



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 ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение эвм и информационные технологии»

ОТЧЕТ

По лабораторной работе №4

«Моделирование простейшего прибора обслуживания»

По курсу «Моделирование»

Студент

Группа

Преподаватель

Т. А. Казаева

ИУ7-76Б

И. В. Рудаков

2022 г.

1. ЗАДАНИЕ

Смоделировать систему, состоящую из генератора, памяти, и обслуживающего аппарата.

Генератор подает сообщения, распределенные по равномерному закону, они приходят в память и выбираются на обработку по нормальному (Гауссову) закону.

Количество заявок конечно и задано. Предусмотреть случай, когда обработанная заявка возвращается обратно в очередь. Необходимо определить оптимальную длину очереди, при которой не будет потерянных сообщений. Реализовать двумя способами: используя пошаговый и событийный подходы.

2. МАТЕМАТИЧЕСКАЯ ФОРМАЛИЗАЦИЯ

2.1 ЗАКОНЫ РАСПРЕДЕЛЕНИЯ

2.1.1 НЕПРЕРЫВНОЕ РАВНОМЕРНОЕ РАСПРЕДЕЛЕНИЕ

Говорят, что случайная величина X имеет непрерывное равномерное распределение на отрезке $[a, b]$, если её функция плотности имеет вид (2.1):

$$f_X(x) = \begin{cases} \frac{1}{b-a}, & x \in [a, b] \\ 0, & x \notin [a, b] \end{cases}. \quad (2.1)$$

Обозначается: $X \sim U[a, b]$.

Функция распределения равномерной случайной величины $X \sim U[a, b]$ (2.2):

$$F_X(x) \equiv \mathbb{P}(X \leq x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x < b \\ 1, & x \geq b \end{cases}. \quad (2.2)$$

2.1.2 НОРМАЛЬНОЕ РАСПРЕДЕЛЕНИЕ

Говорят, что случайная величина X имеет нормальное распределение с параметрами μ и σ^2 ($\sigma^2 > 0$), если её функция плотности имеет вид (2.3):

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, x \in \mathbb{R} \quad (2.3)$$

Обозначается: $X \sim N(\mu, \sigma^2)$.

Функция распределения нормальной случайной величины $X \sim N(\mu, \sigma^2)$:

$$\frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{x-\mu}{\sigma\sqrt{2}} \right) \right], \quad (2.4)$$

где

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \quad (2.5)$$

2.2 УПРАВЛЯЮЩАЯ ПРОГРАММА ИНДУКЦИОННОЙ МОДЕЛИ

2.2.1 ПОШАГОВЫЙ МЕТОД

Заключается в последовательном анализе состояний всех блоков системы в момент $t + \Delta t$. Новое состояние определяется в соответствии с их алгоритмическим описанием с учетом действия случайных факторов. В результате этого анализа принимается решение о том, какие системные события должны имитироваться на данный момент времени.

Основной недостаток принципа Δt : значительные затраты вычислительных ресурсов при моделировании системы. При недостаточно малом Δt появляется опасность пропуска отдельных событий в системе, исключая возможность получения правильных результатов при моделировании.

2.2.2 СОБЫТИЙНЫЙ МЕТОД

Состояния отдельных устройств изменяются в дискретные моменты времени. При использовании событийного принципа, состояния всех блоков системы анализируются лишь в момент возникновения какого либо события. Момент наступления следующего события определяется минимальным значением из списка будущих событий, представляющего собой совокупность моментов ближайшего изменения состояния каждого из блоков системы.

3. РЕЗУЛЬТАТ

3.1 ПРОГРАММНЫЙ ИНТЕРФЕЙС

Пользователю предоставляется следующие способы ввода значений в программу:

- с помощью аргументов командной строки (рис. 3.1), документация к которым может быть получена с помощью специальной команды (рис. 3.3)
- с помощью интерфейса непосредственно – приглашения на ввод поступают последовательно после нажатия клавиши «Enter» с соответствующими комментариями (рис. 3.2).

```
honeycarbs@honeycarbs ~/B/b/l/code> python3 main.py --uniraw "10 19" --gaussraw "0 2.0" --jobs 1000 --prob 30 --delta 0.1
TIME-KEEPING SIMULATION
Maximum buffer capacity without losses: 1
Number of processed jobs: 1452
Number of jobs re-entering buffer: 452
EVENT SIMULATION
Maximum buffer capacity without losses: 1
Number of processed jobs: 1426
Number of jobs re-entering buffer: 426
honeycarbs@honeycarbs ~/B/b/l/code> █
```

Рис. 3.1: Ввод данных с помощью аргументов командной строки

```
Arguments for uniform distribution (separate by space): 10 19
Mean and variance for Gaussian distribution (separate by space): 0 2.0
Amount of jobs in queue: 1000
Probability of job returning in queue (percentage): 30
Time delta for event simulation: 0.1
TIME-KEEPING SIMULATION
Maximum buffer capacity without losses: 1
Number of processed jobs: 1448
Number of jobs re-entering buffer: 448
EVENT SIMULATION
Maximum buffer capacity without losses: 1
Number of processed jobs: 1442
Number of jobs re-entering buffer: 442
honeycarbs@honeycarbs ~/B/b/l/code> █
```

Рис. 3.2: Консольный интерфейс приложения

```
honeycarbs@honeycarbs ~/B/b/L/code> python3 main.py --help
Usage: main.py [OPTIONS]

Options:
  --uniraw TEXT      Arguments for uniform distribution separated by space.
  --gaussraw TEXT    Arguments for Gaussian distribution separated by space.
  --jobs INTEGER     Amount of jobs in queue.
  --prob FLOAT       Probability of job returning in queue (in percentage).
  --delta FLOAT      Time delta for event simulation.
  --help            Show this message and exit.
honeycarbs@honeycarbs ~/B/b/L/code> █
```

Рис. 3.3: Информация об аргументах командной строки

Результат работы программы представлен на рисунках 3.1 и 3.2 – результаты событийного моделирования выделены синим цветом, а пошагового – красным.

3.2 РЕЗУЛЬТАТЫ РАБОТЫ

В таблице приведены результаты работы программы. Параметры: $\mathcal{N}(0, 1.0)$, $\mathcal{U}[1, 10]$, $\Delta = 0.1$.

Таблица 3.1: Результаты работы программы

<i>модель</i>	<i>процент повторений</i>	<i>максимальное количество заявок в буфере</i>	<i>обработано заявок</i>	<i>количество заявок, отправленных назад в буфер</i>
Событийная	0	1	1000	0
Пошаговая		1	1000	0
Событийная	10	1	1113	113
Пошаговая		1	1103	103
Событийная	40	2	1673	673
Пошаговая		1	1713	713
Событийная	80	2	4879	3879
Пошаговая		2	5086	4086

4. ПРОГРАММНЫЙ КОД

Для реализации программы был выбран язык Python.
На листинге 4.1 представлена реализация пошагового метода.

Листинг 4.1: Реализация пошагового метода

```
1 def TimekeepingSimulation(distribution, jobProcessor,
    jobsAmount=0, returnPercentage=0, step=0.001):
2     processedJobs = 0
3     currentTime = step
4     generationTime = distribution.GetRandomValue()
5     processTime = 0
6     currentQueueLen = maxQueueLen = 0
7     generatedJobs = 0
8     returnedJobs = 0
9
10    while processedJobs < jobsAmount + returnedJobs:
11
12        if currentTime > generationTime and generatedJobs <=
            jobsAmount:
13            currentQueueLen += 1
14            generatedJobs += 1
15            if currentQueueLen > maxQueueLen:
16                maxQueueLen = currentQueueLen
17                generationTime += distribution.GetRandomValue()
18        if currentTime > processTime:
19            if currentQueueLen > 0:
20                processedJobs += 1
21
22            if random.randint(1, 100) <= returnPercentage:
23                returnedJobs += 1
24                currentQueueLen += 1
25
26            currentQueueLen -= 1
27            processTime += jobProcessor.GetRandomValue()
28        currentTime += step
29    return maxQueueLen, processedJobs, returnedJobs
```

На листинге 4.2 представлена реализация событийного метода.

Листинг 4.2: Реализация событийного метода

```
1 def EventSimulation(distribution, jobProcessor, jobsAmount=0,
    returnPercentage=0):
2     processedJobs = 0
3     currentQueueLen = 0
4     maxQueueLen = 0
5
6     jobs = [[distribution.GetRandomValue(), 'g']]
7
8     free, isProcessed = True, False
9
10    generatedJobs = 0
11    returnedJobs = 0
12
13    while processedJobs < jobsAmount + returnedJobs:
14
15        job = jobs.pop(0)
16
17        if job[1] == 'g' and generatedJobs <= jobsAmount:
18
19            currentQueueLen += 1
20            generatedJobs += 1
21
22            if currentQueueLen > maxQueueLen:
23                maxQueueLen = currentQueueLen
24
25            enqueueJob(jobs, [job[0] +
26                distribution.GetRandomValue(), 'g'])
27
28            if free:
29                isProcessed = True
30
31        elif job[1] == 'p':
32
33            processedJobs += 1
```



```

34         if random.randint(1, 100) <= returnPercentage:
35             returnedJobs += 1
36             currentQueueLen += 1
37
38         isProcessed = True
39
40     if isProcessed:
41
42         if currentQueueLen > 0:
43             currentQueueLen -= 1
44             t = jobProcessor.GetRandomValue()
45             enqueueJob(jobs, [job[0] + t, 'p'])
46             free = False
47         else:
48             free = True
49         isProcessed = False
50
51     return maxQueueLen, processedJobs, returnedJobs

```

На листинге 4.3 представлена функция помещения заявки в буфер.

Листинг 4.3: Вспомогательная функция помещения заявки в буфер

```

1 def enqueueJob(jobs, job):
2     i = 0
3     while i < len(jobs) and jobs[i][0] < job[0]:
4         i += 1
5     if 0 < i < len(jobs):
6         jobs.insert(i - 1, job)
7     else:
8         jobs.insert(i, job)

```

На листинге 4.4 представлены функции распределений: равномерного и нормального.

Листинг 4.4: Функции распределений

```
1 class UniformDistributon:
2     def __init__(self, lo: float, hi: float):
3         self.lo = lo
4         self.hi = hi
5
6 def GetRandomValue(self):
7     return self.lo + (self.hi - self.lo) * random.random()
8
9     class GaussianDistributon:
10     def __init__(self, mu: float, sigma: float):
11         self.mu = mu
12         self.sigma = sigma
13
14     def GetRandomValue(self):
15         return normal(self.mu, self.sigma)
```
