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
```

4. 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 M, H, E, C, 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