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

Название: Марковские процессы

Дисциплина: Моделирование

Студент ИУ7-75Б _____ Д.В. Сусликов

(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ И.В. Рудаков

(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Содержание

Задание

Для сложной системы S, имеющей не более 10 состояний определить среднее время нахождения системы в предельных состояниях, то есть при установившемся режиме работы.

Система вводится матрицей, на пересечение строк и столбцов интенсивность перехода.

Также требуется отобразить решение графически.

Теория

Случайный процесс, протекающий в системе S, называется марковским, если он обладает следующим свойством: для каждого момента времени t_0 вероятность любого состояния системы в будущем (при $t > t0$) зависит только от ее состояния в настоящем (при $t = t_0$) и не зависит от того, когда и каким образом система пришла в это состояние. Вероятностью i-го состояния называется вероятность p_i того, что в момент t система будет находиться в состоянии . Для любого момента t сумма вероятностей всех состояний равна единице.

Для решения поставленной задачи, необходимо составить систему уравнений Колмогорова по следующим принципам: 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).

Примеры

Пример 1

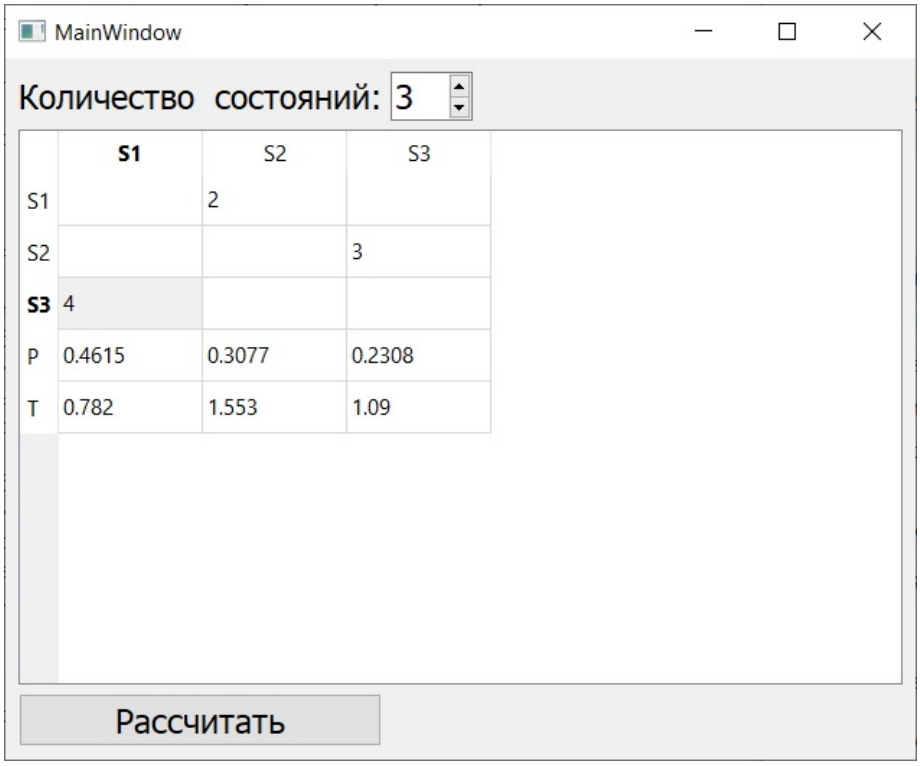


Рисунок 1 – Главное окно

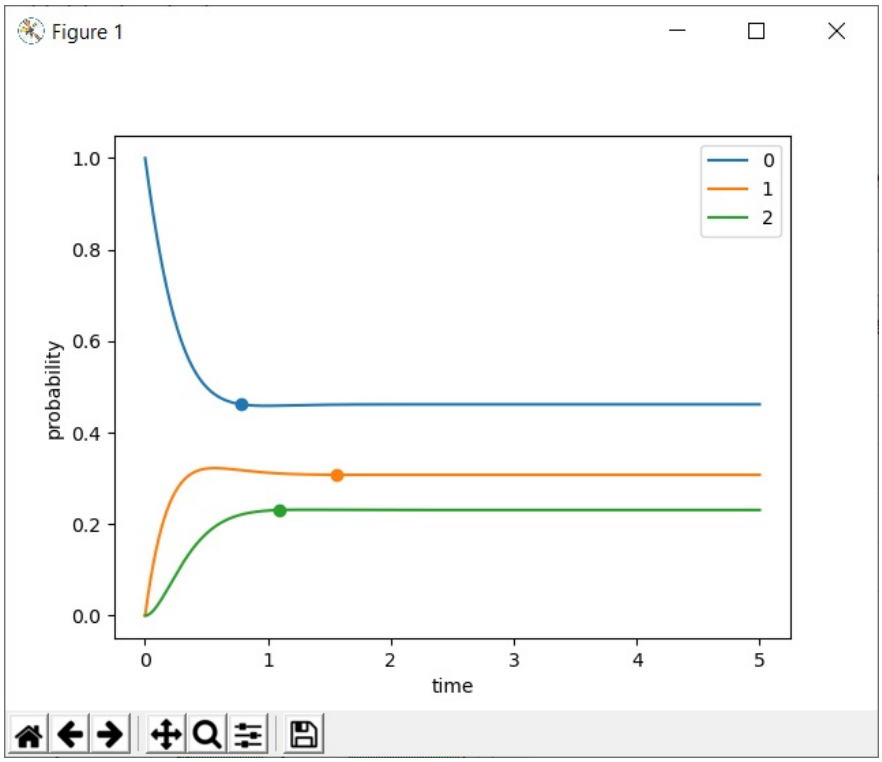


Рисунок 2 – Графики

Пример 2

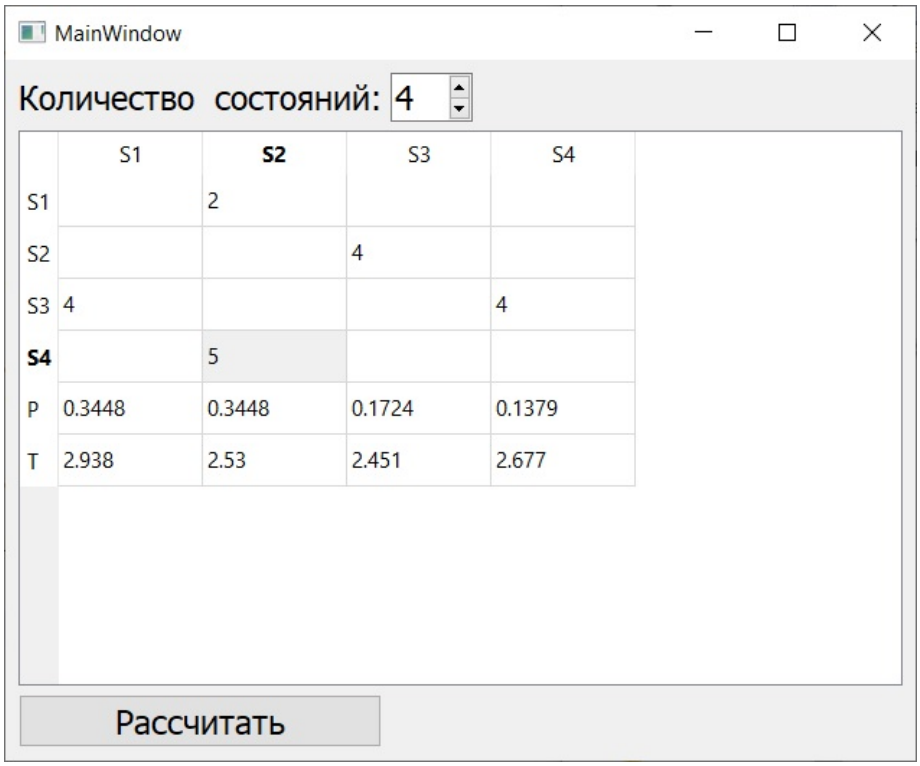


Рисунок 3 – Главное окно

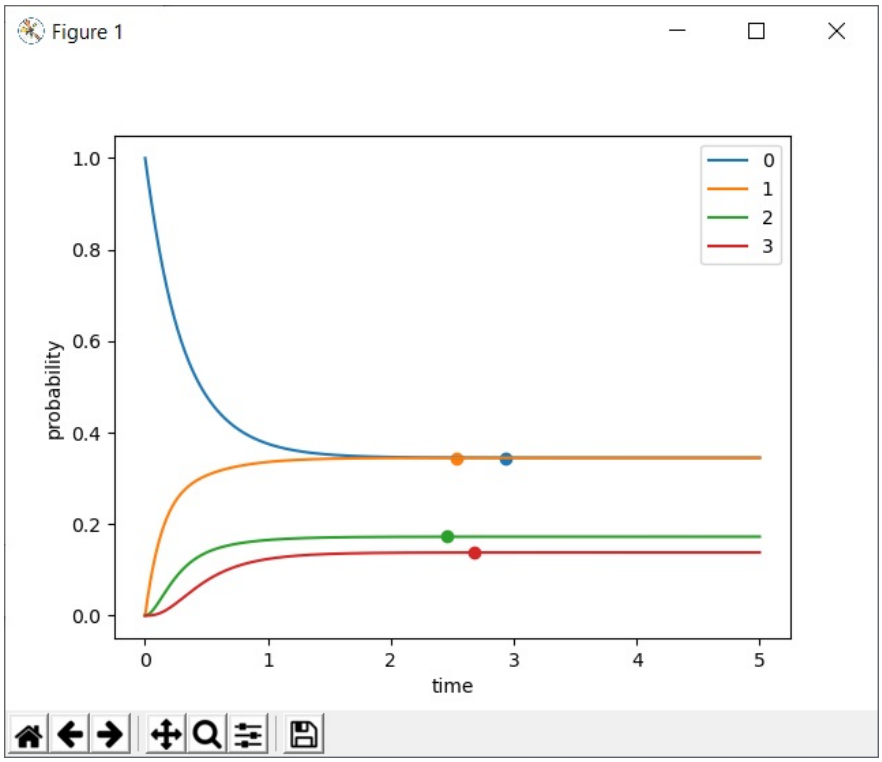
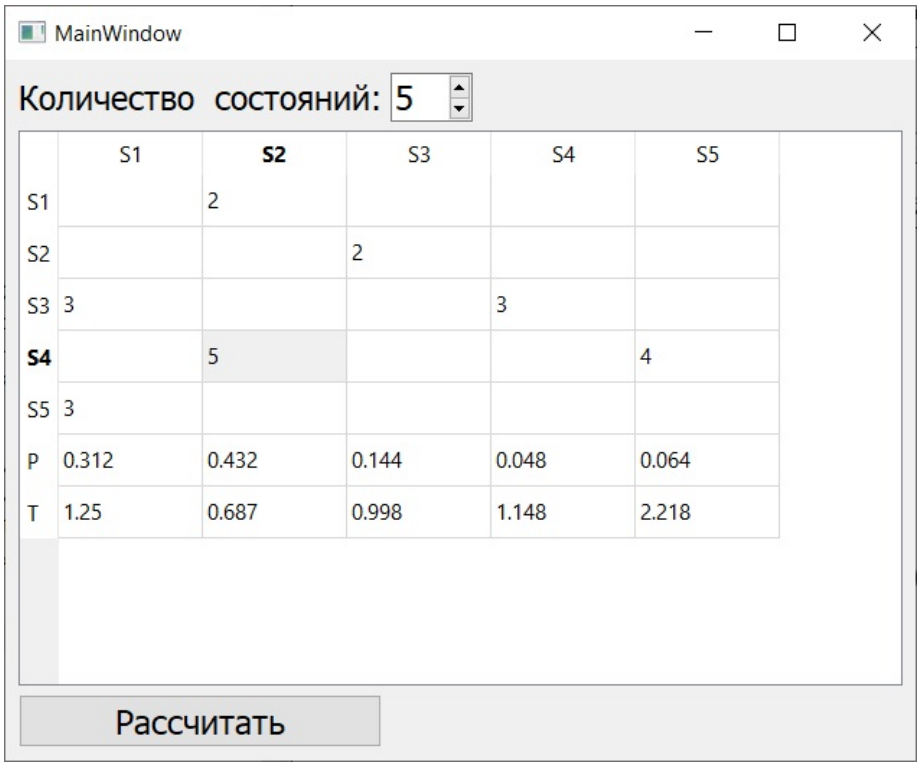


Рисунок 4 – Графики

Пример 3



	S1	S2	S3	S4	S5
S1		2			
S2			2		
S3	3			3	
S4		5			4
S5	3				
P	0.312	0.432	0.144	0.048	0.064
T	1.25	0.687	0.998	1.148	2.218

Рисунок 5 – Главное окно

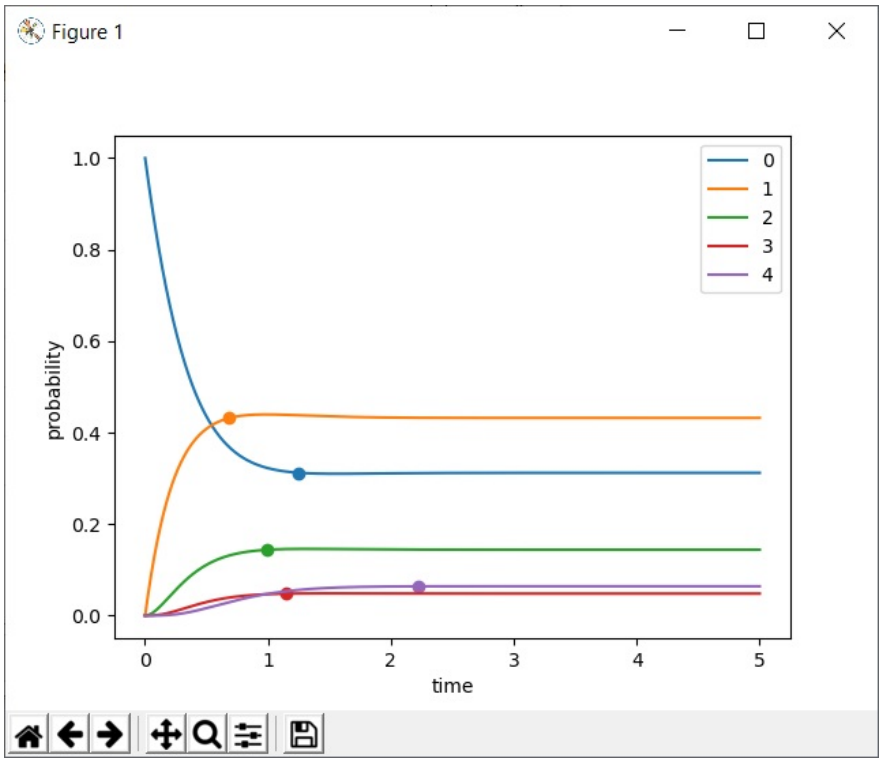


Рисунок 6 – Графики

Листинги

Для запуска программы необходимо установить библиотеку PyQt5 с помощью команды: pip install PyQt5

Листинг 1 – main.py

```
import sys
from PyQt5 import QtWidgets
from PyQt5 import uic, QtWidgets, QtGui
from PyQt5.QtWidgets import QApplication, QWidget,
    QListWidgetItem, QTableWidgetItem, QMessageBox
import design
import solvation
import stabilization
import matplotlib.pyplot as plt

class App(QtWidgets.QMainWindow, design.Ui_MainWindow):
    def __init__(self):
        self.is_ended = False
        self.ended_index = 0

        super().__init__()
        self.setupUi(self)
        self.initUI()

    def initUI(self):
        self.statesBox.valueChanged.connect(self.generateTable)
        self.table.itemChanged.connect(self.inputCheck)
        self.pushButton.clicked.connect(self.calculate)

        #self.setFixedSize(882, 485)

    def generateTable(self, value):
        self.table.setRowCount(value + 2)
        self.table.setColumnCount(value)
```

```
29          self.table.clearContents()
30
31          horizontalLabels = ["S" + str(i) for i in range(1, value +
                1)]
32          self.table.setHorizontalHeaderLabels(horizontalLabels)
33
34          verticalLabels = horizontalLabels.copy()
35          verticalLabels.append("P")
36          verticalLabels.append("T")
37          self.table.setVerticalHeaderLabels(verticalLabels)
38
39        def inputCheck(self, value):
40          try:
41            if value.text() != "":
42              float(value.text())
43          except ValueError:
44            QMessageBox.warning()
45            value.setText("")
46
47        def getMatrixFromTable(self):
48          res = []
49          try:
50            for i in range(self.table.rowCount() - 2):
51              row = []
52            for j in range(self.table.columnCount()):
53              item  = self.table.item(i, j)
54              if item and item.text() != "":
55                row.append(float(item.text()))
56              else:
57                row.append(0)
58            res.append(row)
59          except KeyError:
60            print(res)
61            QtWidgets.QMessageBox.warning()
62          return res
```

```
63
64          def generateFirstProbabilities(self, count):
65              res = [0] * count
66              res[0] = 1
67              return res
68
69          def drawGraphics(self, probabilities, stabilization_time,
                times, probabilities_over_time):
70              plt.close()
71              for i in range(len(probabilities_over_time[0])):
72                  plt.plot(times, [j[i] for j in probabilities_over_time])
73                  plt.scatter(stabilization_time[i], probabilities[i])
74
75              plt.legend(range(len(probabilities)))
76              plt.xlabel('time')
77              plt.ylabel('probability')
78              plt.show()
79
80          def inputProbabilities(self, probability):
81              if len(probability) == 0:
82                  QtWidgets.QMessageBox.critical()
83                  return
84              else:
85                  index = 0
86                  for state in probability:
87                      item = QTableWidgetItem()
88                      item.setText(str(round(state, 4)))
89                      self.table.setItem(self.table.rowCount() - 2, index,
                            item)
90                      index += 1
91
92          def inputTimings(self, timings):
93              if len(timings) == 0:
94                  QtWidgets.QMessageBox.critical()
95                  return
```

```
96          else:
97              index = 0
98              for state in timings:
99                  item = QTableWidgetItem()
100                 item.setText(str(round(state, 4)))
101                 self.table.setItem(self.table.rowCount() - 1, index,
                        item)
102                 index += 1
103
104         def calculate(self):
105             matrix = self.getMatrixFromTable()
106             probability = solvation.solve(matrix)
107             self.inputProbabilities(probability)
108
109             firstProbabilities = self.generateFirstProbabilities(len(
                    matrix))
110             stabilizationTime = stabilization.
                    CalculateStabilizationTimings(matrix, firstProbabilities,
                     probability)
111
112             self.inputTimings(stabilizationTime)
113             times, allProbabilities = stabilization.
                    CalculateAllProbabilities(matrix, firstProbabilities, 5)
114             self.drawGraphics(probability, stabilizationTime, times,
                    allProbabilities)
115
116
117     def main():
118         app = QtWidgets.QApplication(sys.argv)
119         window = App()
120         window.show()
121         app.exec_()
122
123     if __name__ == '__main__':
124         main()
```

Листинг 2 – stabilization.py

```
DELTA_T = 1e-3
EPS = 1e-4

def deltaPs(matrix, probabilities):
    size = len(matrix)
    result = []
    for i in range(size):
        equations = []
        for j in range(size):
            if i == j:
                elem = probabilities[j] * (-sum(matrix[i]) + matrix[i][i
                    ])
            else:
                elem = probabilities[j] * matrix[j][i]
            equations.append(elem)
        result.append(DELTA_T * sum(equations))
    return result

def CalculateStabilizationTimings(matrix, firstProbabilities,
    limitProbabilities):
    size = len(matrix)
    curTime = 0
    currentProbabilities = firstProbabilities.copy()
    stabilizationTimes = [0] * size

    while not all(stabilizationTimes):
        currdeltaPs = deltaPs(matrix, currentProbabilities)
        for i in range(size):
            if (not stabilizationTimes[i] and currdeltaPs[i] <= EPS
                and
                abs(currentProbabilities[i] - limitProbabilities[i]) <=
                    EPS):
                stabilizationTimes[i] = curTime
```

```
30          currentProbabilities[i] += currdeltaPs[i]
31
32        curTime += DELTA_T
33
34        return stabilizationTimes
35
36      def CalculateAllProbabilities(matrix, firstProbabilities,
          finishTime):
37        size = len(matrix)
38        curTime = 0
39        currentProbabilities = firstProbabilities.copy()
40
41        allProbabilities = []
42        timings = []
43
44        while curTime < finishTime:
45          allProbabilities.append(currentProbabilities.copy())
46          currdeltaPs = deltaPs(matrix, currentProbabilities)
47          for i in range(size):
48            currentProbabilities[i] += currdeltaPs[i]
49          curTime += DELTA_T
50          timings.append(curTime)
51
52        return timings, allProbabilities
```

Листинг 3 – solvation.py

```
import numpy

def coefMatrixGenerate(matrix):
    matrix = numpy.array(matrix)
    size = len(matrix)
    result = numpy.zeros((size, size))

    for state in range(size - 1):
        for column in range(size):
            result[state, state] -= matrix[state, column]
        for row in range(size):
            result[state, row] += matrix[row, state]

    for state in range(size):
        result[size - 1, state] = 1

    return result

def createIncreaseMatrix(count):
    result = [0] * count
    result[count - 1] = 1
    return numpy.array(result)

def solve(matrix):
    increaseMatrix = createIncreaseMatrix(len(matrix))
    coefMatrix = coefMatrixGenerate(matrix)
    return numpy.linalg.solve(coefMatrix, increaseMatrix)
```