Astronomy 8824: Problem Set 2

Due Tuesday, September 17, 2019

Numerical Integration of Orbits

Note: Creating good plots for this assignment is a non-trivial amount of work. You are welcome to either use David's sm scripts (available from his web page), or use the matplotlib examples I have put on our class web page in a "Starter Notebook" to create your own.

Part 1: Kepler Potential

Consider a 2-d Kepler potential generated by a central point mass M. Adopt units G = M = 1 so that

$$\Phi(r) = -\frac{1}{r}$$

In these units, what is the circular velocity v_c at r = 1? What is the orbital period for an orbit with semi-major axis a = 1? If we convert to physical units adopting M = 1 $M_\odot$ and a length unit of 1 AU, what is the time unit in years (i.e. what does an interval dt = 1 in system units correspond to in years)?

Write a program to integrate test particle orbits in this potential, in Cartesian coordinates.

First implement this with the Euler method, where you update positions and velocities from time t_i to $t_{i+1} = t_i + \Delta t$ using:

$$x_{i+1} = x_i + v_i \Delta t$$
$$v_{i+1} = v_i - \nabla\Phi(x_i)\Delta t$$

If you wish, you may start with David's program `orbit_euler.py`, available from his web page, and then adapt it for subsequent parts. His implementation takes command line arguments and writes results to standard output in the right format for his plotting scripts `sm.orbits.kep` and `sm.energy.kep`. To go through all the values below and make files with appropriate filenames for the plotting scripts, he used the shell script `sh.kepruns`, which can be run with the command `sh sh.kepruns`.

Modify the Euler program to implement the leapfrog method, where you evaluate the velocities at half-integer timesteps:

$$x_{i+1} = x_i + v_{i+1/2}\Delta t$$
$$v_{i+3/2} = v_{i+1/2} - \nabla\Phi(x_{i+1})\Delta t$$

Remember that in this case you need to take the first half time-step to evaluate $v_{1/2}$ – don't just set $v_{1/2} = v_0$.

Start with a test particle at $x = 1, y = 0, \dot{x} = 0, \dot{y} = 1$. Integrate the orbits up to *t=40* with both methods, first with a timestep Δt = 0.01, then with a timestep Δt = 0.0001.

Plot the orbits for the two methods and two integration steps. Also evaluate the energy

$$E = \frac{1}{2}v^2 + \Phi(\vec{x})$$

and make plots showing the level of energy conservation for the four cases. What other conserved quantity could you use as a check of your integration accuracy?

Repeat with initial velocity $\dot{y} = 0.5, \dot{y} = 0.2$. Include plots for all three values of $\dot{y}$ with your solution.

Comment briefly on the impact of timestep and the difference between Euler and leapfrog integration.

Here are a few issues to consider in your coding and plotting. Rather than get output every timestep, you will want to code so that you can separately specify how often you get outputs; you probably don't need 400,000 outputs for the Δt = 0.0001 case. When using the leapfrog integration, you need to take an extra half step to synchronize the position and velocity to the same time before evaluating the energy at your output time (this is another reason you don't want output at every timestep). Be sure this is just a temporary evaluation; you want to get velocities back on the 1/2 timesteps when you go back to orbit integration. If you plot *y* vs. *x* with the same range and the same units, your plot panels should be square so that the shape of the orbit is faithful. If you let your plotting code automatically select axis ranges based on the variation in your quantities, then it will make even highly elliptical orbits appear circular, and only reading the axis values will reveal their ellipticity – very bad practice! Where practical, if you're comparing different cases (e.g., different timestep choices) keep the same axis ranges so that one can visually evaluate the comparison; in some cases, like energy conservation that varies by orders-of-magnitude from one case to another, this won't be a useful approach. Label your plots well so that it is evident what case you are showing. If you're using sm, David's macros ``fourwin'' and ``sixwin'' and ``sixwin2'' are useful for cases like these where you want to put multiple square panels on a single page for comparison. My matplotlib example `plot4xy()` in the starter notebook shows one way to make similar plots in python.

Part II: Harmonic Oscillator Potential

Now consider a potential

$$\Phi(\vec{x}) = \frac{x^2}{2} + \frac{y^2}{2q^2}$$

Using the leapfrog integrator with Δt = 0.0001, integrate and plot orbits for q = 1, q = 0.9, and q = 0.6, always starting at $x = 1, y = 0, \dot{x} = 0$ with initial y-velocities $\dot{y} = 1, \dot{y} = 0.5$. Create a 6-panel plot that shows the orbits for these six cases. David has a plotting script `sm.orbits.ho` that you may use. Another option is to adopt the plot4xy() routine in my starter notebook to display all six cases.

For the case $q = 0.9, \dot{y} = 0.5,$ compare the energy conservation of the Euler and leapfrog methods for Δt = 0.01, 0.001, and 0.0001. You don't need to include plots of the orbits for

these other cases, but do some spot checks to see whether the orbit is visibly different from leapfrog with Δt = 0.01.

Part III: Planet and Moon

Adapt your leapfrog integration program to integrate the orbits of three bodies interacting gravitationally.

The most massive body, the star, should have a mass $M_s = 1$ and start at rest at position (x,y) = (0,0).

The next most massive body, the planet, should have a mass M_p and start at position (x,y) = (1,0) with an initial velocity $\dot{y}_p$ in the +*y* direction. As in Part I, $\dot{y}_p = 1$ corresponds to a circular orbit for the planet, provided the perturbing effect of the third body can be ignored.

The third body, the moon, should have a mass M_m and start at position (x_m, 0) with an initial velocity $\dot{y}_m$ in the +*y* direction.

Before you start integrating, you should subtract the center-of-mass velocity from all 3 bodies so that you work in a frame in which the center-of-mass stays fixed. Otherwise, your bodies will just drift off the edge of any plot you make.

For the calculations in this part use Δt = 0.0001 as a default, though you may want to experiment with larger or smaller timesteps in some cases.

First consider cases with $M_p = 0.1$ and $M_m = 0.01$. Note that if we imagine $M_s = 1\ M_\odot$ then the "planet" is really a star at the bottom of the main sequence and the "moon" is a brown dwarf 10 times the mass of Jupiter!

Suppose that $x_m = 1.05$. What initial value of $\dot{y}_m$ is required for the moon to be in an approximately circular orbit about the planet? (Think about which frames are relevant.)

3.1 Integrate and plot this case. David's plotting routine `sm.orbits.moon` has several nice features: it plots the position of the moon relative to the planet in an inset window; it plots a zoom around the initial position in another starting window to show perturbations of the planet orbit; and it changes the line color used for the moon's orbit each time the planet completes one orbital period, making it much easier to decode what is happening over time. Note that in the file names and plotting script he specifies the initial $\dot{y}_m$ of the moon *relative* to the planet and in units of the velocity v_c required for a circular orbit at the planet-moon separation (ignoring the star). I have put an equivalent matplotlib version of this routine called `plot3body()` into the starter jupyter notebook on the class website. Both of us do the conversion to a $\dot{y}_m$ in system units in the star's frame within our python integration scripts.

3.2 Calculate and plot the case with $x_m = 1.1$, again with $\dot{y}_p = 1$ and $\dot{y}_m$ chosen to put the moon in a circular orbit about the planet

3.3 Repeat for $\dot{y}_m = 1.2$.

Comment on the results for these three cases. At what separation is the orbit of the moon no longer well described as a circular orbit centered on the moving planet? Can you explain the value of this critical separation physically? (Hint: think about tidal forces.)

3.4 Go back to $x_m = 1.05$, but now set $\dot{y}_p = 0.7$ so the planet is in an elliptical orbit. Compute, plot, and comment.

3.5 Set $\dot{y}_p = 1$ but choose the moon velocity relative to the planet to be 0.7 v_c – i.e. 70% of the velocity required to put the moon in a circular orbit. Compute, plot, and comment.

3.6, 3.7 Increase to $x_m = 1.10$ and try a retrograde orbit for the moon (negative $\dot{y}_m$), first at the circular speed, then at 70% of the circular speed. Compute, plot, and comment.

3.8 – 3.10 Finally, consider a lighter planet and moon: $M_p = 0.01$, $M_m = 0.0001$. Do just the cases with $\dot{y}_p = 1$ and a circular orbit for the moon, setting $x_m = 1.05, 1.08, 1.10$. Comment on the similarities and differences from the case with M_p=0.1 and M_m=0.01.

Part IV: The Three-Body Problem

Time permitting: Experiment with cases in which all three bodies have equal mass. You may need shorter timesteps for some cases.

See if you can set up a “hierarchical triple," in which a distant tertiary star orbits in an approximately circular orbit around a tight inner binary.

After that, play around and see what you find. Include plots of your most interesting results and interpret them.

An interesting set of controlled cases is to set up an inner binary in an approximately circular orbit, then “drop in" a third body from different distances along the +*x* axis, always setting its initial $\dot{y}$ equal to that of the binary center-of-mass so that the third body drops straight down the x-axis.

Note: This problem set was developed by David Weinberg.