Phys 512 PS3

Andrew Lewis

September 30, 2022

Problem 1:

Write an RK4 Integrator
You will find the code for r4k_step(fun,x,y,h) in the file PS3Q1.py

Integrating from x = -20 to 20 with 200 steps
See the plot of the integration as well as the residuals below:

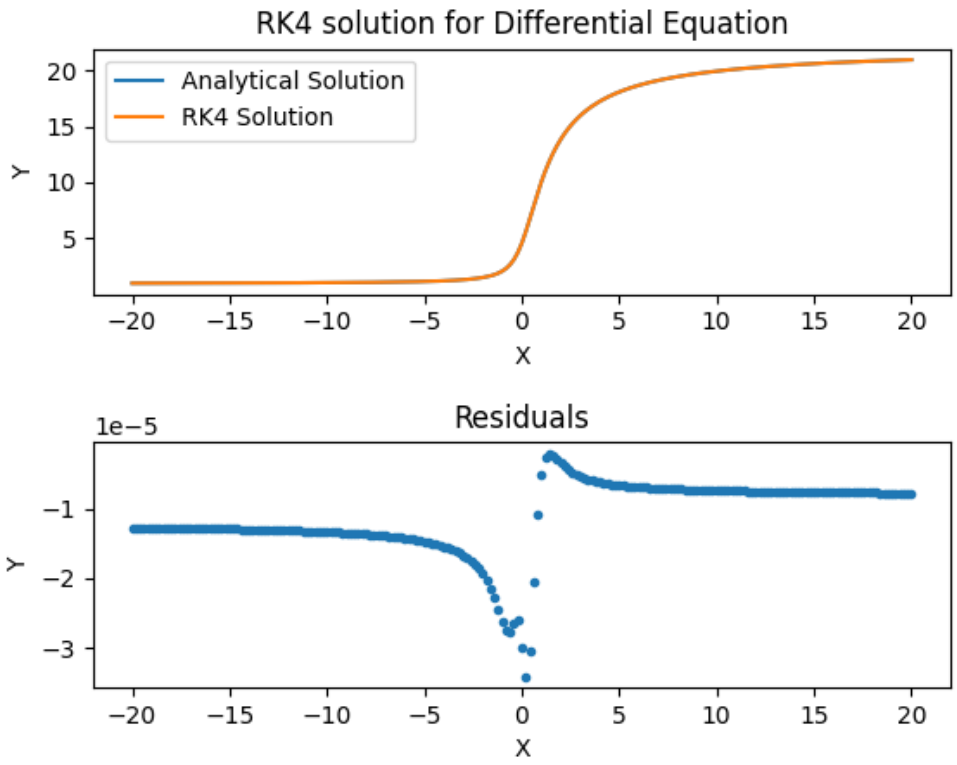

Figure 1: Normal runge kutta stepper compared with analytical solution. Residuals are on an order of 1e-5

Write another stepper

You will find the code for rk4_stepd(fun,x,y,h) in PS3Q1.py

Derivation of method
Lets consider the two different step sizes independently, where I have written out the 5th term order explicitly.
One step h

$$y(x+2h) = y1 + 2h^5 * \frac{y^{5(x)}}{5!} + O(6)$$

two steps

$$y(x+2h) = y2 + (2h)^5 * \frac{y^{5(x)}}{5!} + O(6)$$

Since we know runge kutta methods are accurate to the 4th order we can just write the solutions to those methods as y1 and y2. Rewriting our two step method

$$y(x+2h) = y2 + (32h^5)\frac{y^{5(x)}}{5!} + O(6)$$

subtracting 16 of the first methods from the second one we get

$$-15y(x+2h) = y2 - 16y1 + O(6)$$

As you can see, we have cancelled out the 5th order error term in our equations. Therefore we can write y(x+2h) as:

$$y(x+2h) = \frac{y2 - 16y1}{-15}$$

How many function evaluations per step does this use?
By explicit inspection of the code, the original rk4 stepper will will use 4 function evaluations. You would expect the modified one using 2 steps and comparing them to one step to use:

$$4 + 2steps(4functs) = 12$$

12 function evaluations, however we can use the original k1 from our original function evaluation and just divide it by 2, therefore the new method will use 11 function evaluations per step.

Therefore to evaluate the same range of points we will need to use:

$$200 * 4 = 800$$

$$11 * 73 = 803$$

Around 73 steps

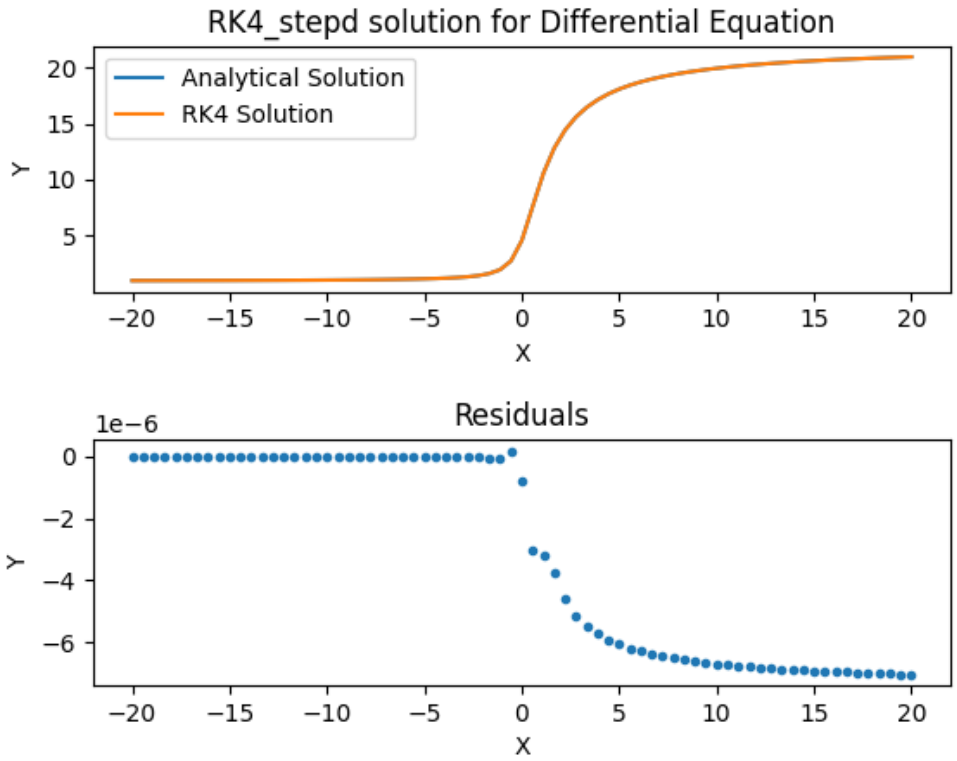

Figure 2: Normal runge kutta stepper compared with analytical solution. Residuals are on an order of 1e-6

Which is more accurate?
As can be seen in the difference between figure 1 and figure 2, the error is smaller in figure 2 therefore the second runge kutta stepper is more accurate as expected. In order to obtain the residuals I simply determined the value for the constant in the analytical solution using the initial condition

Problem 2:

a)

Write a program to solve for the decay products of U238
Please see code attached in PS3Q2.py
Which solver would you use for this problem

Since in half life calculations the rate at which products decay is often massive orders of magnitudes different it will be a stiff equation. Therefore we will use the radau method as seen in class.

b)

Plot the ratio of Pb206 to U238

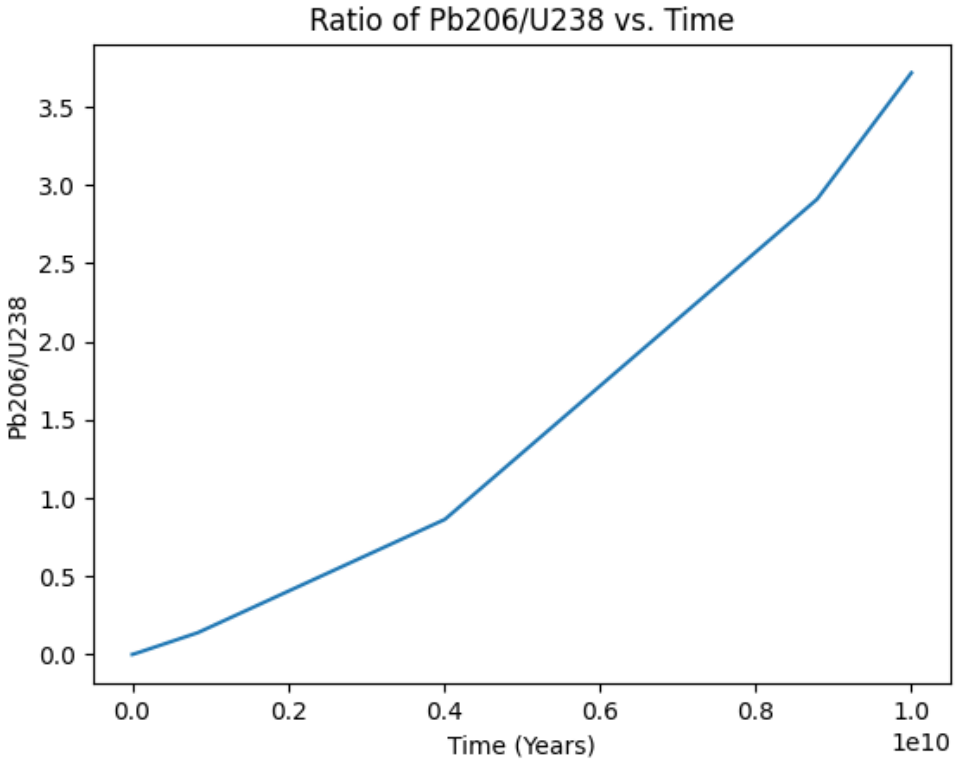

Figure 3: Ratio of Pb206 to U238 vs. Time

The reason the time interval was chosen was because there is a very apparent ratio pb206 to u238 in a region where there are no plateaus. I.e there will be no two places where the ratio of pb206 to u238 is the same. If we take the hint that u238 decays instantly into lead we can see if it makes sense. Since the half life is around 4.5 billion years we should expect the ratio of uranium to lead to be equal to one there. By inspection on the graph it is quite clear that the ratio is approximately one, therefore this graph makes sense analytically.

Plot the ratio of Thorium 230 to U234

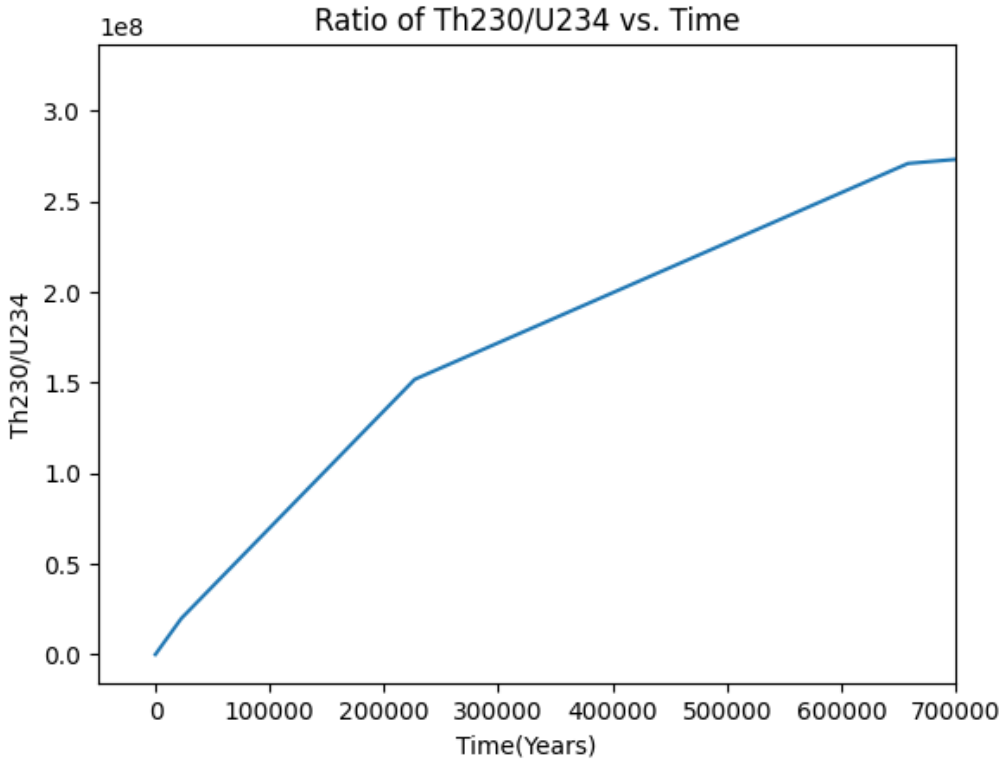

Figure 4: Ratio of Th230 to U234 in a time span of interest.

The reason this time interval was chosen was because at any further time scales the ratio of th230 to u234 plateaus. Therefore there would be no more meaningful information to gain from the ratio of th230 to u234. I.e. we would not be able to tell the difference between two different samples.

Problem 3:

a)

Show that you can pick a new set of parameters that make the problem linear
so first we are stuck with the equation

$$z - z0 = a((x - x0)^2 + (y - y0)^2)$$

But, if we expand the equation we get this:

$$z - z0 = a((x^2 - 2xx0 + x0^2 + y^2 - 2yy0 + y0^2)$$

$$z - z0 = ax^2 - 2axx0 + ax0^2 + ay^2 - 2yy0 + ay0^2$$

Moving the z0 over to the other side

$$z = ax^2 + ay^2 - 2axx0 - 2yy0 + ax0^2 + ay0^2 + z0$$

We can combine the constants into one variable d, and rewrite $-2ax0$ and $-2ay0$ as b and c and our equation will become

$$z = a(x^2 + y^2) + bx + cy + d$$

Which will be a combination of linear terms
How do they relate to the old parameters?
By inspection, the parameters will relate to the old parameters as:

$$a = a$$

$$b = -2ax0, x0- > \frac{b}{-2a}$$

$$c = -2ay- > y0 = \frac{c}{-2a}$$

$$d = ax0^2 + ay0^2 + z0- > z = 0 = d - ax0^2 - ay0^2$$

Since we have 4 equations and 4 unkowns we can solve for this system and retrieve our initial model parameters!.

b)

Carry out the fit. What are your best fit parameters?

The best fit parameters are:

$$a = 0.00016670445477401358(1/mm)$$

$$x0 = -1.360488622197728mm$$

$$y0 = 58.22147608157934mm$$

and

$$z0 = -1512.8772100367878mm$$

c)

Estimate the noise in the data, and from that estimate the uncertainty in a.
The noise in the data was given by subtracting the model from the data:

$$n = z - m$$

next a covariance matrix was determined by multiplying the noise together

$$n * nT = N$$

Then using the equation

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

and reading off the diagonal, we could determine that the error in the paramater a was

$$\pm 9.285e - 16/mm$$

Our target focal length was 1.5 meters. What did we actually get and what is the error bar?
In order to determine focal length we first need to look at our equation for our paraboloid.

$$z - z0 = a((x - x0)^2 + (y - y0)^2)$$

This looks very similar to:

$$y = \frac{x^2}{4f}$$

Except for a 2d parabola we generally write it as:

$$y = ax + b$$

Therefore we can ignore the constant in the z direction, z0 and just focus on the main part of the equation, where we can rewrite our term a in terms of focal length with the formula:

$$a = \frac{1}{4f} -> f = \frac{1}{4a} mm$$

Taking our estimate for a and plugging it into our formula we get:

$$f = \frac{1}{4 * 0.00016670445477401358} mm * \frac{m}{1000mm} = 1.4996599841252158m$$

using simple error propagation for division:

$$\frac{\sigma_x}{x} = \frac{\sigma_c}{c}$$

$$\sigma_x = x * \frac{\sigma_c}{c}$$

$$\sigma_x = 1.4996599841252158 * \frac{9.285e-16}{0.00016670445477401358} = 8.352712e-12$$

So our final estimation for the focal length is:

$$1.499659984125 \pm 8e-12m$$

for our focal length, which is very close (but not quite) our goal of a focal length of 1.5m

Bonus

So you aren't going to like it, it ain't pretty and it ain't elegant. I brute forced the rotations to get a good idea of what the maximum and minimum focal lengths are.
Using the equation

$$z = ax'^2 + by'^2$$

I aimed to find the maximum and minimum focal length for our dish by finding the maximum and minimum A in a pi rotation (as a 2pi rotation would be redundant). Setting

$$x = x'cos(\theta) + y'sin(\theta)$$

$$y = -x'sin(\theta) + y'cos(\theta)$$

I then formed a basis matrix A out of x'^2 and y'^2. Next but doing a 1000000 step rotation and finding the maximum and minimum of the as that were calculated by the linear least squares fit it was possible to determine that the maximum focal length was (one of our principal axis):

$$1.51m$$

and the minimum was (the other principal axis)

$$1.49m$$

Therefore the dish is not quite round, I would say it is roundish