**Практическое занятие № 4**

**Тема:** Расчет СМО GI/M/1.

**Цель:** Приобретение практических навыков расчета показателей оперативности обработки данных с помощью системы массового обслуживания GI/M/1.

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

import smo\_im  
import rand\_destribution as rd  
import numpy as np  
import matplotlib.pyplot as plt  
import gi\_m\_1\_calc

1. В ходе практического занятия необходимо сравнить результаты расчетов среднего времени пребывания в системе, полученные с помощью метода, предложенного Такачем [1, 2 стр. 158], с результатами ИМ.

Оценку среднего времени пребывания в системе необходимо произвести в зависимости от коэффициента загрузки системы **ρ**. Значение начальных моментов распределения интервалов между соседними заявками входного потока следует определить из таблицы. Номер варианта соответствует номеру по журналу.

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Начальные моменты | Вариант | Начальные моменты |
| 1 | 1.00, 1.49, 2.95 | 9 | 1.00, 14.26, 392.71 |
| 2 | 1.00, 1.86, 5.05 | 10 | 1.00, 15.96, 493.78 |
| 3 | 1.00, 2.33, 8.52 | 11 | 1.00, 17.77, 613.56 |
| 4 | 1.00, 2.90, 13.94 | 12 | 1.00, 19.67, 754.26 |
| 5 | 1.00, 3.58, 22.01 | 13 | 1.00, 21.68, 918.24 |
| 6 | 1.00, 4.35, 33.57 | 14 | 1.00, 23.79, 1107.96 |
| 7 | 1.00, 12.67, 308.27 | 15 | 1.00, 9.78, 181.53 |
| 8 | 1.00, 11.17, 238.49 | 16 | 1.00, 8.49, 135.68 |

В ходе практического занятия необходимо:

* Вывести таблицу с зависимостью среднего времени пребывания в системе GI/M/1 для ИМ и рассчитанные с помощью метода Такача от коэффициента загрузки системы;
* Построить график по полученной таблице;
* Построить график для относительной ошибки ИМ.
* Проделать тоже самое, увеличив значение второго (третьего) момента распределения интервалов между соседними заявками входного потока вдвое.

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 пребывания заявок в системе от коэффициента загрузки?
4. Как зависит среднее время пребывания заявок в системе от коэффициента вариации распределения интервалов между соседними заявками?
5. Что произойдет со значением среднего времени пребывания заявок в исследуемой системе, если увеличить значение второго и третьего момента распределения интервалов между соседними заявками? Что произойдет со значением среднего времени пребывания заявок в системе M/G/1, если увеличить значение второго и третьего момента распределения времени обслуживания?
6. Как зависит точность оценок, полученных с помощью ИМ, от числа обработанных заявок?
7. Как зависит точность оценок от коэффициента загрузки системы?
8. В чем идея метода Такача?

**Как пользоваться ИМ**

Для запуска ИМ нужно создать экземпляр класса SmoIm, передав ему количество каналов обслуживания *n*:

smo = smo\_im.SmoIm(n)

Далее необходимо задать входной поток с помощью метода *set\_sources()*, передав список в виде [*params, type*], где *params* – список с параметрами распределения, *type* – тип распределения, заданный в виде текста. Ниже приведен пример для экспоненциального распределения времени обслуживания:

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

Распределение интервала между соседними заявками входного потока задается аналогично, с помощью метода set\_servers(). Ниже приведен пример для Гамма-распределения.

smo.set\_sources([v, alpha], "Gamma")

Чтобы узнать о поддерживаемых типах распределений (в том числе и распределений входного потока), ознакомьтесь с содержимым класса *Server*, содержащемся в файле *smo\_im.py*. Ниже приведен его фрагмент:

class Server:  
 *"""  
 Канал обслуживания  
 """* id = 0  
  
 def \_\_init\_\_(self, params, types):  
 *"""  
 params - параметры распределения  
 types - тип распределения  
 """* if types == "M":  
 self.dist = rd.Exp\_dist(params)  
 elif types == "H":  
 self.dist = rd.H2\_dist(params)  
 elif types == "E":  
 self.dist = rd.Erlang\_dist(params)  
 elif types == "C":  
 self.dist = rd.Cox\_dist(params)  
 elif types == "Gamma":  
 self.dist = rd.Gamma(params)  
 elif types == "Pa":  
 self.dist = rd.Pareto\_dist(params)  
 elif types == "Uniform":  
 self.dist = rd.Uniform\_dist(params)  
 elif types == "D":  
 self.dist = rd.Det\_dist(params)  
 else:  
 raise SetSmoException("Неправильно задан тип распределения сервера. Варианты М, Н, Е, С, Pa, Uniform, D")

Далее необходимо запустить ИМ с помощью метода *run()*, передав количество подлежащих обслуживанию заявок. Начальные моменты времени пребывания можно получить, обратившись к полю ***v*** экземпляра класса. В частности, для получения среднего времени пребывания необходимо вызвать *smo.v[0]*.

jobs\_count = 100000  
smo.run(jobs\_count)  
v\_im.append(smo.v[0])

Ниже приведен код для накопления массивов средних времен пребывания, полученных с помощью ИМ и рассчитанных с помощью метода Такача, в зависимости от коэффициента загрузки системы. Также производится накопление массива относительных ошибок ИМ *v\_errors*

a = [1, 4, 140]  
roes = np.linspace(0.1, 0.9, 6)  
num\_of\_jobs = 800000  
  
vs\_ch = []  
vs\_im = []  
  
v\_errors = []  
  
for ro in roes:  
  
 mu = 1 / ro  
 v, alpha = rd.Gamma.get\_mu\_alpha(a)  
 a = rd.Gamma.calc\_theory\_moments(v, alpha)  
 v\_ch = gi\_m\_1\_calc.get\_v(a, mu)  
 p\_ch = gi\_m\_1\_calc.get\_p(a, mu)  
  
 smo = smo\_im.SmoIm(1)  
 smo.set\_sources([v, alpha], "Gamma")  
 smo.set\_servers(mu, "M")  
 smo.run(num\_of\_jobs)  
 v\_im = smo.v  
 p\_im = smo.get\_p()  
  
 v\_ch = gi\_m\_1\_calc.get\_v(a, mu)  
 v\_im = smo.v  
  
 vs\_ch.append(v\_ch[0])  
 vs\_im.append(v\_im[0])  
 # v\_errors.append(100 \* (v\_im[0] - v\_ch[0]) / v\_ch[0])  
  
 print("\nЗначения начальных моментов времени ожидания заявок в системе:\n")  
  
 print("{0:^15s}|{1:^15s}|{2:^15s}".format("№ момента", "Числ", "ИМ"))  
 print("-" \* 45)  
 for j in range(3):  
 print("{0:^16d}|{1:^15.5g}|{2:^15.5g}".format(j + 1, v\_ch[j], v\_im[j]))  
  
 print("{0:^25s}".format("Вероятности состояний СМО"))  
 print("{0:^3s}|{1:^15s}|{2:^15s}".format("№", "Числ", "ИМ"))  
 print("-" \* 32)  
 for i in range(11):  
 print("{0:^4d}|{1:^15.3g}|{2:^15.3g}".format(i, p\_ch[i], p\_im[i]))

Аналогично следует накопить значения *vs\_im*, v*s\_ch, v\_errors* в зависимости от коэффициента вариации интервалов между соседними заявками.

**Построение графиков**

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

fig, ax = plt.subplots()  
ax.plot(roes, vs\_im, label="ИМ")  
ax.plot(roes, vs\_ch, label="Числ")  
  
ax.plot(roes, v\_errors, label="относ ошибка ИМ")  
plt.legend()  
plt.show()

В данном случае *roes* представляет собой массив значений коэффициентов загрузки системы.

**Литература**

1. Takacs L. Introduction to the Theory of Queues. N.Y.: Oxford Univ. Press, 1960. 12 p
2. Рыжиков Ю.И. Алгоритмический подход к задачам массового обслуживания: монография / Ю.И. Рыжиков. СПб.: ВКА им. А.Ф. Можайского, 2013. 496 с