



**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

По курсу “Моделирование”

**Тема: “Моделирование работы системы, состоящей из
генератора, блока памяти и обслуживающего аппарата”**

Студент Лучина Е.Д (№ в списке 13)

Группа ИУ7-71Б

Преподаватель Рудаков И.В.

Москва

2020 г.

Задание

Необходимо промоделировать систему, состоящую из генератора, буферной памяти и обслуживающего аппарата. Генератор выдает сообщения, моменты прихода в очередь которых распределены по равномерному закону. Время, потраченное обслуживающим аппаратом на обработку одной заявки, имеет пуассоновское распределения. Параметры задаются. Часть выходного потока, задаваемая параметром, снова подается в очередь. Необходимо определить оптимальную длину очереди, при которой не будет потерянных сообщений, используя принцип Δt и событийный принцип.

Теоретическая часть

Управляющая программа (УП) имитирует алгоритм взаимодействия устройств системы: генератора, очереди и обслуживающего аппарата. В основном, УП реализуется по 2-м принципам: принцип Δt и событийный принцип.

Принцип Δt

Принцип Δt заключается в последовательном анализе состояний всех блоков системы в момент $t + \Delta t$ по заданному состоянию блоков в момент t . При этом новое состояние блоков определяется в соответствии с их алгоритмическим описанием с учетом действующих случайных факторов, задаваемых распределениями вероятности. В результате этого анализа принимается решение о том, какие общесистемные события должны имитироваться программной моделью на данный момент времени. Чем меньше Δt , тем выше точность моделирования.

Основной недостаток этого принципа - значительные затраты машинных ресурсов. А при недостаточно малой Δt появляется опасность пропуска отдельных событий в системе, что приводит к неадекватности получаемых результатов.

Событийный принцип

Событийный принцип, заключается в том, что состояние всех блоков имитационной модели анализируется лишь в момент появления какого-либо события. Момент поступления следующего события определяется минимальным значением из списка будущих событий, представляющего собой совокупность моментов ближайшего изменения состояния каждого из блоков системы.

Технологическая часть

Сообщение считается потерянным если в момент его прихода буферная память заполнена. Чтобы оценить оптимальную длину очереди, при которой не будет потерянных сообщений, очередь представляется неограниченной.

Максимальная длина очереди за все время моделирования и будет являться оптимальной для данной системы.

Формируются случайные последовательности событий. Далее разными методами продвигается моделируемое время и анализируются события системы. Отметим что равенство событий осуществляется с некоторой точностью (в данном случае $1e-5$). В методе Δt , время продвигается с заданным шагом Δt и если в этот момент времени произошло событие, система его проанализирует. В событийной модели время продвигается от одного события к следующему ближайшему событию.

Результаты работы программы и выводы

```
a: 0
b: 50
lambda: 3
number of requests: 20
reentry probability: 0
start time: 0
end time: 50
time step: 0.001

moments of request coming:
[1.7691697042330923, 12.648816467374523, 14.234630684726957, 22.650025871379942, 23.513011800656
72, 25.83446920033328, 31.32230356752823, 31.397322812441057, 31.63590114457535, 31.822390603517
725, 39.6012383570269, 39.82415268882545, 42.041246042809426, 44.0663361450081, 44.9464425843134
1, 46.17874733738667, 46.60561752947113, 48.63567225768016, 49.52910061267721, 49.73424912917129
]

moments of process events:
[2.0, 6.0, 10.0, 12.0, 13.0, 14.0, 17.0, 22.0, 28.0, 28.0, 28.0, 31.0, 36.0, 39.0, 44.0, 49.0]

+-----+-----+-----+
|          | max queue length | number of processed requests |
+-----+-----+-----+
| delta_t |          0       |          0                   |
| events  |          10      |          10                  |
+-----+-----+-----+
```

В данном примере равномерно в промежуток времени от 0 до 50 в очередь пришло 20 заявок. Обработанные заявки не отправляются в очередь повторно. Обслуживающий аппарат начинает обработку с приблизительно с момента прихода первой заявки и затрачивает на обработку время, распределенное по закону Пуассона с параметром лямбда, равным трем. Рассмотрим время моделирования от 0 до 50 с шагом 0.001. Очевидно, что метод Δt не дает удовлетворительных результатов, заданных шаг как бы “проходит мимо” всех событий.

Уменьшим шаг:

```

a: 0
b: 50
lambda: 3
number of requests: 20
reentry probability: 0
start time: 0
end time: 50
time step: 1e-05

moments of request coming:
[3.7667851234346905, 9.752610786223837, 13.211106297937619, 14.04146695137305, 15.20502968709999,
15.20528213871265, 15.958959029440118, 16.170208017249493, 22.522939943419466, 22.709500705758
312, 23.881582714127603, 32.438206004545435, 33.99295183388852, 34.00051075554173, 35.2164196369
1915, 35.48641448786287, 40.935261755716425, 43.61537777127836, 44.893449043781224, 47.195930868
06573]

moments of process events:
[4.0, 5.0, 8.0, 9.0, 11.0, 15.0, 18.0, 23.0, 25.0, 28.0, 31.0, 32.0, 36.0, 40.0, 44.0, 48.0]

+-----+-----+-----+
|          | max queue length | number of processed requests |
+-----+-----+-----+
| delta_t |          8          |          13          |
| events  |          8          |          13          |
+-----+-----+-----+

```

Теперь методы дают одинаково хорошие результаты. Однако на моделирование системы первым методом уходит куда больше вычислительных ресурсов.

Рассмотрим разные вероятности возврата заявки в очередь после обработки. 50%.
Максимальная длина очереди за время моделирования увеличилась, Ведь обслуживающий аппарат работает так же, а заявок стало больше.

```

a: 0
b: 50
lambda: 3
number of requests: 20
reentry probability: 0.5
start time: 0
end time: 50
time step: 1e-05

moments of request coming:
[1.9326601790909725, 5.07545675442867, 6.815632671350135, 7.027890812940585, 9.720467388013232,
10.129394977080548, 15.238220710197176, 17.768565334994136, 21.204536795700278, 21.3979622493963
1, 25.62554996151938, 26.03075285804169, 26.395632689553082, 26.781872489417097, 28.255763746968
466, 31.474153560131224, 34.83754903364287, 36.95567567742415, 37.35588377265661, 44.20055835767
5426]

moments of process events:
[2.0, 9.0, 13.0, 15.0, 18.0, 18.0, 19.0, 23.0, 23.0, 24.0, 26.0, 27.0, 30.0, 33.0, 34.0, 36.0, 4
0.0, 43.0, 48.0, 48.0, 49.0]

+-----+-----+-----+
|          | max queue length | number of processed requests |
+-----+-----+-----+
| delta_t |          12          |          21          |
| events  |          17          |          21          |
+-----+-----+-----+

```

Рассмотрим разные параметры для распределений случайных величин.

Параметры а и b определяют диапазон времени прихода заявок. Количество заявок определяют частоту. Увеличивая лямбду, увеличиваем время обработки заявки.

a: 0
b: 10
lambda: 5
number of requests: 60
reentry probability: 0
start time: 0
end time: 30
time step: 1e-05

moments of request coming:
[0.5525267716335647, 0.694405933334522, 0.8064090015324354, 0.8514682610415758, 1.0279278090354627, 1.1180171008999573, 1.308695107741159, 1.4571269513747043, 1.8794779856561128, 2.1084791980473847, 2.17966614475817, 2.377442726459188, 2.5595050673200337, 2.5601450524708023, 2.5989045989224255, 2.7995620900234286, 3.0956266473446146, 3.207614527149647, 3.2213669560202387, 3.376838483814816, 3.417977887897613, 3.5771717337935716, 3.672116291446126, 3.6733456144819012, 4.157213518349385, 4.2615886499705375, 4.473611021703628, 4.505452839768571, 4.5470743167968894, 4.741836078503228, 4.792657067466907, 4.882248852980079, 5.3748495298716, 5.616845765225618, 5.748568201989213, 6.0478758898861615, 6.378881040272383, 6.531329033300235, 6.5350965950168085, 6.595078510587612, 7.077326962978725, 7.0848247684160635, 7.209295406110883, 7.3005396785231005, 7.783235883690033, 8.031198803977384, 8.056103394654219, 8.196304056699232, 8.199192849298294, 8.555315772033161, 8.572132926328532, 8.670891522145709, 8.72349437646966, 8.73375070233392, 9.05658981495428, 9.172172903451154, 9.305473906643416, 9.42266292911071, 9.68386202604738, 9.835472956341588]

moments of process events:
[1.0, 3.0, 8.0, 12.0, 17.0, 20.0, 24.0, 29.0]

max queue length		number of processed requests	
delta_t	57		8
events	57		8