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»

Выполнил

студент: гр. 751004 Тушинская Е.В.

Проверил: Мельник Н.И.

Минск, 2020

**Вариант:** 21

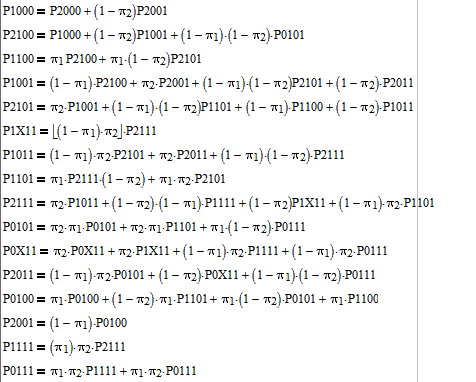
Цель исследования: .

Рисунок 1 – Вид анализируемой системы

Состояния системы кодируются четырьмя числами , где:

* – источник, принимает значения:
  + 0 – источник заблокирован;
  + 1 – один такт до выдачи заявки;
  + 2 – два такта до выдачи заявки;
* – каналы обработки с вероятностями просеивания и . Принимают значения:
  + 0 – канал свободен;
  + 1 – канал занят;
  + X – канал заблокирован;
* – накопитель на 1 заявку. Принимает значения (число накопленных заявок).

Граф состояний системы изображён на рисунке 2. По графу получим систему уравнений:



Нормировочное уравнение:

Результат вычислений в программе MathCad представлен на рисунке 3

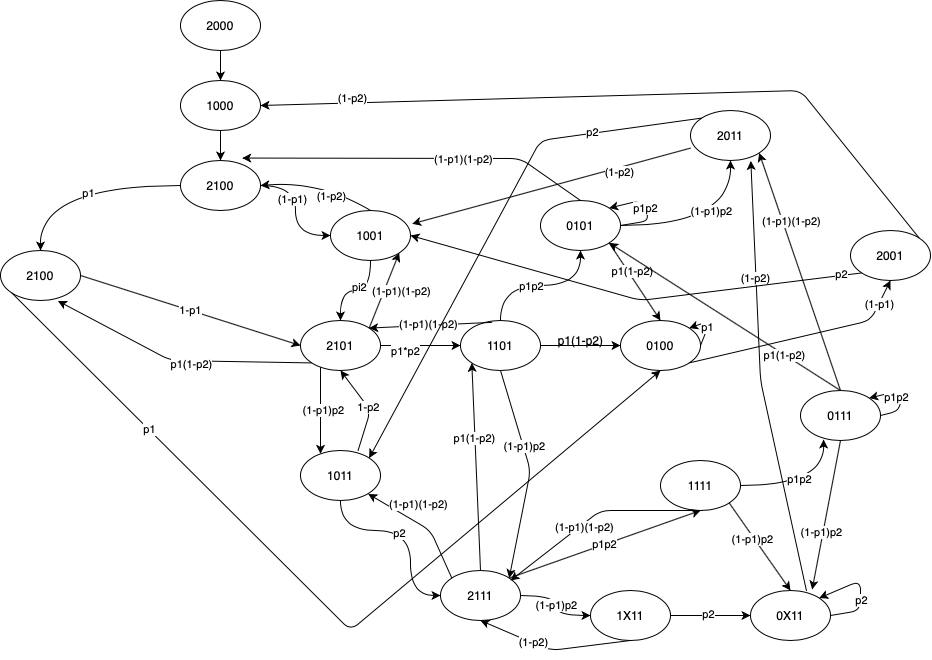
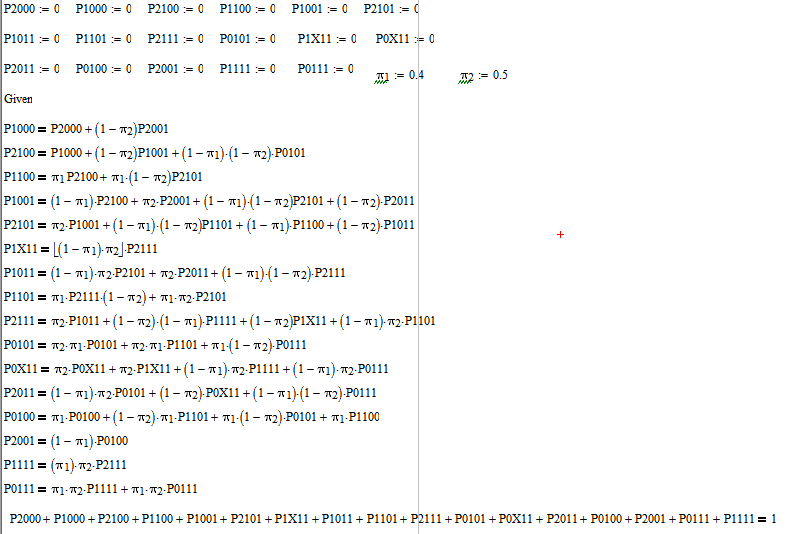


Рисунок 2 – Граф состояний системы



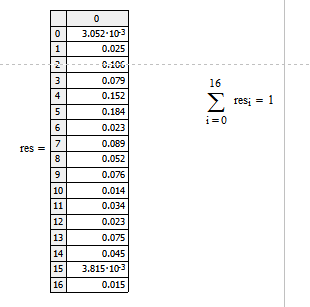


Рисунок 3 – Решение в Mathcad

Определим значения показателей эффективности работы системы: Ротк – вероятность отказа; Рбл – вероятность блокировки; Lоч – средняя длина очереди; Lc – среднее число заявок, находящихся в системе; Q – относительная пропускная способность; А – абсолютная пропускная способность; Wоч – среднее время пребывания заявки в очереди; Wс – среднее время пребывания заявки в системе, Kкан – коэффициент загрузки канала (вероятность занятости канала).

*Woch=Loch/A*

*Wc=Lc/A*

*Potkz = 0*

*Q = 1 - Potkz*

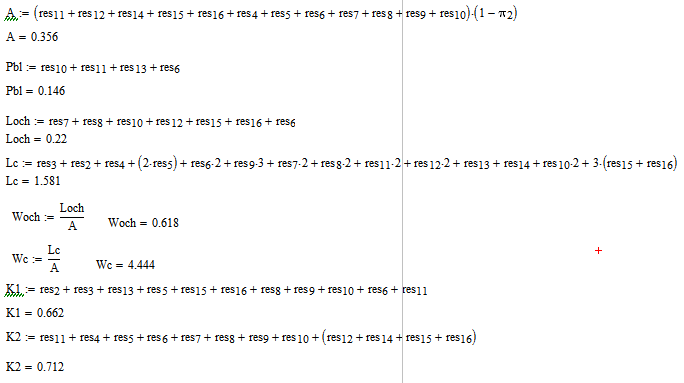


Рисунок 4 – Результаты расчёта показателей эффективности в Mathcad

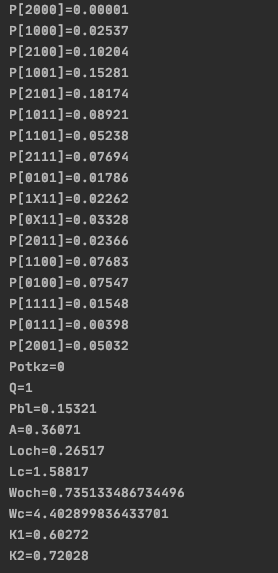


Рисунок 5 – Результат работы имитационной модели

Исходный код имитационной модели:

class SimulatorV21:

def \_\_init\_\_(self, generator):

self.generator = generator

@classmethod

def simulation\_statistics(cls, generator\_type, tact\_count, pi1: int, pi2: int):

statistic = SimulationStatistic(0, 0, 0, 0, 0, 0)

simulator = SimulatorState(Source(SourceState.FIRST\_TIC),

Channel(ChannelState.EMPTY),

0,

Channel(ChannelState.EMPTY))

cls.update\_statistics(simulator, statistic)

for \_ in range(tact\_count-1):

cls.change\_simulator\_state(generator\_type, simulator, statistic, pi1, pi2)

cls.update\_statistics(simulator, statistic)

statistic.A /= tact\_count

statistic.Loch /= tact\_count

statistic.K1 /= tact\_count

statistic.K2 /= tact\_count

statistic.Pbl /= tact\_count

statistic.Lc /= tact\_count

return statistic

@staticmethod

def update\_statistics(simulator: SimulatorState, statistics):

if simulator.channel1.state == ChannelState.BUSY.value:

statistics.K1 += 1

statistics.Lc += 1

if simulator.channel2.state != ChannelState.EMPTY.value:

statistics.K2 += 1

statistics.Lc += 1

if simulator.queue != 0:

statistics.Loch += simulator.queue

statistics.Lc += 1

if simulator.source.state == SourceState.BLOCKED.value or simulator.channel1.state == ChannelState.BLOCKED.value:

statistics.Pbl += 1

state = simulator.source.state + simulator.channel1.state + str(simulator.queue) + simulator.channel2.state

statistics.SimulationStates[state] = statistics.SimulationStates[state] + 1

@staticmethod

def change\_simulator\_state(generator, simulator, statistic, pi1, pi2):

if simulator.channel2.state == ChannelState.BUSY.value and (generator.generator.next > pi2):

simulator.channel2.state = ChannelState.EMPTY

statistic.A += 1

if simulator.queue > 0 and simulator.channel2.state == ChannelState.EMPTY.value:

simulator.channel2.state = ChannelState.BUSY

simulator.queue -= 1

if simulator.channel1.state == ChannelState.BUSY.value and (generator.generator.next > pi1):

if simulator.queue == 0 and simulator.channel2.state == ChannelState.EMPTY.value:

simulator.channel2.state = ChannelState.BUSY

simulator.channel1.state = ChannelState.EMPTY

elif simulator.channel2.state == ChannelState.BUSY.value and simulator.queue < QUEUE\_MAX\_ITEMS\_COUNT:

simulator.queue += 1

simulator.channel1.state = ChannelState.EMPTY

elif simulator.channel2.state == ChannelState.BUSY.value and simulator.queue == QUEUE\_MAX\_ITEMS\_COUNT:

simulator.channel1.state = ChannelState.BLOCKED

elif simulator.channel1.state == ChannelState.BLOCKED.value:

if simulator.queue == 0 and simulator.channel2.state == ChannelState.EMPTY.value:

simulator.channel2.state = ChannelState.BUSY

simulator.channel1.state = ChannelState.EMPTY

elif simulator.channel2.state == ChannelState.BUSY.value and simulator.queue < QUEUE\_MAX\_ITEMS\_COUNT:

simulator.queue += 1

simulator.channel1.state = ChannelState.EMPTY

if simulator.source.state == SourceState.LAST\_TIC.value:

if simulator.channel1.state == ChannelState.EMPTY.value:

simulator.channel1.state = ChannelState.BUSY

simulator.source.state = SourceState.FIRST\_TIC

else:

simulator.source.state = SourceState.BLOCKED

elif simulator.source.state == SourceState.FIRST\_TIC.value:

simulator.source.state = SourceState.LAST\_TIC

elif simulator.source.state == SourceState.BLOCKED.value:

if simulator.channel1.state == ChannelState.EMPTY.value:

simulator.source.state = SourceState.FIRST\_TIC