Rozpad radioaktywny

Mikołaj Wielebnowski Krzysztof Kubień

27 stycznia 2021

1 Wstęp

W niniejszym raporcie opisno prace przeprowadzone nad programem symulującym radioaktywny rozpad pierwiastków przy użyciu metod Monte Carlo w celu zasymulowania zjawiska w lepszym przybliżeniu niż rozwiązania analityczne.

2 Opis projektu

Etapy prac:

1. Założenia teoretyczne.
2. Implementacja metody Monte Carlo w rozpadzie radioaktywnym.
3. Implementacja rozwiązań analitycznych przy pomocy SciPy.
4. Porównanie otrzymanych danych z inną symulacją.
5. Implementacja zjawiska emisji cząstek radioaktywnych.
6. Podsumowanie.

2.1 Założenia teoretyczne.

Rozpad radioaktywny jest zjawiskiem opierającym się na losowym rozpadzie cząstek, do opisu których sotsujemy elementy fizyki statystycznej i pojęcia takie jak czas połowicznego rozpadu, które są dobrym przybliżeniem do obliczeń. Jednak natura zjawiska jest losowa, co pozwala na jej dokładniejsze zasymulowanie sotsując właśnie metody Monte Carlo. Do uzyskania wykresów mających sens oraz ułatwiających napisanie kodu zespół skupił się na wybraniu pierwiastków z szeregu rozpadu, których czas połowicznego rozpadu jest tego samego rzędu oraz następują one po sobie.

^{218}At			α, 99.9% β^-, 0.1%	1.5 s	6.874 2.881314	^{214}Bi ^{218}Rn
^{218}Rn			α	35 ms	7.26254	^{214}Po
^{214}Pb	RaB	Radium B	β^-	26.8 min	1.019237	^{214}Bi
^{214}Bi	RaC	Radium C	β^-, 99.979% α, 0.021%	19.9 min	3.269857 5.62119	^{214}Po ^{210}Tl
^{214}Po	RaC'	Radium C'	α	164.3 µs	7.83346	^{210}Pb
^{210}Tl	RaC"	Radium C"	β^-	1.3 min	5.48213	^{210}Pb
^{210}Pb	RaD	Radium D	β^-, 100% α, $1.9\cdot10^{-6}$%	22.20 a	0.063487 3.7923	^{210}Bi ^{206}Hg
^{210}Bi	RaE	Radium E	β^-, 100% α, $1.32\cdot10^{-4}$%	5.012 d	1.161234 5.03647	^{210}Po ^{206}Tl
^{210}Po	RaF	Radium F	α	138.376 d	5.03647	^{206}Pb
^{206}Hg			β^-	8.32 min	1.307649	^{206}Tl
^{206}Tl	RaE	Radium E	β^-	4.202 min	1.5322211	^{206}Pb
^{206}Pb	RaG	Radium G	stable	-	-	-

Rysunek 1: Część szeregu rozpadu uranowego.

Zespół wybrał rozpad ^{214}Pb i ^{214}Bi, ten wybór pozwolił zasymulować rozpad pierwszego pierwiastka na drugi zwiększając jego ilość w trakcie symulacji przy tym samym rzędzie wielkości rozpadu połowicznego. Dodatkową cechą później zaimplementowaną jest również emijsa cząstek α i β^- z niezerowym prawdopodobieństwem wystąpienia, któreś z nich.

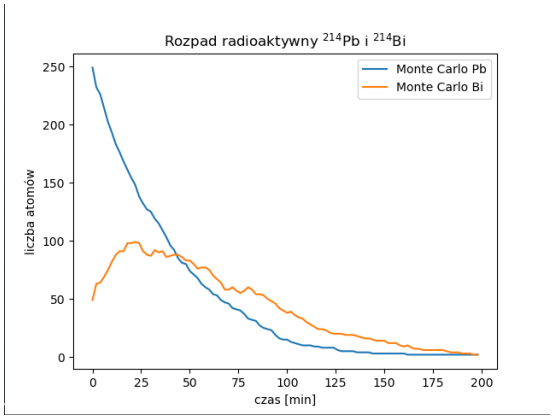
2.2 Implementacja metody Monte Carlo w rozpadzie radioaktywnym.

W wybranym zagadnieniu zastosowano liczby losowe w celu określenia czy dana cząsteczka się rozpadnie na podstawie wzoru na prawdopodobieństwo wystąpienia takiego zjawiska. Wylosowana liczba jest porównywana do wartości prawdopodobieństwa i jeżeli jest od niej większa zjawisko nie następuje. Dane podczas symulacji są zapisywane w formie tablicy, w której różne wartości są wykorzystywane do identyfikacji danego stanu, a następnie zapisywane w postaci kolejnej dla każdego przejścia czasowego i rzutowane na oś czasu.

```
def decayMC(timespan,timestep,N0a,N0b,dta,dt):
    dt = timespan/timestep
    na = np.zeros((timestep))
    nb = np.zeros((timestep))
    state = np.zeros((N0a+N0b))
    pa = 1 - np.exp(-dt / dta * np.log(2)) #prawdopodobienstwo rozpadu dla a
    pb = 1 - np.exp(-dt / dtb * np.log(2)) #prawdopodobienstwo rozpadu dla b
    for i in range(0,N0a-1):
        state[i] = 1
    for i in range(N0a,N0a+N0b-1):
        state[i] = 2
    for x in range(0,timestep):
        na[x] = (state == 1).sum()
        nb[x] = (state == 2).sum()
        for y in range(0,N0a+N0b):
            if state[y] == 1:
                if random.random() <= pa:
                    state[y] = 2
                else:
                    state[y] = 1
            elif state[y] == 2:
                if random.random() <= pb:
                    state[y] = 3
                else:
                    state[y] = 2
    return na, nb
```

Rysunek 2: Funckja wykorzystująca metodę Monte Carlo do symulacji rozpadu cząstek.

W tabeli oznaczenia określają odpowiednio: 1 - cząsteczka pierwiastku a, 2 - cząsteczka pierwiastku b, 3 - rozpad na kolejną cząsteczkę. Jak widać takie podejście umożliwia łatwy rozwój w celu symulacji większej części łańcucha.



Rysunek 3: Wykres ilości cząstek w czasie 200 minut, dla 250 atomów ołowiu i 50 bizmutu.

2.3 Implementacja rozwiązań analitycznych przy pomocy SciPy.

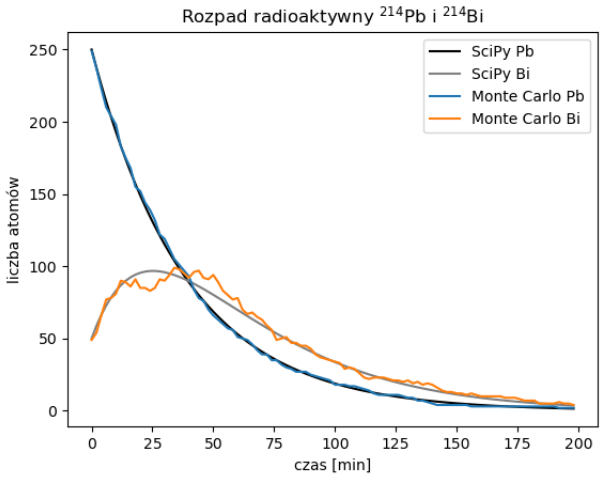
W celu sprawdzenia poprawności działania metody Monte Carlo zespół zaimplementował metodę analityczną przy użyciu odeint z biblioteki SciPy.

```
def f(N, t):
    na, nb = N
    deca = - na * np.log(2) / dta
    decb = - nb * np.log(2)/ dtb - deca
    return np.array((deca, decb))

t = np.arange(0, timespan, timespan/timestep)
decaySP = scipy.integrate.odeint(f, N0, t)
```

Rysunek 4: Część kodu rozwiązująca przewidywany rozpad przy użyciu biblioteki SciPy.

Rozwiązanie analityczne dobrze pokrywa się z wynikami z metody Monte Carlo, przez co zespół uznał, że implementacja została przeprowadzona porpawnie.

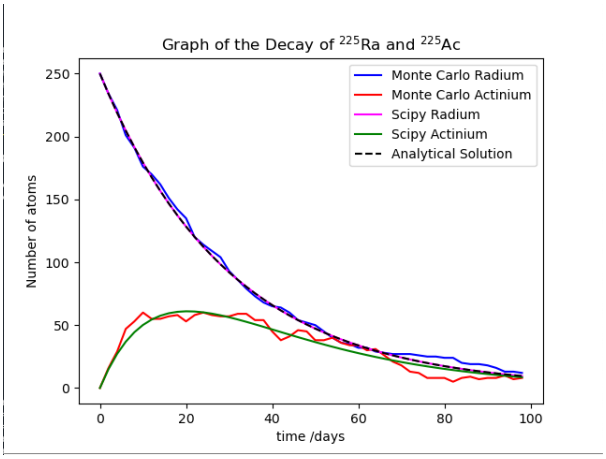


Rysunek 5: Wykres zawierający rozwiązania obu metod.

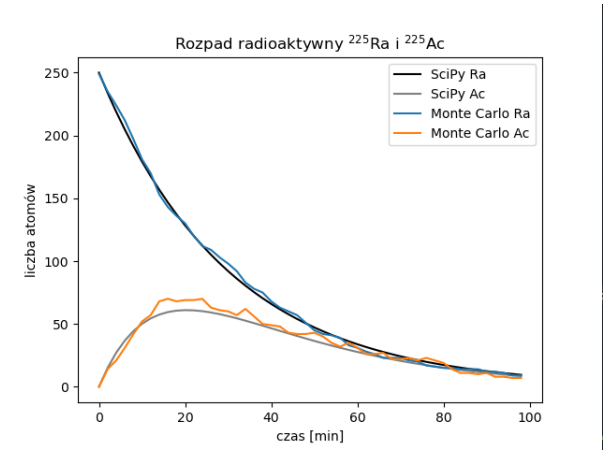
Powyższy wykres pokazuje przewagę metod Monte Carlo w wybranym zagadnieniu dokładniej obrazując naturę zjawiska.

2.4 Porównanie otrzymanych danych z inną symulacją.

W poszukiwaniu podobnych symulacji zespoł odnalazł symulajce w pythonie przeprowadzoną dla ^{225}Ra i ^{225}Az, zespoł przygotował odpowiednio kod, aby ułatwić zmianę zadanych pierwiastków oraz jendostki czasu.



Rysunek 6: Wykres otrzymany w wyniku symulacji znalezionej w internecie.



Rysunek 7: Wykres otrzymany w wyniku symulacji sporządzonej przez zespół.

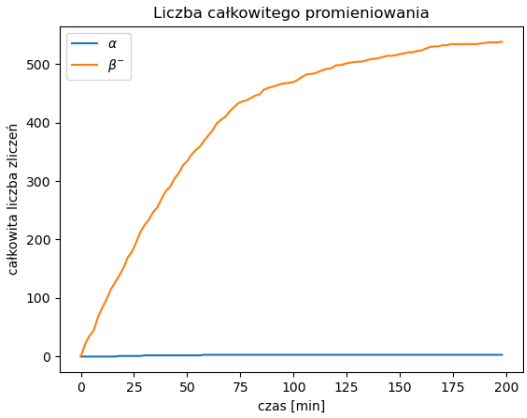
Powyższe wykresy się pokrywają, co poraz kolejny potwierdza poprawność implementacji zarówno naszego programu jak i autora drugiego.

3 Implementacja zjawiska emisji cząstek radioaktywnych.

W celu rozwinięcia programu zespół postanowił dodać wykres zliczeń radioaktywnych cząstek, po raz kolejny wykorzystując metody Monte Carlo, w celu określenia prawdopodobieństwa emisji danej cząsteczki. W tym celu rozwinięto wymiar tablicy o kolejne dwie kolumny, gdzie zostały zapisane wartości odpowiadające emisji cząstek dla rozpadu pierwszego i drugiego pierwiastka. Wartości prawdopodobieństwa dla każdej cząstki są porównywane, a następnie losowana jest liczba z zakresu od 0 do 1, taki zakres został wybrany, ponieważ któraś z cząsteczek musi zostać wyemitowana.

```
def decayMC(timespan,timestep,N0a,N0b,dta,dt,palfa1,palfa2,pbeta1,pbeta2):
    dt = timespan/timestep
    na = np.zeros((timestep))
    nb = np.zeros((timestep))
    nalfa = np.zeros((timestep))
    nbeta = np.zeros ((timestep))
    state = np.zeros((N0a+N0b,3))
    for i in range(0,N0a-1):
        state[i,0] = 1
    for i in range(N0a,N0a+N0b-1):
        state[i,0] = 2
    pa = 1 - np.exp(-dt / dta * np.log(2)) #prawdopodobienstwo rozpadu dla a
    pb = 1 - np.exp(-dt / dtb * np.log(2)) #prawdopodobienstwo rozpadu dla b
    if palfa1 > pbeta1:
        p1 = pbeta1
        a = 5
        b = 4
    else:
        p1 = palfa1
        a = 4
        b = 5
    if palfa2 > pbeta2:
        p2 = pbeta2
        c = 5
        d = 4
    else:
        p2 = palfa2
        c = 4
        d = 5
    for x in range(0,timestep):
        na[x] = (state == 1).sum()
        nb[x] = (state == 2).sum()
        nalfa[x] = (state == 4).sum()
        nbeta[x] = (state == 5).sum()
        for y in range(0,N0a+N0b):
            if state[y,0] == 1:
                if random.random() <= pa:
                    state[y,0] = 2
                    if random.uniform(0,1) <= p1:
                        state[y,1] = a
                    else:
                        state[y,1] = b
                else:
                    state[y,0] = 1
            elif state[y,0] == 2:
                if random.random() <= pb:
                    state[y,0] = 3
                    if random.uniform(0,1) <= p2:
                        state[y,2] = c
                    else:
                        state[y,2] = d
                else:
                    state[y,0] = 2
    return na, nb, nalfa, nbeta
```

Rysunek 8: Kod rozwiniętej metody Monte Carlo o zliczanie wyemitowanych cząstek radioaktywnych (α=3, β^-=538).



Rysunek 9: Wykres ilości cząstek wyemitowanych podczas rozpadu w zadanym okresie dla ołowiu oraz bizmutu.

4 Podsumowanie

Podsumowując prace związane z powyższym programem zespół stwierdza, że program spełnia swoje działanie zarówno dla rozpadu jak i emisji cząstek. Wykresy dla wybranych przez zespół pierwiastków jak i autora kodu dobrze obrazują zarówno zalety metod Monte Carlo, jak i samo w sobie symulowane zjawisko, ze względu na ich podobny czas połowicznego rozpadu. Program jest dość rozwojojowy i małym nakładem pracy można by go rozszerzyć do poziomu symulowania całego szeregu rozpadu radioaktywnego.

5 Załączniki

Do raportu zostaje dołączony plik radioactivedecay.py.

Literatura

[1] $MonteCarlo1.pdf$

[2] $https://en.wikipedia.org/wiki/Decay_chain$

[3] $https://gist.github.com/inTarga/e74dada2545bb5974869$