ASTR GU4260: Problem Set #1

Due: Wednesday, 20 September 2023

GitHub Classroom Assignment Link: `https://classroom.github.com/a/YK4bV2Dn`

As explained in the course description, solutions should be submitted via your GitHub assignment repository Each problem set should be in its own subdirectory in the repository linked at the beginning of this problem set (note that this is a different one than for Problem Set 0). The solution should include either a documented Jupyter notebook (preferable) that solves the problem(s), and acknowledges any collaborators, or the solution as a standalone program, and a copy of the output, accompanied by a README file (which includes your name, UNI, and acknowledgment of collaborators).

Problem 1

Write a program to print out the value of x and $\ln(x)$ where x ranges from 1 to 10 in steps of 0.5. Note that you will need to `import math` to use the python math functions.

Problem 2

Here is a more complicated function. In terms of coding, you don't need to know what it does, but we'll use it again (and again!) in the course, so it's worthwhile understanding a bit of the background. Don't worry if you don't completely understand the following paragraph on first reading, but do try to use it for plausibility testing of your results.

Exoplanets orbiting other stars are too faint to be seen directly, but they can be identified if their orbit takes them between us and their host star, blocking out some of its light and resulting in a short dimming of the star as seen on earth. This period of dimming is called a *transit*. The function below computes how much a star is dimmed based on two input arguments about the planet and its orbit[1]. Basically the two parameters are how big the star is and where it is along the orbit. In more detail: we can model the transit as an eclipse of a spherical star of uniform brightness by an opaque, dark sphere (see Figure 1 on the next page). In what follows, d is the center-to-center distance between the star and the planet (as seen by the observer), r_p is the radius of the planet, r_* is the stellar radius, $z = d/r_*$ is the separation of the centers, normalized by the stellar radius, and $p = r_p/r_*$ is the ratio of the planet radius to the stellar radius. The observed flux relative to the unobscured flux is $F(p, z)$.

The problem is to write code to evaluate the following function:

$$F(p, z) = 1 - \lambda(p, |z|) \tag{1}$$

where

$$\lambda(p, z) = \begin{cases} 0 & 1 + p < z \\ \frac{1}{\pi}\left[p^2\kappa_0 + \kappa_1 - \sqrt{\frac{4z^2 - (1+z^2-p^2)^2}{4}}\right] & |1 - p| < z \leq 1 + p \\ p^2 & z \leq 1 - p \\ 1 & z \leq p - 1, \end{cases} \tag{2}$$

and $\kappa_1 = \cos^{-1}[(1 - p^2 + z^2)/(2z)]$, $\kappa_0 = \cos^{-1}[(p^2 + z^2 - 1)/(2pz)]$. You may need to write code specifically to deal with floating point round-off errors. Note that the inverse cosine $\cos^{-1}()$ is written in python as `math.acos()`, and similarly for $\sin^{-1}()$. Finally, the absolute value of $|x|$ is given in python by `math.fabs(x)`.

Use your function to evaluate $F(p, z)$ for $p = 0.2$ and z going from -1.2 to 1.2 by steps of 0.05.

Graduates: Use `matplotlib` to plot the result, with axes labeled.

[1] See Mandel & Agol, 2002, ApJ, 580, L171

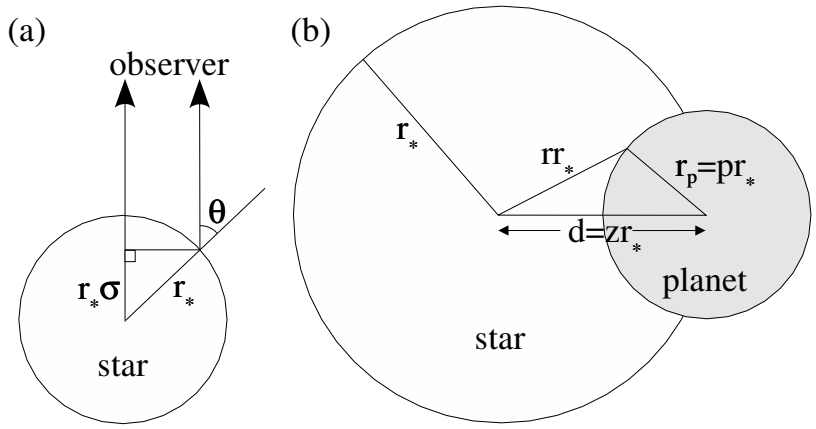

Figure 1: (a) Geometry of the eclipse. The star is seen edge on, with the observer off the top of the page. The star has a radius r_* and θ is defined as the angle between the observer and the normal to the stellar surface, while $\mu = \cos\theta$ (although we will not use definitions in this problem). (b) The transit geometry from the perspective of the observer.

Problem 3: Simple numerical integration

Write a function to compute an integral using a simple "rectangle"-rule: in other words approximate it as a sum with a small step in r. For example, you can write an integral of the form:

$$\int_{x_0}^{x_N} f(x)dx \approx \sum_{i=0}^{N-1} f(x_i)\Delta x \tag{3}$$

where $\Delta x = (x_N - x_0)/N$ and $x_i = x_0 + i\Delta x$ (Note that we covered this rule in class.) For future work, it will be useful to use a function definition for the integrand, so that you can use the same integration code with different integrands.

Use your function to integrate the following function:

$$f(x) = \sin x \tag{4}$$

from $x = 0$ to $x = \pi/2$.

You can test your numerical result against the analytic solution - how close do you get? Do this for $N = 10, 10^2, 10^3, 10^4, 10^5$ and determine how the fractional error (i.e., the absolute difference between the true and the estimate result, divided by the true result) decreases as N increases. Based on this, what is the order of the scheme (i.e. is it first-order, second-order, etc.)?

Problem 4: [Graduates] Singularities

As an example of a function with a singularity at the edge of the integration range, consider the integral

$$\int_0^1 \frac{\sin x}{x} dx. \tag{5}$$

If you try to use your previous code to integrate this function, it will generate a divide by zero error at the lower boundary. Modify the code to avoid this while still getting an accurate result. (It may be useful to plot the function to understand its behavior.)