Earth to Jupiter at each loop iteration (for the 'i' iteration) you can use the following in your loop:
  rej = sqrt((j_x[i] - e_x[i])**2 + (j_y[i]-e_y[i])**2)
- To calculate the x-component of the force on the Earth (the y-component is similar) use:
  force_x = -G*Ms*e_x[i]/e_r**3 + G*Mj*(j_x[i]-e_x[i])/rej**3
  where e_r is the distance from Earth to the sun (so you will have to calculate that before hand.

8. You should find that Jupiter doesn't have a big effect on Earth. Lets see what happens if Jupiter were more massive. Set Jupiter's mass to be 1000x its actual mass (i.e. if it had the same mass as the sun, note: now our approximation of Jupiter not affecting the Sun is terrible, but hey, just for fun). Run your code again with the same initial condition for Jupiter as in question (7) and plot the orbit in the x, y plane for 3 years (if you run any longer, Earth gets ejected, try it).

Appendix A: Constants

- Because we are working on astronomical scales, it will be easier to work in units larger than the meter, second and kilogram. For distances, we will use the AU, for mass, M_s and for time, the year. I've provided some constants you will need in these units below:

  - $M_s = 2.0 * 10^{30}$ kg $= 1M_s$ is the mass of the Sun.
  - 1 AU $= 1.496 * 10^{11}$m. It is roughly the average Earth-Sun distance.
  - $G = 6.67 * 10^{-11}$ $\mathrm{m^3 kg^{-1} s^{-2}} = 39.5$ $\mathrm{AU^3} M_s^{-1} \mathrm{yr^{-2}}$.
  - $\alpha_M = 1.1 * 10^{-8}$ $\mathrm{AU^2}$. For the code, use $\alpha_M = 0.01$ instead.
  - $M_J = 10^{-3} M_s$ is the mass of Jupiter in solar units.
  - $a_J = 5.2$ AU is Jupiter's orbital radius.

  Notes:

- The text uses python version 3 whereas if you are using the UofT physics distribution, you are probably using version 2.7. There are slight differences between these versions. The text discusses them in Appendix B, but this probably won't affect much in this lab.