Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Моделирование

ОТЧЕТ

по лабораторной работе № 3

на тему

ПОСТРОЕНИЕ И ИССЛЕДОВАНИЕ ИМИТАЦИОННОЙ МОДЕЛИ ДИСКРЕТНО-СТОХАСТИЧЕСКОЙ СМО,

ВАРИАНТ № 5

Студенты: А.В. Гуринович

И.В. Клишевский

Проверила: Ю.О. Герман

МИНСК 2022

# 1. Цель работы

Изучить методы имитационного моделирования поведения дискретно-стохастической СМО.

## 2. Краткие теоретические сведения

## 2.1 Линейный конгруэнтный алгоритм

Данный алгоритм может быть выражен следующей формулой:

Новое случайное число формируется из суммы константы с и произведения предыдущего случайного числа и константы a. Обе константы должны быть простыми числами.

Для получения чисел в определённом диапазоне результат предыдущей операции делится по модулю на константу m, которая в случае определения вероятностей должна быть равна 101, чтобы получить числа в диапазоне от 0 до 100 включительно.

Методом подбора установлено, что константы a и с равные 911 и 733 соответственно и начальное число 32 позволяют добиться близкого к равномерному распределяя.

Просчитаем несколько шагов генерации псевдослучайных чисел с заданными выше параметрами.

Теперь, получив первое число последовательности, подставим его в формулу для получения второго, такое действие означает, что формула является рекуррентной, то такой, в которой для выражения следующего члена последовательности используются предыдущие челны этой же последовательности. Продолжим последовательность:

Используем эту последовательность в подразделе 2.3.

## 2.2 Представление участков переходов состояний на единичном отрезке

Рассмотрим матрицу переходных состояний 3.1 из третьего раздела. Пусть система находится в состоянии S0, тогда она имеет вероятности соответственно 0.6, 0,3 и 0.1 для отсутствия перехода в другое состояние, перехода в состояние S1 и перехода в состояние S2 соответственно, тогда можно представить в виде участков отрезка от нуля до единицы, где части равны вероятностям переходов, то есть поровый отрезок будет находится в пределах [0; 0.6], второй отрезок – (0.6; 0.6 + 0.3], проведя операцию сложения получим – (0.6; 0.9], а последний – (0.9; 1].

На рисунках полужирным шрифтом обозначено описываемое им состояние.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S0** | | | | | S1 | | | | S2 |
| 0 |  |  |  | 0.6 | |  |  | 0.9 1 | |

Рисунок 2.2.1 – единичный отрезок с обозначением участков переходов из состояния S0

Проведём аналогичную операцию для состояний S1 и S2 и изобразим участки на единичных отрезках.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S0 | | | **S1** | | | | S2 | | |
| 0 |  | 0.3 | |  |  | 0.7 | |  | 1 |

Рисунок 2.2.2 – единичный отрезок с обозначением участков переходов из состояния S1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S0 | | S1 | | **S2** | | | | | |
| 0 | 0.4 | | 0.6 | |  |  |  |  | 1 |

Рисунок 2.2.3 – единичный отрезок с обозначением участков переходов из состояния S2

## 2.3 Пример последовательности переходов

Используем последовательность псевдослучайных чисел из раздела 2.1, приведём её к вероятностям, то есть поделим на 100, получим: 0.9, 0.04, 0.34, 0.94, 0.12, 0.5, 0.25, 0.76.

Возьмём за начальное состояние S0, тогда имея вероятность 0.9 мы попадаем в участок перехода к S1, так как он включает 0.9, а участок перехода к S2 начинается от 0.9 не включительно.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S0** | | | | | S1 | | | | | S2 |
| 0 |  |  |  | 0.6 | |  |  | | 0.9 1 | |
|  |  |  |  |  |  |  |  | **↑** | | |

Рисунок 2.3.1 – первый шаг примера последовательности переходов

На втором шаге возвращаемся в состояние S0. Продолжим операцию до того, как закончится последовательность псевдослучайных чисел.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S0 | | | **S1** | | | | S2 | | |
| 0 **0.04** |  | 0.3 | |  |  | 0.7 | |  | 1 |
| **↑** |  |  |  |  |  |  |  |  |  | |

Рисунок 2.3.2 – второй шаг примера последовательности переходов

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S0** | | | | | S1 | | | | | S2 |
| 0 |  | **0.34** |  | 0.6 | |  |  | | 0.9 1 | |
|  |  | **↑** |  |  |  |  |  |  | | |

Рисунок 2.3.3 – третий шаг примера последовательности переходов

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S0** | | | | | S1 | | | | | S2 |
| 0 |  |  |  | 0.6 | |  |  | | 0.9 **0.94** 1 | |
|  |  |  |  |  |  |  |  | **↑** | | |

Рисунок 2.3.4 – четвёртый шаг примера последовательности переходов

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S0 | | S1 | | **S2** | | | | | |
| 0 **0.12** | 0.4 | | 0.6 | |  |  |  |  | 1 |
| **↑** |  |  |  |  |  |  |  |  | | |

Рисунок 2.3.5 – пятый шаг примера последовательности переходов

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S0** | | | | | S1 | | | | | S2 |
| 0 |  |  | **0.5** | 0.6 | |  |  | | 0.9 1 | |
|  |  |  | **↑** |  |  |  |  |  | | |

Рисунок 2.3.6 – шестой шаг примера последовательности переходов

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S0** | | | | | S1 | | | | | S2 |
| 0 | **0.25** |  |  | 0.6 | |  |  | | 0.9 1 | |
|  | **↑** |  |  |  |  |  |  |  | | |

Рисунок 2.3.7 – седьмой шаг примера последовательности переходов

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S0** | | | | | S1 | | | | | S2 |
| 0 |  |  |  | 0.6 | | **0.76** |  | | 0.9 1 | |
|  |  |  |  |  |  | **↑** |  |  | | |

Рисунок 2.3.8 – восьмой шаг примера последовательности переходов

Теперь можно высчитать количество попаданий: пять раз в S0 (не включая начальное состояние), два раза в S1 (учитывая попадание на восьмом шаге) и один раз в S2. Теперь можно посчитать устоявшееся состояние системы: поделить количество попаданий для каждого объекта на изначальный размер последовательности псевдослучайных чисел.

Однако при малом количество итерации ещё нельзя говорить о устоявшемся состоянии, рекомендуется проводить минимум 100 итераций, при этом процесс переходов нужно прекратить при условии малого изменения устоявшихся состояний между ними.

# 3. Задание

Для всех вариантов в задании необходимо выполнить не менее 100 итераций.

Пусть матрица переходных вероятностей P:

Таблица 3.1 – матрица переходных вероятностей

|  |  |  |  |
| --- | --- | --- | --- |
|  | S0 | S1 | S2 |
| S0 | 0.6 | 0.3 | 0.1 |
| S1 | 0.3 | 0.4 | 0.3 |
| S2 | 0.2 | 0.2 | 0.6 |

3.1 Найти установившиеся вероятности состояний системы: P0, P1, P2 методом имитационного моделирования.

3.2 Рассчитать на основе моделирования число шагов до попадания поглощающее состояние для матрицы вероятностей переходов.

Таблица 3.2 – матрица вероятностей переходов

|  |  |  |  |
| --- | --- | --- | --- |
|  | S0 | S1 | S2 |
| S0 | 1.0 | 0 | 0 |
| S1 | 0.3 | 0.4 | 0.3 |
| S2 | 0.2 | 0.4 | 0.4 |

# 4. Ход работы

Для выполнения работы написана программа с графическим интерфейсом для ОС macOS на языке Swift. Из-за неточности переменных с плавающей точкой вероятности представлены в виде целых чисел, которые могут принимать значение от 0 до 100.

Для описания состояния, вероятностей переходов из него и счётчиков попаданий создан класс Entity:

class Entity {

var transitionsProbabilities: [Int]

var probabilitiesRanges: [ClosedRange<Int>] = []

var hitsCounter: Int = 0

init(transitionsProbabilities: [Int]) {

self.transitionsProbabilities = transitionsProbabilities

var currentProbability = 0

for probability in transitionsProbabilities {

//Ranges in form []

probabilitiesRanges.append(currentProbability...(probability + currentProbability))

currentProbability += probability

}

}

///Finds next state from probability

func next(probability: Int) -> Int {

for index in probabilitiesRanges.indices {

if probabilitiesRanges[index].contains(probability) {

return index

}

}

return probabilitiesRanges.count - 1

}

func isAbsorbing(stateNumber: Int) -> Bool {

return transitionsProbabilities[stateNumber] == 100

}

//MARK: Hits

func hit() {

hitsCounter += 1

}

func cleanHitsCounter() {

self.hitsCounter = 0

}

}

Для описания совокупности состояний и переходов из них создан класс Entities:

class Entities {

var currentState: Int

var currentEntity: Entity {

entities[currentState]

}

var entities: [Entity]

init(entities: [Entity], initialState: Int) {

self.entities = entities

self.currentState = initialState

}

}

## 4.1 Найти установившиеся вероятности состояний системы: P0, P1, P2 методом имитационного моделирования.

Для нахождения вероятности состояний системы необходимо воспользоваться методом из разделов 2.2 и 2.3, алгоритм приведён ниже:

func calculateStableState(from intArray: [Int]) -> [(Int, Double)] {

let randomDoubles = intArray

for probability in randomDoubles {

currentState = currentEntity.next(probability: probability)

entities[currentState].hit()

}

return entities.map {($0.hitsCounter, (Double($0.hitsCounter) / Double(randomDoubles.count)))}

}

## 4.2 Рассчитать на основе моделирования число шагов до попадания поглощающее состояние для матрицы вероятностей переходов.

Для расчёта среднего числа шагов до попадания системы в устойчивое состояние нужно сначала вычислить эти состояния, для этого используется следующий алгоритм:

func notAbsorbingAndAbsorbingIndexes() -> (notAbsorbingEntitiesIndexes: [Int], absorbingEntitiesIndexes: [Int]) {

var notAbsorbingEntitiesIndexes: [Int] = []

var absorbingEntitiesIndexes: [Int] = []

for index in entities.indices {

if !entities[index].isAbsorbing(stateNumber: index) {

notAbsorbingEntitiesIndexes.append(index)

} else {

absorbingEntitiesIndexes.append(index)

}

}

return (notAbsorbingEntitiesIndexes, absorbingEntitiesIndexes)

}

После этого используется следующий алгоритм: каждого из непоглощающих состояний для одинаковой последовательности псевдослучайных чисел вычисляется проводится процесс переходов до попадания в поглощающее состояние, в случае попадания в поглощающее состояние система возвращается к изначальному состоянию и проводит цикл заново, но последовательность псевдослучайных чисел при этом не сбрасывается, цикл продолжается пока не закончатся числа в последовательности, после чего аналогичная операция проводится для следующего непоглощающего состояния.

При каждом попадании в поглощающее состояние количество шагов записывается в соответствующий непоглощающему состоянию массив, после завершения цикла вычисляется среднее число шагов для начала из каждого непоглощающего состояния.

Код алгоритма приведён ниже:

func calculateAbsorbingStateSteps(from intArray: [Int]) -> [Double] {

let randomDoubles = intArray

var hitsBeforeAbsorbingState: [Int: [Int]] = [:]

let (notAbsorbingEntitiesIndexes, absorbingEntitiesIndexes) = notAbsorbingAndAbsorbingIndexes()

for index in notAbsorbingEntitiesIndexes {

hitsBeforeAbsorbingState[index] = []

var localRandomDoubles = randomDoubles

repeat {

currentState = index

repeat {

let nextEntityIndex = currentEntity.next(probability: localRandomDoubles.removeFirst())

currentEntity.hit()

if absorbingEntitiesIndexes.contains(nextEntityIndex) {

let hits = entities.map { $0.hitsCounter }.reduce(0, +)

hitsBeforeAbsorbingState[index]?.append(hits)

for entity in entities {

entity.cleanHitsCounter()

}

break

} else {

currentState = nextEntityIndex

}

} while (!localRandomDoubles.isEmpty)

} while (!localRandomDoubles.isEmpty)

}

var resultArray: [Double] = []

for index in entities.indices {

if notAbsorbingEntitiesIndexes.contains(index) {

let steps = hitsBeforeAbsorbingState[index]!.reduce(0, +)

let averageSteps = Double(steps) / Double(hitsBeforeAbsorbingState[index]!.count)

resultArray.append(averageSteps)

} else {

resultArray.append(0)

}

}

return resultArray

}

## 5. Вывод

Изучены методы имитационного моделирования поведения дискретно-стохастической СМО.

## 5.1 Что такое марковский процесс?

Рассматривается некоторая система S, которая в процессе функционирования может находится в различных состояниях. Состояния системы меняются случайным образом, следовательно, последовательность состояний системы образует случайный процесс.

Для того, чтобы процесс был марковским необходимо, чтобы для него выполнялось свойство – отсутствие последействия, то есть, для любого момента времени tj, вероятность любого состояния системы при t > tj зависит только от её состояния при t = tj и не зависит от того, как и когда система пришла в это состояние.

## 5.2 Опишите общую идею имитационного моделирования дискретно-стохастической СМО?

Имитация работы системы с использованием параметров полученных случайно.

## 5.3 Как вы понимаете установившиеся вероятности состояний системы?

Устоявшееся вероятности – это те вероятности состояния системы, к которым она стремиться с увеличением количества переходов.

## 5.4 Как оценить число шагов моделирования?

Минимальное количество шагов моделирования составляет 100. Для определения окончания шагов моделирования можно использовать критерий изменения устоявшейся вероятности между шагами. К примеру, если изменение между шагом k и k-1 составляется менее 0.01 – прекратить моделирование.

## 5.5 Как оценить число шагов до попадания в поглощающее состояние на основе имитационного моделирования?

Аналогично обычному моделированию, однако прекращать его в момент попадания в поглощающее состояние. Также необходимо проводить моделирования с началом из всех непоглощающих состояний отдельно. Проведя такое моделирование многократно можно судить о среднем количестве шагов до попадания в погашающее состояние.