

PHYS 512: Problem Set 3

Jade Ducharme (260929684)

October 2021

Problem 1

For the first part of this problem, I constructed an RK4 stepper similar to the one we saw in class during Lecture 5. The ODE used to test it is given by Equation 1:

$$\frac{dy}{dx} = \frac{y}{1+x^2} \quad (1)$$

We are given initial condition $y(-20) = 1$. Plugging all this information in *WolframAlpha*, I get that the solution to this ODE (with initial condition) is as shown in Equation 2.

$$y(x) = \exp(\arctan(x) + \arctan(-20)) \quad (2)$$

Figure 1 compares the ODE approximation I get using the RK4 stepper vs. the true solution given by Equation 2:

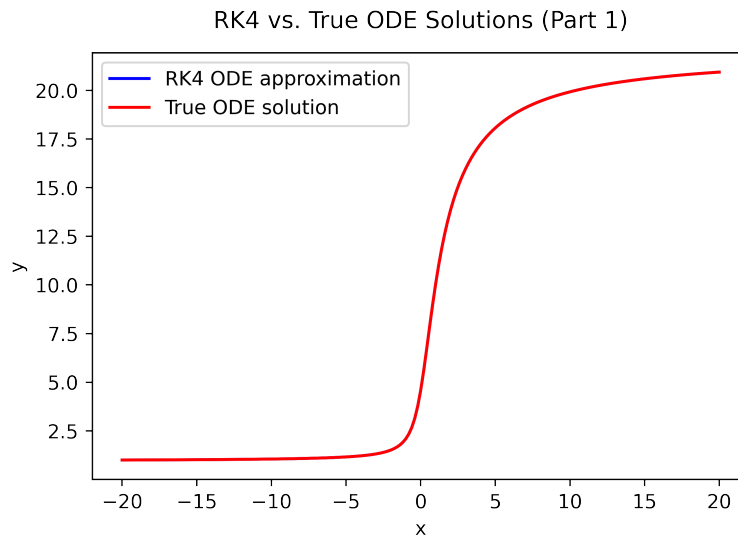


Figure 1: (Blue) approximate solution to the ODE presented in Equation 1 found using an RK4 stepper. (Red) true solution to the same ODE found using *WolframAlpha*. The two curves are indistinguishable to the eye.

Figure 2 shows the residuals for the two curves of Figure 1.

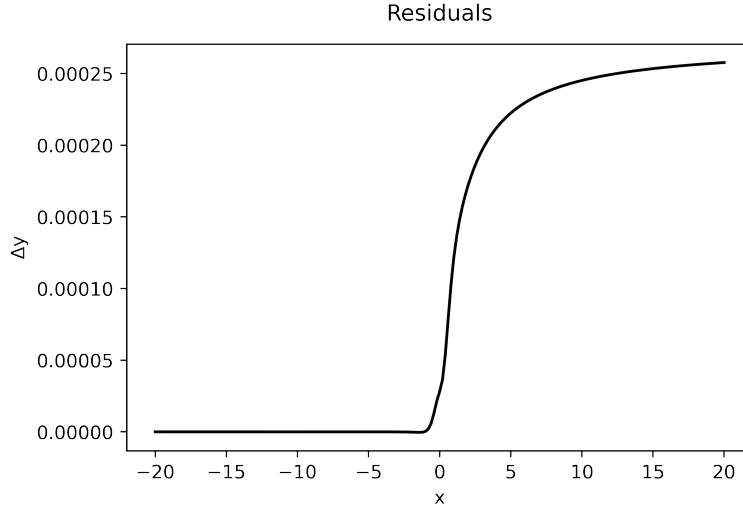


Figure 2: Residuals between the approximate ODE solution and the true ODE solution. The curve tells us the RK4 solution is more accurate for lower values of x .

For the second part of this problem, I need to figure out how to cancel out the leading-order error term from RK4. First, we know the error from a regular RK4 step with step length h is $O(h^5)$. Therefore, if we take an RK4 half-step with length $h_* = \frac{h}{2}$, the error should be $O(h_*^5) = O((\frac{h}{2})^5) = \frac{1}{32}O(h^5)$. But that only takes us from x_n to $x_{n+\frac{1}{2}}$. To get to x_{n+1} , we take an additional half-step, which will also have error $\frac{1}{32}O(h^5)$. So, the total error for both half steps is $2 \times \frac{1}{32}O(h^5) = \frac{1}{16}O(h^5)$.

In order to cancel out the $O(h^5)$ term completely, I need to think of a linear combination of 1 (corresponds to the full step with full error $O(h^5)$) and $\frac{1}{16}$ (corresponds to the two half-steps) that gives zero. The corresponding system of equations is:

$$\begin{cases} a + \frac{b}{16} = 0 \\ a + b = 1 \end{cases} \quad (3)$$

Where the second line comes from the fact that the combination must be normalized, otherwise the solution will blow up. Solving this yields $a = -\frac{1}{15}$ and $b = \frac{16}{15}$. So, if the RK4 approximation from one big step is y_1 , and the approximation from two half-steps is y_2 , the following linear combination cancels out the leading-order error term:

$$y_{n+1} = -\frac{1}{15}y_1 + \frac{16}{15}y_2 \quad (4)$$

Figure 3 shows a plot showing the true vs. approximated ODE curve using Equation 4.

Next, Figure 4 shows the residuals between the true solution and the approximation from Equation 4.

To the eye, there doesn't seem to be a big difference between the approximation using just one full step and the approximation combining two half-steps. In fact, I computed the percent difference between the mean of the residuals for both methods and found that they only differ by about $0.00297 = 0.297\%$. That would imply that using two half-steps instead of one full-step is only about 0.3% more accurate. So much

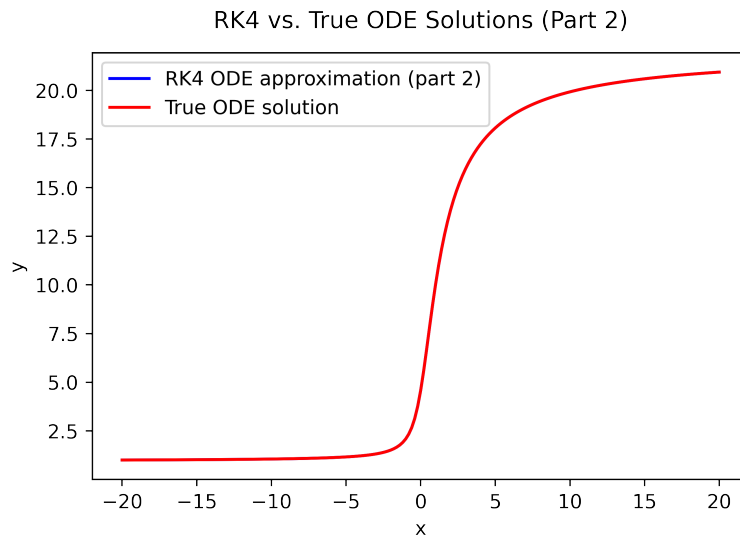


Figure 3: (Blue) approximate solution to the ODE presented in Equation 1 found using Equation 4. (Red) true solution to the same ODE. The two curves are indistinguishable to the eye.

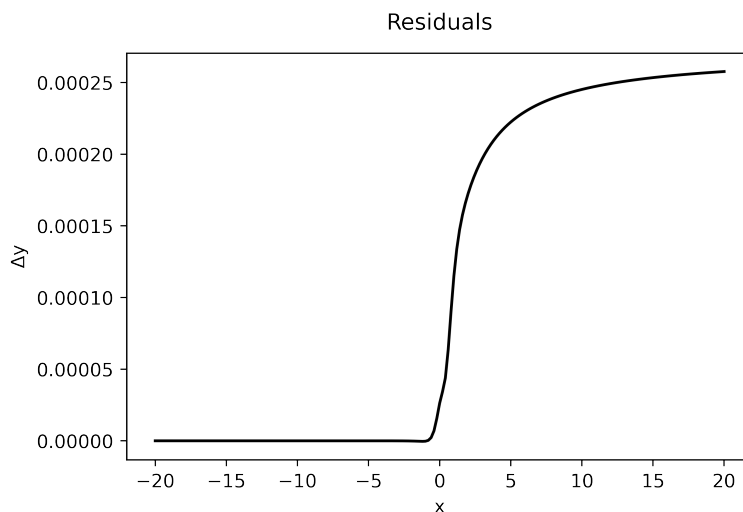


Figure 4: Residuals between the approximate ODE solution and the true ODE solution for Part 2.

work for so little results!

Problem 2

(a) First, I grabbed all the elements of the decay chain as well as their corresponding half-lives and ordered them in a tsv file to keep track of which half-life corresponds to which element. Then, I tweaked the code we saw during Lecture 5 in order to account for every element in the chain. I used `scipy.interpolate.solve_ivp` and specified that I wanted to use the “Radau” method (for implicit integration). That’s because I have so many elements in the decay chain and their decay rates vary so wildly that it would take way too long using *RK4* or other explicit methods.

In Figure 5, I present a few plots.

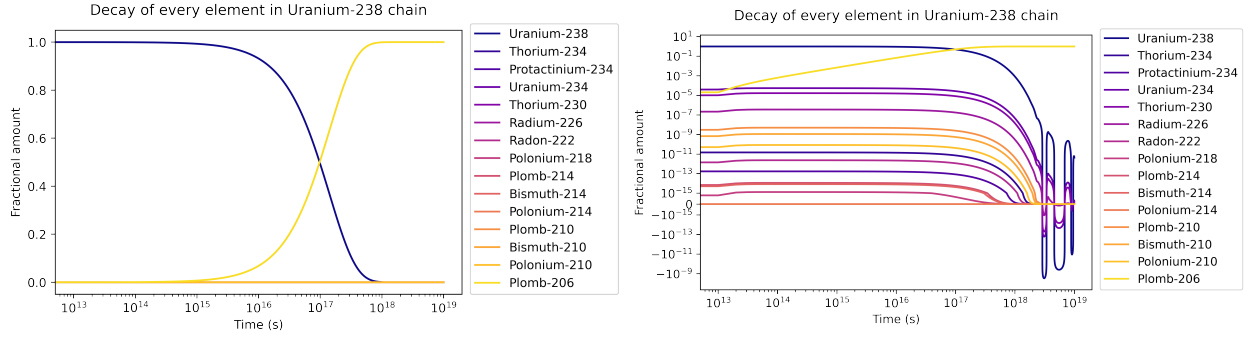


Figure 5: Rate of change for every element in Uranium-238 decay chain. (Left) The only visible elements are Uranium-238 and Plomb-206. The decay rates of intermediate elements are so short compared to U-238 that they appear to decay instantly and don't show in the plot. (Right) In order to observe the rate of change of the intermediate elements, I place the y-axis on a log scale (technically, a *symlog* scale so that zeros and negative values are treated in a more aesthetically pleasing manner). There are a few negatives near the end of the chain, but I'm pretty sure those are just due to round-off error.

(b) That awful yellow makes it hard to really compare Uranium-238 and Plomb-206, so I present a similar plot with less aggressive colours in Figure 6.

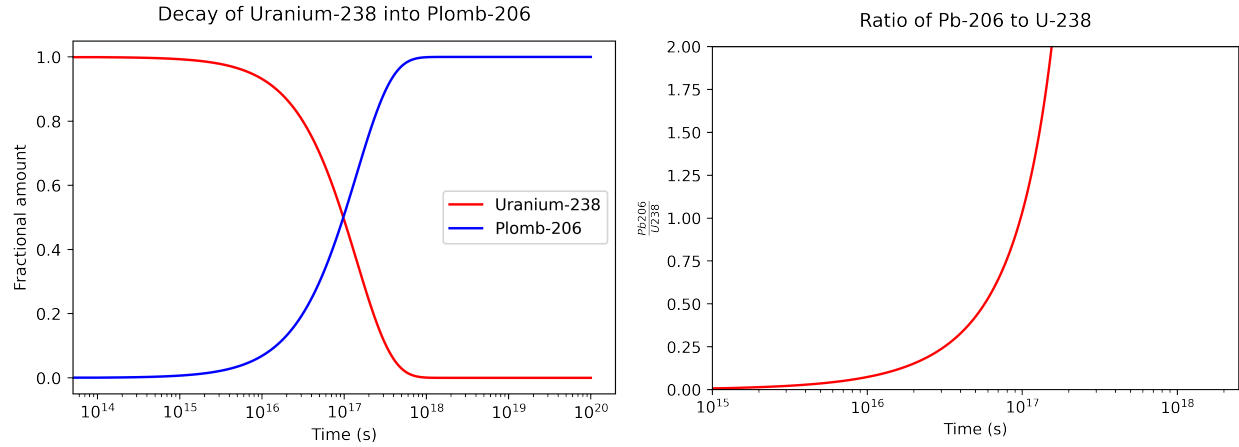


Figure 6: (Left) Decay path for U-238 and Pb-206 (neglecting intermediate elements). (Right) Ratio of Pb-206 to U-238 ($\frac{Pb_{206}}{U_{238}}$).

Analytically, the plots of Figure 6 makes perfect sense: I start off with 100% U-238 and end up with 100% Pb-206. Additionally, the half-life of U-238 is $\sim 10^{17}$ s, which corresponds precisely to the spot where the U-238 curve reaches half-height and the ratio reaches 1. The ratio plot also blows up as the U-238 curve gets closer and closer to zero – also makes sense because U-238 is the denominator. Everything seems to be in order!

Next, I look at the curves for Thorium-230 and Uranium-234 and present them in Figure 7.

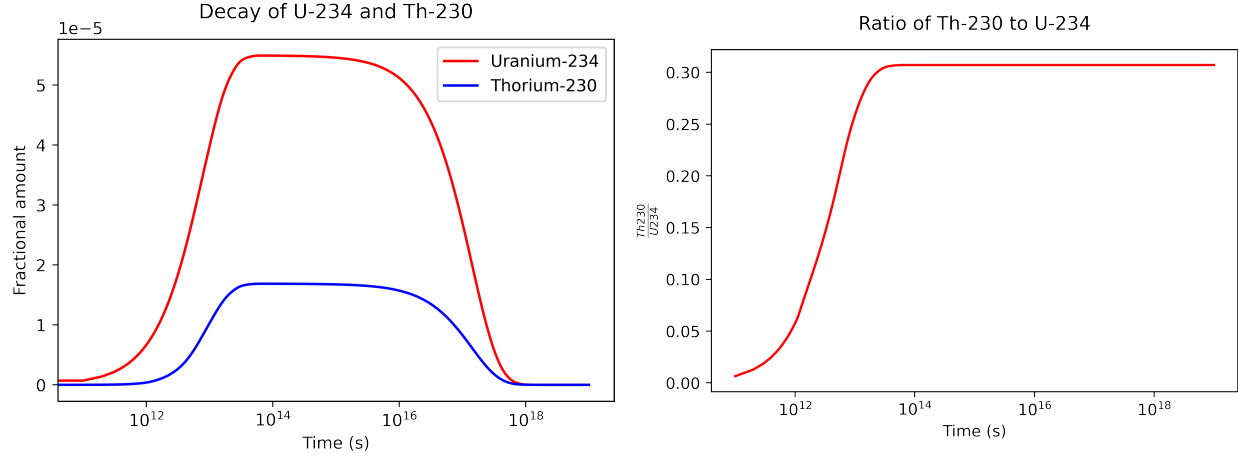


Figure 7: (Left) Decay paths for U-234 and Th-230. (Right) Ratio of Th-230 to U-234 ($\frac{Th^{230}}{U^{234}}$).

We see that the ratio of Th-230 to U-234 reaches equilibrium around the $t = 10^{14}$ s mark, which corresponds to ~ 3.2 million years. So, as long as the sample we're interested in dating isn't older than that, we can use the difference between the ratio of Th-230 to U-234 when the sample was formed and its ratio today to estimate its age.

Problem 3

(a) We have the equation for a symmetric paraboloid:

$$\begin{aligned}
 z - z_0 &= a((x - x_0)^2 + (y - y_0)^2) \\
 z &= z_0 + a(x^2 - 2x_0x + x_0^2) + a(y^2 - 2y_0y + y_0^2) \\
 z &= z_0 + ax^2 - 2ax_0x + ax_0^2 + ay^2 - 2ay_0y + ay_0^2 \\
 z &= (z_0 + ax_0^2 + ay_0^2) + (-2ax_0)x + (-2ay_0)y + a(x^2 + y^2)
 \end{aligned} \tag{5}$$

Therefore, if we define $u \equiv z_0 + ax_0^2 + ay_0^2$, $v \equiv -2ax_0$, and $w \equiv -2ay_0$, the new, *linearized* equation looks like this:

$$z = u + vx + wy + a(x^2 + y^2) \tag{6}$$

We can then obtain an expression for our \mathbf{A} and \mathbf{m} matrices:

$$\mathbf{A} = \begin{pmatrix} 1 & x_1 & y_1 & x_1^2 + y_1^2 \\ 1 & x_2 & y_2 & x_2^2 + y_2^2 \\ 1 & x_3 & y_3 & x_3^2 + y_3^2 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{7}$$

$$\mathbf{m} = \begin{pmatrix} u \\ v \\ w \\ a \end{pmatrix} \quad (8)$$

I know then that the best fit line for my data is given by $z_{fit} = \mathbf{A}\mathbf{m}$. In order to actually *find* the \mathbf{m} matrix, I will use the `numpy.linalg.lstsq` function.

(b) Figure 8 presents plots of z vs. x and y for the raw data as well as the line constructed using $z_{fit} = \mathbf{A}\mathbf{m}$.

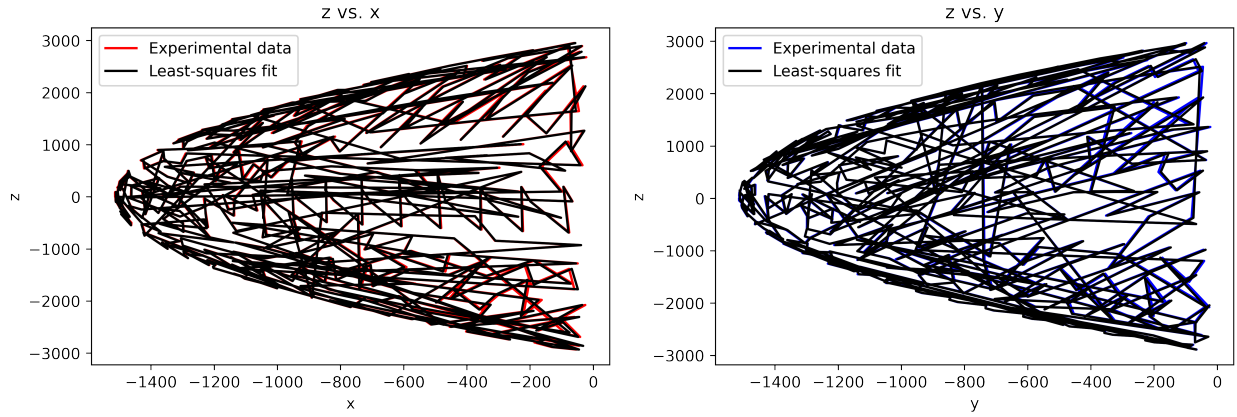


Figure 8: (Left) z vs. x and (right) z vs. y for the raw data (red and blue curves) and the least-squares fit (black curves). To the eye, they look pretty good!

The best fit parameters are:

$$\begin{aligned} u &= -1.51 \times 10^3 \\ v &= 4.54 \times 10^{-4} \\ w &= -1.94 \times 10^{-2} \\ a &= 1.67 \times 10^{-4} \end{aligned} \quad (9)$$

From these, I can easily recover the true parameters for the symmetric parabola:

$$\begin{aligned} a &= 1.67 \times 10^{-4} \\ x_0 &= -1.36 \\ y_0 &= 58.22 \\ z_0 &= -1512.88 \end{aligned} \quad (10)$$