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 курсу "Моделирование"

Тема Цепи Маркова

Студент Тэмуужин Янжинлхам

Группа ИУ7И-77б (ИУ7-73Б)

Преподаватель Рудаков И. В.

Москва — 2022 г.

1 Задание

Реализовать программу, позволяющую определить время пребывания сложной системы, работающей на базе цепей Маркова, во всех ее состояниях, определить момент достижения вероятностной константы, а также ее значение.

2 Программа

В программе реализованы следующие методы решения задачи:

1. Повторное умножение матрицы интенсивности. Для нахождении вероятности пребывания от состоянии i в состояние j при m шагов необходимо возведить матрицу A в степень m , то есть $P_{ij}(m) = A_{ij}^m$ Следовательно, при достаточно больших m для неразложимых цепей Маркова (цепь Маркова, в которой все состояния образуют один неразложимый класс) все строки матрицы A^∞ приравняются вероятностям в стационарном режиме. В данном методе не учитывается начальные вероятности состнояний.

2. Нахождение левого собственного вектора (left Eigen vectors). Выполняется умножение вектора начальных верятностей состояний и матрицы $\pi A = \pi$, пока выходной вектор вероятностей не будет равен входному. В качестве начального входного вектора (начальных вероятностей состнояний) используется вектор $\{1, 0, ..., 0\}$.

3 Результаты работы программы

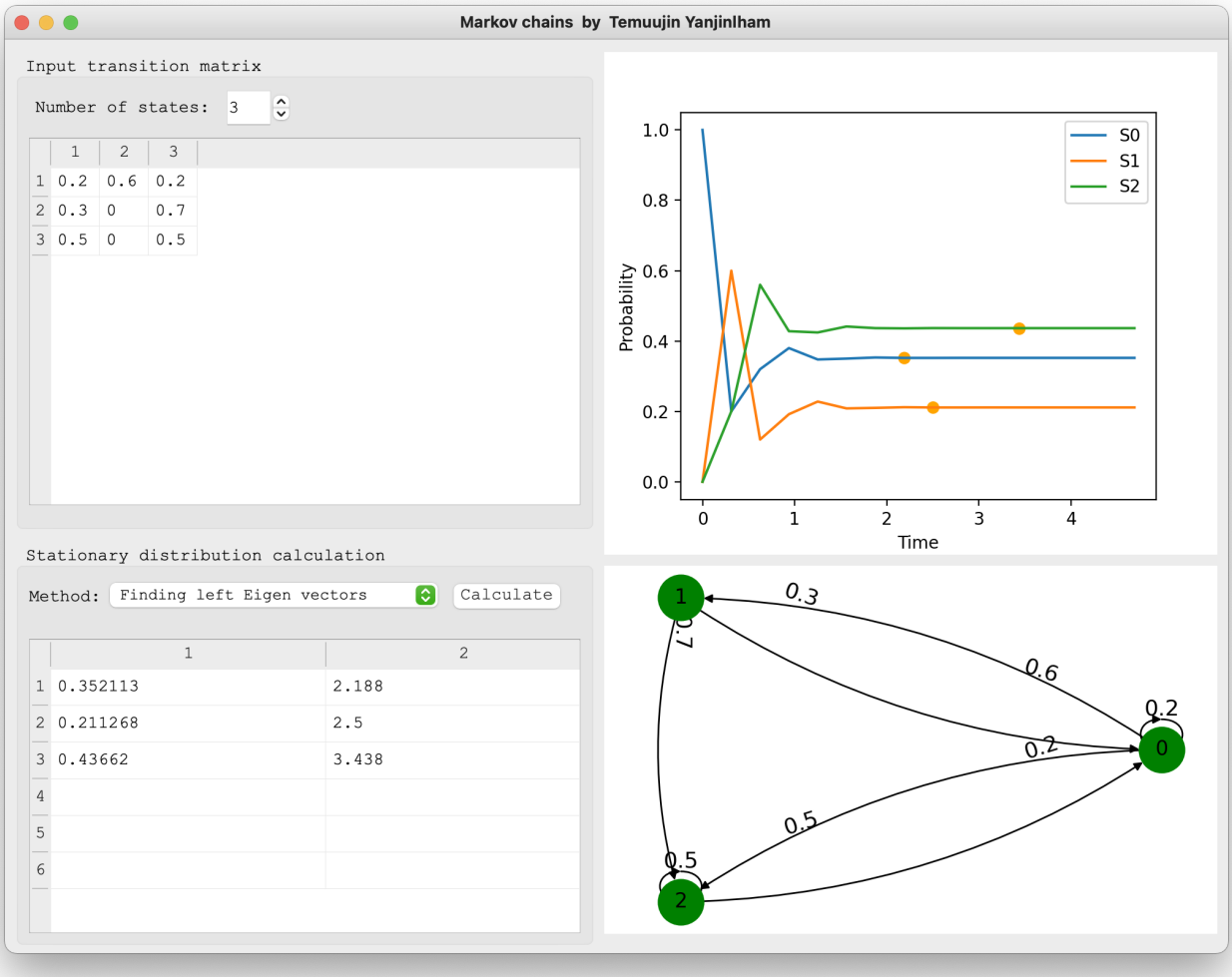

Рис. 3.1: Пример работы программы при 3 состояний, метод – повторное умножение матрицы

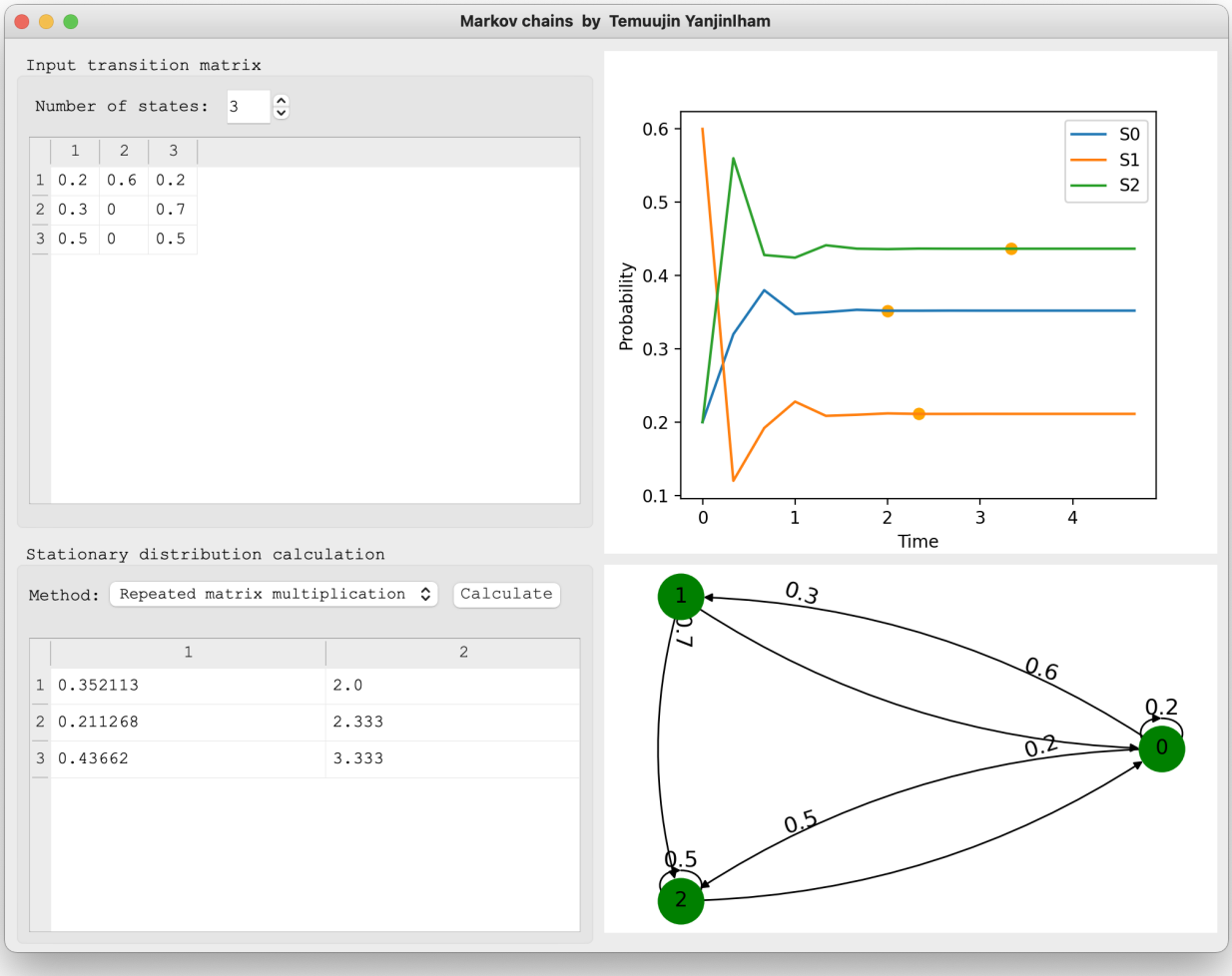

Рис. 3.2: Пример работы программы при 3 состояний, метод – нахождение левого собственного вектора

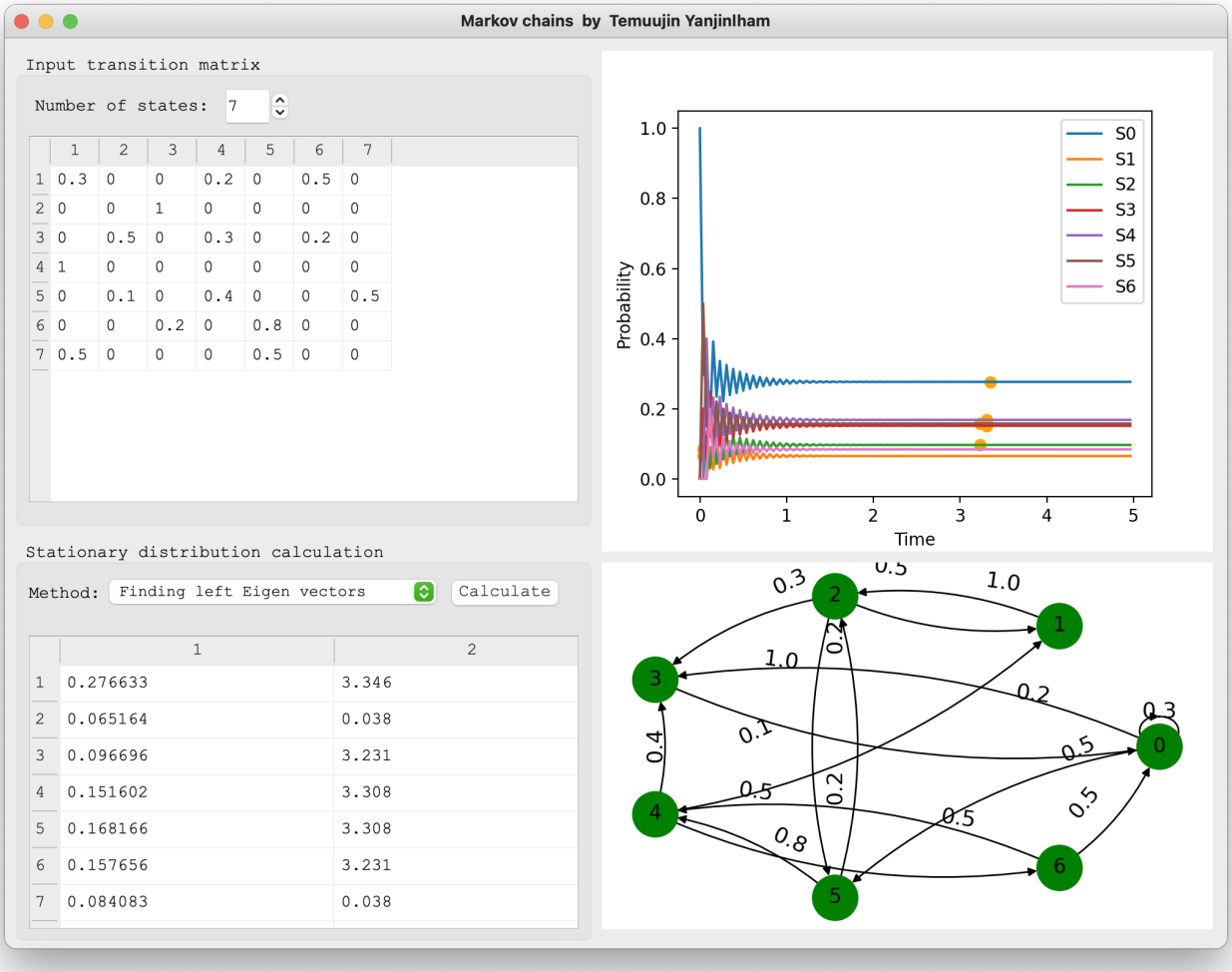

Рис. 3.3: Пример работы программы при 7 состояний, метод – повторное умножение матрицы

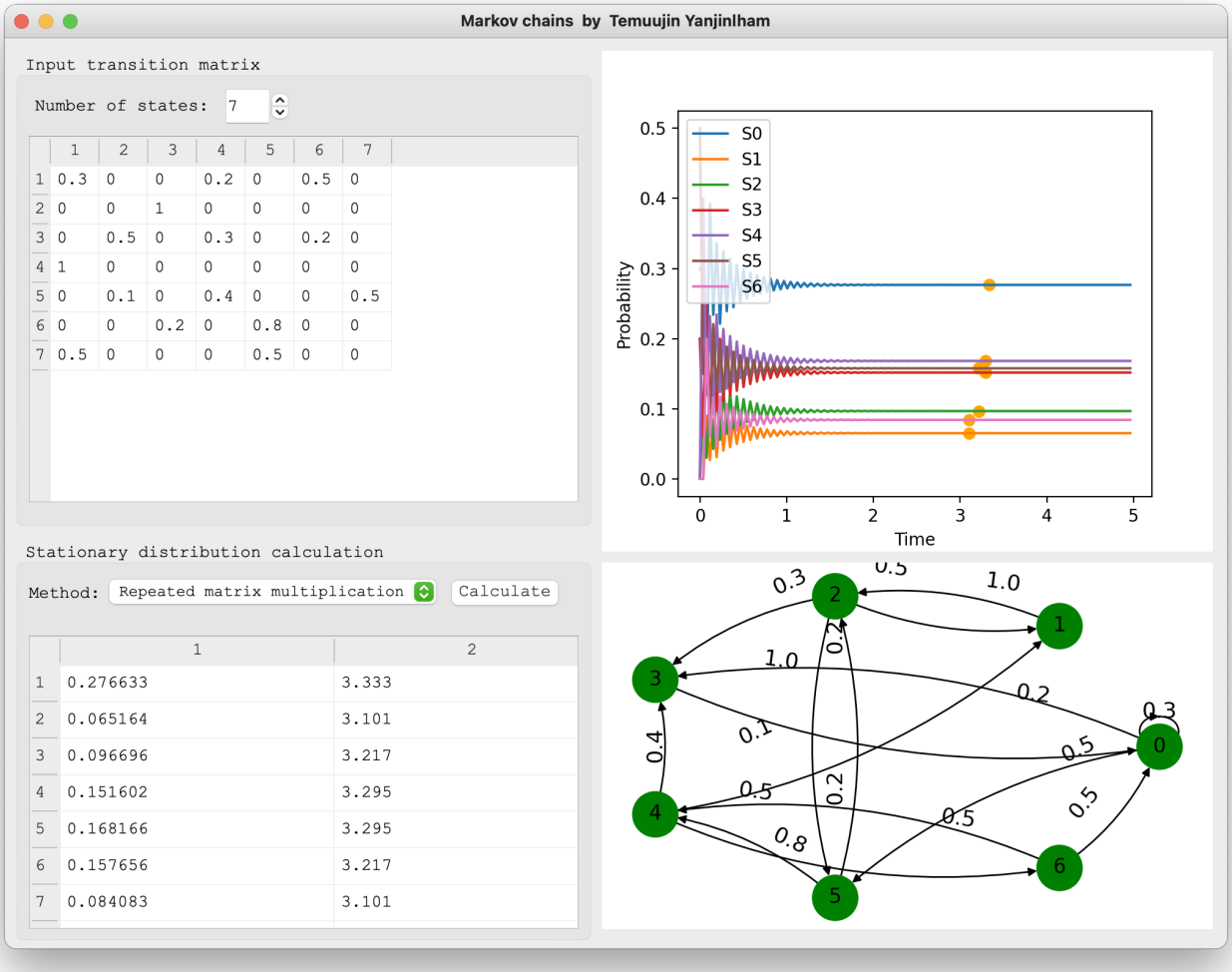

Рис. 3.4: Пример работы программы при 7 состояний, метод – нахождение левого собственного вектора

4 Код программы

Листинг 4.1: markov.py – Реализация методов

```
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx

PRECISION = 1e-5
TIME_DELTA = 5
GRAPH_OX = 1.5


def repeated_mult(tm, n):
    tm_n = tm
    stable_time = [0 for i in range(n)]
    probs = [[] for i in range(n)]
    max_stable_time = -1

    i = 0
    while i != int(max_stable_time*GRAPH_OX) or not all(s != 0 for s in stable_time):
        cur_tm = np.matmul(tm_n, tm)

        for state in range(n):
            probs[state].append(tm_n[0][state])
            if stable_time[state]==0 and abs(cur_tm[0][state] - tm_n[0][state]) <
                PRECISION:
                stable_time[state] = i
                if i > max_stable_time:
                    max_stable_time = i
        tm_n = cur_tm
        i += 1
    return tm_n[0], probs, stable_time


def left_eigenvector(tm, n):
    start_state = 0
    pi = np.array([0 for i in range(n)])
    pi[start_state] = 1
    max_stable_time = -1
    pi_n = pi
    stable_time = [0 for i in range(n)]
    probs = [[] for i in range(n)]

    i = 0
    while not all(s != 0 for s in stable_time) or i != int(max_stable_time*GRAPH_OX):
```

```
        cur_pi = np.matmul(pi_n, tm)

        for state in range(n):
            probs[state].append(pi_n[state])
            if stable_time[state]==0 and abs(cur_pi[state] - pi_n[state]) < PRECISION:
                stable_time[state] = i
                if i > max_stable_time:
                    max_stable_time = i
        i += 1
        pi_n = cur_pi
    return cur_pi, probs, stable_time
```

Листинг 4.2: main.py – Построение графов и интерфейса

```
import sys
from PyQt5.QtCore import pyqtSlot
from PyQt5 import uic, QtWidgets
from PyQt5.QtWidgets import QApplication, QWidget, QTableWidgetItem
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
from matplotlib.figure import Figure

from markov import *


class TimeGCanvas(FigureCanvasQTAgg):
    def __init__(self, parent=None, width=5, height=4, dpi=100):
        self.fig = Figure(figsize=(width, height), dpi=dpi)
        self.axes = self.fig.add_subplot()
        super(TimeGCanvas, self).__init__(self.fig)


class MainWindow(QWidget):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.ui = uic.loadUi("window.ui", self)
        self.tableWidgetMatrix.itemChanged.connect(lambda x: self._item_changed(x))
        self.spinBoxStatesCount.setValue(3)

        self.chooseMethod.setCurrentIndex(0)
        self.methods = [self.chooseMethod.itemText(i) for i in
            range(self.chooseMethod.count())]
        self.method = self.methods[1]
        self.time_graph = TimeGCanvas(self, width=5, height=4, dpi=100)

        self.figure = plt.figure()
        # plt.rcParams["figure.figsize"] = [7.50, 3.50]
        # plt.rcParams["figure.autolayout"] = True
        self.graph = FigureCanvasQTAgg(self.figure)
        self.vlStateGraph.addWidget(self.graph)
```

```
35
36
37      def _item_changed(self, value):
38          try:
39              if value.text() != "":
40                  float(value.text())
41          except ValueError:
42              QtWidgets.QMessageBox.critical(None, "Invalid␣input", "Enter␣a␣float␣
                    number!\nSum␣of␣the␣values␣per␣row␣must␣be␣equal␣to␣1")
43              value.setText("")
44
45
46      @pyqtSlot()
47      def on_chooseMethod_valueChanged(self, value):
48          self.method = value
49
50
51      @pyqtSlot()
52      def on_pushButtonCalc_clicked(self):
53          tm = self._get_matrix_from_table()
54          # tm = np.array([[0.2, 0.6, 0.2], [0.3, 0, 0.7], [0.5, 0, 0.5]])
55          tm = np.array(tm)
56          n = len(tm)
57          self.ui.resultTable.clear()
58          self.method = self.chooseMethod.currentText()
59          print("self.method", self.method)
60
61
62          if self._check_transition_matrix(tm) is False:
63              return
64
65          if self.method == self.methods[0]: # Repeated matrix multiplication
66              pi, probs, stable_time = repeated_mult(tm, n)
67              print(pi, probs, stable_time)
68          else:
69              pi, probs, stable_time = left_eigenvector(tm, n)
70              print(pi, probs, stable_time)
71
72          xlen = int(max(stable_time)*GRAPH_OX)
73          x = [i * TIME_DELTA / xlen for i in range(xlen)]
74          stable_time = [i * TIME_DELTA / xlen for i in stable_time]
75
76          currentRowCount = self.resultTable.rowCount()
77          for i in range(n):
78              self.resultTable.insertRow(currentRowCount)
79
80
81          # Result Table
```

```
        for state in range(n):
            # QListWidgetItem("n: time:0.5f".format(n = i, time = round(state, 5)),
                self.ui.resultTable)
            self.resultTable.setItem(state,0, QTableWidgetItem(str(round(pi[state], 6))))
            self.resultTable.setItem(state,1,
                QTableWidgetItem(str(round(stable_time[state], 3))))

        self.time_graph.close()
        self.time_graph = TimeGCanvas(self, width=5, height=4, dpi=100)

        # Time graph
        for i in range(n):
            self.time_graph.axes.plot(x, probs[i], label = 'S' + str(i))
            self.time_graph.axes.scatter(stable_time[i], pi[i], color='orange', s=40,
                marker='o')
        self.time_graph.axes.legend()
        self.time_graph.axes.set_xlabel('Time')
        self.time_graph.axes.set_ylabel('Probability')
        self.vlTimeGraph.addWidget(self.time_graph)


        # Weighed graph
        self.figure.clf()
        print(tm)
        DG = nx.DiGraph(tm, format='weighted_adjacency_matrix')
        pos = nx.circular_layout(DG)
        pos_nodes = self._nudge(pos, 0, 0.23)
        DG.graph['edge'] = {'arrowsize': '0.6', 'splines': 'curved'}

        nx.draw(DG, pos, with_labels=True, connectionstyle='arc3,␣rad=0.15',
            node_size=700, node_color='green', )
        labels = nx.get_edge_attributes(DG, 'weight')
        print(labels)
        nx.draw_networkx_edge_labels(DG, pos_nodes, edge_labels=labels, label_pos=0.75,
            font_size=13, bbox=dict(alpha=0))
        self.graph.draw_idle()


    @pyqtSlot('int')
    def on_spinBoxStatesCount_valueChanged(self, value):
        self.ui.tableWidgetMatrix.setRowCount(value)
        self.ui.tableWidgetMatrix.setColumnCount(value)
        self.ui.tableWidgetMatrix.clearContents()


    def _get_matrix_from_table(self):
        res = []
        try:
            for i in range(self.ui.tableWidgetMatrix.rowCount()):
```

```
126                 row = []
127                 for j in range(self.ui.tableWidgetMatrix.columnCount()):
128                     item = self.ui.tableWidgetMatrix.item(i, j)
129                     val = item.text() if item and item.text() != "" else "0"
130                     row.append(float(val))
131                 res.append(row)
132         except KeyError:
133             print(res)
134             QtWidgets.QMessageBox.critical(None, "Invalid input", "Enter a float
                    number!\nSum of the values per row must be equal to 1")
135         return res
136 
137 
138     def _check_transition_matrix(self, matrix):
139         for i in range(len(matrix)):
140             if sum(matrix[i]) != 1 or sum(matrix[i]) != 0.9999999999999999:
141 
142                 print("in _check_transition_matrix")
143                 print(sum(matrix[i]))
144                 print(matrix)
145                 QtWidgets.QMessageBox.critical(None, "Invalid input", "Enter a float
                        number!\nSum of the values per row must be equal to 1")
146                 return False
147         return True
148 
149 
150     def _matrix_to_nx(self, matrix):
151         result = []
152         for i in range(len(matrix)):
153             for j in range(len(matrix[i])):
154                 result.append((i, j, matrix[i][j]))
155         return result
156 
157 
158     def _nudge(self, pos, x_shift, y_shift):
159         return {n:(x + x_shift, y + y_shift) for n,(x,y) in pos.items()}
160 
161 
162 def main():
163     app = QApplication(sys.argv)
164     window = MainWindow()
165     window.show()
166     sys.exit(app.exec_())
167 
168 if __name__ == '__main__':
169     main()
```