Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра программного обеспечения информационных технологий

ОТЧЁТ

по лабораторной работе №5

по курсу «Системный анализ и машинное моделирование»

на тему

«Построение аналитической и имитационной моделей непрерывно-стохастической системы»

Вариант 36

|  |  |
| --- | --- |
| Проверил:  Мельник Н.И. | Выполнил:  студент гр. 751002  Авко Д.А. |

Минск 2020

**Цель работы:**

Построение аналитической и имитационных моделей непрерывно-стохастической системы и расчет параметров данной системы с использованием построенных моделей.

**Исходные данные к работе:**

В СМО вида М/М/1/2 поступают заявки двух видов. Заявка первого вида появляется на входе с вероятностью р, второго с вероятностью (1-р). Заявка первого вида имеет более высокий приоритет и может вытеснить заявку второго вида из канала в очередь, если в очереди есть свободное место или из системы, если оба места заняты. В случае, когда заявка первого вида застает систему в состоянии обслуживания заявки первого вида, то она ставится в очередь, если есть свободное место ожидания или вытесняет из очереди заявку второго вида (менее приоритетная заявка теряется). Загрузка заявок в канал из очереди производится в соответствии с их приоритетами. Найти относительные пропускные способности Q1 и Q2. =0,5, =0,9, р=0,3.

P0=P00+0;

P1=P00+1;

P2=P00+2;

P3=P01+1;

P4=P02+1;

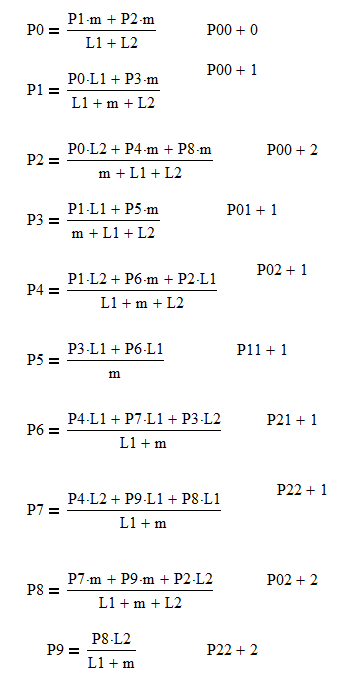
P5=P11+1;

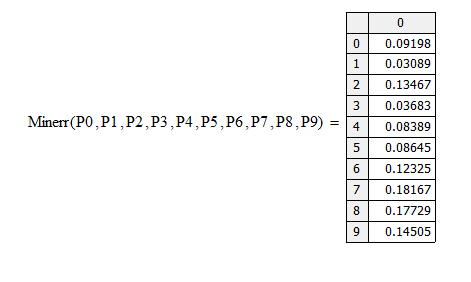
P6=P21+1;

P7=P22+1;

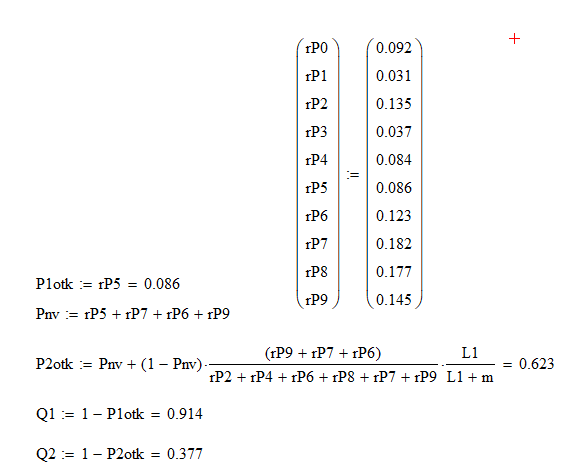
P8=P02+2;

P9=P22+2;





**Расчет Q1 и Q2:**



Результаты имитационной модели были получены на основе 100\_000\_000 тактов программного средства.

**Код программы:**

from model import Model

if \_\_name\_\_ == '\_\_main\_\_':

model = Model(n=5\_000\_000, lmb=0.9, m=0.5, p=0.3)

model.simulate()

generator.py

class Generator:

def \_\_init\_\_(self, lmb, p, rnd):

self.lmb = lmb

self.p = p

self.rnd = rnd

def generate(self):

if self.rnd.random() < self.lmb:

if self.rnd.random() < self.p:

return 1

else:

return 2

return 0

сlass Queue:

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

self.t\_type1 = 0

self.t\_type2 = 0

def enqueue(self, t\_type):

if t\_type < self.t\_type1 or not self.t\_type1:

tmp = self.t\_type1

self.t\_type1 = t\_type

tmp2 = self.t\_type2

self.t\_type2 = tmp

return tmp2

elif t\_type < self.t\_type2 or not self.t\_type2:

tmp = self.t\_type2

self.t\_type2 = t\_type

return tmp

return t\_type

def dequeue(self):

tmp = self.t\_type1

self.t\_type1 = self.t\_type2

self.t\_type2 = 0

return tmp

def get\_state(self):

return str(self.t\_type2) + str(self.t\_type1)

lass Handler:

def \_\_init\_\_(self, m, rnd):

self.m = m

self.t\_type = 0

self.rnd = rnd

def set\_task(self, t\_type):

tmp = self.t\_type

self.t\_type = t\_type

return tmp

def handle(self):

if self.t\_type:

if self.rnd.random() < self.m:

tmp = self.t\_type

self.t\_type = 0

return tmp

return 0

def get\_state(self):

return str(self.t\_type)

**Выводы:**

В данной работе была исследована непрерывно-стохастическая модель. Для нее был построена диаграмма интенсивностей переходов. Были посчитаны необходимые характеристики системы и сравнены с результатами, получившимися при имитационном моделировании.