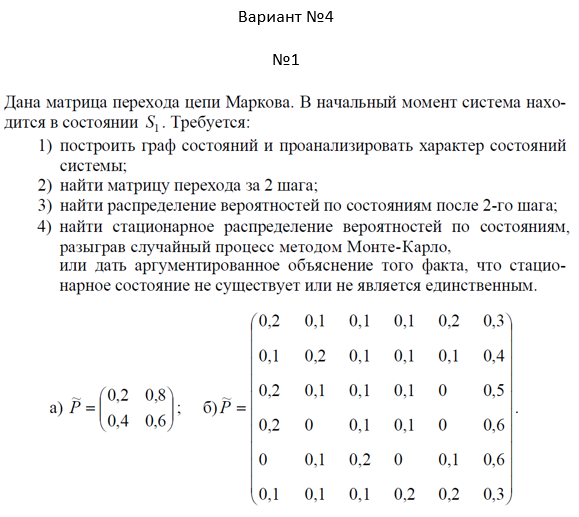
# Контрольная работа №3

Студент: Болдинов Алексей ЭФМО-02-24

**Вариант – 4**

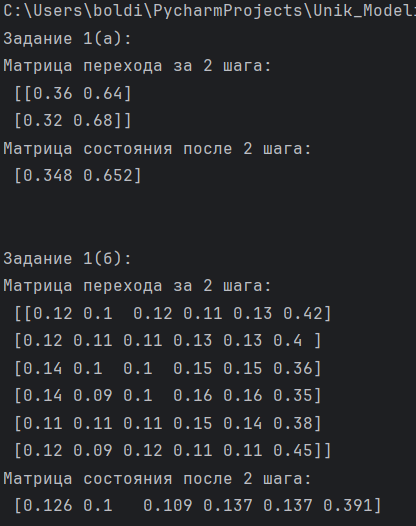
**Задание 1**:



Код для решения поставленной задачи:

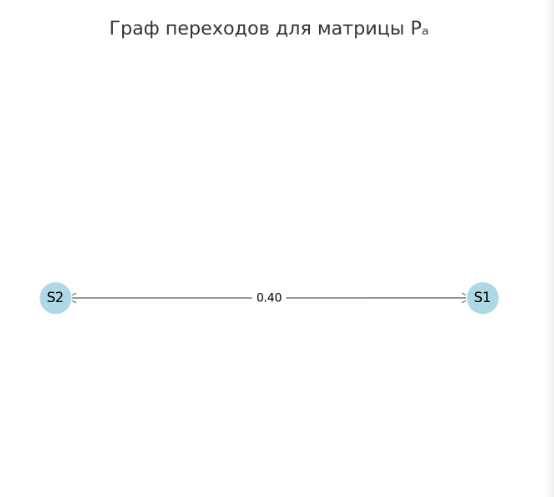
import numpy as np  
  
def zadanie1\_1():  
 times = 2  
 start = np.array([0.7, 0.3])  
 matricaPerehoda = np.array([  
 [0.2, 0.8],  
 [0.4, 0.6]  
 ])  
 # check = [[0.4,0.6],[0.3,0.7]] #Возведение в степень работет и без numpy для массива из списка  
 matricaPerehodaForTimes = np.linalg.matrix\_power(matricaPerehoda, times)  
 print(f"Матрица перехода за {times} шага:\n", matricaPerehodaForTimes)  
 end = start.dot(matricaPerehodaForTimes)  
 print(f"Матрица состояния после {times} шага:\n", end)  
 #print(sum(end))  
  
def zadanie1\_2():  
 times = 2  
 start = np.array([0.1, 0.1, 0.3, 0.1, 0.2, 0.2])  
 matricaPerehoda = np.array([  
 [0.2, 0.1, 0.1, 0.1, 0.2, 0.3],  
 [0.1, 0.2, 0.1, 0.1, 0.1, 0.4],  
 [0.2, 0.1, 0.1, 0.1, 0.0, 0.5],  
 [0.2, 0.0, 0.1, 0.1, 0.0, 0.6],  
 [0.0, 0.1, 0.2, 0.0, 0.1, 0.6],  
 [0.1, 0.1, 0.1, 0.2, 0.2, 0.3]  
 ])  
 # check = [[0.4,0.6],[0.3,0.7]] #Возведение в степень работет и без numpy для массива из списка  
 matricaPerehodaForTimes = np.linalg.matrix\_power(matricaPerehoda, times)  
 print(f"Матрица перехода за {times} шага:\n", matricaPerehodaForTimes)  
 end = start.dot(matricaPerehodaForTimes)  
 print(f"Матрица состояния после {times} шага:\n", end)  
  
 #print(sum(end))

Вывод программы:



1. Ответ:

**Для матрицы 1(а):**



**Для матрицы 1(б):**



1. Ответ в выводе программы
2. Ответ в выводе программы
3. Ответ:

### **Стационарное распределение**

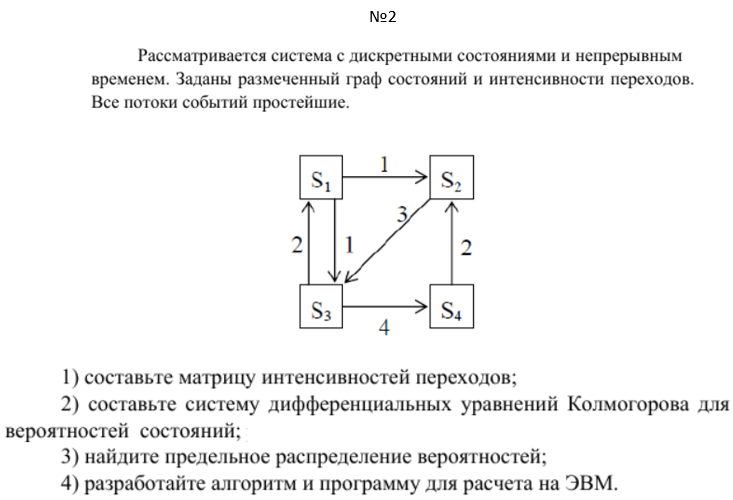
* Для матрицы **Pₐ**:

[0.3333,0.6667][0.3333, 0.6667][0.3333,0.6667]

* Для матрицы **Pᵦ**:

[0.1092,0.0140,0.0154,0.0017,0.0154,0.8442][0.1092, 0.0140, 0.0154, 0.0017, 0.0154, 0.8442][0.1092,0.0140,0.0154,0.0017,0.0154,0.8442]

**Задание 2**.



Код для решения поставленной задачи:

def zadanie2():  
 from cvxopt.modeling import variable, op  
 # Задаем матрицу интенсивностей переходов  
 lam = [  
 [0, 1, 1, 0], # Интенсивности переходов из S1  
 [0, 0, 0, 3], # Интенсивности переходов из S2  
 [2, 0, 0, 4], # Интенсивности переходов из S3  
 [0, 2, 0, 0], # Интенсивности переходов из S4  
 ]  
  
 # Переменные вероятностей состояний  
 s = variable(4, 's')  
  
 # Уравнения Колмогорова (в стационарном режиме dP/dt = 0)  
 mass1 = ((lam[0][1] + lam[0][2]) \* s[0] == lam[1][0] \* s[1] + lam[2][0] \* s[2])  
 mass2 = ((lam[1][3]) \* s[1] == lam[0][1] \* s[0] + lam[3][1] \* s[3])  
 mass3 = ((lam[2][0] + lam[2][3]) \* s[2] == lam[0][2] \* s[0] + lam[3][2] \* s[3])  
 mass4 = ((lam[3][1]) \* s[3] == lam[1][3] \* s[1] + lam[2][3] \* s[2])  
  
 # Условие нормировки: сумма всех вероятностей равна 1  
 normalization = (s[0] + s[1] + s[2] + s[3] == 1)  
  
 # Ограничения вероятностей (все вероятности должны быть в диапазоне [0, 1])  
 constraints = [0 <= s, s <= 1]  
  
 # Оптимизационная задача для нахождения решения системы уравнений  
 problem = op(0, [mass1, mass2, mass3, mass4, normalization] + constraints)  
 problem.solve(solver='glpk')  
  
 # Вывод результатов  
 stationary\_distribution = [round(value, 4) for value in s.value]  
 print("1) Матрица интенсивностей переходов:")  
 for row in lam:  
 print(row)  
  
 print("\n2) Стационарное распределение вероятностей:")  
 print(stationary\_distribution)

)

Вывод программы:

