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

Название Марковские процессы

Дисциплина Моделирование

Студент Егорова П. А.

Группа ИУ7-74Б

Оценка (баллы) ______________________

Преподаватель Рудаков И. В.

Москва — 2023 г.

1 Задание

Написать программу, которая позволяет определить время пребывания сложной системы в каждом из состояний в установившемся режиме работы. Количество состояний ≤ 10.

Реализовать интерфейс, который позволяет указать количество состояний и значения матрицы вероятностей переходов, а также отображает результаты работы программы: графики вероятностей состояний, время стабилизации вероятности каждого состояния, стабилизировавшееся значение вероятности каждого состояния.

2 Теоретические сведения

Случайный процесс, протекающий в некоторой системе S, называется Марковским, если он обладает следующим свойством: для каждого момента времени вероятность любого состояния системы в будущем зависит только от её состояния в настоящем и не зависит от того, когда и каким образом она пришла в это состояние (то есть не зависит от прошлого).

Для марковского процесса обычно составляются уравнения Колмогорова:

$$F = (P'(t), P(t), \lambda) = 0,$$

где λ - некоторый набор коэффициентов.

Интегрирование системы уравнений даёт искомые вероятности как функции времени. Начальное условие берется в зависимости от того, какое было начальное состояние системы. Кроме того, необходимо добавить условие нормировки: $\sum_{i=1}^{n} P_i(t) = 1$ для любого момента t. $P_i(t)$ – вероятность того, что в момент t система будет находиться в i-м состоянии.

Уравнения Колмогорова строятся по следующим правилам:

- В левой части каждого уравнения стоит производная вероятности i-ого состояния, а правая часть содержит столько членов, сколько переходов связано с данным состоянием.

- Если переход осуществляется из этого состояния, то соответствующий член имеет знак минус, если в это состояние, то плюс.

- Каждый член равен произведению плотности вероятности перехода (интенсивности), соответствующей данному переходу, и вероятности того состояния, из которого осуществляется переход.

Для определения предельных вероятностей при $t \to \infty$ (то есть вероятностей в стационарном режиме работы), необходимо приравнять левые части уравнений (то есть производные) к нулю и решить полученную систему линейных уравнений.

Чтобы найти время стабилизации, необходимо найти момент времени t_s, когда значение производной $P_i^{'}(t_s)$ меньше заранее заданного ε. Тогда приращение соответствующей вероятности к следующему моменту времени $\Delta P_i = P_i^{'}(t_s)\Delta t$ будет меньше некоторой погрешности.

3 Результаты работы программы

В начале работы система находится в первом состоянии, ε для определения стабилизации вероятности принимается равным 10^{-5}.

На рисунке 1 приведен пример работы программы для 3 состояний.

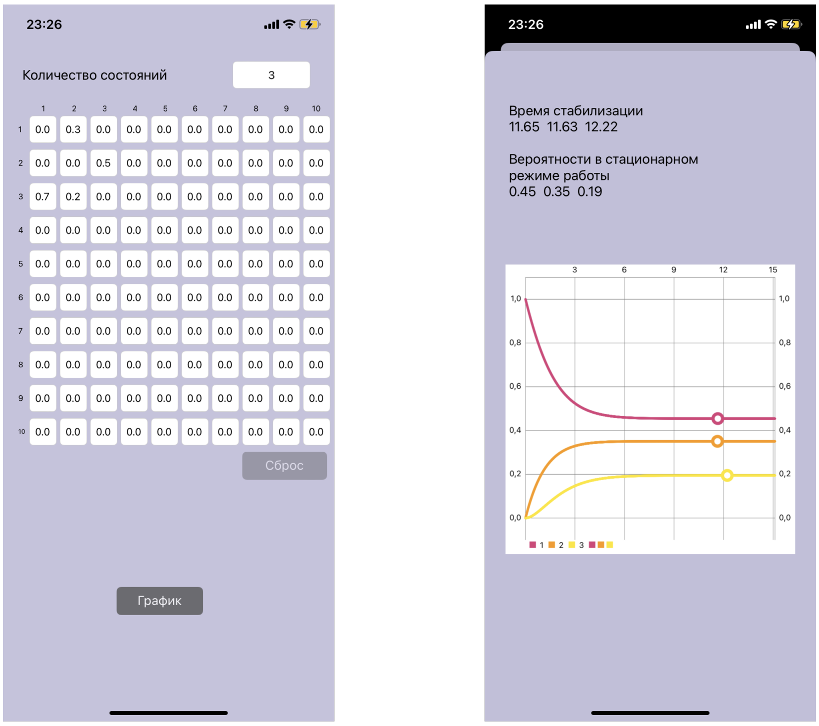

Рисунок 1 – Пример работы программы для 3 состояний

Проверим результаты, приведенные на рисунке выше. Составим систему линейных уравнений для определения вероятностей в стационарном режиме: составим уравнения Колмогорова и приравняем левые части к 0, а также добавим условие нормировки.

$$\begin{cases} 0 = 0.7 \cdot P_3 - 0.3 \cdot P_1 \\ 0 = 0.3 \cdot P_1 + 0.2 \cdot P_3 - 0.5 \cdot P_2 \\ 0 = 0.5 \cdot P_2 - 0.7 \cdot P_3 - 0.2 \cdot P_3 \\ P_1 + P_2 + P_3 = 1 \end{cases} \tag{1}$$

$$\begin{cases} P_1 = \frac{5}{11} \\ P_2 = \frac{27}{77} \\ P_3 = \frac{15}{77} \end{cases} \tag{2}$$

Вычисленные значения совпадают с результатами программы.

На рисунке 2 представлен пример работы для 5 состояний.

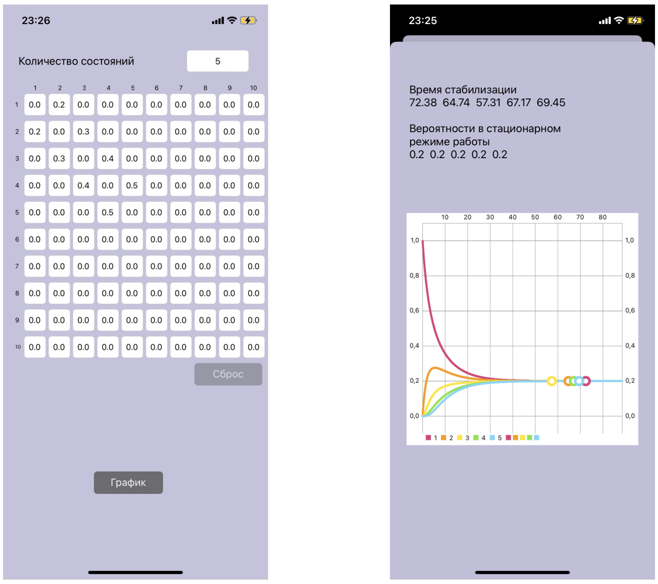

Рисунок 2 – Пример работы программы для 5 состояний

На рисунке 3 представлен пример работы для 10 состояний.

Рисунок 3 – Пример работы программы для 10 состояний

4 Код программы

Класс Model, используемый для расчетов и построения графиков, приведен в листинге 1 (используемый язык – Swift).

Листинг 1 – Класс Model

```
class Model {
    var nStates = 0
    var matrix = [[Double]]()
    var pArr = [Double]()
    var tStableArr = [Double]()
    let step = 0.01
    let stabEpsilon = 1e-5
    let zeroEpsilon = 1e-8

    var chartsData = [[ChartDataEntry]]()
    var stabPoints = [ChartDataEntry]()

    init(nStates: Int, matrix: [[Double]]) {
        self.nStates = nStates
        self.matrix = matrix
        self.pArr = Array(repeating: 0.0, count: nStates)
        self.tStableArr = Array(repeating: 0.0, count: nStates)
        self.chartsData = Array(repeating: [ChartDataEntry](), count:
nStates)
        self.stabPoints = Array(repeating: ChartDataEntry(), count:
nStates)

        pArr[0] = 1.0
    }

    func emulate() {
        var deltaProbArray = Array(repeating: 0.0, count: nStates)
        deltaProbArray[0] = 2 * stabEpsilon

        var currentT: Double = step
        while !isModelStabelized(deltaProbArray) {
            setChartCordinates(currentT, pArr)

            deltaProbArray = Array(repeating: 0.0, count: nStates)
```

```
            var PderivativeArr = Array(repeating: 0.0, count: nStates)

            for i in 0..<nStates {
                for j in 0..<nStates {
                    let probDensityToAdd = matrix[j][i] * pArr[j] - matrix[i][j] * pArr[i]
                    PderivativeArr[i] += probDensityToAdd
                    deltaProbArray[i] += probDensityToAdd * step
                }
                pArr[i] += deltaProbArray[i]
            }

            isSomeStatesStabelized(currentT, PderivativeArr)

            currentT += step
        }

        getStabPoints()
    }

    private func isModelStabelized(_ arr: [Double]) -> Bool {
        for i in 0..<arr.count {
            if arr[i] > zeroEpsilon { return false }
        }
        return true
    }

    private func setChartCordinates(_ currentT: Double, _ arr: [Double]) {
        for i in 0..<nStates {
            chartsData[i].append(ChartDataEntry(x: currentT, y: arr[i]))
        }
    }

    private func isSomeStatesStabelized(_ currentT: Double, _ klmArr: [Double]){
        for i in 0..<nStates {
            if abs(klmArr[i]) < stabEpsilon && tStableArr[i] == 0 {
                tStableArr[i] = currentT
            } else {
                if abs(klmArr[i]) > stabEpsilon && tStableArr[i] != 0 {
```

```
                    tStableArr[i] = 0
                }
            }
        }
    }

    private func getStabPoints() {
        for i in 0..<nStates {
            stabPoints[i] = ChartDataEntry(x: tStableArr[i], y: pArr[i])
        }
    }
}
```