Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра программного обеспечения информационных технологий

ОТЧЁТ

по лабораторной работе №3

по курсу «Системный анализ и машинное моделирование»

на тему

«Аналитическое моделирование дискретно-стохастической СМО»

Вариант 28

|  |  |
| --- | --- |
| Проверил:  Мельник Н.И. | Выполнил:  студент гр. 751005  Яковлевич П. О. |

Минск 2020

**Цель работы:**

* Изучить принципы построения и функционирования СМО.
* Освоение методики построения графа состояний СМО.
* Научиться по графу состояний строить аналитическую модель и на ее основании определять вероятности состояний

**Схема СМО:**



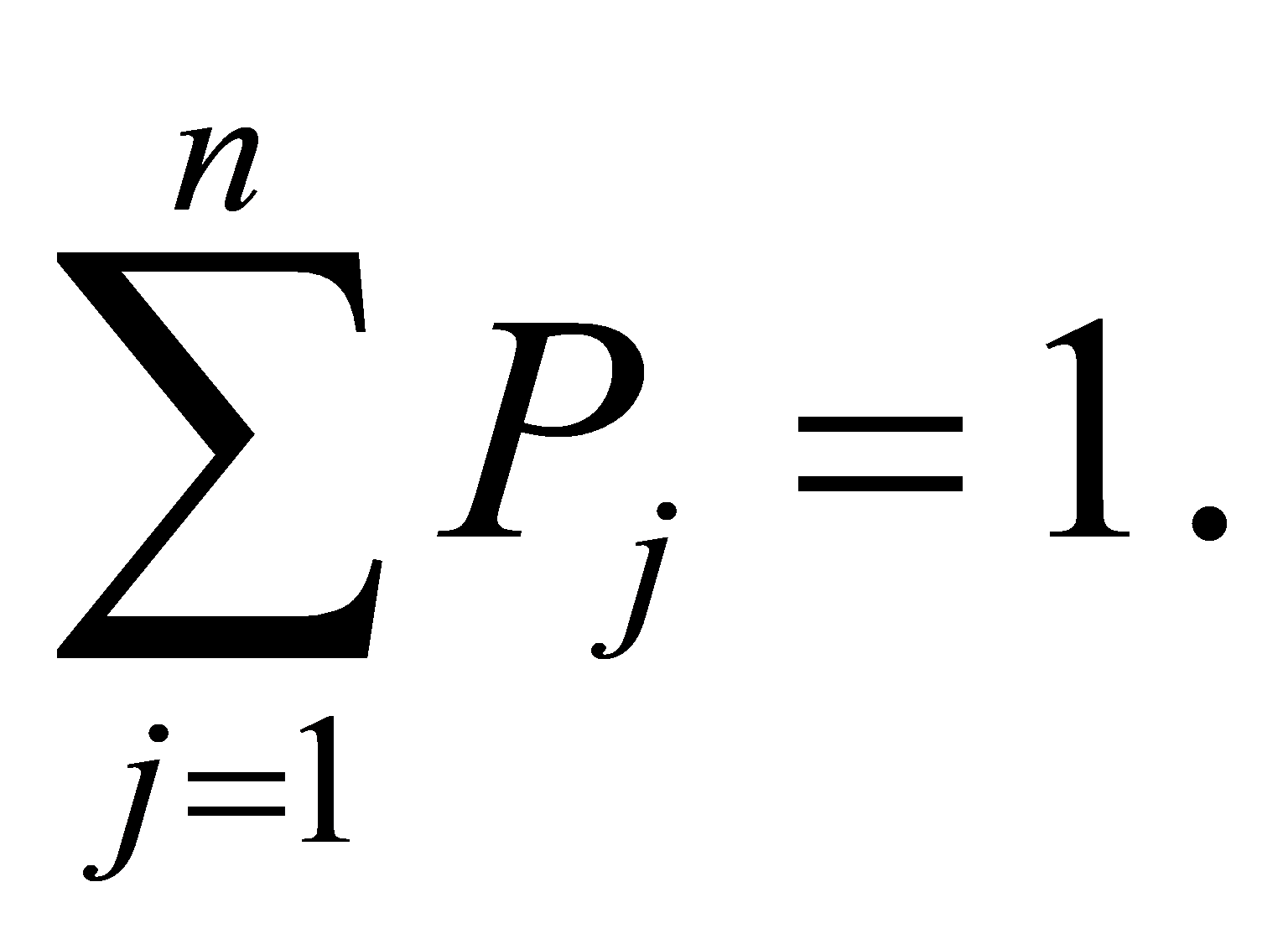
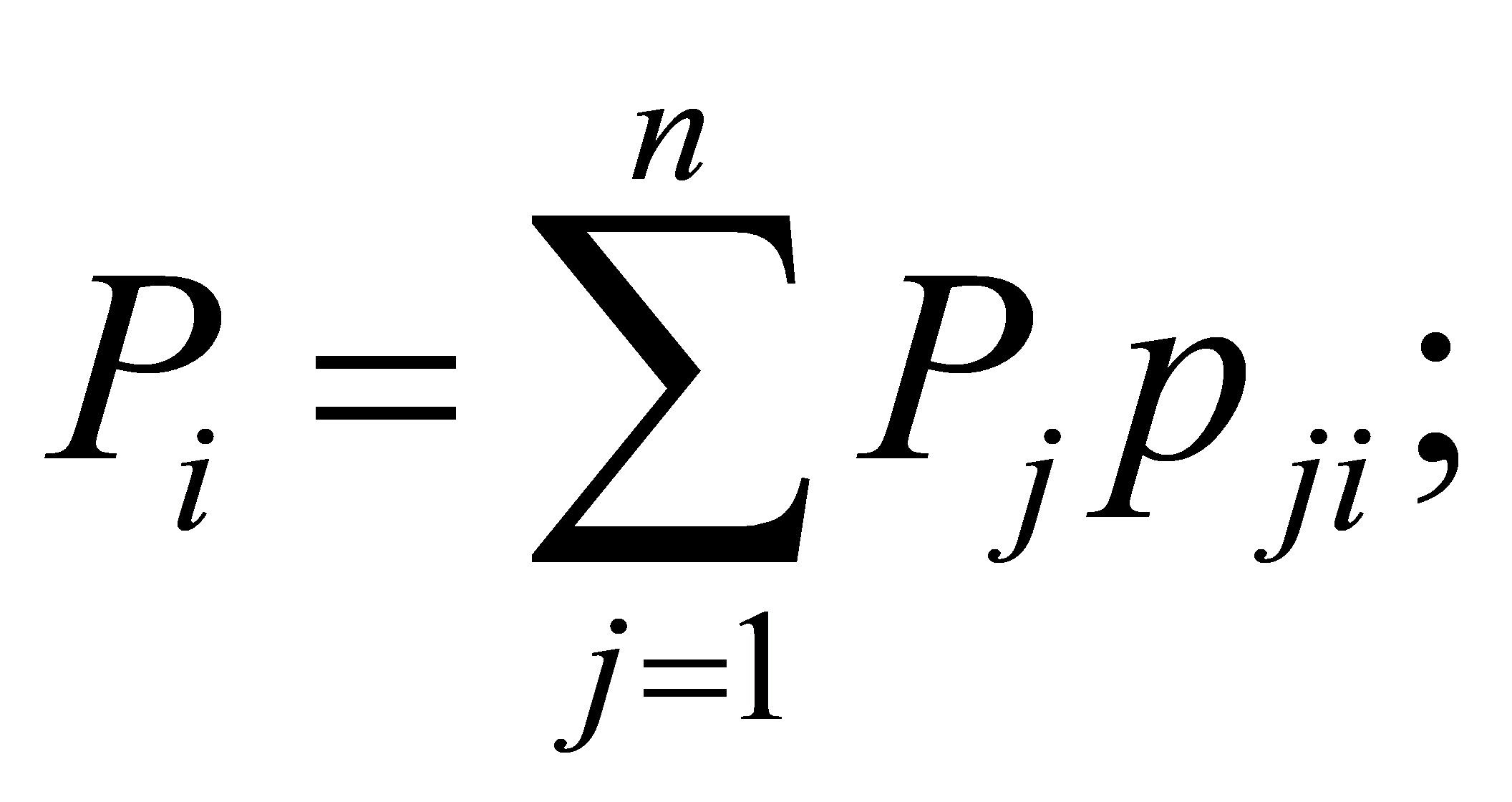
p = 0.75

Pi1 = 0.7

Pi2 = 0.65

**Система уравнений и ее решение:**

Граф состояний описывается трехкомпонентным вектором *abc,* где *a,c* – состояния каналов обслуживания: 0 – в данный момент заявки в канале нет, 1 – канал занят обслуживанием заявки, *a,c*=(0,1); *b* – количество заявок в накопителе(количество заявок в очереди), b=(0,1,2). Исходя из построенного графа состояний, получим СЛАУ для нахождения вероятностей состояний: и заменим одно из уравнений нормировочным



P1=P000;

P2=P100;

P3=P001;

P4=P101;

P5=P111;

P6=P011;

P7=P021;

P8=P121;

P1= P1\*p +

P3\*p\*(1- π2)

P2= P1\*(1-p) +

P3\*((1-p)\*((1- π2)) +

P4\*( π1\*(1- π2)) +

P2\*π1

P3= P2\*p\*(1- π1) +

P3\*p\*π2 +

P6\*p\*(1-π2) +

P4\*p\*(1-π1)\*(1-π2)

P4= P2\*(1-p)\*(1-π1) +

P3\*(1-p)\*π2 +

P5\*π1\*(1-π2) +

P6\*(1-p)\*(1-π2) +

P4\*(π1\*π2 + (1-p)\*(1-π1)\*(1-π2))

P5= P4\*((1-p)\*(1-π1)\* π2) +

P5\*(π1\*π2 + (1-p)\*(1-π1)\*(1-π2)) +

P7\*(1-p)\*(1-π2) +

P6\*((1-p)\* π2) +

P8\*(π1\*(1-π2))

P6= P4\*(p\*(1-π1)\* π2) +

P7\*(p\*(1-π2)) +

P6\*(p\*π2) +

P5\*(p\*(1-π1)\*(1-π2))

P7= P5\*(p\*(1-π1)\* π2) +

P7\*(p\*π2) +

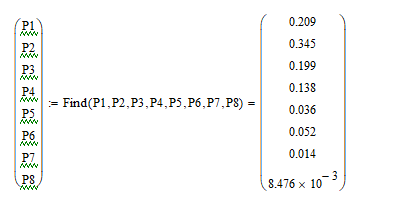
P8\*(p\*(1-π1)\*π2 + (p\*(1-π1)\*(1-π2))

P8= P5\*((1-p)\*(1-π1)\* π2) +

P7\*((1-p)\*π2) +

P8\*(π1\*π2 + ((1-p)\*(1-π1)\*π2) +((1-p)\*(1-π1)\*(1-π2))

Решение системы в MathCad



**Расчет показателей системы:**

Ротк – вероятность отказа:

**Ротк =** P2 \* π1 + P4 \* π1 + P5 \* π1 + P8 \* π1 + (P8 / P2 + P4 + P5) \* π2 = 0.37

Рбл – вероятность блокировки:

**Рбл**=0

Lоч – средняя длина очереди:

**Lоч**= 1 \* P5 + 1 \* P6 + 2 \* P7 + P8 \* 2 = 0.132

Lc – среднее число заявок:

**Lc**= 1 \* P2 + 1 \* P3 + 2 \* P4 + P5 \* 3 + P6 \* 2 + P7 \* 3 + P8 \* 4 = 1.105

Q – относительная пропускная способность

**Q**= 1 - Ротк= 0.63;

А – абсолютная пропускная способность:

**А** = (1-p)\*Q = 0.157

Wоч – среднее время пребывания заявки в очереди:

**Wоч** = Lоч/A= 0.838

Wс – среднее время пребывания заявки в системе:

**Wс**= Lc/A= 7.015

Kкан – коэффициент загрузки канала (вероятность занятости канала):

**Kкан1**=P2+P4+P5+P8= 0.526

**Kкан2**= P3+P4+P5+P6+P8+P7= 0.446

**Результат:**

|  |  |  |
| --- | --- | --- |
| Состояние\характеристика | Аналитическая модель | Результаты моделирования |
| **P000(P1)** | 0.209 | 0.208 |
| **P100(P2)** | 0.345 | 0.345 |
| **P001(P3)** | 0.199 | 0.199 |
| **P101(P4)** | 0.138 | 0,139 |
| **P111(P5)** | 0.036 | 0.035 |
| **P011(P6)** | 0.052 | 0.052 |
| **P021(P7)** | 0.014 | 0.014 |
| **P121(P8)** | 0.008 | 0.009 |
| **Ротк** | 0.37 | 0.376 |
| **Рбл** | 0 | 0 |
| **Lоч** | 0.132 | 0.132 |
| **Lc** | 1.105 | 1.107 |
| **Q** | 0.63 | 0.62 |
| **A** | 0.157 | 0.156 |
| **Wоч** | 0.838 | 1.844 |
| **Wс** | 7.015 | 7.08 |
| **Kкан1** | 0.526 | 0.528 |
| **Kкан2** | 0.446 | 0.446 |

Результаты моделирования были получены на основе 1000000 тактов программного средства.

**Вывод:**

В ходе выполнения данной лабораторной работы была построена модель дискретно-стохастической СМО, включающая в себя граф состояний модели и аналитическую модель. В результате решения аналитической модели были получены вероятности состояний модели, на основании которых были в последствии рассчитаны такие характеристики СМО как абсолютная пропускная способность, средняя длина очереди и вероятность отказа.

Помимо этого, была создана программная реализация заданной СМО и практически рассчитаны необходимые параметры системы.

**Исходный код на языке C#:**

dataGridView1.Rows.Clear();

MarkovChain chain = new MarkovChain();

Generator generator = new Generator(Convert.ToDouble(txtBoxP.Text), true, chain.Listener);

HandlerNode handlerNode1 = new HandlerNode(Convert.ToDouble(txtBoxPI1.Text), true, chain.Listener);

QueueNode queue = new QueueNode(false, 2, chain.Listener);

HandlerNode handlerNode2 = new HandlerNode(Convert.ToDouble(txtBoxPI2.Text), false, chain.Listener);

chain.IntializeChain(generator, handlerNode1, queue, handlerNode2);

chain.TiksCount = Convert.ToInt32(txtBoxTiksCount.Text);

chain.Work();

txtBoxDenyvalue.Text = chain.Statistics.DenyChanceValue.ToString();

txtBoxAvgQueueLen.Text = chain.Statistics.AverageQueueLength.ToString();

txtBoxAverageReqCountInQueue.Text = chain.Statistics.AvgRequestsCountInSystem.ToString();

txtBoxQ.Text = chain.Statistics.RelativePassAbility.ToString();

txtBoxA.Text = chain.Statistics.AbsolutePassAbility.ToString();

txtBoxAvgReqTimeInQueue.Text = chain.Statistics.AvgTimeRequestInQueue.ToString();

txtBoxAvgReqTimeInSystem.Text = chain.Statistics.AvgTimeRequestInSystem.ToString();

txtBoxLoadCoef1.Text = chain.Statistics.CanalLoadCoef[0].ToString();

txtBoxLoadCoef2.Text = chain.Statistics.CanalLoadCoef[1].ToString();

int i = 0;

foreach (var item in chain.Statistics.StatesFrequency)

{

dataGridView1.Rows.Add("P" + item.Key, chain.Statistics.StatesChances[i]);

i++;

}

**Канал обработки:**

public class HandlerNode : BlockableNode

{

public HandlerNode(double chance, bool canDropValue, NodeListener listener) :

base(chance, canDropValue, listener)

{

}

public override void RefreshState()

{

//

}

public override void TransportValueFurther()

{

if (Next != null)

{

(Next as BlockableNode).IncomeValue = Value;

}

}

public override void Work()

{

if(Value == 0)

{

ReceiveIncomeValue();

return;

}

double chance = random.NextDouble();

if(chance > ChanceValue)

{

if (Next == null)

{

nodeListener.ValueWasHandled();

}

else

{

TransportValueFurther();

}

ReceiveIncomeValue();

nodeListener.ValueWasHandledByHandler(Id);

}

else

{

if(IncomeValue == 1)

{

Previous.DropValue();

}

}

IncomeValue = 0;

}

}

**Очередь:**

public class QueueNode : BlockableNode

{

public int Size { get; private set; }

public QueueNode(bool canDropValue, int size, NodeListener listener) :

base(1, canDropValue, listener)

{

Size = size;

}

public override void TransportValueFurther()

{

(Next as BlockableNode).IncomeValue = Value > 0 ? 1 : 0;

}

public override void Work()

{

if (IncomeValue == 1)

{

if (Value < Size)

{

Value++;

IncomeValue = 0;

}

}

TransportValueFurther();

}

public void DequeueValue()

{

Value--;

}

public override void RefreshState()

{

if (IncomeValue == 1)

{

if (Value < Size)

{

Value++;

}

else

{

Previous.DropValue();

}

}

IncomeValue = 0;

}

}

**Источник:**

class Generator : Node

{

public Generator(double chance, bool canDropValue, NodeListener listener) :

base(chance, canDropValue, listener)

{

}

private void GenerateValue()

{

double chance = random.NextDouble();

Success = chance > ChanceValue ? true : false;

if (Success)

{

nodeListener.ValueWasGenerated();

}

}

public override void TransportValueFurther()

{

(Next as BlockableNode).IncomeValue = Convert.ToInt32(Success);

}

public override void Work()

{

GenerateValue();

TransportValueFurther();

}

}

