Практическое занятие № 5

Тема: Расчет СМО GI/M/n.
Цель: Приобретение практических навыков расчета показателей оперативности и определения оптимального числа каналов с помощью системы массового обслуживания GI/M/n.
Язык программирования, ПО и библиотеки: python 3.x, установленный пакет библиотек Anaconda.

Порядок выполнения практического занятия

1. Создайте новый проект. Скопируйте файлы *pz5.py* и *pz5_cost.py* в свою локальную папку.

2. Добавьте файлы с библиотекой генерации случайных величин (*rand_destribution.py*) и имитационной моделью (*smo_im.py*) в директорию с файлом *pz5.py.* Также в указанную директорию необходимо поместить библиотеки, обеспечивающие расчет СМО GI/M/n:

- `gi_m_n_calc.py`;
- `sv_sum_calc.py`;
- `diff5dots.py`.

В ходе практического занятия необходимо:

1) Сравнить результаты расчетов среднего времени пребывания в системе, полученные с помощью метода, предложенного Такачем [1, 2 стр. 158], с результатами имитационного моделирования (ИМ).
2) Определить оптимальное число каналов системы при заданных параметрах.

Часть 1. Сравнение оценок ИМ с численным расчетом методом Такача

1. Откройте *pz5.py* и запустите программу. В результате ее выполнения должны быть построены графики, подобные представленным на рисунках 1 и 2.

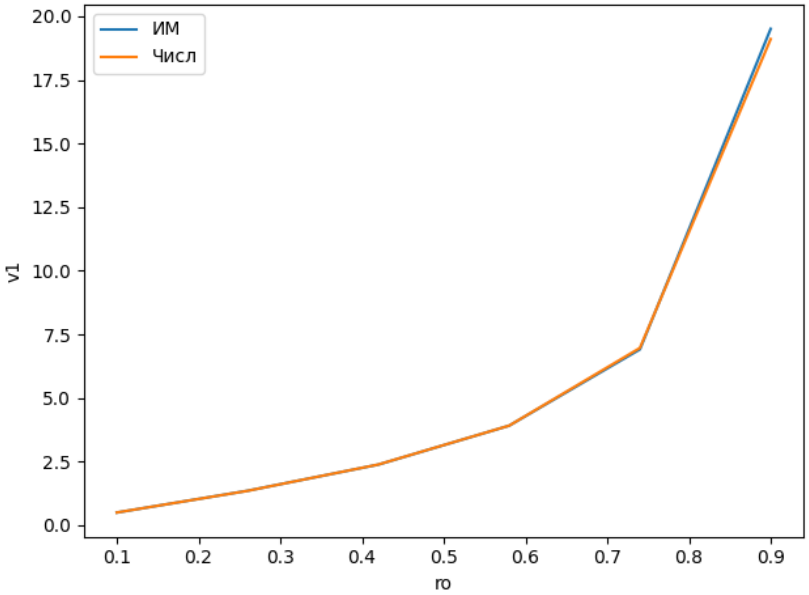

Рисунок 1. Зависимость среднего времени пребывания заявок в СМО от коэффициента загрузки

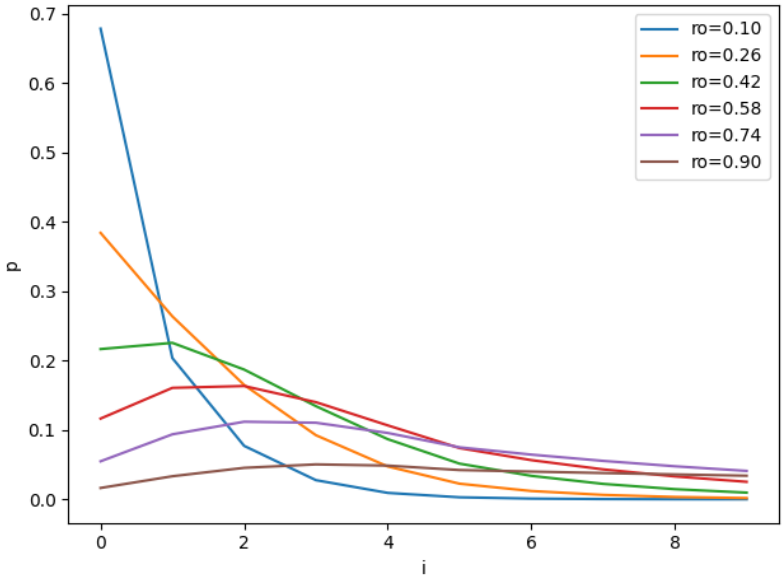

Рисунок 2. Распределение вероятностей состояний в СМО

На рисунке 1 представлены два графика с зависимостями среднего времени пребывания заявок в СМО от коэффициента загрузки, полученные в ходе ИМ и в результате численных расчетов методом Такача. Как видно из графиков, оценки ИМ достаточно близки к полученным расчетным значениям. На рисунке 2 представлено распределение вероятностей состояний в СМО,

полученные при различных значениях коэффициента загрузки. Также в процессе выполнения программы выводятся таблицы, подобные представленным ниже.

```
-------------------------------------------------------------
   № момента   |     Числ      |      ИМ       |    Err, %
-------------------------------------------------------------
       1       |    19.108     |     19.03     |   -0.40778
       2       |    674.65     |    600.42     |    -11.003
       3       |     34832     |     25206     |    -27.635
Вероятности состояний СМО
----------------------------------
 № |     Числ      |      ИМ
----------------------------------
 0  |    0.0164     |    0.0163
 1  |    0.0332     |    0.0291
 2  |    0.0454     |    0.0417
 3  |    0.0503     |    0.0484
 4  |    0.0486     |    0.0438
 5  |    0.0422     |    0.0386
 6  |     0.04      |    0.0379
 7  |    0.0379     |    0.0369
 8  |    0.0359     |    0.0352
 9  |     0.034     |    0.0341
 10 |    0.0322     |    0.0305
Среднее время моделирования, сек 0.417
Среднее время численного расчета, сек 0.005
Ускорение в 80.000 раз
```

Здесь под «Err» обозначена относительная погрешность оценок ИМ для начальных моментов от 1 до 3, где первый начальный момент – среднее время пребывания заявок в системе. Обратите внимание на то, во сколько раз численный расчет осуществляется быстрее ИМ.

2. Проанализируем код программы. В строках ниже задаются параметры модели – средний интервал между соседними заявками входного потока a_1, коэффициент вариации распределения интервалов между соседними заявками во входном потоке *coev,* число заявок *num_of_jobs*, требуемых к обслуживанию ИМ, число каналов СМО *n*:

```
a1 = 1
coev = 1.7
num_of_jobs = 100000
n = 5
```

Далее создаются пустые массивы для накопления исследуемых значений – два массива для накопления средних времен пребывания заявок в системе, полученных численным методом и в результате ИМ:

```
vs_ch = []
vs_im = []
```

массив для накопления стационарных вероятностей состояний системы:

```
p_ch_mass = []
```

массивы для накопления оценок времени расчета и ИМ:

```
im_times = []
ch_times = []
```

Далее запускается цикл, в котором осуществляется ИМ и численный расчет для каждого значения коэффициента загрузки СМО.

Требуемое значение коэффициента загрузки СМО будем получать изменением интенсивности обслуживания заявок μ. В строках ниже представлен фрагмент, в котором осуществляется пересчет значения μ, вызов метода *get_mu_alpha_by_mean_and_coev()* класса Гамма-распределения для определения параметров аппроксимирующего распределения входного потока заявок по двум параметрам – среднему значению и коэффициенту вариации:

```
roes = np.linspace(0.1, 0.9, 6)

for ro in roes:

    mu = 1 / (ro * n)
    v, alpha = rd.Gamma.get_mu_alpha_by_mean_and_coev(a1, coev)
```

Далее происходит вызов функции *calc_theory_moments()* для расчета теоретических значений начальных моментов распределения входного потока ***a*** (представляет собой список из трех начальных моментов, можете убедиться в этом, вызвав *print(a)*). По найденным значениям начальных моментов ***a*** осуществляется расчет начальных моментов времени пребывания заявок в системе *v_ch* с помощью уже написанной функции *get_v()*, которая находится в библиотеке *gi_m_n_calc*. Ниже также производится расчет вероятностей состояний системы *p_ch*. Переменная *start* нужна для профилирования – определения времени расчета.

```
a = rd.Gamma.calc_theory_moments(v, alpha)

start = time.process_time()
v_ch = gi_m_n_calc.get_v(a, mu, n)
ch_times.append(time.process_time() - start)

p_ch = gi_m_n_calc.get_p(a, mu, n)
p_ch_mass.append(p_ch)
```

Ниже происходит уже знакомый нам по предыдущим практическим занятиям вызов ИМ. На этот раз входной поток задан Гамма-распределением с параметрами ν и α, а время обслуживания – экспоненциальным с параметром μ:

```
start = time.process_time()
smo = smo_im.SmoIm(n)
smo.set_sources([v, alpha], "Gamma")
smo.set_servers(mu, "M")
smo.run(num_of_jobs)
v_im = smo.v
im_times.append(time.process_time() - start)

p_im = smo.get_p()
```

Дальнейший код связан с выводом полученных значений и построением графиков.

3. Далее вам необходимо повторить расчеты при новых значениях параметров *coev* и *n* СМО. Для этого определите упомянутые значения из таблицы 1.

Таблица 1. Выбор варианта выполнения практического задания

Номер по журналу	ρ_{fix}	coev	V_{cost}	N_{cost}	n
1	0.65	0.3	1.0	0.5	1
2	0.7	1.57	2.0	1.0	2
3	0.75	1.2	3.0	1.0	3
4	0.8	1.5	1.0	0.7	4
5	0.85	2.4	2.5	1.0	5
6	0.9	3.67	3.5	1.0	6
7	0.65	1.4	2.9	0.7	7
8	0.7	2.7	4.1	1.1	1
9	0.75	2.45	2.0	0.5	2
10	0.8	5.18	1.9	1.4	3
11	0.85	2.9	4.0	1.1	4
12	0.9	3.5	3.0	0.7	5
13	0.65	1.2	4.5	0.2	6
14	0.7	2.9	2.5	0.8	7
15	0.75	2.6	1.9	0.7	1
16	0.8	0.9	1.2	0.9	2
17	0.85	1.45	2.9	1.1	3
18	0.9	1.11	3.1	1.7	4

Здесь:

coev и *n* – параметры, которые задаются в начале программы – коэффициент вариации распределения интервалов между заявками входного потока и число каналов обслуживания;

ρ_{fix} – значение фиксированного коэффициента загрузки одноканальной СМО, которое в будем использовать для отыскания оптимального числа каналов;

V_{cost} и N_{cost} – значения для определения функции стоимости.

Задайте *coev* и *n* и запустите программу на выполнение. Сделайте выводы о зависимости времени пребывания в СМО от коэффициента загрузки. Посмотрите внимательно на график с распределением вероятностей состояний СМО. Что происходит с вероятностями состояний СМО при увеличении коэффициента загрузки и почему?

Часть 2. Определение оптимального числа каналов

1. Функция стоимости обслуживания системы будет вычисляться следующим образом

$$Z = v_1 V_{cost} + nN_{cost}, \quad (1)$$

где v_1 – среднее время пребывания заявок в СМО.

Таким образом, стоимость обслуживания системы будет расти при увеличении числа каналов, что можно объяснить расходами на электроэнергию, если канал – это, например, сервер. Или расходами на заработную плату, если, исследуемая система, например, магазин. Также стоимость обслуживания системы будет расти при увеличении времени пребывания заявок в системе. Это также можно представить в виде штрафов за несоблюдение условий по качеству обслуживания для распределенных систем или потерями при уходе клиентов, если исследуемая система магазин.

2. Определите значения параметров ρ_{fix}, и V_{cost} и N_{cost} по таблице 1.

3. Откройте файл *pz5_cost.py*. Ниже представлен его код:

```
a1 = 1
coev = 2.7

ns = [x for x in range(1, 10)]

ro_fix = 0.7
mu = 1 / (ro_fix)
v_cost = 1
n_cost = 0.7
```

```
total_costs = []
for n in ns:
    # !!! добавьте свой код здесь
    pass

print("Минимальное значение стоимости: {0:1.3f} при n =
{1:d}".format(min(total_costs), np.argmin(total_costs)+1))
fig, ax = plt.subplots()
ax.plot(ns, total_costs)
ax.set_xlabel('n')
ax.set_ylabel('Стоимость')
ax.xaxis.set_major_locator(MaxNLocator(integer=True))
ax.set_title("Распределение стоимости обслуживания системы")
plt.show()
```

4. Задайте параметры *coev, ro_fix, v_cost, n_cost* согласно вашему варианту. Расчет оптимального значения числа каналов будем определять численно, поэтому библиотека *smo_im* не включена в раздел импорта.

Вам необходимо заменить текст *pass* в цикле на свой код. В коде необходимо на каждом шаге производить численный расчет среднего времени пребывания в СМО (пример вызова функции – в части 1). Полученное значение необходимо использовать для расчета стоимости, вызывая функцию *get_cost(),* которая реализует формулу (1). Значения функции стоимости следует накапливать в массиве *total_cost*. После верной реализации цикла должен получиться график похожий на представленный на рисунке 3, а также вывод оптимального значения числа каналов в консоль.

Рисунок 3. График функции стоимости от числа каналов

После успешного выполнения программы попробуйте изменить значения коэффициента вариации распределения входного потока *coev* и числа каналов обслуживания *n*. Как изменится вид графика функции стоимости?

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) Как зависит точность оценок, полученных с помощью ИМ, от числа обработанных заявок?
6) Как зависит точность оценок от коэффициента загрузки системы?
7) В чем идея метода Такача?
8) Всегда для функции стоимости, представленной формулой (1), будет минимальное значение?

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