PHYS-512, PS3

Muath Hamidi

September, 2022

Problem 1

The differential equation we have here is

$$\frac{dy}{dx} = \frac{y}{1+x^2} \tag{1}$$

This is a first-order linear ordinary differential equation, with solution

$$y = c_0 \exp\left(\arctan\left(x\right)\right) \tag{2}$$

Given the boundary condition $y(-20) = 1$, then $c_0 = 1/\exp\left(\arctan\left(-20\right)\right)$. So,

$$y = \frac{1}{\exp\left(\arctan\left(-20\right)\right)} \exp\left(\arctan\left(x\right)\right) \tag{3}$$

Use the following code:

```
#====================================================
# Course: PHYS 512
# Problem: PS3 P1
#====================================================
# By: Muath Hamidi
# Email: muath.hamidi@mail.mcgill.ca
# Department of Physics, McGill University
# September 2022

#====================================================
# Libraries
#====================================================
import numpy as np # For math
import matplotlib.pyplot as plt # For graphs

#====================================================
# Derivative Function
#====================================================
def fun(x,y):
    global counter
    counter +=1

    dydx = y / (1 + x**2)
    return dydx

#====================================================
# RK4 (Step)
#====================================================
def rk4_step(fun, x, y, h):
    k1 = h * fun(x, y)
```

```
    k2 = h * fun(x + h/2, y + k1/2)
    k3 = h * fun(x + h/2, y + k2/2)
    k4 = h * fun(x + h, y + k3)
    dy = (k1 + 2 * k2 + 2 * k3 + k4) / 6
    return y + dy

#====================================================
# Parameters & Definitions
#====================================================
steps = 201
x = np.linspace(-20, 20, steps)
h = np.median(np.diff(x))
y = np.zeros(steps)
y[0] = 1 # boundary condition

#====================================================
# Numerical Solution
#====================================================
counter = 0
for i in range(steps-1):
    y[i+1] = rk4_step(fun, x[i], y[i], h)

#====================================================
# Analytical Solution
#====================================================
c0 = 1 / np.exp(np.arctan(-20)) # true solution amplitude
y_true = c0 * np.exp(np.arctan(x)) # true solution

#====================================================
# Plot
#====================================================
plt.plot(x, y, linewidth=5, label="Numerical RK4")
plt.plot(x, y_true, linewidth=5, ls=":", label="Analytical")
plt.title("Given Function Integration, step")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.savefig('3.1.1.pdf', format='pdf', dpi=1200)
plt.show()
plt.close()

#====================================================
# Error & Function Evaluations
#====================================================
print("In rk4_step, the difference mean is", np.mean(abs(y_true-y)), "with",
   counter,"function evaluations.")

#====================================================
# RK4 (Stepd)
#====================================================
def rk4_stepd(fun, x, y, h):
    # Step size h
    y1 = rk4_step(fun, x, y, h)

    # Step size h/2
    y2i = rk4_step(fun, x, y, h/2)
```

```python
    y2 = rk4_step(fun, x + h/2, y2i, h/2)

    return y2 + (y2 - y1) / 15

#==================================================
# Parameters & Definitions
#==================================================
y = np.zeros(steps)
y[0] = 1 # boundary condition

#==================================================
# Numerical Solution
#==================================================
counter = 0
for i in range(steps-1):
    y[i+1] = rk4_stepd(fun, x[i], y[i], h)

#==================================================
# Plot
#==================================================
plt.plot(x, y, linewidth=5, label="Numerical RK4")
plt.plot(x, y_true, linewidth=5, ls=":", label="Analytical")
plt.title("Given Function Integration, stepd")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.savefig('3.1.2.pdf', format='pdf', dpi=1200)
plt.show()
plt.close()

#==================================================
# Error & Function Evaluations
#==================================================
print("In rk4_stepd, the difference mean is", np.mean(abs(y_true-y)), "with",
    counter,"function evaluations.")
```

Where we have the errors and functions evaluations as the following:
In rk4_step, the difference mean is 0.00011382485934170593 with 800 function evaluations.
In rk4_stepd, the difference mean is 2.0613363261088876e-07 with 2400 function evaluations.

As you see, the rk4_stepd is more accurate than rk4_step, but it has 3 times more evaluations.

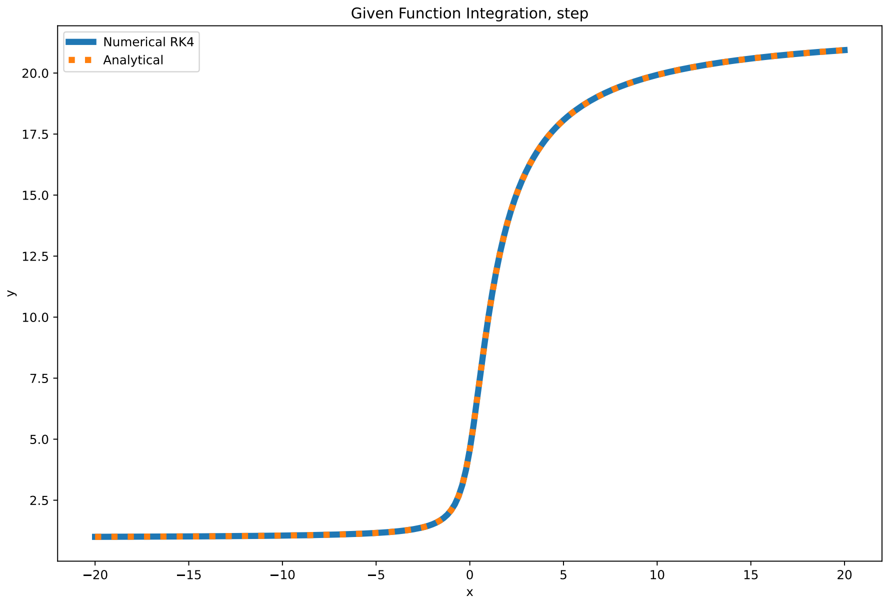

Figure 1: Given Function Integration, rk4_step.

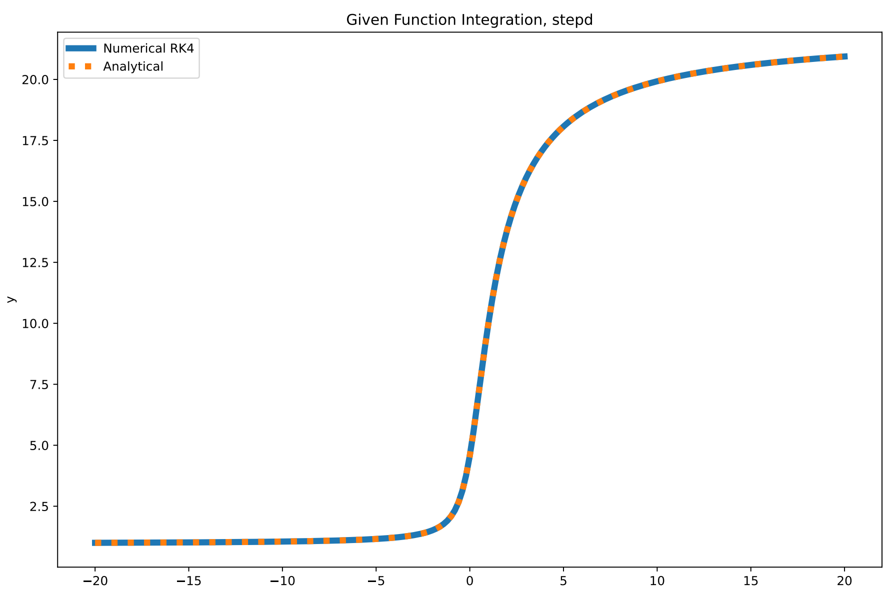

Figure 2: Given Function Integration, rk4_stepd.

Problem 2

a.

In this problem, we will see the number of atoms or their share of the complete sample. This share represented by a number from 0 to 1. We will use ”scipy.integrate.solve_ivp” function.

Now, use the code:

```python
# Course: PHYS 512
# Problem: PS3 P2
#=====================================================
# By: Muath Hamidi
# Email: muath.hamidi@mail.mcgill.ca
# Department of Physics, McGill University
# September 2022

#=====================================================
# Libraries
#=====================================================
import numpy as np # For math
import matplotlib.pyplot as plt # For graphs
from scipy.integrate import solve_ivp # For math

#=====================================================
# Times
#=====================================================
minutes = 60 # minutes to seconds
hours = 60 * minutes # hours to seconds
days = 24 * hours # days to seconds
years = 365.25 * days # years to seconds

#=====================================================
# Half Lives
#=====================================================
U238 = 4.468e9 * years
Th234 = 24.1 * days
Pr234 = 6.7 * hours
U234 = 245500 * years
Th230 = 75380 * years
Ra226 = 1600 * years
Rn222 = 3.8235 * days
Po218 = 3.1 * minutes
Pb214 = 26.8 * minutes
Bi214 = 19.9 * minutes
Po214 = 164.3e-6
Pb210 = 22.3 * years
Bi210 = 5.015 * years
Po210 = 138.376 * days

half_life = [U238, Th234, Pr234, U234, Th230, Ra226, Rn222, Po218, Pb214, Bi214,
    Po214, Pb210, Bi210, Po210]

#=====================================================
# Parameters
#=====================================================
steps = len(half_life)

#=====================================================
# Decay Solver
#=====================================================
def Decay(x, y, half_life=half_life):
    dydx = np.zeros(steps+1) # active elements + Pb206

```

```python
    dydx[0] = -y[0] / half_life[0] # for U238

    for i in range(1, steps):
        dydx[i] = y[i-1] / half_life[i-1] - y[i] / half_life[i]

    dydx[steps] = y[steps-1] / half_life[steps-1] # for Pb206

    return dydx

#==================================================
# Solve
#==================================================
t0 = 0 # initial time
tf = half_life[0] # final time (U238 half life)
y0 = np.zeros(steps+1)
y0[0] = 1 # initial value

ans = solve_ivp(Decay, [t0, tf], y0, method='Radau')

#print(ans)
np.savetxt("anst.txt", ans.t) # save times
np.savetxt("ansy.txt", ans.y) # save values

#==================================================
# Ratio of Pb206 to U238 (Numerical)
#==================================================
plt.plot(ans.t / years, ans.y[14,:] / ans.y[0,:], linewidth=5)
plt.title("Ratio of Pb206 to U238, Numerical")
plt.xlabel("Time (years)")
plt.ylabel("Pb206/U238")
plt.savefig('3.2.1.pdf', format='pdf', dpi=1200)
plt.show()
plt.close()

#==================================================
# Ratio of Pb206 to U238 (Analytical)
#==================================================
N_U238 = np.exp(-ans.t/U238) # remained U238 share
N_Pb206 = 1 - N_U238 # remained Pb206 share
Ratio_Pb206_U238 = N_Pb206 / N_U238 # Pb206/U238 ratio

plt.plot(ans.t / years, Ratio_Pb206_U238, linewidth=5, label="Analytical")
plt.plot(ans.t / years, ans.y[14,:] / ans.y[0,:], linewidth=5, ls=":", label="Numerical")
plt.title("Ratio of Pb206 to U238, Analytical")
plt.xlabel("Time (years)")
plt.ylabel("Pb206/U238")
plt.legend()
plt.savefig('3.2.2.pdf', format='pdf', dpi=1200)
plt.show()
plt.close()

#==================================================
# Ratio of Th230 to U234
#==================================================
tf = 10**6 * years # final time (of order 10 U234 half life)
```

```python
ans = solve_ivp(Decay, [t0, tf], y0, method='Radau')

plt.plot(ans.t / years, ans.y[4,:] / ans.y[3,:], linewidth=5)
plt.title("Ratio of Th230 to U234")
plt.xlabel("Time (years)")
plt.ylabel("Th230/U234")
plt.savefig('3.2.3.pdf', format='pdf', dpi=1200)
plt.show()
plt.close()
```

Times:

```
0.000000000000000000e+00
9.999999999999999124e-05
1.100000000000000066e-03
1.110000000000000049e-02
1.111000000000000043e-01
1.110999999999999977e+00
1.111100000000000042e+01
1.111109999999999933e+02
1.111111000000000079e+03
1.111111100000000190e+04
1.111111110000000092e+05
1.111111111000000096e+06
1.111111111100000126e+07
1.111111111110000180e+08
1.111111111111000197e+09
1.111111111111100306e+10
1.111111111111111298e+11
1.111111111111111328e+12
7.806995782095624023e+12
1.973830379741707422e+13
3.991286183778035156e+13
7.463820897839914062e+13
1.615799280210896250e+14
5.723379524784597500e+14
3.959287074183670000e+15
3.782877829123576800e+16
1.207580455609994880e+17
1.409993568000000000e+17
```

Values:

```
1.000000000000000000e+00 1.000000000000000000e+00 1.000000000000000000e+00
    1.000000000000000000e+00 1.000000000000000000e+00 1.000000000000000000e+00
    9.999999999999998890e-01 9.999999999999992228e-01 9.999999999999921174e-01
    9.999999999999211742e-01 9.999999999992119637e-01 9.999999999921197480e-01
    9.999999999211974799e-01 9.999999992119743553e-01 9.999999921197434416e-01
    9.999999211974371915e-01 9.999992119746511365e-01 9.999921197744555679e-01
    9.999446325158467541e-01 9.998600211891672807e-01 9.997169688274828436e-01
    9.994707886671511110e-01 9.988546942000359197e-01 9.959490742942525010e-01
    9.723104055146319924e-01 7.646856796007812651e-01 4.246718018430449271e-01
    3.678813939284799606e-01
0.000000000000000000e+00 7.092230934034922930e-22 7.801454025565082162e-21
    7.872376315984778271e-20 7.879468357693541879e-19 7.880175688532696049e-18
    7.880227688332687336e-17 7.880045558802150938e-16 7.878154383572887939e-15
    7.859268881852139830e-14 7.673696465060417917e-13 6.106707793026649210e-12
    1.454882508545967492e-11 1.475815947616202790e-11 1.476766913952306419e-11
```

```
    1.476772574101613202e-11 1.476771530331216094e-11 1.476761056763850841e-11
    1.476690928877021270e-11 1.476565977180172291e-11 1.476354721358615240e-11
    1.475991169220016507e-11 1.475081337734015470e-11 1.470790397597820513e-11
    1.435881457020012546e-11 1.129266931177550751e-11 6.271437209015131084e-12
    5.432771972084276403e-12
0.000000000000000000e+00 1.703029171538913218e-32 2.060665268754244370e-30
    2.098301919579122687e-28 2.102081408103300628e-26 2.102430445831488696e-24
    2.102174400382396829e-22 2.099242629736514945e-20 2.070219217701588426e-18
    1.810327461554213250e-16 7.023415087825284174e-15 6.956405057484188022e-14
    1.684992160937186736e-13 1.709524848838443532e-13 1.710639327548455300e-13
    1.710645962344495223e-13 1.710644753323101817e-13 1.710632621078750107e-13
    1.710551387184946736e-13 1.710406647148831985e-13 1.710161935184026928e-13
    1.709740808052523999e-13 1.708686888454287115e-13 1.703716401090436170e-13
    1.663279004501336084e-13 1.308106576571729859e-13 7.264631621785713408e-14
    6.293148722851120043e-14
0.000000000000000000e+00 2.353550541921111253e-41 3.132575738452239261e-38
    3.218788309962499245e-35 3.227492193507637799e-32 3.228329933211482247e-29
    3.228078633657557669e-26 3.224705764297112088e-23 3.191196817395493212e-20
    2.884429212058432034e-17 1.363259762641349785e-14 1.703984666742564772e-12
    6.408520259135098487e-11 7.730911065135007686e-10 7.864754911576600367e-09
    7.873116413260100086e-08 7.823867100343429344e-07 7.341203629082692909e-06
    3.488542667956645040e-05 5.063429577758912423e-05 5.460303880456662123e-05
    5.491169289441687451e-05 5.488589279053303810e-05 5.472668885196898723e-05
    5.342777987932052135e-05 4.201894583607989128e-05 2.333541991934088943e-05
    2.021482653328221909e-05
0.000000000000000000e+00 7.594655423909445880e-59 1.111933491287768148e-54
    1.152922060941437695e-50 1.157081382280991148e-46 1.157488320557744717e-42
    1.157432906106262040e-38 1.156466995830706780e-34 1.146843609501139009e-30
    1.057692505677180980e-26 5.674284489378231140e-23 8.310744626182830755e-20
    3.903947229501656887e-17 5.440487858275754938e-15 5.628303940749809672e-13
    5.637211089274540178e-11 5.537053524632521197e-09 4.632132972634793115e-07
    8.280447018885588660e-06 1.496894976773721656e-05 1.672131870857804407e-05
    1.685960674815576240e-05 1.685276056817257357e-05 1.680393790713749180e-05
    1.640510790433112354e-05 1.290200234334538279e-05 7.165187904929428693e-06
    6.207003391289794694e-06
0.000000000000000000e+00 6.385250942078950746e-76 1.028351041910200632e-70
    1.075951830113731823e-65 1.080806404890106923e-60 1.081284944210266081e-55
    1.081254009113019976e-50 1.080463631388037930e-45 1.072581120715875310e-40
    9.999456971721490861e-36 5.859398974883894502e-31 9.748312606017899391e-27
    5.323537221727716008e-23 8.311519170354216227e-20 8.699090063358690631e-17
    8.317095587210996352e-14 5.403509338588368430e-11 9.051496844268907094e-09
    1.746586076742250768e-07 3.174656396186657768e-07 3.549030675185291022e-07
    3.578580725650613937e-07 3.577132446567480611e-07 3.566769725274883543e-07
    3.482114890668466961e-07 2.738552817952854543e-07 1.520868234714543004e-07
    1.317485935586369991e-07
0.000000000000000000e+00 2.318437830817747792e-91 3.944634562964573329e-85
    4.152984940021652396e-79 4.174338636766720416e-73 4.176443247433275349e-67
    4.176293289937802620e-61 4.172678083680730913e-55 4.136758307750692687e-49
    3.816189523453361382e-43 2.195091053281786226e-37 2.682951640010545313e-32
    3.152231499899348028e-28 5.388803410194833546e-25 5.686399908636559338e-22
    5.441072889234180780e-19 3.535277057537993086e-16 5.922036046811514511e-14
    1.142722722304511976e-12 2.077053159692839532e-12 2.321991577949625883e-12
    2.341325017869255235e-12 2.340377465687126011e-12 2.333597543569501827e-12
    2.278211205424722077e-12 1.791727703536494352e-12 9.950444379611018016e-13
    8.619793762365311852e-13
0.000000000000000000e+00 1.658827393512713108e-101 2.587327525072834187e-94
```

```
    2.709824895772168632e-87 2.722029470183345198e-80 2.719309177898640449e-73
    2.680339594737623062e-66 2.352376532829009614e-59 1.171165646618336311e-52
    1.958449493245202085e-46 1.224548870316074396e-40 1.509434449596337012e-35
    1.774722907955013978e-31 3.034088327299721768e-28 3.201660575191815859e-25
    3.063533470792830072e-22 1.990497150467447499e-19 3.334334404792682876e-17
    6.433969092929998088e-16 1.169460935121292398e-15 1.307370699410505204e-15
    1.318256170792224556e-15 1.317722662414835717e-15 1.313905305106117315e-15
    1.282720663298723979e-15 1.008811712829130524e-15 5.602483467880323096e-16
    4.853275914907171379e-16
0.000000000000000000e+00 3.121449285267101734e-108 4.240894833873194875e-100
    4.383178744239747680e-92 4.396924823052601917e-84 4.389834737498218028e-76
    4.305966350882929882e-68 3.597373591802629156e-60 1.252465649579015127e-52
    9.021706792801974493e-46 9.788877745556145191e-40 1.296225287648463275e-34
    1.533493235875833035e-30 2.622901972103069249e-27 2.767875186165628853e-24
    2.648472968212651155e-21 1.720816826977211059e-18 2.882585864613888207e-16
    5.562270053416987642e-15 1.011017840658918793e-14 1.130243056262525418e-14
    1.139653721846142078e-14 1.139192495248964643e-14 1.135892328285301465e-14
    1.108932702464715871e-14 8.721339968974518529e-15 4.843437320619173137e-15
    4.195735307081085941e-15
0.000000000000000000e+00 7.238613199826900013e-116 9.502124230993033920e-107
    9.770050834889260227e-98 9.795315735239347308e-89 9.777687920891465060e-80
    9.577594458092058516e-71 7.860279746651382337e-62 1.997197056575570233e-53
    4.041171106527290440e-46 6.849546055692198384e-40 9.577140518027768640e-35
    1.138244155417248726e-30 1.947538448711963757e-27 2.055243976872912715e-24
    1.966589379963904372e-21 1.277770668108936144e-18 2.140427559284976869e-16
    4.130193061436822189e-15 7.507184712208821363e-15 8.392476425221677115e-15
    8.462354128633449929e-15 8.458929349050203279e-15 8.434424377939434078e-15
    8.234239096659714072e-15 6.475920350096804730e-15 3.596432935832923728e-15
    3.115490022795310929e-15
0.000000000000000000e+00 1.684074672306155585e-123 9.027098239394695239e-114
    1.285906675782262953e-104 1.341733053085584562e-95 1.344839028920742951e-86
    1.317861560332727497e-77 1.081606449735012476e-68 2.748235437938658536e-60
    5.560840474992358824e-53 9.425296545024371270e-47 1.317859452287201699e-41
    1.566277342757888130e-37 2.679904247252308664e-34 2.828112105528221842e-31
    2.706119222178018471e-28 1.758272368260447588e-25 2.945328710138372270e-23
    5.683339363434420987e-22 1.033023825976473436e-21 1.154844117808979807e-21
    1.164459617533061860e-21 1.163988351799789108e-21 1.160616352843759924e-21
    1.133069919247228678e-21 8.911170129990830594e-22 4.948860396627716731e-22
    4.287060391501420530e-22
0.000000000000000000e+00 2.889145649605530182e-124 1.766472507469623993e-113
    2.621240859862683194e-103 2.744109342519344199e-93 2.752018686928002568e-83
    2.704177322685962219e-73 2.275131479565872885e-63 6.427152201796331594e-54
    9.019138730520223327e-46 1.057054019179808838e-38 1.552558809084745595e-32
    2.278457307518736748e-27 4.303081847530604672e-23 3.595598026452033578e-19
    9.677444168119778202e-16 7.408887210367839739e-13 1.260036076821610128e-10
    2.434090469052879897e-09 4.424626402884112384e-09 4.946457592797285482e-09
    4.987646810858350736e-09 4.985628366670328638e-09 4.971185329192103609e-09
    4.853197653101644222e-09 3.816858009080932460e-09 2.119710112717979202e-09
    1.836246031892635397e-09
0.000000000000000000e+00 1.004630320161697714e-137 6.901553139234404536e-126
    1.079855068101911623e-114 1.138377818255150313e-103 1.142718581461253485e-92
    1.125420346543401213e-81 9.672679997646335879e-71 3.127597506844408160e-60
    4.890618875698469429e-51 3.745684740205029633e-43 4.079098380843649745e-36
    6.243629355355336713e-30 1.192571760146778685e-24 5.079695830729783353e-20
    2.082362273129719226e-16 1.660002029890719612e-13 2.832902160525665656e-11
    5.473866637458213527e-10 9.950423307289133623e-10 1.112398225080370582e-09
```

```
    1.121661375395671564e-09 1.121207456527835920e-09 1.117959392538943939e-09
    1.091425392718452737e-09 8.583651541267196095e-10 4.766971402337419018e-10
    4.129495004608849282e-10
0.000000000000000000e+00 1.451481325444621096e-150 1.059601844307657010e-137
    1.734835890090768308e-125 1.842073386558337505e-113 1.850706583252921676e-101
    1.824987350478761905e-89 1.588059617377494810e-77 5.704172383921928124e-66
    1.145625142226107320e-55 8.523376285880507858e-47 6.370092533203784625e-39
    6.158413076517610730e-32 5.692340686801571531e-26 3.671330341780377970e-21
    1.567747266802261375e-17 1.253678988276082538e-14 2.140042065320770926e-12
    4.135168155365558360e-11 7.516939974505877343e-11 8.403493952209118091e-11
    8.473471546583785356e-11 8.470042467514744106e-11 8.445505313762371791e-11
    8.245057034542574123e-11 6.484428252682627452e-11 3.601157839695285424e-11
    3.119583075857746713e-11
0.000000000000000000e+00 2.936259161605844844e-162 2.057413551960968298e-148
    3.426602477060327817e-135 3.654623294776613978e-122 3.673437426699245411e-109
    3.621617373598194679e-96 3.149340078118592299e-83 1.187726234867972375e-70
    3.001983967933352300e-59 2.637123695934813163e-49 1.941544876299556669e-40
    1.204372900709233822e-32 8.635566317083085336e-26 6.360665660448736908e-20
    3.433697456548261236e-15 3.169164941911068794e-11 6.658565012930239751e-08
    1.202391292729417557e-05 7.405258777707455901e-05 2.113457517681151699e-04
    4.570759637973645460e-04 1.073203224541161276e-03 3.979032211182991635e-03
    2.761940734219757790e-02 2.352591208424584246e-01 5.752975428225401977e-01
    6.320920502062449264e-01
```

b.

The ratio of Pb206 to U238 appears in Fig.(3)

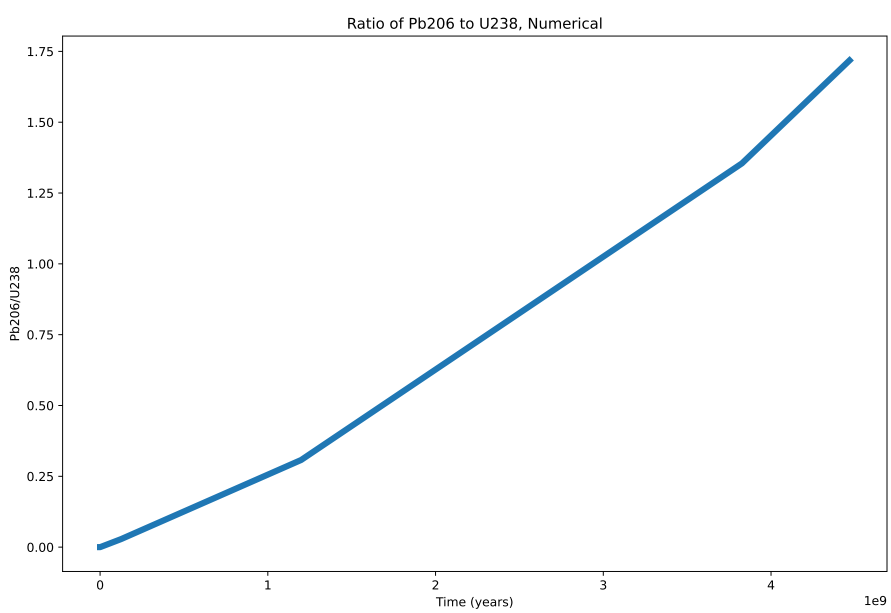

Figure 3: Ratio of Pb206 to U238, numerical

Looking to it analytically, while making the assumption that U238 decaying instantly to lead directly since all the half-lives are short compared to U238,

$$N_R(Element) = N_0 \times \exp(-\frac{t}{t_{1/2}}) \tag{4}$$

Where N_R is the remained amount of the element and N_0 is the initial amount of the element. Here is what we found from the code:

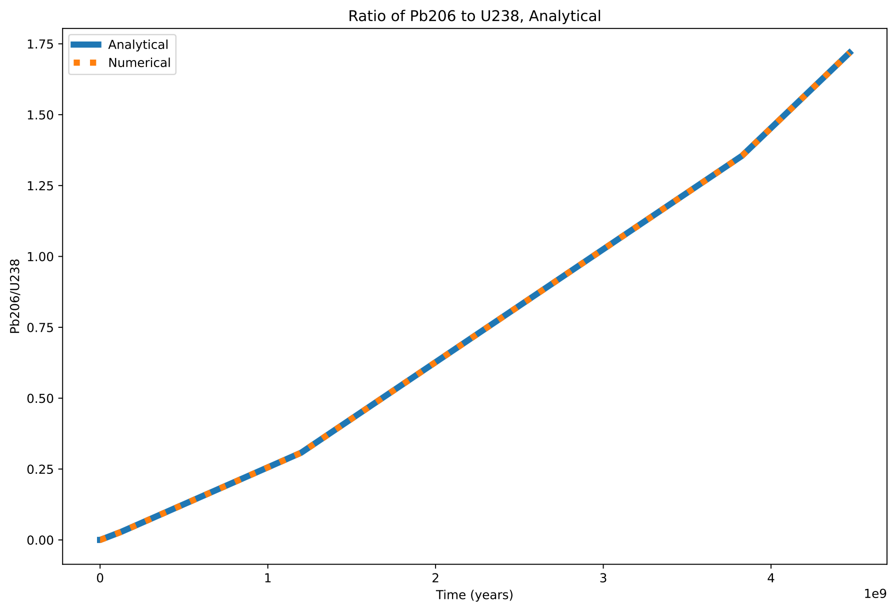

Figure 4: Ratio of Pb206 to U238, analytical

They're pretty identical.

Now, the ratio of Th230 to U234:

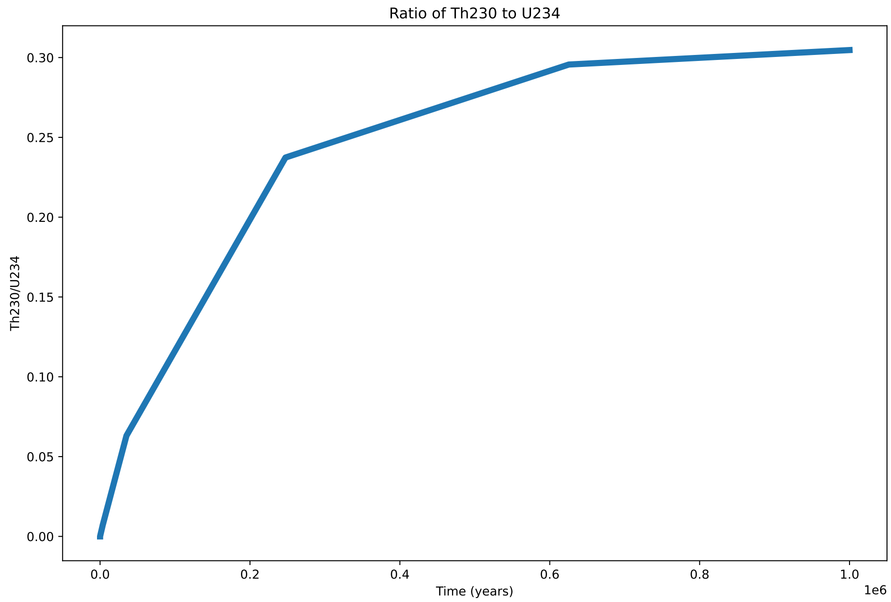

Figure 5: Ratio of Th230 to U234

Problem 3

a.

Simplify the function:

$$z = z_0 + a(x^2 + y^2) + a(x_0^2 + y_0^2) - 2axx_0 - 2ayy_0 \tag{5}$$

Rewrite it as:

$$z = a(x^2 + y^2) + bx + cy + d \tag{6}$$

Where $b = -2ax_0$, $c = -2ay_0$, and $d = a(x_0^2 + y_0^2) + z_0$

b.

Use the code:

```
# Course: PHYS 512
# Problem: PS3 P3
#=====================================================
# By: Muath Hamidi
# Email: muath.hamidi@mail.mcgill.ca
# Department of Physics, McGill University
# September 2022

#=====================================================
# Libraries
#=====================================================
import numpy as np # For math
```

```python
import matplotlib.pyplot as plt # For graphs

#==================================================
# Data
#==================================================
data = np.loadtxt('dish_zenith.txt')
x = np.array(data[:,0])
y = np.array(data[:,1])
z = np.array(data[:,2])

#==================================================
# Matrix
#==================================================
A = np.zeros([len(x),4])
A[:,0] = x**2 + y**2 # with a
A[:,1] = x # with b
A[:,2] = y # with c
A[:,3] = 1 # with d

#==================================================
# Fit
#==================================================
lhs = A.T@A
rhs = A.T@z
v = np.linalg.inv(lhs)@rhs
z_pred = A@v

#==================================================
# Parameters
#==================================================
a, b, c, d = np.linalg.inv(lhs)@rhs

# Our parameters
x0 = b/(-2*a)
y0 = c/(-2*a)
z0 = d - a * (x0**2 + y0**2)
print("The best-fit parameters: a = {}, x0 = {}, y0 = {}, z0 = {}".format(a, x0, y0
    , z0))

#==================================================
# Error
#==================================================
Noise = np.mean((z - z_pred)**2)  # noise
uncertainty = np.sqrt(Noise * np.diag(np.linalg.inv(lhs)))
print("The uncertainty in a is {}".format(uncertainty[0]))

#==================================================
# Error Bar
#==================================================
focal_length = 1/(4*a) # focal length
print("Focal length =", focal_length)

Error = 1/4 * 1/(a)**2 * uncertainty[0]
print("Focal length error =", Error)
```

The best-fit parameters:
$a = 0.00016670445477401358$,

$x_0 = -1.360488622197728,$
$y_0 = 58.22147608157934,$
$z_0 = -1512.8772100367878$

c.

From the code, the uncertainty in a is 6.451899757263455e-08.

Considering the equation in the question, the focal length is,

$$Focal = \frac{1}{4a} \tag{7}$$

With error,

$$Error(Focal) = \frac{\Delta a}{4a^2} \tag{8}$$

So, we had the results:
Focal length = 1499.659984125216
Focal length error = 0.5804077581892833