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 №5

«Построение и исследование имитационной модели непрерывно – стохастической СМО»

Вариант 2

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

#### Минск 2022

1. **Цель работы**

Целью работы является:

- изучить методы имитационного моделирования поведения непрерывно-стохастической СМО.

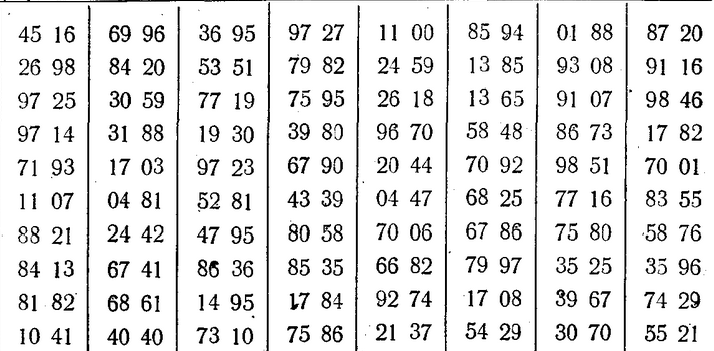
1. **Краткое теоретическое введение**

Будем рассматривать СМО с одним прибором. Поток заявок на обслуживание задается в форме экспоненциального закона:

*F(t)* = 1−*e-*λ*t*.

F(t) определяет вероятность того, что хотя бы одна заявка придет за время t. Мы видим, что при *t* →∝ эта вероятность стремится к 1. В формуле выше λ - интенсивность входного потока заявок (например, число заявок в секунду)

Итак, нужно показать, как генерировать случайные числа с показательным законом распределения. Воспользуемся методом обратных функций. Сначала сгенерируем последовательность равномерно распределенных чисел в интервале от 0 до 1. Можно использовать метод Лемера или линейный конгруэнтный метод, описанный в предыдущей лабораторной работе. Мы просто используем заготовленные таблицы, которые можно скачать в Интернет. Вот фрагмент такой таблицы



Можно брать двузначные случайные числа по строкам таблицы. Например:

45, 16, 69, 96, 36, 95, 97, 27 и т.д. (1)

Чтобы перевести их в вероятность, достаточно разделить число на 100. Итак, у нас есть последовательность значений функции F(t) – (1)

Возьмем первую вероятность из (1) – 0.45. Тогда имеем:

F(t) = 1−e-λt = 0.45.

Отсюда:

*e*-λ*t* = 0.55.

Интенсивность нам должна быть задана заранее. Пусть:

λ = 0.1 (сек-1).

Получаем:

e-0.1t =0.55.

Логарифмируем:

−0.1t = ln 0.55 = -0.597;

t = 5.97 ≈ 6.

Итак, первая заявка придет через 6 сек. Аналогично разыгрываем момент поступления второй заявки. Берем вторую вероятность – 0.16. Решаем уравнение:

1−e-λt = 0.16.

Отсюда:

e-λt =0.84;

e-0.1t =0.84;

−0.1t = ln 0.84 = -0.17;

t = 1.7 ≈ 2.

Вторую заявку ожидаем через 2 сек. По аналогии рассчитываем моменты поступления остальных заявок на обслуживание. Эти моменты времени имею экспоненциальное распределение. Вообще, распределение может быть любым – главное, чтобы была задана соответствующая функция распределения F(t).

Теперь обратимся к процессу обслуживания. Нам нужно заранее знать функцию распределения времени обслуживания. Пусть в нашем примере это будет распределение Вейбулла. Формула такова:

G (t) = 1− exp[−(t/μ)k].

Здесь как и ранее: μ − интенсивность потока обслуживаня заявок, t − время, k − коэффициент формы, например, возьмем k=1. Пусть:

μ = 2 (сек-1).

Мы будем использовать ту же последовательность равномерно распределенных случайных чисел, представляющих вероятности. Итак, первая вероятность равна 0.45.

Имеем уравнение:

0.45= 1− exp[−(t/2)].

Отсюда:

0.55 = exp[−(t/2);

Ln (0.55) = -t/2;

0.597 \*2 = t;

t = 1.2.

Итак, первая заявка будет обслужена за время, равное 1.2 сек. Аналогичные вычисления проделаем со второй заявкой.

0.16 = exp[−(t/2);

Ln (0.16) = -t/2;

1.83 \*2 = t;

t = 3.7.

Вторая заявка будет обслужена приблизительно за 3.7 сек. И т.д.

Итак, общий механизм описан. Нужно:

1. Сгенерировать последовательность равномерно распределенных вероятностей от 0 до 1. Можно использовать готовые таблицы.

2. По этим вероятностям и заданному закону F(t) времен поступления заявок в систему сгенерировать последовательность времен t1, t2, …, tz в соответствии с законом распределения F(t).

3. По этим вероятностям и заданному закону G(t) времен обслуживания заявок в системе сгенерировать последовательность времен τ1, τ2, …, τz в соответствии с законом распределения G (t).

4. На временных осях отложить соответствующие времена, как показано ниже:

t–время поступления

t=6 t=8

t=7.2 t=11.7

t–время обслуживания

Заметим, что в интервале от 7.2 до 8 система простаивает – нет заявок, ничего не обслуживается. Кроме того, возможно появление очереди на обслуживание (в нашем примере очередь не успела сформироваться).

1. **Задание по варианту**

Пусть заданы следующие законы распределения вероятностей времен поступления F(t), времени обслуживания G(t):

F(t) = t/10;

G (t) = 1−e−μt .

По результатам моделирования найти – среднее время обслуживания, среднее время пребывания заявки в системе, среднее число заявок в системе, процент загрузки обслуживающего прибора (канала).

1. **Выполнение**

**4.1 Листинг кода**

int x1 = 37;

int a = 131;

int c = 1021;

int m = 100;

double d = 0.001;

double lambda = 0.1;

int mu\_param = 2;

int currentState = 0;

int prevNumber = x1;

List<double> randomProbabilities = new List<double>();

List<double> fList = new List<double>();

List<double> gList = new List<double>();

for (int i = 0; i < 100; i++)

{

var generatedNumber = RandomNumber(prevNumber);

prevNumber = generatedNumber;

randomProbabilities.Add((double)generatedNumber / 100);

fList.Add(F(randomProbabilities[i]));

gList.Add(G(randomProbabilities[i]));

}

Console.WriteLine("PROBABILITIES:");

foreach (var p in randomProbabilities)

{

Console.Write($"{p} ");

}

Console.WriteLine();

Console.WriteLine("F functions:");

foreach (var f in fList)

{

Console.Write($"{f} ");

}

Console.WriteLine();

Console.WriteLine("G functions:");

foreach (var g in gList)

{

Console.Write($"{g} ");

}

Console.WriteLine();

Console.WriteLine("Average service time:");

Console.WriteLine(AverageServiceTime(gList));

Console.WriteLine("Average staying time:");

Console.WriteLine(AverageStaying(AverageDowntime(Downtime(gList, fList)), AverageServiceTime(gList)));

Console.WriteLine("Load percentage:");

Console.WriteLine(LoadPercentage(fList, gList, Downtime(gList, fList)));

int RandomNumber(int prevNum)

{

return (a \* prevNum + c) % m;

}

double F(double p)

{

double log = Math.Log(1 - p) \* (-1);

return Math.Round(log / lambda);

}

double G(double p)

{

double log = Math.Log(1 - p) \* (-1);

double temp = Math.Sqrt(log) \* mu\_param;

return Math.Round(temp, 1);

}

double AverageServiceTime(List<double> gList)

{

return gList.Average();

}

List<double> Downtime(List<double> gList, List<double> fList)

{

List<double> downtime = new List<double>();

double delay;

for (int i = 0; i < fList.Count - 1; i++)

{

delay = gList[i] = fList[i + 1];

if (delay > 0)

{

downtime.Add(delay);

}

}

return downtime;

}

double AverageDowntime(List<double> downtime)

{

return downtime.Average();

}

double AverageStaying(double averageDowntime, double averageService)

{

return averageDowntime + averageService;

}

double LoadPercentage(List<double> fList, List<double> gList, List<double> downtime)

{

return gList.Sum() / fList.Sum() \* 100 + downtime.Sum();

}

**4.2 Результаты**

Первые десять значений F={ 11, 3, 2, 5, 25, 13, 18, 3, 32, 35 }.

Первые десять значений G={ 2.1, 1.2, 0.9, 1.5, 3.2, 2.3, 2.7, 1.1, 3.6, 3.7 }.

На временных осях отложим соответствующие времена:

1. **Вывод**

В ходе работы были изучены методы имитационного моделирования поведения непрерывно-стохастической СМО.