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 работе №5

Вариант 38

Выполнил

студент: гр. 851004 Пашкевич А.Л.

Проверил: Мельник Н.И.

Минск 2021

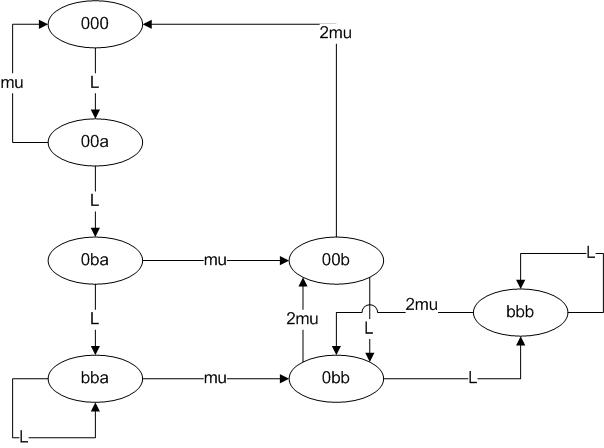
# Постановка задачи

В СМО вида M/M/1/2 интенсивность обслуживания µ, если заявка попала в канал, минуя очередь, или 2µ, если она попала в канал, побывав в очереди. Найти значения А, Wc, Wоч. =0,8, =0,9.

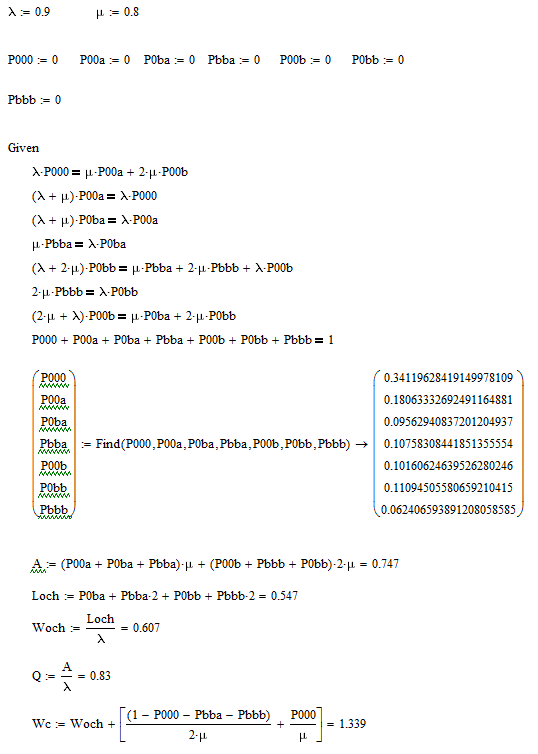
# Ход выполнения работы

Введём два рода заявок: *a* – заявки, не побывавшие в очереди, *b* – заявки, прошедшие через очередь. Так как очередь рассчитана на две заявки, то состояния системы будут кодироваться как (q1, q2, c), где q1 – первое место в очереди, q2 – второе место в очереди, c – канал.

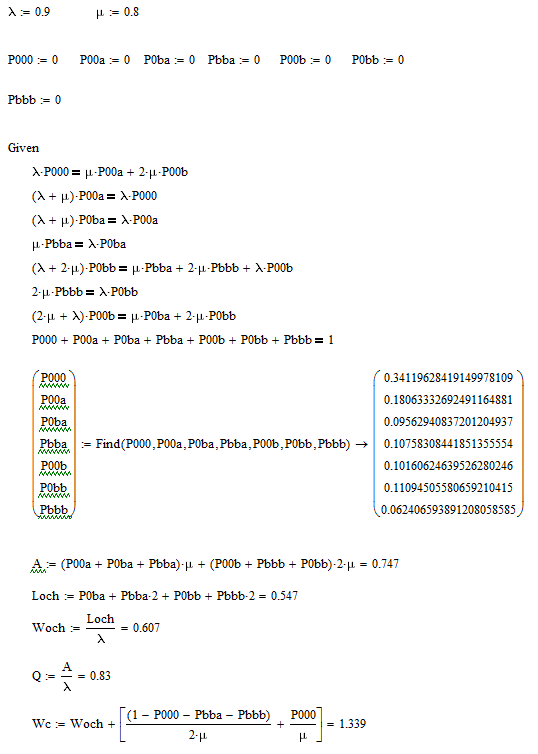
Диаграмма интенсивности переходов:



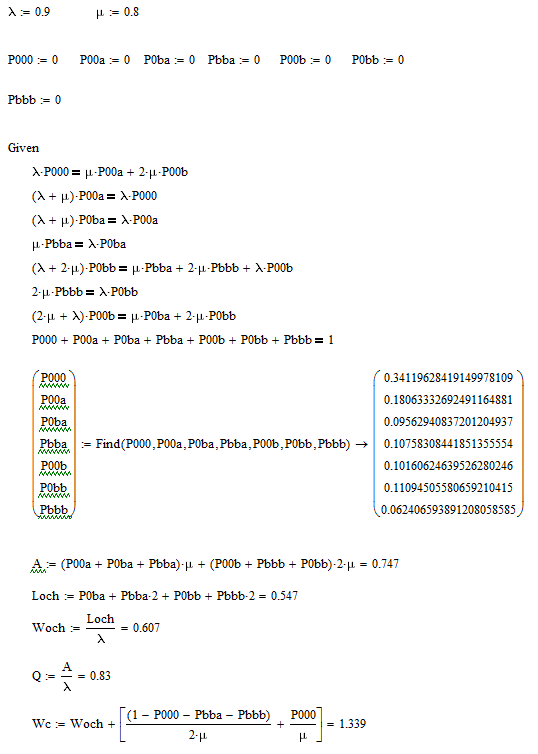
Решение в маткаде, исходные данные:



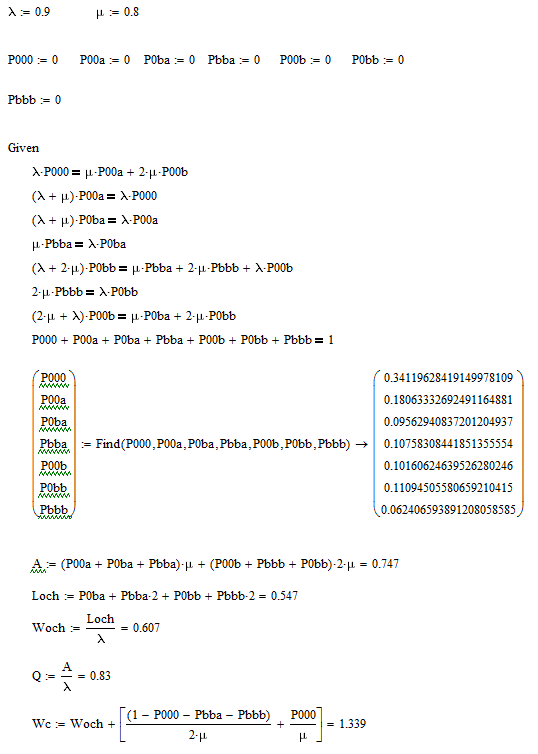
Система уравнений:



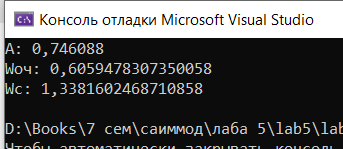
Вероятности состояний системы:



Расчёт показателей эффективности (A – абсолютная пропускная способность, Woch – среднее время пребывания заявки в очереди, Wc – среднее время пребывания заявки в системе):



Результаты иммитационного моделирования:



Листинг иммитационной модели:

using System;

using System.Linq;

namespace lab5

{

public enum CMOState

{

Free,

Task

}

public struct CMOResult

{

public int countOfReceivedTask;

public int countOfProcessedTask;

public double allTimeInQueue;

public double allTimeInChanel;

}

public class CMO

{

public CMOState[] queue;

public CMOState chanelState;

public double timeBeforeGetTask;

public double timeUntilEndOfTask;

public CMO(double timeBeforeGetTask)

{

this.timeBeforeGetTask = timeBeforeGetTask;

queue = new CMOState[2];

timeUntilEndOfTask = double.MaxValue;

}

public void UpdateTime(double time)

{

timeBeforeGetTask -= time;

timeUntilEndOfTask -= time;

}

public void AddToQueue()

{

if (queue.Any(elem => elem == CMOState.Free))

{

if (queue[1] == CMOState.Free)

{

queue[1] = CMOState.Task;

}

else

{

if (queue[0] == CMOState.Free)

{

queue[0] = CMOState.Task;

}

}

}

}

}

internal class Program

{

// Time

private const double TIME = 1000000;

// Values

private const double LAMBDA = 0.9;

private const double MU = 0.8;

private static Random rand = new Random();

private static Func<double, double> distribution =

(param) => -1 \* Math.Log(rand.NextDouble()) / param;

static void Main(string[] args)

{

var cmoResult = new CMOResult();

var cmo = new CMO(distribution(LAMBDA));

double time = 0;

while (time < TIME)

{

if (cmo.timeBeforeGetTask == 0)

{

cmoResult.countOfReceivedTask++;

if (cmo.chanelState == CMOState.Free)

{

cmo.chanelState = CMOState.Task;

cmo.timeUntilEndOfTask = distribution(MU);

}

else

{

cmo.AddToQueue();

}

cmo.timeBeforeGetTask = distribution(LAMBDA);

}

if (cmo.timeUntilEndOfTask == 0)

{

cmoResult.countOfProcessedTask++;

cmo.chanelState = CMOState.Free;

if (cmo.queue.Any(elem => elem != CMOState.Free))

{

cmo.chanelState = CMOState.Task;

if (cmo.queue[0] == CMOState.Free)

{

cmo.queue[1] = CMOState.Free;

}

else

{

cmo.queue[0] = CMOState.Free;

}

cmo.timeUntilEndOfTask = distribution(2 \* MU);

}

else

{

cmo.timeUntilEndOfTask = double.MaxValue;

}

}

double timeTemp = Math.Min(cmo.timeBeforeGetTask, cmo.timeUntilEndOfTask);

if (cmo.chanelState != CMOState.Free)

{

cmoResult.allTimeInChanel += timeTemp;

}

cmoResult.allTimeInQueue += cmo.queue.Count(elem => elem != CMOState.Free) \* timeTemp;

time += timeTemp;

cmo.UpdateTime(timeTemp);

}

Console.WriteLine($"A: {cmoResult.countOfProcessedTask / TIME}");

Console.WriteLine($"Wоч: {cmoResult.allTimeInQueue / cmoResult.countOfReceivedTask}");

Console.WriteLine($"Wс: {(cmoResult.allTimeInQueue + cmoResult.allTimeInChanel) / cmoResult.countOfReceivedTask}");

}

}

}