



Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий фізико-технічний інститут

Практичне застосування алгоритмів МСМС

предмет «Марковські моделі та їхнє застосування»

Роботу виконав:
Студент групи ФІ-91,
Цибульник Антон Владиславович

Роботу перевірила:
Ніщенко Ірина Іванівна

Завдання 1

Змоделювати ланцюг Маркова із заданою матрицею перехідних імовірностей:

$$P = \begin{pmatrix} 0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.5 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.3 & 0.2 & 0.0 \\ 0.0 & 0.0 & 0.3 & 0.0 & 0.0 & 0.7 \\ 0.0 & 0.0 & 0.2 & 0.0 & 0.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.7 & 0.0 & 0.3 \end{pmatrix}$$

З вигляду матриці перехідних імовірностей робимо висновок, що ланцюг Маркова складається з шести станів. Тож, наприклад, покладемо множину станів таким чином: $E = \{1, 2, 3, 4, 5, 6\}$. Наступним кроком на основі матриці $P = (p_{ij})_{i,j \in E}$ введемо допоміжні інтервали D_{ij} так, щоб

$$\begin{aligned} \forall i, j \in E : |D_{ij}| &= p_{ij} \\ \forall i \in E : \sum_{j \in E} D_{ij} &= 1 \end{aligned}$$

Крім того, покладемо функцію $\psi : E \times [0, 1] \rightarrow E$ так:

$$\psi(i, u) = \sum_{j \in E} \mathbb{1}(u \in D_{ij}) \cdot j$$

Тоді, задаючи першу ланку ланцюга рівномірно над множиною станів E , основний ітераційний алгоритм генерування усіх наступних елементів ланцюга Маркова матиме вид:

Лістинг 1: Генерування ланцюга Маркова

```
1 x0 = random.randint(0,E[-1])
2 x.append(x0)
3
4 for i in range(N):
5     u = random.uniform(0,1)
6     offer = psi(u,x[-1])
7     x.append(offer)
```

Тож виконаємо моделювання достатньо великого за довжиною ланцюга з параметром $N = 5000$. Обчисливши частоту відвідин ланцюгом кожного з шести станів, матимемо наближення інваріантного розподілу:

$$\pi = (0.0, 0.0, 0.2434, 0.2446, 0.2648, 0.2474)$$

В той час як справжній інваріантний розподіл π^* , який знаходиться як розв'язок рівняння $\pi^*P = \pi^*$, має вид:

$$\pi^* = (0.0, 0.0, 0.25, 0.25, 0.25, 0.25)$$

Обчислимо середню абсолютну похибку між отриманим наближенням π та розподілом π^* :

$$\delta_{\text{MAE}}(\pi, \pi^*) = \frac{1}{|E|} \sum_{i \in E} |\pi_i - \pi_i^*| = 0.00489$$

Зауважуємо, що значення похибки є малим, що свідчить про достатньо високу ефективність алгоритму МСМС. Наочанок проведемо генерування ланцюга Маркова для, наприклад, таких значень довжин ланцюга:

$$N = 100, 200, 300, \dots, 99\,800, 99\,900, 10\,000$$

При цьому щоразу генеруватимемо ланцюг, починаючи з іншого початкового стану, який обиратимемо рівномірно на множині E . На графіку нижче показані значення середньої абсолютної похибки в залежності від величини N :

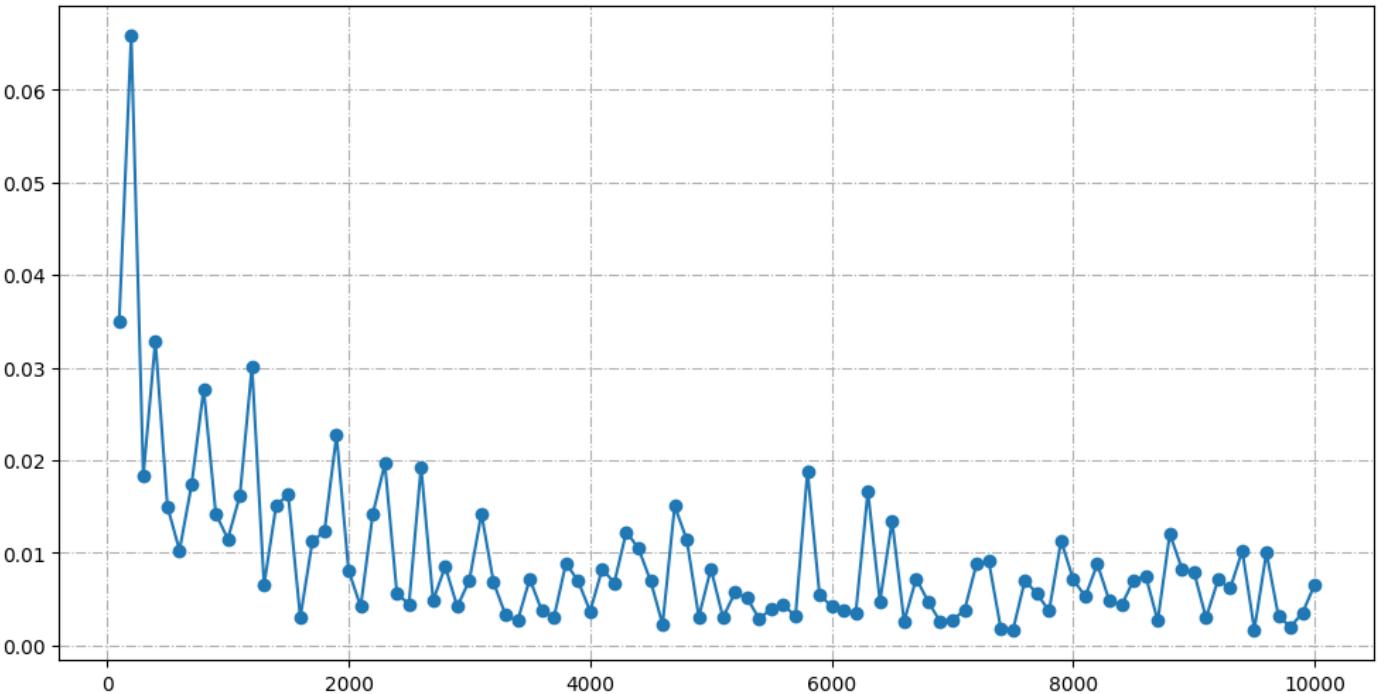


Рис. 1: Значення $\delta_{\text{MAE}}(\pi, \pi^*)$ (вісь ординат) в залежності від N (вісь абсцис)

Бачимо, що вже при перевищенні межі в $N = 2000$ станів досягається достатньо висока точність відповідності між інваріантними розподілами згенерованого та «оригінального» ланцюгів.

Завдання 2

Змоделювати ланцюг Маркова, для якого розподіл Зіффа з параметром $a = 4$ є інваріантним.

Схематично генерування ланцюга Маркова відрізняється від попереднього завдання тим, що цього разу функція $\psi(i, u)$ обчислюватиметься за допоміжними інтервалами деякої симетричної матриці Q , яка є наближенням (або припущенням щодо) матриці перехідних імовірностей. Наприклад, покладемо матрицю Q такою, яка відповідає випадковому блуканню на множині станів $E = \{1, 2, 3, \dots, M\}$, де візьмемо $M = 50$:

$$Q = \begin{pmatrix} 0.5 & 0.5 & 0.0 & \dots & 0.0 & 0.0 & 0.0 \\ 0.5 & 0.0 & 0.5 & \dots & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.0 & \dots & 0.0 & 0.0 & 0.0 \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ 0.0 & 0.0 & 0.0 & \dots & 0.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.0 & \dots & 0.5 & 0.0 & 0.5 \\ 0.0 & 0.0 & 0.0 & \dots & 0.0 & 0.5 & 0.5 \end{pmatrix}$$

Крім того, приймання/відхилення для стану i чергової пропозиції нового стану j залежатиме від величин α_{ij} :

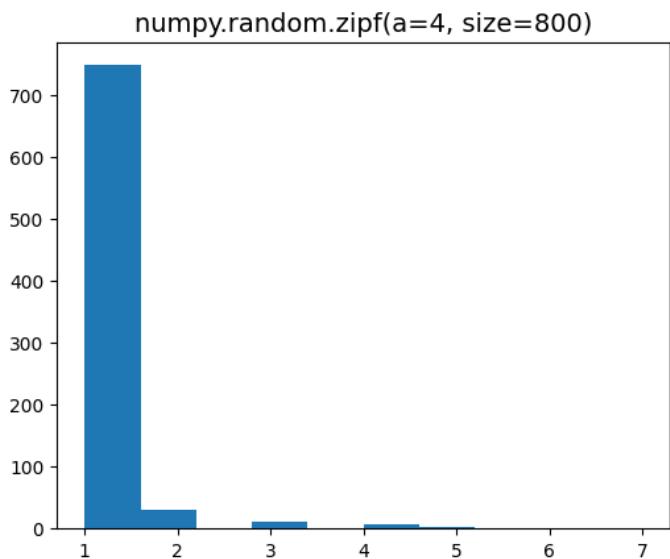
$$\alpha_{ij} = \min \left\{ \frac{i^a}{j^a}, 1 \right\}$$

Таким чином, задаючи перший стан ланцюга рівномірно над множиною станів E , основний ітераційний алгоритм генерування усіх наступних $N = 800$ елементів ланцюга Маркова матиме вид:

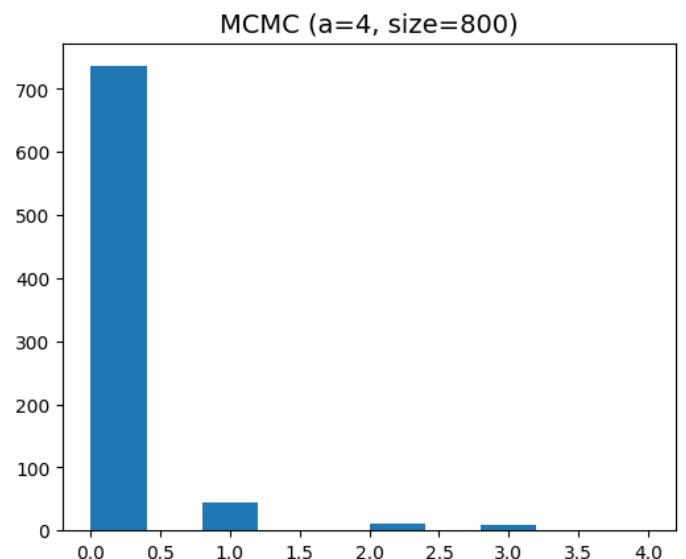
Лістинг 2: Генерування ланцюга Маркова

```
1 x0 = random.randint(0,M-1)
2 x.append(x0)
3
4 for i in range(N):
5     u = random.uniform(0,1)
6     offer = psi(u,x[-1])
7
8     alpha = min(pow(x[-1],a)/pow(offer,a), 1)
9
10    v = random.uniform(0,1)
11    if v <= alpha:
12        x.append(offer)
13    else:
14        x.append(x[-1])
```

Отримавши згенерований ланцюг, порівняємо гістограму отриманого розподілу та гістограму вибірки з розподілу Зіффа (згенерованої вбудованими методами мови Python). Бачимо, що графіки мають високу схожість:



а) Генерація вбудованими методами мови Python



б) Генерація методами згідно умов Завдання 2

Завдання 3

Змоделювати Бета-розподіл з параметрами $a = 2$ та $b = 5$ за допомогою алгоритму Метрополіса - Гастінга.

У цьому завданні генерування ланцюга Маркова зводитиметься виключно до роботи з величинами $\alpha_{xx'}$. Наприклад, для поточного стану x прийняття чи відхилення нової пропозиції x' спиратиметься на коефіцієнти виду:

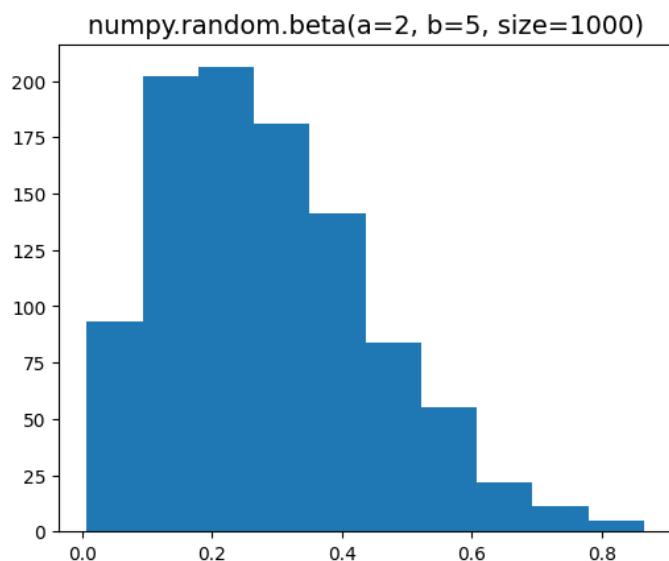
$$\alpha_{xx'} = \min \left\{ \frac{x'^a (1-x')^{b-1}}{x^a (1-x)^{b-1}}, 1 \right\}$$

Отже, задаючи перший стан ланцюга $X_0 \sim u(0, 1)$ рівномірно на відрізку $[0, 1]$, основний ітераційний алгоритм генерування усіх наступних $N = 1000$ елементів ланцюга Маркова матиме вид:

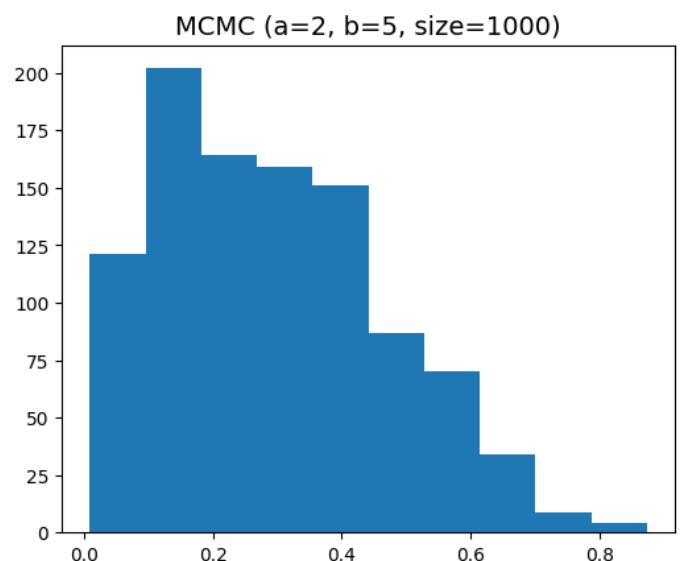
Лістинг 3: Генерування ланцюга Маркова

```
1 x0 = random.uniform(0,1)
2 x.append(x0)
3
4 for i in range(N):
5     offer = random.uniform(0,1)
6
7     numerator = pow(offer,a-1)*pow((1-offer),b-1)
8     denominator = pow(x[-1],a-1)*pow((1-x[-1]),b-1)
9     alpha = min(numerator/denominator, 1)
10
11    v = random.uniform(0,1)
12    if v <= alpha:
13        x.append(offer)
14    else:
15        x.append(x[-1])
```

Згенерувавши ланцюг, порівняємо гістограму отриманого розподілу та гістограму вибірки з Бета-розподілу (згенерованої вбудованими методами мови Python). Бачимо, що графіки, знову ж таки, дуже подібні (порівняльні графіки зображені на наступній сторінці).



а) Генерація вбудованими методами
мови Python



б) Генерація методами згідно умов
Завдання 3

Завдання 4

Побудувати байесову оцінку за допомогою алгоритму МСМС.

Нехай задано розподіл виду:

$$f_{\Theta|Y}(\theta|y) \sim e^{-\frac{1}{2\sigma^2}(y-\theta)^2} \cdot e^{-\frac{1}{2\sigma^2}(\theta-\mu)^2},$$

де параметри $y = 3$, $\mu = 0$, $\sigma^2 = 1$. Маємо на меті згенерувати такий ланцюг Маркова, для якого цей розподіл є інваріантним, при чому прийняття/відхилення для стану x чергової пропозиції x' відбудуватиметься в результаті аналізу значень таких величин:

$$\alpha_{xx'} = \min \left\{ \frac{f_{\Theta|Y}(x' | y)}{f_{\Theta|Y}(x | y)}, 1 \right\}$$

Задаючи перший стан ланцюга $X_0 \sim N(\mu, \tau^2)$, основний ітераційний алгоритм генерування усіх наступних $N = 10\,000$ елементів ланцюга Маркова (при заданих значеннях $\tau^2 = 4$, $d = 1$) матиме вид:

Лістинг 4: Генерування ланцюга Маркова

```
1 x0 = np.random.normal(mu, tau**2)
2 x.append(x0)
3
4 for i in range(N):
5     offer = x[-1] + np.random.normal(mu, d**2)
6
7     numerator = np.exp(-pow(y-offer,2)/(2*sigma**2)) *
8         np.exp(-pow(offer-mu,2)/(2*sigma**2))
9     denominator = np.exp(-pow(y-x[-1],2)/(2*sigma**2)) *
10        np.exp(-pow(x[-1]-mu,2)/(2*sigma**2))
11    alpha = min(numerator/denominator, 1)
12
13    v = random.uniform(0,1)
14    if v <= alpha:
15        x.append(offer)
16    else:
17        x.append(x[-1])
```

При цьому, теоретичне математичне сподівання та дисперсія заданого розподілу матимуть значення:

$$M(\theta|Y = y) = 2.4, D(\theta|Y = y) = 0.8$$

Спробуємо порівняти щойно наведені показники з характеристиками згенерованого в результаті алгоритму МСМС ланцюга Маркова. Зобразимо на наступній сторінці гістограму елементів змодельованого ланцюга (Рис. 2).

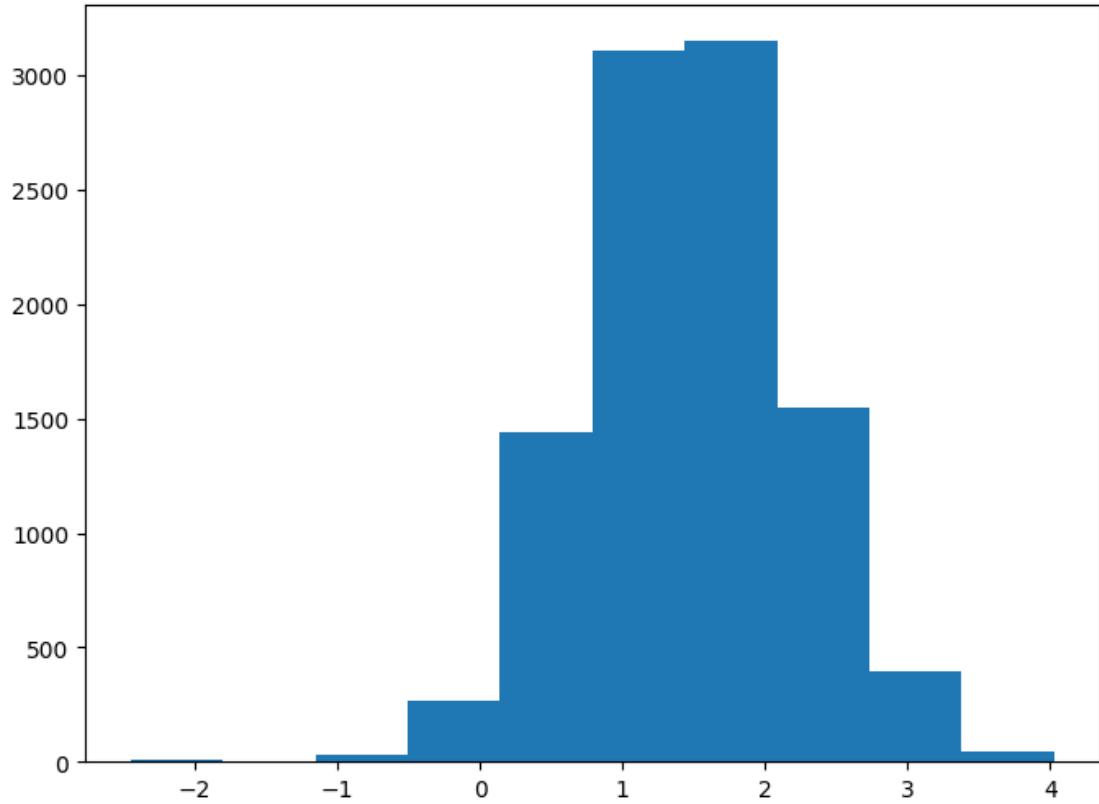


Рис. 2: Гістограма утвореного ланцюга Маркова

Вибіркове середнє та вибіркова дисперсія в даному випадку дорівнюють:

$$\bar{X} = 1.4665, S^2 = 0.5318$$

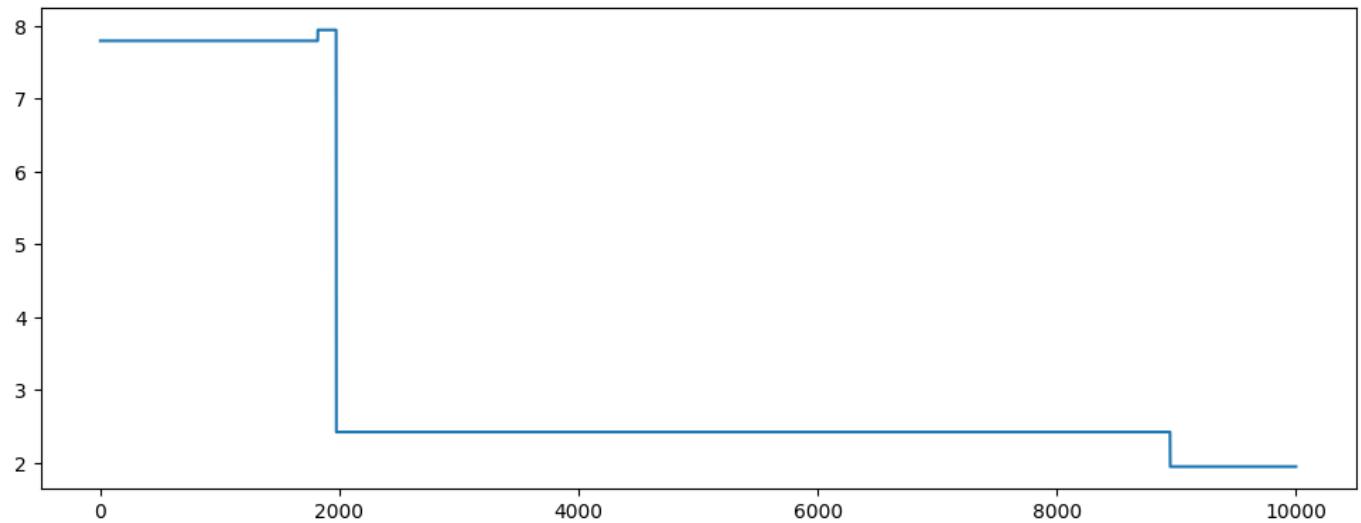
Наочності прослідкуємо за елементами X_n від ітерації до ітерації при різних значеннях параметра d :

$$d_1 = 100, d_2 = 1, d_3 = 0.1$$

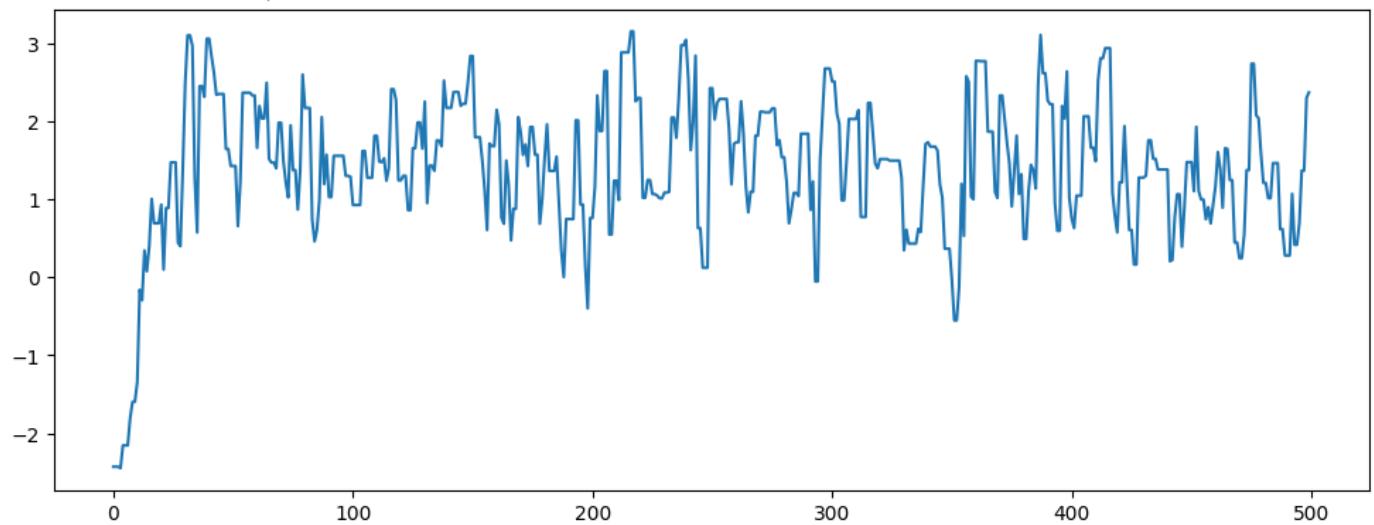
З графіку, наведеному на наступній сторінці (Рис. 3), можна помітити, що при значенні параметра $d_1 = 100$ елементи змінюються «надзвичайно грубими сходинками». При $d_2 = 1$ «сходинки» є доволі дрібними (особливо в порівнянні з попереднім випадком). Водночас при значенні $d_3 = 100$ згенеровані елементи ланцюга змінюються дуже динамічно, так що «сходинки» майже не проглядаються.

Зауважимо, що для наочності відмінностей між згенерованими ланцюгами, графіки, які відповідають значенням параметрів d_2 та d_3 продемонстровані лише на перших 500 елементах ланцюга (однак вибіркові середні та дисперсія обчислені на усьому масиві з $N = 10\,000$ ланок).

$d = 100 : \bar{X} = 3.43, S^2 = 4.72$



$d = 1 : \bar{X} = 1.47, S^2 = 0.53$



$d = 0.1 : \bar{X} = 2.12, S^2 = 0.15$

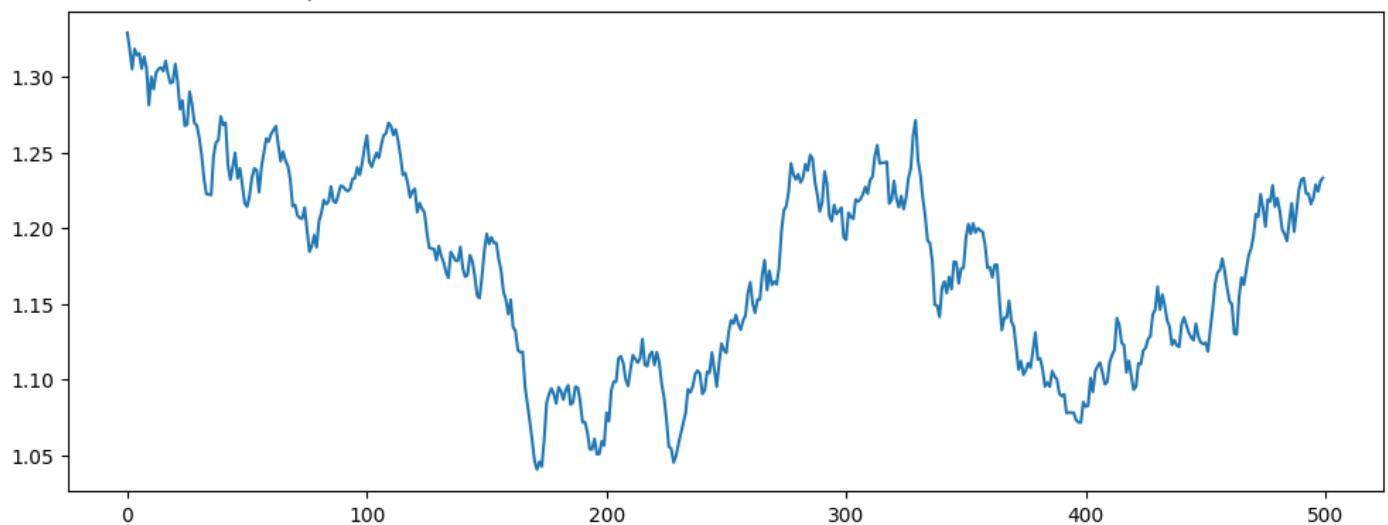


Рис. 3: Значення елементів ланцюга від ітерації до ітерації

Завдання 5

Оцінити кількість k -розфарбовок графа, що складається з 10 вершин, при кольорах $k = 3$ та $k = 4$ як кількість різних станів, спостережених за $N = 10\,000$ ітерацій.

Спершу розглянемо випадок k -розмальовок графа над такими трьома кольорами: {«green», «blue», «yellow»}. Отже, нехай початковим станом ланцюга Маркова задана така 3-розмальовка з десяти вершин:

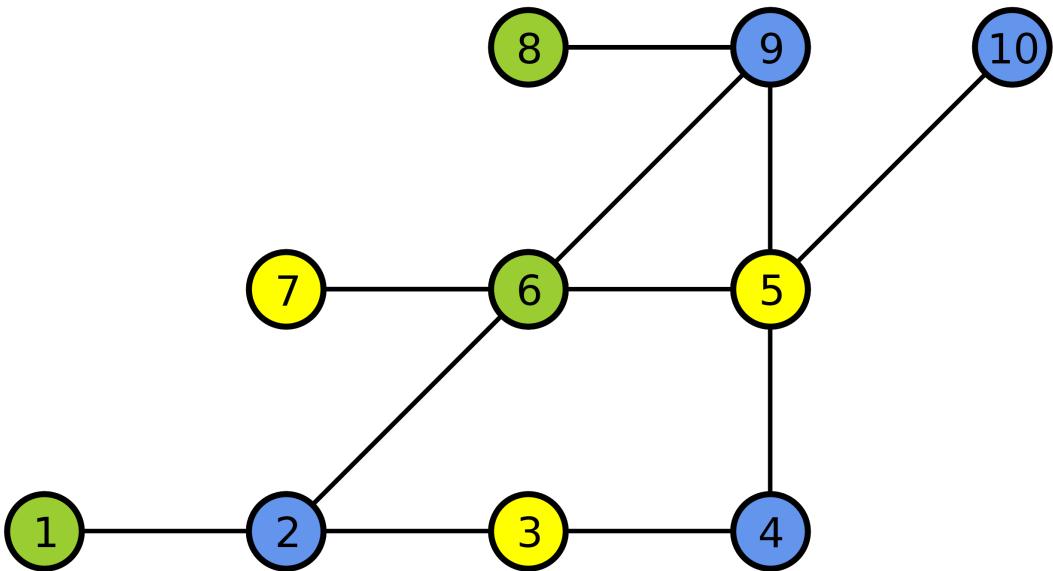


Рис. 4: Початковий стан ланцюга 3-розмальовок

При цьому основне правило розташування кольорів у графі полягає в тому, що ніякі дві сполучені ребром вершини не мають мати одинаковий колір. Побудуємо на множині всіх 3-розфарбовок графа такий ланцюг Маркова, щоб його інваріантний розподіл збігався з рівномірним розподілом на цій множині.

Тож нехай задана деяка початкова розфарбовка (Рис. 4). Алгоритм генерування наступних ланок ланцюга Маркова полягатиме у рівномірному виборі однієї з вершин графа та подальшому рівномірному виборі для обраної вершини з множини її допустимих кольорів (при фіксованих кольорах інших вершин) деякого нового кольору (включаючи можливий вибір того самого кольору, який наразі має вершина).

Продемонструємо моделювання такого ланцюга для, наприклад, $N = 9$ ланок. На графіку нижче біляожної 3-розфарбовки позначений її порядковий номер (тобто номер ітерації алгоритму) та індекс поточного стану (початкову ланку на Рис. 4 позначимо станом 0).

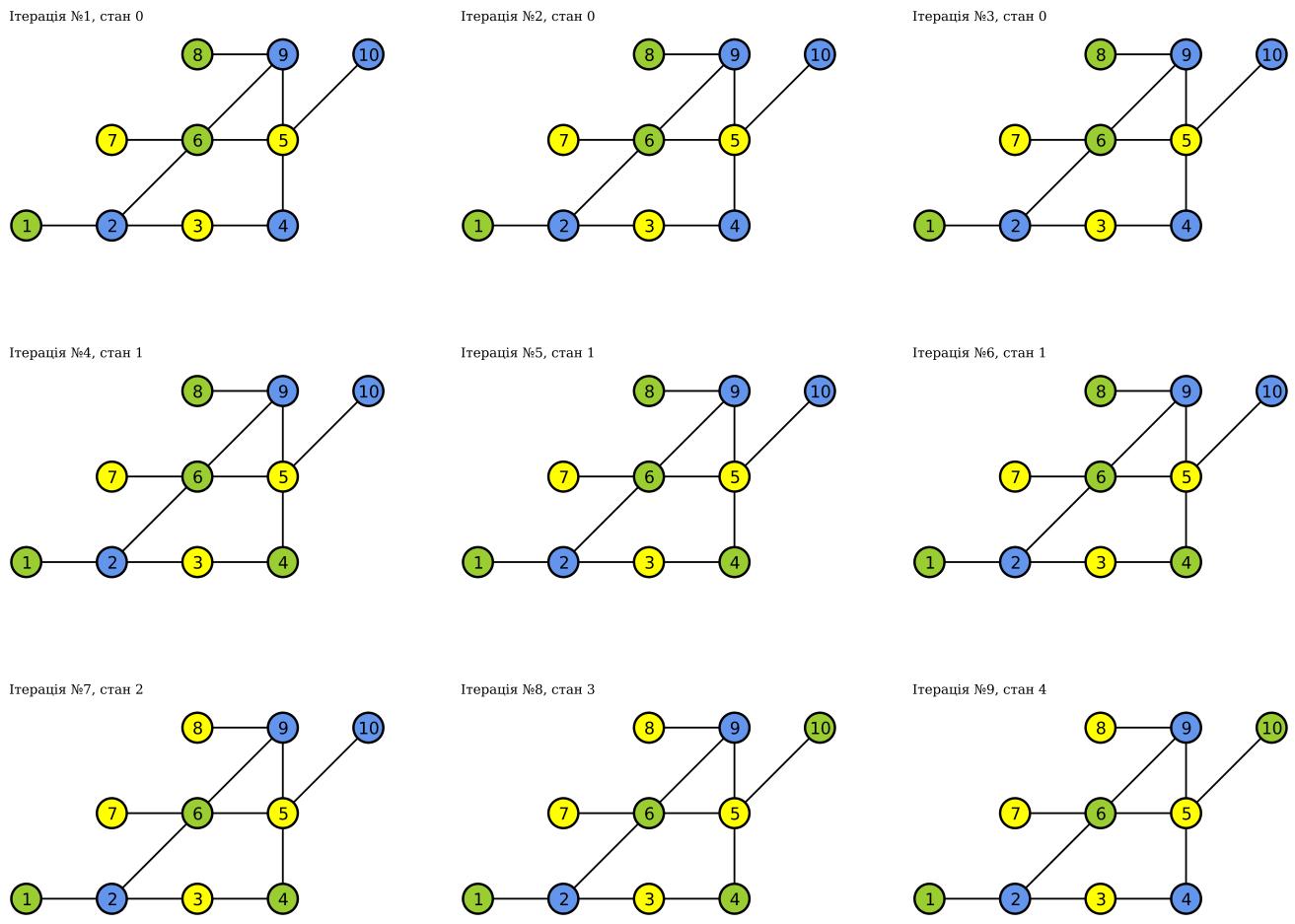


Рис. 5: Графік $N = 9$ ітерацій алгоритму МСМС

Таким чином, якщо протягом алгоритму чергова розфарбовка є новою для згенерованого ланцюга, то її присвоюється новий унікальний порядковий індекс стану. Якщо ж утворений граф повторює якусь вже наявну розфарбовку, йому присвоюється такий індекс стану, який відповідає повтореній розфарбовці.

Виконавши $N = 10\,000$ ітерацій алгоритму генерування ланцюга Маркова, побудуємо гістограму послідовності станів (бачимо, що усього налічується 128 різних 3-розфарбовок):

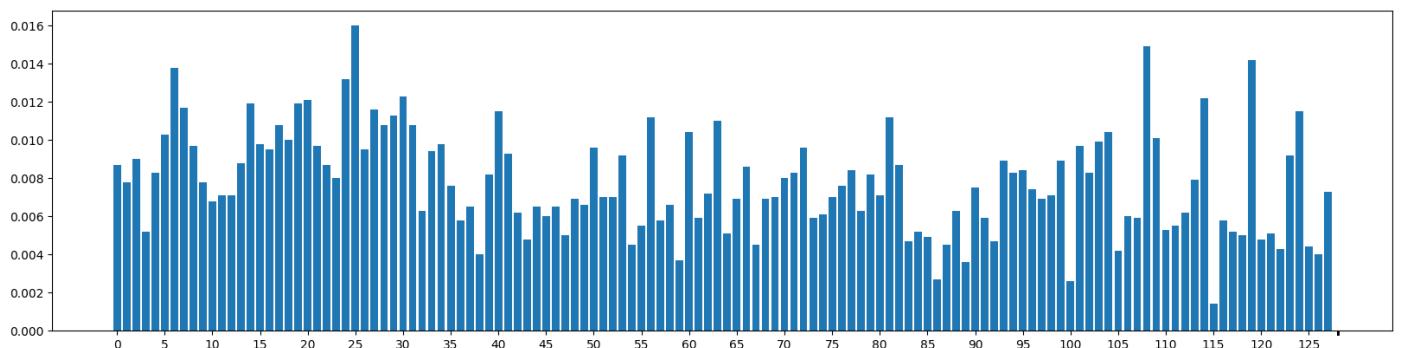


Рис. 6: Гістограма станів 3-розфарбовок

Аналогічним чином розглянемо випадок k -роздмальовки графа над чотирма кольорами: {«green», «blue», «yellow», «red»}. Нехай початковим станом ланцюга Маркова задана така 4-роздмальовка з десяти вершин:

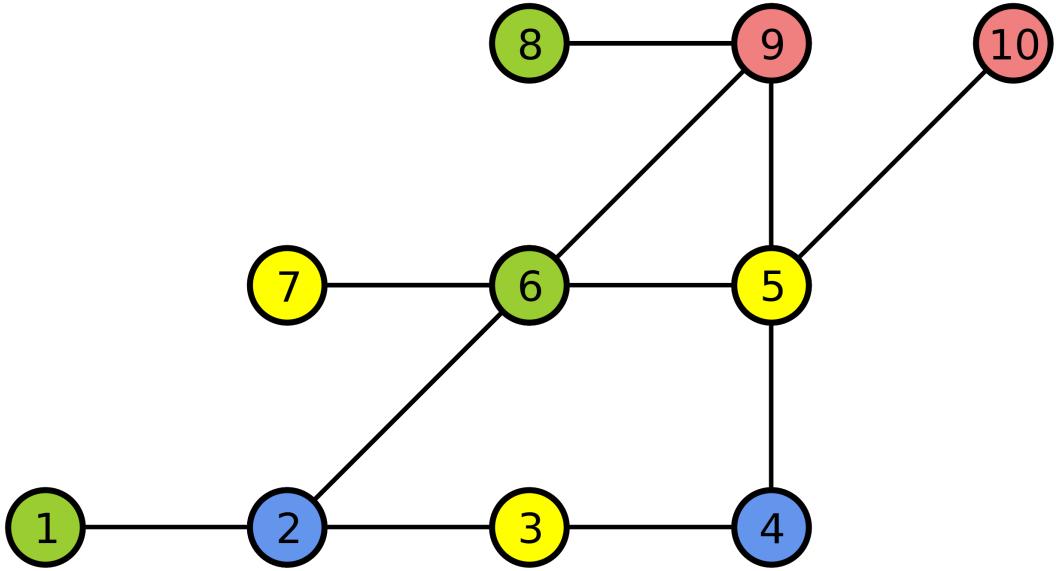


Рис. 7: Початковий стан ланцюга 4-роздмальовок

Задля наочності знову продемонструємо моделювання такого ланцюга для, наприклад, $N = 9$ ланок. На графіку (Рис. 9) біля кожної 4-роздфарбовки позначений її порядковий номер (тобто номер ітерації алгоритму) та індекс поточного стану (при цьому початкову ланку на Рис. 7 позначимо станом 0).

Виконавши $N = 10\,000$ ітерацій алгоритму генерування ланцюга Маркова, побудуємо гістограму послідовності станів. Як висновок бачимо, що усього налічується 145 різних 4-роздфарбовок:

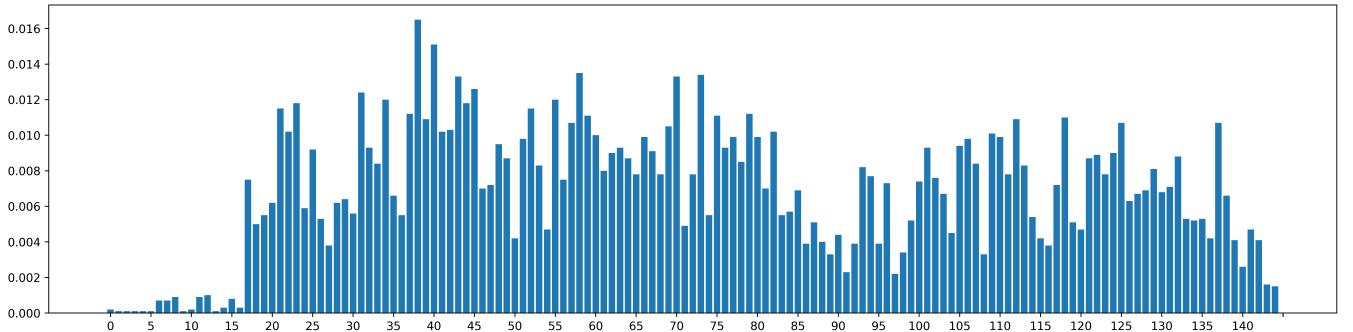
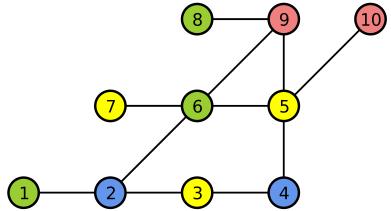
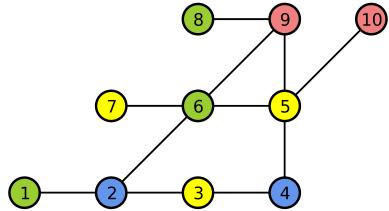


Рис. 8: Гістограма станів 4-роздфарбовок

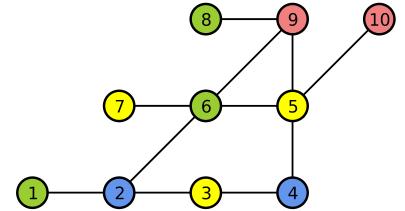
Ітерація №1, стан 0



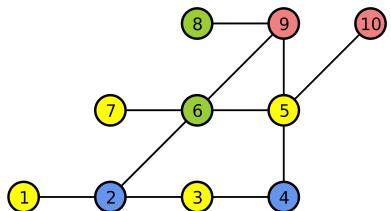
Ітерація №2, стан 0



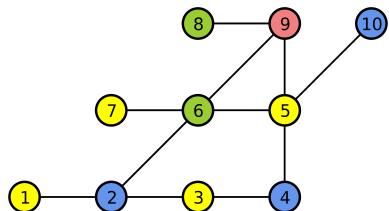
Ітерація №3, стан 0



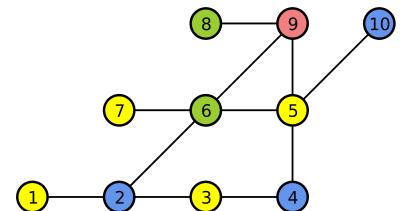
Ітерація №4, стан 1



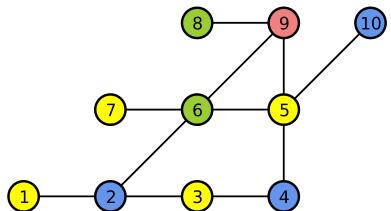
Ітерація №5, стан 2



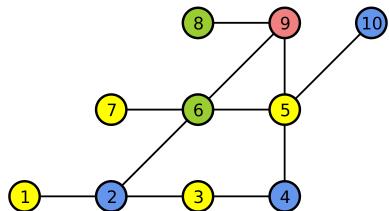
Ітерація №6, стан 2



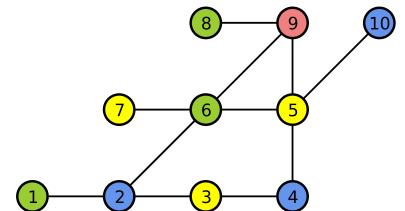
Ітерація №7, стан 2



Ітерація №8, стан 2



Ітерація №9, стан 3

Рис. 9: Графік $N = 9$ ітерацій алгоритму МСМС

Завдання 6

За допомогою алгоритму вибірки Гіббса оцінити параметри ймовірності прокльовування яйця p та кількості винесених куркою яєць n .

Вибірка Гіббса – різновид алгоритму МСМС для отримання вибірки із сумісного розподілу випадкових величин через почергний вибір з умовних розподілів: на кожному кроці одна змінна підправляється при фіксованих значеннях інших випадкових величин.

Тож при заданих параметрах $\lambda = 10$, $a = b = 1$, $x = 7$, починаючи із наближення для кількості яєць $n = 8$, алгоритм побудови вибірки Гіббса на $N = 10\,000$ ланок матиме вид:

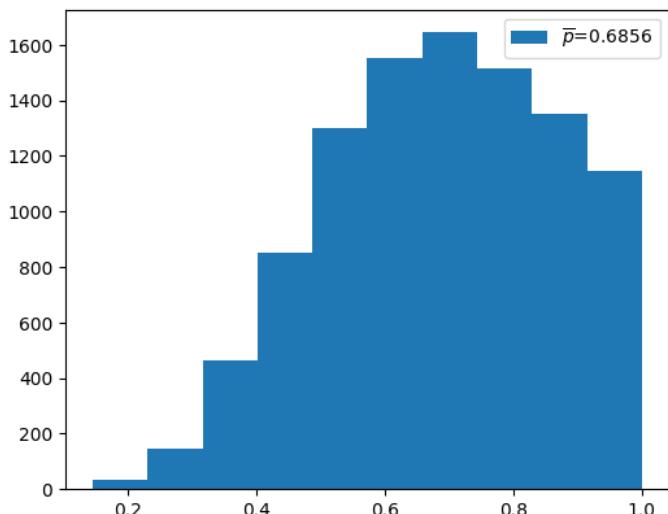
Лістинг 5: Генерування вибірки Гіббса

```
1 p, n = [], [8]
2 for i in range(N):
3     p.append(np.random.beta(x+a, n[-1]-x+b))
4     n.append(x + np.random.poisson(lambda*(1-p[-1])))
```

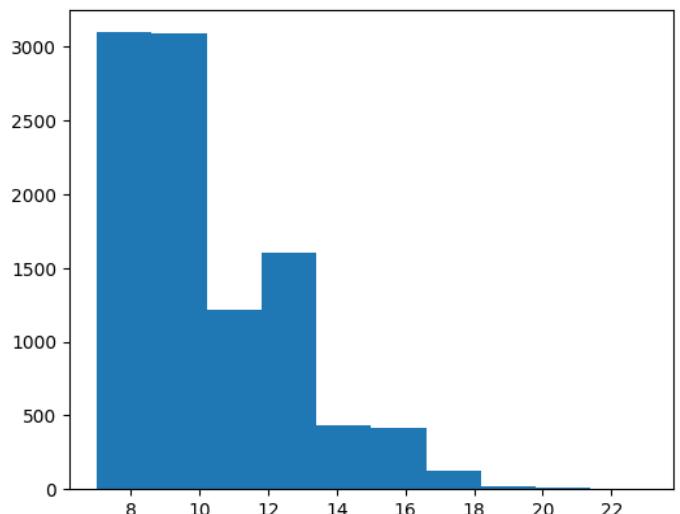
В якості оцінки для параметра ймовірності прокльовування яйця p візьмемо вибіркове середнє \bar{p} :

$$\bar{p} = 0.6856$$

Водночас, гістограми шуканих параметрів мають вид:



а) Гістограма значень p



б) Гістограма значень n

Завдання 7

Змоделювати заданий гаусовий розподіл як вибірку Гіббса.

Розглянемо такий двовимірний гаусовий вектор:

$$\begin{pmatrix} x \\ y \end{pmatrix} \sim N \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right),$$

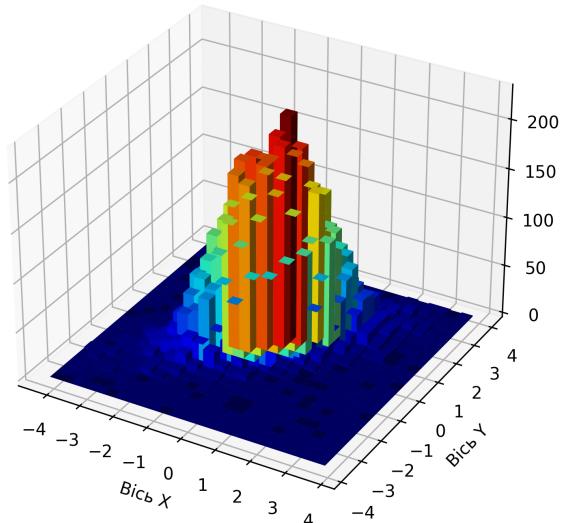
де параметр коваріаційної матриці покладемо $\rho = 0.7$. Побудуємо такий ланцюг Маркова, для якого заданий розподіл є інваріантним. Тож починаючи з точки $(x_0, y_0) = (0.05, 0.05)$, подальший алгоритм побудови ланцюга з $N = 10\,000$ елементів виглядатиме так:

Лістинг 6: Генерування вибірки Гіббса

```
1 x0, y0 = 0.05, 0.05
2 x, y = [x0], [y0]
3
4 for i in range(N):
5     x.append(np.random.normal(p*y[-1], 1-p**2))
6     y.append(np.random.normal(p*x[-1], 1-p**2))
```

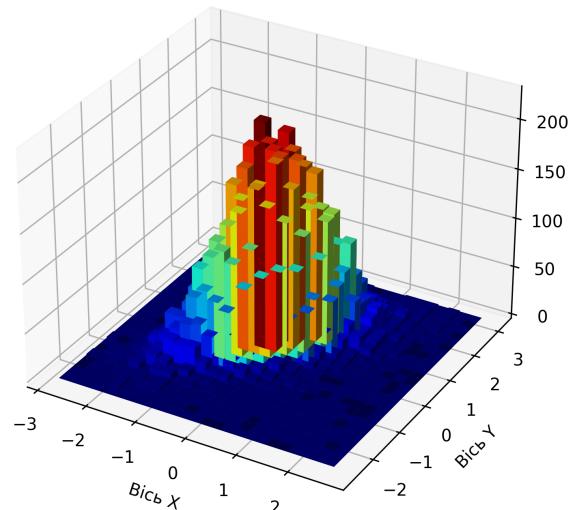
Зобразимо порівняльні тривимірні гістограми сумісних розподілів, тобто графіки частот потрапляння точок розподілу (x, y) в певну числову область. Бачимо, що графік вибірки, згенерованої засобами мови Python, доволі схожий з результатами генерації вибірки Гіббса:

`np.random.multivariate_normal(mean, cov)`



а) Генерація вбудованими методами мови Python

MCMC



б) Генерація методами згідно умов Завдання 7

Завдання 8

За допомогою MCMC реалізувати випадкове блукання на множині таблиць спряженості ознак.

Таблиця спряженості ознак T – це бінарна таблиця з фіксованим вектором \vec{r} сум по стрічках та фіксованим вектором \vec{c} сум по стовпцям. Позначимо множину $A(\vec{r}, \vec{c})$ як множину всіх таких таблиць із заданими векторами \vec{r} й \vec{c} .

Змоделюємо ланцюг Маркова на $A(\vec{r}, \vec{c})$, де $\vec{r} = (3, 2, 1)$ й $\vec{c} = (2, 2, 1, 1)$, так щоб інваріантний розподіл такого ланцюга збігався з рівномірним розподілом на $A(\vec{r}, \vec{c})$. Еволюція таблиць спряженості в процесі генерування відбудуватиметься через зміни так званих $[2 \times 2]$ -шахматок – пари стрічок та пари стовпців, на перетині яких стоїть матриця або $(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix})$, або $(\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix})$.

Аналогічно до способу виявлення множини k -розфарбовок у Завданні 5, згенеруємо $N = 10\,000$ ланок ланцюга, починаючи з таблиці $(\begin{smallmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{smallmatrix})$, й таким чином спробуємо оцінити кількість різних, унікальних таблиць спряженості ознак в множині $A(\vec{r}, \vec{c})$.

Проте, перш ніж переходити до алгоритму MCMC, продемонструємо спосіб моделювання, за якого інваріантний розподіл утвореного ланцюга виявиться нерівномірним. Схематично такий алгоритм описано на Лістингу нижче:

Лістинг 7: Генерування ланцюга Маркова методом I (не MCMC)

```
1 T0 = [[1,1,1,0],  
2      [1,1,0,0],  
3      [0,0,0,1]]  
4  
5 tables = [T0]  
6  
7 for i in range(N):  
8     T = tables[-1]  
9  
10    indexes = [k for k in range(len(T))]  
11    u = random.choice(indexes)  
12    indexes.remove(u)  
13    v = random.choice(indexes)  
14  
15    indexes = [k for k in range(len(T[0]))]  
16    i = random.choice(indexes)  
17    indexes.remove(i)  
18    j = random.choice(indexes)  
19  
20    current_checkerboard = [  
21        [T[min(u,v)][min(i,j)], T[min(u,v)][max(i,j)]],  
22        [T[max(u,v)][min(i,j)], T[max(u,v)][max(i,j)]]  
23    ]  
24  
25
```

```

26     if current_checkerboard == [[1,0],[0,1]]:
27         T[min(u,v)][min(i,j)], T[min(u,v)][max(i,j)] = 0,1
28         T[max(u,v)][min(i,j)], T[max(u,v)][max(i,j)] = 1,0
29         tables.append(T)
30
31     elif current == [[0,1],[1,0]]:
32         T[min(u,v)][min(i,j)], T[min(u,v)][max(i,j)] = 1,0
33         T[max(u,v)][min(i,j)], T[max(u,v)][max(i,j)] = 0,1
34         tables.append(T)
35
36 else:
37     tables.append(T)

```

Бачимо, що чергова таблиця може долучатися до ланцюга навіть якщо обрана $[2 \times 2]$ -матриця не є шахматкою. Таким чином, за допомогою цього алгоритму виокремлено 8 різних станів серед елементів змодельованого ланцюга. Відстеження унікальних станів виконано аналогічним чином, як це описано в завданні генерування графів k -розфарбовок. Гістограма відвідин кожного з восьми станів зображена на Рис. 10:

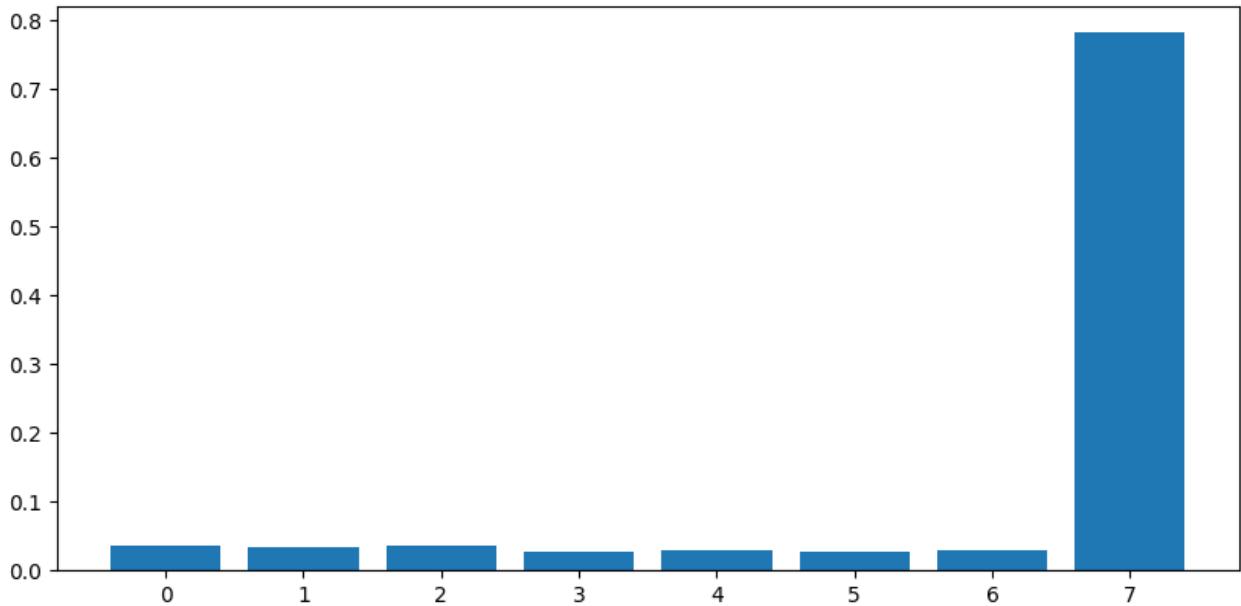


Рис. 10: Гістограма згенерованого ланцюга за методом I (не МСМС)

Бачимо, що в стані «7», який позначає таблицю $\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$, ланцюг проводить значну частину часу. Отже, попри те, що кількість різних станів ланцюга віднайдено коректно, інваріантний розподіл не є рівномірним.

У другому методі моделювання ланцюнга Маркова виникне потреба обчислення степені d_i поточної таблиці i , тобто кількості $[2 \times 2]$ -шахматок в i -тій таблиці спряжених ознак. Спосіб програматичного обчислення степені таблиці показано на Листингу нижче. В силу великої кількості вкладених циклів `for loop` (а також в силу наявного в основному коді алгоритму циклу `while loop`) наведений раніше метод I (не МСМС) виявиться швидшим.

Лістинг 8: Функція визначення степені таблиці

```

1 def table_degree(T):
2     checkerboard = [[[1,0],[0,1]],[[0,1],[1,0]]]
3
4     d = 0
5     for u in range(len(T)):
6         for v in range(u+1,len(T)):
7             for i in range(len(T[0])):
8                 for j in range(i+1,len(T[0])):
9                     current_checkerboard = [
10                         [T[u][i], T[u][j]],
11                         [T[v][i], T[v][j]]
12                     ]
13
14             if current_checkerboard in checkerboard:
15                 d += 1
16
17     return d

```

Ймовірності $\alpha_{TT'}$ прийняття/відхилення для таблиці T чергової пропозиції T' матимуть вид:

$$\alpha_{TT'} = \min \left\{ \frac{d_T}{d_{T'}}, 1 \right\},$$

де d_T та $d_{T'}$ є степенями поточної та запропонованої таблиці відповідно. Отже, враховуючи усі введені позначення, схематично алгоритм генерування МСМС можна задати таким чином:

Лістинг 9: Генерування ланцюга Маркова методом II (МСМС)

```

1 T0 = [[1,1,1,0],
2     [1,1,0,0],
3     [0,0,0,1]]
4
5 tables = [T0]
6
7 for i in range(N):
8     T = tables[-1]
9
10    while True:
11        indexes = [k for k in range(len(T))]
12        u = random.choice(indexes)
13        indexes.remove(u)
14        v = random.choice(indexes)
15
16        indexes = [k for k in range(len(T[0]))]
17        i = random.choice(indexes)
18        indexes.remove(i)
19        j = random.choice(indexes)
20
21        current_checkerboard = [[T[min(u,v)][min(i,j)], T[min(u,v)][max(i,j)]],
22                                [T[max(u,v)][min(i,j)], T[max(u,v)][max(i,j)]]]

```

```

24     if current_checkerboard in [[[0,1],[1,0]], [[1,0],[0,1]]]:
25         break
26
27     if current_checkerboard == [[1,0],[0,1]]:
28         T[min(u,v)][min(i,j)], T[min(u,v)][max(i,j)] = 0,1
29         T[max(u,v)][min(i,j)], T[max(u,v)][max(i,j)] = 1,0
30
31     alpha = min(table_degree(tables[-1])/table_degree(T), 1)
32
33     u = random.uniform(0,1)
34     if u <= alpha:
35         tables.append(T)
36     else:
37         tables.append(tables[-1])
38
39     elif current_checkerboard == [[0,1],[1,0]]:
40         T[min(u,v)][min(i,j)], T[min(u,v)][max(i,j)] = 1,0
41         T[max(u,v)][min(i,j)], T[max(u,v)][max(i,j)] = 0,1
42
43     alpha = min(table_degree(tables[-1])/table_degree(T), 1)
44
45     u = random.uniform(0,1)
46     if u <= alpha:
47         tables.append(T)
48     else:
49         tables.append(tables[-1])

```

У такому разі згенерований ланцюг відвідуватиме 8 різних станів рівноміно, при чому кожна з компонент інваріантного розподілу буде мати достаньо близьке до $\frac{1}{8}$ значення:

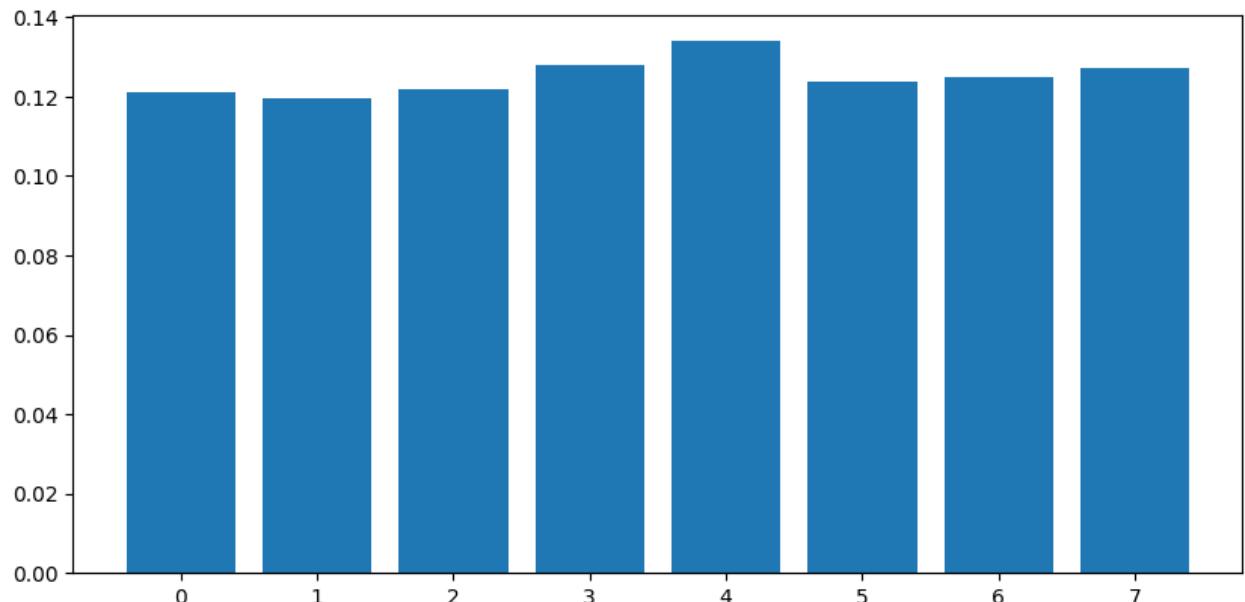


Рис. 11: Гістограма згенерованого ланцюга за методом II (МСМС)