1. Цель работы

Моделирование наблюдаемых и скрытых последовательностей.

2. Задание

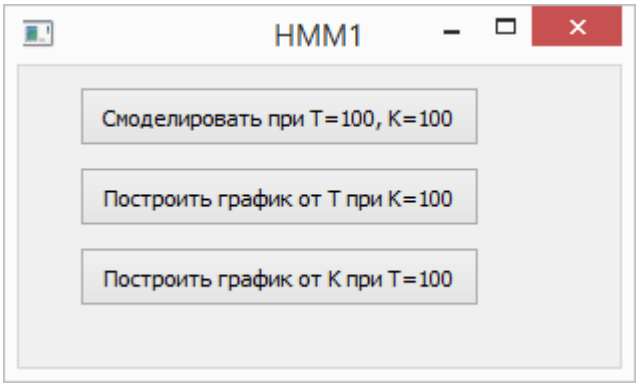
1) Смоделировать последовательность скрытых состояний длиной *T=100 (см [3], [4] стр 160-163)*
2) Смоделировать последовательность наблюдаемых состояний длиной *T=100 по* последовательности скрытых состояний *(см [3], [4] стр 160-163)*
3) Представить полученную наблюдаемую последовательность в графическом виде.
4) Привести графики достигнутой точности по параметрам (ρ_A , ρ_B) в зависимости от *T* при *K*=100 и от *K* при *T*=100.
5) Подготовить по 2 набора (каждый набор включает по K=100) наблюдаемых последовательностей.

3. Вариант задания

Вариант	*Алфавит: V*	*Матрица переходных вероятностей A*	*Матрица эмиссей B*
2	{0,1}	$\begin{pmatrix} 0,3 & 0,3 & 0,4 \\ 0 & 0,5 & 0,5 \\ 0,5 & 0 & 0,5 \end{pmatrix}$	$\begin{pmatrix} 0,2 & 0,8 \\ 0,8 & 0,2 \\ 0,5 & 0,5 \end{pmatrix}$

4. Описание разработанного программного средства

Программа разработана средствами Python + Qt5. Предназначена для моделирования скрытых и наблюдаемых последовательностей.



Интерфейс программного средства позволяет пользователю выбирать режим моделирования: Смоделировать при T=100, K=100 (с последующей записью последовательности в файл и представлением полученной последовательности в графическом виде), Построить график от T при K=100 (моделирование производится при различных длинах последовательности T), Построить график от K при T=100 (моделирование производится при различных K – количество проведения эксперимента). Результатом моделирования последовательность скрытых и наблюдаемых состояний.

5. Текст программы

```
import sys
from PyQt5.QtCore import QObject,Qt
from PyQt5.QtWidgets import QWidget, QLabel, QApplication, QCheckBox, QComboBox,
QPushButton, QLineEdit
import matplotlib
import numpy as np
import scipy as sp
matplotlib.use('Qt5Agg')
```

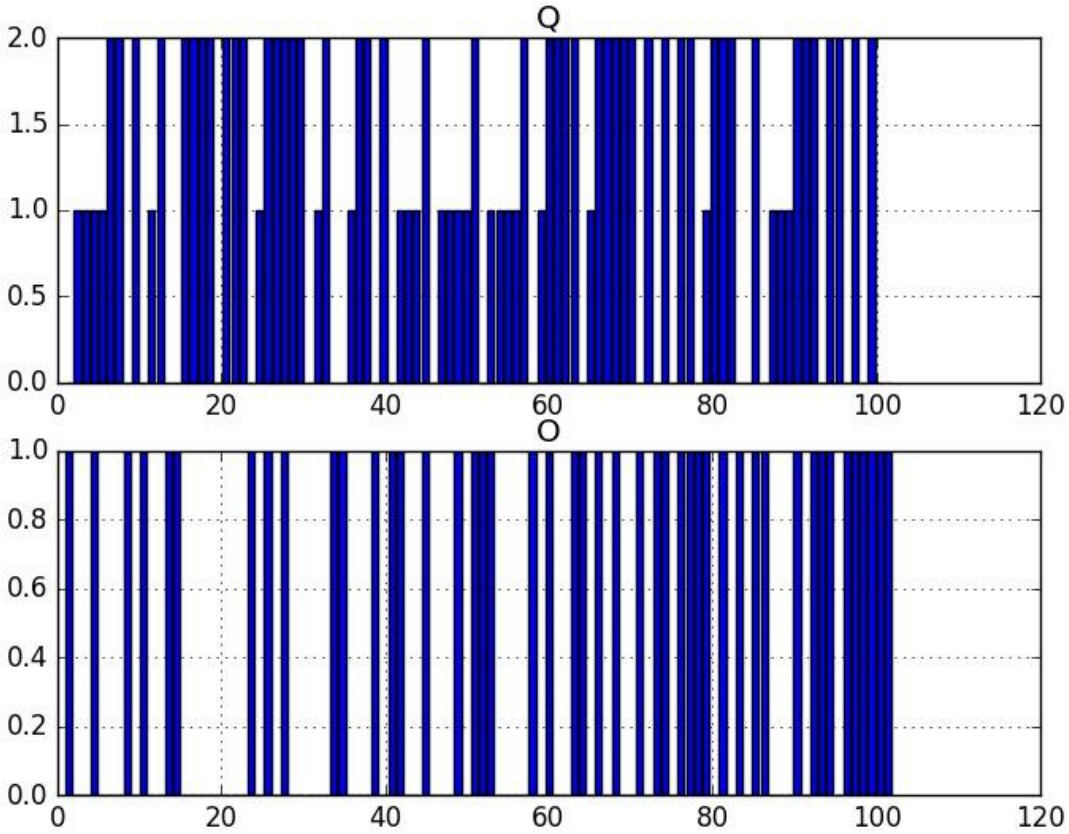
```
import matplotlib.pyplot as plt
from functools import reduce
def get_data(fname):
    A = np.array([[i for i in line.split()] for line in open(fname, encoding="utf-8")],
dtype = np.double)
    return A

def WritingInFile(names, sequences, fileName):
    with open(fileName, "w") as file:
        for line in sequences:
            print(line, file=file)
#моделирование последовательности скрытых состояний
def model_hiden_sequence(A, T):
    s = 0
    Q = [0]
    i = 0
    for j in range(T - 1):
        alpha = np.random.uniform(0,1)
        if 0. <= alpha <= A[i][0]:
             i = 0
        elif A[i][0] <= alpha <= A[i][0] + A[i][1]:
             i = 1
        elif A[i][0] + A[i][1] <= alpha <= 1:
             i = 2
        Q.append(i)
    return Q

#моделирование последовательности наблюдаемых состояний
#по последовательности скрытых состояний
def model_observable_sequence(Q, B, T):
    a = 0
    O = []
    for j in range(T):
        i = Q[j]
        alpha = np.random.uniform(0,1)
        if 0. <= alpha <= B[i][0]:
             a = 0
        elif B[i][0] <= alpha <= 1:
            a = 1
        O.append(a)
    return O

#подсчет пар для оценки переходной матрицы
def cnt_pair_matr_A(Q, t_1, t):
    cnt = 0
    for i in range(1, len(Q)):
        if Q[i-1] == t_1 and Q[i] == t:
            cnt += 1
    return cnt
#оценка матрицы A
def est_matr_A(Q):
    est_A = np.eye(3)
    tmp = np.array([[cnt_pair_matr_A(Q, i, j) for j in range(len(est_A))]
            for i in range(len(est_A))])
    tmp1 = np.sum(tmp, axis = 1)
    tmp1[tmp1 == 0] = 1
    est_A = list(map(lambda x, y: x / y, tmp, tmp1))
    return est_A

#подсчет пар для оценки матрицы эмиссей
def cnt_pair_matr_B(Q, O, s, a):
    cnt = 0
    for i in range(len(O)):
        if Q[i] == s and O[i] == a:
            cnt += 1
    return cnt
#оценка матрицы B
def est_matr_B(Q, O):
    est_B = np.zeros((3,2))
    tmp = np.array([[cnt_pair_matr_B(Q, O, i, j) for j in range(est_B.shape[1])]
            for i in range(est_B.shape[0])])
```

```
    tmp1 = np.sum(tmp, axis = 1)
    tmp1[tmp1 == 0] = 1
    est_B = list(map(lambda x, y: x / y, tmp, tmp1))
    return est_B

#метод Монте-Карло
def Monte_Karlo_method0(T, K):
    A = get_data('A.txt')
    B = get_data('B.txt')
    ro_A = []
    ro_B = []
    #моделируем
    Q = np.array([model_hiden_sequence(A, T) for i in range(K)])
    O = np.array(list(map(lambda x: model_observable_sequence(x, B, T), Q)))
    #оцениваем
    est_A = np.array(list(map(lambda x: est_matr_A(x), Q)))
    est_B = np.array(list(map(lambda x, y: est_matr_B(x, y), Q, O)))
    #усредненная оценка матриц
    est_A_mean = np.array(list(reduce((lambda a, x: a + x), est_A))) / K
    est_B_mean = np.array(list(reduce((lambda a, x: a + x), est_B))) / K
    #норма разности
    ro_A.append(np.linalg.norm(A -  est_A_mean))
    ro_B.append(np.linalg.norm(B - est_B_mean))

    WritingInFile(['Q'], Q, 'Q.txt')
    WritingInFile(['O'], O, 'O.txt')
    return np.array(ro_A), np.array(ro_B), Q, O

def Monte_Karlo_method(T):
    A = get_data('A.txt')
    B = get_data('B.txt')
    ro_A = []
    ro_B = []
    for K in range(1, 101):
        #моделируем
        Q = np.array([model_hiden_sequence(A, T) for i in range(K)])
        O = np.array(list(map(lambda x: model_observable_sequence(x, B, T), Q)))
        #оцениваем
        est_A = np.array(list(map(lambda x: est_matr_A(x), Q)))
        est_B = np.array(list(map(lambda x, y: est_matr_B(x, y), Q, O)))
        #усредненная оценка матриц
        est_A_mean = np.array(list(reduce((lambda a, x: a + x), est_A))) / K
        est_B_mean = np.array(list(reduce((lambda a, x: a + x), est_B))) / K
        #норма разности
        ro_A.append(np.linalg.norm(A -  est_A_mean))
        ro_B.append(np.linalg.norm(B - est_B_mean))
    return np.array(ro_A), np.array(ro_B), Q, O

def Monte_Karlo_method1(K):
    A = get_data('A.txt')
    B = get_data('B.txt')
    ro_A = []
    ro_B = []
    for T in range(1, 101):
        #моделируем
        Q = np.array([model_hiden_sequence(A, T) for i in range(K)])
        O = np.array(list(map(lambda x: model_observable_sequence(x, B, T), Q)))
        #оцениваем
        est_A = np.array(list(map(lambda x: est_matr_A(x), Q)))
        est_B = np.array(list(map(lambda x, y: est_matr_B(x, y), Q, O)))
        #усредненная оценка матриц
        est_A_mean = np.array(list(reduce((lambda a, x: a + x), est_A))) / K
        est_B_mean = np.array(list(reduce((lambda a, x: a + x), est_B))) / K
        #норма разности
        ro_A.append(np.linalg.norm(A -  est_A_mean))
        ro_B.append(np.linalg.norm(B - est_B_mean))
    return np.array(ro_A), np.array(ro_B), Q, O

def plot_graph(roA, roB):
        plt.xlabel('K')
        plt.ylabel('ro')
```

```
        plt.xlim(0,101)
        k = np.linspace(1, 101, 100)
        f = plt.plot(k, roA)
        f1 = plt.plot(k, roB, 'r-')
        plt.show()

def plot_graph1(roA, roB):
        plt.xlabel('T')
        plt.ylabel('ro')
        plt.xlim(0,101)
        k = np.linspace(1, 101, 100)
        f = plt.plot(k, roA)
        f1 = plt.plot(k, roB, 'r-')
        plt.show()

def plot_hist(ydata):
    xdata = np.linspace(1, 101, 100)
    plt.xlim(0, 100)
    plt.bar(xdata, ydata[0])
    plt.show()

class Interface(QWidget):

    def __init__(self):
        super().__init__()
        self.Tmax = 100 #длина последовательности
        self.Kmax = 100 #кол-во моделирования в методе Монте-Карло
        self.ro_A = 0
        self.ro_B = 0
        self.Q = 0
        self.O = 0
        self.initUI()

    def initUI(self):
        btn1 = QPushButton('Смоделировать при T=100, K=100', self)
        btn1.move(30, 10)
        btn1.resize(200,30)
        btn1.clicked.connect(self.buttonClicked)
        btn2 = QPushButton('Построить график от T при K=100', self)
        btn2.move(30, 50)
        btn2.resize(200,30)
        btn2.clicked.connect(self.buttonClicked1)
        btn3 = QPushButton('Построить график от K при T=100', self)
        btn3.move(30, 90)
        btn3.resize(200,30)
        btn3.clicked.connect(self.buttonClicked2)
        self.setGeometry(270, 270, 300, 150)
        self.setWindowTitle('HMM1')
        self.show()

    def buttonClicked(self):
        self.ro_A, self.ro_B, self.Q, self.O = Monte_Karlo_method0(self.Tmax, self.Kmax)
        plot_hist(self.Q)
        plot_hist(self.O)

    def buttonClicked1(self):
        self.ro_A, self.ro_B, self.Q, self.O = Monte_Karlo_method1(self.Kmax)
        plot_graph1(self.ro_A, self.ro_B)

    def buttonClicked2(self):
        self.ro_A, self.ro_B, self.Q, self.O = Monte_Karlo_method(self.Tmax)
        plot_graph(self.ro_A, self.ro_B)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = Interface()
    sys.exit(app.exec_())
```

6. Исследования

Моделирование последовательностей при T=100, K=100



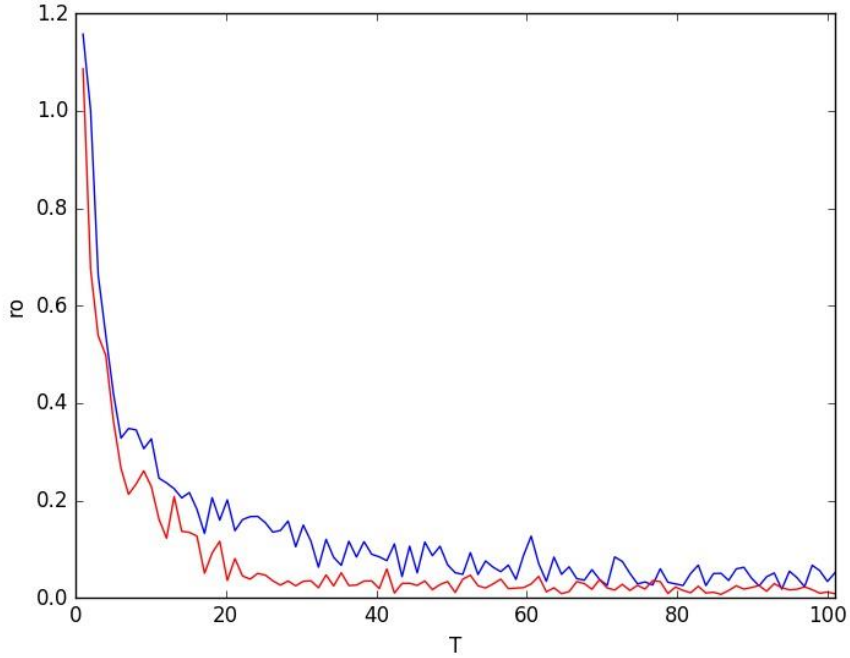
Q.txt

```
[0 1 1 1 1 2 2 0 2 0 1 2 0 0 2 2 2 2 0 2 2 2 0 1 2 2 2 2 2 0 1 2 0 0 1 2 2 0 2 0 1 1 1
2 0 1 1 1 1 2 0 1 1 1 1 2 0 1 2 2 2 2 0 1 2 2 2 2 2 0 2 0 2 0 2 2 0 1 2 2 2 0 0 2 0 1
1 1 2 2 2 0 2 2 0 2 0 2 0 0]
[0 2 0 2 2 0 0 1 1 2 0 2 0 2 0 0 2 0 1 2 2 2 0 2 2 0 1 1 2 0 1 1 1 1 1 1 2 2 0 0 2 2 2
2 2 2 2 2 2 0 1 1 2 2 2 2 0 0 1 1 2 2 2 0 2 0 1 1 2 0 2 0 0 1 1 1 1 1 1 2 0 0 1 2 0 0
0 1 2 0 1 1 1 2 0 1 1 2 0 2]
[0 1 1 1 2 0 1 1 1 1 1 2 0 0 2 0 2 0 0 0 2 2 0 2 0 0 1 1 1 1 2 2 2 0 2 0 0 0 1 1 2 0 1
2 2 2 2 0 2 0 2 0 2 2 2 2 2 0 2 0 1 2 2 2 2 2 0 1 1 2 2 0 2 0 2 0 1 1 2 2 2 2 2 0 2 0 2
2 2 0 0 2 2 0 0 1 1 1 2 0 0]
[0 2 2 2 0 0 0 2 2 2 2 0 1 2 2 0 2 0 2 2 0 0 2 2 2 0 1 1 1 1 2 2 2 2 0 1 2 2 2 2 2 0 2
2 0 2 0 1 2 2 0 1 1 1 2 2 0 1 1 2 0 2 0 1 2 0 1 2 0 0 0 0 1 2 2 2 2 2 2 2 0 0 2 0 2 2 0
2 0 0 0 2 2 0 1 1 1 2 0 1 1]  …
```

O.txt

```
[1 0 0 1 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 1 0 0
1 0 0 0 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 0 1 0 0 1 0 1 1 0 1 1 1 1 0 1 0 1 0 1 1 0
0 0 1 0 1 1 1 0 1 1 1 1 1 1]
[1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0
0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1
1 1 0 1 0 0 0 1 1 0 1 1 1 1]
[1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 1 1 0 0 1 0 1 0
1 0 0 0 1 1 1 1 1 0 1 0 0 0 1 1 1 0 0 0 1 1 0 1 0 1 1 0 1 0 1 1 1 0 0 1 1 1 0 1 1 1 1
0 0 0 1 1 1 1 0 0 0 1 1 1 1]
[1 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 1 0 1 1 0 1 0 0 0 1 1 1
0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0
0 0 1 1 0 1 1 0 0 1 1 1 0 0]  ….
```

Построить график от T при K=100



Построить график от K при T=100