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 Информационные технологии и системы управления
КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ ПО КУРСУ «ПЛАНИРОВАНИЕ ЭКСПЕРИМЕНТОВ»

Лабораторная работа №1

Студент ИУ7-83Б — Зыкин Д.А.
(Группа) (И.О.Фамилия)

Преподаватель _____ Куров А.В,
(Подпись, дата) (И.О.Фамилия)

2020 г.

Цель работы

Разработать имитационную модель функционирования одноканальной разомкнутой СМО с одним типом заявок. Буфер бесконечный. Для законов распределения с двумя параметрами предусмотреть возможность ввода только самих параметров. Для законов распределения с одним параметром:

- предусмотреть возможность ввода интенсивности;
- предусмотреть возможность ввода параметра распределения.

Рассчитать интенсивность поступления и обработки заявок и сравнить с полученными результатами.

Условия работы

Законы распределения: Рэлей для генератора, Нормальный для обслуживающего аппарата.

Теоретическая часть

В данной работе проводится моделирование одноканальной разомкнутой СМО. По условию генерация заявок происходит по закону Рэлея, а обслуживание по нормальному закону распределения.

Закон распределения Рэлея зависит только от одного параметра, поэтому для него также необходимо предусмотреть ввод интенсивности. Его функция плотности:

$$f(x;\sigma) = \frac{x}{\sigma^2} e^{(-\frac{x^2}{2\sigma^2})}, x \geq 0, \sigma > 0 \quad (1)$$

Нормальный закон распределения зависит от двух параметров, следовательно для него предусмотрен ввод только параметров. Его функция плотности:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)}{2\sigma^2}} \quad (2)$$

Для теоретической оценки интенсивности используются генератора используется формула:

$$I = \frac{1}{\sqrt{\frac{\pi}{2}}\sigma} \tag{3}$$

Для теоретической оценки интенсивности обработчика заявок используется формула:

$$I = \frac{1}{m} \tag{4}$$

Реальная интенсивность оценивается по завершении моделирования и рассчитывается как количество заявок, обработанных элементом, поделенное на время моделирования.

Реализованные классы и методы

Для проведения моделирования были созданы следующие классы:

- Generator;
- ModellingController;
- Processor;
- Queue;
- TimeElement.

Класс TimeElement

Данный класс является базовым классом для всех элементов, чья работа основывается на генерируемых задержках. Он хранит в себе общее время работы элемента, а также содержит интерфейс для генерации задержки на основе передаваемой в него функции.

```
1  class TimeElement:
2      def __init__(self, rand_func, param1, param2):
3          self.rand_func = rand_func
4          self.param1 = param1
5          self.param2 = param2
6          self.__total_time = 0
7
8      @property
9      def total_time(self):
10         return self.__total_time
11
12     @total_time.setter
13     def total_time(self, value):
14         self.__total_time = value
15
```

```
    def generate_delay(self):
        self.total_time += self.rand_func(
            self.param1, self.param2)
        return self.total_time
```

Класс ModellingController

Данный класс предназначен является аналогом блока управления. Он отвечает за отслеживание происходящих в системе событий, увеличивает счетчик текущего времени и подсчитывает результаты моделирования.

```
class ModellingController:
    @staticmethod
    def get_min_time_el_index(lst):
        min_time = lst[0].total_time
        min_index = 0
        for i, elem in enumerate(lst):
            if elem.total_time < min_time:
                if isinstance(elem, Processor) and
elem.queue.counter <= 0:
                    continue
                min_time = elem.total_time
                min_index = i

        return min_time, min_index

    @property
    def modelling_time(self):
        return self.__modelling_time

    @modelling_time.setter
    def modelling_time(self, value):
        self.__modelling_time = value

    def __init__(self, gen_rand_func, gen_param1, gen_param2,
                 proc_rand_func, proc_param1, proc_param2,
modelling_time):
        self.__modelling_time = modelling_time
        self.queue = Queue()
        self.gen = Generator(gen_rand_func, gen_param1,
gen_param2, self.queue)
        self.proc = Processor(proc_rand_func, proc_param1,
proc_param2, self.queue)

    def start_modelling(self):
```

```
        lst = [self.gen, self.proc]
        current_time = 0

        while current_time < self.modelling_time:
            current_time, index =
self.get_min_time_el_index(lst)
            if isinstance(lst[index], Generator):
                lst[index].generate_element()
            if isinstance(lst[index], Processor):
                lst[index].process_next()

        gen_intense = self.gen.generated /
self.modelling_time
        oa_intense = self.proc.processed /
self.modelling_time

        return gen_intense, oa_intense
```

Класс Queue

Данный класс реализует очередь без потерь. Позволяет добавлять/забирать заявку, а также содержит текущую длину.

```
class Queue:
    @property
    def counter(self):
        return self.__counter

    @counter.setter
    def counter(self, value):
        self.__counter = value

    def __init__(self):
        self.counter = 0

    def enqueue(self):
        self.counter += 1

    def dequeue(self):
        if self.counter <= 0:
            return False
        else:
            self.counter -= 1
            return True
```

Класс Processor

Данный класс предназначен для моделирования обработчика заявок и содержит методы для начала и окончания обработки заявки, которые вызываются в классе ModellingController.

```
class Processor(TimeElement):
    def __init__(self, rand_func, param1, param2, queue):
        super().__init__(rand_func, param1, param2)
        self.queue = queue
        self.processing = False
        self.processed = 0

    def start_processing(self):
        if self.processing:
            self.queue.enqueue()
        else:
            self.generate_delay()
            self.processing = True

    def process_next(self):
        self.processed += 1
        self.processing = False
        if self.queue.dequeue():
            self.processing = True
            self.generate_delay()
```

Класс Generator

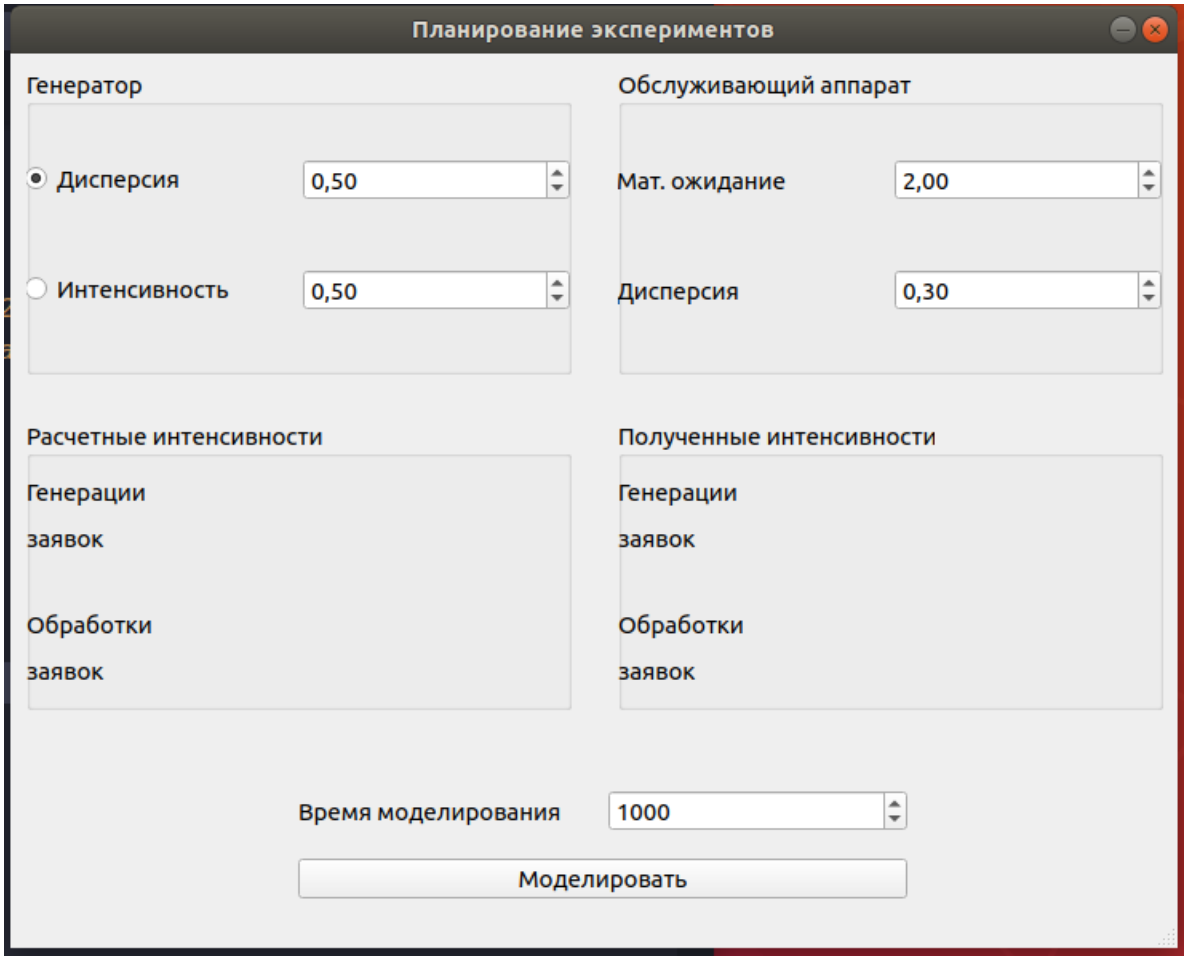
Предназначен для создания новых заявок в системе.

```
class Generator(TimeElement):
    def __init__(self, rand_func, param1, param2, queue):
        super().__init__(rand_func, param1, param2)
        self.generated = 0
        self.queue = queue

    def generate_element(self):
        self.generated += 1
        self.queue.enqueue()
        return self.generate_delay()
```

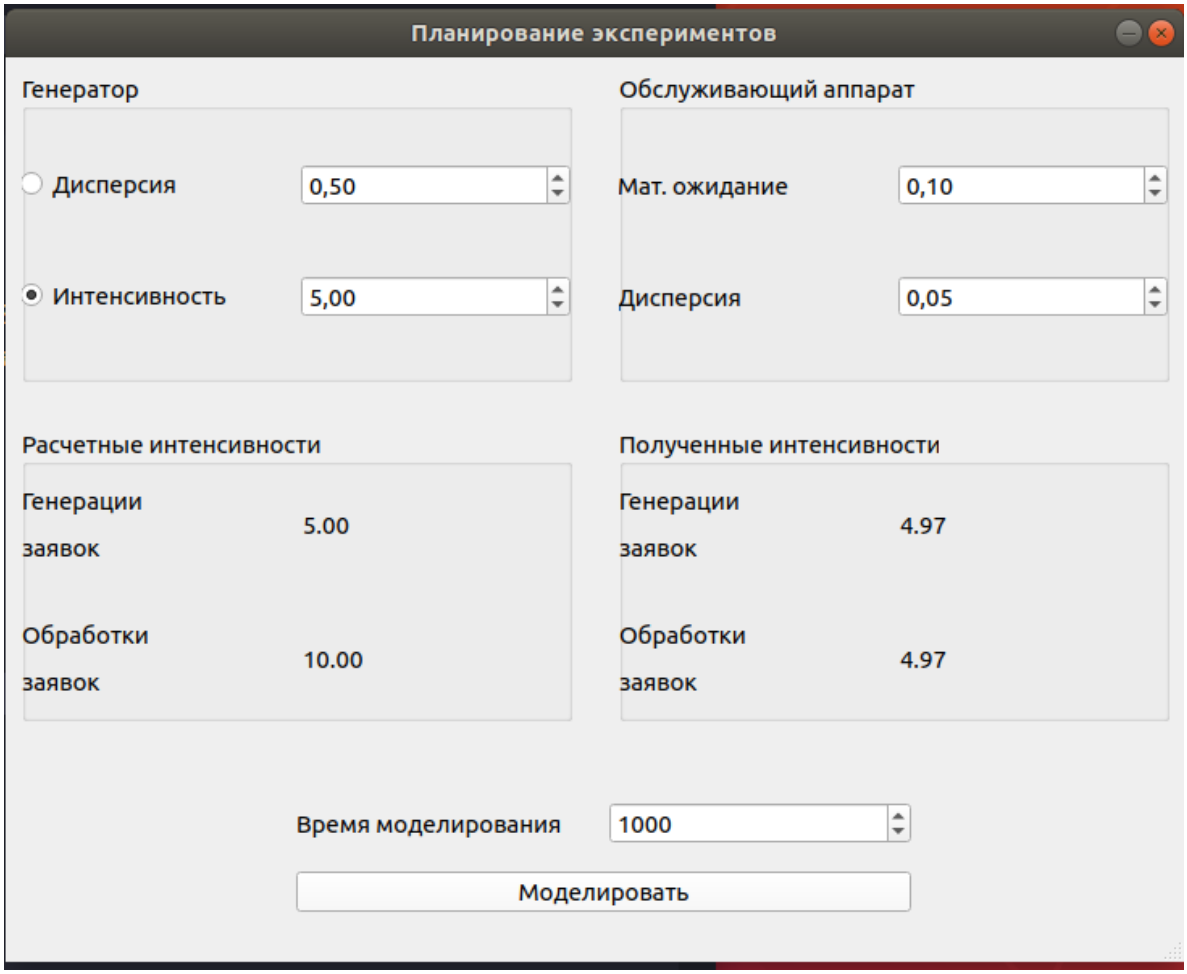
Интерфейс программы

Лабораторная работа была реализована на языке Python v3.8 с использованием библиотеки PyQt5 для создания графического интерфейса.



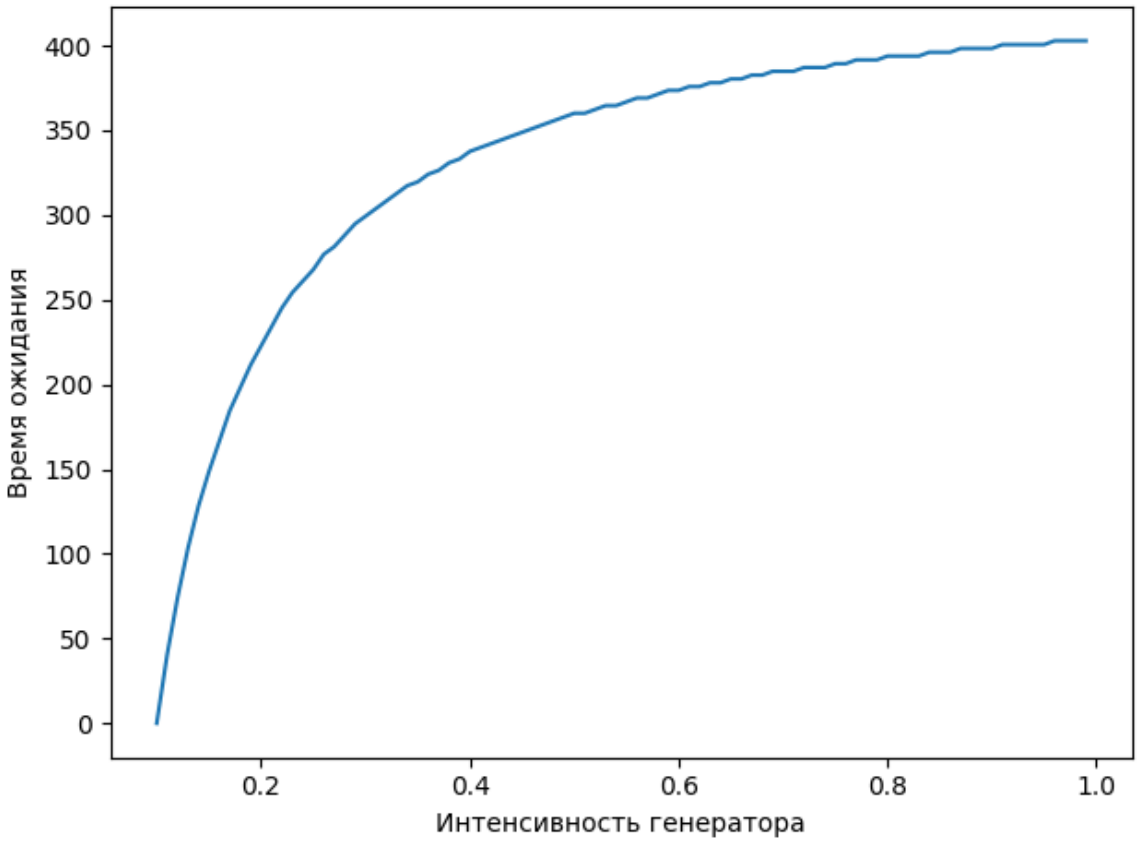
Пользователь может задать параметры законов распределения для генератора и обработчика заявок. В случае генератора предусмотрен выбор между заданием дисперсии и интенсивности.

После задания параметров законов распределения также задается время моделирования, после чего по нажатии кнопки *Моделировать* будут выведены расчетные и полученные интенсивности.

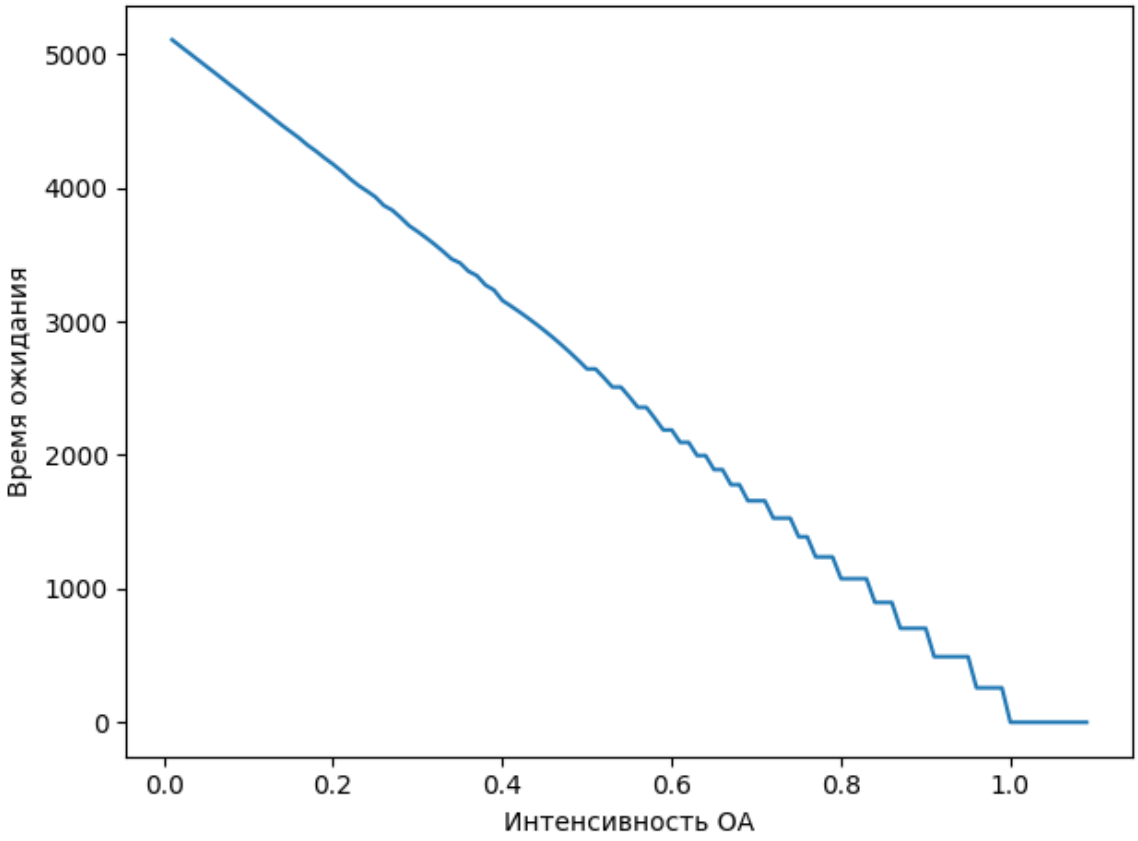


После проведения моделирования получилось так, что интенсивность генерации заявок превышает теоретическую интенсивность обработчика, что приводит к тому, что реальная интенсивность обработчика будет равна интенсивности генератора.

Графики зависимости времени ожидания от интенсивности



При росте интенсивности генератора заявок и неизменной интенсивности ОА время ожидания заявок в очереди будет расти, после того как интенсивность генератора превысит интенсивность ОА. Для данного графика интенсивность ОА равна 0.1. До этого момента время ожидания заявок в очереди будет равно 0.



При росте интенсивности ОА и неизменной интенсивности генератора заявок время ожидания заявок в очереди будет снижаться, пока интенсивности не сравняются. После этого время ожидания будет равно 0.