ASTP 720 Homework Assignment 1

Isabella Cox – `igc5972@rit.edu`

September 2, 2020

Task 1

Write a library (i.e., in a separate file that you can call) for the three root-finding algorithms we discussed in class: Bisection, Newton, Secant. These functions should each take functions rather than data points. Make sure that each takes an optional argument the threshold and that it also takes a variable that allows the user to print out or return the number of iterations it took to hit that threshold.

This library exists in a file called: `root_library.py`.

The bisection method works by updating the bounds on either side of the root until the functional value of the midpoint between the bounds is within the threshold (and thus, the root is found). In the case that the original bounds are not possibly on either side of the root, an error message is returned to user.

The Newton method works by constructing a linear function (tangent to the curve) and when the tangent's functional value at a given x value hits the threshold, the root has been found.

The secant method works by setting initial bounds to the left and right of the root, between which a line secant to the curve is drawn. When the y-value derived from the two previous y-values hits threshold, the root has been found. And as with the case of the bisection method, the code enforces that the initial bounds must be on either side of the root.

In class, we used these methods to calculate $\sqrt{45}$, and I did so with my code to compare the results:

Method	No. Iterations	$\sqrt{45}$ Calculation
Bisection	35	6.70820393250
Newton	1	6.70820393249
Secant	3	6.68208092485
Calculator		6.70820390892

From these results, we can see that the bisection and Newton methods agree best with the calculator (Wolfram Alpha), and the Newton method took the fewest number of iterations.

Task 2

For the pseudo-isothermal sphere, using your root-finding algorithms, numerically calculate the full width at half maximum, i.e., what is the width (in terms of r_c) when $N_e(x) = N0/2$, half the amplitude.Drawing pictures for yourself might be useful! Do so with each of your root-finding algorithms and show how many iterations each takes as a function of your threshold. Please plot the result.

From the assignment, we start with an expression for the column density,

$$N_e(x) = \frac{N_0}{2} = N_0 \left[1 + (\frac{x}{r_c})^2\right]^{-\frac{1}{2}} \tag{1}$$

where N_e is the column density, N_0 is the central column density, and r_c is the critical radius.

To use the root finder algorithms, we can rearrange this expression to set it equal to 0

$$0 = \left[1 + (\frac{x}{r_c})^2\right]^{-\frac{1}{2}} - \frac{1}{2} \tag{2}$$

and solve for $\frac{x}{r_c}$ as the independent variable.

I used the three three root-finding codes written for Task 1 to find x in terms of r_c (by finding the root with $\frac{x}{r_c}$ being the independent variable and then rearranging to get get $x = ROOT \times r_c$.

I ran the three root finding codes many times over a large scale of different threshold inputs (ranging from 1×10^{-4} to 0.1, and Figure 1 shows the number of iterations it took for each

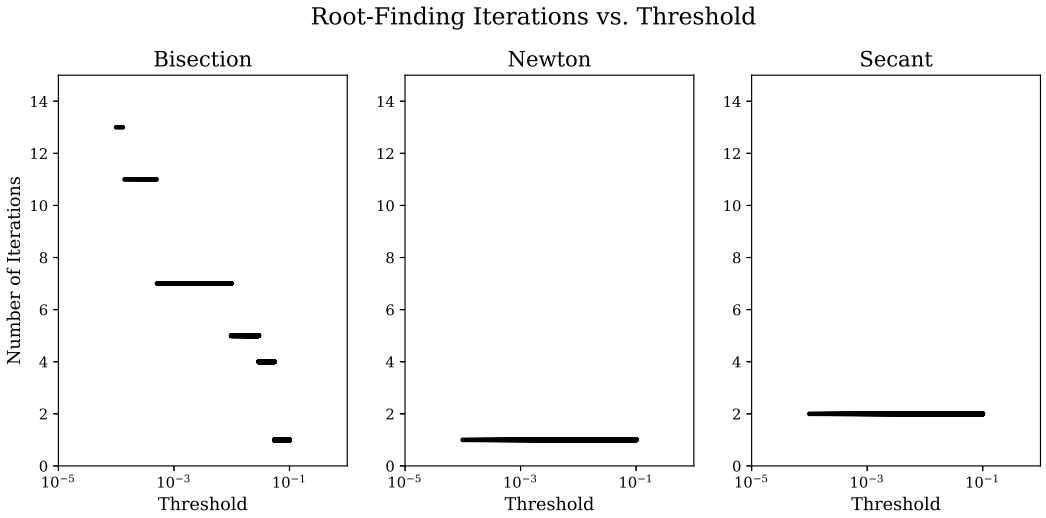

Figure 1: The three root-finding codes are compared by evaluating the number of iterations it takes to reach threshold for different threshold values. The left panel is for the bisection method, and it is clear that the number of iterations decreases as the threshold increases. The middle panel (Newton method) and right panel (Secant method) have a constant number of iterations required regardless of threshold.

code to reach threshold as a function of threshold. The converged upon root solution was that $x = \sqrt{3}r_c$ (this was the agreed value with the tighter thresholds).

Task 3

You can probably see from the Gaussian Lens equations that if you have a light ray hitting x, and you know the other parameters of the lens (a,N0,D, etc.), then you know what x_0 is. But that's boring and not what you actually observe. Let's instead say you are an observer in a "circular orbit" along the x' axis with radius 1 AU and a period of 1 year but centered at x'= 1AU. Then, you know where your position x' is but not where the light rays from the source are intersecting the lens plane at x- as expected,analytically solving for x is not really an option. Using one of your root-finding algorithms, solve the lens equation for each value of x_0 and make a ray-tracing plot as on the first page. Assume D= 1kpc, a= 1AU, λ= 21cm, and N_0 = 0.01pc cM^{-3}(these are observer units, probably best to convert to something like cm^{-2}).

For this task, the goal is to solve for the x-values, given x'-values. The expression for x' in terms of x and some given constants is

x [AU]	x' [AU]
0.0000	0.0000
0.2199	0.2222
0.4414	0.4444
0.6637	0.6667
0.8864	0.8889
1.1091	1.1111
1.3318	1.3333
1.5544	1.5556
1.7769	1.7778
1.9994	2.0000

Table 1: The calculated x values for the investigated x' values for the Gaussian lens case. Note that these values are given in units of astronomical units.

$$x' = x\left[1 + \frac{\lambda^2 r_e N_0 D}{\pi a^2} e^{-(x/a)^2}\right] \tag{3}$$

where[1] $\lambda = 21\ cm$, $r_e = 2.8 \times 10^{-16}\ cm$, $D = 3.086 \times 10^{21}\ cm$, $a = 1.496 \times 10^{13}\ cm$, and $N0 = 3.09 \times 10^{16}\ cm\ cm^{-3}$.

The x' values were calculated using the geometry setup of the problem where the observer is in a circular radius of Earth-like properties. These x' values can be generated across a grid of 0 AU to 2 AU evenly spaced (as the circle was centered at 1AU). The meaning of these values is that they are the projection of a circle (what you would see edge on) along the x' axis.

Then, the root finding equation (I chose to use the bisection method) can be used to solve the following equation for each x' value that was generated in the previous step using the equation below.

$$0 = x\left[1 + 0.016747 e^{-(\frac{x}{1.496\times 10^{13}})^2}\right] - x' \tag{4}$$

The results of x-values for the investigated x' values are tabulated below and a ray equation for the situation x' = 1AU is shown in Figure 2.

[1]These values were converted from the values provided by the homework to uniform units using Wolfram Alpha

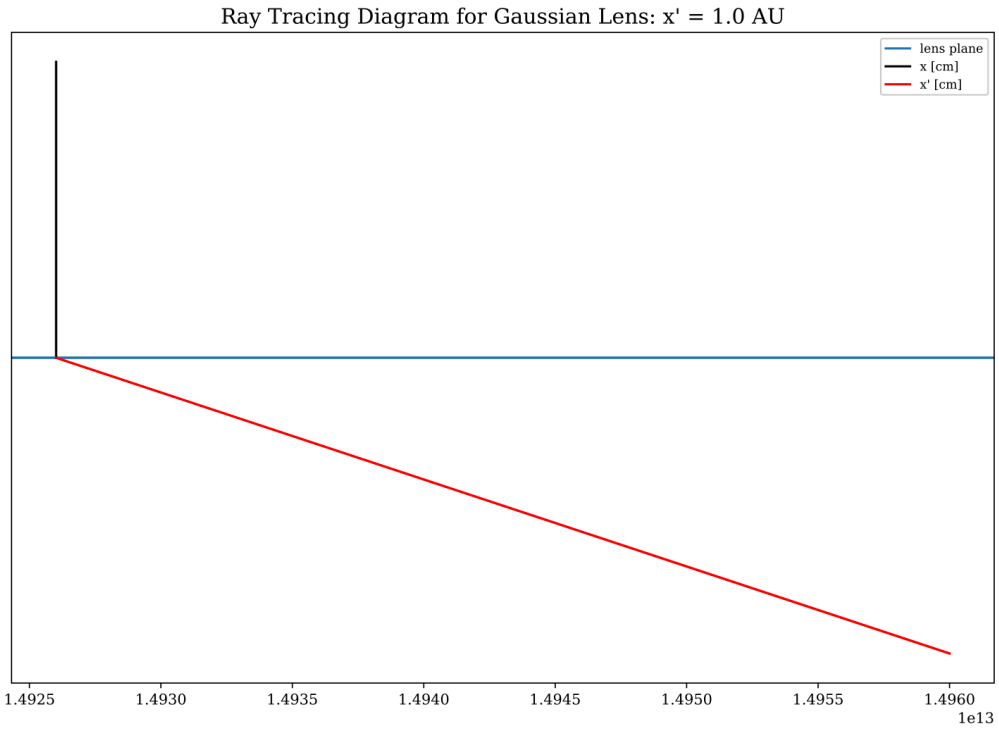

Figure 2: Ray Tracing for Gaussian lens where x' = 1AU. You can clearly see that x < x'.

Task 4

Repeat but for the pseudo-isothermal sphere with the same parameters but r_c = 1AU.

$$x' = x - \frac{\lambda^2 D r_e}{2\pi} \frac{d}{dx} N_e(x) \tag{5}$$

where $\frac{d}{dx} N_e(x) = -\frac{N_0 x}{r_c^2[1+(x/r_c)^2]^{3/2}}$.

Then, the root finding equation (I chose to use the bisection method) can be used to solve the following equation for each x' value that was generated in the previous step using the equation below.

$$0 = x \left[1 + \frac{\lambda^2 D r_e N_0}{2\pi r_c^2 [1 + (x/r_c)^2]^{3/2}} \right] - x' \tag{6}$$

which, with the constants put into place becomes,

$$0 = x\left[1 + 0.008235[1 + (x/1.496 \times 10^{13})^2]^{-3/2}\right] - x' \tag{7}$$

The results of x-values for the investigated x' values are tabulated below and a ray equation for the situation x' = 1AU is shown in Figure 3.

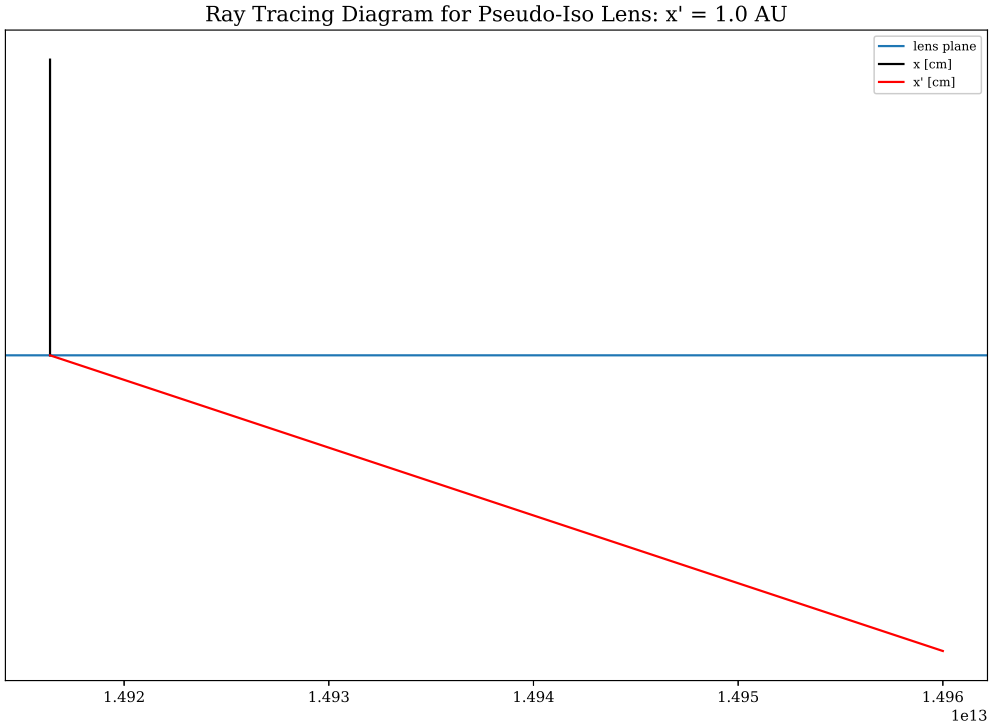

Figure 3: Ray Tracing for Pseudo-Isothermal lens where x' = 1AU. You can clearly see that x < x'.

x [AU]	x' [AU]
0.0000	0.0000
0.2205	0.2222
0.4417	0.4444
0.6635	0.6667
0.8858	0.8889
1.1084	1.1111
1.3310	1.3333
1.5535	1.5556
1.7761	1.7778
1.9985	2.0000

Table 2: The calculated x values for the investigated x' values for the pseudo-isothermal lens case. Note that these values are given in units of astronomical units.

Task 5

Write a library for piece-wise linear interpolation, given a set of x and y data points. This should return a function f that one can use to calculate a new point f(x_{new}) $\longrightarrow y_{new}$.

This library exists in a file called: `interpolator.py`. The function accepts x-values and y-values for the existing data point, and then internally calls another function, `apply` which performs the linear interpolation (finding the midpoint on line drawn between the two existing bounding points) on a new x-point of which interpolation is desired. The main function returns a function that calls this sub function. So the output of `interpolate` being called on x- and y-points is a function than can be called on x points to find the new corresponding y-values.

Task 6

In the file ***`lens_density.txt`*** *are a series of values of x and $N_e(x)$ for some shape. Use your interpolator to plot the values of $N_e(x)$ halfway in between all of the given x values, i.e., when x=0.5,1.5,...*

The raw data file is visualized below, in the left panel of Figure XX. To interpolate this data, I used the interpolation function I wrote to return a functional value for x-values located halfway between the existing integer values. In the right panel of Figure XX, the interpolated points are shown on the same plot as the left panel.

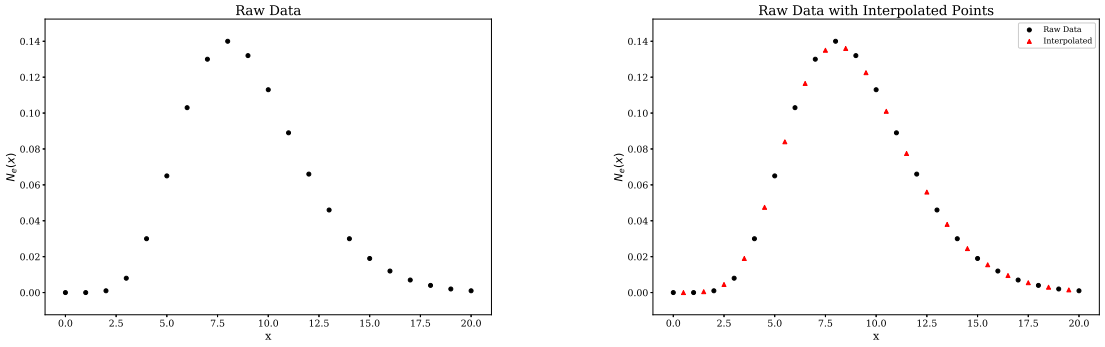

Figure 4: Left panel: original data points. Right panel: original data points (black circles) and interpolated points in between each data point (red triangles).