|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт комплексной безопасности и специального приборостроения

(наименование института, филиала)

\_\_\_\_Аппаратное, программное и математическое обеспечение вычислительных систем

(наименование кафедры)

**КУРСОВОЙ ПРОЕКТ (РАБОТА)**

по дисциплине \_\_\_Методы моделирования информационных систем\_\_\_\_\_\_\_

(наименование дисциплины)

**Тема курсового проекта** (работы): \_\_\_Разработка имитационной модели \_\_\_\_\_\_\_\_ вычислительной системы\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Студент группы** БСБО-01-17, Высокая Александра Ильинична \_\_\_\_\_\_\_\_\_\_\_\_

(учебная группа, фамилия, имя, отчество студента) (подпись студента)

**Руководитель курсового проекта (работы)** ст. преподаватель

должность, звание, ученая степень (подпись руководителя)

**Рецензент** (при наличии) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность, звание, ученая степень подпись рецензента

Работа представлена к защите « »\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

Допущен к защите « »\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт комплексной безопасности и специального приборостроения

(наименование института, филиала)

\_\_\_\_Аппаратное, программное и математическое обеспечение вычислительных систем

(наименование кафедры)

Утверждаю

Заведующий кафедрой\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

подпись ФИО

«\_\_»\_\_\_\_\_\_\_\_\_20\_\_г.

**ЗАДАНИЕ**

**на выполнение курсового проекта (работы)** по дисциплине

«\_\_\_\_\_\_\_\_\_\_\_\_\_Методы моделирования информационных систем\_\_\_\_\_\_\_\_\_\_»

Студент \_\_\_Высокая Александра Ильинична Группа \_\_\_\_БСБО-01-17\_\_\_\_\_

**Тема \_\_\_\_\_**Разработка имитационной модели ВС

**Исходные данные:**

Вычислительная система (ВС) состоит из 4-х серверов, обрабатывающих программы. Программы поступают случайным образом, распределенные по линейному закону: Tzmin=1/2 сек, Tzmax=5/6 сек. Если 1-ый сервер занят, то программы обрабатываются 2-ым сервером и т.д. Если и 1-ый и 2-ой, 3-ий и 4-ый серверы заняты, то программа поступает в буфер. Число программ в буфере ограничено – не более 3-х. Если буфер заполнен, то программа покидает ВС необработанной.

Время обработки одной программы каждым сервером – случайная величина, распределенная по линейному закону: Tsmin=1 сек, Tsmax=5 сек. Разработать программу, моделирующую работу ВС и найти ее характеристики за время работы 1 час.

Характеристики ВС:

· P0 – вероятность того, что ВС не загружена,

· P1 – вероятность того, что загружен только один сервер,

· P2 – вероятность того, что загружены два сервера,

· P3 – вероятность того, что загружены три сервера,

· P4 – вероятность того, что загружены четыре сервера,

· P5 – вероятность того, что в буфере находится 1-на программа,

· P6 – вероятность того, что в буфере находится 2-ве программы,

· P7 – вероятность того, что в буфере находится 3-ри программы,

· Q – относительная пропускная способность ВС – средняя доля программ, обработанных ВС,

· S – абсолютная пропускная способность – среднее число программ, обработанных в единицу времени,

· Pотк – вероятность отказа, т.е. того, что программа будет не обработанной,

· K - среднее число занятых серверов,

· Nпрог.- среднее число программ в ВС,

· Tпрог – среднее время нахождения программы в ВС,

· Nбуф.- среднее число программ в буфере,

· Tбуф – среднее время нахождения программы в буфере.

Найти характеристики ВС, если программы поступают случайным образом, распределенные по экспоненциальному закону с частотой λ=1,5 1/сек, а среднее время обработки программы каждым сервером составляет tобр= 2 сек (закон распределения -экспоненциальный).

Срок предоставления к защите курсового проекта: до «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Задание на курсовой проект выдал: \_\_\_\_\_\_\_\_\_\_\_\_ (Александров А.Е.)

*(подпись) (ФИО)*

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Задание на курсовой проект получил: \_\_\_\_\_\_\_\_\_\_\_\_ (Высокая А.И.)

*(подпись) (ФИО)*

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

**Аннотация**

Представленная курсовая работа состоит из введения, имитационной модели, модели ВС на основе теории массового обслуживания, верификации и проверки достоверности логической схемы модели, анализа результатов, заключения, списка литературных источников и приложения.

В курсовой работе рассматривается имитационная модель вычислительной системы, обрабатывающей входной поток данных в виде задач, заданных определенных законом распределения. Также рассчитываются характеристики данной модели и делается вывод о ее соответствии поставленной задаче.

Работа состоит из 33 страницы, содержит 3 использованных источника, 1 таблицу, 6 рисунков и 1 приложение.

**Содержание**

[Введение 6](#_Toc58607002)

[1. Имитационная модель вычислительной системы 7](#_Toc58607003)

[1.1 Вероятностные характеристики входного потока и обработки 7](#_Toc58607004)

[1.2 Логическая схема модели ВС, ее описание 9](#_Toc58607005)

[1.3 Управление модельным временем 9](#_Toc58607006)

[2. Модель ВС на основе теории массового обслуживания 14](#_Toc58607007)

[3. Верификация и проверка достоверности логической схемы модели 17](#_Toc58607008)

[4. Анализ результатов машинных экспериментов 17](#_Toc58607009)

[Заключение 21](#_Toc58607010)

[Список использованных источников 21](#_Toc58607011)

[Приложение 22](#_Toc58607012)

# **Введение**

Реализация вычислительной системы выполнена с использованием язык программирования JavaScript. Язык был выбран из-за наличия опыта работы с ним, его универсальности и совместимости с языком разметки HTML, на котором написан интерфейс к данной программе.

Цель работы – получение практических навыков разработки имитационной модели ВС, анализ ее работы, проверка и сопоставление результатов с аналитическими зависимостями, проверка правильности логической части имитационной модели.

Актуальной задачей на сегодняшний день является создание моделей вычислительных систем, так как такие системы окружают повсюду. Компьютерные модели стали обычным инструментом математического моделирования и применяются в различных сферах. Компьютерные модели используются для получения новых знаний о моделируемом объекте или для приближенной оценки поведения систем, слишком сложных для аналитического исследования.

Моделируя реальные процессы, которые могут проходить в жизни, мы можем прогнозировать качество работы системы. Изменяя параметры системы можно проследить как зависит эффективность работы от них. Таким образом, изучив результаты данного моделирования, можно применить их на практике.

Идеальную работу системы нельзя гарантировать, но результаты данного моделирования будут максимально приближенными к реальным показателям.

# **Имитационная модель вычислительной системы**

## **1.1 Вероятностные характеристики входного потока и обработки**

Системы массового обслуживания (СМО) – это такие системы, в которые:

* в случайные моменты времени поступают заявки на обслуживание;
* поступившие заявки обслуживаются с помощью имеющихся в распоряжении системы каналов обслуживания (приборы, люди, сервера);
* обслуживание заявок продолжается какое-то случайное время;
* после чего канал освобождается и готов к приему следующей заявки.

Поступление заявки в СМО называется событием. Последовательность событий, заключающихся в поступлении заявок в СМО, называется входящим потоком заявок.

Для описания потока заявок необходимо задать интервалы времени между соседними моментами поступления заявок.

Основной характеристикой потока заявок является его интенсивность λ

(среднее число заявок, поступивших на вход ВС за единицу времени). Величина T = 1/λ определяет средний интервал времени между двумя последовательными заявками.

Для полного описания регулярного потока заявок достаточно знать интенсивность потока λ или значение интервала T.

Мы можем создать для заданной модели входной поток программ двух видов:

1. Программы поступают случайным образом, распределенные по линейному закону. Время прихода программы находится в диапазоне: = 1/2 сек, = 5/6 сек.

Время обработки одной программы сервером – случайная величина, распределенная по линейному закону: = 1 сек, = 5 сек.

1. Программы поступают случайным образом, распределенные по экспоненциальному закону с частотой λ = 1,5 1/сек, а среднее время обработки программы каждым сервером составляет = 2 сек.

Интенсивность, с которой обслуживающий прибор способен продвигать заявки на выход равно µ = 1/, где – среднее время обслуживания заявки.

## **1.2 Логическая схема модели ВС, ее описание**

На вход поступает поток заявок. В программе потоком является массив из объектов. Объект – это заявка, в которой хранятся данные: момент времени, когда поступила заявка, и время ее обработки.

При распределении по линейному закону время прихода заявки вычисляется по формуле:

*,*

а время обработки по формуле:

.

При распределении по экспоненциальному закону время прихода заявки вычисляется по формуле:

,

а время обработки по формуле:

.

Сервера в программе представлены в виде объектов. Данный объект хранит массив с заявками, которые поступили на этот сервер, и время, когда сервер закончит обработку очередной программы.

При поступлении заявки производится проверка каждого сервера, свободен ли он в момент прихода программы. Если сервер свободен, то программа проверяет, есть ли что-то в буфере. Если есть, то на сервер, который освободился раньше остальных, поступает заявка из буфера. После этого снова проверяем, может ли поступившая заявка обработаться каким-то сервером. Если да, и если буфер пуст, то заявка поступает на сервер. Иначе снова достаем заявку из буфера. Если же нет ни одного сервера, который может обработать заявку, то она попадает в буфер, но только если там есть место, иначе она остается необработанной.

Окончание работы программы происходит при появлении заявки с временем прихода больше заданного времени работы программы (по заданию 1 час, но в интерфейсе программы его можно изменить).

Блок-схем модели ВС представлена на рисунке 1.

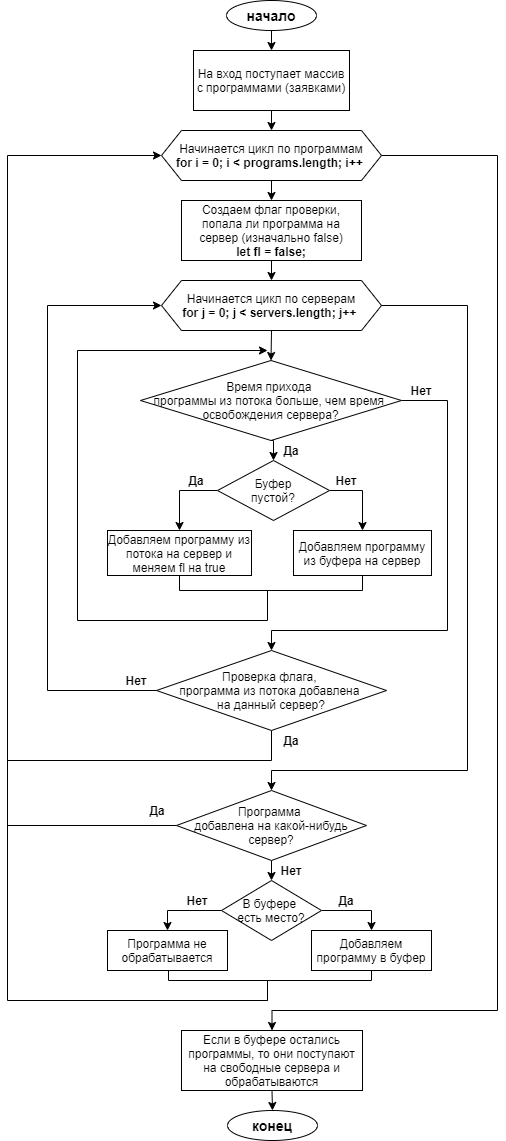


Рисунок 1. Блок-схема модели ВС

## **Управление модельным временем**

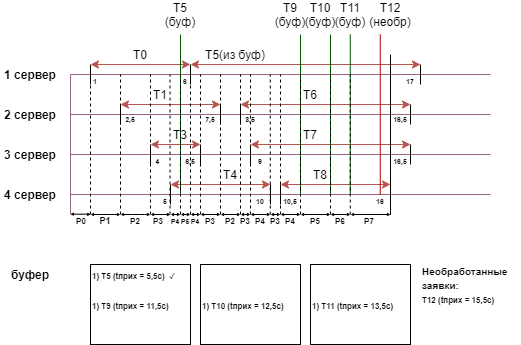


Рисунок 2. Моделирование времени по моментам прихода заявок

– это вероятность нахождения в состоянии k (k = n + m), которая находится как . Время работы системы фиксированное значение, которое пользователь задает самостоятельно в интерфейсе программы. Время нахождения системы в состоянии k программа считает по алгоритму каждый раз при поступлении на сервер новой заявки.

При добавлении заявки из потока время вычисляется по алгоритму, указанному на рисунке 3.1.

При добавлении заявки из буфера время вычисляется по алгоритму, указанному на рисунке 3.2.

При добавлении заявки в буфер время вычисляется по алгоритму, указанному на рисунке 3.3.

Самое главное, вычисление времени производить перед добавлением заявки на сервер или в буфер и до удаления заявки из буфера.

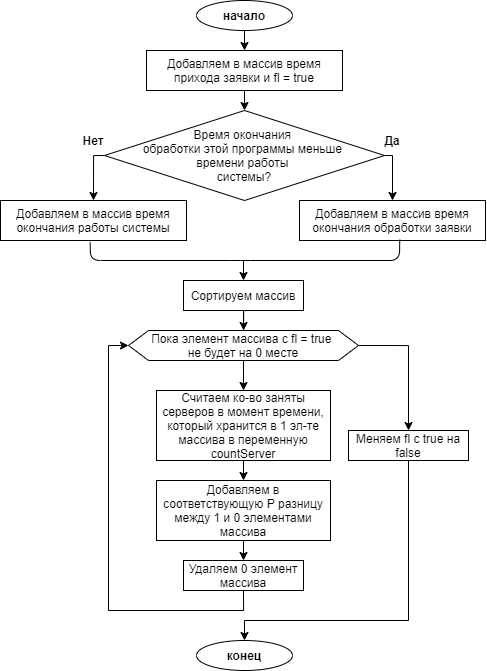


Рисунок 3.1 Блок схема вычислений времени нахождения системы в состоянии n при добавлении заявки из потока.

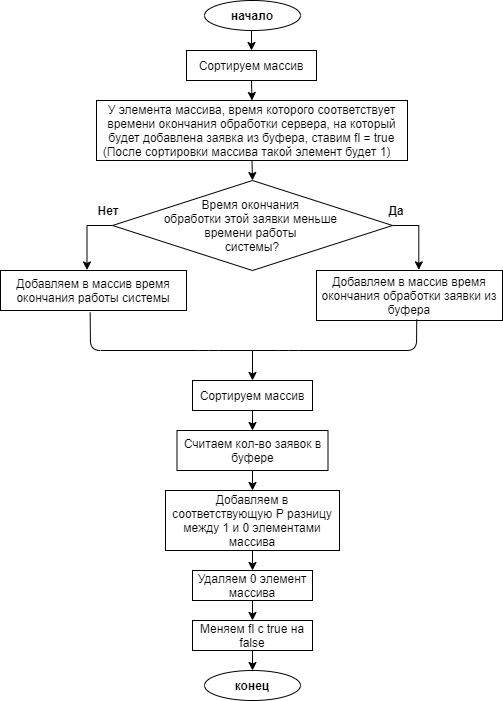


Рисунок 3.2 Блок схема вычислений времени нахождения системы в состоянии n при добавлении заявки из буфера.

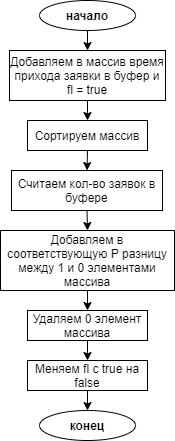


Рисунок 3.3 Блок схема вычислений времени нахождения системы в состоянии n при добавлении заявки в буфер.

# **Модель ВС на основе теории массового обслуживания**

Системой массового обслуживания называется любая система для выполнения заявок, поступивших в нее в случайные моменты времени.

Каналом обслуживания в СМО называется устройство в СМО, обслуживающее заявку. СМО, содержащее один канал обслуживания, называется одноканальной, а содержащее более одного канала обслуживания – многоканальной.

Если заявка, поступающая в СМО, может получить отказ в обслуживании, в силу занятости всех каналов обслуживания, и в случае отказа вынуждена покинуть СМО, то такие СМО называются СМО с отказами.

Если в случае отказа в обслуживании заявки могут встать в очередь, то такие СМО называются с очередью. При этом различают СМО с ограниченной и неограниченной очередью.

В нашем случае имитационная модель ВС представляется собой многоканальную СМО с ограниченной очередью.

Граф состояний многоканальной СМО с ограниченной очередью представлен на рисунке 4.

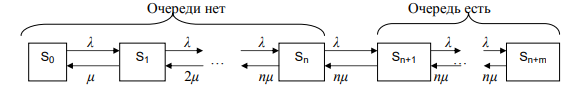
****

Рисунок 4. Граф состояний многоканальной

СМО описывается некоторыми параметрами, которые характеризуют эффективность работы системы.

n – кол-во каналов в системе массового обслуживания;

m – допустимое кол-во заявок в очереди;

µ - интенсивность обслуживания заявок;

λ – интенсивность поступления в СМО заявок;

ρ = λ / µ - коэффициент использования СМО;

Для нахождения предельных вероятностей можно использовать следующие формулы:

– вероятность отказа в обслуживании поступившей в СМО заявки;

Q – относительная пропускная способность СМО (вероятность обслуживания поступившей в СМО заявки):

А – абсолютная пропускная способность СМО (среднее число заявок, обслуживаемых в СМО в единицу времени):

K – среднее число занятых серверов:

– среднее число находящихся в очереди заявок:

– среднее число находящихся в системе заявок:

– среднее время пребывания заявки в очереди:

– среднее время пребывания заявки в системе:

# **Верификация и проверка достоверности логической схемы модели**

Проверку достоверности логической схемы модели можно осуществить двумя способами:

1. через сопоставление результатов работы программы с аналитическими расчетами;
2. через сопоставление результатов работы программы с результатами работы программы-образца.

**Сопоставление с аналитическими расчетами**

Сравнивать будем со значениями, полученными при распределении входного потока данных по экспоненциальному закону.

|  |  |  |
| --- | --- | --- |
| **Параметр** | **Аналитический расчет по формулам** | **Значения из программы** |
| N – кол-во каналов в СМО | 4 | 4 |
| M – размер буфера | 3 | 3 |
| λ – частота поступления программ | 1,5 | 1,5 |
| **-** среднее время обработки программы; | 2 | 2 |
| µ = 1/– интенсивность обслуживания заявок; | 0,5 | 0,5 |
| ρ = λ / µ - коэффициент использования СМО; | 3 | 3 |
| – вероятность того, что ВС не загружена; |  | 0,01048 |
| - вероятность того, что загружен один сервер; | 0,1349 | 0,08339 |
| - вероятность того, что загружены два сервера; | 0,2024 | 0,22691 |
| - вероятность того, что загружены три сервера; | 0,2024 | 0,28887 |
| - вероятность того, что загружены четыре сервера; | 0,1518 | 0,19241 |
| - вероятность того, что в буфере 1-на заявка; | 0,1138 | 0,11574 |
| - вероятность того, что в буфере 2-ве заявка; | 0,0854 | 0,0533 |
| - вероятность того, что в буфере 3 заявка; | 0,06404 | 0,0289 |
| – вероятность отказа; | 0,06404 | 0,032 |
| – относительная пропускная способность ВС (средняя доля программ, обработанных ВС) | 0,93596 | 0,968 |
| – абсолютная пропускная способность (среднее число заявок, обслуживаемых за единицу времени); | 1,40394 | 1,451 |
| – среднее число занятых серверов; | 2,80788 | 2,903 |
| – среднее число заявок в буфере; | = 0,47677 | 0,111 |
| – среднее число заявок в системе; | 3,28465 | 3,014 |
| – среднее время нахождения заявки в буфере; | 0,31785 | 0,074 |
| – среднее время пребывания заявки в системе. | 2,18977 | 2,009 |

Таблица 1. Сравнительная характеристика результатов расчета

При сравнении результатов работы логической модели с результатами аналитических формул видна некоторая погрешность. Однако она стремится к нулю и в некоторых отдельных случаях исчезает при округлении или составляет не больше 0,5, что не является результатом, влияющим на итоговый результат. Можно считать, что логическая схема модели ВС рассчитывает параметры правильно и полностью соответствует теоретической модели многоканальной СМО с ограниченной очередью.

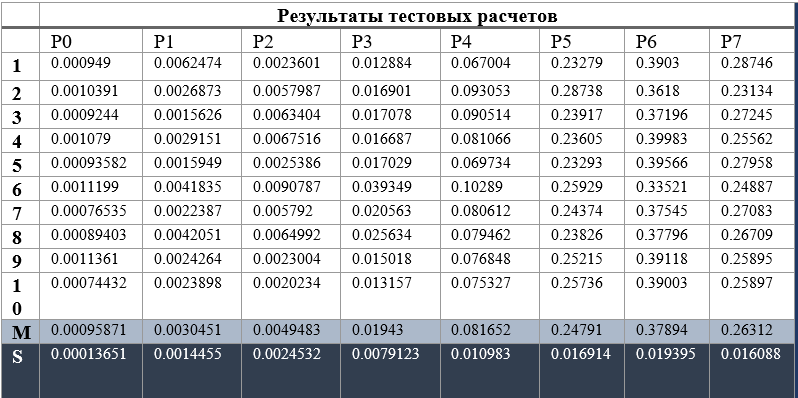
**Сопоставление с результатами работы программы-образца**

Рисунок 5.1 Результаты работы программы-образца

****

Рисунок 5.2 Результаты работы программы студента

При сравнении результатов среднеквадратичного отклонения в логической модели и тестовой программе видна небольшая погрешность, порядка десятитысячных долей. Это означает, что разброс вероятностей в разработанной программе соответствует разбросу вероятностей в программе-образце.

# **Анализ результатов машинных экспериментов**

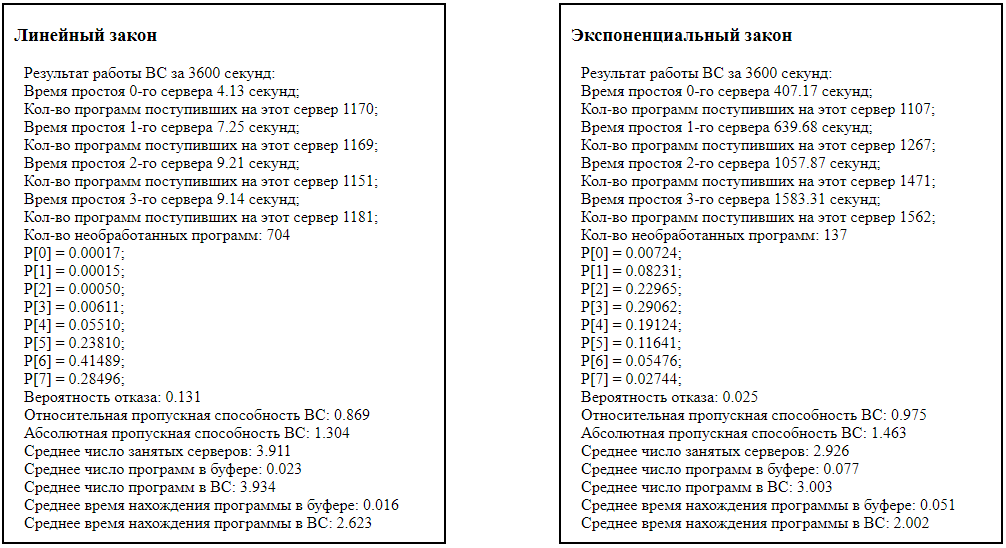
****

Рисунок 6.1 Результаты работы программы по заданным в задании параметрам

По результатам вычислений можно заметить, что при распределении по линейному закону система вероятнее находится в состоянии полного буфера, чем в состоянии, близкому к простою.

При распределении по линейному закону время прихода программы находится в диапазоне [1/2; 5/6] (данные параметры указаны в задании), а при распределении по экспоненциальному закону подобных ограничений нет, поэтому время бывает и 0,1 сек, и 1 сек, а иногда и 2 сек.

Так же, распределении по линейному закону интенсивность обслуживания заявок µ = 0,3, а при распределении по экспоненциальному закону µ = 0,5. Из этого следует, что система с линейным распределением обрабатывает заявки медленнее.

По этим причинам при линейном распределении большее кол-во заявок остаются необработанными, каналы постоянно заняты, а стоит им освободится, как приходится обрабатывать заявки из буфера.

Чтобы приравнять частоту поступления программ (λ) и интенсивность обслуживания заявок (µ), возьмем Tzmin = 0, Tzmax = 4/3, Tsmin = 0, Tsmax = 4. При таких параметрах результаты работы программы, следующие:

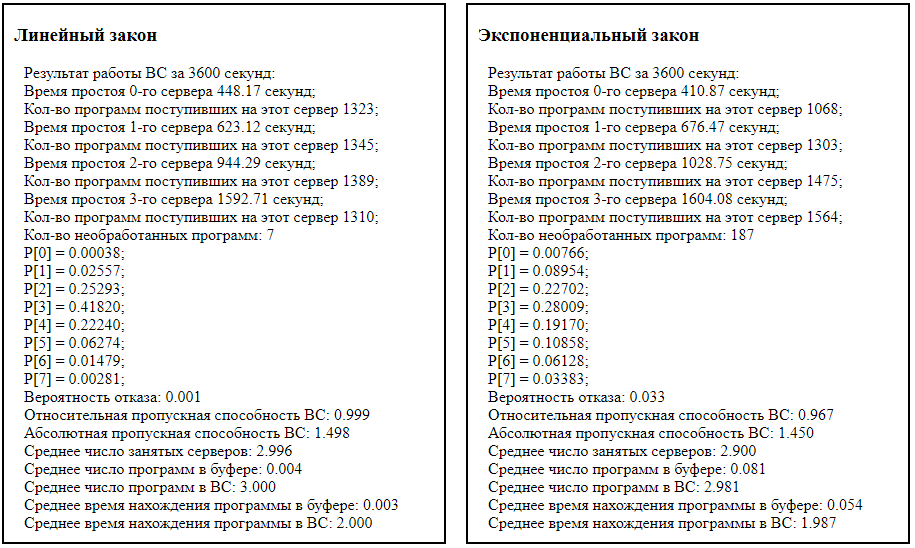


Рисунок 6.2 Результаты работы программы по рассчитанным параметрам

Исходя из полученных результатов, можно сделать вывод, что система при распределении по линейному закону работает лучше. Это видно по всем параметрам. Также, можно отметить, что абсолютная пропускная способность и среднее число занятых серверов практически одинаковы.

Коэффициент отношения абсолютных пропускных способностей ВС: 1,498/1,450 = 1,033

# **Заключение**

По результатам тестирования можно сделать вывод, что полученные значения программы близки к значениям аналитических расчетов. Значения, полученные программным путем, близки к теоретическим. Наличие погрешности в результатах работы программы, а также разница результатов тестовой программы и аналитических расчетов объясняется природой псевдослучайных чисел.

Таким образом, логическая часть имитационной модели прошла проверку правильности своей работы. Получены значения всех требуемых характеристик для данных по исходному заданию.

Отметим, что разработанная программа может моделировать различные виды систем массового обслуживания.

# **Список использованных источников**

1. Теория массового обслуживания. Методические указания, учебная программа и задания для контрольных работ № 1, 2 для студентов заочной формы обучения специальности 071900 “Информационные системы в технике и технологиях”. - Самара: СамГАПС, 2002.- 38с.
2. С. Н. Колупаева. Математическое и компьютерное моделирование. Учебное пособие. – Томск, Школьный университет, 2008. – 208с.
3. http://e-biblio.ru/book/bib/06\_management/teor\_mass\_obslug/158.9.17.html

# **Приложение**

function randomInteger(*min*, *max*) { *//функция рандома*

*// случайное число от min до max*

  let rand = *min* + Math.random() \* (*max* + 1 - *min*);

  return Math.floor(rand);

}

*//нахождение программ по линейному закону*

function calcProgramsLine(*TzMIN*, *TzMAX*, *TsMIN*, *TsMAX*, *TIMEWORK*) {

  let programs = [];

  let t = 0;

  let lastT = 0;

  while (t <= TIMEWORK) {

    const U = randomInteger(1, 100) / 100;

    const tPrihod = (TzMAX - TzMIN) \* U + TzMIN;

    const tObrabotka = (TsMAX - TsMIN) \* U + TsMIN;

    const T = lastT + tPrihod;

    if (T <= TIMEWORK) {

      programs.push({

        U: U,

        tPrihod: tPrihod,

        tObrabotka: tObrabotka,

        time: T,

        inBoof: false,

      });

    }

    lastT = T;

    t = T - programs[0].time; *//т.к. ВС начинает работу с приходом первой*

*//программы*

  }

  return programs;

}

*//нахождение программ по экспоненциальному закону*

function calcProgramsExp(*lambda*, *Mu*, *TIMEWORK*) {

  let programs = [];

  let t = 0;

  let lastT = 0;

  while (t <= *TIMEWORK*) {

    const U = randomInteger(1, 100) / 100;

    const tPrihod = -1 / *lambda* \* Math.log(U);

    const tObrabotka = -1 / *Mu* \* Math.log(U);

    const T = lastT + tPrihod;

    if (T <= *TIMEWORK*) {

      programs.push({

        U: U,

        tPrihod: tPrihod,

        tObrabotka: tObrabotka,

        time: T,

        inBoof: false,

      });

    }

    lastT = T;

    t = T - programs[0].time;

  }

  return programs;

}

function creatServers(*n*) { *// создание серверов*

  let servers = [];

  for (let i = 0; i < *n*; i++) {

    servers.push({ id: i + 1, array: [], t: 0, downTime: 0 })

  }

  return servers;

*//array - массив программ поступивших на данный сервер,*

*//downTime - время простоя сервера*

*//t - время, когда освободится сервер, id - номер сервера*

}

const factorial = (*n*) => { *//функция нахождения факториала*

  return (*n* != 1) ? *n* \* factorial(*n* - 1) : 1

}

const math = (*r*, *n*, *m*) => { *//функция вычисления*

  return (Math.pow(*r*, (*n* + *m*)) / (Math.pow(*n*, *m*) \* factorial(*n*)) )

}

const calcBoofN = (*r*, *n*, *m*, *x*, *p0*) => { *//функция нахождения среднего числа*

*//программ в буфере*

  let E = 0;

  if (*x* === 1) {

    for (let i = 0; i < *m*; i ++) {

      E += (i + 1) \* Math.pow(*x*, i);

    }

  } else {

    E = (1 - Math.pow(*x*, *m*) \* (*m* + 1 - *m*\**x*))/Math.pow( (1-*x*), 2);

  }

  return (math(*r*, *n*, 1) \* *p0* \* E);

}

*//моделирование работы ВС с расчетом характеристик*

function modeling(*programs*, *servers*, *n*, *m*, *time*, *L*, *Mu*) {

  let boof = []; *//буфер*

  let count = 0; *//кол-во необработанных программ*

  let P = []; *//массив вероятноесте P0 - P7*

  for (let i = 0; i <= *n* + *m*; i++) { *//заполняем массив вероятностей нулями,*

*//чтобы не было ошибок в вычислении*

    P.push(0);

  }

  let arrayForParametrs = [{time: 0, flagPrihoda: false}]; *//массив для*

*//вычисления вероятностей*

  for (let i = 0; i < *programs*.length; i++) { *//цикл по заявкам (программам)*

    let fl = true;  *//если true, то программа не попадает на сервер*

    for (let j = 0; j < *n*; j++) { *//цикл по серверам*

      while (*programs*[i].time > *servers*[j].t) { *//пока сервер свободен*

        if (boof.length > 0) { *//если в буфере есть заявки,*

*//то обработаем сначала их*

*//вычисляем время для вероятностей*

*servers*.sort((*elem1*, *elem2*) => *elem1*.t - *elem2*.t);

          arrayForParametrs.sort((*elem1*, *elem2*) => *elem1*.time - *elem2*.time);

          arrayForParametrs[1].flagPrihoda = 'true';

*//если время превысило 3600*

          if (arrayForParametrs[1].time + boof[0].tObrabotka <= *time*) {

            arrayForParametrs.push({ time: arrayForParametrs[1].time +

boof[0].tObrabotka, flagPrihoda: false });

          } else {

          arrayForParametrs.push({ time: *time*, flagPrihoda: false });

          }

          arrayForParametrs.sort((*elem1*, *elem2*) => *elem1*.time - *elem2*.time);

          P[*n* + boof.length] += arrayForParametrs[1].time –

arrayForParametrs[0].time;

          arrayForParametrs.shift();

          arrayForParametrs[0].flagPrihoda = false;

*//добавляем программу из буфера на сервер*

          const progaOutBoof = boof.shift();

*servers*[0].array.push(progaOutBoof);

*servers*[0].t += progaOutBoof.tObrabotka;

*servers*.sort((*elem1*, *elem2*) => *elem1*.id - *elem2*.id);

        } else { *//в буфере пусто, поэтому добавляем программу на сервер*

*//вычисляем время для вероятностей*

            arrayForParametrs.push({ time: *programs*[i].time, flagPrihoda: true })

            if (*programs*[i].time + *programs*[i].tObrabotka <= *time*) {

              arrayForParametrs.push({ time: *programs*[i].time +

*programs*[i].tObrabotka, flagPrihoda: false });

            } else {

              arrayForParametrs.push({ time: *time*, flagPrihoda: false });

            }

            let flag = false; *//флаг проверки первого элемента массива*

*//для вычисления вероятностей*

            while (!flag) {

              arrayForParametrs.sort((*elem1*, *elem2*) => *elem1*.time - *elem2*.time);

              let countServers = 0;

              for (let k = 0; k < *n*; k++) {

                if (arrayForParametrs[1].time <= *servers*[k].t) {

                  countServers++;

                }

              }

              P[countServers] += arrayForParametrs[1].time –

arrayForParametrs[0].time;

              arrayForParametrs.shift();

              if (arrayForParametrs[0].flagPrihoda) {

                flag = true;

                arrayForParametrs[0].flagPrihoda = false;

              }

            }

*//добавляем программу на сервер*

*servers*[j].array.push(*programs*[i]);

*servers*[j].downTime += *programs*[i].time - *servers*[j].t;

*servers*[j].t = *programs*[i].time + *programs*[i].tObrabotka;

            fl = false; *//программа попала на сервер*

        }

      }

      if (!fl) { *//если программа попала на сервер,*

*//то прекращаем перебор по серверам*

        break;

      }

    }

    if (fl) { *//если программа не попала на сервер, то*

      if (boof.length < *m*) { *//есть ли место в буфере*

*//считаем время для вероятностей*

        arrayForParametrs.push({ time: *programs*[i].time, flagPrihoda: true });

        arrayForParametrs.sort((*elem1*, *elem2*) => *elem1*.time - *elem2*.time);

        P[*n* + boof.length] += arrayForParametrs[1].time –

arrayForParametrs[0].time;

        arrayForParametrs.shift();

        arrayForParametrs[0].flagPrihoda = false;

*//добавляем программу в буфер*

*programs*[i].inBoof = true;

        boof.push(*programs*[i]);

      } else { *//если программа не обрабатывается*

        count++; *//считаем кол-во необработанных программ*

      }

    }

  }

  while (boof.length > 0) { *//если в буфере остались программы в конце*

*servers*.sort((*elem1*, *elem2*) => *elem1*.t - *elem2*.t)

*//вычисляем время для вероятностей*

    arrayForParametrs.sort((*elem1*, *elem2*) => *elem1*.time - *elem2*.time);

    arrayForParametrs[1].flagPrihoda = 'true';

    if (arrayForParametrs[1].time + boof[0].tObrabotka <= time) {

      arrayForParametrs.push({ time: arrayForParametrs[1].time +

boof[0].tObrabotka, flagPrihoda: false });

    } else {

      arrayForParametrs.push({ time: time, flagPrihoda: false });

    }

    arrayForParametrs.sort((*elem1*, *elem2*) => elem1.time - elem2.time);

    P[n + boof.length] += arrayForParametrs[1].time - arrayForParametrs[0].time;

    arrayForParametrs.shift();

    arrayForParametrs[0].flagPrihoda = false;

*//вытаскиваем программы из буфера и отправляем на сервер*

    const progaOutBoof = boof.shift();

    servers[0].array.push(progaOutBoof);

    servers[0].t += progaOutBoof.tObrabotka;

    servers.sort((*elem1*, *elem2*) => elem1.id - elem2.id)

  }

*//вычисление остальных хар-к*

  const Ro = L / Mu;

  const X = Ro/n;

*//вероятность отказа в обслуживании заявки*

  const otkazP = count/programs.length;

  const Q = 1 - otkazP; *//Относительная пропускная способность*

  const S = L \* Q; *//Абсолютная пропускная способность*

  const K = S / Mu; *//Среднее число занятых каналов*

  const BoofN = calcBoofN(Ro, n, m, X, (P[0]/time)); *//среднее число программ*

*//в буфере*

  const ProgN = K + BoofN; *//среднее число программ в ВС*

  const BoofT = BoofN / (Ro \* Mu); *//среднее время нахождения программы в буфере*

  const ProgT = BoofT + Q / Mu; *//среднее время нахождения программы в ВС*

let text = `Результат работы ВС за ${time} секунд:`;

for (let i = 0; i < n; i++) {

  text += `\nВремя простоя ${i}-го сервера ${servers[i].downTime.toFixed(2)}

секунд;\nКол-во программ поступивших на этот сервер

${servers[i].array.length};`

}

text += `\nКол-во необработанных программ: ${count}`

for (let i = 0; i <= n + m; i++) {

  text += `\nP[${i}] = ${(P[i] / time).toFixed(5)};`

}

text += `\nВероятность отказа: ${otkazP.toFixed(3)}

Относительная пропускная способность ВС: ${Q.toFixed(3)}

Абсолютная пропускная способность ВС: ${S.toFixed(3)}

Среднее число занятых серверов: ${K.toFixed(3)}

Среднее число программ в буфере: ${BoofN.toFixed(3)}

Среднее число программ в ВС: ${ProgN.toFixed(3)}

Среднее время нахождения программы в буфере: ${BoofT.toFixed(3)}

Среднее время нахождения программы в ВС: ${ProgT.toFixed(3)}`

  return (text);

}

*//поле вывода результата при линейном распределении*

let coclusionLine = document.querySelector('.conclusion-line');

*//поле вывода результата при экспоненциальном распределении*

let conclusionExponential = document.querySelector('.conclusion-exponential');

function parse(*str*) { *//функция парсинга данных из input*

  if (str.indexOf('/', 0) != -1) {

    const arr = str.split('/');

    return +arr[0] / +arr[1];

  } else {

    return + str;

  }

}

function getData() { *//функция получения данных из интерфейса*

  return ({

    N: + document.getElementById('N').value, *//кол-во серверов*

    M: + document.getElementById('M').value, *//кол-во мест в буфере*

    timework: parse(document.getElementById('timework').value),

    TzMIN: parse(document.getElementById('Tzmin').value),

    TzMAX: parse(document.getElementById('Tzmax').value),

    TsMIN: parse(document.getElementById('Tsmin').value),

    TsMAX: parse(document.getElementById('Tsmax').value),

    lambda: parse(document.getElementById('lambda').value),

    tobr: parse(document.getElementById('Tobr').value),

    Mu: 1 / parse(document.getElementById('Tobr').value),

  })

}

*//функция для получения данных о выборе закона распределения*

function checkDistribution() {

  const inp = document.getElementsByName('distribution');

  for (let i = 0; i < inp.length; i++) {

    if (inp[i].type === "radio" && inp[i].checked) {

      return inp[i].value;

    }

  }

}

function clickButton () { *//функция клика на кнопку*

  const data = getData();

  const method = checkDistribution();

  if (!data.N || !data.timework) { *//проверка ввода необходимых данных*

    alert ('Заполните поля N и время работы программы');

    return false;

  }

  let prog = [];

  const elem = { *//объект с данными*

    n: data.N,

    m: data.M

  }

  if (method === 'line') { *//в зависимости от выбранного закона распределения,*

*//хар-ки программ вычисляются по разному*

    prog = calcProgramsLine(data.TzMIN, data.TzMAX, data.TsMIN, data.TsMAX,

data.timework);

    elem.l = Math.pow((data.TzMIN + data.TzMAX) / 2, -1);

    elem.mu = Math.pow((data.TsMIN + data.TsMAX) / 2, -1);

  } else {

    if (!data.tobr || !data.lambda) { *//проверка ввода необходимых данных*

      alert ('Заполните поля λ и tобр');

      return false;

    }

    prog = calcProgramsExp(data.lambda, data.Mu, data.timework);

    elem.l = data.lambda;

    elem.mu = data.Mu;

  }

  let servers = creatServers(data.N); *//создание серверов*

  let conclusion = modeling(prog, servers, data.N, data.M, data.timework, elem.l, elem.mu); *//моделирование и расчет хар-к ВС*

  if (method === 'line') {

    coclusionLine.textContent = conclusion;

  } else {

    conclusionExponential.textContent = conclusion;

  }

}