Министерство образования и науки 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

На тему: Цепи Маркова

Студент:
Андреев Александр Алексеевич
Группа: ИУ7-74Б

Преподователь:
Рудаков Игорь Владимирович

Москва, 2022 г.

Содержание

1 Задание **2**

2 Теоритическая часть **2**

3 Результаты **3**

4 Листинг кода **6**

1 Задание

Для сложной системы S, имеющей не более 10 состояний, необходимо определить время пребывания сложной системы в каждом из состояний. На вход подается матрица, на пересечении строк и столбцов которой находятся интенсивности переходов.

2 Теоритическая часть

Случайный процесс называется Марковским, если он обладает следующим свойством: для каждого момента времени t_0 вероятность любого состояния системы в будущем $t > t_0$ зависит только от ее состояния в настоящем, то есть при $t = t_0$, и не зависит от того когда и каким образом система пришла в это состояние. Не зависит от того, как процесс развивался в прошлом. В природе нет таких процессов, но существует ряд процессов, которые могут некоторыми методами быть сведены к Марковским процессам. Для Марковских процессов хорошо работают Уравнения Колмагорова.

По модели из условия строятся уравнения Колмогорова: в левой части уравнений находится производная вероятности состояний, а правая часть содержит члены по количеству переходов, связанных с текущим состоянием. Если направление перехода в текущее состояние, то соответствующий член имеет знак минус, если направление из состояния, то плюс. Каждый член равен произведению плотности вероятности перехода на веропятость того состояния, из которого идет этот переход.

Поскольку модель имеет установившийся режим, то левые части уравненя будут равно нулю. Далее вводится уравнение нормировки и производится подсчет.

Получившиеся вероятности являются средним относительным временем пребывания системы в данном состоянии.

Среднее время находится по формуле 1

$$t_i = \frac{1 - p_i}{p_i \cdot \sum_{i \neq j} \lambda_{ij}} \tag{1}$$

3 Результаты

Состояние	Время стабилизации
1	0.875
2	0.7917
3	0.5417
4	0.9167

Рис. 1: Пример работы для 4 состояний

Добавить Удалить

	1	2	3	4	5	6
1		6			8	
2			4			1
3				2		
4	8					
5			3			
6		2			2	

Вычислить

Состояние	Время стабилизации
1	0.9504
2	1.9458
3	0.5218
4	0.8968
5	1.3173
6	10.4792

Рис. 2: Пример работы для 6 состояний

	1	2	3	4	5	6	7	8	9	10
1		6			8					
2			4			1			9	
3				2		5		6		3
4	8									
5			3	3			2	8		5
6		2			2					
7	8						5			6
8			9		3					
9			1		9			9	7	8
10	1									

Состояние	Время стабилизации
1	1.0504
2	1.7746
3	1.0509
4	3.8847
5	0.9942
6	2.8813
7	10.8861
8	1.1751
9	2.8422
10	0.657

Рис. 3: Пример работы для 10 состояний

4 Листинг кода

```python
# Created by Andreev A.A. 25.10.2022
from flask import Flask, render_template, session, request
import numpy as np

app = Flask(__name__)
app.secret_key = "A0Zr98j/3yX R~XHH!jmN]LWX/,?RT"


@app.route("/")
@app.route("/<int:count>")
def index(count=1):
    if "username" not in session:
        session["username"] = "user"
        session["data"] = []
        session["result"] = []
    session["count"] = count
    return render_template("index.html", session=session)


@app.route("/plus")
def add_count():
    session["count"] += 1
    session["result"] = []
    return render_template("index.html", session=session)


@app.route("/minus")
def remove_count():
    if session["count"] > 1:
        session["count"] -= 1
        session["result"] = []
    return render_template("index.html", session=session)

def get_session_data(count):
    data = []
    for i in range(count):
        data.append([])
```

```
        for j in range(count):
            s = request.form[str(i * count + j)]
            data[i].append(int(s) if s != "" else 0)
    return data

def get_params(count, data):
    coef = np.zeros((count, count))
    summ = np.zeros(count)

    for i in range(count):
        summ[i] = sum(data[i])
        for j in range(count):
            coef[i][j] = data[j][i]
        coef[i][i] = -summ[i]

    return coef, summ

@app.route("/", methods=["POST"])
@app.route("/<int:count>", methods=["POST"])
@app.route("/plus", methods=["POST"])
@app.route("/minus", methods=["POST"])
def solve():
    count = session["count"]
    session["data"] = get_session_data(count)
    coef, summ = get_params(count, session["data"])

    m = np.zeros(count)
    m[-1] = 1
    coef[-1] = 1
    out = np.linalg.solve(coef, m)
    time = (1 - out) / out / summ
    session["result"] = [round(t, 4) for t in time.tolist()]

    return render_template("index.html", session=session)


if __name__ == "__main__":
```

```
75      app.run(debug=True)
```

Листинг 1: Программная реализация определения времени пребывания сложной системы в каждом из состояний