МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
федеральное государственное автономное образовательное учреждение высшего образования
«Санкт–Петербургский государственный университет
аэрокосмического приборостроения»

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ _____

преподаватель		
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине МДК 02.03

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	С326		Э.С. Тигранян
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

Лабораторная работа No5.

Решение задач массового обслуживания методами имитационного моделирования

Цель работы: научиться оценивать надежность простейших систем методом Монте-Карло; научиться рассчитывать СМО с отказами методом Монте-Карло.

Вариант 3

1. Система состоит из двух блоков, соединенных последовательно. Первый блок содержит три элемента: A, B, C, а второй – два элемента: D, E. Элементы каждого блока соединены параллельно.

а) Найти методом Монте-Карло оценку P* надежности системы, зная вероятности безотказной работы элементов: P(A)=0,9; P(B)=0,88; P(C)=0,95; P(D)=0,8; P(E)=0,7;

б) найти абсолютную погрешность |P*-P|, где P- надежность системы, вычисленная аналитически. Произвести 85 испытаний.

2. В двухканальную систему массового обслуживания с отказом поступает пуассоновский поток заявок. Время между поступлениями двух последовательных заявок распределено по показательному закону $f(\tau) = 4e^{-4\tau}$.

Длительность обслуживания каждой заявки равна 1 мин. Найти методом Монте-Карло математическое ожидание a числа обслуженных заявок за время

T=6 мин.

Задание 1:

Дано:

- Система состоит из двух блоков, соединенных последовательно.
- Первый блок: элементы A, B, C соединены параллельно.
- Второй блок: элементы D, E соединены параллельно.
- Вероятности безотказной работы:
  - $P(A) = 0.9$
  - $P(B) = 0.88$
  - $P(C) = 0.95$
  - $P(D) = 0.8$
  - $P(E) = 0.7$
- Количество испытаний: $N = 85$

Шаг 1. Аналитический расчёт надёжности системы

Первый блок (параллельное соединение A, B, C):

Отказ блока происходит, если откажут все три элемента:

$$P_{отк.1} = (1 - P(A)) \cdot (1 - P(B)) \cdot (1 - P(C))$$

$$P_{отк.1} = (1 - 0.9)(1 - 0.88)(1 - 0.95) = 0.1 \cdot 0.12 \cdot 0.05 = 0.0006$$

$$P_1 = 1 - P_{отк.1} = 1 - 0.0006 = 0.9994$$

Второй блок (параллельное соединение D, E):

$$P_{отк.2} = (1 - P(D)) \cdot (1 - P(E)) = (1 - 0.8)(1 - 0.7) = 0.2 \cdot 0.3 = 0.06$$

$$P_2 = 1 - P_{отк.2} = 1 - 0.06 = 0.94$$

Общая надёжность системы (последовательное соединение блоков):

$$P = P_1 \cdot P_2 = 0.9994 \cdot 0.94 \approx 0.9394$$

Шаг 2. Имитационное моделирование по методу Монте-Карло

Проводим $N = 85$ испытаний. Для каждого элемента генерируем случайное число $r \sim U(0,1)$. Если $r < P(\text{элемент})$, элемент работает.

Правила:

- Первый блок работает, если хотя бы один из A, B, C работает.
- Второй блок работает, если хотя бы один из D, E работает.
- Система работает, если оба блока работают.

85 испытаний:

Код на python:

```
print("\n" + "=" * 50)
print("ЗАДАНИЕ 1: Расчет надежности системы")
print("=" * 50)

# Вероятности безотказной работы элементов
P_A, P_B, P_C = 0.9, 0.88, 0.95
P_D, P_E = 0.8, 0.7

# Аналитический расчет
P_otkaz_1 = (1 - P_A) * (1 - P_B) * (1 - P_C)
P1 = 1 - P_otkaz_1

P_otkaz_2 = (1 - P_D) * (1 - P_E)
P2 = 1 - P_otkaz_2

P_analytical = P1 * P2

print(f"Аналитический расчет:")
print(f"P1 (блок 1) = 1 - (1-{P_A})*(1-{P_B})*(1-{P_C}) = {P1:.4f}")
print(f"P2 (блок 2) = 1 - (1-{P_D})*(1-{P_E}) = {P2:.4f}")
print(f"P (система) = P1 * P2 = {P_analytical:.4f}")

# Метод Монте-Карло
np.random.seed(42)  # Для воспроизводимости результатов
N_tests = 85
success_count = 0

print(f"\nМетод Монте-Карло ({N_tests} испытаний):")

# Таблица для результатов
results = []

for i in range(N_tests):
    # Генерирация случайных чисел для каждого элемента
    r_A, r_B, r_C = np.random.random(), np.random.random(),
np.random.random()
    r_D, r_E = np.random.random(), np.random.random()

    # Работоспособность элементов
    work_A = 1 if r_A < P_A else 0
    work_B = 1 if r_B < P_B else 0
    work_C = 1 if r_C < P_C else 0
    work_D = 1 if r_D < P_D else 0
    work_E = 1 if r_E < P_E else 0

    # Работоспособность блоков
    block1 = 1 if (work_A + work_B + work_C) >= 1 else 0
    block2 = 1 if (work_D + work_E) >= 1 else 0
```

```
    # Работоспособность системы
    system_works = 1 if (block1 == 1 and block2 == 1) else 0
    success_count += system_works

    results.append([i + 1, r_A, r_B, r_C, r_D, r_E,
                    work_A, work_B, work_C, work_D, work_E,
                    block1, block2, system_works])

# DataFrame для вывода
df = pd.DataFrame(results, columns=[
    'Испытание', 'r_A', 'r_B', 'r_C', 'r_D', 'r_E',
    'A', 'B', 'C', 'D', 'E', 'Блок1', 'Блок2', 'Система'
])

# Расчет оценки надежности
P_monte_carlo = success_count / N_tests
error = abs(P_analytical - P_monte_carlo)

print(f"Количество успешных испытаний: {success_count} из {N_tests}")
print(f"Оценка надежности P* = {success_count}/{N_tests} =
{P_monte_carlo:.4f}")
print(f"Абсолютная погрешность |P - P*| = {error:.4f}")

print(f"\nИспытания:")
print(df.head(85).to_string(index=False))
```

Результат:

Аналитический расчет:

P1 (блок 1) = 1 - (1-0.9)*(1-0.88)*(1-0.95) = 0.9994

P2 (блок 2) = 1 - (1-0.8)*(1-0.7) = 0.9400

P (система) = P1 * P2 = 0.9394

Испытания:

Испытание	r_A	r_B	r_C	r_D	r_E	A	B	C	D	E	Блок1	Блок2	Система
1	0.374540	0.950714	0.731994	0.598658	0.156019	1	0	1	1	1	1	1	1
2	0.155995	0.058084	0.866176	0.601115	0.708073	1	1	1	1	0	1	1	1
3	0.020584	0.969910	0.832443	0.212339	0.181825	1	0	1	1	1	1	1	1
4	0.183405	0.304242	0.524756	0.431945	0.291229	1	1	1	1	1	1	1	1

5 0.611853 0.139494 0.292145 0.366362 0.456070 1 1 1 1 1 1 1
1

6 0.785176 0.199674 0.514234 0.592415 0.046450 1 1 1 1 1 1 1
1

7 0.607545 0.170524 0.065052 0.948886 0.965632 1 1 1 0 0 1 0
0

8 0.808397 0.304614 0.097672 0.684233 0.440152 1 1 1 1 1 1 1
1

9 0.122038 0.495177 0.034389 0.909320 0.258780 1 1 1 0 1 1 1
1

10 0.662522 0.311711 0.520068 0.546710 0.184854 1 1 1 1 1 1 1
1

11 0.969585 0.775133 0.939499 0.894827 0.597900 0 1 1 0 1 1 1
1

12 0.921874 0.088493 0.195983 0.045227 0.325330 0 1 1 1 1 1 1
1

13 0.388677 0.271349 0.828738 0.356753 0.280935 1 1 1 1 1 1 1
1

14 0.542696 0.140924 0.802197 0.074551 0.986887 1 1 1 1 0 1 1
1

15 0.772245 0.198716 0.005522 0.815461 0.706857 1 1 1 0 0 1 0
0

16 0.729007 0.771270 0.074045 0.358466 0.115869 1 1 1 1 1 1 1
1

17 0.863103 0.623298 0.330898 0.063558 0.310982 1 1 1 1 1 1 1
1

18 0.325183 0.729606 0.637557 0.887213 0.472215 1 1 1 0 1 1 1
1

19 0.119594 0.713245 0.760785 0.561277 0.770967 1 1 1 1 0 1 1
1

20 0.493796 0.522733 0.427541 0.025419 0.107891 1 1 1 1 1 1 1 1

21 0.031429 0.636410 0.314356 0.508571 0.907566 1 1 1 1 0 1 1 1

22 0.249292 0.410383 0.755551 0.228798 0.076980 1 1 1 1 1 1 1 1

23 0.289751 0.161221 0.929698 0.808120 0.633404 1 1 1 0 1 1 1 1

24 0.871461 0.803672 0.186570 0.892559 0.539342 1 1 1 0 1 1 1 1

25 0.807440 0.896091 0.318003 0.110052 0.227935 1 0 1 1 1 1 1 1

26 0.427108 0.818015 0.860731 0.006952 0.510747 1 1 1 1 1 1 1 1

27 0.417411 0.222108 0.119865 0.337615 0.942910 1 1 1 1 0 1 1 1

28 0.323203 0.518791 0.703019 0.363630 0.971782 1 1 1 1 0 1 1 1

29 0.962447 0.251782 0.497249 0.300878 0.284840 0 1 1 1 1 1 1 1

30 0.036887 0.609564 0.502679 0.051479 0.278646 1 1 1 1 1 1 1 1

31 0.908266 0.239562 0.144895 0.489453 0.985650 0 1 1 1 0 1 1 1

32 0.242055 0.672136 0.761620 0.237638 0.728216 1 1 1 1 0 1 1 1

33 0.367783 0.632306 0.633530 0.535775 0.090290 1 1 1 1 1 1 1 1

34 0.835302 0.320780 0.186519 0.040775 0.590893 1 1 1 1 1 1 1 1

35 0.677564 0.016588 0.512093 0.226496 0.645173 1 1 1 1 1 1 1
1
36 0.174366 0.690938 0.386735 0.936730 0.137521 1 1 1 0 1 1 1
1
37 0.341066 0.113474 0.924694 0.877339 0.257942 1 1 1 0 1 1 1
1
38 0.659984 0.817222 0.555201 0.529651 0.241852 1 1 1 1 1 1 1
1
39 0.093103 0.897216 0.900418 0.633101 0.339030 1 0 1 1 1 1 1
1
40 0.349210 0.725956 0.897110 0.887086 0.779876 1 1 1 0 0 1 0
0
41 0.642032 0.084140 0.161629 0.898554 0.606429 1 1 1 0 1 1 1
1
42 0.009197 0.101472 0.663502 0.005062 0.160808 1 1 1 1 1 1 1
1
43 0.548734 0.691895 0.651961 0.224269 0.712179 1 1 1 1 0 1 1
1
44 0.237249 0.325400 0.746491 0.649633 0.849223 1 1 1 1 0 1 1
1
45 0.657613 0.568309 0.093675 0.367716 0.265202 1 1 1 1 1 1 1
1
46 0.243990 0.973011 0.393098 0.892047 0.631139 1 0 1 0 1 1 1
1
47 0.794811 0.502637 0.576904 0.492518 0.195243 1 1 1 1 1 1 1
1
48 0.722452 0.280772 0.024316 0.645472 0.177111 1 1 1 1 1 1 1
1
49 0.940459 0.953929 0.914864 0.370159 0.015457 0 0 1 1 1 1 1
1

50 0.928319 0.428184 0.966655 0.963620 0.853009 0 1 0 0 0 1 0
0
51 0.294449 0.385098 0.851137 0.316922 0.169493 1 1 1 1 1 1 1
1
52 0.556801 0.936155 0.696030 0.570061 0.097176 1 0 1 1 1 1 1
1
53 0.615007 0.990054 0.140084 0.518330 0.877373 1 0 1 1 0 1 1
1
54 0.740769 0.697016 0.702484 0.359491 0.293592 1 1 1 1 1 1 1
1
55 0.809361 0.810113 0.867072 0.913241 0.511342 1 1 1 0 1 1 1
1
56 0.501516 0.798295 0.649964 0.701967 0.795793 1 1 1 1 0 1 1
1
57 0.890005 0.337995 0.375583 0.093982 0.578280 1 1 1 1 1 1 1
1
58 0.035942 0.465598 0.542645 0.286541 0.590833 1 1 1 1 1 1 1
1
59 0.030500 0.037348 0.822601 0.360191 0.127061 1 1 1 1 1 1 1
1
60 0.522243 0.769994 0.215821 0.622890 0.085347 1 1 1 1 1 1 1
1
61 0.051682 0.531355 0.540635 0.637430 0.726091 1 1 1 1 0 1 1
1
62 0.975852 0.516300 0.322956 0.795186 0.270832 0 1 1 1 1 1 1
1
63 0.438971 0.078456 0.025351 0.962648 0.835980 1 1 1 0 0 1 0
0
64 0.695974 0.408953 0.173294 0.156437 0.250243 1 1 1 1 1 1 1
1

65 0.549227 0.714596 0.660197 0.279934 0.954865 1 1 1 1 0 1 1
1

66 0.737897 0.554354 0.611721 0.419600 0.247731 1 1 1 1 1 1 1
1

67 0.355973 0.757846 0.014393 0.116073 0.046003 1 1 1 1 1 1 1
1

68 0.040729 0.855461 0.703658 0.474174 0.097834 1 1 1 1 1 1 1
1

69 0.491616 0.473472 0.173202 0.433852 0.398505 1 1 1 1 1 1 1
1

70 0.615850 0.635094 0.045304 0.374613 0.625860 1 1 1 1 1 1 1
1

71 0.503136 0.856490 0.658694 0.162934 0.070569 1 1 1 1 1 1 1
1

72 0.642419 0.026511 0.585776 0.940230 0.575474 1 1 1 0 1 1 1
1

73 0.388170 0.643288 0.458253 0.545617 0.941465 1 1 1 1 0 1 1
1

74 0.386103 0.961191 0.905351 0.195791 0.069361 1 0 1 1 1 1 1
1

75 0.100778 0.018222 0.094443 0.683007 0.071189 1 1 1 1 1 1 1
1

76 0.318976 0.844875 0.023272 0.814468 0.281855 1 1 1 0 1 1 1
1

77 0.118165 0.696737 0.628943 0.877472 0.735071 1 1 1 0 0 1 0
0

78 0.803481 0.282035 0.177440 0.750615 0.806835 1 1 1 1 0 1 1
1

79 0.990505 0.412618 0.372018 0.776413 0.340804 0 1 1 1 1 1 1
1

80 0.930757 0.858413 0.428994 0.750871 0.754543 0 1 1 1 0 1 1 1

81 0.103124 0.902553 0.505252 0.826457 0.320050 1 0 1 0 1 1 1 1

82 0.895523 0.389202 0.010838 0.905382 0.091287 1 1 1 0 1 1 1 1

83 0.319314 0.950062 0.950607 0.573438 0.631837 1 0 0 1 1 1 1 1

84 0.448446 0.293211 0.328665 0.672518 0.752375 1 1 1 1 0 1 1 1

85 0.791579 0.789618 0.091206 0.494420 0.057559 1 1 1 1 1 1 1 1

В 79 испытаниях система работала. Тогда:

$$P^* = \frac{79}{85} = 0.9294$$

Шаг 3. Абсолютная погрешность

$$| P - P^* | = | 0.9394 - 0.9294 | = 0.01$$

Аналитическая надёжность: $P = 0.9394$

Оценка Монте-Карло: $P^* = 0.9294$

Абсолютная погрешность: $= 0.01$

Задание 2. Расчёт СМО с отказами методом Монте-Карло

Дано:

- Двухканальная СМО с отказами.
- Время между заявками: $\tau \sim f(\tau) = 4e^{-4\tau}; \lambda = 4$
- Время обслуживания: $t_{обс} = 1$ мин.
- Время моделирования: $T = 6$ мин.

Шаг 1. Моделирование потока заявок

Момент поступления заявки:

$$T_i = T_{i-1} + \tau_i, \tau_i = -\frac{1}{\lambda}\ln r_i = -0.25\ln r_i$$

Шаг 2. Алгоритм моделирования

- Каналы 1 и 2 начинают свободны.
- Заявка поступает:
  - Если есть свободный канал → обслуживается, канал занят до $T_i + 1$.
  - Если оба канала заняты → отказ.
- Моделируем до $T_i > 6$.

6 испытаний:

Код на Python:

```
# Параметры системы
lambda_param = 4  # λ = 4
service_time = 1  # Время обслуживания = 1 мин
T_max = 6  # Время моделирования = 6 мин
N_experiments = 6  # Количество испытаний

print(f"Параметры СМО:")
print(f"- λ = {lambda_param}")
print(f"- Время обслуживания = {service_time} мин")
print(f"- Время моделирования T = {T_max} мин")
print(f"- Количество испытаний = {N_experiments}")


def simulate_smo(experiment_num):
    """Моделирование одного испытания СМО"""
    np.random.seed(experiment_num * 10)  # Разные seed для каждого испытания
```

```
    # Начальные условия
    time_current = 0
    channel1_free = 0  # Время освобождения канала 1
    channel2_free = 0  # Время освобождения канала 2
    served_count = 0
    request_count = 0

    results = []

    # Первая заявка
    request_count += 1
    arrival_time = 0
    # Назначение на канал 1
    channel1_free = arrival_time + service_time
    served_count += 1
    results.append([request_count, np.nan, np.nan, np.nan, arrival_time,
                    channel1_free, channel2_free, 1, served_count])

    # Обработка последующих заявок
    while True:
        request_count += 1

        # Генерирация времени до следующей заявки
        r = np.random.random()
        tau = - (1 / lambda_param) * math.log(r) if r > 0 else 0.1
        arrival_time = results[-1][4] + tau

        # Если время поступления превышает T_max - остановка
        if arrival_time > T_max:
            break

        # Определение, есть ли свободные каналы
        channel1_available = (channel1_free <= arrival_time)
        channel2_available = (channel2_free <= arrival_time)

        served = 0
        if channel1_available:
            # Назначение на канал 1
            channel1_free = arrival_time + service_time
            served = 1
            served_count += 1
        elif channel2_available:
            # Назначение на канал 2
            channel2_free = arrival_time + service_time
            served = 1
            served_count += 1
        # Если оба канала заняты - отказ (served = 0)

        results.append([request_count, r, -math.log(r), tau, arrival_time,
                        channel1_free, channel2_free, served, served_count])

    return results, served_count


# 6 испытаний
print(f"\nМоделирование СМО:")
served_counts = []
all_results = []

for exp in range(N_experiments):
    results_exp, count = simulate_smo(exp)
    served_counts.append(count)
    all_results.append(results_exp)
```

```
    print(f"Испытание {exp + 1}: обслужено {count} заявок")

    # Детали испытания
    print(f"\nДетали первого испытания:")
    df_smo = pd.DataFrame(results_exp, columns=[
        'Заявка', 'r_i', '-ln(r_i)', 'τ_i', 'T_i',
        'Канал1', 'Канал2', 'Обслужена', 'Счетчик'
    ])
    print(df_smo.to_string(index=False, float_format='%.3f'))

# Расчет среднего значения
average_served = np.mean(served_counts)
print(f"\nРезультаты:")
print(f"Количество обслуженных заявок по испытаниям: {served_counts}")
print(f"Среднее количество обслуженных заявок: {average_served:.1f}")
```

Результат:

Моделирование СМО:

Испытание 1: обслужено 10 заявок

Детали испытания:

Заявка	r_i	-ln(r_i)	τ_i	T_i	Канал1	Канал2	Обслужена	Счетчик
1	NaN	NaN	NaN	0.000	1.000	0.000	1	1
2	0.549	0.600	0.150	0.150	1.000	1.150	1	2
3	0.715	0.335	0.084	0.234	1.000	1.150	0	2
4	0.603	0.506	0.127	0.360	1.000	1.150	0	2
5	0.545	0.607	0.152	0.512	1.000	1.150	0	2
6	0.424	0.859	0.215	0.727	1.000	1.150	0	2
7	0.646	0.437	0.109	0.836	1.000	1.150	0	2
8	0.438	0.826	0.207	1.043	2.043	1.150	1	3
9	0.892	0.115	0.029	1.071	2.043	1.150	0	3
10	0.964	0.037	0.009	1.081	2.043	1.150	0	3
11	0.383	0.959	0.240	1.320	2.043	2.320	1	4
12	0.792	0.234	0.058	1.379	2.043	2.320	0	4
13	0.529	0.637	0.159	1.538	2.043	2.320	0	4

14	0.568	0.566	0.141	1.679	2.043	2.320	0	4
15	0.926	0.077	0.019	1.699	2.043	2.320	0	4
16	0.071	2.645	0.661	2.360	3.360	2.320	1	5
17	0.087	2.440	0.610	2.970	3.360	3.970	1	6
18	0.020	3.901	0.975	3.945	4.945	3.970	1	7
19	0.833	0.183	0.046	3.991	4.945	4.991	1	8
20	0.778	0.251	0.063	4.054	4.945	4.991	0	8
21	0.870	0.139	0.035	4.088	4.945	4.991	0	8
22	0.979	0.022	0.005	4.094	4.945	4.991	0	8
23	0.799	0.224	0.056	4.150	4.945	4.991	0	8
24	0.461	0.773	0.193	4.343	4.945	4.991	0	8
25	0.781	0.248	0.062	4.405	4.945	4.991	0	8
26	0.118	2.135	0.534	4.939	4.945	4.991	0	8
27	0.640	0.446	0.112	5.050	6.050	4.991	1	9
28	0.143	1.942	0.486	5.536	6.050	6.536	1	10
29	0.945	0.057	0.014	5.550	6.050	6.536	0	10
30	0.522	0.650	0.163	5.713	6.050	6.536	0	10
31	0.415	0.880	0.220	5.933	6.050	6.536	0	10

Испытание 2: обслужено 10 заявок

Детали испытания:

Заявка	r_i	-ln(r_i)	τ_i	T_i	Канал1	Канал2	Обслужена	Счетчик
1	NaN	NaN	NaN	0.000	1.000	0.000	1	1
2	0.771	0.260	0.065	0.065	1.000	1.065	1	2
3	0.021	3.875	0.969	1.034	2.034	1.065	1	3

4	0.634	0.456	0.114	1.148	2.034	2.148	1	4
5	0.749	0.289	0.072	1.220	2.034	2.148	0	4
6	0.499	0.696	0.174	1.394	2.034	2.148	0	4
7	0.225	1.493	0.373	1.767	2.034	2.148	0	4
8	0.198	1.619	0.405	2.172	3.172	2.148	1	5
9	0.761	0.274	0.068	2.240	3.172	3.240	1	6
10	0.169	1.777	0.444	2.685	3.172	3.240	0	6
11	0.088	2.427	0.607	3.291	4.291	3.240	1	7
12	0.685	0.378	0.094	3.386	4.291	4.386	1	8
13	0.953	0.048	0.012	3.398	4.291	4.386	0	8
14	0.004	5.534	1.384	4.781	5.781	4.386	1	9
15	0.512	0.669	0.167	4.949	5.781	5.949	1	10
16	0.813	0.207	0.052	5.001	5.781	5.949	0	10
17	0.613	0.490	0.123	5.123	5.781	5.949	0	10
18	0.722	0.326	0.082	5.205	5.781	5.949	0	10
19	0.292	1.231	0.308	5.512	5.781	5.949	0	10
20	0.918	0.086	0.021	5.534	5.781	5.949	0	10
21	0.715	0.336	0.084	5.618	5.781	5.949	0	10
22	0.543	0.611	0.153	5.771	5.781	5.949	0	10

Испытание 3: обслужено 11 заявок

Детали испытания:

Заявка	r_i	-ln(r_i)	τ_i	T_i	Канал1	Канал2	Обслужена	Счетчик
1	NaN	NaN	NaN	0.000	1.000	0.000	1	1
2	0.588	0.531	0.133	0.133	1.000	1.133	1	2

3	0.898	0.108	0.027	0.160	1.000	1.133	0	2
4	0.892	0.115	0.029	0.188	1.000	1.133	0	2
5	0.816	0.204	0.051	0.239	1.000	1.133	0	2
6	0.036	3.327	0.832	1.071	2.071	1.133	1	3
7	0.692	0.369	0.092	1.163	2.071	2.163	1	4
8	0.379	0.971	0.243	1.406	2.071	2.163	0	4
9	0.519	0.657	0.164	1.570	2.071	2.163	0	4
10	0.658	0.419	0.105	1.675	2.071	2.163	0	4
11	0.194	1.641	0.410	2.085	3.085	2.163	1	5
12	0.272	1.301	0.325	2.410	3.085	3.410	1	6
13	0.719	0.330	0.083	2.493	3.085	3.410	0	6
14	0.783	0.245	0.061	2.554	3.085	3.410	0	6
15	0.850	0.162	0.041	2.595	3.085	3.410	0	6
16	0.775	0.255	0.064	2.658	3.085	3.410	0	6
17	0.037	3.306	0.826	3.485	4.485	3.410	1	7
18	0.117	2.148	0.537	4.022	4.485	5.022	1	8
19	0.751	0.286	0.071	4.093	4.485	5.022	0	8
20	0.239	1.430	0.358	4.451	4.485	5.022	0	8
21	0.255	1.367	0.342	4.793	5.793	5.022	1	9
22	0.858	0.154	0.038	4.831	5.793	5.022	0	9
23	0.950	0.052	0.013	4.844	5.793	5.022	0	9
24	0.562	0.577	0.144	4.988	5.793	5.022	0	9
25	0.179	1.722	0.430	5.418	5.793	6.418	1	10
26	0.770	0.261	0.065	5.484	5.793	6.418	0	10
27	0.492	0.709	0.177	5.661	5.793	6.418	0	10

28	0.631	0.460	0.115	5.776	5.793	6.418	0	10
29	0.839	0.175	0.044	5.820	6.820	6.418	1	11

Испытание 4: обслужено 11 заявок

Детали испытания:

Заявка	r_i	-ln(r_i)	τ_i	T_i	Канал1	Канал2	Обслужена	Счетчик
1	NaN	NaN	NaN	0.000	1.000	0.000	1	1
2	0.644	0.440	0.110	0.110	1.000	1.110	1	2
3	0.381	0.966	0.241	0.351	1.000	1.110	0	2
4	0.663	0.411	0.103	0.454	1.000	1.110	0	2
5	0.164	1.810	0.453	0.907	1.000	1.110	0	2
6	0.963	0.038	0.010	0.916	1.000	1.110	0	2
7	0.347	1.059	0.265	1.181	2.181	1.110	1	3
8	0.992	0.008	0.002	1.183	2.181	2.183	1	4
9	0.235	1.448	0.362	1.545	2.181	2.183	0	4
10	0.586	0.535	0.134	1.679	2.181	2.183	0	4
11	0.407	0.900	0.225	1.904	2.181	2.183	0	4
12	0.136	1.993	0.498	2.402	3.402	2.183	1	5
13	0.544	0.609	0.152	2.554	3.402	3.554	1	6
14	0.518	0.657	0.164	2.719	3.402	3.554	0	6
15	0.767	0.265	0.066	2.785	3.402	3.554	0	6
16	0.934	0.068	0.017	2.802	3.402	3.554	0	6
17	0.090	2.411	0.603	3.405	4.405	3.554	1	7
18	0.196	1.631	0.408	3.813	4.405	4.813	1	8
19	0.994	0.006	0.001	3.814	4.405	4.813	0	8

20	0.235	1.447	0.362	4.176	4.405	4.813	0	8
21	0.239	1.431	0.358	4.534	5.534	4.813	1	9
22	0.629	0.463	0.116	4.650	5.534	4.813	0	9
23	0.735	0.308	0.077	4.727	5.534	4.813	0	9
24	0.688	0.373	0.093	4.820	5.534	5.820	1	10
25	0.031	3.470	0.867	5.687	6.687	5.820	1	11
26	0.903	0.103	0.026	5.713	6.687	5.820	0	11

Испытание 5: обслужено 11 заявок

Детали испытания:

Заявка	r_i	-ln(r_i)	τ_i	T_i	Канал1	Канал2	Обслужена	Счетчик
1	NaN	NaN	NaN	0.000	1.000	0.000	1	1
2	0.408	0.897	0.224	0.224	1.000	1.224	1	2
3	0.055	2.894	0.723	0.948	1.000	1.224	0	2
4	0.789	0.238	0.059	1.007	2.007	1.224	1	3
5	0.287	1.247	0.312	1.319	2.007	2.319	1	4
6	0.450	0.798	0.199	1.518	2.007	2.319	0	4
7	0.304	1.191	0.298	1.816	2.007	2.319	0	4
8	0.526	0.642	0.160	1.977	2.007	2.319	0	4
9	0.624	0.472	0.118	2.095	3.095	2.319	1	5
10	0.777	0.253	0.063	2.158	3.095	2.319	0	5
11	0.686	0.377	0.094	2.252	3.095	2.319	0	5
12	0.981	0.019	0.005	2.257	3.095	2.319	0	5
13	0.601	0.509	0.127	2.384	3.095	3.384	1	6
14	0.814	0.206	0.051	2.435	3.095	3.384	0	6

15	0.709	0.344	0.086	2.522	3.095	3.384	0	6
16	0.028	3.592	0.898	3.420	4.420	3.384	1	7
17	0.904	0.101	0.025	3.445	4.420	4.445	1	8
18	0.450	0.799	0.200	3.644	4.420	4.445	0	8
19	0.119	2.129	0.532	4.177	4.420	4.445	0	8
20	0.835	0.180	0.045	4.222	4.420	4.445	0	8
21	0.202	1.598	0.400	4.621	5.621	4.445	1	9
22	0.174	1.748	0.437	5.058	5.621	6.058	1	10
23	0.449	0.800	0.200	5.258	5.621	6.058	0	10
24	0.670	0.401	0.100	5.359	5.621	6.058	0	10
25	0.957	0.044	0.011	5.369	5.621	6.058	0	10
26	0.833	0.182	0.046	5.415	5.621	6.058	0	10
27	0.508	0.677	0.169	5.584	5.621	6.058	0	10
28	0.630	0.462	0.115	5.700	6.700	6.058	1	11
29	0.835	0.180	0.045	5.745	6.700	6.058	0	11
30	0.447	0.804	0.201	5.946	6.700	6.058	0	11

Испытание 6: обслужено 10 заявок

Детали испытания:

Заявка	r_i	-ln(r_i)	τ_i	T_i	Канал1	Канал2	Обслужена	Счетчик
1	NaN	NaN	NaN	0.000	1.000	0.000	1	1
2	0.495	0.704	0.176	0.176	1.000	1.176	1	2
3	0.228	1.478	0.370	0.546	1.000	1.176	0	2
4	0.255	1.365	0.341	0.887	1.000	1.176	0	2
5	0.396	0.926	0.231	1.118	2.118	1.176	1	3

6	0.377	0.975	0.244	1.362	2.118	2.362	1	4
7	0.997	0.003	0.001	1.363	2.118	2.362	0	4
8	0.408	0.896	0.224	1.587	2.118	2.362	0	4
9	0.772	0.259	0.065	1.651	2.118	2.362	0	4
10	0.761	0.274	0.068	1.720	2.118	2.362	0	4
11	0.310	1.171	0.293	2.013	2.118	2.362	0	4
12	0.347	1.060	0.265	2.277	3.277	2.362	1	5
13	0.352	1.045	0.261	2.539	3.277	3.539	1	6
14	0.145	1.928	0.482	3.021	3.277	3.539	0	6
15	0.973	0.028	0.007	3.028	3.277	3.539	0	6
16	0.909	0.095	0.024	3.051	3.277	3.539	0	6
17	0.560	0.580	0.145	3.196	3.277	3.539	0	6
18	0.314	1.160	0.290	3.486	4.486	3.539	1	7
19	0.888	0.119	0.030	3.516	4.486	3.539	0	7
20	0.675	0.394	0.098	3.614	4.486	4.614	1	8
21	0.391	0.939	0.235	3.849	4.486	4.614	0	8
22	0.507	0.679	0.170	4.019	4.486	4.614	0	8
23	0.524	0.646	0.162	4.180	4.486	4.614	0	8
24	0.928	0.075	0.019	4.199	4.486	4.614	0	8
25	0.571	0.560	0.140	4.339	4.486	4.614	0	8
26	0.668	0.403	0.101	4.440	4.486	4.614	0	8
27	0.052	2.952	0.738	5.177	6.177	4.614	1	9
28	0.327	1.118	0.279	5.457	6.177	6.457	1	10

Шаг 3. Оценка мат. ожидания числа обслуженных заявок

Количество обслуженных заявок по испытаниям: [10, 10, 11, 11, 11, 10]

Мат. ожидание числа обслуженных заявок:

$$a = \bar{x} = \frac{10 + 10 + 11 + 11 + 11 + 10}{6} = \frac{63}{6} = 10.5$$