#1.

The function `rk4_step(fun,x,y,h)` is pretty much the same one as used in lecture: it simply calculates k_1, k_2, k_3 and k_4 and returns $y + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$. Now, if we want to solve $\frac{dy}{dx} = \frac{y}{1+x^2}$ in the interval $[-20, 20]$ with $y(-20) = 1$, we can separate our interval in 200 steps and compute each y_i using `rk4_step(fun,x`$_{i-1}$`,y`$_{i-1}$`,h)`. Comparing our result with the analytical solution the the IVP, which is $y = \exp(\arctan(20) + \arctan(x))$, we get an error at most on the order of 10^{-4} (Figure 1).

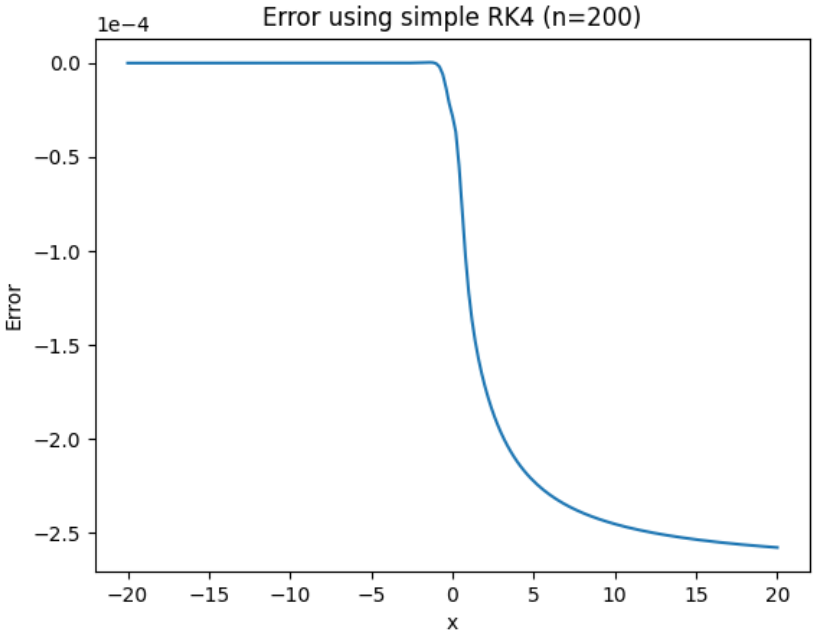

Figure 1: Error on the solution when using the traditional RK4 method with 200 steps

Now, let's consider approximating $y(x+h)$ in two ways: first by taking a single step of size h, and second, by taking two steps of size $h/2$. Since we're using RK4, our error will be of 5^{th} order. So, in our first case, our approximation will be:

$$y_1 = y(x+h) + \frac{h^5}{5!}y^{(5)} + O(h^6)$$

In our second case, the error on each half-step combine and we end up with:

$$y_2 = y(x+h) + 2 \cdot \frac{h}{2^5 \cdot 5!}y^{(5)} + O(h^6) = y(x+h) + \frac{h}{16 \cdot 5!}y^{(5)} + O(h^6)$$

If we combine these results to get $y_3 = (16y_2 - y_1)/15$, we end up cancelling the fifth order terms and are left with:

$$y_3 = y(x+h) + O(h^6)$$

Since for each step we called `rk4_step`, which itself does 4 function calls (one for each k), our new version of RK4 will have 12 function calls per step instead of the original 4 per step. So, if we were to try to solve the same IVP as above while using the same number of function calls, we ought to divide the number of total steps by 3. So, now solving using only 67 steps, we have an error that is also on the order of 10^{-4} (Figure 2). However, comparing the largest error between each run, we see that this new version's is smaller by about a factor of 3, which is still good considering the difference in number of steps! So, we can conclude that in general this version is more accurate.

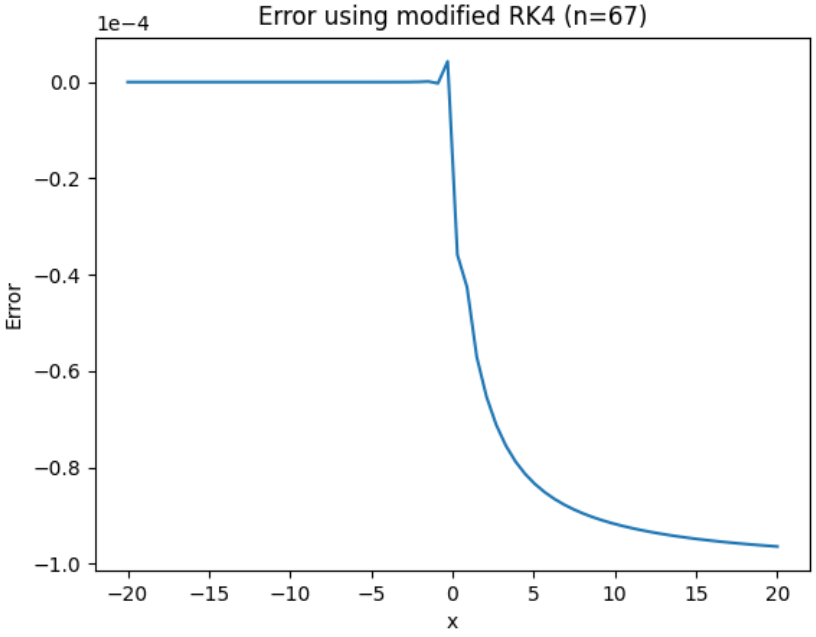

Figure 2: Error on the solution when using the modified RK4 method with 67 steps

#2.

First, let's set-up the system of ODE that describes this decay chain. If we have k products with relative abundance N_i, the amount each product loses in a single time interval dt is proportional to the amount of this particular product. Also, they will gain an amount proportional to the amount of the product that came before it in the chain since the latter decay into the former. However, the very first element in the chain doesn't gain anything since we assume that nothing decays into it during the process and also, the last element in the chain is assumed to be infinitely stable, so it doesn't decay into anything. This thus gives us this system of ODEs:

$$\begin{cases} \dfrac{dN_1}{dt} &= & -\dfrac{1}{\tau_1}N_1 \\ \dfrac{dN_i}{dt} &= -\dfrac{1}{\tau_i}N_i + \dfrac{1}{\tau_{i-1}}N_{i-1} & (1 < i < k) \\ \dfrac{dN_k}{dt} &= & \dfrac{1}{\tau_{k-1}}N_{k-1} \end{cases}$$

With the initial condition that $N_1 = 1$ and $N_i = 0$ for $i > 1$.

We are given the half life of each product, so $\tau_i = t_{1/2}/\ln(2)$. However, the time scales can vary from micro seconds to billions f years, which makes this system of ODEs stiff. So, we ought to use `scipy.integrat.solve_ivp` with the argument `method="Radau"` which handles these kinds of equations.

By noticing that the half life of U238 is much much larger than any of its intermediary products, we can approximate this scenario by assuming U238 decays directly into Pb206, which reduces our system of ODE to:

$$\frac{dN_U}{dt} = -\frac{1}{\tau_U}N_U$$
$$\frac{dN_{Pb}}{dt} = \frac{1}{\tau_U}N_U$$

This is easily solved by

$$N_U(t) = e^{-t/\tau_U}$$
$$N_{Pb}(t) = 1 - e^{-t/\tau_U}$$

Hence, their ratio is:

$$\frac{N_{Pb}}{N_U} = e^{-t/\tau_U} - 1$$

Comparing this with this ratio obtained using the numerical solution, we see that both are in agreement (Figure 3).

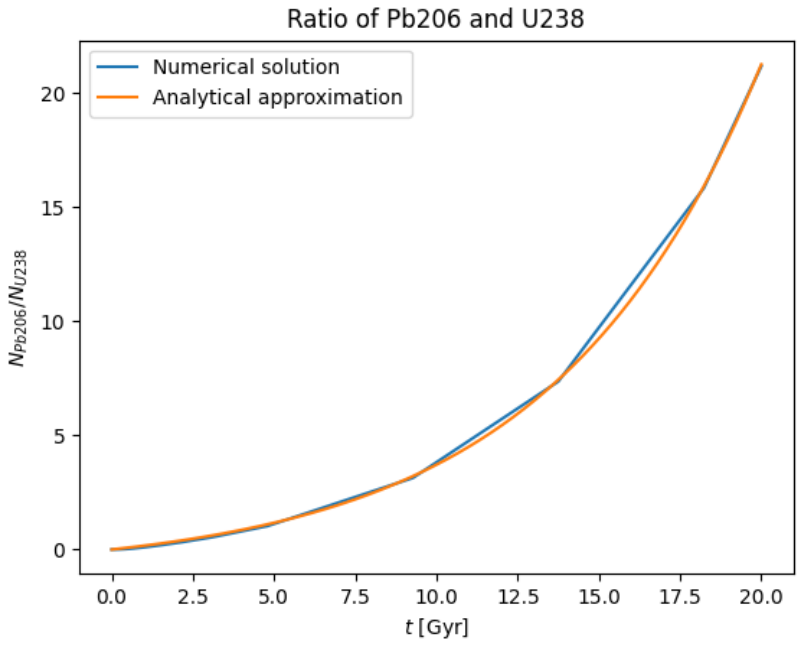

Figure 3: Comparison between numerical solution and analytical approximation of N_{Pb}/N_U

We can also plot the ratio of Thorium 230 to Uranium 234 using our numerical solution. Interestingly, it converges to the ratio of their half-life, which makes sense (Figure 4)!

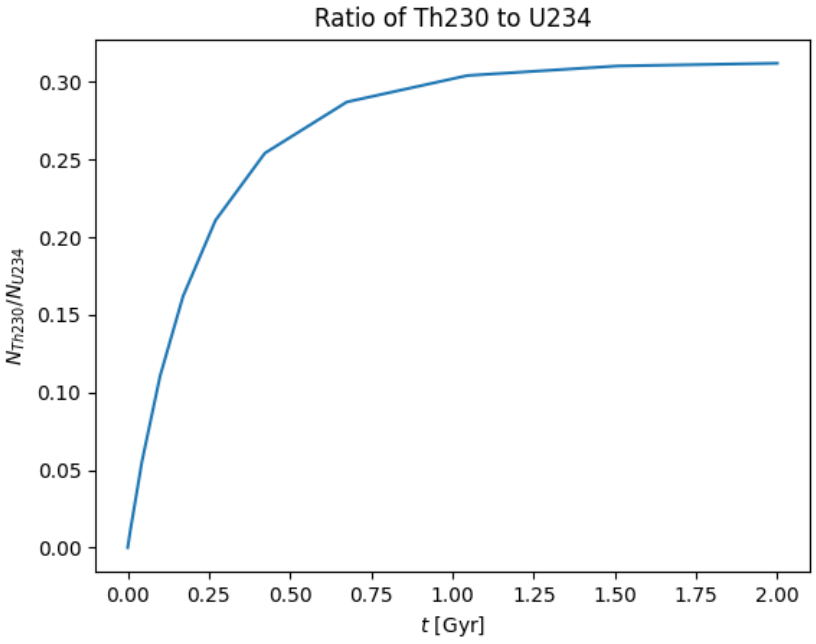

Figure 4: Numerical solution for N_{Th}/N_U

#3.

Let's expand and re-arrange our equation:

$$z = z_0 + a(x^2 + y^2 - 2xx_0 - 2yy_0 + x_0^2 + y_0^2)$$
$$= (x^2 + y^2) \cdot a - 2x \cdot (ax_0) - 2y \cdot (ay_0) + (ax_0^2 + ay_0^2 + z_0)$$

If we define the following new parameters $u \equiv ax_0$, $v \equiv ay_0$ and $w \equiv ax_0^2 + ay_0^2 + z_0$, our equation for z becomes linear. So, we can do a linear fit using

$$Am = \begin{pmatrix} x_1^2 + y_1^2 & -2x_1 & -2y_1 & 1 \\ x_2^2 + y_2^2 & -2x_2 & -2y_2 & 1 \\ x_3^2 + y_3^2 & -2x_3 & -2y_3 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} a \\ u \\ v \\ w \end{pmatrix}$$

Doing so, we get values for a, u, v and w and it is straightforward to get the values for our original parameters: `a= 0.00016670445477401347 , x0= -1.3604886221976673 , y0= 58.22147608157938 , z0= -1512.8772100367873`.

To estimate the noise N, we compare our data point z_d with the values found using our model z_m by averaging $(z_d - z_m)^2$, which gives us $N = 14.2$
We can use this value to estimate the covariance matrix for our parameters:

$$C = (A^T N^{-1} A)^{-1} = N(A^T A)^{-1}$$

Then, the error on our fit parameters is the square root of the diagonal elements of C. Hence, the error on a is

$$\Delta a = 6.45 \times 10^{-8} \text{ mm}^{-1}$$

Finally, the focal length can be found using $f = \frac{1}{4a}$ and its error is $\Delta f = \left| \frac{df}{da} \cdot \Delta a \right| = \frac{\Delta a}{4a^2}$.
Computing this gives us:

$$f = (1.4997 \pm 0.0006)\text{m}$$

This value agrees with the expected value of 1.5m