



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет имени
Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №4

Название Обслуживающий аппарат

Дисциплина Моделирование

Студент Прохорова Л. А.

Группа ИУ7-73Б

Оценка (баллы) _____

Преподаватель Рудаков И. В.

Москва — 2022 г.

1 Задание

Промоделировать систему, состоящую из генератора, памяти и обслуживающего аппарата. Генератор подает сообщения, распределенные по равномерному закону, они приходят в память и выбираются на обработку по закону из ЛР1 (по закону Пуассона). Количество заявок конечно и задано. Предусмотреть случай, когда обработанная заявка возвращается обратно в очередь. Определить оптимальную длину очереди, при которой не будет потерянных сообщений. Реализовать двумя способами: используя пошаговый и событийный подходы.

2 Теоретическая часть

2.1 Распределения

2.1.1 Равномерное распределение

Случайная величина X имеет непрерывное равномерное распределение $X \sim R(a, b)$ на отрезке $[a, b]$, где $a, b \in R$, если ее функция плотности $f(x)$ имеет следующий вид:

$$f(x) = \begin{cases} \frac{1}{b-a}, & x \in [a, b] \\ 0, & \text{иначе.} \end{cases} \quad (1)$$

Функция распределения $F(x) = \int_{-\inf}^x f(t)dt$ принимает вид:

$$F(x) = \begin{cases} \frac{x-a}{b-a}, & x \in [a, b] \\ 1, & x > b \\ 0, & x < a \end{cases} \quad (2)$$

2.1.2 Распределение Пуассона

Распределение Пуассона — распределение дискретного типа случайной величины, представляющей собой число событий, произошедших за фиксированное время, при условии, что данные события происходят с некоторой фиксированной средней интенсивностью и независимо друг от друга.

Распределение Пуассона имеет один параметр, обозначаемый символом λ — среднее количество успешных испытаний в заданной области возможных исходов. Дисперсия распределения Пуассона также равна λ , а его стандартное отклонение равно $\sqrt{\lambda}$. Количество успешных испытаний X пуассоновской случайной величины изменяется от 0 до бесконечности. Случайная величина Y имеет распределение Пуассона с математическим ожиданием λ записывается: $Y \sim P(\lambda)$.

Для дискретной случайной величины не существует функции плотности распределения вероятностей.

Функция вероятности при $\lambda > 0, k \geq 0$:

$$P(x = k) = \frac{\lambda^k}{k!} e^{-\lambda} \quad (3)$$

где

k — количество событий,

λ — математическое ожидание случайной величины (среднее количество событий за фиксированный промежуток времени),

$e=2,718281828\dots$ - основание натурального логарифма.

Функция распределения при $\lambda > 0, k \geq 0$:

$$F(x) = \sum_{k=0}^N \frac{\lambda^k}{k!} e^{-\lambda} \quad (4)$$

2.2 Подходы к моделированию работы системы

2.2.1 Пошаговый подход

Заключается в последовательном анализе состояний всех блоков системы в момент $t + \Delta t$ по заданному состоянию в момент t . При этом новое состояние блоков определяется в соответствии с их алгоритмическим описанием с учетом действующих случайных факторов. В результате этого анализа принимается решение о том, какие системные события должны имитироваться на данный момент времени. Основным недостатком являются значительные затраты машинных ресурсов, а при недостаточном малых Δt появляется опасность пропуска события.

2.2.2 Событийный принцип

Состояние отдельных устройств изменяется в дискретные моменты времени, совпадающие с моментами поступления сообщения, окончания решения задачи, возникновения аварийных сигналов и т. д. При использовании событийного принципа состояния всех блоков системы анализируются лишь в момент появления какого-либо события. Момент наступления следующего события определяется минимальным значением из списка будущих событий, представляющий собой совокупность моментов ближайшего изменения состояния каждого из блоков. Момент наступления следующего события определяется минимальным значением из списка событий.

3 Результаты работы программы

На рисунках представлены результаты программы при $\lambda = 1, 8$ и при проценте возвращаемых в очередь задач не более чем 0, 5, 25, 50, 75, 95, 99, 100. Параметры a и b для равномерного распределения будут постоянны: $a = 0$, $b = 8$.

3.1 $\lambda = 1$

```
Введите(через пробел) значения параметров а и b для равномерного распределения: 0 0
Введите λ для распределения Пуассона: 1
Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 0
```

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	0	5	39786.5
Пошаговый	10000	0	6	40214.43

Рисунок 1 – Результат работы программы при отсутствии возвращения обработанных заявок в очередь

```
Введите(через пробел) значения параметров а и b для равномерного распределения: 0 0
Введите λ для распределения Пуассона: 1
Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 5
```

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	475	5	38011.946
Пошаговый	10000	497	5	37567.01

Рисунок 2 – Результат работы программы при возвращении в очередь не более 5% обработанных заявок

```
Введите(через пробел) значения параметров а и b для равномерного распределения: 0 0
Введите λ для распределения Пуассона: 1
Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 25
```

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	2484	6	29953.421
Пошаговый	10000	2500	6	29910.06

Рисунок 3 – Результат работы программы при возвращении в очередь не более 25% обработанных заявок

```
Введите(через пробел) значения параметров а и b для равномерного распределения: 0 0
Введите λ для распределения Пуассона: 1
Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 50
```

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	5000	9	19914.861
Пошаговый	10000	4968	14	20189.57

Рисунок 4 – Результат работы программы при возвращении в очередь не более 50% обработанных заявок

Введите(через пробел) значения параметров а и b для равномерного распределения: 0 0

Введите λ для распределения Пуассона: 1

Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 75

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	7500	95	10152.194
Пошаговый	10000	7495	58	10080.83

Рисунок 5 – Результат работы программы при возвращении в очередь не более 75% обработанных заявок

Введите(через пробел) значения параметров а и b для равномерного распределения: 0 0

Введите λ для распределения Пуассона: 1

Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 95

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	9500	2007	10026.991
Пошаговый	10000	9489	1974	9961.46

Рисунок 6 – Результат работы программы при возвращении в очередь не более 95% обработанных заявок

Введите(через пробел) значения параметров а и b для равномерного распределения: 0 0

Введите λ для распределения Пуассона: 1

Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 99

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	9899	2393	10090.242
Пошаговый	10000	9900	2364	9904.84

Рисунок 7 – Результат работы программы при возвращении в очередь не более 99% обработанных заявок

Введите(через пробел) значения параметров а и b для равномерного распределения: 0 0

Введите λ для распределения Пуассона: 1

Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 100

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	10000	2548	10012.011
Пошаговый	10000	10000	2490	9929.04

Рисунок 8 – Результат работы программы при возвращении в очередь всех обработанных заявок

3.2 $\lambda = 8$

Введите(через пробел) значения параметров а и b для равномерного распределения: 0 0
Введите λ для распределения Пуассона: 8
Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 0

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	0	10162	80124.989
Пошаговый	10000	0	9952	79723.15

Рисунок 9 – Результат работы программы при отсутствии возвращения обработанных заявок в очередь

Введите(через пробел) значения параметров а и b для равномерного распределения: 0 0
Введите λ для распределения Пуассона: 8
Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 5

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	500	10477	79871.293
Пошаговый	10000	500	10399	79924.94

Рисунок 10 – Результат работы программы при возвращении в очередь не более 5% обработанных заявок

Введите(через пробел) значения параметров а и b для равномерного распределения: 0 0
Введите λ для распределения Пуассона: 8
Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 25

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	2500	12360	79339.304
Пошаговый	10000	2465	12506	80108.62

Рисунок 11 – Результат работы программы при возвращении в очередь не более 25% обработанных заявок

Введите(через пробел) значения параметров а и b для равномерного распределения: 0 0
Введите λ для распределения Пуассона: 8
Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 50

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	5000	14937	79951.974
Пошаговый	10000	5000	15033	79716.99

Рисунок 12 – Результат работы программы при возвращении в очередь не более 50% обработанных заявок

Введите(через пробел) значения параметров a и b для равномерного распределения: 0 0

Введите λ для распределения Пуассона: 0

Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 75

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	7500	17462	79890.339
Пошаговый	10000	7500	17545	80070.78

Рисунок 13 – Результат работы программы при возвращении в очередь не более 75% обработанных заявок

Введите(через пробел) значения параметров a и b для равномерного распределения: 0 0

Введите λ для распределения Пуассона: 0

Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 95

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	9500	19612	80241.058
Пошаговый	10000	9496	19507	80137.86

Рисунок 14 – Результат работы программы при возвращении в очередь не более 95% обработанных заявок

Введите(через пробел) значения параметров a и b для равномерного распределения: 0 0

Введите λ для распределения Пуассона: 0

Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 99

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	9884	19829	80429.795
Пошаговый	10000	9899	19880	80192.2

Рисунок 15 – Результат работы программы при возвращении в очередь не более 99% обработанных заявок

Введите(через пробел) значения параметров a и b для равномерного распределения: 0 0

Введите λ для распределения Пуассона: 0

Введите максимальное значение процента задач, которые возвращаются в очередь после обработки (в %): 100

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	10000	20073	80049.135
Пошаговый	10000	10000	19939	79712.38

Рисунок 16 – Результат работы программы при возвращении в очередь всех обработанных заявок

4 Код программы

Программа разработана в интегрированной среде разработки для языка программирования Python - PyCharm. В листинге 1 приведена реализация лабораторной работы.

```
0 from prettytable import PrettyTable
1 from scipy.stats import poisson, uniform
2 from numpy import random as numpy_random
3 import sys
4 COUNT = 10000
5 class Generator:
6     def __init__(self, generator):
7         self.generator = generator
8         self.receivers = set()
9
10    def add_receiver(self, receiver):
11        self.receivers.add(receiver)
12
13    def remove_receiver(self, receiver):
14        try:
15            self.receivers.remove(receiver)
16        except KeyError:
17            print("Ошибка!")
18            sys.exit(0)
19
20    def get_time(self):
21        return self.generator.generate()
22
23    def request(self):
24        for receiver in self.receivers:
25            receiver.receive_request()
26
27 class Processor(Generator):
28     def __init__(self, generator, reenter_prob=0):
29         super().__init__(generator)
30         self.queue_size = 0
31         self.max_queue_size = 0
```

```

32     self.processed_requests = 0
33     self.reentered_requests = 0
34     self.reenter_prob = reenter_prob
35
36     # Обработка запроса
37     def process(self):
38         if self.queue_size > 0:
39             self.processed_requests += 1
40             self.queue_size -= 1
41             self.request()
42             # Возвращение запроса в очередь при соблюдении условий вероятности
43             if numpy_random.random_sample() <= self.reenter_prob and self.
reentered_requests < self.reenter_prob * COUNT:
44                 self.reentered_requests += 1
45                 self.receive_request()
46
47     # Добавление запроса в очередь
48     def receive_request(self):
49         self.queue_size += 1
50         if self.queue_size > self.max_queue_size:
51             self.max_queue_size = self.queue_size
52
53     class UniformDistribution:
54         def __init__(self, a: float, b: float):
55             self.a = a
56             self.b = b
57             self.scale = self.b - self.a
58
59         def generate(self):
60             return uniform.rvs(loc=self.a, scale=self.scale, size=1)[0]
61
62
63     class PoissonDistribution:
64         def __init__(self, lambda):
65             self.lambda = lambda
66
67         def generate(self):
68             return poisson.rvs(self.lambda, size=1)[0]
69
70

```

```

71 class Model:
72     def __init__(self, uniform_a, uniform_b, lambda, reenter_prop):
73         self.generator = Generator(UniformDistribution(uniform_a,
74 uniform_b))
75         self.processor = Processor(PoissonDistribution(lambda),
76 reenter_prop)
77         self.generator.add_receiver(self.processor)
78
79     def event_based_system(self, request_count):
80         generator = self.generator
81         processor = self.processor
82
83         gen_period = generator.get_time()
84         proc_period = gen_period + processor.get_time()
85
86         while processor.processed_requests < request_count:
87             if gen_period <= proc_period:
88                 generator.request()
89                 gen_period += generator.get_time()
90             if gen_period >= proc_period:
91                 processor.process()
92
93             if processor.queue_size > 0:
94                 proc_period += processor.get_time()
95             else:
96                 proc_period = gen_period + processor.get_time()
97
98         return (processor.processed_requests, processor.reentered_requests
99 ,
100             processor.max_queue_size, round(proc_period, 3))
101
102     def time_based_modelling(self, request_count, dt):
103         generator = self.generator
104         processor = self.processor
105
106         gen_period = generator.get_time()
107         proc_period = gen_period
108         current_time = 0
109         while processor.processed_requests < request_count:
110             if gen_period <= current_time:

```

```

108         generator.request()
109         gen_period += generator.get_time()
110         if current_time >= proc_period:
111             processor.process()
112             if processor.queue_size > 0:
113                 proc_period += processor.get_time()
114             else:
115                 proc_period = gen_period + processor.get_time()
116
117         current_time += dt
118
119         return (processor.processed_requests, processor.reentered_requests
120 ,
121
122             processor.max_queue_size, round(current_time, 3))
123
124 if __name__ == '__main__':
125
126     a, b = map(int, input("Введите через ( пробел ) значения параметров a и b для
127 равномерного распределения: ").split())
128
129     if a >= b:
130         print("Параметр a должен быть меньше параметра b")
131         sys.exit(0)
132
133     lmbda = int(input("Введите лямбда для распределения Пуассона: "))
134
135     if lmbda < 0:
136         print("лямбда должна быть больше 0")
137         sys.exit(0)
138
139     repeat_probability = float(input("Введите максимальное значение процента задач,
140 которые возвращаются в очередь после обработки в ( % ): "))
141     if repeat_probability >= 0 and repeat_probability <= 100:
142         repeat_probability = repeat_probability / 100
143     else:
144         print("Неверный ввод")
145         sys.exit(0)
146
147     total_tasks = COUNT
148     step = 0.01

```

```
145
146     model = Model(a, b, lambda, repeat_probality)
147     result1 = model.event_based_system(total_tasks)
148     model2 = Model(a, b, lambda, repeat_probality)
149     result2 = model2.time_based_modelling(total_tasks, step)
150
151     table = PrettyTable()
152     table.add_column("Метод", ["Событийный", "Пошаговый"])
153     table.add_column("Колво - обработанных запросов", [result1[0], result2[0]])
154     table.add_column("Колво - возвращенных запросов", [result1[1], result2[1]])
155     table.add_column("Максимальный размер очереди", [result1[2], result2[2]])
156     table.add_column("Время работы ", [result1[3], result2[3]])
157     print(table)
```