1830

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе №2 по курсу "Моделирование"

Тема Марковские цепи
Студент <u>Леонов В.В.</u>
Группа ИУ7-76Б
Оценка (баллы)
Преподаватель Рудаков И.В.

Формальная постановка задачи

Реализовать программное обеспечение для определения вероятности и времени пребывания системы в каждом из состоянии в установившемся режиме работы СМО.

Исходные данные:

- количество состояний системы (до 10);
- матрица интенсивностей переходов из состояния в состояние.

Краткие теоритические сведения

Случайный процесс, протекающий в сложной системе S, называется **марковским**, если он обладает следующим свойством: для каждого момента времени t_0 вероятность любого состояния системы в будущем при $t > t_0$ зависит только от состояния системы в настоящем $t = t_0$ и не зависит от того, когда и каким образом система перешла в это состояние (как процесс развивался в прошлом).

В марковском случайном процессе будущее развитие зависит только от настоящего состояния и не зависит от предыстории процесса.

Для марковского процесса составлены уравнения Колмогорова:

$$F = (P'(t), P(t), \lambda) = 0$$

Вероятностью i-го состояния называется вероятность $P_i(t)$ того, что в момент времени t система будет находится в состоянии S_i . Для любого момента t сумма вероятностей всех состояний равно единице (правило нормировки).

Для нахождения предельных вероятностей используется система уравнений, определяемая следующим образом:

• в левой части каждого из уравнений стоит производная вероятности i-го состояния;

• в правой части - сумма произведений вероятностей всех состояний (из которых идут стрелки в данное состояние), умноженная на интенсивности соответствующих потоков событий, минус суммарная интенсивность всех потоков, выводящих систему из данного состояния, умноженная на вероятность данного i-го состояния.

После нахождения вероятностей, необходимо вычислить время пребывания системы в каждом из состояний. Для этого необходимо с заданным интервалом dt вычислять приращение вероятности для i-го состояния. Вычисления завершаются, когда найденная вероятность будет равна соответствующей предельной с точностью до заданной погрешности.

Для приращения вероятности dp необходимо задать начальные значения, например, 1/n, где n — число состояний системы.

Средства реализации

Язык программирования — Python. $\mathrm{GUI} - \mathrm{QT}.$

Листинг кода

```
from numpy import linalg
2 import random
4 TIME DELTA = 1e-3
5 \text{ EPS} = 1e-5
6
   def random mtr(size: int) -> list[list[float]]:
8
       mtr = [[round(random.random(), 2) * random.randint(0, 1)
9
                for _ in range(size) | for _ in range(size) |
10
       return mtr
11
12
   def get coef mtr(mtr: list[list[float]]) -> list[list[float]]:
14
       size = len(mtr)
15
       coef mtr = [[0.0 for in range(size)] for in range(size)]
16
17
       for i in range (size):
18
            for j in range(size):
19
                if (i = j):
20
                    coef_mtr[i][i] = -sum(mtr[i]) + mtr[i][i]
21
                else:
22
                    coef mtr[i][j] = mtr[j][i]
23
       return coef mtr
24
25
26
   def calc_prob(mtr: list[list[float]]) -> list[float]:
27
       size = len(mtr)
28
       coef mtr = get coef mtr(mtr)
29
       coef_mtr[size - 1] = [1 for _ in range(size)]
30
       ordinate values = [0 if i != size - 1 else 1 for i in range(size)]
31
       return linalg.solve(coef mtr, ordinate values).tolist()
32
33
   def calc prob delta(mtr: list[list[float]], prob curr: list[float]):
34
35
       size = len(mtr)
36
37
       prob delta = []
38
       coef mtr = get coef mtr(mtr)
39
       for i in range(size):
40
41
            for j in range(size):
42
                coef_mtr[i][j] *= prob_curr[j]
            prob_delta.append(sum(coef_mtr[i]) * TIME_DELTA)
43
44
       return prob delta
```

```
def calc_time(mtr: list[list[float]], prob: list[float]):
 2
       size = len(mtr)
 3
       time curr = 0.0
 4
       prob_curr = [1.0 / size for _ in range(size)]
 5
 6
 7
       time = [0.0 \text{ for } \_in \text{ range}(size)]
 8
       while not all(time):
9
           prob_delta = calc_prob_delta(mtr, prob_curr)
            for i in range(size):
10
                if not time[i] and abs(prob_curr[i] - prob[i]) <= EPS:
11
12
                    time[i] = time\_curr
                prob curr[i] += prob delta[i]
13
            14
15
16
       return time
17
18
   def calc_res(mtr: list[list[float]]) -> tuple[list[float], list[float]]:
19
20
       prob = calc_prob(mtr)
21
       time = calc_time(mtr, prob)
       return prob, time
22
```

Демонстрация работы программы

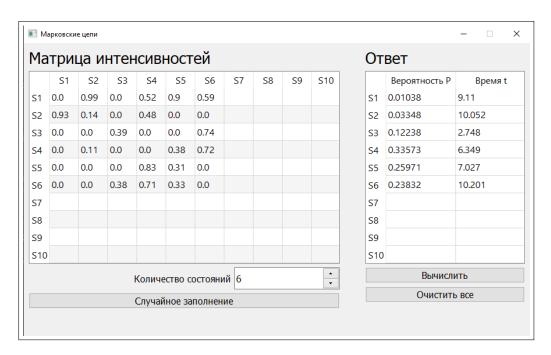


Рисунок 1 – Работа программы для системы с 6 состояниями

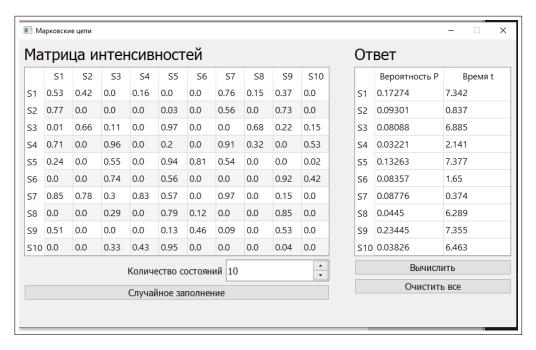


Рисунок 2 – Работа программы для системы с 10 состояниями

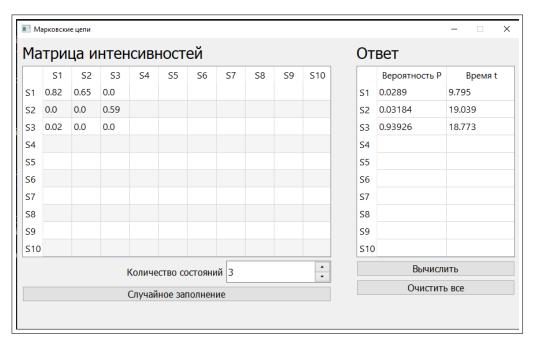


Рисунок 3 – Работа программы для системы с 3 состояниями