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 №3

Построение и исследование аналитической модели дискретно-стохастической системы массового обслуживания

Вариант 12

Выполнил

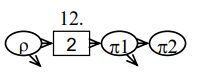
студент: гр. 851004 Пашкевич А.Л.

Проверил: Мельник Н.И.

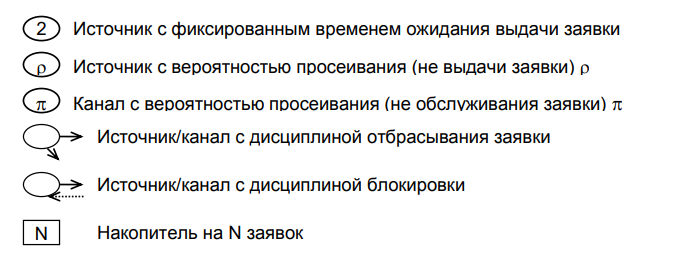
Минск 2021

# Постановка задачи

Построить граф состояний схемы:



Элементы схемы:



Кодировка состояний системы:

n = {0, 1, 2} – количество заявок в очереди;

k1 = {0, 1} – состояние первого канала (0 – свободен, 1 – занят);

k2 = {0, 1} – состояние второго канала (0 – свободен, 1 – занят).

Общий вид:

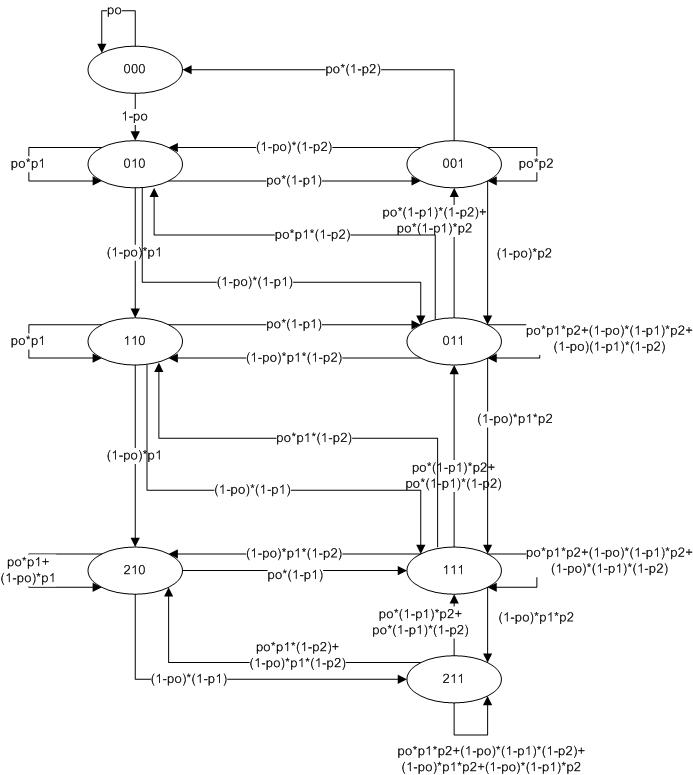
{n, k1, k2}

Вероятность не выдачи заявки источником:

Вероятности не обслуживания заявок:

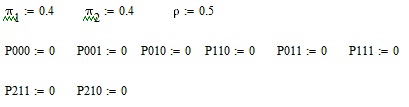
# Ход выполнения работы

Граф состояний:

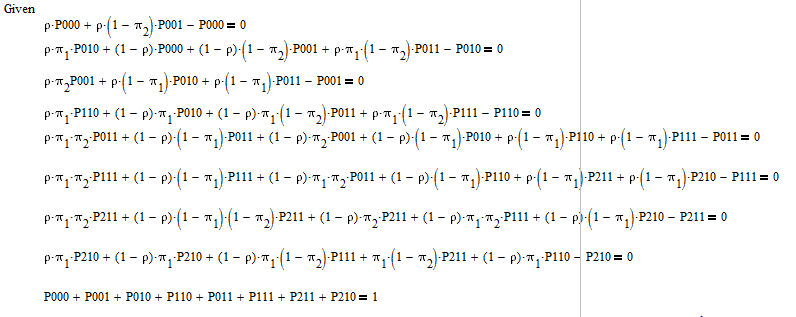


Система уравнений по графу состояний:

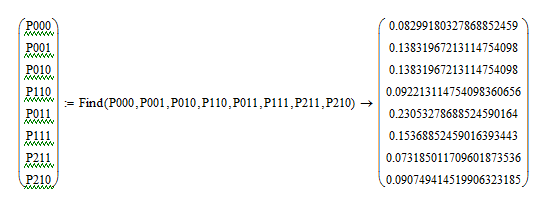
Решение в маткаде:



Система уравнений:

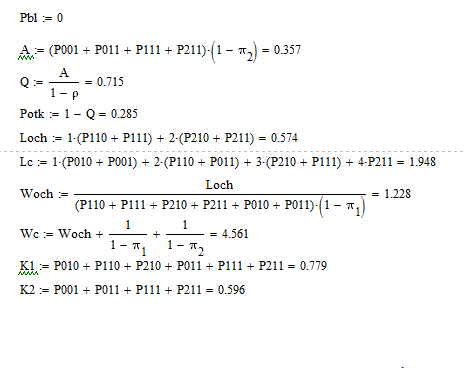


Найденные вероятности состояний системы:

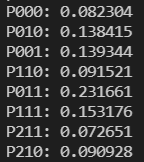


Расчет показателей эффективности:

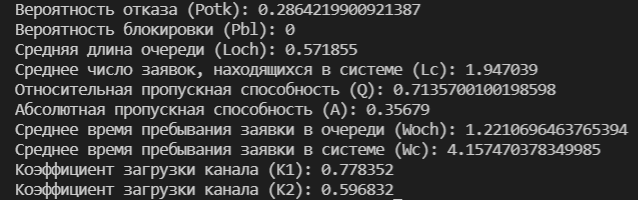
Ротк – вероятность отказа; Рбл – вероятность блокировки; Lоч – средняя длина очереди; Lc – среднее число заявок, находящихся в системе; Q – относительная пропускная способность; А – абсолютная пропускная способность; Wоч – среднее время пребывания заявки в очереди; Wс – среднее время пребывания заявки в системе; K1 – коэффициент загрузки первого канала; K2 – коэффициент загрузки второго канала.



Вероятности состояний системы при имитационном моделировании:



Расчёт показателей эффективности системы при иммитационном моделировании:



Листинг имммитационной модели:

from StateMachine import StateMachine

from Lehmer import Lehmer

startValue = 995053

multiplier = 550000

modulus = 1000002

startState = "000"

N = 1000000

T = 2

def main():

try:

po = float(input("po: "))

p1 = float(input("p1: "))

p2 = float(input("p2: "))

except:

po = 0.5

p1 = 0.4

p2 = 0.4

generator = Lehmer(multiplier, startValue, modulus)

stateMachine = StateMachine(startState, po, p1, p2)

for i in range(N):

po = generator.GetNext()

p1 = generator.GetNext()

p2 = generator.GetNext()

stateMachine.SwitchState(po <= stateMachine.po, p1 <= stateMachine.p1, p2 <= stateMachine.p2)

print(f"P000: {stateMachine.P000 / N}")

print(f"P010: {stateMachine.P010 / N}")

print(f"P001: {stateMachine.P001 / N}")

print(f"P110: {stateMachine.P110 / N}")

print(f"P011: {stateMachine.P011 / N}")

print(f"P111: {stateMachine.P111 / N}")

print(f"P211: {stateMachine.P211 / N}")

print(f"P210: {stateMachine.P210 / N}")

print(f"Вероятность отказа (Potk): {stateMachine.dcl / stateMachine.ga}")

print(f"Вероятность блокировки (Pbl): {0}")

print(f"Средняя длина очереди (Loch): {stateMachine.Loch / N}")

print(f"Среднее число заявок, находящихся в системе (Lc): {stateMachine.Lc / N }")

print(f"Относительная пропускная способность (Q): {stateMachine.A / stateMachine.ga}")

print(f"Абсолютная пропускная способность (A): {stateMachine.A / N}")

print(f"Среднее время пребывания заявки в очереди (Woch): {stateMachine.Loch / stateMachine.en}")

print(f"Среднее время пребывания заявки в системе (Wc): {stateMachine.Lc / stateMachine.en}")

print(f"Коэффициент загрузки канала (K1): {stateMachine.K1 / N}")

print(f"Коэффициент загрузки канала (K2): {stateMachine.K2 / N}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

class Lehmer:

def \_\_init\_\_(self, multiplier, start, modulus):

self.a = multiplier

self.R = start

self.m = modulus

self.x = self.R

def GetNext(self):

self.x = self.a \* self.x % self.m

return self.x / self.m

class StateMachine:

def \_\_init\_\_(self, startState, po, p1, p2):

self.state = startState

self.po = po

self.p1 = p1

self.p2 = p2

self.P000 = 0

self.P001 = 0

self.P010 = 0

self.P110 = 0

self.P011 = 0

self.P111 = 0

self.P211 = 0

self.P210 = 0

self.ga = 0

self.dcl = 0

self.en = 0

self.Pbl = 0

self.Potk = 0

self.Q = 0

self.Loch = 0

self.Lc = 0

self.A = 0

self.K1 = 0

self.K2 = 0

def SwitchState(self, po, p1, p2):

if not po:

self.ga += 1

if self.state[0] != "2" and not po:

self.en += 1

if self.state == "000":

self.P000 += 1

if po:

self.state = "000"

if not po:

self.state = "010"

elif self.state == "010":

self.K1 += 1

self.Lc += 1

self.P010 += 1

if po and p1:

self.state = "010"

if (not po) and p1:

self.state = "110"

if po and (not p1):

self.state = "001"

if (not po) and (not p1):

self.state = "011"

elif self.state == "001":

if not p2:

self.A += 1

self.K2 += 1

self.Lc += 1

self.P001 += 1

if po and (not p2):

self.state = "000"

if po and p2:

self.state = "001"

if (not po) and (not p2):

self.state = "010"

if (not po) and p2:

self.state = "011"

elif self.state == "110":

self.K1 += 1

self.Loch += 1

self.Lc += 2

self.P110 += 1

if po and p1:

self.state = "110"

if po and (not p1):

self.state = "011"

if (not po) and p1:

self.state = "210"

if (not po) and (not p1):

self.state = "111"

elif self.state == "011":

if not p2:

self.A += 1

elif not p1:

self.dcl += 1

self.K1 += 1

self.K2 += 1

self.Lc += 2

self.P011 += 1

if (po and p1 and p2) or ((not po) and (not p1)):

self.state = "011"

elif po and p1 and (not p2):

self.state = "010"

elif po and not p1:

self.state = "001"

elif (not po) and p1 and (not p2):

self.state = "110"

elif (not po) and p1 and p2:

self.state = "111"

elif self.state == "111":

if not p2:

self.A += 1

elif not p1:

self.dcl += 1

self.Loch += 1

self.K1 += 1

self.K2 += 1

self.Lc += 3

self.P111 += 1

if (po and p1 and p2) or ((not po) and (not p1)):

self.state = "111"

if po and (not p1):

self.state = "011"

if po and p1 and (not p2):

self.state = "110"

if (not po) and p1 and p2:

self.state = "211"

if (not po) and p1 and (not p2):

self.state = "210"

elif self.state == "211":

if not p2:

self.A += 1

elif not p1:

self.dcl += 1

if not po and p1:

self.dcl += 1

self.Loch += 2

self.K1 += 1

self.K2 += 1

self.Lc += 4

self.P211 += 1

if (po and p1 and p2) or

((not po) and (not p1) and (not p2)) or (not po) and p2:

self.state = "211"

if po and (not p1):

self.state = "111"

self.en += 1

if p1 and (not p2):

self.state = "210"

elif self.state == "210":

if not po and p1:

self.dcl += 1

self.K1 += 1

self.Loch += 2

self.Lc += 3

self.P210 += 1

if po and p1 or (not po) and p1:

self.state = "210"

if po and (not p1):

self.state = "111"

self.en += 1

if (not po) and (not p1):

self.state = "211"