НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Лабораторна робота №3

«Розрахунок характеристик ланцюгів Маркова та їх моделювання»

з кредитного модуля «Випадкові процеси»

Варіант 11

Виконав:
студент групи КМ-93
Костенко Олександр Андрійович
Викладач:
Пашко Анатолій Олексійович

Київ — 2021

Зміст

Завдання 1

Завдання 1. Для поглинаючого ланцюга Маркова знайти:

- матрицю переходів;
- фундаментальну матрицю;
- середню кількість кроків, яку ланцюг знаходиться в стані j коли процес почався зі стану n;
- середню кількість кроків, яку ланцюг знаходиться в стані j коли початковий стан не заданий;
- середній час поглинання;
- ймовірність поглинання.

11	0,18	0,23	0,09	0,12	0,12	0,29	0,22	0,15	0,01	0,0	0,3	0,1	0,6

1.1

```
[[1.   0.   0.   0.  ]
 [0.18 0.23 0.5  0.09]
 [0.12 0.12 0.29 0.47]
 [0.22 0.62 0.15 0.01]]
```

1.2

```
Фундаментальна матриця:
[[2.628 2.113 1.242]
 [1.705 2.936 1.549]
 [1.904 1.768 2.023]]
```

1.5

```
Середній час поглинання:
[[5.983]
 [6.19 ]
 [5.695]]
```

1.6

```
Ймовірність поглинання:
[[0.99984]
 [1.     ]
 [0.99994]]
```

Код програми:

```
#1
import numpy as np
from matplotlib import pyplot as plt
```

```
from colorsys import hls_to_rgb



p0=[0.0,0.3,0.1,0.6]
p=[1,0,0,0,0.18,0.23,0.5,0.09,0.12,0.12,0.29,0.47,0.22,0.62,0.15,0.01]
shape=(4,4)
matrix=np.array(p).reshape(shape)
print(matrix)

#2
I=[]
O=[]
R=[]
Q=[]
I=matrix[0][0]
O=matrix[0][1:4]
for i in range(1,4):
    R.append(matrix[i][0])
    Q.append(matrix[i][1])
    Q.append(matrix[i][2])
    Q.append(matrix[i][3])
R=np.array(R).reshape((3,1))
Q=np.array(Q).reshape((3,3))
E=np.array([1,0,0,0,1,0,0,0,1]).reshape((3,3))
IminusQ=np.subtract(E,Q)
print(IminusQ)
fundmatrix=np.linalg.matrix_power(IminusQ,-1)
fundmatrix=np.matrix.round(fundmatrix, 3)
print("Фундаментальна матриця:")
print(fundmatrix)

#3

#4

#5
m1=[[1],[1],[1]]
time=np.matmul(fundmatrix,m1)
print("Середній час поглинання:")
print(time)

#6
B=np.matmul(fundmatrix,R)
print("Ймовірність поглинання:")
print(B)
```

Завдання 2

Завдання 2. Для регулярного ланцюга Маркова знайти:

- матрицю переходів;
- фінальні ймовірності;
- фундаментальну матрицю;
- середній час перебування в заданому стані за $n = 4$ кроків;
- середній час виходу ланцюга в заданий стан (в стан J, коли процес почався зі стану n);
- середній час виходу ланцюга в заданий стан (коли початковий стан не заданий);
- середній час виходу ланцюга в заданий стан в стаціонарному режимі (коли початковий стан не заданий).

11	0,40	0,33	0,13	0,08	0,27	0,32	0,04	0,42	0,54

2.1

```
Матриця переходів:
[[0.4  0.33 0.27]
 [0.79 0.13 0.08]
 [0.27 0.41 0.32]]
```

2.2

```
Матриця фінальних ймовірностей:
[[0.4837258 0.2900796 0.2261946]
 [0.4837258 0.2900796 0.2261946]
 [0.4837258 0.2900796 0.2261946]]
```

2.3

```
Фундаментальна матриця:
[[ 0.92519874  0.03586289  0.03893837]
 [ 0.267304    0.85823131 -0.12553532]
 [-0.1828345   0.10511497  1.07771953]]
```

2.5

```
Матриця середньої кількості кроків:
[[2.06728688 2.83497502 4.59242247]
 [1.36005716 3.44732963 5.31955603]
 [2.29062258 2.59624029 4.42097203]]
```

```
Введіть стан, у який здіснюється перехід (нумерація починається з 1): 1
Введіть стан, з який здіснюється перехід (нумерація починається з 1): 3
Середній час виходу ланцюга в заданий стан:
2.290622582043856
```

2.6

```
Середній час виходу ланцюга в заданий стан (коли початковий стан не задано):
[2.06728688029458, 1.3600571580885397, 2.290622582043856]
```

2.7

```
Середній час виходу ланцюга в заданий стан в стаціонарному режимі:
[2.06728688029458, 0.0, 0.0]
```

Код програми:

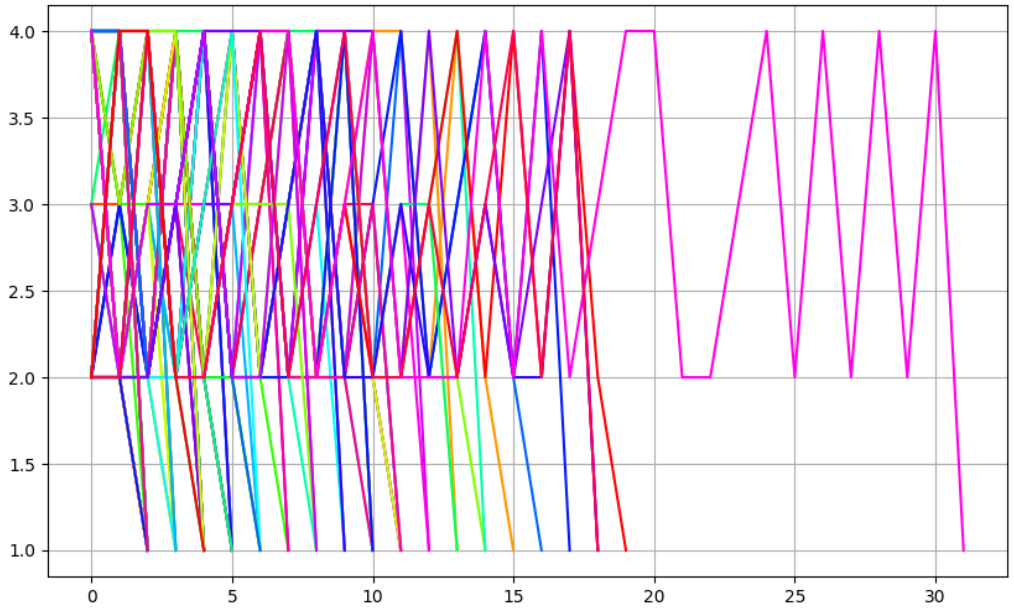
```
#1
p=np.array([0.4,0.33,0.27,0.79,0.13,0.08,0.27,0.41,0.32]).reshape((3,3))
p0=[0.04,0.42,0.54]

print("Матриця переходів:")
print(p)

#2
wcurrent=np.around(np.matmul(p0, np.linalg.matrix_power(p,1)),7)

for i in range(2,15):

        wnext=np.around(np.matmul(p0, np.linalg.matrix_power(p,i)), 7)
        isequal=(wcurrent==wnext).all()
        if isequal==True:
            wline=wnext
            k=i-1
            break

        wcurrent=wnext

W=[]
for i in range(3):
    W.append(wline)

W=np.array(W).reshape((3,3))
print("Матриця фінальних ймовірностей:")
print(W)

#3
I=np.array([1,0,0,0,1,0,0,0,1]).reshape((3,3))
Z=np.linalg.matrix_power(np.subtract(I,np.subtract(p,W)),-1)
print("Фундаментальна матриця:")
print(Z)

#5
E=np.array([1,1,1,1,1,1,1,1,1]).reshape((3,3))
D=[]
Zdg=[]
for i in range(9):
```

```
    D.append(0.0)
    Zdg.append(0.0)
D=np.array(D).reshape((3,3))
Zdg=np.array(Zdg).reshape((3,3))

for i in range(3):
    D[i][i]=1/wline[i]
    Zdg[i][i]=Z[i][i]


M=np.matmul(np.add(np.subtract(I,Z),np.matmul(E,Zdg)),D)
print("Матриця середньої кількості кроків:")
print(M)

j=int(input("Введіть стан, у який здіснюється перехід (нумерація
починається з 1): "))
n=int(input("Введіть стан, з який здіснюється перехід (нумерація
починається з 1): "))
m=M[n-1][j-1]
print("Середній час виходу ланцюга в заданий стан:")
print(m)

#6
m1=[]
for i in range(3):
    m1.append(M[i][j-1])
print("Середній час виходу ланцюга в заданий стан (коли початковий стан
не задано):")
print(m1)

#7
Wdg=[]
for i in range(9):
    Wdg.append(0.0)

Wdg=np.array(Wdg).reshape((3,3))

for i in range(3):
    Wdg[i][i]=W[i][i]

Mw=np.matmul(np.add(np.subtract(I,W),np.matmul(E,Wdg)),D)
mw=[]
for i in range(3):
    mw.append(Mw[i][j-1])
print("Середній час виходу ланцюга в заданий стан в стаціонарному
режимі:")
```

```
print(mw)
```

Завдання 3

Завдання 3. Змоделювати ланцюг Маркова з поглинанням для заданих перехідних і початкових ймовірностей (варіант із завдання 1, довжина реалізації – до поглинання). Кількість реалізацій – більше 100.



Код програми:

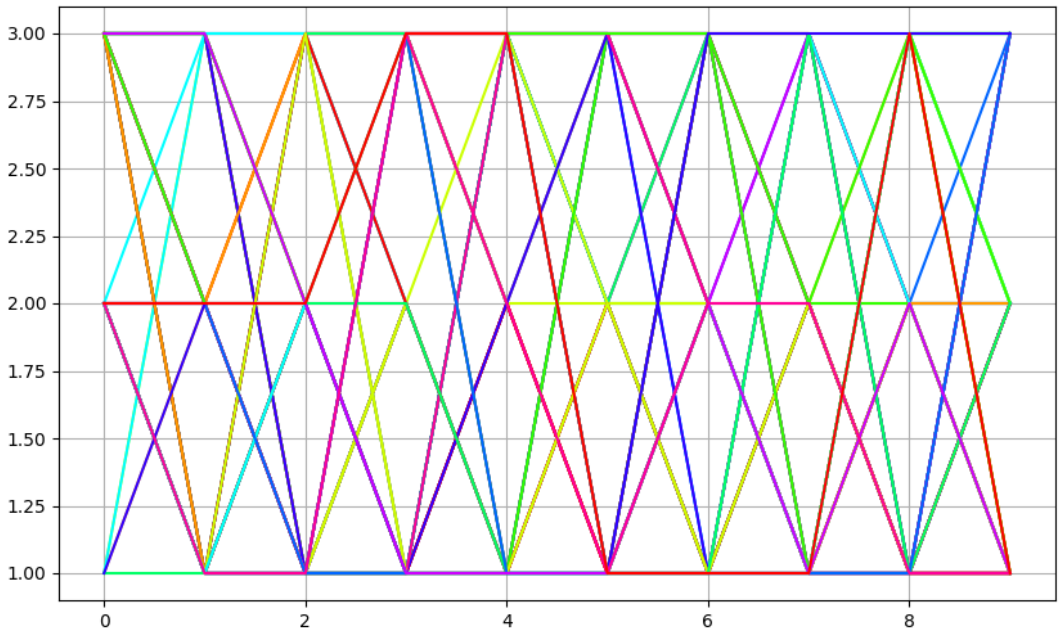
```
p=np.array([1,0,0,0,0.18,0.23,0.5,0.09,0.12,0.12,0.29,0.47,0.22,0.62,0.15
,0.01]).reshape((4,4))
state=[]
time=[]
currentRow=p[0]


for i in range(100):
    state.append([])

def first(state,i):
    x=np.random.uniform(0.0,1.0)
    if x<p0[1]:
        k=2
        currentRow=p[1]
    elif x>=p0[1] and x<p0[2]+p0[1]:
        k=3
```

```
        currentRow=p[2]
    else:
        k=4
        currentRow=p[3]

    state[i].append(k)
    result=[]
    result.append(k)
    result.append(currentRow)
    return result

for i in range(100):
    k=first(state,i)[0]
    currentRow=first(state,i)[1]
    while k!=1:
        x=np.random.uniform(0.0,1.0)
        if x<currentRow[0]:
            k=1
            state[i].append(k)
            break
        elif x>=currentRow[0] and x<currentRow[1]+currentRow[0]:
            k=2
            currentRow=p[1]
        elif x>=currentRow[1] and x< currentRow[2]+currentRow[1]:
            k=3
            currentRow=p[2]
        else:
            k=4
            currentRow=p[3]
        state[i].append(k)



hue=0
fig=plt.subplots(figsize=(10,6))
plt.grid()

for i in range(len(state)):
    time=[]
    for j in range(len(state[i])):
        time.append(j)
    hue+=0.05
    color=hls_to_rgb(hue, 0.5, 1)
    plt.plot(time, state[i], color=color)
```

```
plt.show()
```

Завдання 4

Завдання 4. Змоделювати регулярний ланцюг Маркова для заданих перехідних і початкових ймовірностей (варіант із завдання 2, довжина реалізації – не менше 10). Кількість реалізацій – більше 100.



Код програми:

```
state=[]
fig, ax=plt.subplots(figsize=(10,6))
hue=0
luminosity= 0.5
saturation=1
time=[]
for i in range(10):
    time.append(i)


for i in range(100):
    hue+=1/10
    color=hls_to_rgb(hue, luminosity, saturation)
    state.append([])
    currentRow=p0
    for j in range(10):
        x=np.random.uniform(0.0,1.0)
        if x>=0 and x<currentRow[0]:
```

```
            k=1
            currentRow=p[0]
        elif x>=currentRow[0] and x<(currentRow[1]+currentRow[0]):
            k=2
            currentRow=p[1]
        else:
            k=3
            currentRow=p[2]
        state[i].append(k)
    ax.plot(list(time),list(state[i]), color=color, label=str(i+1)+"-ий
ряд")

plt.grid()
plt.show()
```