Report

Exercise 5.1: Radioactive decay chain

According to the equation (10.6) in the text book,

$$-\Delta N = \lambda N \Delta t$$

$$-\frac{\Delta N}{N} = \lambda \Delta \equiv P$$

where N is the number of number of nuclei which not yet decayed at time t, $-\Delta N$ denotes the number of nuclei that will decay in one time step, and λ is the decay constant.

Then, apply the equation $\tau = 1/\lambda$ to the equation (10.6)

$$\Longrightarrow P \equiv \frac{\Delta t}{\tau}$$

where τ is the half life.

Above equation allows us to find the probability of decay in one time step:

$$P_{Pb} = \frac{\Delta t}{\tau} = \frac{1}{3.3 * 60}$$

$$P_{Ti} = \frac{\Delta t}{\tau} = \frac{1}{2.2 * 60}$$

$$P_{Bi} = \frac{\Delta t}{\tau} = \frac{1}{46 * 60}$$

After found theses probabilities, we can apply random sampling and get the number of $Bi209$, $Pb209$, $Ti209$, and $Bi213$ at any time t.

Python code:

```
from numpy import arange
from matplotlib import pyplot as plt
import random as rnd

# initial number of these four isotope in the decay chain
Bi209 = 0
Pb209 = 0
Ti209 = 0
Bi213 = 10000

deltaT = 1.   # denotes  t  = 1s (divide time into slices of length of 1 s)

#Half Life of Pb, Ti, and Bi in seconds
hPb = 3.3 * 60
hTi = 2.2 * 60
hBi = 46 * 60

# Probility of decay in a single time slice (1 second here)
pPb = deltaT * 1/hPb
pTi = deltaT * 1/hTi
pBi = deltaT * 1/hBi

Bi209_points = []
Pb209_points = []
```

```python
Ti209_points = []
Bi213_points = []

t_start = 0     # Start time
t_end = 20000   # End time
tpoints = arange(t_start, t_end, deltaT)
for t in tpoints:
    Bi209_points.append(Bi209)
    Pb209_points.append(Pb209)
    Ti209_points.append(Ti209)
    Bi213_points.append(Bi213)

    # Part(a)
    for i in range(Pb209):
        x = rnd.random()
        if x<pPb:
            Pb209-=1
            Bi209+=1

    # Part(b)
    for i in range(Ti209):
        y = rnd.random()
        if y<pTi:
            Ti209-=1
            Pb209+=1

    # Part(c)
    for i in range(Bi213):
        z = rnd.random()
        if z<pBi:
            Bi213 -=1
            if rnd.random()<0.9791:
                Pb209+=1
            else:
                Ti209+=1

plt.plot(tpoints, Bi209_points, label='Bi209')
plt.plot(tpoints, Pb209_points, label='Pb209')
plt.plot(tpoints, Ti209_points, label='Ti209')
plt.plot(tpoints, Bi213_points, label='Bi213')
plt.xlabel('t(s)')
plt.ylabel('Number of atoms')
plt.legend()
plt.show()
```

Sample output: