Name: Amanda Di Nitto
ID: 260689696
Homework 2
PHYS-512
Prof. Jon Seivers
Due: September 25th 2020

Problem 1:

In order to cut down on the amount of calls for F(x), the end points needed to be stored in the code so that the recursion function did not need to recalculate them each time it ran, which adds a lot of time to the runtime if the range of x is large. This was done by adding in the F(x) boundary terms right into the definition for the integral. By including y1 and y2 the area functions reduced the amount of function calls for F(x) by a total of 4 for a single run. This obviously gets bigger the more times the recursion is passed since it saves 4 in a single run so if the recursion passes 100 times that's 400 calls reduced. Therefor in the case for the Lorentz function between -1 and 1, the integration passes 7 times meaning 28 calls are saved. Comparing the errors between the original recursion to the new one displays the exact same errors, which makes sense as the integration should not change, only the efficiency of the routine. The routine was performed for both a Lorentz function and a cubic function in order to better check the routine works the same no matter the function called.

Problem 2:

In order to do a Chebyshev fit for a function other than sin(x) for [-1, 1], a new variable has to be introduced (known as the Chebyshev nodes[1]. The variable *u* is used to convert a function over the range [a,b] to a range of [-1,1]. The reason for this is because Chebyshev polynomials are most stable in this range, therefore it is best to adjust the function to work in that range. Being that our function is a log base 2, -1 and 1 cannot just be substituted straight as the function does not exist there. The parameter is given by;

$$u = \frac{2x - a - b}{b - a}$$

By replacing all *x* values with *u* the function can now be defined over [-1,1]. Other than this adjustment, the rest of the Chebyshev code remains the same; the polynomial recurrence relationship is the same since the Chebyshev polynomial do not need to be changed. The fit takes around 9-10 terms to reach at least an accuracy of 10^{-6} in the coefficient values for an order of 50.For the errors, the polynomial fit had both the bigger max error and rms error, but what was interesting was the fact that the rms error was only noticeably bigger at the last number; the rms for Chebyshev was 0.4438422404366025 but the rms for the polynomial fit was 0.4438422404366026. For the max error there was a more noticeable difference with the Chebyshev having an error of 1.5176196831870419 and the polynomial fit was

[1] As described in Chebyshev expansion (https://archive.siam.org/books/ot99/OT99SampleChapter.pdf)

1.7708233763202568. Based on the graph [Figure 1], it does not seem like the rms should be that similar for the two as they are clearly very different in their oscillations but the most probable reason is that the polynomial fit has much smaller oscillations and generally around the same size whereas the Chebyshev has some very large and other very small oscillations so that may have balanced the rms out.

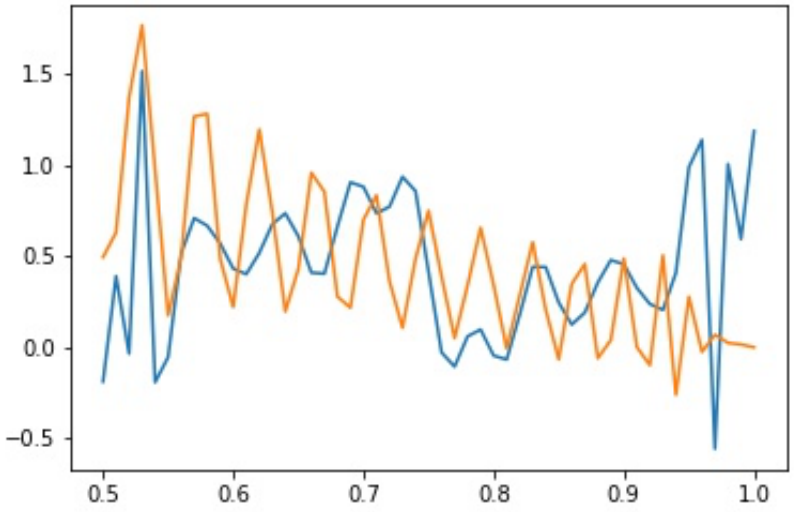


Figure 1: Chebyshev residuals (blue) vs the polynomial Legendre fit residuals (orange).

Problem 3A:

In order to do a full U238 decay, the program will need to run the decay for 14 products. On top of the large number of decay products the time for each decay is not the same, with the smallest being of a few microseconds. As seen in the code for a 2-state decay in class, the Rung Kuta 4 routine takes quite a bit more time than the implicit and almost 300 times the number of evaluations. Given how many more terms are required to complete the decay, the process would most likely take so long an output would not be produced for quite a bit of time. Based on these factors it makes more sense to run an implicit routine. The routine itself is very similar to the one in class. We begin again with only y values for the first term. The difficulty was in estimating the values for the half-life times. Due to machine precision, if the half-life value for U238 was put in, the calculation did not work and essentially estimated that everything was converted perfectly (i.e. final value was 0.999). In order to bypass this but still have an accurate relationship for the times, 4.4 billion years was equated to 4 and the other values were then based on this; an example is that 24 days was converted to $2x10^{-11}$. This worked much better and actually received a final value that makes a bit more sense of 0.778. The graph is not as smooth of a curve as expected [Figure 2], but this appears to be due to the machine precision and the large difference in half-life times; When testing the same code but with half-life times slightly closer together a smoother more quadratic looking curve was produced.

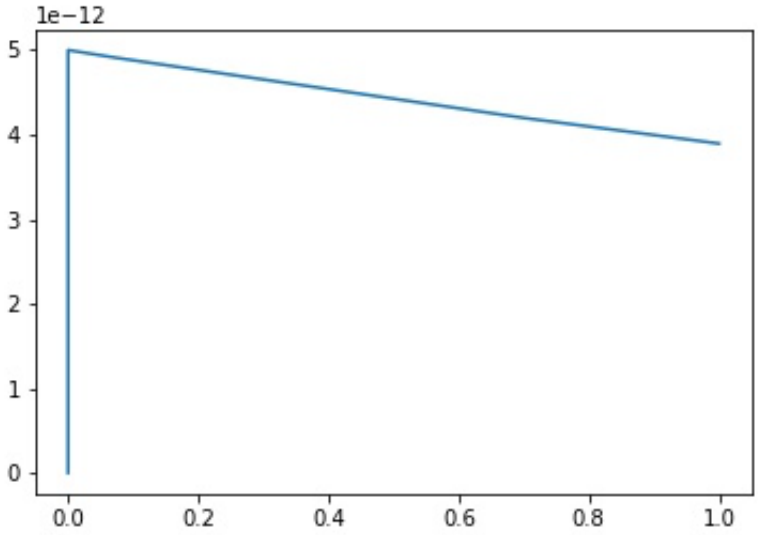
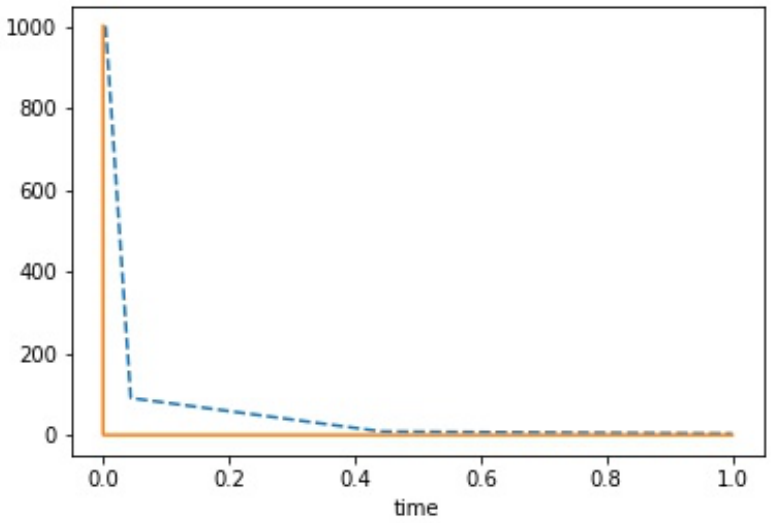


Figure 2: U238 decay for all decay products over time.

Problem 3B:

Unfortunately, due to the precision it was very hard to plot the U234 decay on the same graph as the U238 without it being a very harsh fast decline [Figure 3]. Obviously it is expected that U234 will decay much quicker than U238 but it is still expected to be a somewhat smooth curve whereas the data appears pretty sharp. For this U238 decay, it is assumed that it decays directly to lead, and just as with part A, the half-life was put as 4. The curve is again not as quadratic and smooth but it is displaying what is expected; initially there is a large amount of U238 which decays to lead but the curve smooths out as it gets closer to being mainly lead. If the two curves were plotted against eachother we expect them to have an exact opposite relationship in that as U238 decreases, Lead increase and that is what the ratio displays. For U234 to TH230 it is harder to see but the relationship is the same, only the rate at which the curve becomes entirely Th230 is much faster that of U238. Either way they display the same relationship just shifted in time, with one slower than the other.



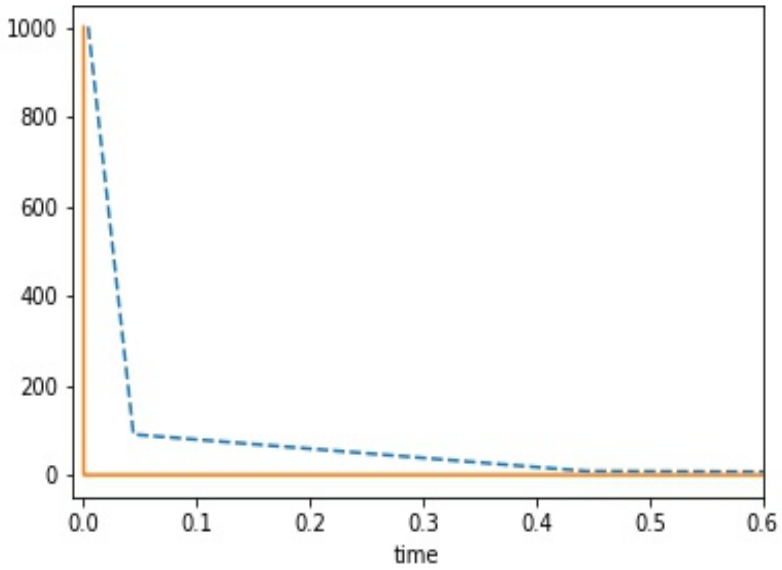


Figure 3: Full ratio vs time (upper), ratio vs time zoomed in for curve (lower)