Spis treści

Temat: Modele błądzenia
Omawiamy krótko algorytmy stosowane w tzw. 2-wymiarowych modelach błądzenia. Trzy rodzaje modeli. Próbkowanie Monte Carlo. Zadania.

Podstawa: Binder, Heermann, Monte Carlo simulations in statistical physics, Springer, 1997

1 Średniowanie w układach statystycznych

Statystyczna wartość średnia fizycznej wielkości $A(\mathbf{x})$, obliczona po zespole możliwych konfiguracji $\mathbf{x}$ układu fizycznego jest dana wyrażeniem

$$\langle A(\mathbf{x})\rangle = \mathbf{\frac{1}{Z}} \int \mathbf{dx} \exp[-\mathcal{H}(\mathbf{x})/\mathbf{k_B T}]\mathbf{A}(\mathbf{x})\,, \tag{1}$$

gdzie $\mathcal{H}(\mathbf{x})$ jest Hamiltonianem układu, k_B jest stałą Boltzmana, T temperaturą. Wielkość $\mathbf{x}$ reprezentuje przestrzeń parametrów układu (stopni swobody).

Dla przypomnienia podamy jeszcze wyrażenie na tzw. sumę statystyczna Z układu, która wchodzi do definicji średniej A:

$$Z = \int d\mathbf{x} \exp[-\mathcal{H}(\mathbf{x})/k_B T]\,. \tag{2}$$

Znajomość Z pozwala wyznaczyć ważne własności fizyczne układu.

Wielkość

$$p(\mathbf{x}) = \mathbf{\frac{1}{Z}} \int \mathbf{dx} \exp[-\mathcal{H}(\mathbf{x})/\mathbf{k_B T}]\,, \tag{3}$$

gra rolę prawdopodobieństwa pojawienia sie konfiguracji $\mathbf{x}$.

2 Losowanie konfiguracji. Metoda Monte Carlo

Metoda Monte Carlo wychodzi z założenia, że całkę (1) można zastąpić suma po skończonej liczbie konfiguracji wybranych losowo. Mamy

$$\langle A(\mathbf{x})\rangle = \frac{\mathbf{1}}{\mathbf{Z}} \frac{\sum_{\mathbf{l=1}} \exp[-\mathcal{H}(\mathbf{x_l})/\mathbf{k_B T}]\mathbf{A}(\mathbf{x_l})}{\sum_{\mathbf{l=1}} \exp[-\mathcal{H}(\mathbf{x_l})/\mathbf{k_B T}]} \,. \tag{4}$$

Losowy wybór konfiguracji pprzeprowadzany jest numerycznie z wykorzystaniem generatorów liczb pseudolosowych. Takie metody noszą nazwę metod Monte Carlo.

3 Błądzenie

Model błądzenia można stosować w wielu sytuacjach fizycznych, np. w zagadnieniach makromolekuł zawieszonych w cieczach. Najprostszy model błądzenia-polega na przypadkowym odwiedzaniu równoodległych punktów sieci (1-o, 2-u lub więcej wymiarowej) z jednakowym prawdopodobieństwem. Zadanie polega na znalezieniu konfiguracji (ciągu odwiedzanych punktów) złożonej z określonej liczby punktów.

Odróżnia się przy tym model prosty bez ograniczeń (BBO), model bez cofania się do poprzedniego punktu (BBC) oraz model bez przecięć (BBP). Kolejne mnemoniki oznaczaja *błądzenie bez ograniczeń / cofania / zapętleń*.

Sumy statystyczne dla odpowiadających modeli można obliczyć. Są one następujące:

$$\begin{aligned} Z_N^{BBO} &= z^N \qquad (5) \\ Z_N^{BBC} &= (z-1)^N \qquad (6) \\ Z_N^{BBP} &\to N^{\gamma-1} z_{\text{eff}}^N, \quad z_{\mathbf{eff}} < z-1 \,. \qquad (7) \end{aligned}$$

γ jest wykładnikiem krytycznym. Wielkość z jest liczbą współrzędnych. W przypadku 3 wymiarów ani *gamma* ani z_{eff} nie daja sie wyznaczyc analitycznie.

Średnia odległość $< R >$ pomiędzy początkiem i końcem drogi błądzącego wędrowca jest dana wzorem

$$< R >= \frac{1}{\text{Liczba}_\text{p}\text{rbek}} \sum_{i=1}^{\text{Liczba}_\text{p}\text{rbek}} r_i \,, \tag{8}$$

gdzie r_i jest odległością początek-koniec (p-k) i-tej próbki.

4 Generatory liczb pseudolosowych w Java^{TM}

Klasa `Random` wyposażona jest w wiele generatorów liczb losowych.

Generator liczb pseudolosowych rozpoczyna działanie korzystając z czasu zegara systemowego. Można też uruchomić go zadając tzw. `ziarno`, np. z konstruktorem `Random(long ziarno)` lub `setSeed(long ziarno)`.

Z a d a n i e 1.
(Arnold, Gosling) Napisz program, który testuje metodę `nextGaussian()`, wypisując rozkład wyników losowania dla dużej liczby prób w postaci wykresu (wystarczy wykres tekstowy z gwiazdek).

Tablica 1: *Publiczne (*`public`*) metody klasy* `Random`

metoda	**następna liczba pseudolosowa typu**
boolean nextBoolean()	boolean: `true`, `false`
double nextDouble()	double, $< 0.0, 1.0 >$
float nextFloat()	float, $< 0.0, 1.0 >$
int nextInt()	int $< Integer.MIN\_VALUE, Integer.MAX\_VALUE >$
int nextInt(int n)	int, $< -n, n >$
long nextLong()	long, $< Long.MIN\_VALUE, Long.MAX\_VALUE >$
nextGaussian()	z rozkładem Gaussa, średnia=0, odchylenie=1.
...	...

5 Zadania (BBO)

Z a d a n i e 2.
W bibliotece `Math` Java^{TM} znajdziemy metodę `random()`, która generuje liczby pseudolosowe z przedziału $(0,1)$. Przeprowadzić następujący test generatora. Wygenerować dostatecznie długi ciąg par liczb pseudolosowych (x_i, y_i). Odwzorować je na kwadrat (L, L), gdzie $L = 200$. Sprawdzić ńa okoćzy gęstość wygenerowanych punktów jest jednakowa. Obliczyć średnią gęstość punktów $< \rho(i,j) >$, ważąc liczbę punktów w otoczeniu węzła (i,j) według wzoru

$$< \rho(i,j) >= frac14\Delta^2 \sum_{k=i-\Delta,l=j-\Delta}^{k=i+\Delta,l=j+\Delta} \rho(k,l)\,,$$

gdzie $\rho(i,j)$ jest liczbą punktów w kwadracie <i,j,i+1,j+1>. Narysować $\rho(i,j)$ (Kolory). Wykonaj obliczenia dla $\Delta = 1, 2$.

Z a d a n i e 3.
Wykonaj opisany wyżej test dla metod podanych w Tabeli 1.

Z a d a n i e 4.
Napisać algorytm BBO. Na tej podstawie utworzyć program w Java^{TM}, który oblicza $< R >$ - odległość początek-koniec, w funkcji liczby kroków N, na sieci 2-u wymiarowej.

Z a d a n i e 5.
Wyznaczyć liczbę pętli wykonanych w BBO w funkcji N.

Z a d a n i e 6.
Sprawdzić numerycznie prawo Einsteina $< R^2 >\sim t$, gdzie t jest ćzasem"błądzenia, wykonując algorytm BBO w 2, 3 i 4 wymiarach.

6 Zadania (BBC)

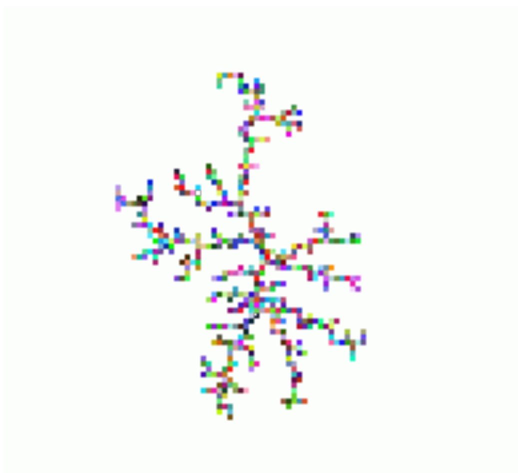
Z a d a n i e 7.
Napisz program (Java^{TM}) BBC.

7 Zadania (BBP)

Z a d a n i e 8.
Napisz program (Java^{TM}) BBP.

Z a d a n i e 9.
Napisać program realizujący algorytm BBP. Zmienieniając krok i rozmiar próbki wyznaczyć średnią długość początek-koniec w kierunkach x i y oddzielnie, fluktuacje tych wielkości i średni czas wykonania w funkcji długości kroku. Wyniki porównac z dokładnymi.

Z a d a n i e 10.
Diffusion Limited Agregation (DLA). W problemie DLA startuje sie z zarodzią kryształu umieszczoną w centrum sieci. Z brzegu sieci wyrusza atom - "błądzący wędrowiec". Jeśli znajdzie się on w węźle przylegającym do zarodzi to zostaje tam umieszczony na stałe. Wyrusza wtedy z brzegu sieci następny błądzący atom, itd. Napisać program realizujący schemat DLA.



Diffusion-limited aggregation