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
Обслуживающий аппарат

Студент Соколов Ефим
Группа ИУ7-73Б
Дисциплина Моделирование

Преподаватель: ________ Рудаков И.В.
подпись, дата Фамилия, И.О.

Оценка ________________________

Москва — 2022 г.

Задание

Необходимо промоделировать систему, состоящую из генератора заявок и обслуживающего аппарата. Генератор подает сообщения, распределенные по равномерному закону, они приходят в память и выбираются на обработку по нормальному закону (как в лабораторной работе №1). Предусмотреть случай, когда обработанная заявка возвращается обратно в очередь. Необходимо определить оптимальную длину очереди, при которой не будет потерянных сообщений.

Реализовать двумя методами: используя пошаговый и событийный подходы.

Теоретическая часть

Распределения

Равномерное распределение

Равномерное распределение непрерывной случайной величины – это распределение, в котором значения случайной величины с двух сторон ограничены и в границах интервала имеют одинаковую вероятность. Плотность вероятности в данном интервале постоянна. Равномерное распределение обозначают $X \sim R(a, b)$, где $a, b \in \mathbb{R}$.

Функция плотности $f_X(x)$ имеет вид:

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

Проинтегрировав функцию плотности, получим функция распределения:

$$F_X(x) = \begin{cases} 0, x < a \\ \frac{x-a}{b-a}, a \leq x < b \\ 1, x \geq b \end{cases}$$

На рисунке 1 приведены графики функций плотности и распределения равномерно распределенной непрерывной случайной величины.

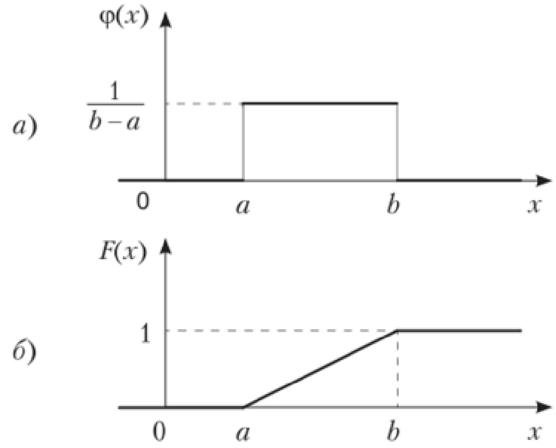

Рисунок 1: Графики плотности и распределения равномерной непрерывной случайной величины

Нормальное распределение

Случайная величина имеет нормальное распределение (обозначается $X \sim N(\mu, \sigma^2)$), если функция ее плотности имеет следующий вид:

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

где

$$x, \mu \in \mathbb{R}, \sigma > 0$$

Проинтегрировав функцию плотности, получим функция распределения:

$$F_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{(x-\mu)^2}{2\sigma^2}} dx,$$

На рисунке 2 приведены графики функций плотности и распределения нормально распределенной непрерывной случайной величины.

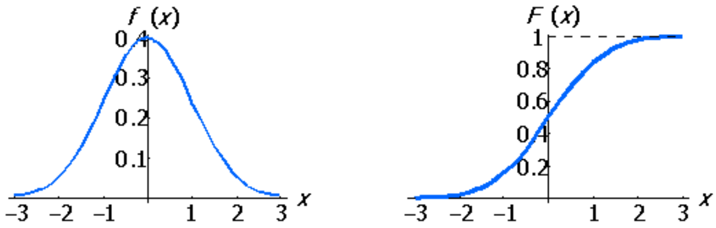

Рисунок 2: Графики плотности и распределения равномерной непрерывной случайной величины

Математическое ожидание (μ) характеризует положение «центра тяжести» вероятностной массы нормального распределения. График плотности распределения нормальной случайной величины симметричен относительно прямой $x = \mu$.

Дисперсия (σ^2) характеризует разброс значений случайной величины относительно мат. ожидания.

Подходы к моделирования

Пошаговый подход

Пошаговый подход 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: значительные затраты и опасность пропуска события при больших Δt.

Событийный подход

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 событий.

Листинги

На листинге 1 приведена функция обработки модели пошаговой обработки.

Листинг 1: Реализация пошагового подхода

```
def step_model(generator, handler, total_tasks=0, repeat=0, step=1e-3):
  t_current = step
  t_gen = generator.yield_value()
  t_gen_prev = t_proc = 0
  processed_tasks = 0
  current_queue_length = 0
  max_queue_length = 0
  free = True

  while processed_tasks < total_tasks:
    # Generator
    if t_current > t_gen:
      current_queue_length += 1
      if current_queue_length > max_queue_length:
        max_queue_len = current_queue_length
      t_gen_prev = t_gen
      t_gen += generator.yield_value()

    # Handler
    if t_current > t_proc:
      if current_queue_length > 0:
        was_free = free
        if free:
          free = False
        else:
          processed_tasks += 1
          if random.randint(1, 100) <= repeat:
            current_queue_length += 1
        current_queue_length -= 1
        if was_free:
          t_proc = handler.yield_value() + t_gen_prev
        else:
          t_proc += handler.yield_value()
      else:
        free = True

    t_current += step

  return max_queue_len
```

На листинге 2 приведена функция обработки модели событийной обработки.

Листинг 2: Реализация событийного подхода

```
def event_model(generator, handler, total_tasks=0, repeat=0):
  handled_tasks = 0
  current_queue_length = 0
  max_queue_length = 0
  events = [{ "value": generator.yield_value(), "type": "generator" }]
```

```
  free, handle_flag = True, False

  while handled_tasks < total_tasks:
    event = events.pop(0)

    # Generator
    if event["type"] == "generator":
      current_queue_length += 1
      if current_queue_length > max_queue_length:
        max_queue_length = current_queue_length
      add_event({ "value": event["value"] + generator.yield_value(), "type": "generator"}, events)
      if free:
        handle_flag = True
    # Handler
    elif event["type"] == "handler":
      handled_tasks += 1
      if randint(1, 100) <= repeat:
        current_queue_length += 1
      handle_flag = True

    if handle_flag:
      if current_queue_length > 0:
        current_queue_length -= 1
        add_event({ "value": event["value"] + handler.yield_value(), "type": "handler" }, events)
        free = False
      else:
        free = True
      handle_flag = False

  return max_queue_length
```

Результаты выполнения работы

Использованные параметры при моделировании:

- равномерного распределения: $a = 1$, $b = 11$;
- нормального распределения: $\mu = 6$, $\sigma = 0.2$;
- шаг для пошагового подхода: 0.01.

На рисунках 3-8 приведены результаты работы системы с, соответственно:

- 1000 заявками, 0% повторов заявок;
- 1000 заявками, 50% повторов заявок;
- 1000 заявками, 100% повторов заявок;

- 10000 заявками, 10% повторов заявок;
- 10000 заявками, 50% повторов заявок;
- 10000 заявками, 100% повторов заявок.

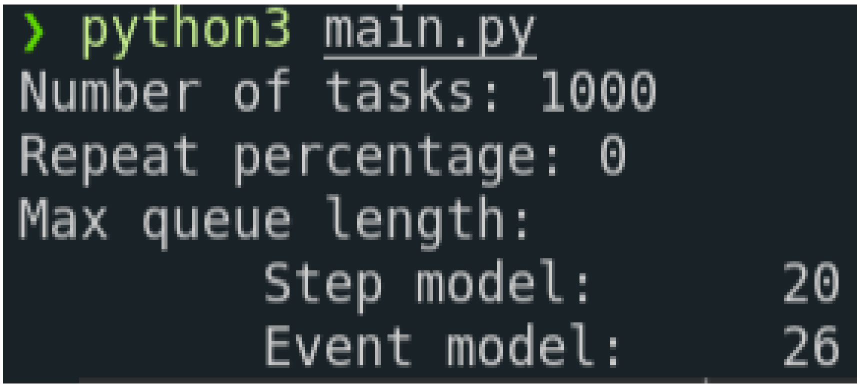

Рисунок 3: Результат моделирования системы с 1000 заявками, 0% повторов заявок

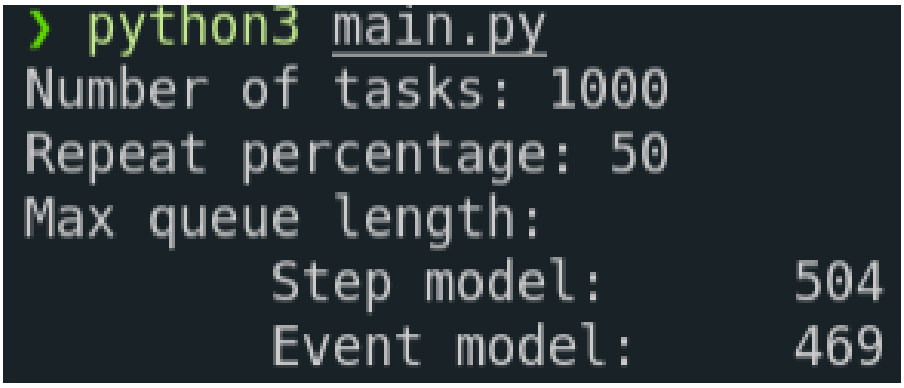

Рисунок 4: Результат моделирования системы с 1000 заявками, 50% повторов заявок

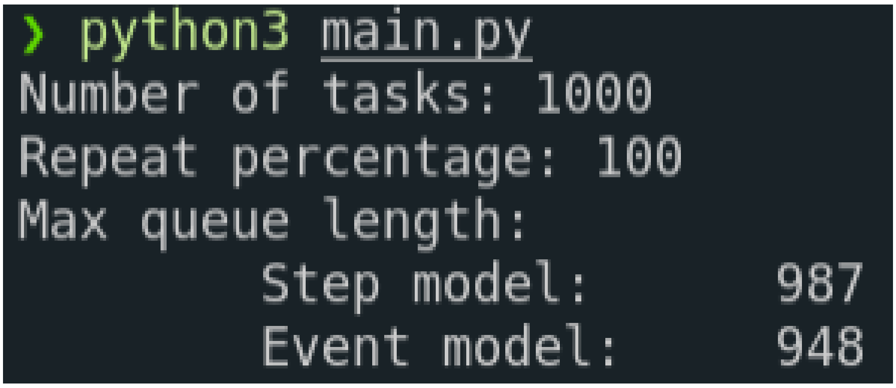

Рисунок 5: Результат моделирования системы с 1000 заявками, 100% повторов заявок

```
❯ python3 main.py
Number of tasks: 10000
Repeat percentage: 10
Max queue length:
        Step model:      1012
        Event model:     981
```

Рисунок 6: Результат моделирования системы с 10000 заявками, 10% повторов заявок

```
❯ python3 main.py
Number of tasks: 10000
Repeat percentage: 50
Max queue length:
        Step model:      5117
        Event model:     5015
```

Рисунок 7: Результат моделирования системы с 10000 заявками, 50% повторов заявок

```
❯ python3 main.py
Number of tasks: 10000
Repeat percentage: 100
Max queue length:
        Step model:      10039
        Event model:     9962
```

Рисунок 8: Результат моделирования системы с 10000 заявками, 100% повторов заявок

Вывод

На основании полученных результатов можно сделать вывод что моделирование с использованием событийного подхода лучше проявляет себя при увеличивающемся процента повтора повторяющихся заявок.