PROGRAMMING ASSIGNMENT 4

Due: Thursday, October 22 by 11am. Upload *one* Jupyter notebook to Gradescope.

1. KEPLER'S LAWS AND THE STABILITY OF PLANETARY ORBITS

Write a program that computes the *elliptical* orbit of the Earth using the Euler method *and* the Euler-Cromer method. You may use the familiar force of gravity equation, $F = G\frac{Mm}{r^2}$. *Hint*: verify that your code successfully produces a circular orbit when the eccentricity is set to zero!

- For which method does Earth's orbit become *unstable*? Does it matter what time step size you use? Create one or more plots, with the Sun at the center, showing Earth's orbit over several years using both algorithms and at least three choices of time step (ranging from good to very bad).
- Using whichever method produces a stable orbit, measure the perihelion and aphelion distances. Comment on the agreement between your code's output values with the accepted and/or input values.

Include a few grammatically correct statements describing which method(s) fail to produce stable planetary orbits, time step size considerations, as well as your measurements answering the above questions.

Important hint! You *should* find yourself needing to compute an angle θ at some point in your program using Python's built-in trigonometric functions. Python has *two* different tools for doing this: one tool that forces θ to remain in the first quadrant (e.g., $|\theta| = 0° - 90°$) and one that allows all four quadrants (e.g., $|\theta| = 0° - 360°$). Example: np.arctan only yields angles in the first quadrant, but np.arctan2 will span the full range of angles that are necessary to compute planetary orbits. Whatever trigonometric command you use, *verify* that it returns the full range of possible orbital angles, otherwise your orbits will spiral out of control very quickly!

Grading Rubric: Program has required features described in prompt (30%), graphs show correct behavior and are easy to interpret (30%), written response is factually correct (20%) and uses proper grammar and spelling (10%), code compiles without errors and reproduces all output (10%)

2. CONSERVATION LAWS IN PLANETARY ORBITS

Since gravity is a conservative, central force, we expect specific physical quantities of an orbiting body to be conserved. Modify your code from the previous problem to create a program that simulates the orbit of a planet using the Euler-Cromer method, and calculates the energy (kinetic, potential, and total) and the angular momentum of the planet. Show that:

- with a circular orbit, the kinetic energy, potential energy, and angular momentum are constants.
- with an elliptical ($e = 0.05$) orbit, while the kinetic and potential energies may vary as the planet moves through its orbit, their sum (the total energy) is constant, and the angular momentum is constant.

You don't *have* to use Earth as your test planet for this problem, but it's convenient to do so. If you do, you can use a starting position of 1 AU and an orbital speed of 2π AU/yr for your initial conditions. *Note*: it is okay if your simulation uses non-standard units of energy and angular momentum (we only care about the overall trends in energy and angular momentum with time, not their absolute values).

Describe how your plots support the conclusion that gravity is a conservative, central force.

Grading Rubric: Program has required features described in prompt (30%), graphs show correct behavior and are easy to interpret (30%), written response is factually correct (20%) and uses proper grammar and spelling (10%), code compiles without errors and reproduces all output (10%)

3. The Precession of Mercury's Orbit

We're going to do something very similar to Einstein's famous 1916 work on Mercury's rate of orbital precession – one of the earliest tests of the theory of general relativity. Use a modified gravitational force law:

$$F = G\frac{Mm}{r^2}\left(1 + \frac{\alpha}{r^2}\right)$$

where m is the mass of Mercury, M is the mass of the Sun, r is the distance between the Sun and Mercury, and α is a parameter that we will investigate. You should read Section 4.3 of your book *very carefully*!

Start with $\alpha = 0$ (the regular Newtonian formulation of gravity) and find the initial conditions that replicate Mercury's orbit: perihelion at 0.31 AU, aphelion at 0.46 AU, and orbital period of 0.24 years. You must verify, with a set of plots and/or calculations from your code, that you have successfully replicated Mercury's orbit in the $\alpha = 0$ case.

Once you've replicated Mercury's orbit, gradually increase the value of α and see how the precession rate (in radians per orbit *or* degrees per year, whichever you prefer) depends on α. Hint: It is helpful to create a plot of the precession angle as a function of time, which *should* look roughly linear if your code is correct. Then you can fit a line to your "data," and the slope gives you the change in angle with time (i.e., the precession rate). Do this for $\alpha = 0.001, 0.002, 0.003, 0.004$, and see if your graph of the precession rate has the same slope as Figure 4.10 on page 112. *Note*: Figure 4.10 uses precession rate units of *degrees* per year, so if you are measuring *radians* per orbit, you will need to convert radians/orbit → degrees/year to compare your results to Figure 4.10.

Mercury's orbit is observed to precess at a rate of about 43 arcseconds per century. Using this and your results, what is the true value of α?

Grading Rubric: Program has required features described in prompt (30%), graphs show correct behavior and are easy to interpret (30%), written response is factually correct (20%) and uses proper grammar and spelling (10%), code compiles without errors and reproduces all output (10%)

4. A Two-Planet Solar System

Example 4.2 in the textbook sketches out a program that simulates the (presumed to be circular) orbits of both Earth *and* Jupiter around the Sun. Use this outline to write your own two-planet Solar System code, and verify that it works as expected by plotting the orbits of Earth and Jupiter around the Sun. Be sure to run your simulations for several complete orbits of Jupiter (not Earth) to verify the stability of the orbits.

Once your code is working, modify it to investigate the following (show plots supporting your conclusions):

- How much does Earth's orbital distance from the Sun vary because of perturbations from Jupiter?

- What is the minimum mass Jupiter would have to possess, at its present orbital distance, to disrupt Earth's orbit? Define "disrupt" to mean that Earth's orbit would cross either Venus' orbit (at its innermost point, 0.72 AU) or Mars' orbit (at its outermost point, 1.5 AU).
- How close would Jupiter (at its current mass) have to be to Earth to eject it from the Solar System? *Hint*: Either you can "brute force" this question and manually adjust Jupiter's distance, or you can use what you found in Question 2, that planets that are gravitationally *bound* to their stars will have negative potential energies. As a planet becomes *unbound*, $U \rightarrow 0$.

Now consider the fact that Jupiter is much closer to Mars in its orbit, and Mars is a much lower-mass planet than Earth. Adjust the mass and orbital distance of your inner planet to be that of Mars, and answer the same question as above for Mars. In this case, Earth's orbit at 1 AU will set the innermost limit of Mars' orbit, and the asteroid belt (about 2.5 AU) sets the outermost limit. Note: the best type of code is one where you write a function to compute the orbits, then give it as input the planets you wish to model and any modifications to the orbital parameters.

Grading Rubric: Program has required features described in prompt (30%), graphs show correct behavior and are easy to interpret (30%), written response is factually correct (20%) and uses proper grammar and spelling (10%), code compiles without errors and reproduces all output (10%)