НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені Ігоря Сікорського»

Кафедра прикладної математики

Звіт

із лабораторної роботи №*2*

із дисципліни «Випадкові процеси»

|  |  |  |
| --- | --- | --- |
| Виконав: | | Керівник: |
| студент групи КМ-63 | | *Пашко Анатолій Олексійович* |
| *Артеменко Я.К.* |  | |

Київ — 2017

**Тема:** Моделювання ланцюгів Маркова

**Завдання:**

1. Змоделювати ланцюг Маркова для заданих перехідних і

початкових ймовірностей:

а) при вибраному початковому стані;

б) при випадковому початковому стані.

2. Змоделювати ланцюг Маркова з поглинанням для заданих

перехідних і початкових ймовірностей:

а) при вибраному початковому стані;

б) при випадковому початковому стані.

3. Знайти і порівняти теоретичні і експериментальні

характеристики: час перебування в заданому стані марковського ланцюга,

час поглинання і ймовірність поглинання:

а) при вибраному початковому стані;

б) при випадковому початковому стані.

4. Змоделювати регулярний ланцюг Маркова для заданих

перехідних і початкових ймовірностей:

а) при вибраному початковому стані;

б) при випадковому початковому стані.

5. Знайти і порівняти теоретичні і експериментальні

характеристики: час перебування в заданому стані марковського ланцюга, та

інші характеристики:

а) при вибраному початковому стані;

б) при випадковому початковому стані.

**Код програми та результати його виконання**

1. Змоделювати ланцюг Маркова для заданих перехідних і

початкових ймовірностей:

а) при вибраному початковому стані;

б) при випадковому початковому стані.

get\_state = function(pn, p){

F = pn[1]

i = 1

while(F < p){

i = i + 1

F = F + pn[i]

}

i

}

n = 5

p01 = c(0, 0, 1, 0, 0)

p02 = c(0.1, 0.2, 0.3, 0.3, 0.1)

m = 10

states1 = c()

states2 = c()

P = matrix(data = c(1/n), nrow = n, ncol = n)

P[1, ] = c(1, 0, 0, 0, 0)

ps1 = runif(m + 1)

ps2 = runif(m + 1)

pn1 = c()

pn2 = c()

states1[1] = get\_state(p01, ps1[1])

for (i in 1:(m - 1)){

states1[i + 1] = get\_state(P[states1[i], ], ps1[i + 1])

}

states2[1] = get\_state(p02, ps2[1])

for (i in 1:(m - 1)){

pn = p02\*P

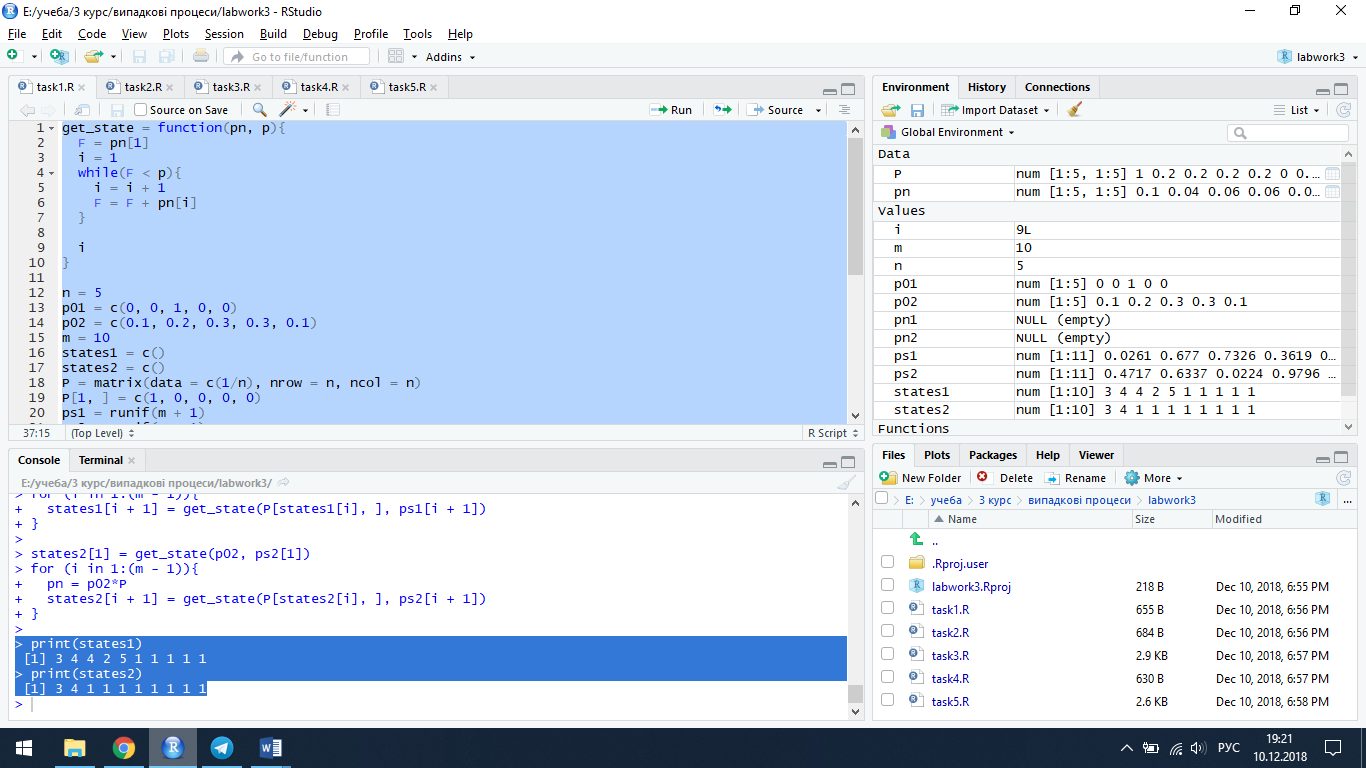
states2[i + 1] = get\_state(P[states2[i], ], ps2[i + 1])

}

print(states1)

print(states2)

**Результат**



2. Змоделювати ланцюг Маркова з поглинанням для заданих

перехідних і початкових ймовірностей:

а) при вибраному початковому стані;

б) при випадковому початковому стані.

get\_state = function(pn, p){

F = pn[1]

i = 1

while(F < p){

i = i + 1

F = F + pn[i]

}

i

}

n = 5

p01 = c(0, 0, 1, 0, 0)

p02 = c(0.1, 0.2, 0.3, 0.3, 0.1)

m = 10

states1 = c()

states2 = c()

P = matrix(data = c(1/n), nrow = n, ncol = n)

P[1, ] = c(0, 1, 0, 0, 0)

P[2, ] = c(1, 0, 0, 0, 0)

ps1 = runif(m + 1)

ps2 = runif(m + 1)

pn1 = c()

pn2 = c()

states1[1] = get\_state(p01, ps1[1])

for (i in 1:(m - 1)){

states1[i + 1] = get\_state(P[states1[i], ], ps1[i + 1])

}

states2[1] = get\_state(p02, ps2[1])

for (i in 1:(m - 1)){

pn = p02\*P

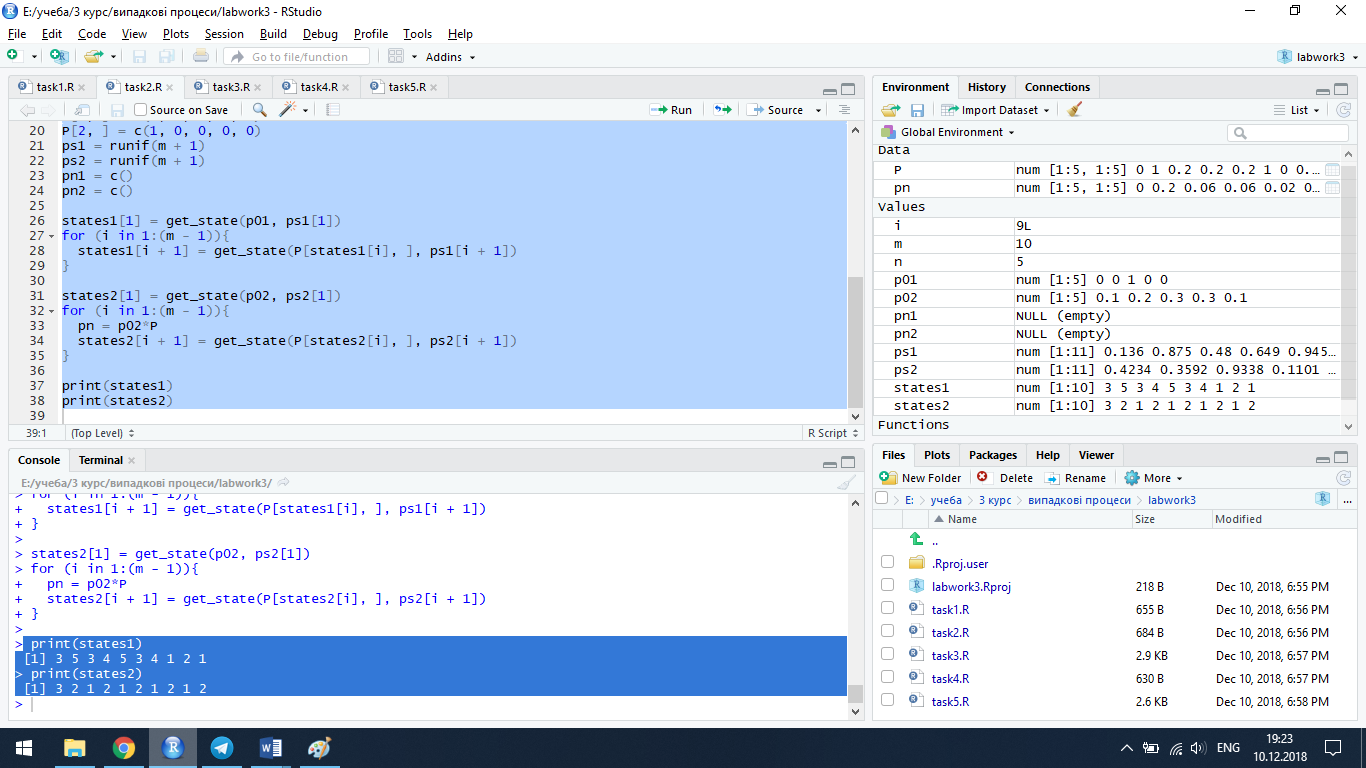
states2[i + 1] = get\_state(P[states2[i], ], ps2[i + 1])

}

print(states1)

print(states2)

**Результат**



3. Знайти і порівняти теоретичні і експериментальні

характеристики: час перебування в заданому стані марковського ланцюга,

час поглинання і ймовірність поглинання:

а) при вибраному початковому стані;

б) при випадковому початковому стані.

get\_state = function(pn, p){

F = pn[1]

i = 1

while(F < p){

i = i + 1

F = F + pn[i]

}

i

}

standartize\_matrix = function(matr){

size = length(matr[, 1])

I\_size = 0

temp\_vector = c()

for (i in 1:size){

if(matr[i, i] == 1){

I\_size = I\_size + 1

temp\_vector = matr[, I\_size]

matr[, I\_size] = matr[, i]

matr[, i] = temp\_vector

temp\_vector = matr[I\_size, ]

matr[I\_size, ] = matr[i, ]

matr[i, ] = temp\_vector

}

}

matr

}

get\_I\_size = function(stan\_matr){

I\_size = 0

size\_matr = length(stan\_matr[1, ])

for (i in 1:size\_matr){

if(stan\_matr[i, i] == 1){

I\_size = I\_size + 1

}

}

I\_size

}

eye = function(dimens){

matr = diag(1, dimens)

matr

}

n = 5

p01 = c(0, 0, 1, 0, 0)

p02 = c(0.1, 0.2, 0.3, 0.3, 0.1)

m = 100

states1 = c()

states2 = c()

P = matrix(data = c(1/n), nrow = n, ncol = n)

P[3, ] = c(0, 0, 1, 0, 0)

P[5, ] = c(0, 0, 0, 0, 1)

ps1 = runif(m + 1)

ps2 = runif(m + 1)

pn1 = c()

pn2 = c()

P = standartize\_matrix(P)

P\_size = length(P[, 1])

I\_size = get\_I\_size(P)

I = eye(I\_size)

R = P[(I\_size + 1):P\_size, 1:I\_size]

Q = P[(I\_size + 1):P\_size, (I\_size + 1):P\_size]

M = solve(eye(P\_size - I\_size) - Q)

pi\_vector1 = p01[(I\_size + 1):P\_size]

tau\_vector1 = pi\_vector1%\*%M

B = M%\*%R

t\_absorp1 = sum(tau\_vector1)

tau\_vector\_exp1 = rep(0, P\_size - I\_size)

t\_absorp\_exp1 = 0

states1[1] = get\_state(p01, ps1[1])

print(states1[1])

for (i in 1:(m - 1)){

states1[i + 1] = get\_state(P[states1[i], ], ps1[i + 1])

if(states1[i] > I\_size){

tau\_vector\_exp1[states1[i] - I\_size] = tau\_vector\_exp1[states1[i] - I\_size] + 1

}

}

t\_absorp\_exp1 = sum(tau\_vector\_exp1)

pi\_vector2 = p02[(I\_size + 1):P\_size]

tau\_vector2 = pi\_vector2%\*%M

t\_absorp2 = sum(tau\_vector2)

tau\_vector\_exp2 = rep(0, P\_size - I\_size)

t\_absorp\_exp2 = 0

states2[1] = get\_state(p02, ps2[1])

for (i in 1:(m - 1)){

states2[i + 1] = get\_state(P[states2[i], ], ps2[i + 1])

if(states1[i] > I\_size){

tau\_vector\_exp2[states1[i] - I\_size] = tau\_vector\_exp2[states1[i] - I\_size] + 1

}

}

t\_absorp\_exp2 = sum(tau\_vector\_exp2)

print("Вірогідність поглинання:")

print(B)

print("Точний стан:")

print("Середня тривалість перебування в відповідних станах:")

print(tau\_vector1)

print("Середній час поглинання:")

print(t\_absorp1)

print("Середня тривалість перебування у відповідних станах (експериментальні):")

print(tau\_vector\_exp1)

print("Середній час поглинання(експериментальний):")

print(t\_absorp\_exp1)

print("Нестабільний стан:")

print("Середня тривалість перебування в відповідних станах:")

print(tau\_vector2)

print("Середній час поглинання:")

print(t\_absorp2)

print("Середня тривалість перебування у відповідних станах (експериментальні):")

print(tau\_vector\_exp2)

print("Середній час поглинання(експериментальний):")

print(t\_absorp\_exp2)

**Результат:**

> print("Вірогідність поглинання:")

[1] "Вірогідність поглинання:"

> print(B)

[,1] [,2]

[1,] 0.5 0.5

[2,] 0.5 0.5

[3,] 0.5 0.5

> print("Точний стан:")

[1] "Точний стан:"

> print("Середня тривалість перебування в відповідних станах:")

[1] "Середня тривалість перебування в відповідних станах:"

> print(tau\_vector1)

[,1] [,2] [,3]

[1,] 1.5 0.5 0.5

> print("Середній час поглинання:")

[1] "Середній час поглинання:"

> print(t\_absorp1)

[1] 2.5

> print("Середня тривалість перебування у відповідних станах (експериментальні):")

[1] "Середня тривалість перебування у відповідних станах (експериментальні):"

> print(tau\_vector\_exp1)

[1] 1 1 0

> print("Середній час поглинання(експериментальний):")

[1] "Середній час поглинання(експериментальний):"

> print(t\_absorp\_exp1)

[1] 2

> print("Нестабільний стан:")

[1] "Нестабільний стан:"

> print("Середня тривалість перебування в відповідних станах:")

[1] "Середня тривалість перебування в відповідних станах:"

> print(tau\_vector2)

[,1] [,2] [,3]

[1,] 0.65 0.65 0.45

> print("Середній час поглинання:")

[1] "Середній час поглинання:"

> print(t\_absorp2)

[1] 1.75

> print("Середня тривалість перебування у відповідних станах (експериментальні):")

[1] "Середня тривалість перебування у відповідних станах (експериментальні):"

> print(tau\_vector\_exp2)

[1] 1 1 0

> print("Середній час поглинання(експериментальний):")

[1] "Середній час поглинання(експериментальний):"

> print(t\_absorp\_exp2)

[1] 2

4. Змоделювати регулярний ланцюг Маркова для заданих

перехідних і початкових ймовірностей:

а) при вибраному початковому стані;

б) при випадковому початковому стані.

get\_state = function(pn, p){

F = pn[1]

i = 1

while(F < p){

i = i + 1

F = F + pn[i]

}

i

}

n = 5

p01 = c(0, 0, 1, 0, 0)

p02 = c(0.1, 0.2, 0.3, 0.3, 0.1)

m = 10

states1 = c()

states2 = c()

P = matrix(data = c(1/n), nrow = n, ncol = n)

ps1 = runif(m + 1)

ps2 = runif(m + 1)

pn1 = c()

pn2 = c()

states1[1] = get\_state(p01, ps1[1])

for (i in 1:(m - 1)){

states1[i + 1] = get\_state(P[states1[i], ], ps1[i + 1])

}

states2[1] = get\_state(p02, ps2[1])

for (i in 1:(m - 1)){

pn = p02\*P

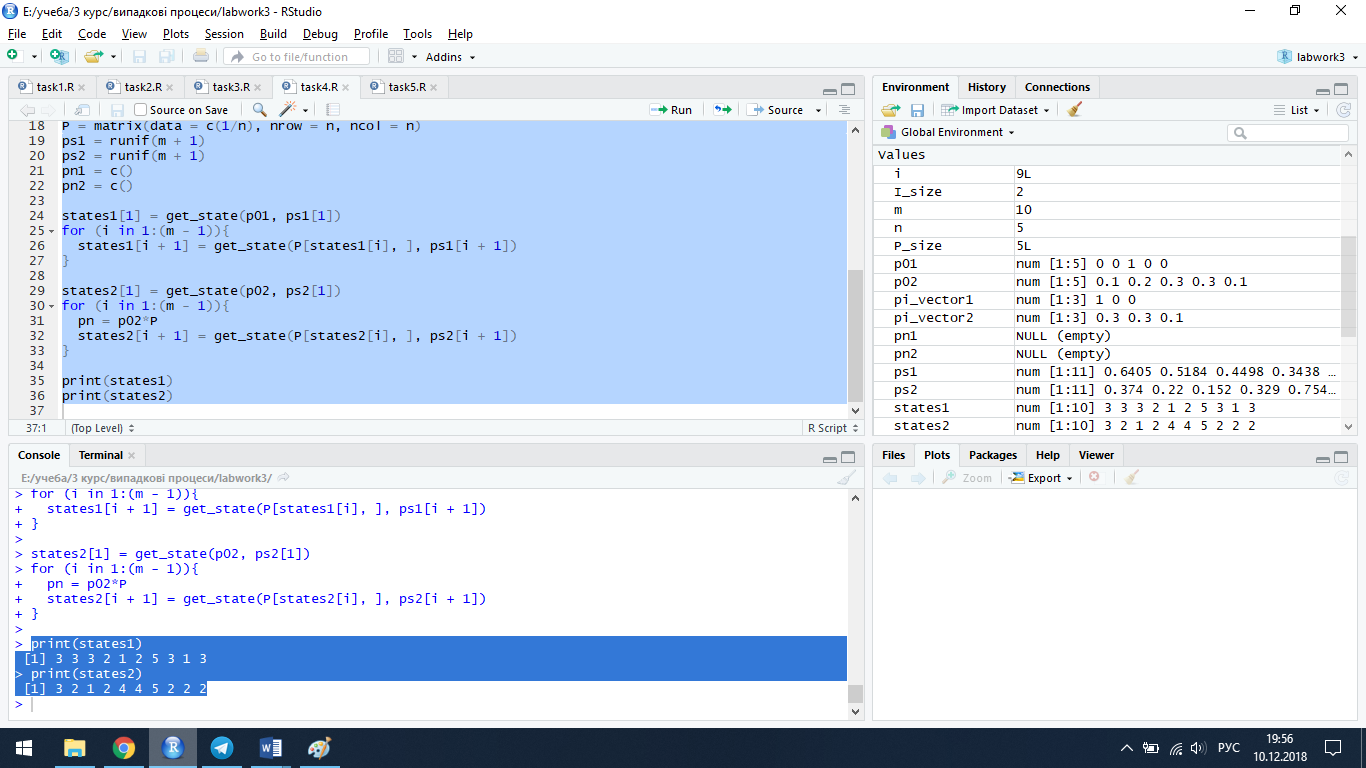
states2[i + 1] = get\_state(P[states2[i], ], ps2[i + 1])

}

print(states1)

print(states2)

**Результат**



5. Знайти і порівняти теоретичні і експериментальні

характеристики: час перебування в заданому стані марковського ланцюга, та

інші характеристики:

а) при вибраному початковому стані;

б) при випадковому початковому стані.

get\_state = function(pn, p){

F = pn[1]

i = 1

while(F < p){

i = i + 1

F = F + pn[i]

}

i

}

matr\_pow = function(matr, pow){

for (i in 2:pow){

matr = round(matr%\*%matr, 8)

}

matr

}

eye = function(dimens){

matr = diag(1, dimens)

matr

}

get\_V = function(P, m){

P\_size = length(P[1, ])

V = matrix(data = c(0), nrow = P\_size, ncol = P\_size)

P\_pow = eye(P\_size)

for (i in 1:(m - 1)){

P\_pow = P\_pow%\*%P

V = V + P\_pow

}

V

}

get\_M = function(Z, f){

size = length(f)

I = eye(size)

E = matrix(data = c(1), nrow = size, ncol = size)

DZ = diag(diag(Z), size)

DM = diag(1/f, size)

M = (I - Z + E%\*%DZ)%\*%DM

M

}

n = 5

p01 = c(0, 0, 1, 0, 0)

p02 = c(0.1, 0.2, 0.3, 0.3, 0.1)

m = 100

states1 = c()

states2 = c()

P = matrix(data = c(1/n), nrow = n, ncol = n)

ps1 = runif(m + 1)

ps2 = runif(m + 1)

pn1 = c()

pn2 = c()

P\_size = length(P[1, ])

F = matr\_pow(P, 100)

f = F[1, ]

Z = solve(eye(P\_size) - (P - F))

V = get\_V(P, m)

M = get\_M(Z, f)

tn1 = p01%\*%V

t1 = p01%\*%M

states1[1] = get\_state(p01, ps1[1])

tn\_exp1 = rep(0, P\_size)

t\_exp1 = rep(0, P\_size)

for (i in 1:(m - 1)){

states1[i + 1] = get\_state(P[states1[i], ], ps1[i + 1])

tn\_exp1[states1[i]] = tn\_exp1[states1[i]] + 1

if(t\_exp1[states1[i]] == 0){

t\_exp1[states1[i]] = i

}

}

tn2 = p02%\*%V

t2 = p02%\*%M

states2[1] = get\_state(p02, ps2[1])

tn\_exp2 = rep(0, P\_size)

t\_exp2 = rep(0, P\_size)

for (i in 1:(m - 1)){

pn = p02\*P

states2[i + 1] = get\_state(P[states2[i], ], ps2[i + 1])

tn\_exp2[states2[i]] = tn\_exp2[states2[i]] + 1

if(t\_exp2[states2[i]] == 0){

t\_exp2[states2[i]] = i

}

}

print("Фінальна матриця F:")

print(F)

print("Z:")

print(Z)

print("Середня тривалість переміщення між відповідними станами:")

print(V)

print("Середні моменти першого трансферу між відповідними станами:")

print(M)

print("Точний стан:")

print("Середня тривалість перебування в відповідних станах:")

print(tn1)

print("Середні моменти першого прибуття в відповідні стани:")

print(t1)

print("Середня тривалість перебування в відповідних станах (експериментальна):")

print(tn\_exp1)

print("Середні моменти першого прибуття в відповідні стани (експериментальні):")

print(t\_exp1)

print("Нестабільний стан:")

print("Середня тривалість перебування в відповідних станах:")

print(tn2)

print("Середні моменти першого прибуття в відповідні стани:")

print(t2)

print("Середня тривалість перебування в відповідних станах (експериментальна):")

print(tn\_exp2)

print("Середні моменти першого прибуття в відповідні стани (експериментальні):")

print(t\_exp2)

**Результат**

|  |
| --- |
| > print("Фінальна матриця F:")  [1] "Фінальна матриця F:"  > print(F)  [,1] [,2] [,3] [,4] [,5]  [1,] 0.2 0.2 0.2 0.2 0.2  [2,] 0.2 0.2 0.2 0.2 0.2  [3,] 0.2 0.2 0.2 0.2 0.2  [4,] 0.2 0.2 0.2 0.2 0.2  [5,] 0.2 0.2 0.2 0.2 0.2  > print("Z:")  [1] "Z:"  > print(Z)  [,1] [,2] [,3] [,4] [,5]  [1,] 1 0 0 0 0  [2,] 0 1 0 0 0  [3,] 0 0 1 0 0  [4,] 0 0 0 1 0  [5,] 0 0 0 0 1  > print("Середня тривалість переміщення між відповідними станами:")  [1] "Середня тривалість переміщення між відповідними станами:"  > print(V)  [,1] [,2] [,3] [,4] [,5]  [1,] 19.8 19.8 19.8 19.8 19.8  [2,] 19.8 19.8 19.8 19.8 19.8  [3,] 19.8 19.8 19.8 19.8 19.8  [4,] 19.8 19.8 19.8 19.8 19.8  [5,] 19.8 19.8 19.8 19.8 19.8  > print("Середні моменти першого трансферу між відповідними станами:")  [1] "Середні моменти першого трансферу між відповідними станами:"  > print(M)  [,1] [,2] [,3] [,4] [,5]  [1,] 5 5 5 5 5  [2,] 5 5 5 5 5  [3,] 5 5 5 5 5  [4,] 5 5 5 5 5  [5,] 5 5 5 5 5  >  > print("Точний стан:")  [1] "Точний стан:"  > print("Середня тривалість перебування в відповідних станах:")  [1] "Середня тривалість перебування в відповідних станах:"  > print(tn1)  [,1] [,2] [,3] [,4] [,5]  [1,] 19.8 19.8 19.8 19.8 19.8  > print("Середні моменти першого прибуття в відповідні стани:")  [1] "Середні моменти першого прибуття в відповідні стани:"  > print(t1)  [,1] [,2] [,3] [,4] [,5]  [1,] 5 5 5 5 5  > print("Середня тривалість перебування в відповідних станах (експериментальна):")  [1] "Середня тривалість перебування в відповідних станах (експериментальна):"  > print(tn\_exp1)  [1] 27 21 19 17 15  > print("Середні моменти першого прибуття в відповідні стани (експериментальні):")  [1] "Середні моменти першого прибуття в відповідні стани (експериментальні):"  > print(t\_exp1)  [1] 6 7 1 8 2  >  >  > print("Нестабільний стан:")  [1] "Нестабільний стан:"  > print("Середня тривалість перебування в відповідних станах:")  [1] "Середня тривалість перебування в відповідних станах:"  > print(tn2)  [,1] [,2] [,3] [,4] [,5]  [1,] 19.8 19.8 19.8 19.8 19.8  > print("Середні моменти першого прибуття в відповідні стани:")  [1] "Середні моменти першого прибуття в відповідні стани:"  > print(t2)  [,1] [,2] [,3] [,4] [,5]  [1,] 5 5 5 5 5  > print("Середня тривалість перебування в відповідних станах (експериментальна):")  [1] "Середня тривалість перебування в відповідних станах (експериментальна):"  > print(tn\_exp2)  [1] 21 21 15 22 20  > print("Середні моменти першого прибуття в відповідні стани (експериментальні):")  [1] "Середні моменти першого прибуття в відповідні стани (експериментальні):"  > print(t\_exp2)  [1] 3 6 2 1 5 |
|  |
| |  | | --- | |  | |

**Використані функції:**

с() – створення вектора;

matrix (data, nrow, ncol) – створення матриці;

runif – генерація чисел рівномірного розподілу;

print() – виведення в консоль;

sum – сума всіх елементів вектора;

length – довжина об’єкту;

round – округлення;

rep – повторює значення в х;

solve – отримання розв’язку матричного рівняння AX = B;

eye – створює основні матриці;

diag – отримання діагоналі матриці або створення діагональної матриці;