Белорусский государственный университет информатики и радиоэлектроники

# Кафедра ПОИТ

### Отчет по лабораторной работе № 4

«Аналитическое моделирование непрерывно-стохастической СМО и построение её имитационной модели»

Выполнил:

студент группы 150501

Белов А.В.

Проверил:

Огородник Р.В.

Минск 2014

1. **Задание.**

Двухфазная СМО с отказами

n1 n2

λ μ1=5 μ2=5

Варианты:

а) Построить зависимости Ротк, Ротк1, Ротк2 при изменении λ от 1 до 6 с шагом 0.5. Входной поток и потоки обслуживаний – простейшие, n1 = 2, n2 = 2.

б) Построить зависимости Ротк, Ротк1, Ротк2 при изменении n1 от 1 до 6. Входной поток и потоки обслуживаний - простейшие, n2 = 2, λ = 4.5.

1. **Вывод формул для аналитической модели.**

Каждая из фаз данной двухфазной СМО является одноканальной СМО с фиксированным числом мест ожидания.

Входной поток заявок простейший с интенсивностью λ, поток обслуживаний простейший с интенсивностью μ, количество мест ожидания n. Заявка, заставшая очередь полностью заполненной, теряется. Максимальное количество заявок, присутствующих в фазе – n+1 (n заявок в очереди и одна заявка в канале).

Диаграмма интенсивности переходов для каждой фазы:



Воспользовавшись правилом равенства встречных потоков вероятностей через сечение диаграммы и, введя обозначения P0= p и λ/μ = ω, получим:

λ P0 = μ P1, P1 = (λ/μ)P0 = ω p;

λ P1 = μ P2, P2 = (λ/μ)P1 = ω2 p;

λ P2 = μ P3, P3 = (λ/μ)P2 = ω3 p;

…

По индукции получаем Pi = ωi p.

Для определения значения P0 = p воспользуемся нормировочным уравнением:

Учитывая, что полученная сумма представляет собой сумму геометрической прогрессии, в которой n+2 слагаемых, получим:

Окончательно:

.

Если , то

**Вероятность отказа для 1-й фазы:**

,

где .

Интенсивность входного потока 2-й фазы есть абсолютная пропускная способность 1-й фазы:

*.*

**Вероятность отказа для 2-й фазы:**

,

где , *.*

**Вероятность отказа для СМО:**

.

1. **Выполнение задания варианта А.**

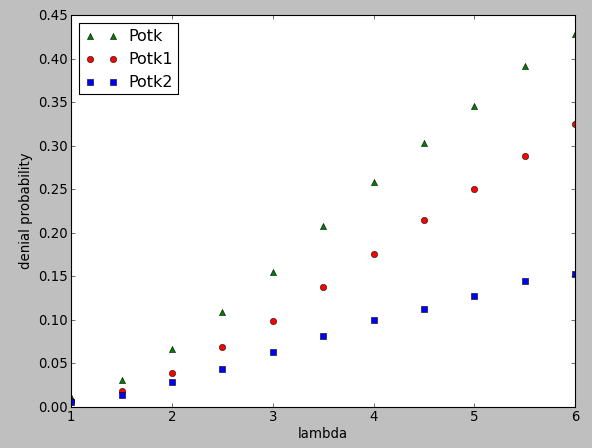
Зависимости Ротк, Ротк1, Ротк2 при изменении λ от 1 до 6 с шагом 0.5. Входной поток и потоки обслуживаний – простейшие, n1 = 2, n2 = 2.

***Аналитическая модель***



λ

***Имитационная модель***



Для сравнения при :

* в аналитической модели: Ротк1 = 0,25, Ротк2 = 0,15429, Ротк = 0,36571;
* в имитационной модели: Ротк1 = 0,25023, Ротк2 = 0,1299, Ротк = 0,3482;

Различие Ротк2 и Ротк в различных моделях объясняется тем, что очередь к 1-й фазе ограничена (n1 = 2). Таким образом выходной поток этой фазы не является простейшим. Но в аналитической модели входной поток 2-й фазы предполагается простейшим.

При увеличении мест ожидания выходной поток 1-й фазы будет приближаться к простейшему и результаты аналитического и имитационного моделирования для Ротк2 и Ротк будут почти одинаковые.

1. **Листинг программы.**

require 'colored'

require 'json'

class Queue

def initialize λ, μ1, μ2, max\_queue\_count\_1, max\_queue\_count\_2

@λ = λ

@phase1 = Phase.new(μ1, max\_queue\_count\_1)

@phase2 = Phase.new(μ2, max\_queue\_count\_2)

@denial\_count\_1 = 0

@denial\_count\_2 = 0

@denial\_count = 0

@requests\_number = 0

@requests\_number\_2 = 0

@denial\_probability\_1 = 0

@denial\_probability\_2 = 0

@denial\_probability = 0

instance\_variables.each do |var|

eval "def #{var.to\_s.sub('@','')}; #{var}; end"

end

end

def imitate requests\_number = 100\_000

@requests\_number = requests\_number

request\_stream = []

current\_time = 0

request\_stream.push 0

requests\_number.times do

current\_time += -Math.log(rand) / @λ

request\_stream.push current\_time

end

request\_stream\_2 = @phase1.imitate request\_stream

stream = @phase2.imitate request\_stream\_2

@requests\_number\_2 = request\_stream\_2.size

@denial\_count\_1 = @phase1.denial\_count

@denial\_count\_2 = @phase2.denial\_count

@denial\_count = @denial\_count\_1 + @denial\_count\_2

@denial\_probability\_1 = @denial\_count\_1 / @requests\_number.to\_f

@denial\_probability\_2 = @denial\_count\_2 / @requests\_number\_2.to\_f

@denial\_probability = @denial\_count / @requests\_number.to\_f

end

end

class Phase

def initialize μ, max\_queue\_count

@μ = μ

@max\_queue\_count = max\_queue\_count

@denial\_count = 0

instance\_variables.each do |var|

eval "def #{var.to\_s.sub('@','')}; #{var}; end"

end

end

def imitate input\_stream

current\_time = 0

queue\_count = 0

denial\_count = 0

output\_stream = []

current\_request = 0

while current\_request < input\_stream.size

if input\_stream[current\_request] < current\_time

if queue\_count == @max\_queue\_count

denial\_count += 1

else

queue\_count += 1

end

current\_request += 1

else

if queue\_count > 0

current\_time += -Math.log(rand) / @μ

queue\_count -= 1

else

current\_time = input\_stream[current\_request] + -Math.log(rand) / @μ

current\_request += 1

end

output\_stream.push current\_time

end

end

@denial\_count = denial\_count

return output\_stream

end

end

q = Queue.new 5,5,5,2,2

q.imitate

puts "For λ = 5".cyan

puts "Potk1 = %5.5f".red % q.denial\_probability\_1

puts "Potk2 = %5.5f".red % q.denial\_probability\_2

puts "P = %5.5f".red % q.denial\_probability

plot\_data = []

denial\_1 = []

denial\_2 = []

denial = []

for λ in (1..6).step 0.5

q = Queue.new λ,5,5,2,2

q.imitate

denial\_1 << q.denial\_probability\_1

denial\_2 << q.denial\_probability\_2

denial << q.denial\_probability

end

plot\_data << denial\_1 << denial\_2 << denial

File.open('plot\_data.json','w') do |f|

f.write plot\_data.to\_json

end

exec 'python draw\_plot.py'