AST 384C: Computational Astrophysics

HW 1 (due by noon central time on Friday, February 14)

Complete the following 2 questions by submitting (documented) code and any accompanying answers / plots in a github repository. Email me the repository link once you've committed your solutions. Make sure to clearly document your code; when in doubt, over-explain!

1. One of the simplest methods for integration is the Rectangle Rule (piecewise constant):

$$I = \sum_{i=1}^{N} f(x_i)\Delta x, \tag{1}$$

where N is the number of (equal-sized) integration steps and $\Delta x = (b-a)/N$, $x_i = a + (i-1)\Delta x$.

Another *slightly* more sophisticated approach uses the Trapezoid Rule (piecewise linear):

$$I = \Delta x \sum_{i=1}^{N} \frac{f(x_i) + f(x_{i+1})}{2}. \tag{2}$$

(This may seem to require twice as many function evaluations as the method above, but there is a simple way to avoid this, which you should try to figure out and implement.)

A third, more accurate approach is to adopt Simpson's rule, which uses parabolas. Here, the integral on the interval (a, b) can be expressed as

$$I(a,b) = \frac{\Delta x}{3}\left[f(a) + f(b) + 4\sum_{i=1}^{N/2} f(a + (2i-1)\,\Delta x) + 2\sum_{i=1}^{N/2-1} f(a + 2i\,\Delta x)\right] \tag{3}$$

Compute the integral

$$\int_1^5 \frac{1}{x^{3/2}} dx \tag{4}$$

with the three methods above and plot the error in the numerical integral against the step size Δx (or number of bins N) for several values of $N \in [5, 1000]$ for each method. Time each calculation as well (`time.timeit` will work well in python). Approximately how many steps are required to get an answer with a relative error of $|I - I_{\rm exact}|/I_{\rm exact} < 10^{-3}$? What about 10^{-5}? What did you lean about the trade-off between method, accuracy and calculation speed? (*Note: you can test your implementation against the implementations in* `scipy.integrate`.)

2. In class (Thursday, January 30), we talked about generating positions that follow a Hernquist (1990) density profile, which is a good match to galaxy bulges and dark matter halos. It is also convenient in that (1) the total mass is finite, and (2) the density profile can be expressed directly as a function of the gravitational potential.

   For this question, you will go one step farther and also generate **velocities**. These are determined by requiring that the Hernquist halo be spherical and isotropic and obey the Poisson equation, which relates density to potential. This means the distribution function $f(\mathbf{x}, \mathbf{v})$ is a function of energy E alone, $f = f(E)$ (see equation 17 of Hernquist).

   Assume the following parameters for the density profile:

   - $M = 10^{12}\, M_\odot$
   - $a = 35\,\mathrm{kpc}$
   - total number of particles: $N = 10^6$

   Your solution for this problem should include (1) executable code (that I can compile and run or just run) that produces the positions and velocities of the particles, and (2) a description of what your code does (i.e., what you've done and why; basically, extensive commenting). You can use as a starting point the notebook that I distributed from class on 2025.1.30 if you like.

   Some hints:

   - For each particle i, you'll need to randomly sample from the underlying velocity distribution function, $p(v|r_i)$. Note that $p(E|r_i) \propto f(E)\,\sqrt{\phi_i - E}$ where $\phi_i = \phi(r_i)$ is the gravitational potential for a particle at position r_i (see Eq. 5 from Hernquist) and $E = \phi(r_i) - v^2/2$ is the binding energy. This means you can use the distribution function $f(E)$ to select an energy corresponding to each particle's radius, then convert that energy to a velocity. For each position r_i, there is a maximum energy attainable, which you'll have to take into account when sampling from $f(E)\,\sqrt{\phi_i - E}$. One way to sample the distribution function $f(E)\,\sqrt{\phi_i - E}$ is via the acceptance-rejection technique (which can be found in Press et al. 1993, Numerical Recipes; I put a copy of this in the Files section on Canvas on 2025.01.30). This is a very handy technique to have in your toolbox.
   - Then, you'll have to get random velocity components v_i from the magnitude of the velocity vector $|v|$.

   Test that your velocities are correctly sampled by computing the radial component of the velocities, $v_r \equiv \vec{v} \cdot \hat{r}$ and computing $\langle v_r^2 \rangle^{1/2}$ in (logarithmic) radial bins; this quantity is the velocity dispersion and the square of it is given in Eq. 10 of Hernquist, so compare your result to that.

   How efficient was your code? What part was the slowest? Is it equally efficient in selecting velocities at all radii, and why or why not?