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»

Отчет по лабораторной работе №2 по дисциплине «Моделирование»

Тема Марковские цепи

Студент Слепокурова М.Ф.

Группа ИУ7-76Б

Оценка (баллы) ___________

Преподаватель Рудаков И.В.

Москва — 2023 г.

Постановка задачи

Определить вероятность и время пребывания системы в каждом состоянии в установившемся режиме работы СМО. Исходные данные: кол-во состояний системы (max 10) и матрица интенсивностей переходов из состояния в состояние.

Теория

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 находиться в состоянии S_i. Для любого момента t сумма вероятностей всех состояний равна единице.

Для нахождения предельных вероятностей используется система уравнений вида:

$$\begin{cases} p_0' = \lambda_{10}p_1 + \lambda_{20}p_2 - (\lambda_{01} + \lambda_{02})p_0, \\ p_1' = \lambda_{01}p_0 + \lambda_{31}p_3 - (\lambda_{10} + \lambda_{13})p_1, \\ p_2' = \lambda_{02}p_0 + \lambda_{32}p_3 - (\lambda_{20} + \lambda_{23})p_2, \\ p_3' = \lambda_{13}p_1 + \lambda_{23}p_2 - (\lambda_{31} + \lambda_{32})p_3. \end{cases}$$

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.

Так как предельные вероятности постоянны, то, заменяя в уравнениях Колмогорова их производные нулевыми значениями, получим систему линейных алгебраических уравнений, описывающих стационарный режим при $t \to \infty$. Для решения полученной системы необходимо добавить условие нормировки $(p_0 + p_1 + p_2 + p_3)$ вместо одного из уравнений.

После нахождения вероятностей, необходимо вычислить время пребывания системы в каждом из состояний. Для этого необходимо с заданным интервалом δt вычислять приращение вероятности для i-го состояния по формуле вида:

$$dp_0 = (\lambda_{10}p_1 + \lambda_{20}p_2 - (\lambda_{01} + \lambda_{02})p_0) * \delta t,$$

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

Для реализации приложения был выбран язык программирования Python, в стандартную библиотеку которого входит графическая библиотека Tkinter, использовавшаяся для реализации пользовательского интерфейса, а также библиотека numpy, использовавшаяся для решения системы уравнений методом Крамера.

Листинг кода

```
from numpy import linalg

TIME_DELTA = 1e-3
EPS = 1e-5

def __getCoefMatrix(matrix):
  count = len(matrix)
  coefMatrix = [[0.0 for j in range(count)] for i in range(count)]

  for i in range(count):
    for j in range(count):
      if (i == j): coefMatrix[i][i] = -sum(matrix[i]) + matrix[i][i]
      else: coefMatrix[i][j] = matrix[j][i]
  return coefMatrix

def calculateProbability(matrix):
  count = len(matrix)
  coefMatrix = __getCoefMatrix(matrix)
  coefMatrix[count - 1] = [1 for j in range(count)]

  ordinateValues = [0 if i != count - 1 else 1 for i in range(count)]
  return linalg.solve(coefMatrix, ordinateValues).tolist()

def __calculateProbDelta(matrix, probCurr):
  count = len(matrix)
  probDelta = []
  coefMatrix = __getCoefMatrix(matrix)
  for i in range(count):
    for j in range(count):
      coefMatrix[i][j] *= probCurr[j]
    probDelta.append(sum(coefMatrix[i]) * TIME_DELTA)
  return probDelta
```

```
def calculateTime(matrix, prob):
  count = len(matrix)
  timeCurr = 0.0
  probCurr = [1.0 / count for i in range(count)]
  time = [0.0 for i in range(count)]

  while not all(time):
    probDelta = __calculateProbDelta(matrix, probCurr)
    for i in range(count):
      if not time[i] and abs(probCurr[i] - prob[i]) <= EPS:
        time[i] = timeCurr
      probCurr[i] += probDelta[i]
    timeCurr += TIME_DELTA
  return time
```

Демонстрация работы программы

На рисунке 1 изображен пример работы программы для системы с 5 состояниями.

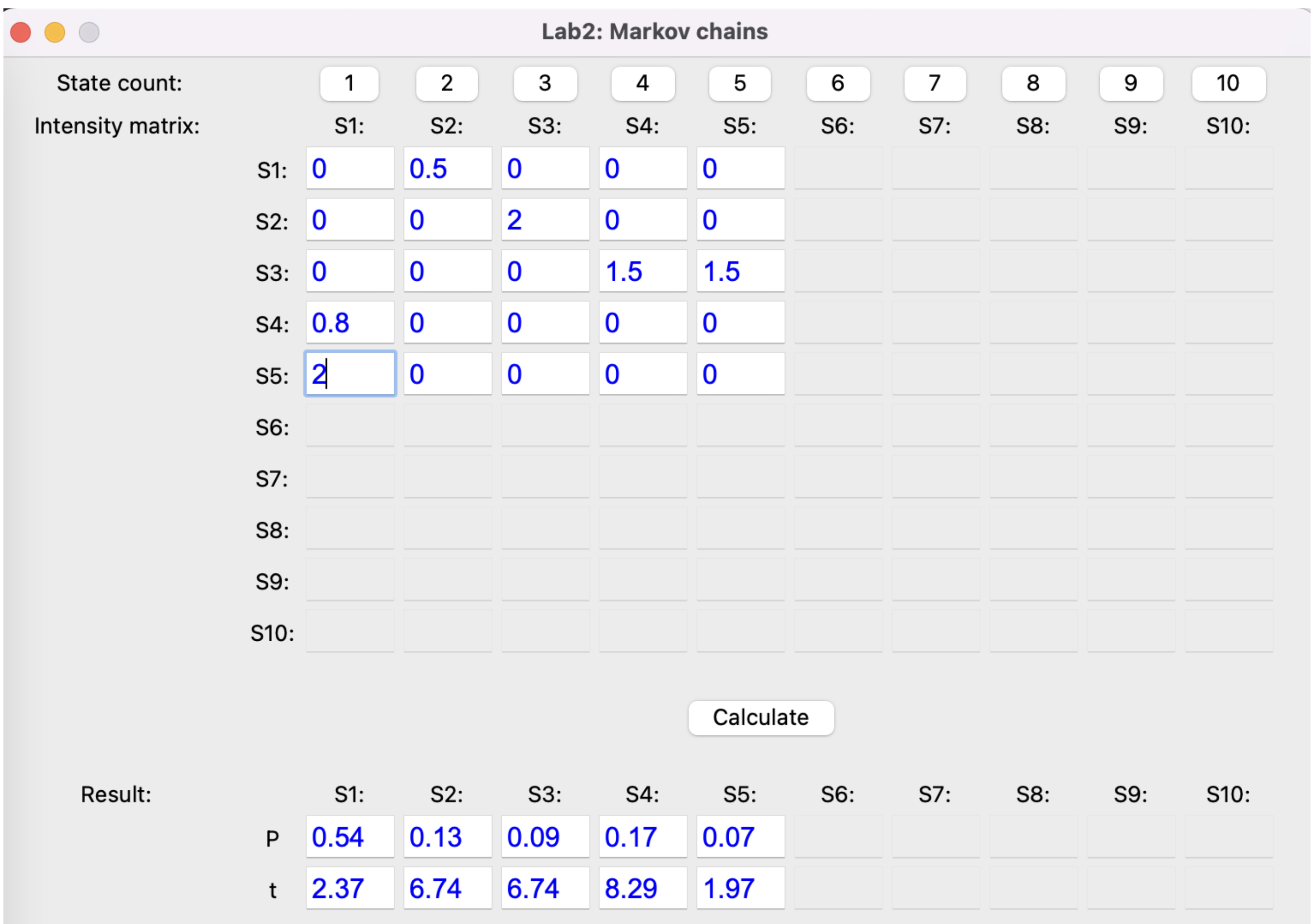


Рисунок 1 – Пример работы программы — 1

На рисунке 1 изображен пример работы программы для системы с 2 состояниями.

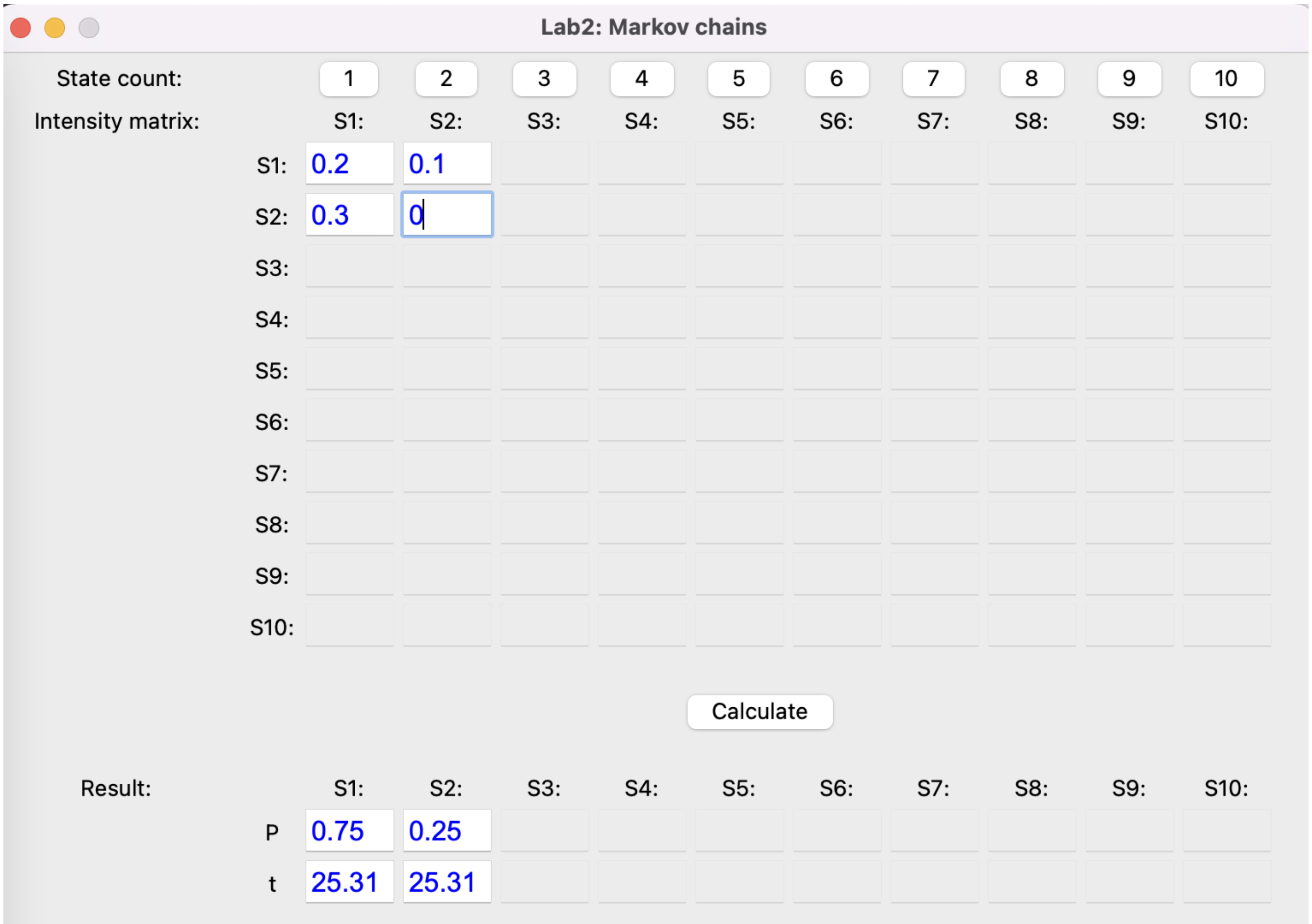
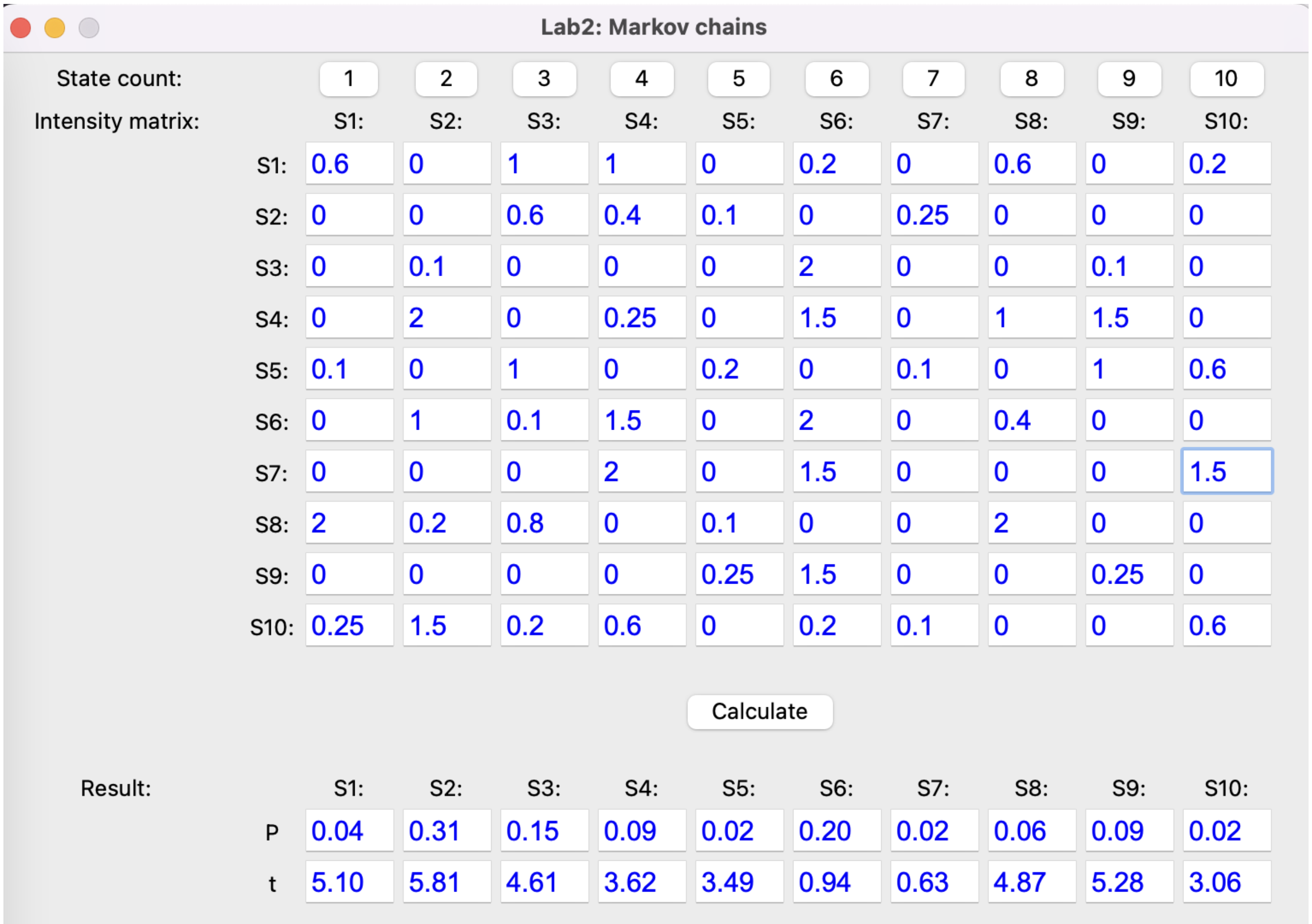


Рисунок 2 – Пример работы программы — 2

На рисунке 1 изображен пример работы программы для системы с 10 состояниями.



Lab2: Markov chains

State count: 1 2 3 4 5 6 7 8 9 10

Intensity matrix:

	S1:	S2:	S3:	S4:	S5:	S6:	S7:	S8:	S9:	S10:
S1:	0.6	0	1	1	0	0.2	0	0.6	0	0.2
S2:	0	0	0.6	0.4	0.1	0	0.25	0	0	0
S3:	0	0.1	0	0	0	2	0	0	0.1	0
S4:	0	2	0	0.25	0	1.5	0	1	1.5	0
S5:	0.1	0	1	0	0.2	0	0.1	0	1	0.6
S6:	0	1	0.1	1.5	0	2	0	0.4	0	0
S7:	0	0	0	2	0	1.5	0	0	0	1.5
S8:	2	0.2	0.8	0	0.1	0	0	2	0	0
S9:	0	0	0	0	0.25	1.5	0	0	0.25	0
S10:	0.25	1.5	0.2	0.6	0	0.2	0.1	0	0	0.6

Calculate

Result:

	S1:	S2:	S3:	S4:	S5:	S6:	S7:	S8:	S9:	S10:
P	0.04	0.31	0.15	0.09	0.02	0.20	0.02	0.06	0.09	0.02
t	5.10	5.81	4.61	3.62	3.49	0.94	0.63	4.87	5.28	3.06

Рисунок 3 – Пример работы программы — 3