**Практическое занятие № 2**

**Тема:** Расчет СМО M/M/n/r.

**Цель:** Приобретение практических навыков расчета показателей оперативности обработки данных с помощью системы массового обслуживания M/M/n/r.

**Язык программирования, ПО и библиотеки:** python 3.x, установленный пакет библиотек Anaconda или отдельно следующие: pandas, numpy, matplotlib. Среда разработки – PyCharm, Spyder 3 или Jypyter.

Порядок выполнения практического занятия.

1. Откройте PyCharm. Создайте новый файл с расширением *.py* в проекте.
2. Добавьте в папку с проектом файл *mmnr\_calc.py* и *smo\_im.py*.
3. Пропишите секцию import в следующем виде:

import mmnr\_calc  
import smo\_im  
import numpy as np  
import matplotlib.pyplot as plt

1. Для моделирования СМО M/M/n/r необходимо задать интенсивности поступления заявок λ и обслуживания μ, размер буфера r и число каналов обслуживания n. Определите значения количества каналов (n) и максимальную длину очереди (r) по номеру варианта (номеру по журналу группы) с помощью следующей таблицы.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **n** | | | | | | |
| **r** |  | **1** | **3** | **5** | **7** | **9** | **10** |
| **10** | 1 | 2 | 3 | 4 | 5 | 6 |
| **20** | 7 | 8 | 9 | 10 | 11 | 12 |
| **30** | 13 | 14 | 15 | 16 | 17 | 18 |
| **50** | 19 | 20 | 21 | 22 | 23 | 24 |

1. Для системы M/M/n/r осталось задать интенсивности входящего потока λ и обслуживания μ. Для дальнейших расчетов зададим λ = 1.0. Интенсивность обслуживания μ будем пересчитывать для различных коэффициентов загрузки системы, которые определяются по формуле

,

где для рассматриваемой системы .

1. Требуется сравнить оценку среднего времени ожидания заявок в очереди, полученную для имитационной модели со значением, рассчитанным по формуле:

,

где Q – средняя длина очереди:

,



,

Примечание: для данных формул необходимо принять .

Сравнение необходимо производить для коэффициента загрузки системы ρсмо в диапазоне 0.1­‒ 0.95.

**Отчет по практическому занятию должен содержать:**

1. график с зависимостью среднего времени ожидания в СМО от коэффициента загрузки для имитационной модели и теоретически рассчитанных значений ;
2. график с зависимостью односильной ошибки оценки среднего времени ожидания в СМО от коэффициента загрузки системы имитационной моделью.
3. график с зависимостью односильной ошибки оценки среднего времени ожидания в СМО от числа обслуженных заявок имитационной моделью.

*Примечание.* Относительная ошибка вычисляется как

\*100%

Итоговый отчет должен быть представлен в виде программы. Секции отображения графиков должны быть готовы к отображению по просьбе принимающего. Возможен вариант предварительного формирования графиков – для этого используйте в конце программы вместо *plt.show()* метод *plt.savefig(«имя\_файла.jpg»)* для сохранения графика в директорию проекта.

Будьте готовы ответить на **контрольные вопросы** по практическому занятию:

1. Каким образом устроена нотация Кендалла?
2. Какую модель СМО вы исследовали?
3. Как зависит среднее время ожидания заявок в системе от коэффициента загрузки?
4. Как зависит точность оценок, полученных с помощью ИМ, от числа обработанных заявок?
5. Как зависит точность оценок от коэффициента загрузки системы?

**Как пользоваться ИМ**

1. Задайте параметры для СМО типа М/M/n/r

n = 2

r = 20

l = 1.8

mu = 1.0

smo = smo\_im.SmoIm(n, r)

smo.set\_sources(l, 'M')

smo.set\_servers(mu, 'M')

1. Запустите ИМ, передав методу *run* количество заявок, которые требуется обслужить:

smo.run(100000)

1. Для того, чтобы получить список с вероятностями состояний системы, необходимо вызвать

p = smo.get\_p()

1. Начальные моменты времен ожидания и пребывания в СМО содержатся в списках smo.w, smo.v соответственно.
2. В классе SmoIm метод \_\_str\_\_ перегружен, поэтому при вызове функции print(smo) выводится информация о результатах модулирования в виде:

Queueing system M/M/2/20

Load: 0.900

Sojourn moments:

4.0465 26.2894 216.5013

Wait moments:

3.0434 18.1995 137.8252

Stationary prob:

0.05719 0.10480 0.09277 0.08426 0.07572 0.06925 0.06194 0.05663 0.04906

**Расчет M/M/n/r**

В файле *mmnr\_calc.py* содержатся методы расчета среднего числа заявок в очереди *getQ* и вероятностей состояний системы *getp* для системы M/M/n/r. Ниже приведен листинг файла *mmnr\_calc.py*:

import math  
  
class M\_M\_n\_formula:  
  
 @staticmethod  
 def getPI(l, mu, n, r):  
  
 ro = l / mu  
 p = M\_M\_n\_formula.getp(l, mu, n, r)  
  
 chisl = math.pow(ro, n + r) \* p[0]  
 znam = math.factorial(n) \* math.pow(n, r)  
 return chisl / znam  
  
 @staticmethod  
 def getQ(l, mu, n, r):  
 ro = l / mu  
 p = M\_M\_n\_formula.getp(l, mu, n, r)  
 sum = 0  
 for i in range(1, r+1):  
 sum += i \* math.pow(ro / n, i)  
 return p[n] \* sum  
  
 @staticmethod  
 def getW(l, mu, n, r):  
 q = M\_M\_n\_formula.getQ(l, mu, n , r)  
 w = q\*l  
 return w  
  
 @staticmethod  
 def getV(l, mu, n, r):  
 w = M\_M\_n\_formula.getW(l, mu, n, r)  
 return w + 1/mu  
  
 @staticmethod  
 def getp(l, mu, n, r):  
  
 p = []  
 sum1 = 0  
 ro = l / mu  
 sum2 = 0  
 for i in range(1, r+1):  
 sum2 +=math.pow(ro/n, i)  
 sum2 \*= (math.pow(ro, n)/math.factorial(n))  
  
 for i in range(n+1):  
 sum1 += math.pow(ro, i) / math.factorial(i)  
  
 p.append(1 / (sum1 + sum2))  
  
 for i in range(1, n+r+1):  
 if (i <= n):  
 p.append(math.pow(ro, i) \* p[0] / math.factorial(i))  
 else:  
 p.append(math.pow(ro, i) \* p[0] / math.factorial(n) \* math.pow(n, i - n))  
 return p

Пример расчета средней длины очереди:

import mmnr\_calc

l = 1.0

mu = 0.4

n = 3

r = 50

q = mmnr\_calc.M\_M\_n\_formula.getQ(l, mu, n, r)

print("{0:5.3f}".format(q)) # выведет значение 4.209

**Построение графиков**

Для построения графиков воспользуйтесь библиотекой matplotlib. После формирования массивов, содержащих средние времена пребывания заявок для имитационной модели *w1\_im* и рассчитанных теоретически *w1\_teor*, можно построить график следующим образом

fig, ax = plt.subplots()  
  
ax.plot(roes, w1\_im)  
ax.plot(roes, w1\_teor)  
plt.show()

В данном случае *roes* представляет собой массив значений загрузки СМО. Ниже приведен пример автоматического заполнения массива 20 значениями в диапазоне от 0.1 до 0.9 с помощью метода *linspace* библиотеки *numpy*

roes = np.linspace(0.1, 0.9, 20)

После формирования массива относительных ошибок *error*, соответствующий график можно построить так

fig, ax = plt.subplots()  
ax.plot(roes, error)  
plt.show()

Ниже приведен фрагмент листинга программы, в котором в цикле происходит накопление средних времен ожидания в системе для ИМ и рассчитанных теоретически в зависимости от коэффициента загрузки системы

roes = np.linspace(0.1, 0.9, 20)  
w1\_im = []  
w1\_teor = []  
error = []  
for ro in roes:  
 smo = smo\_im.SmoIm(n, r)  
 smo.set\_sources(l, 'M')  
 mu = l/(ro\*n)  
 smo.set\_servers(mu, 'M')  
 smo.run(jobs\_count)  
 q = mmnr\_calc.M\_M\_n\_formula.getQ(l, mu, n, r)  
 w1\_teor.append(q/l)  
 w1\_im.append(smo.w[0])  
 error.append(100\*(q/l - smo.w[0])/(q/l))

Для формирования графика относительной ошибки оценки среднего времени ожидания в СМО имитационной моделью от числа обслуженных заявок следует также воспользоваться библиотекой *matplotlib*

fig, ax = plt.subplots()  
ax.plot(jobs\_count, error, label="относ ошибка ИМ")  
plt.legend()  
plt.show()

В данном случае jobs\_count представляет собой массив, содержащий значения числа обслуженных заявок в диапазоне от 10000 до 300000

jobs\_count = [x\*10000 for x in range(1, 30)]

Массив *error* следует сформировать в цикле, по аналогии с ранее представленным листингом, сделав соответствующие изменения.