PHY411 Report: Assignment 1

Name: Athira Sreejith
Roll no: MS18033
email: ms18033@iisermohali.ac.in
Lab session: 1 (10/9/2020)

Contents

---

1 Introduction

Coin tosses and nuclear decays of unstable nuclei are simulated using pseudo-random numbers, and the results are analysed.

2 Problem 1

2.1 Sub problem 1

2.1.1 Problem

Toss n coins for 1000 times, and estimate the mean of occurrence of heads and tails. Repeat the same by reducing the number of coins to n/2, and keep repeating the experiment until number of coins is a single digit number, and plot the following:

- Mean of heads vs total tossed coins
- Mean of tails vs total tossed coins
- Mean of heads vs mean of tails

2.1.2 Algorithm

1. Define n = number of coins, $tossnumber = numberoftimeseachcoinistossed$
2. Draw a random number r from a uniform distribution between [0,1].
    - If $r > 0.5$, a head is obtained, and the number of heads is updated.
    - else, a tail is obtained, and the number of tails is updated.
3. Repeat the last step 1000 times to simulate 1000 tosses of a coin.
4. Calculate the mean number of heads and tails.
5. Repeat the step 2,3,4 n times.
6. redefine n = n/2, and add one if n/2 is odd. If n ¿ 9,repeat steps 2-5.
   If $n \leq 9$, plot the mean occurrences of heads and tails vs the total tossed coins, and the mean occurrence of heads vs tails.

2.1.3 Code

```
import random
import matplotlib.pyplot as pl


#Subproblem 1
n = 20000 #The number of coins to be tossed

mean_heads = []
mean_tails = []
toss_number = []

def toss(num,toss_num):
    toss_number.append(num)
    heads = 0 #count of the number of heads
    tails = 0 #count of the number of heads

    for i in range(num): #iterating through each coin
        for j in range(toss_num): #iterating through each toss of a single coin
            if random.random() > 0.5:
                heads += 1 #Head is obtained
            else:
                tails += 1 #Head is obtained
```

```
    mean_heads.append(heads/toss_num) #mean number of heads obtained
    mean_tails.append(tails/toss_num) #mean number of tails obtained


    num = int(num/2) #Halve the number of coins to be tossed

    if num%2 != 0:
        num += 1 #add 1 if the number of coins to be tossed is odd

    if num > 9:
        toss(num,toss_num) #repeat the experiment if number of coins to be tossed is not a single digit number

toss(n,1000)

pl.plot(toss_number,mean_heads,marker = 'o')
pl.ylabel("Mean number of heads")
pl.xlabel("Number of coins tossed")
pl.grid(True)
pl.savefig("Heads_vs_toss_number{}.png".format(n))
pl.show()

pl.plot(toss_number,mean_tails,marker = 'o')
pl.ylabel("Mean number of tails")
pl.xlabel("Number of coins tossed")
pl.grid(True)
pl.savefig("Tails_vs_toss_number{}.png".format(n))
pl.show()

pl.plot(mean_heads,mean_tails,marker = 'o')
pl.ylabel("Mean number of tails")
pl.xlabel("Mean number of heads")
pl.grid(True)
pl.savefig("Heads vs tails_{}tosses.png".format(n))
pl.show()
```

2.1.4 Results

Figure 1: Mean of heads vs total tossed coins

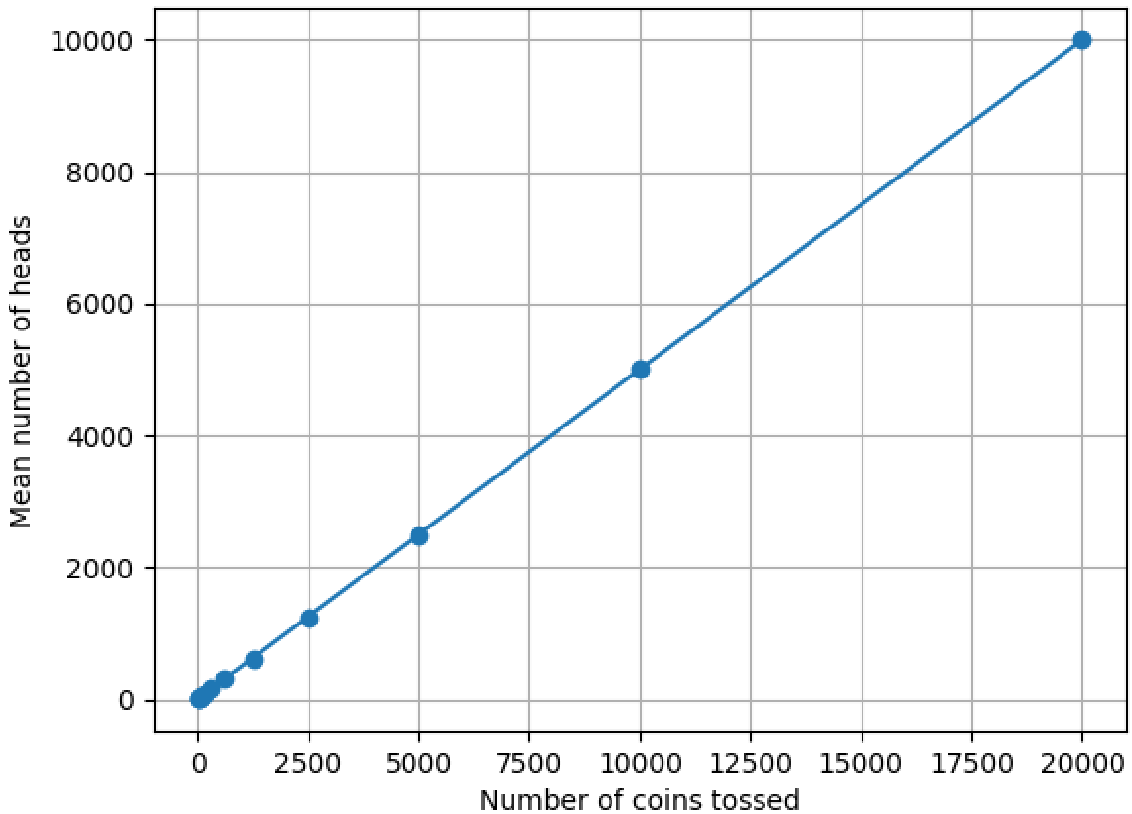


Figure 2: Mean of tails vs total tossed coins

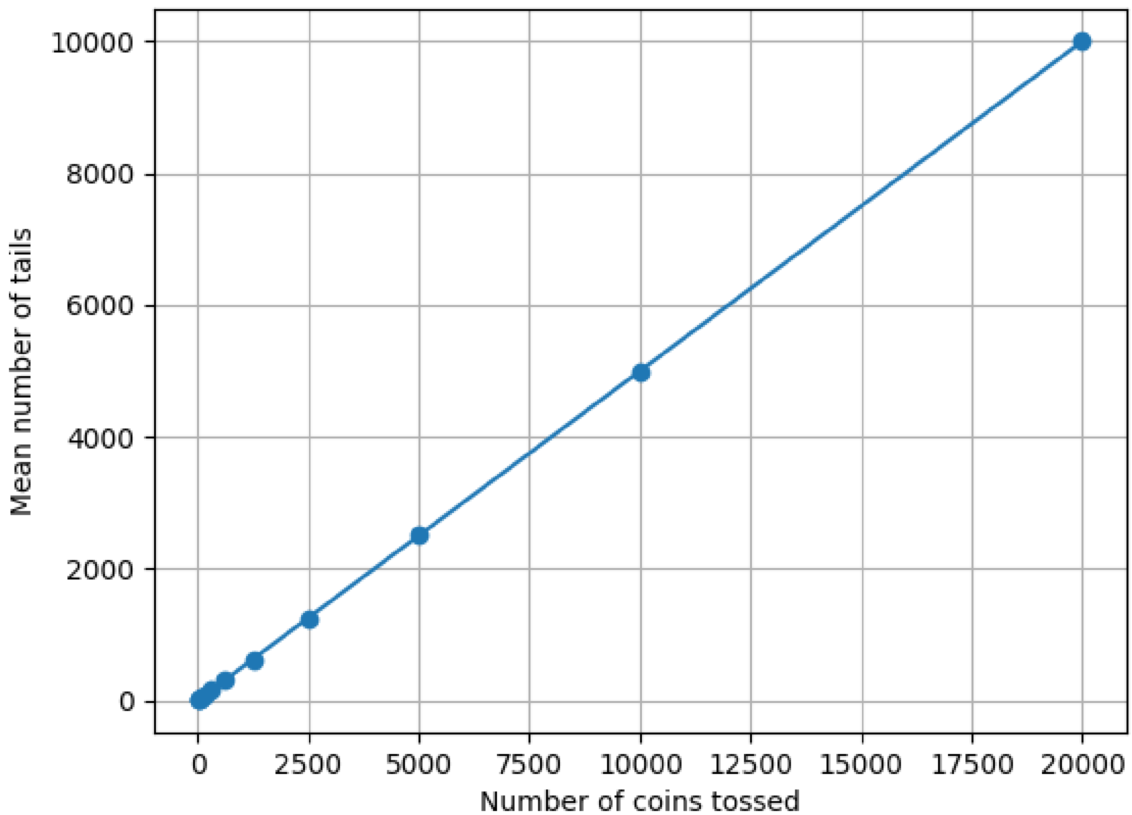
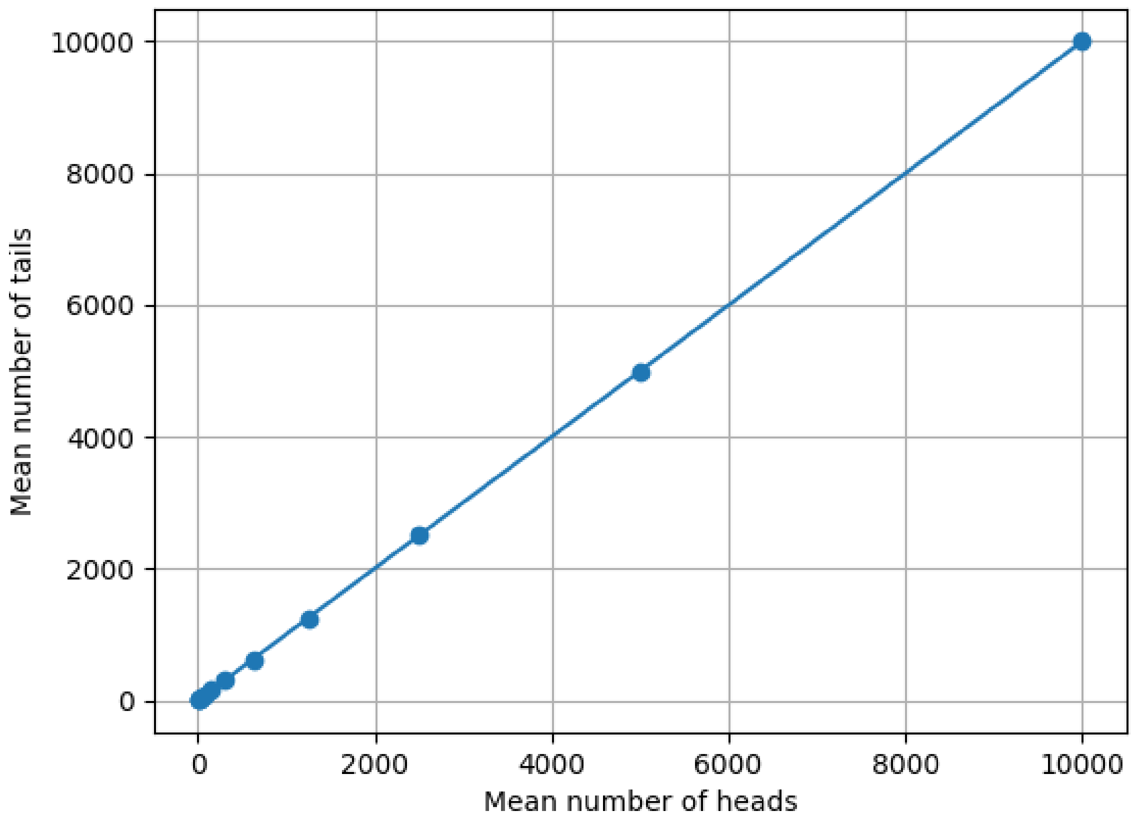


Figure 3: Mean of heads vs mean of tails



2.1.5 Observations

The mean number of occurrence of heads and tails are the same, and is roughly half the number of coin tosses, as expected for unbiased coins. For larger number of coin tosses, the approximation gets better, as is expected from the law of large numbers. We thus observe a straight-line graph for mean occurrences of heads(and tails) vs the total tossed coins, as well as for the mean of heads vs mean of tails.

2.2 Subproblem 2

2.2.1 Problem

Starting with 1,00,000 coins, toss each coin once, and count the number of heads. Toss each coin with head in the last round of tossing, and repeat this until the total number of coins to be tossed is a single-digit number. Plot the number of heads (H) remaining vs. number of tosses (T).

2.2.2 Algorithm

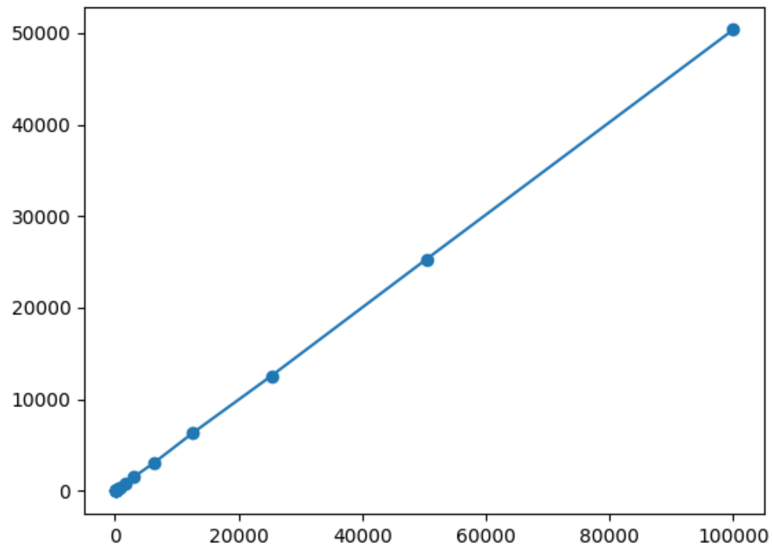
1. Define n = number of coins
2. Draw a random number r from a uniform distribution between [0,1].
   - If $r > 0.5$, a head is obtained, and the number of heads is updated.
   - else, a tail is obtained.
3. Repeat step-2 n times
4. If number of heads > 9, redefine n = number of heads, and repeat step 2,3.
5. If number of heads $\leq$ 9, plot the number of heads vs number of tosses for each value of n.

2.2.3 Code

```python
import random
import matplotlib.pyplot as pl

num_heads = []
toss_number = []

def toss_2(num):
    toss_number.append(num)
    heads = 0 #count of number of heads
    tails = 0 #count of number of tails

    for i in range(num): #iterating through each coin

        if random.random() > 0.5:
                heads += 1 #Head is obtained, update number of heads

    num_heads.append(heads)

    if heads > 9:
        toss_2(heads) #Repeat the experiment if the number of coins to be tossed is not a single digit number

    else:
        pl.plot(toss_number,num_heads,marker = 'o')
        pl.savefig("Q2_Heads_vs_Toss_number.png")
        pl.show()

toss_2(100000)
```

2.2.4 Results

Figure 4: Number of heads vs number of tosses



2.2.5 Observations

The number of heads is roughly equal to half the number of tosses for an unbiased coin, and we obtain a straight line plot for number of heads vs number of tosses, similar to sub problem 1. The number of tosses gets almost halved with each iteration, similar to sub problem 1 where we divide the number of coins by 2 in each step.

3 Problem 2

3.1 Problem

Radioactive decay process is a truly random process, with probability independent of the nucleus age. In a time of t, the probability that a nucleus undergoes decay is $p = \alpha\Delta t$ How does the number of parent Nuclei, N, change with time for a system with N_0 number of unstable nuclei initially?

3.2 Algorithm

1. Define N = number of nuclei
2. Initialise time = 0
3. Choose a random number r from uniform distribution between 0 and 1.
   - If r ¡ α Δ t Nuclei decays, N -= 1
   - Else, No decay
4. Repeat step-3 N times.
5. Update the number of nuclei N
6. Update time += time step

7. Repeat steps 3-6 until the time equals the time for which observation is to be made.

8. Plot the number of nuclei vs time.

3.3 Code

```python
import random
import matplotlib.pyplot as pl

def nuclei_track(n,alpha,dt,start,stop):
    time = [i for i in range(start,stop+1,dt)] #time
    print(len(time))
    n_list = [n] #list of number of nuclei at each time step

    for j in range(start,stop,dt):
        for i in range(n):
            if random.random() < alpha*dt:
                n -= 1 #Nucleus decays, update n

        n_list.append(n) #Update the list of nuclei at each time step

    return(n_list,time)


n0 = [100,9000,6*10**6] #List of initial number of nuceli
a = [0.01,0.05,0.2] #List of alpha values
delt = [1,1,10*60] #List of values of time-step

for k in range(len(n0)):
    nl,tl = nuclei_track(n0[k],a[k],delt[k],0,100*3600)
    pl.plot(tl,nl,marker = 'o')
    pl.xlabel("Time")
    pl.ylabel("Number of nuclei")
    pl.savefig("Nuclei_vs_time_{}n.png".format(k))
    pl.show()
```

3.4 Results

Figure 5: Number of nuclei vs time

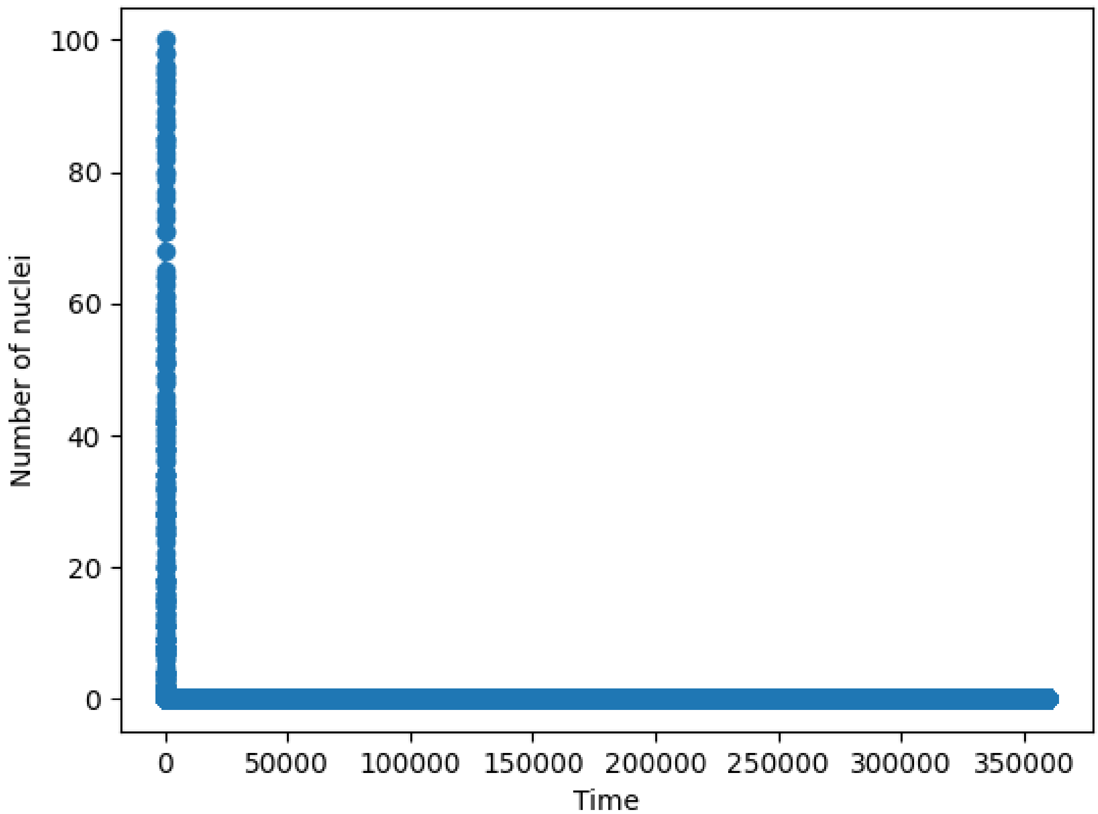


Figure 6: Number of nuclei vs time

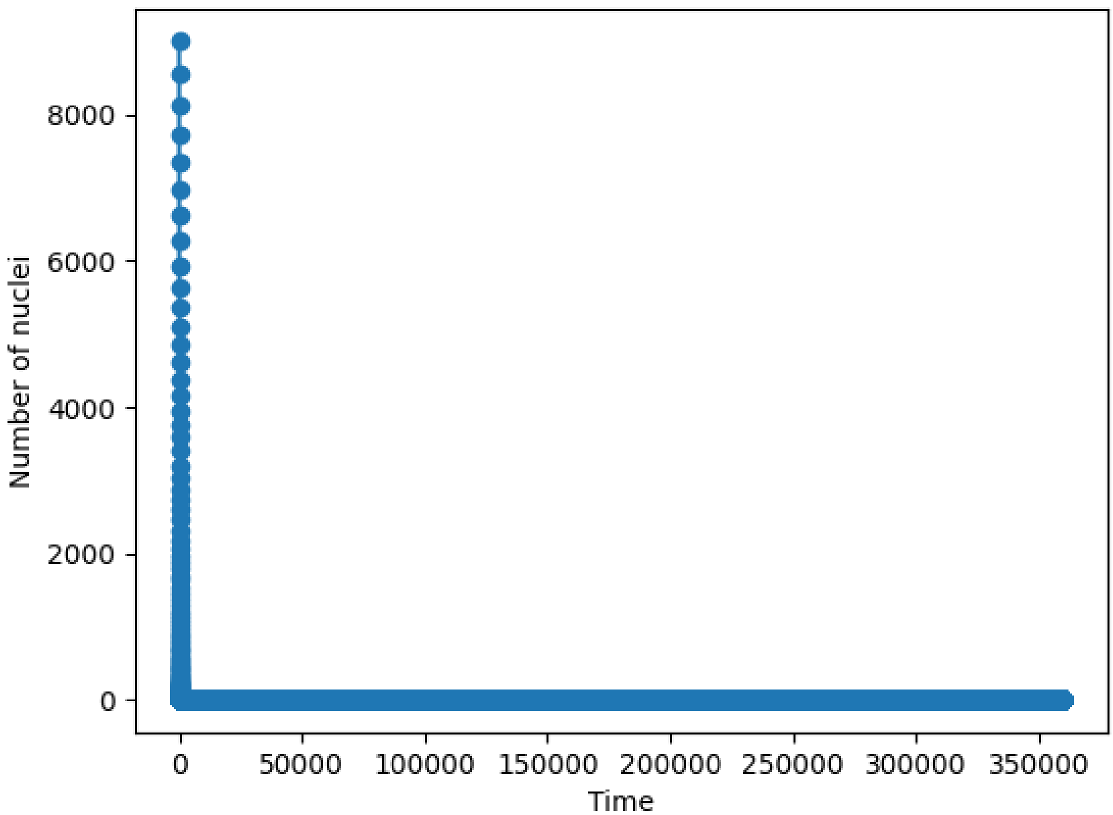
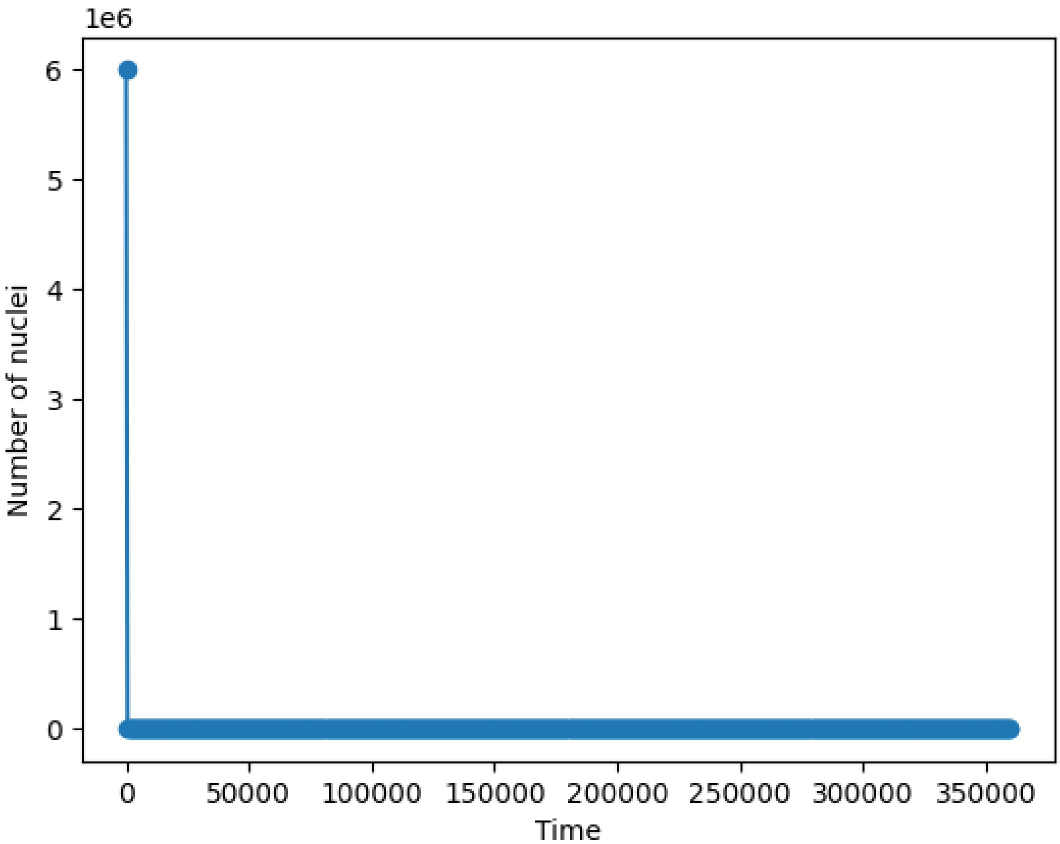


Figure 7: Number of nuclei vs time



3.5 Observations

The total number of nuclei vs time shows an exponential decay, as expected from $p = \alpha\Delta t$, due to which the number of nuclei that remains after each time-step is given by $N_t = N_{t-1} - N_{t-1}p = N_{t-1}(1 - (\alpha\Delta t))$.
The first plot shows more random deviations from an exponential curve, while the second plot with greater number of initial nuclei presents a better fit to exponential curve.
The third plot shows an abrupt fall in the number of nuclei, as the probability of decay is greater than 1, and all nuclei decays in the first time-step.

4 Conclusion

There is an interesting similarity in the two simulations performed.
The first sub problem of problem 1 shows how the mean number of heads obtained is roughly half the number of coins tossed for an unbiased coin, where probability of obtaining a head = probability of obtaining a tail $= \frac{1}{2}$.
Using the law of large numbers, the number of coin tosses as well as the number of remaining nuclei in each time step, can be obtained by $N_t = N_{t-1}(1 - p)$, where

- p = probability of getting a tail for coin-toss experiment
- p = the probability that a nucleus decays in a time step

Thus, both these show an exponential decay when plotted against time or number of iterations.
The second problem also illustrates how the choice of time-step is important in observing the actual trends

in the decay. In the third part of problem 2, all the nuclei decayed in a single time step. Here, we had probability greater than 1, as the chosen time-step was too large.

5 List of files submitted

1. PHY411-Report:Assignment 1.tex
2. PHY411-Report:Assignment 1.pdf
3. $PHY411\_1.1.1.py$
4. $PHY411\_1.1.2.py$
5. $PHY411\_1.2.py$