Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Системный анализ и машинное моделирование (САиММод)

ОТЧЕТ

По лабораторной работе № 3

Построение и исследование аналитической модели дискретно - стохастической системы массового обслуживания

Вариант № 9

Выполнил студент гр. 851006 П.Л.Мискевич

Проверил Н.И.Мельник

Минск, 2021

1. ЗАДАНИЕ
   1. Построить граф состояний P-схемы, представленной на рисунке 1.



Рисунок 1 – Исходная P-схема.

* 1. Раскрыть смысл кодировок состояний.
  2. Построить аналитическую модель по графу, решить ее и определить вероятности состояний.
  3. Рассчитать теоретические значения показателей эффективности.
  4. Построить имитационную модель для данной СМО.

Элементы:

|  |  |
| --- | --- |
|  | Источник с фиксированным временем ожидания выдачи заявки |
|  | Очередь на 1 заявку |
|  | Канал обработки заявок с вероятностью просеивания π1 и дисциплиной отбрасывания |
|  | Канал обработки заявок с вероятностью просеивания π2 |

Кодирование состояний:

– состояние источника заявок (1 или 2 – количество тактов до появления новой заявки в источнике);

– количество заявок в очереди;

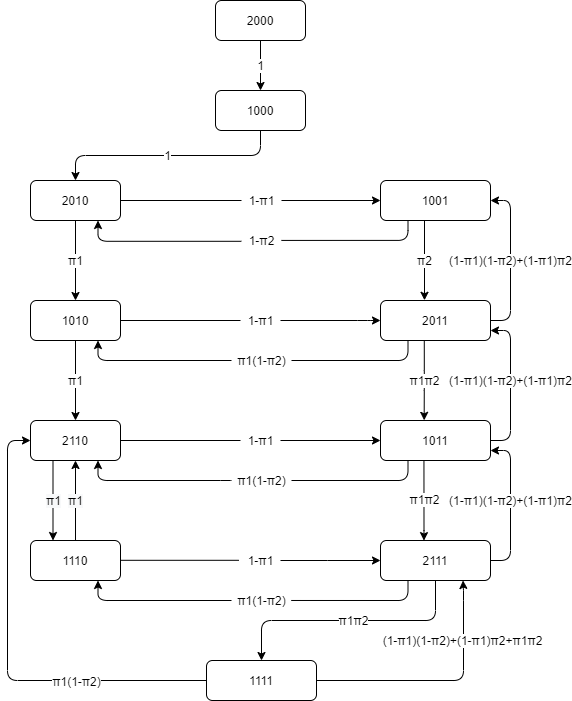
– состояние первого канала (0 – свободен, 1 – занят);

– состояние второго канала (0 – свободен, 1 – занят).

Общая кодировка состояний системы: .

Вероятности просеивания: ; .

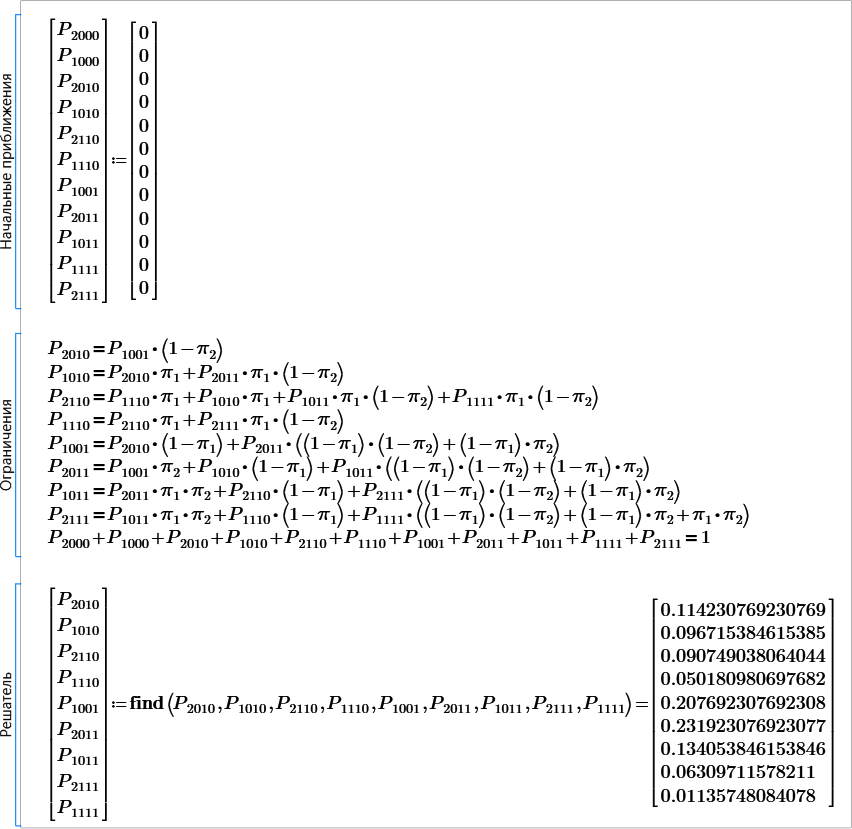
Граф состояний:



Решение системы в MathCad:







Произведем расчет теоретических значений показателей эффективности:

– абсолютная пропускная способность;

– средняя длина очереди;

– среднее число заявок, находящихся в системе;

– среднее время пребывания заявки в очереди;

– среднее время пребывания заявки в системе;

– вероятность отказа;

– относительная пропускная способность;

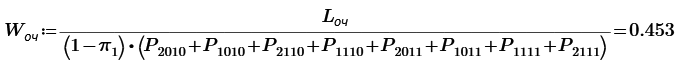
– коэффициент загрузки канала 1;

– коэффициент загрузки канала 2.

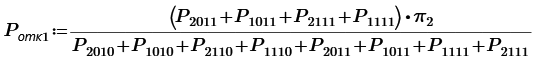


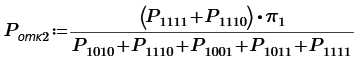












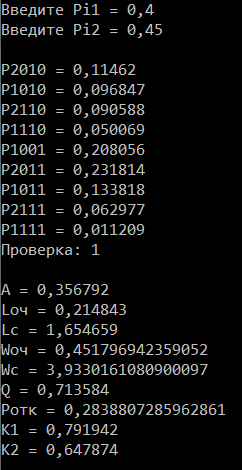








Пример работы имитационной модели:



Листинг программы:

using System;

const int N = 1000000;

string state = "2000";

double pi1, pi2;

do

{

Console.Write("Введите Pi1 = ");

pi1 = Convert.ToDouble(Console.ReadLine());

}

while (pi1 < 0 || pi1 > 1);

do

{

Console.Write("Введите Pi2 = ");

pi2 = Convert.ToDouble(Console.ReadLine());

}

while (pi2 < 0 || pi2 > 1);

Console.WriteLine();

Random random = new();

double currentPi1, currentPi2;

int P2000 = 0, P1000 = 0, P2010 = 0, P1010 = 0,

P2110 = 0, P1110 = 0, P1001 = 0, P2011 = 0,

P1011 = 0, P2111 = 0, P1111 = 0;

int J = 0, T1 = 0, T2 = 0,

FirstChannel = 0, SecondChannel = 0,

QueueLength = 0, RequestLength = 0,

Processed1Count = 0, Processed2Count = 0,

DeclinedCount = 0, GeneratedCount = 0;

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

{

currentPi1 = random.NextDouble();

currentPi2 = random.NextDouble();

switch (state)

{

case "2000":

P2000++;

J = 0;

T1 = 0;

T2 = 0;

state = new("1000");

break;

case "1000":

P1000++;

J = 0;

T1 = 0;

T2 = 0;

state = new("2010");

break;

case "2010":

P2010++;

J = 0;

T1 = 1;

T2 = 0;

if (currentPi1 > pi1) { state = new("1001"); Processed1Count++; }

if (currentPi1 <= pi1) { state = new("1010"); }

break;

case "1010":

P1010++;

J = 0;

T1 = 1;

T2 = 0;

if (currentPi1 > pi1) { state = new("2011"); Processed1Count++; }

if (currentPi1 <= pi1) { state = new("2110"); }

break;

case "2110":

P2110++;

J = 1;

T1 = 1;

T2 = 0;

if (currentPi1 > pi1) { state = new("1011"); Processed1Count++; }

if (currentPi1 <= pi1) { state = new("1110"); }

break;

case "1110":

P1110++;

J = 1;

T1 = 1;

T2 = 0;

if (currentPi1 > pi1) { state = new("2111"); Processed1Count++; }

if (currentPi1 <= pi1) { state = new("2110"); DeclinedCount++; }

break;

case "1001":

P1001++;

J = 0;

T1 = 0;

T2 = 1;

if (currentPi2 > pi2) { state = new("2010"); Processed2Count++; }

if (currentPi2 <= pi2) { state = new("2011"); }

break;

case "2011":

P2011++;

J = 0;

T1 = 1;

T2 = 1;

if ((currentPi1 <= pi1) && (currentPi2 <= pi2)) { state = new("1011"); }

if ((currentPi1 <= pi1) && (currentPi2 > pi2)) { state = new("1010"); Processed2Count++; }

if ((currentPi1 > pi1) && (currentPi2 <= pi2)) { state = new("1001"); Processed1Count++; DeclinedCount++; }

if ((currentPi1 > pi1) && (currentPi2 > pi2)) { state = new("1001"); Processed1Count++; Processed2Count++; }

break;

case "1011":

P1011++;

J = 0;

T1 = 1;

T2 = 1;

if ((currentPi1 <= pi1) && (currentPi2 <= pi2)) { state = new("2111"); }

if ((currentPi1 <= pi1) && (currentPi2 > pi2)) { state = new("2110"); Processed2Count++; }

if ((currentPi1 > pi1) && (currentPi2 <= pi2)) { state = new("2011"); Processed1Count++; DeclinedCount++; }

if ((currentPi1 > pi1) && (currentPi2 > pi2)) { state = new("2011"); Processed1Count++; Processed2Count++; }

break;

case "2111":

P2111++;

J = 1;

T1 = 1;

T2 = 1;

if ((currentPi1 <= pi1) && (currentPi2 <= pi2)) { state = new("1111"); }

if ((currentPi1 <= pi1) && (currentPi2 > pi2)) { state = new("1110"); Processed2Count++; }

if ((currentPi1 > pi1) && (currentPi2 <= pi2)) { state = new("1011"); Processed1Count++; DeclinedCount++; }

if ((currentPi1 > pi1) && (currentPi2 > pi2)) { state = new("1011"); Processed1Count++; Processed2Count++; }

break;

case "1111":

P1111++;

J = 1;

T1 = 1;

T2 = 1;

if ((currentPi1 <= pi1) && (currentPi2 <= pi2)) { DeclinedCount++; }

if ((currentPi1 > pi1) && (currentPi2 <= pi2)) { state = new("2111"); Processed1Count++; DeclinedCount++; }

if ((currentPi1 > pi1) && (currentPi2 > pi2)) { state = new("2111"); Processed1Count++; Processed2Count++; }

break;

}

if (state[0] == '1')

{

GeneratedCount++;

}

QueueLength += J;

FirstChannel += T1;

SecondChannel += T2;

RequestLength += J + T1 + T2;

}

Console.WriteLine($"P2000 = {(double)P2000 / N}");

Console.WriteLine($"P1000 = {(double)P1000 / N}");

Console.WriteLine($"P2010 = {(double)P2010 / N}");

Console.WriteLine($"P1010 = {(double)P1010 / N}");

Console.WriteLine($"P2110 = {(double)P2110 / N}");

Console.WriteLine($"P1110 = {(double)P1110 / N}");

Console.WriteLine($"P1001 = {(double)P1001 / N}");

Console.WriteLine($"P2011 = {(double)P2011 / N}");

Console.WriteLine($"P1011 = {(double)P1011 / N}");

Console.WriteLine($"P2111 = {(double)P2111 / N}");

Console.WriteLine($"P1111 = {(double)P1111 / N}");

Console.WriteLine($"Проверка: {(double)(P2000 + P1000 + P2010 + P1010 + P2110 + P1110 + P1001 + P2011 + P1011 + P2111 + P1111) / N}");

Console.WriteLine();

Console.WriteLine($"A = {(double)Processed2Count / N}");

Console.WriteLine($"Lоч = {(double)QueueLength / N}");

Console.WriteLine($"Lc = {(double)RequestLength / N}");

Console.WriteLine($"Wоч = {(double)QueueLength / Processed1Count}");

Console.WriteLine($"Wc = {((double)(QueueLength + FirstChannel) / Processed1Count) + ((double)SecondChannel / Processed2Count)}");

Console.WriteLine($"Q = {(double)Processed2Count / GeneratedCount}");

Console.WriteLine($"Pотк = {(double)DeclinedCount / GeneratedCount}");

Console.WriteLine($"K1 = {(double)FirstChannel / N}");

Console.WriteLine($"K2 = {(double)SecondChannel / N}");

Console.ReadLine();