Jakie zmiany należałoby wprowadzić do programu, aby mógł on generować dłuższe łańcuchy?

3. Wyniki uzyskiwane przez program przedstawiony w ćwiczeniu 4 w niewielkim stopniu różnią się od wartości średnich dla wszystkich konformacji (dlaczego?). Można je poprawić przez wprowadzenie średniej ważonej Rosenblutha–Rosenblutha zdefiniowanej równaniem:

$$\langle A \rangle_{N,V,T} = \lim_{\tau \to \infty} \frac{\sum_{i=1}^{\tau} W_i A}{\sum_{i=1}^{\tau} W_i} \tag{5.7}$$

gdzie waga W_i wynosi:

$$W_{m+1} = \left(\frac{\omega_{\text{eff}}}{\omega - 1} \right) W_m \tag{5.8}$$

i $W_1 = 1$. W jaki sposób należałoby zmodyfikować program, aby obliczał on średni ważony kwadrat odległości pomiędzy końcami łańcucha w ćwiczeniu 4 oraz średnią ważoną wartość entropii konformacyjnej w ćwiczeniu 5 metodą Rosenblutha–Rosenblutha?

4. W jaki sposób należałoby zmienić program, aby mógł posłużyć do badania zachowania się łańcucha w ciasnej wnęce, np. w kapsydzie wirusa? Jakich wartości entropii łańcucha należałoby się wtedy spodziewać? Jaka zmiana entropii towarzyszyłaby wtedy uwolnieniu łańcucha do cytoplazmy komórki? Czy znając entropię łańcucha w kapsydzie i w cytoplazmie komórki, można oszacować siłę, jaką łańcuch (w tym przypadku DNA lub RNA) zostanie wtłoczony do komórki? (por. z p. 2.8.2. Entropia idealnego łańcucha).

Literatura uzupełniająca

1. Atkins A., de Paula J., *Chemia Fizyczna*, WN PWN, Warszawa 2021.
2. Teraoka I., *Polymer Solutions*, A John Wiley & Sons, Inc. Publication, New York 2002.
3. Sokal A.D., Monte Carlo and Molecular Dynamics Simulation w: *Polymer Science*, praca zbior. pod red. Bindera K., Oxford University Press, New York 1995.

Zagadnienia dodatkowe

Teoria Flory'ego–Hugginsa, liniowe rozmiary kłębka polimerowego: średnia kwadratowa odległość pomiędzy końcami łańcucha, średni promień bezwładności, błądzenie z samounikaniem.

DODATEK 1. Moduł graph

```
from graphics import *

def rysuj0(dim):
    win = GraphWin('Bladzenie przypadkowe', dim, dim)
    win.setCoords(0, 0, dim - 1, dim - 1)
    return win

def rysuj1(win, dim, edge, C, ML):

    p1=1
    p2=2
    asp = 10
    L=[0]
    for i in range(ML):
        L.append(0)
    for i in range(ML):
        x1 = int(((C[i,p1]) - edge / 2) * asp + dim / 2)
        y1 = int(((C[i,p2]) - edge / 2) * asp + dim / 2)
        x2 = int(((C[i+1,p1]) - edge / 2) * asp + dim / 2)
        y2 = int(((C[i+1,p2]) - edge / 2) * asp + dim / 2)
        L[i] = Line(Point(x1, y1), Point(x2, y2))
        L[i].draw(win)
    time.sleep(0.1)
    for i in range(ML):
        L[i].undraw()
```

DODATEK 2. Opracowanie wyników:entropia_dopasowanie.py

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import math



def dopasowanie():
   plik1='wyniki.txt'
   #plik1 = str(input(,Podaj nazwe pliku:  ,))
   funkcja1 = lambda x,C,gamma,omega: math.log(C)+(gamma-\
      1)*math.log(x)+x*math.log(omega)
   funkcja=np.vectorize(funkcja1)
   AA=np.loadtxt(plik1,delimiter=',',skiprows = 0)
   daneX=AA[:,0]
   daneY = AA[:, 3]
   plt.figure( num=None, figsize=(8, 6), dpi=80, facecolor='w', \
      edgecolor='k' )
```