Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ОТЧЕТ

по лабораторной работе №3

по теме

«Построение и исследование имитационной модели дискретно – стохастической СМО»

|  |  |
| --- | --- |
| Выполнили:  студент гр. 950501  Лабецкий А. А.  Кумище В. Г. | Проверила:  Герман Ю. О. |

Минск 2022

**1. Цель работы**

Изучить методы имитационного моделирования поведения дискретно-стохастической СМО.

**2. Задание**

1. Найти установившиеся вероятности состояний системы: P0, P1, P2 методом имитационного моделирования.
2. Рассчитать на основе моделирования число шагов до попадания в поглощающее состояние для матрицы вероятностей переходов
3. **Теоретические сведения**

Рассмотрим матрицу переходных вероятностей из предыдущей лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
|  | S0 | S1 | S2 |
| S0 | 0.1 | 0.2 | 0.7 |
| S1 | 0.4 | 0.3 | 0.3 |
| S2 | 0.6 | 0.2 | 0.2 |

Сумма вероятностей по каждой строке равна 1. Графически изобразим состояния и переходы между ними

0.2

0.4

0.7

0.2 0.3

0.6

Пусть в начальный момент система находится в состоянии S0. Из этого состояния система может перейти в состояние S1 с вероятностью 0.2, в состояние S2 с вероятностью 0.7 и остаться в состоянии S0 с вероятностью 0.1. Таким образом, нужно разыграть равномерно распределенное случайное число в диапазоне от 0 до 1. Мы предполагаем равномерное распределение, поскольку иное не оговорено. Чтобы выбрать новое состояние, в которое можно перейти из S0, разобьем отрезок единичной длины на части, длины которых пропорциональны вероятностям переходов (см. рисунок)

S0 S1 S2

0 0.1 0.3 1

Этот общий принцип назовем принципом выбора состояния.

Первый интервал длиной 0.1 соответствует переходу из S0 в S0. Второй интервал длиной 0.3 − 0.1= 0.2 соответствует переходу из S0 в S1; наконец, третий интервал, длиной 1 – 0.3 = 0.7 соответствует переходу из S0 в S2. Нам нужен алгоритм для получения псевдослучайных равномерно распределенных чисел. Воспользуемся теоретическим материалом. Относительно простой метод генерации случайных чисел — линейный конгруэнтный алгоритм. Выраженный в символьном виде, он представляет собой следующее модифицированное выражение:

X(i) = (a \* X(i-1) + c) mod m

«Новое случайное число является предыдущим случайным числом, умножаемым на константу a, после чего добавляем константу c и над результатом выполняется операция деления по модулю константы m. При этом значение a \* X(i-1) округлим, отбросив дробную часть».

Зададим m = 101. Тогда остаток от деления будет лежать в диапазоне от 0 до 100. Единственное, что потребуется – выполнять округление величины a \* X(i-1) + c до целого (например, просто отбрасывая дробную часть). Таким образом, числа Xi будут попадать в интервал от 0 до 100. Из этих чисел можно получить вероятность простым делением на 100. Итак, возьмем, например, Х(1) =37, a= 131, c = 1021. Обеспечим, чтобы a и c были взаимно простыми. Генерируем следующие случайные величины

X(2)=[131\*37+1021)] mod 100 = 68

X(3)=[131\*68+1021)] mod 100 = 29

X(4)=[131\*29+1021)] mod 100 = 20

X(5)=[131\*20+1021)] mod 100 = 41

X(6)=[131\*41+1021)] mod 100 = 92

и т.д.

Переходя к вероятностям, получаем последовательность:

0.68; 0.29; 0.20; 0.41; 0.92 …

Теперь используем принцип выбора состояния по вероятности. В самом начале система находится в состоянии S0. Разыграна вероятность 0.68. Эта вероятность соответствует диапазону [0.3, 1] (попадает в данный диапазон), а сам диапазон соответствует состоянию S2.

S0 S1 S2

0 0.1 0.3 1

Итак, на первом шаге система смоделировала переход из S0 в S2. Далее поступаем по аналогии. Теперь рассматриваем переходы из состояния S2. Откладываем следующие интервалы:

S0 S1 S2

0 0.6 0.8 1

Следующая сгенерированная вероятность в случайной последовательности после 0.68 есть 0.29. Она попадает в отрезок [0;0.6], соответствующий переходу в состояние S0. Таким образом, система смоделировала возврат в S0. Наш очередной шаг опять использует рисунок

S0 S1 S2

0 0.1 0.3 1

Теперь разыграно значение вероятности, равное 0.2. Это значение соответствует переходу в S1. На этот раз мы имеем следующее разбиение отрезка

S0 S1 S2

0 0.4 0.7 1

Разыграна вероятность 0.41. Эта вероятность соответствует переходу в S1. Таким образом, были смоделированы следующие переходы:

S0 -> S2 -> S0 -> S1 -> S1 -> …

Этот процесс, теоретически можно продолжать до бесконечности. Однако вероятности состояний стремятся к установившимся значениям. Как найти эти установившиеся значения (предельные значения)? Просто: подсчитаем, сколько раз система была в состояний S0 (пусть это будет N0), сколько раз она была в состоянии S1 (число N1) и в состоянии S2 (число N2). Тогда вероятность каждого состояния равна соответствующему частному от деления Ni/N, где N = N0+N1+N2. Собственно, вычисление этих установившихся вероятностей и есть цель настоящей работы.

Ясно, что процесс переходов в новые состояния следует прекратить, когда установившиеся вероятности изменяются не более,чем на наперед задаваемую малую величину d (например, d=0.01). При этом минимальное число переходов, как указывается в литературе, должно быть не менее 100.

**4. Выполнение работы**

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

Исходный код программы:

int previousNumber = 37;

int a = 131;

int m = 100;

int c = 1021;

int[] states = new int[3];

State lastState = State.S0;

double p0 = 0, p1 = 0, p2 = 0;

double diff = 0.0001;

double[,] p = new double[,]

{

{ 0.1, 0.8, 0.1 },

{ 0.3, 0.5, 0.2 },

{ 0.7, 0.2, 0.1 },

};

int iterations = 100;

int counter = 0;

while(true)

{

double randValue = GenerateRandom();

double value = 0;

for (int i = 0; i < 3; i++)

{

if (randValue <= (p[(int)lastState, i] + value))

{

states[i]++;

lastState = (State)i;

break;

}

value += p[(int)lastState, i];

}

counter++;

if (counter >= iterations)

{

if (

p0 - ((double)states[0] / counter) < diff &&

p1 - ((double)states[1] / counter) < diff &&

p2 - ((double)states[2] / counter) < diff)

{

p0 = (double)states[0] / counter;

p1 = (double)states[1] / counter;

p2 = (double)states[2] / counter;

break;

}

}

p0 = (double)states[0] / counter;

p1 = (double)states[1] / counter;

p2 = (double)states[2] / counter;

}

Console.WriteLine($"P0 = {p0}, P1 = {p1}, P2 = {p2}");

Console.WriteLine($"Iterations: {counter}");

double GenerateRandom()

{

previousNumber = (a \* previousNumber + c) % m;

return (double)previousNumber / 100;

}

enum State

{

S0,

S1,

S2,

}

Пример вывода программы представлен на рисунке 4.1.

****

Рисунок 4.1

**4.2 Рассчитать на основе моделирования число шагов до попадания в поглощающее состояние для матрицы вероятностей переходов**

Исходный код программы:

int previousNumber = 37;

int a = 131;

int m = 101;

int c = 1021;

State lastState = State.S0;

double[,] p = new double[,]

{

{ 0.2, 0.1, 0.7 },

{ 0.0, 1, 0 },

{ 0.3, 0.6, 0.1 },

};

int counter = 0;

while(true)

{

double randValue = GenerateRandom();

double value = 0;

for (int i = 0; i < 3; i++)

{

if (randValue <= (p[(int)lastState, i] + value))

{

lastState = (State)i;

break;

}

value += p[(int)lastState, i];

}

if (lastState == State.S1)

{

Console.WriteLine(counter + 1);

break;

}

counter++;

}

double GenerateRandom()

{

previousNumber = (a \* previousNumber + c) % m;

return (double)previousNumber / 100;

}

enum State

{

NoState = -1,

S0,

S1,

S2,

}

Пример вывода программы представлен на рисунке 4.2.

****

Рисунок 4.2

**Выводы**

В ходе лабораторной работы были исследованы методы имитационного моделирования поведения дискретно стохастической СМО. Была построена и исследована модель дискретно-стохастической СМО, с её помощью найдены установившиеся вероятности состояний системы и рассчитано число шагов до перехода в поглащающее состояние.