Physics 307 Homework 6

Due Hallowe'en before class

In this homework assignment, you will simulate two objects moving under each other's mutual gravity, then either extend this to three-body systems, or add a drag force that crudely approximates gravitational wave emission and observe the inspiral of the two objects, optionally producing a sound file that sounds (at least in a general sense) like the "chirp" observed by LIGO.

1. **Before** you write any code, write the section of your report that describes how you will generalize from "one object orbiting the fixed Sun" to "two objects orbiting each other". How many dynamical variables will you have, and what are they? How do you handle the fact that the attracting object is no longer at the origin, and in fact isn't at a fixed location at all? How will you implement the leapfrog algorithm? (You don't have to write many paragraphs, but you should think about how you'll do these things in some substance before you write code.)

2. Implement a simulation of the mutual gravity of orbiting stars. Note that if you measure masses in solar masses, $G = 4\pi^2$. Note also that the total momentum must be zero or your simulation will drift away from the viewport; to prevent this, compute the center of mass velocity and subtract it:

   ```
   vx_com = (vx1*m1 + vx2*m2) / (m1 + m2);
   vx1 = vx1 - vx_com;
   vx2 = vx2 - vx_com;
   ```

   and similarly for y.

   Play around with your initial conditions until you get two stars orbiting each other.

3. Try simulating the Jupiter + Sun system, looking up the mass and orbital radius of Jupiter on Wikipedia. (The orbital velocity can be found with a bit of Physics 211 math, since $GMm/r^2 = mv^2/r$.) How far does the Sun move? Would this be observable from another star? How *fast* does the Sun move? (In practice, the wobbling of a star is most easily detected by the Doppler shift due to this velocity.)

4. For now, return to circular orbits. (Two stars orbiting each other in a circle with radius 1 AU will have velocity π AU/yr.) Then add a drag term to your stars to simulate radiative loss from gravitational waves.

   This drag force will have the form

   $$\vec{F}_d = \beta(-\hat{\vec{v}})v^n$$

where n is some power (2 for air drag, for instance) and β is a constant. At long distances, $n = 9$ for gravitational waves, but this is quite fiddly to work with in this simulation at first.

Converting this to Cartesian coordinates, you have

$$(F_d)_x = -\beta v_x v^{n-1} \tag{1}$$
$$(F_d)_y = -\beta v_y v^{n-1} \tag{2}$$

which you can put in your leapfrog integrator.

Play around with the value of the drag constant and power (observing that β must decrease as the power increases). For objects around one solar mass which start their orbit at around 1 AU from each other, β around 10^{-3} and $n = 2$ is a good starting point.

As time elapses, what do you observe happening to the stars' orbit? How does its size change? How does its period change? Is this consistent with the LIGO observations?

5. Now change your initial conditions so that your stars move in an elliptical orbit, and observe how this orbit changes with time. Does the orbital eccentricity increase or decrease with time as the orbit decays? Look at the spectrogram in the recent gravitational-wave paper (https://arxiv.org/pdf/1710.05834.pdf) on page 10, and note that there is only one frequency, i.e. the neutron stars' orbit was a pure sine wave. Is this consistent with your simulation?

6. (Ambitious folks, particularly anyone familiar with *Audacity* or other sound-editing software) Using the sound output program I will upload on Thursday, print the x-velocity of one of the stars as a function of time to a file and "listen" to the chirp for different initial conditions. Does this sound similar to audio you've heard of the "chirp" from LIGO?

7. **Graduate students and ambitious undergrads:**

   Once you know how to simulate two planets, it shouldn't be a challenge to simulate N planets. You can do this using a programming construct called an *array*; I will post the notes on arrays next week, or you can ask me.

   It'll take three nested `for` loops:

   - One looping over timesteps
   - One looping over planets feeling forces
   - One looping over planets applying forces

   ... since every planet pulls on every other planet.

   Use this code to simulate a system with a star, a planet, and a moon.