Министерство науки и высшего образования Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Институт прикладной математики и компьютерных наук

Кафедра прикладной математики

**КУРСОВАЯ РАБОТА**

СТАТИСТИЧЕСКИЕ ЭКСПЕРИМЕНТЫ НА ИМИТАЦИОННОЙ МОДЕЛИ ПОЛУСИНХРОННОГО ПОТОКА СОБЫТИЙ ВТОРОГО ПОРЯДКА В СХЕМЕ С ПРОДЛЕВАЮЩИМСЯ МЕРТВЫМ ВРЕМЕНЕМ

Останин Роман Олегович

Руководитель работы

д-р физ.-мат. наук, профессор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Л.А. Нежельская

*подпись*

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Автор работы

студент 3 курса группы № 932120

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Р.О. Останин

*подпись*

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Томск – 2024

Оглавление

[Введение 3](#_Toc167827441)

[1. Постановка задачи 5](#_Toc167827442)

[2. Матрица инфинитезимальных характеристик 8](#_Toc167827443)

[3. Имитационное моделирование 11](#_Toc167827444)

[3.1. Основные понятия 11](#_Toc167827445)

[3.2. Метод обратных функций 11](#_Toc167827446)

[4. Блок-схема имитационного моделирования 12](#_Toc167827447)

[5. Статистические эксперименты 15](#_Toc167827448)

[Заключение 25](#_Toc167827449)

[Литература 26](#_Toc167827450)

[ПРИЛОЖЕНИЕ 1 27](#_Toc167827451)

[ПРИЛОЖЕНИЕ 2 51](#_Toc167827452)

# Введение

Первые задачи теории массового обслуживания (ТМО) были рассмотрены сотрудником Копенгагенской телефонной компании, датским ученым А.К. Эрлангом, в период между 1908 и 1922 годами. В своей статье, опубликованной в 1909 году [1], ученый использовал модель пуассоновского потока как математическую модель поступающих на телефонную станцию вызовов. Далее теорию потока однородных событий, которая легла в основу ТМО, разработал советский математик А. Я. Хинчин [2]. Также основы и фундаментальные результаты по ТМО изложены в трудах советских ученых Б. В. Гнеденко,

И. Н. Коваленко [3].

К задачам телефонного дела добавились различные задачи автоматизации и организации производства, эксплуатации аэропортов, железнодорожных станций, магазинов, пунктов проката, задачи в области компьютерных сетей, телекоммуникационные задачи. В связи с колоссальным ростом информационных потоков модель пуассоновского потока событий стала устаревать и терять адекватность отображения реальных потоков данных. На смену ему были разработаны дважды стохастические потоки событий, которые способны адекватно отображать реальные информационные потоки.

Впервые модель дважды стохастического потока была опубликована в 1955 году Д. Коксом [4]. В своей работе Д. Кокс рассматривал дважды стохастические потоки событий, интенсивность которых является непрерывным случайным процессом. Аналогичная модель потока рассмотрена в работе Дж. Кингмена в 1964 году [5]. В 1979 году в работах Г.П. Башарина, В.А. Кокотушкина и В. А. Наумова [6, 7], М. Ньютса [8] и в 1991 году в работе Д. Лукантони [9] были рассмотрены дважды стохастические потоки с интенсивностью, являющейся кусочно-постоянным случайным процессом с конечным числом состояний.

Систематизированный материал по дважды стохастическим потокам событий, интенсивность которых является кусочно-постоянным случайным процессом с конечным числом состояний, представлен в учебном пособии Л.А. Нежельской [10].

В данной курсовой работе исследована одна из таких моделей дважды стохастического потока событий, интенсивность которого является кусочно-постоянным случайным процессом с двумя состояниями. Построена имитационная модель полусинхронного потока событий второго порядка в схеме с продлевающимся мертвым временем фиксированной длительности и приведен ряд статистических экспериментов с целью получения результатов, численно доказывающих адекватность построенной модели.

Впервые математическая модель полусинхронного потока событий второго порядка как в условиях полной наблюдаемости, так и при наличии непродлевающегося мертвого времени была рассмотрена в диссертационной работе Д. Тумашкиной [11].

В данной работе рассмотрена модель потока, функционирующего в условиях продлевающегося мертвого времени. В связи с этим в курсовой работе присутствует новизна.

# Постановка задачи

Рассматривается полусинхронный поток событий второго порядка с двумя состояниями, сопровождающий случайный процесс которого  является кусочно-постоянным ненаблюдаемым процессом с двумя состояниями: . Когда , то имеет место *i*-е состояние (),  процесса (потока). Обязательное условие, налагаемое на процесс .

Длительность интервала между событиями потока в 1-ом состоянии определяется случайной величиной, где случайная величина  имеет функцию распределения , случайная величина  имеет функцию распределения , ,  независимые случайные величины. В момент наступления события потока процесс  переходит из 1-го во 2-е состояние либо с вероятностью, либо с вероятностьюв зависимости от того, какая из случайных величин ,  приняла минимальное значение. Также в момент наступления события потока процесс  остается в 1-ом состоянии либо с вероятностью, либо с вероятностью в зависимости от того, какая из случайных величин ,  приняла минимальное значение. Вероятности по каждой из случайных величин удовлетворяют условиям:, . Длительность пребывания процесса  в 1-ом состоянии является случайная величина с функцией распределения  [11].

Длительность пребывания процесса  во 2-ом состоянии является случайной величиной с функцией распределения. В течение времени пребывания процесса  во 2-ом состоянии имеет место пуассоновский поток событий с параметром . Переход процесса  из 2-го состояния в 1-е происходит в произвольный момент времени, не связанный с моментом времени наступления событий пуассоновского потока с параметром .

Рассмотрим полусинронный поток событий второго порядка при его неполной наблюдаемости. К условиям математической модели, изложенной выше, добавляются условия функционирования потока в схеме с продлевающимся мертвым временем. Схема формирования наблюдаемого потока событий при наличии продевающегося мертвого времени определяется следующим образом.

После каждого зарегистрированного в момент времени события исходного потока, наступает период мертвого времени фиксированной длительности *T*, в течение которого каждое последующее пришедшее событие не наблюдается, однако продлевает период ненаблюдаемости исходного потока на фиксированную длительность *T*. Таким образом, поскольку событие наступает в случайный момент времени, то период ненаблюдаемости является случайной величиной. После окончания периода ненаблюдаемости первое наступившее событие снова создает период мертвого времени длительности *T*, в течение которого каждое последующее пришедшее событие не наблюдается и продлевает его на фиксированный период длительности *T* и т.д. Для наглядности на рисунке 1 приведен пример возникающей ситуации, где − моменты наступления событий в наблюдаемом потоке; формирование периодов мертвого времени длительности *T* обозначено заштрихованными серыми прямоугольниками; общий период ненаблюдаемости обозначен черным закрашенным прямоугольником; черными закрашенными кружками обозначены события полусинхронного потока второго порядка, недоступные для наблюдения.

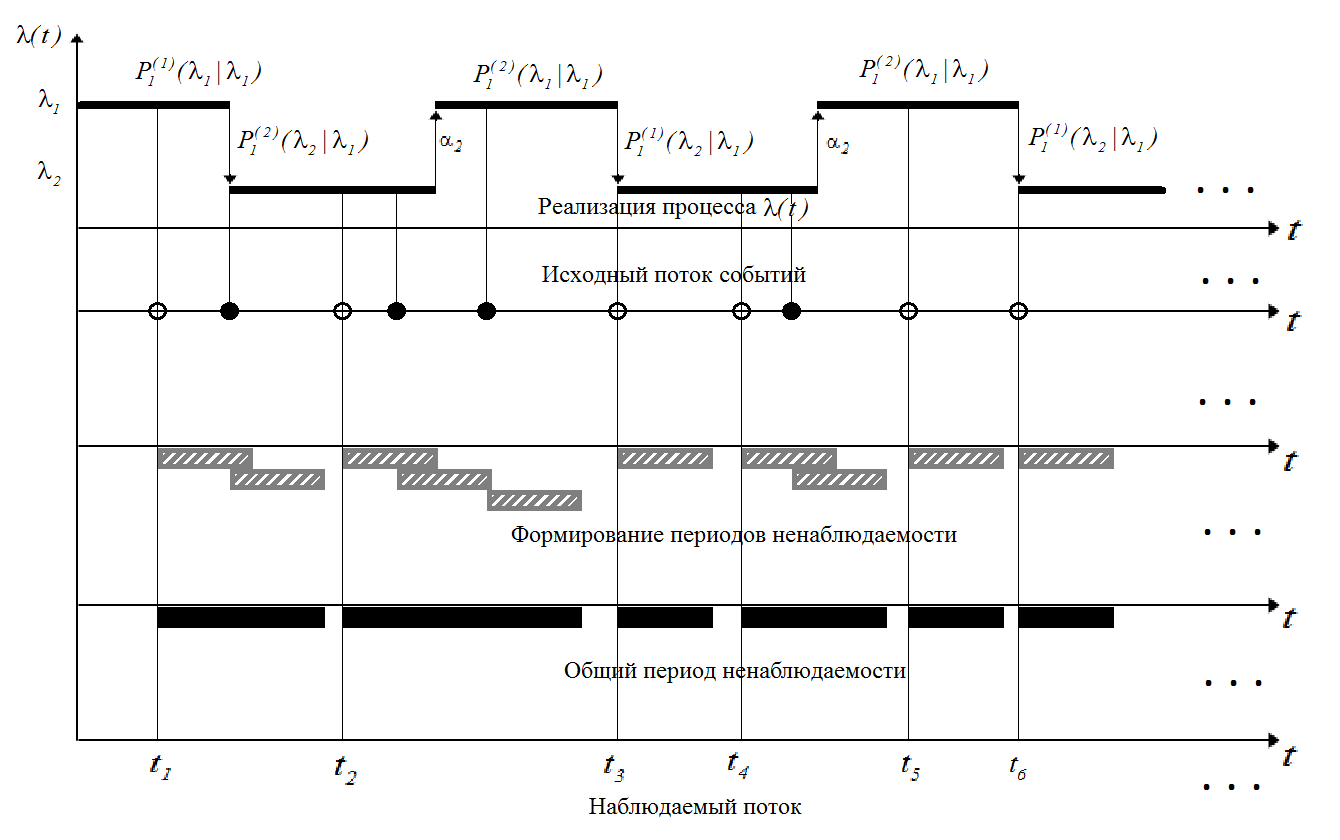


Рис. 1. Формирование наблюдаемого потока событий

На первой временной оси представлена реализация процесса ; на второй – моменты наступления событий исходного полусинхронного потока событий второго порядка; на третьей – ступенчатое формирование периодов ненаблюдаемости; на четвертой – общий период ненаблюдаемости; на пятой – моменты наступления событий наблюдаемого потока.

*Цель данной курсовой работы:*

1. Построить имитационную модель полусинхронного потока событий второго порядка.

*Задачи данной курсовой работы:*

1. Изучение литературы по теории массового обслуживания, по теории дважды стохастических потоков событий и имитационному моделированию;
2. построение матрицы инфинитезимальных характеристик процесса ;
3. построение блок-схемы имитационного моделирования;
4. постановка статистических экспериментов и анализ полученных результатов для подтверждения правильности работы построенной модели.

# Матрица инфинитезимальных характеристик

Рассмотрим на временной оси *t* полуинтервал (рис.2).

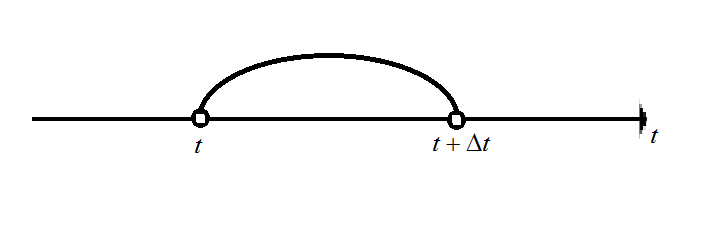


Рис. 2. Полуинтервал.

Опишем все возможные события, которые могут произойти на данном полуинтервале:

А) Пусть процесс  в момент времени *t* находится в 1-ом состоянии, . Возможны следующие ситуации:

1. на полуинтервале  наступит событие потока с вероятностью , и не наступит событие потока с вероятностью , и в момент наступления события потока процесс  останется в 1-ом состоянии с вероятностью ; вероятность этого сложного события равна

;

1. на полуинтервале  не наступит событие потока с вероятностью, равной, и наступит событие потока с вероятностью , и в момент наступления события потока процесс  останется в 1-ом состоянии с вероятностью; вероятность этого сложного события равна

;

Тогда вероятность ситуации – наступление события и переход процесса  из 1-го состояния в 1-ое равна

;

1. аналогично находится вероятность ситуации – наступление события потока и переход процесса  из 1-го состояния во 2-ое

;

1. на полуинтервале  не наступит событие потока с вероятностью  и не наступит событие потока с вероятностью ; вероятность этого сложного события равна

;

Другие возможные ситуации, связанные с нахождением процесса  на полуинтервале  в 1-ом состоянии, имеют вероятность .я

Б) Пусть процесс  в момент времени *t* находится во 2-ом состоянии, . Возможны следующие ситуации:

1) на полуинтервале  2-ое состояние процесса  не закончится с вероятностью, равной , и наступит событие пуассоновского потока с вероятностью ; вероятность этого сложного события равна

;

2) на полуинтервале  2-ое состояние процесса  закончится с вероятностью, равной , и не наступит событие пуассоновского потока с вероятностью ; вероятность этого сложного события равна

;

3) на полуинтервале  2-ое состояние процесса  не закончится с вероятностью, равной , и не наступит событие пуассоновского потока с вероятностью ; вероятность этого сложного события равна

.

Другие возможные ситуации, связанные с нахождением процесса  на полуинтервале  во 2-ом состоянии, имеют вероятность .

Тогда можно определить инфинитезимальные характеристики процесса :

1. интенсивность перехода процесса  из 1-го состояния в 1-ое с наступлением события потока выглядит следующим образом:

;

1. интенсивность перехода процесса  из *1*-го состояния во *2*-ое с наступлением события потока есть

;

1. интенсивность перехода процесса  из 2-го состояния в 1-ое с наступлением события потока есть

;

1. интенсивность перехода процесса  из 2-го состояния во 2-ое с наступлением события потока есть

;

1. интенсивность перехода процесса  из 1-го состояния в 1-ое без наступления события потока есть

;

1. интенсивность перехода процесса  из 1-го состояния во 2-ое без наступления события потока есть

;

1. интенсивность перехода процесса  из 2-го состояния в 1-ое без наступления события потока есть

;

1. интенсивность перехода процесса  из 2-го состояния во 2-ое без наступлением события потока есть

.

Вычислив интенсивности переходов процесса  из состояния в состояние , можно выписать матрицы инфинитезимальных характеристик  и процесса :

,

.

Элементами матрицы  являются интенсивности переходов процесса  из состояния в состояние с наступлением события. Диагональные элементы матрицы  – интенсивности выхода процесса  из состояний без наступления события, взятые с противоположным знаком. Недиагональные элементы матрицы  – интенсивности переходов из состояния в состояние без наступления событий потока.

# Имитационное моделирование

## 3.1. Основные понятия

Имитационное моделирование – разработка компьютерных моделей и постановка экспериментов на них. Целью моделирования в конечном счете является принятие обоснованных, целесообразных управленческих решений.

Имитационная модель – это компьютерная программа, которая описывает структуру и воспроизводит поведение реальной системы во времени. Имитационная модель позволяет получать подробную статистику о различных аспектах функционирования системы в зависимости от входных данных.

У имитационной модели, как у метода исследования какой-либо реальной системы, есть множество преимуществ перед другими методами. Из них можно выделить универсальность, наглядность, способность к множественному воспроизведению для постановки статистических экспериментов.

## 3.2. Метод обратных функций

Имитационная модель исследуемого потока событий построена с помощью метода обратных функций. В общем случае метод обратных функций работает таким образом. Пусть – значение длительности пребывания процесса  в состоянии. Случайная величина  – длительность пребывания процесса  в состоянии – имеет функцию распределения . В соответствии с методом обозначим Имеем , тогда , откуда находим формулу моделирования значений длительности интервалов пребывания процесса  в состоянии:

;

где  – равномерно распределенная на интервале  случайная величина,  – параметры потока.

# Блок-схема имитационного моделирования

На рис. 3 представлена блок-схема алгоритма имитационного моделирования полусинхронного потока событий второго порядка в схеме с продлевающимся мертвым временем.

При построении блок-схемы введены обозначения:

1. Tm – время моделирования (время наблюдения за потоком);
2. параметры L1 – , L2 – , a1 – , a2 – ;
3. вероятности P.1.1 – , P.1.2 – , P.2.1 – ,P.2.2 – ;
4. RAN(0, 1) – датчик случайных чисел равномерно распределенных на интервале [0, 1);
5. L – текущее состояние потока;
6. t – текущее время моделирования;
7. ty – длительность пребывания процесса  в 1-ом и 2-ом состоянии, ty1, ty2 – периоды между событиями в 1-ом состоянии по 1-ой и 2-ой случайной величине соответственно, ty.s – период между событиями пуассоновского потока с параметром ;
8. SV – случайная величина, по которой реализовался минимальный период между событиями в 1-ом состоянии;
9. td.1, td.2 – правая граница периода ненаблюдения в 1-ом и 2-ом состоянии соответственно;
10. T2 – дополнительная переменная значения времени моделирования во 2-ом состоянии;

Результатом алгоритма является вектор  моментов наступления событий полусинхронного потока второго порядка в схеме с продлевающимся мертвым временем, где .

*Описание блоков, представленных на рис. 3:*

Блок 1. Начало алгоритма, инициализация параметров.

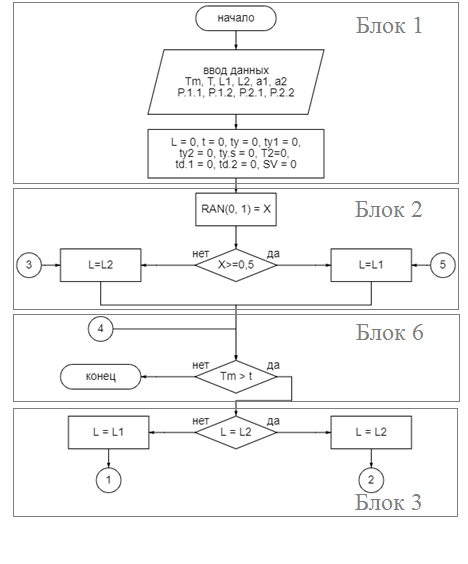
Блок 2. Розыгрыш начального состояния процесса .

Блок 3. Переключение имитационной модели между состояниями процесса .

Блок 4. Генерация длительности интервала между состояниями потока в 1-ом состоянии процесса ;

Блок 5. Генерация длительности интервала между состояниями потока во 2-ом состоянии процесса ;

Блок 6. Завершение алгоритма по достижении значения .



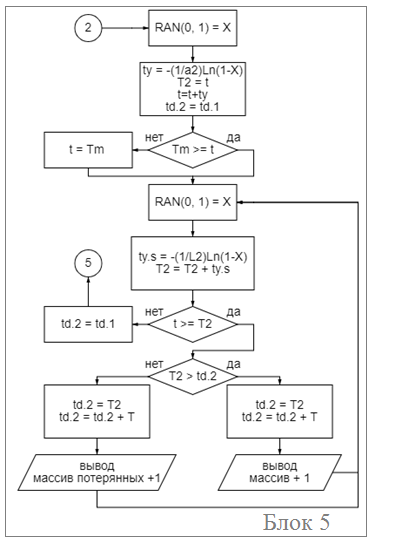
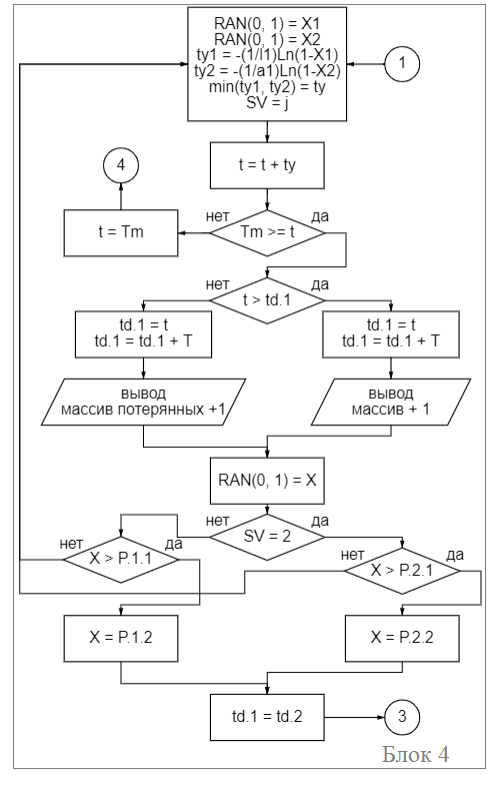


Рис. 3. Блок-схема имитационного моделирования полусинхронного потока событий второго порядка в схеме с продлевающимся мертвым временем

# Статистические эксперименты

ХАРАКТЕРИСТИКИ ПОТОКА

Введем случайные величины:

**** – случайная величина − длительность интервала между наблюдаемыми событиями;

 – случайная величина − длительность периода ненаблюдаемости потока;

 – случайная величина − длительность пребывания процесса  в 1-ом состоянии;

 – случайная величина − длительность пребывания процесса  в 2-ом состоянии.

Проводится  реализаций. В *j*-ой реализации,, имеем *l* интервалов между событиями: . Вычисляется значение оценки среднего случайной величины  для *j*-ой реализации: . Затем рассчитывается оценка среднего случайной величины **** по всем *N* реализациям: .

Аналогично находится значение оценки среднего для , ,  в *j*-ой реализации:

, , .

Затем вычисляется оценка среднего , ,  по всем *N* реализациям:

, , .

Характеристиками потока будем считать:

 – оценка среднего случайной величины – длительности интервала между событиями в наблюдаемом потоке;

 – оценка среднего случайной величины –периода ненаблюдаемости потока;

 – оценка среднего случайной величины – длительности пребывания в 1-ом состоянии;

 – оценка среднего случайной величины – длительности пребывания во 2-ом состоянии.

ЭКСПЕРИМЕНТ 1. Проверка схемы работоспособности модели в случае полной наблюдаемости потока.

Проводится одна реализация имитационной модели. Задаются параметры потока,  и . Тогда в данной реализации количество моментов наступлений событий исходного потока и наблюдаемого потока должно совпадать.

В таблице 1 представлены исходные данные для эксперимента 1. В таблице 2 представлены: в первой строке – номер момента наступления события исходного потока, во второй – моменты наступления событий исходного потока и в третьей – моменты наступления событий наблюдаемого потока.

Таблица 1. Исходные данные для эксперимента 1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *T* |  |  |  |  |  |  |  |  |
| 10 | 0 | 1,1 | 0,5 | 0,7 | 0,8 | 0,4 | 0,6 | 0,6 | 0,4 |

Таблица 2. Результаты эксперимента 1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  | 0,9608 | 1,0852 | 3,6355 | 4,8507 | 5,1786 | 6,1987 | 7,4506 | 8,7526 | 9,0387 |
|  | 0,9608 | 1,0852 | 3,6355 | 4,8507 | 5,1786 | 6,1987 | 7,4506 | 8,7526 | 9,0387 |

ВЫВОД. Исходя из таблицы 2 видно, что все моменты наступления событий исходного потока отразились в наблюдаемом потоке; это подтверждает наше предположение.

ЭКСПЕРИМЕНТ 2. Проверка схемы формирования мертвого времени.

Проводится одна реализация имитационной модели. Задаются параметры потока, и . В данной реализации некоторые события исходного потока не отражаются в наблюдаемом потоке в силу наличия мертвого времени .

В таблице 3 представлены исходные данные для эксперимента 2. В таблице 4 представлены: в первой строке – номер момента наступления события исходного потока, во второй – моменты наступления событий исходного потока и в третьей – моменты наступления событий наблюдаемого потока.

Таблица 3. Исходные данные для эксперимента 2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *T* |  |  |  |  |  |  |  |  |
| 10 | 0,5 | 1,1 | 0,5 | 0,7 | 0,8 | 0,4 | 0,6 | 0,6 | 0,4 |

Таблица 4. Результаты эксперимента 2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  | 0,4806 | 1,5355 | 2,2084 | 2,3752 | 4,4302 | 5,2301 | 5,7907 | 7,2907 | 9,0121 |
|  | 0,4806 | 1,5355 | 2,2084 |  | 4,4302 | 5,2301 |  | 7,2907 | 9,0121 |

ВЫВОД. Исходя из таблицы 4 видно, что некоторые моменты наступления событий исходного потока не отразились в наблюдаемом потоке.

По результатам экспериментов 1 и 2 можно сказать, что построена работоспособная имитационная модель, и схема формирования мертвого времени в модели корректна.

ЭКСПЕРИМЕНТ 3. Установление стационарного режима.

Изменяется время моделирования  от 100 до 4000 с шагом 100. Фиксируются параметры потока. Рассматривается изменение характеристик потока. Исходные данные для эксперимента 3 представлены в таблице 5; численные результаты − в таблице 6.

Таблица 5. Исходные данные для эксперимента 3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *T* |  |  |  |  |  |  |  |  |
| 0,1 | 1,5 | 0,9 | 1,1 | 0,2 | 0,3 | 0,7 | 0,4 | 0,6 |

Таблица 6. Результаты эксперимента 3

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 100 | 200 | 300 | … | 2900 | 3000 | … | 3900 | 4000 |
|  | 1,0403 | 1,0203 | 1,0322 | … | 1,0306 | 1,0305 | … | 1,0268 | 1,0291 |
|  | 0,1055 | 0,1052 | 0,1053 | … | 0,1055 | 0,1054 | … | 0,1054 | 0,1054 |
|  | 0,5752 | 0,5872 | 0,5770 | … | 0,5854 | 0,5872 | … | 0,5858 | 0,5802 |
|  | 5,1510 | 4,9356 | 5,0463 | … | 5,0029 | 5,0004 | … | 4,9732 | 4,9801 |

На рисунках 4−7 представлены графики зависимостей вычисляемых характеристик потока , , ,  от времени моделирования .

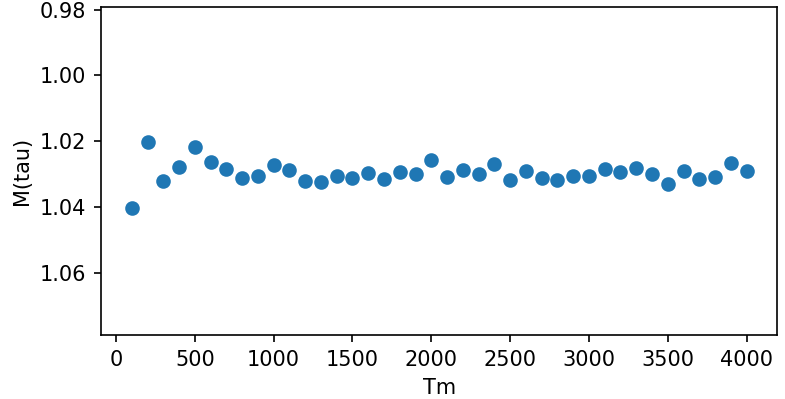


Рис. 4. График зависимости  от 

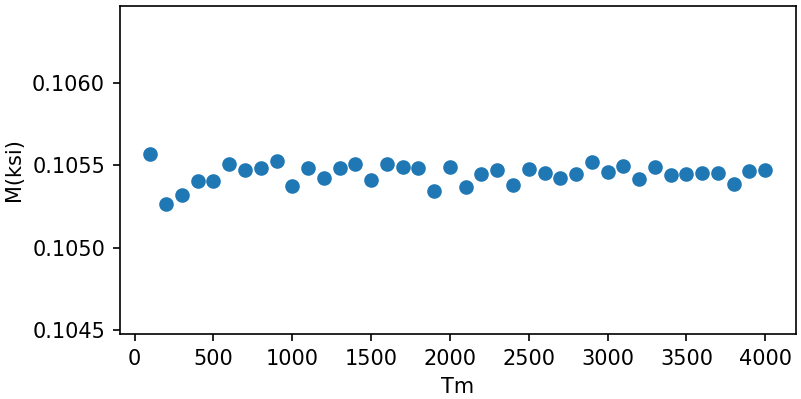


Рис. 5. График зависимости  от 

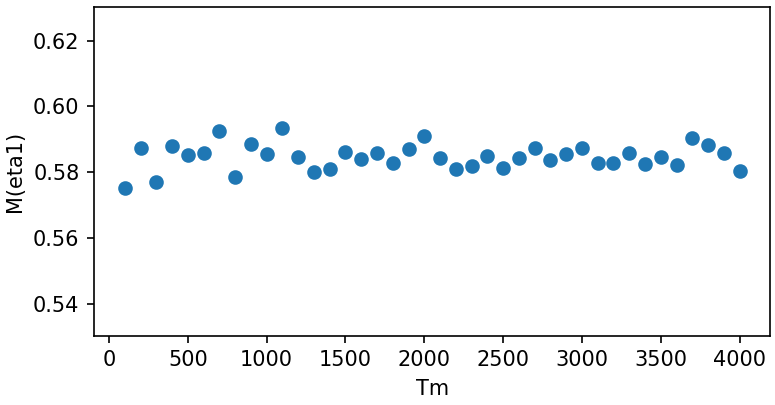


Рис. 6. График зависимости  от 

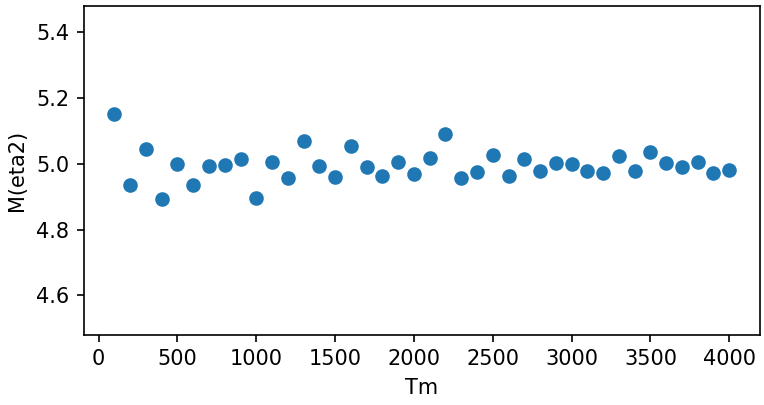


Рис. 7. График зависимости  от 

ВЫВОД. По численным и графическим результатам видно установление стационарного режима при времени моделирования . В дальнейших экспериментах будет использоваться значение времени моделирования 

ЭКСПЕРИМЕНТ 4. Зависимость характеристик потока от длительности мертвого времени *T.*

Изменяется длительность мертвого времени *T* от 0,1 до 1,5 с шагом 0,1. В каждой реализации задается . Фиксируются параметры потока. Исследуется изменение характеристик потока в зависимости от *T*. Исходные данные для эксперимента4 представлены в таблице 7; численные результаты − в таблице 8.

Таблица 7. Исходные данные для эксперимента 4

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| 3000 | 1,5 | 0,9 | 1,1 | 0,2 | 0,3 | 0,7 | 0,4 | 0,6 |

Таблица 8. Результаты эксперимента 4

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *T* | 0,1 | 0,2 | 0,3 | … | 0,8 | 0,9 | … | 1,3 | 1,4 | 1,5 |
|  | 1,0283 | 1,1425 | 1,4191 | … | 2,1771 | 2,4380 | … | 3,7581 | 4,1974 | 4,6929 |
|  | 0,1054 | 0,2225 | 0,3532 | … | 1,2675 | 1,5206 | … | 2,4590 | 3,2843 | 3,7799 |
|  | 0,5830 | 0,5864 | 0,5843 | … | 0,5820 | 0,5829 | … | 0,5839 | 0,5892 | 0,5866 |
|  | 5,0235 | 4,9966 | 4,9906 | … | 5,0083 | 4,9805 | … | 5,0080 | 5,0047 | 5,0358 |

На рисунках 8−11 представлены графики зависимостей вычисляемых характеристик потока , , ,  от длительности мертвого времени *T*.

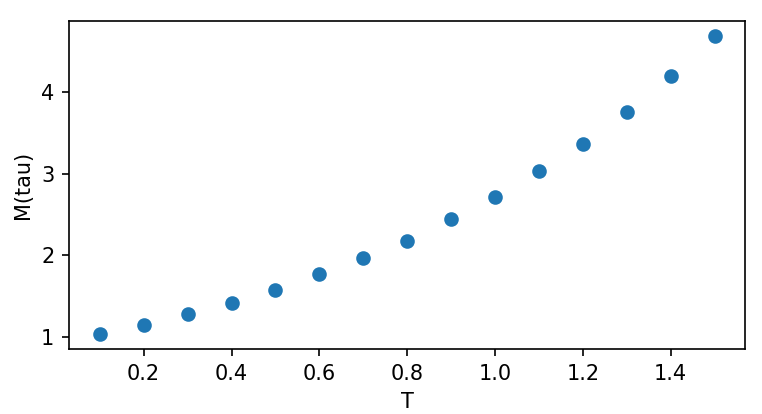


Рис. 8. График зависимости  от *T*

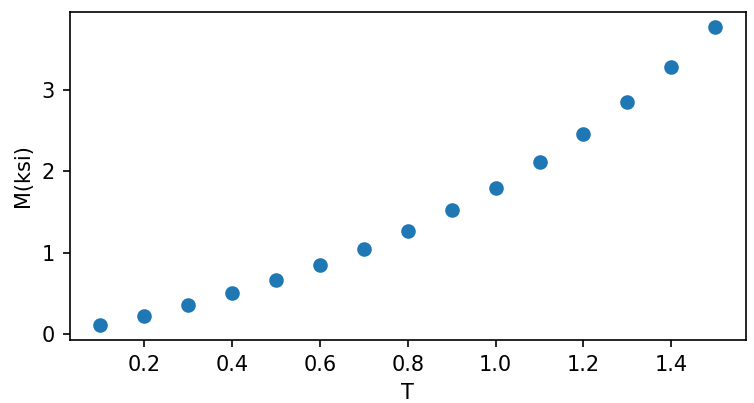


Рис. 9. График зависимости  от *T*

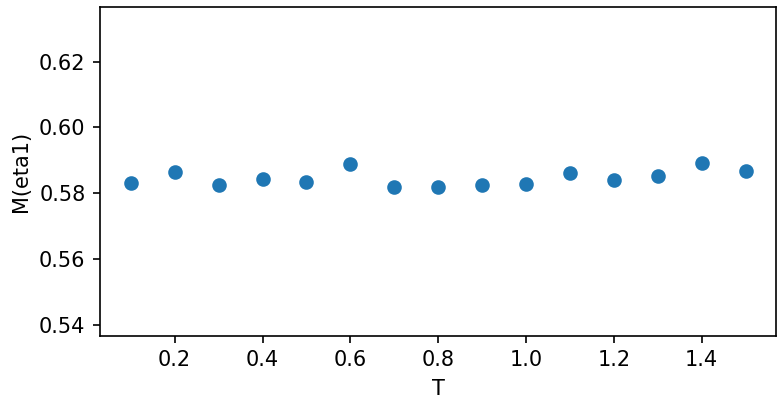


Рис. 10. График зависимости  от *T*

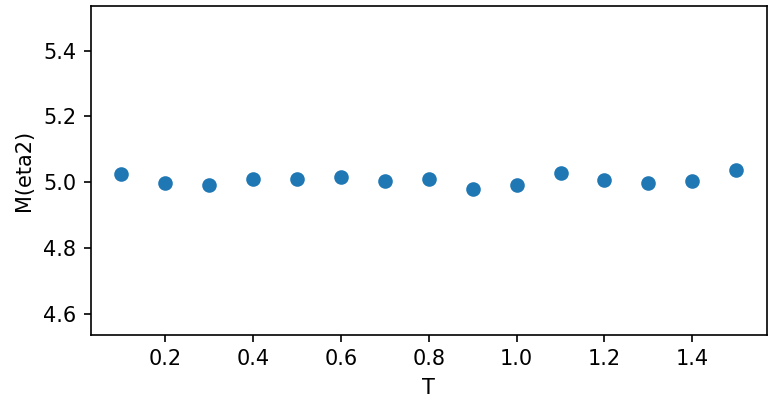


Рис. 11. График зависимости  от *T*

ВЫВОД. Из численных и графических результатов видно, что с ростом мертвого времени растет значение  и . Рост мертвого времени влечет за собой рост периода ненаблюдаемости, поэтому приходящие события чаще не наблюдаются и, соответственно, значение длительности интервала между событиями увеличивается. Далее можно заметить, что значение  и  почти не изменяются. Стационарность этих оценок объяснима тем, что они не зависят от значения мертвого времени.

ЭКСПЕРИМЕНТ 5. Зависимость  от интенсивности .

Приведем два случая: в *первом случае* увеличивается вероятность перехода процесса из 1-го во 2-ое состояние по обеим случайным величинам, а во *втором случае* увеличивается вероятность остаться в 1-ом состоянии по обеим случайным величинам. В обоих случаях задается ; исходные данные для эксперимента 4 зафиксированы в таблице 9 (случай 1) и в таблице 11 (случай 2). Интенсивность  изменяется от 0,5 до 5 с шагом 0,5. Исследуется поведение характеристики. Численные результаты *первого и второго случаев* представлены в таблице 10 и таблице 12 соответственно.

На рисунках 12 (случай 1), 13 (случай 2) представлены графики зависимостей характеристики потока  от значения параметра .

Таблица 9. Исходные данные для эксперимента 5 (случай 1)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *T* |  |  |  |  |  |  |  |
| 3000 | 0,1 | 0,2 | 1,1 | 0,2 | 0,3 | 0,7 | 0,3 | 0,7 |

Таблица 10. Результаты эксперимента 5 (случай 1)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0,5 | 1,0 | 1,5 | … | 3,5 | 4,0 | 4,5 | 5,0 |
|  | 0,8950 | 0,6789 | 0,5494 | … | 0,3131 | 0,2793 | 0,2553 | 0,2346 |

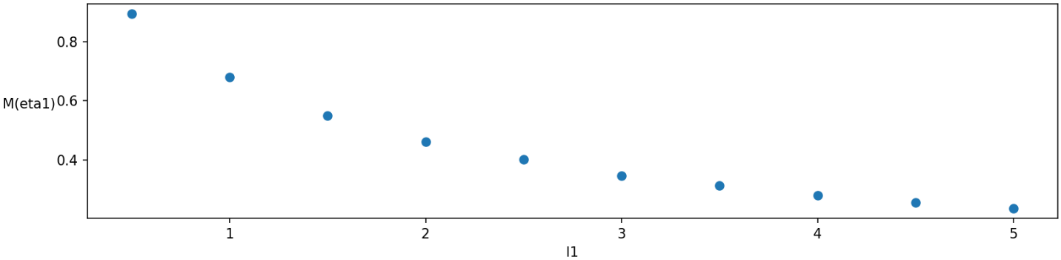


Рис. 12. График зависимости  от  (случай 1)

Таблица 11. Исходные данные для эксперимента 5 (случай 2)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *T* |  |  |  |  |  |  |  |
| 3000 | 0,1 | 0,2 | 1,1 | 0,2 | 0,7 | 0,3 | 0,7 | 0,3 |

Таблица 12. Результаты эксперимента 5 (случай 2)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0,5 | 1,0 | 1,5 | … | 3,5 | 4,0 | 4,5 | 5,0 |
|  | 2,0658 | 1,5897 | 1,2765 | … | 0,7210 | 0,6545 | 0,5961 | 0,5469 |

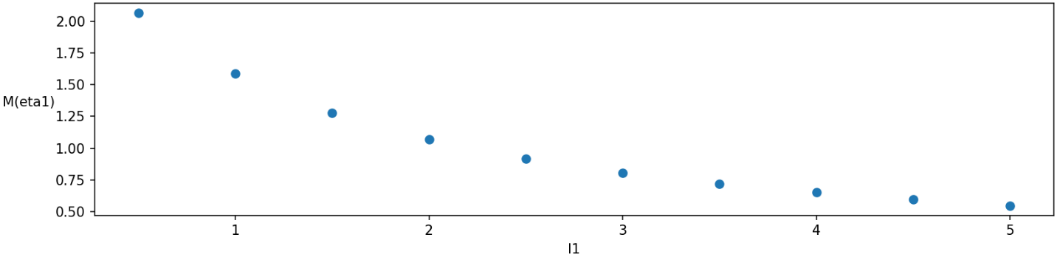


Рис. 13. График зависимости  от  (случай 2)

ВЫВОД. Из численных и графических результатов видно, что при одинаковых значениях интенсивности  значение  различно. Это связано с тем, что в *первом случае* вероятность перейти во 2-ое состояние по обеим случайным величинам выше и, соответственно, с приходом события потока в первом состоянии процесс  чаще уходит во 2-ое состояние. Во *втором случае* вероятность остаться в 1-ом состоянии по обеим случайным величинам выше и, соответственно, с приходом события процесс  чаще остается в 1-ом состоянии, поэтому значение  выше. Однако, в обоих случаях с ростом интенсивности  наблюдается уменьшение значения. Это связано с тем, что при увеличении интенсивности  учащается поток событий в 1-ом состоянии, и розыгрыш уйти или остаться процессу  происходит чаще. А так как в *первом случае* вероятность уйти выше, а во *втором случае* она меньше, но все же есть, то процесс уходит из 1-го состояния.

ЭКСПЕРИМЕНТ 6. Зависимость  от интенсивности .

Приведем два случая. Пусть в *первом случае* интервал изменения  от 0,1 до 1 с шагом 0,1, а во *втором* – от 0,5 до 5 с шагом 0,5. В обоих случаях задаем , фиксируем параметры. Рассматриваем изменение. Численные результаты *первого* и *второго случаев* представлены, соответственно, в таблице 14 и таблице 16.

Таблица 13. Исходные данные для эксперимента 6 (случай 1)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *T* |  |  |  |  |  |  |  |
| 3000 | 0,1 | 1,5 | 0,2 | 1,1 | 0,4 | 0,6 | 0,6 | 0,4 |

Таблица 14. Результаты эксперимента 6 (случай 1)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0,01 | 0,02 | 0,03 | 0,04 | 0,05 | 0,06 | 0,07 | 0,08 | 0,09 | 0,1 |
|  | 100,43 | 49,45 | 33,10 | 24,95 | 20,04 | 16,76 | 14,29 | 12,73 | 11,23 | 9,96 |

На рисунках 14 (случай 1), 15 (случай 2) представлены графики зависимостей характеристики потока  от значения параметра .

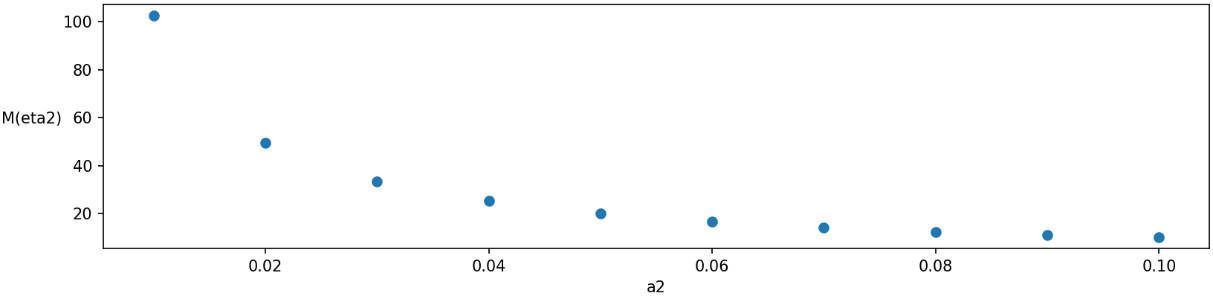


Рис. 14. График зависимости  от  (случай 1)

Таблица 15. Исходные данные для эксперимента 6 (случай 2)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *T* |  |  |  |  |  |  |  |
| 3000 | 0,1 | 1,5 | 0,2 | 1,1 | 0,4 | 0,6 | 0,6 | 0,4 |

Таблица 16. Результаты эксперимента 6 (случай 2)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0,5 | 1,0 | 1,5 | 2,0 | 2,5 | 3,0 | 3,5 | 4,0 | 4,5 | 5,0 |
|  | 2,006 | 1,002 | 0,665 | 0,501 | 0,400 | 0,333 | 0,286 | 0,249 | 0,223 | 0,199 |

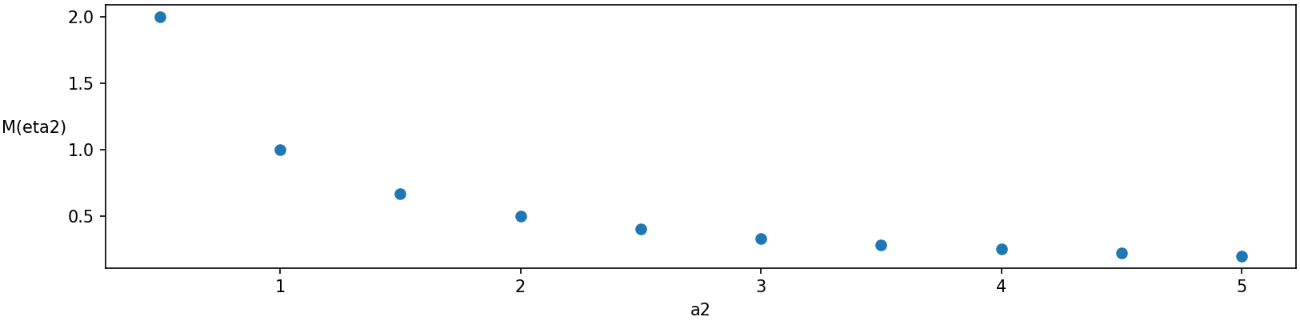


Рис. 15. График зависимости  от  (случай 1)

ВЫВОД. Из численных и графических результатов видно, что с увеличением значения интенсивности  значение  в обоих случаях уменьшается. Это связано с тем, что интенсивность  влияет на длительность интервала пребывания процесса  во 2-ом состоянии. Также видно, что от разных интервалов изменения значений интенсивности  меняется интервал изменения значения .

# Заключение

В данной курсовой работе исследован полусинхронный поток событий второго порядка в схеме с продлевающимся мертвым временем. В ходе работы построена математическая модель потока и построена матрица инфинитезимальных характеристик процесса .

Построена имитационная модель данного потока в виде программного кода на языке программирования Python с использованием различных библиотек (numpy, tkinter, matplotlib и т.д.).

Анализ корректности результатов проведенных статистических экспериментов 1−6 дает возможность утверждать, что получена работоспособная модель, непротиворечащая математической модели потока.

Все задачи, поставленные в данной курсовой работе, выполнены. Код программы, моделирующий полусинхронный поток событий второго порядка в схеме с продлевающимся мертвым временем, приведен в приложении 1. Результаты работы программы, реализующей созданную имитационную модель потока, представлены в приложении 2 в виде скриншота программы.

# Литература

1. Erlang A. K. The theory of probabilities and telephone conversations // Nyt Tidsskrift for Matematik. Seria B. 1909. Vol. 20, is. B. P. 33–39.
2. Хинчин А. Я. Работы по математической теории массового обслуживания. M. : Физматгиз, 1963. 236 c.
3. Гнеденко Б. В., Коваленко И. Н. Введение в теорию массового обслуживания. Изд. 4-е, испр. М. : Изд-во ЛКИ, 2007. 400 с.
4. Cox D.R. The analysis of non–Markovian stochastic processes by the inclusion of supplementary variables // Proceedings of the Cambridge Philosophical Society. – 1955. – Vol. 51, No. 3. – P. 433–441.
5. Kingman J. F. C. On doubly stochastic Poisson process // Proceedings of the Cambridge Philosophical Society. – 1964. – Vol. 60, is. 4. – P. 923–930.
6. Башарин Г. П. О методе эквивалентных замен расчета фрагментов сетей связи / Г. П. Башарин, В. А. Кокотушкин, В. А. Наумов // Изв. АН СССР. Техн. кибернетика. – 1979. – Ч. 1, № 6. – С. 92–99.
7. Башарин Г. П. О методе эквивалентных замен расчета фрагментов сетей связи / Г. П. Башарин, В. А. Кокотушкин, В. А. Наумов // Изв. АН СССР. Техн. кибернетика. – 1980. – Ч. 2, № 1. – С. 55–61.
8. Neuts M.F. A versatile Markov point process // Journal of Applied Probability. – 1979. – Vol. 16. – P. 764–779
9. Lucantoni D. M. New results on the single server queue with a batch Markovian arrival process // Communications in Statistics. Stochastic Models. – 1991. – Vol. 7, No. 1. – P. 1–46.
10. Нежельская Л.А. Оценка состояний дважды стохастических потоков событий : учебное пособие. – Томск : Издательство Томского государственного университета, 2020. – 210 с.
11. Тумашкина Д. Оценка состояний, длительности мертвого времени и параметров распределения в полусинхронном потоке событий второго порядка // дис. … канд. физ.-мат. наук: 05.13.01. – Томск, 2021. – 154 с.
12. Эльберг, М. С. Имитационное моделирование : учеб. пособие // М. С. Эльберг, Н. С. Цыганков. – Красноярск : Сиб. федер. ун-т, 2017. – 128 с.
13. Численные мотоды Монте-Карло, И.М. Соболь. Главная редакция физико-математической литературы изд-ва «Наука», 1973.

# ПРИЛОЖЕНИЕ 1

Код программы, реализующей имитационную модель потока

import math

import numpy

def potokSimulator(tm, t, L1, L2, A1, A2, p11, p12, p21, p22):

Tm = tm

T = t

P11 = p11

P12 = p12

P21 = p21

P22 = p22

L = 0

l1 = L1

l2 = L2

a1 = A1

Lj = [l1, a1]

a2 = A2

t = 0

td1 = 0

td2 = 0

MassInitial = [ ]

TempListIntervS1 = [ ] # интервалы в 1-м состоянии

MassIntervS = [ ] # интервалы в состояниях

Interv\_s1 = 0

Interv\_s2 = 0

x = numpy.random.rand(1)[0]

if (x >= 0.5):

L = l1

else:

L = l2

i = 0

while (Tm > t):

if (L == l1):

x1 = numpy.random.rand(1)[0]

x2 = numpy.random.rand(1)[0]

ty1 = -(1 / Lj[0]) \* math.log(1 - x1)

ty2 = -(1 / Lj[1]) \* math.log(1 - x2)

if ty1 > ty2:

ty = ty2

SV = 2

else:

ty = ty1

SV = 1

t = t + ty

TempListIntervS1.append([ty, SV])

if (Tm < t):

TempListIntervS1[len(TempListIntervS1) - 1][0] = Tm - (t - ty)

Interv\_s1 = 0

for i in range(len(TempListIntervS1)):

Interv\_s1 = Interv\_s1 + TempListIntervS1[i][0]

TempList = [Interv\_s1, 1, TempListIntervS1]

MassIntervS.append(TempList)

TempListIntervS1 = [ ]

break

if (t > td1):

td1 = t

td1 = td1 + T

TempList = [t, 'O', 1, SV]

MassInitial.append(TempList)

else:

td1 = t

td1 = td1 + T

TempList = [t, 'M', 1, SV]

MassInitial.append(TempList)

x = numpy.random.rand(1)[0]

if (SV == 2):

if (x > P21):

L = l2

td2 = td1

Interv\_s1 = 0

for i in range(len(TempListIntervS1)):

Interv\_s1 = Interv\_s1 + TempListIntervS1[i][0]

TempList = [Interv\_s1, 1, TempListIntervS1]

MassIntervS.append(TempList)

TempListIntervS1 = [ ]

else:

L = l1

else:

if (x > P11):

L = l2

td2 = td1

Interv\_s1 = 0

for i in range(len(TempListIntervS1)):

Interv\_s1 = Interv\_s1 + TempListIntervS1[i][0]

TempList = [Interv\_s1, 1, TempListIntervS1]

MassIntervS.append(TempList)

TempListIntervS1 = [ ]

else:

L = l1

else:

if (L == l2):

x = numpy.random.rand(1)[0]

T2 = t

ty = -(1 / a2) \* (math.log(1 - x))

t = t + ty

if (Tm < t):

Interv\_s2 = Tm - (t - ty)

t = Tm

TempList = [Interv\_s2, 2]

MassIntervS.append(TempList)

else:

Interv\_s2 = ty

TempList = [Interv\_s2, 2]

MassIntervS.append(TempList)

while (L == l2):

x = numpy.random.rand(1)

tys = -(1 / l2) \* (math.log(1 - x))

T2 = T2 + tys

if (t > T2):

if (T2 > td2):

td2 = T2

td2 = td2 + T

TempList = [T2, 'O', 2]

MassInitial.append(TempList)

else:

td2 = T2

td2 = td2 + T

TempList = [T2, 'M', 2]

MassInitial.append(TempList)

else:

L = l1

td1 = td2

A = [MassIntervS, MassInitial]

return A

Код программы реализующей графический интерфейс

from tkinter import \*

def potokArt(h, w, kt, a):

A = potokSimulator(a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7], a[8], a[9])

MassIntervS = A[0]

MassInitial = A[1]

H = h

W = w

Kt = kt

Window = Tk()

Window.title('Графика')

left = 30

right = 10

down = 20

up = 10

line\_1 = h\*(1/5)

line\_2 = h\*(2/5)

line\_3 = h\*(3/5)

line\_4 = h\*(4/5)

line\_5 = h-20

c = Canvas(Window, height=H, width=W)

c.pack(side=BOTTOM)

c.create\_line(left, h-down, left, up, arrow=LAST)

c.create\_line(left, line\_1, w, line\_1, arrow=LAST)

c.create\_line(left, line\_2, w, line\_2, arrow=LAST)

c.create\_line(left, line\_3, w, line\_3, arrow=LAST)

c.create\_line(left, line\_4, w, line\_4, arrow=LAST)

c.create\_line(left, line\_5, w, line\_5, arrow=LAST)

c.create\_text(left-15, 20, text='L(t)')

c.create\_text(w-right, line\_1+10, text='t')

c.create\_text(w-right, line\_2+10, text='t')

c.create\_text(w-right, line\_3+10, text='t')

c.create\_text(w-right, line\_4+10, text='t')

c.create\_text(w-right, line\_5+10, text='t')

c.create\_text(w\*(1/2), line\_1-10, text='процесс L(t)')

c.create\_text(w\*(1/2), line\_2-10, text='исходный поток событий')

c.create\_text(w\*(1/2), line\_3-10, text='формирование периодов ненаблюдения')

c.create\_text(w\*(1/2), line\_4-10, text='общий период ненаблюдения')

c.create\_text(w\*(1/2), line\_5-10, text='наблюдаемый поток')

cx = left

line\_s1 = line\_1\*(1/4)

line\_s2 = line\_1\*(3/4)

for i in range(len(MassIntervS) - 1):

if MassIntervS[i][1] == 1:

for j in range(len(MassIntervS[i][2])):

if MassIntervS[i][2][j][1] == 1:

c.create\_line(cx, line\_s1, cx+MassIntervS[i][2][j][0]\*kt, line\_s1, fill='blue', width=5)

else:

c.create\_line(cx, line\_s1, cx+MassIntervS[i][2][j][0]\*kt, line\_s1, fill='green', width=5)

cx = cx + MassIntervS[i][2][j][0] \* kt

c.create\_line(cx, line\_s1, cx, line\_s2, arrow=LAST)

else:

c.create\_line(cx, line\_s2, cx+MassIntervS[i][0]\*kt, line\_s2)

c.create\_line(cx + MassIntervS[i][0]\*kt, line\_s2, cx + MassIntervS[i][0]\*kt, line\_s1, arrow=LAST)

cx = cx + MassIntervS[i][0]\*kt

if MassIntervS[len(MassIntervS)-1][1] == 1:

for j in range(len(MassIntervS[len(MassIntervS)-1][2])):

if MassIntervS[len(MassIntervS)-1][2][j][1] == 1:

c.create\_line(cx, line\_s1, cx+MassIntervS[len(MassIntervS)-1][2][j][0]\*kt, line\_s1, fill='blue', width=5)

else:

c.create\_line(cx, line\_s1, cx+MassIntervS[len(MassIntervS)-1][2][j][0]\*kt, line\_s1, fill='green', width=5)

cx = cx + MassIntervS[len(MassIntervS)-1][2][j][0] \* kt

else:

c.create\_line(cx, line\_s2, cx + MassIntervS[len(MassIntervS) - 1][0]\*kt, line\_s2)

Hi = 1

RadOval = 3

HiRec = h\*(1/120)

cx = 0

count\_event = 0

for i in range(len(MassInitial)):

cx = MassInitial[i][0]

if MassInitial[i][1] == 'M':

Hi = Hi + 1

if MassInitial[i][2] == 1:

c.create\_line(cx\*kt+left, line\_s1, cx\*kt+left, line\_2, dash=(5, 1))

c.create\_oval((cx\*kt+left-RadOval), line\_2-RadOval, (cx\*kt+left+RadOval), line\_2+RadOval, fill='black')

c.create\_rectangle(cx\*kt+left, (line\_3+10)+HiRec\*Hi, ((cx+a[1])\*kt+left), (line\_3+10)+HiRec+HiRec\*Hi, fill='gray')

c.create\_rectangle(cx\*kt+left, (line\_4+10)+HiRec, ((cx+a[1])\*kt+left), (line\_4+10)+HiRec\*2, fill='black')

else:

c.create\_line(cx\*kt+left, line\_s2, cx\*kt+left, line\_2, dash=(5, 1))

c.create\_oval((cx\*kt+left-RadOval), line\_2-RadOval, (cx\*kt+left+RadOval), line\_2+RadOval, fill='black')

c.create\_rectangle(cx\*kt+left, (line\_3+10)+HiRec\*Hi, ((cx+a[1])\*kt+left), (line\_3+10)+HiRec+HiRec\*Hi, fill='gray')

c.create\_rectangle(cx\*kt+left, (line\_4+10)+HiRec, ((cx+a[1])\*kt+left), (line\_4+10)+HiRec\*2, fill='black')

else:

Hi = 1

count\_event = count\_event + 1

if MassInitial[i][2] == 1:

c.create\_line(cx\*kt+left, line\_s1, cx\*kt+left, line\_5)

c.create\_oval((cx\*kt+left-RadOval), line\_2-RadOval, (cx\*kt+left+RadOval), line\_2+RadOval, fill='white')

c.create\_oval((cx\*kt+left-RadOval), line\_5-RadOval, (cx\*kt+left+RadOval), line\_5+RadOval, fill='white')

c.create\_rectangle(cx\*kt+left, (line\_3+10)+HiRec, ((cx+a[1])\*kt+left), (line\_3+10)+HiRec\*2, fill='gray')

c.create\_rectangle(cx\*kt+left, (line\_4+10)+HiRec, ((cx+a[1])\*kt+left), (line\_4+10)+HiRec\*2, fill='black')

c.create\_text(cx\*kt+left, h-10, text='t%s' % count\_event)

else:

c.create\_line(cx\*kt+left, line\_s2, cx\*kt+left, line\_5)

c.create\_oval((cx\*kt+left-RadOval), line\_2-RadOval, (cx\*kt+left+RadOval), line\_2+RadOval, fill='white')

c.create\_oval((cx\*kt+left-RadOval), line\_5-RadOval, (cx\*kt+left+RadOval), line\_5+RadOval, fill='white')

c.create\_rectangle(cx\*kt+left, (line\_3+10)+HiRec, ((cx+a[1])\*kt+left), (line\_3+10)+HiRec\*2, fill='gray')

c.create\_rectangle(cx\*kt+left, (line\_4+10)+HiRec, ((cx+a[1])\*kt+left), (line\_4+10)+HiRec\*2, fill='black')

c.create\_text(cx\*kt+left, h-10, text='t%s' % count\_event)

def LabelW(t, a):

l = LabelFrame(Window, text=t)

e = Entry(l)

e.insert(0, a)

l.pack(side=LEFT)

e.pack()

LabelW('Парам. Tm:', a[0])

LabelW('Парам. T:', a[1])

LabelW('Парам. l1:', a[2])

LabelW('Парам. l2:', a[3])

LabelW('Парам. a1:', a[4])

LabelW('Парам. a2:', a[5])

LabelW('Вер-ть P.1.1:', a[6])

LabelW('Вер-ть P.1.2:', a[7])

LabelW('Вер-ть P.2.1:', a[8])

LabelW('Вер-ть P.2.2:', a[9])

Window.mainloop()

return 0

potokArt(550, 1150, 30, [35, 0.5, 1.5, 0.8, 0.8, 0.7, 0.3, 0.7, 0.3, 0.7])

Код программ реализующий статистические эксперименты

Эксперимент 1.1.

from tkinter import ttk

A = potokSimulator(10, 0, 1.1, 0.5, 0.7, 0.8, 0.4, 0.6, 0.6, 0.4)  
MassInitial = A[1]  
MassObserved = [ ]  
for i in range(len(MassInitial)):  
 if MassInitial[i][1] == 'O':  
 MassObserved.append(round(MassInitial[i][0], 4))  
 else:  
 MassObserved.append(' ')  
row1 = [ ]  
for i in range(len(MassInitial)):  
 row1.append(round(MassInitial[i][0], 4))  
  
w = Tk()  
w.geometry("800x200")  
columns = ['ti']  
for i in range(len(MassInitial)):  
 columns.append('t%s' %i)  
tree = ttk.Treeview(columns=columns, show="headings")  
tree.pack(fill=BOTH, side=BOTTOM, expand=1)  
for i in columns:  
 tree.heading(i, text=i)  
 tree.column(i, width=1)  
row1.insert(0, 't исх.п.')  
MassObserved.insert(0, 't набл.п.')  
table = [row1, MassObserved]  
for i in table:  
 tree.insert("", END, values=i)  
w.mainloop()

Эксперимент 1.2.

A = potokSimulator(10, 0.5, 1.1, 0.5, 0.7, 0.8, 0.4, 0.6, 0.6, 0.4)  
MassInitial = A[1]  
MassObserved = [ ]  
for i in range(len(MassInitial)):  
 if MassInitial[i][1] == 'O':  
 MassObserved.append(round(MassInitial[i][0], 4))  
 else:  
 MassObserved.append(' ')  
row1 = [ ]  
for i in range(len(MassInitial)):  
 row1.append(round(MassInitial[i][0], 4))  
  
w = Tk()  
w.geometry("800x200")  
columns = ['ti']  
for i in range(len(MassInitial)):  
 columns.append('t%s' %i)  
tree = ttk.Treeview(columns=columns, show="headings")  
tree.pack(fill=BOTH, side=BOTTOM, expand=1)  
for i in columns:  
 tree.heading(i, text=i)  
 tree.column(i, width=1)  
row1.insert(0, 't исх.п.')  
MassObserved.insert(0, 't набл.п.')  
table = [row1, MassObserved]  
for i in table:  
 tree.insert("", END, values=i)  
w.mainloop()

Эксперимент 2.

f = open('Exp2.txt', 'w')  
a = 100  
b = 4100  
h = 100  
MassM\_tau = [ ]  
MassM\_ksi = [ ]  
MassM\_eta1 = [ ]  
MassM\_eta2 = [ ]  
for q in range(a, b, h):  
 MassTau = [ ]  
 MassKsi = [ ]  
 MassEta1 = [ ]  
 MassEta2 = [ ]  
 Tm = q  
 for r in range(100):  
 T = 0.1  
 A = potokSimulator(q, T, 1.5, 0.9, 1.1, 0.2, 0.3, 0.7, 0.4, 0.6)  
 MassIntervS = A[0]  
 MassInitial = A[1]  
 MassObserved = [ ]  
 for i in range(len(MassInitial)):  
 if MassInitial[i][1] == 'O':  
 MassObserved.append(MassInitial[i])  
 tau = 0  
 M\_tau = 0  
 xi = 0  
 xj = 0  
 if len(MassObserved) > 1:  
 for i in range(1, len(MassObserved)):  
 xi = MassObserved[i-1]  
 xj = MassObserved[i]  
 #print('tau', (xj[0]-xi[0]))  
 tau = tau + (xj[0]-xi[0])  
 tau = tau/(len(MassObserved)-1)  
 else:  
 tau = 0  
 print(' Имитационная модель содержит только одно событие')  
 MassTau.append(tau)  
 MassIntervNotObs = [ ] # периоды ненаблюдения  
 ksi = 0  
 M\_ksi = 0  
 x = 0  
 x = MassInitial[0][0]  
 for i in range(1, len(MassInitial)):  
 if MassInitial[i][0] - MassInitial[i - 1][0] > T:  
 if x == MassInitial[i - 1][0]:  
 MassIntervNotObs.append(T)  
 x = MassInitial[i][0]  
 else:  
 MassIntervNotObs.append(((MassInitial[i - 1][0] - x)+T))  
 x = MassInitial[i][0]  
 if len(MassIntervNotObs) != 0:  
 for i in range(len(MassIntervNotObs)):  
 ksi = ksi + MassIntervNotObs[i]  
 ksi = ksi/(len(MassIntervNotObs))  
 else:  
 ksi = 0  
 print(' Имитационная модель не имеет периодов ненаблюдения')  
 MassKsi.append(ksi)  
 eta1 = 0  
 eta2 = 0  
 M\_eta1 = 0  
 M\_eta2 = 0  
 count\_eta1 = 0  
 count\_eta2 = 0  
 for i in range(len(MassIntervS)):  
 if MassIntervS[i][1] == 1:  
 eta1 = eta1 + MassIntervS[i][0]  
 count\_eta1 = count\_eta1 + 1  
 else:  
 eta2 = eta2 + MassIntervS[i][0]  
 count\_eta2 = count\_eta2 + 1  
 if count\_eta1 != 0:  
 eta1 = eta1/(count\_eta1)  
 else:  
 eta1 = 0  
 print(' Иммитационная модель не содержит 1-го состояния')  
 if count\_eta2 != 0:  
 eta2 = eta2/(count\_eta2)  
 else:  
 eta2 = 0  
 print(' Иммитационная модель не содержит 2-го состояния')  
 MassEta1.append(eta1)  
 MassEta2.append(eta2)  
 if len(MassTau) != 0:  
 for i in range(len(MassTau)):  
 M\_tau = M\_tau + MassTau[i]  
 M\_tau = (M\_tau/(len(MassTau)))  
 else:  
 M\_tau = 0  
 MassM\_tau.append(round(M\_tau, 4))  
 if len(MassKsi) != 0:  
 for i in range(len(MassKsi)):  
 M\_ksi = M\_ksi + MassKsi[i]  
 M\_ksi = (M\_ksi/(len(MassKsi)))  
 else:  
 M\_ksi = 0  
 MassM\_ksi.append(round(M\_ksi, 6))  
 if len(MassEta1) != 0:  
 for i in range(len(MassEta1)):  
 M\_eta1 = (M\_eta1 + MassEta1[i])  
 M\_eta1 = (M\_eta1/(len(MassEta1)))  
 else:  
 M\_eta1 = 0  
 MassM\_eta1.append(round(M\_eta1, 4))  
 if len(MassEta2) != 0:  
 for i in range(len(MassEta2)):  
 M\_eta2 = (M\_eta2 + MassEta2[i])  
 M\_eta2 = (M\_eta2/(len(MassEta2)))  
 else:  
 M\_eta2 = 0  
 MassM\_eta2.append(round(M\_eta2, 4))  
for i in range(len(MassM\_tau)):  
 f.write('%s\t' % MassM\_tau[i])  
f.write('\n')  
for i in range(len(MassM\_ksi)):  
 f.write('%s ' % MassM\_ksi[i])  
f.write('\n')  
for i in range(len(MassM\_eta1)):  
 f.write('%s\t' % MassM\_eta1[i])  
f.write('\n')  
for i in range(len(MassM\_eta2)):  
 f.write('%s\t' % MassM\_eta2[i])  
f.write('\n')  
f.close()

Вывод численных и графических результатов эксперимента 2.

def ax\_(a, x, y):  
 a.set\_xlabel(x)  
 a.set\_ylabel(y)  
  
f = open('Exp2.txt', 'r')  
MassM\_tau = [ ]  
MassM\_ksi = [ ]  
MassM\_eta1 = [ ]  
MassM\_eta2 = [ ]  
x = [ ]  
nam = f.readline()  
MassM\_tau = list(map(float, nam.split()))  
nam = f.readline()  
MassM\_ksi = list(map(float, nam.split()))  
nam = f.readline()  
MassM\_eta1 = list(map(float, nam.split()))  
nam = f.readline()  
MassM\_eta2 = list(map(float, nam.split()))  
print(MassM\_tau)  
for i in range(100, 4100, 100):  
 x.append(i)  
  
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)  
ax1.scatter(x, MassM\_tau)  
y = MassM\_tau[len(MassM\_tau)-1]  
ax1.set\_ylim(y+0.05, y-0.05)  
ax\_(ax1, 'Tm', 'M(tau)')  
  
ax2.scatter(x, MassM\_ksi)  
y = MassM\_ksi[len(MassM\_ksi)-1]  
ax2.set\_ylim(y-0.001, y+0.001)  
ax\_(ax2, 'Tm', 'M(ksi)')  
  
ax3.scatter(x, MassM\_eta1)  
y = MassM\_eta1[len(MassM\_eta1)-1]  
ax3.set\_ylim(y-0.05, y+0.05)  
ax\_(ax3, 'Tm', 'M(eta1)')  
  
ax4.scatter(x, MassM\_eta2)  
y = MassM\_eta2[len(MassM\_eta2)-1]  
ax4.set\_ylim(y-0.5, y+0.5)  
ax\_(ax4, 'Tm', 'M(eta2)')  
plt.show()  
  
w = Tk()  
w.geometry("1200x200")  
x.insert(0, 'Tm')  
tree = ttk.Treeview(columns=x, show="headings")  
tree.pack(fill=BOTH, expand=1)  
for i in x:  
 tree.heading(i, text=i)  
 tree.column(i, width=1)  
MassM\_tau.insert(0, 'M(tau)')  
MassM\_ksi.insert(0, 'M(ksi)')  
MassM\_eta1.insert(0, 'M(eta1)')  
MassM\_eta2.insert(0, 'M(eta2)')  
table = [MassM\_tau, MassM\_ksi, MassM\_eta1, MassM\_eta2]  
for i in table:  
 tree.insert("", END, values=i)  
w.mainloop()

Эксперимент 3.

f = open('Exp3.txt', 'w')  
a = 1  
b = 16  
h = 1  
MassM\_tau = [ ]  
MassM\_ksi = [ ]  
MassM\_eta1 = [ ]  
MassM\_eta2 = [ ]  
for q in range(a, b, h):  
 MassTau = [ ]  
 MassKsi = [ ]  
 MassEta1 = [ ]  
 MassEta2 = [ ]  
 T = q/10  
 for r in range(100):  
 A = potokSimulator(3000, T, 1.5, 0.9, 1.1, 0.2, 0.3, 0.7, 0.4, 0.6)  
 MassIntervS = A[0]  
 MassInitial = A[1]  
 MassObserved = [ ]  
 for i in range(len(MassInitial)):  
 if MassInitial[i][1] == 'O':  
 MassObserved.append(MassInitial[i])  
 tau = 0  
 M\_tau = 0  
 xi = 0  
 xj = 0  
 if len(MassObserved) > 1:  
 for i in range(1, len(MassObserved)):  
 xi = MassObserved[i-1]  
 xj = MassObserved[i]  
 tau = tau + (xj[0]-xi[0])  
 tau = tau/(len(MassObserved)-1)  
 else:  
 tau = 0  
 print(' Имитационная модель содержит только одно событие')  
 MassTau.append(tau)  
 MassIntervNotObs = [ ] # периоды ненаблюдения  
 ksi = 0  
 M\_ksi = 0  
 x = 0  
 x = MassInitial[0][0]  
 for i in range(1, len(MassInitial)):  
 if MassInitial[i][0] - MassInitial[i - 1][0] > T:  
 if x == MassInitial[i - 1][0]:  
 MassIntervNotObs.append(T)  
 x = MassInitial[i][0]  
 else:  
 MassIntervNotObs.append(((MassInitial[i - 1][0] - x)+T))  
 x = MassInitial[i][0]  
 if len(MassIntervNotObs) != 0:  
 for i in range(len(MassIntervNotObs)):  
 ksi = ksi + MassIntervNotObs[i]  
 ksi = ksi/(len(MassIntervNotObs))  
 else:  
 ksi = 0  
 print(' Имитационная модель не имеет периодов ненаблюдения')  
 MassKsi.append(ksi)  
 eta1 = 0  
 eta2 = 0  
 M\_eta1 = 0  
 M\_eta2 = 0  
 count\_eta1 = 0  
 count\_eta2 = 0  
 for i in range(len(MassIntervS)):  
 if MassIntervS[i][1] == 1:  
 eta1 = eta1 + MassIntervS[i][0]  
 count\_eta1 = count\_eta1 + 1  
 else:  
 eta2 = eta2 + MassIntervS[i][0]  
 count\_eta2 = count\_eta2 + 1  
 if count\_eta1 != 0:  
 eta1 = eta1/(count\_eta1)  
 else:  
 eta1 = 0  
 print(' Иммитационная модель не содержит 1-го состояния')  
 if count\_eta2 != 0:  
 eta2 = eta2/(count\_eta2)  
 else:  
 eta2 = 0  
 print(' Иммитационная модель не содержит 2-го состояния')  
 MassEta1.append(eta1)  
 MassEta2.append(eta2)  
 if len(MassTau) != 0:  
 for i in range(len(MassTau)):  
 M\_tau = M\_tau + MassTau[i]  
 M\_tau = (M\_tau/(len(MassTau)))  
 else:  
 M\_tau = 0  
 MassM\_tau.append(round(M\_tau, 4))  
 if len(MassKsi) != 0:  
 for i in range(len(MassKsi)):  
 M\_ksi = M\_ksi + MassKsi[i]  
 M\_ksi = (M\_ksi/(len(MassKsi)))  
 else:  
 M\_ksi = 0  
 MassM\_ksi.append(round(M\_ksi, 6))  
 if len(MassEta1) != 0:  
 for i in range(len(MassEta1)):  
 M\_eta1 = (M\_eta1 + MassEta1[i])  
 M\_eta1 = (M\_eta1/(len(MassEta1)))  
 else:  
 M\_eta1 = 0  
 MassM\_eta1.append(round(M\_eta1, 4))  
 if len(MassEta2) != 0:  
 for i in range(len(MassEta2)):  
 M\_eta2 = (M\_eta2 + MassEta2[i])  
 M\_eta2 = (M\_eta2/(len(MassEta2)))  
 else:  
 M\_eta2 = 0  
 MassM\_eta2.append(round(M\_eta2, 4))  
for i in range(len(MassM\_tau)):  
 f.write('%s\t' % MassM\_tau[i])  
f.write('\n')  
for i in range(len(MassM\_ksi)):  
 f.write('%s ' % MassM\_ksi[i])  
f.write('\n')  
for i in range(len(MassM\_eta1)):  
 f.write('%s\t' % MassM\_eta1[i])  
f.write('\n')  
for i in range(len(MassM\_eta2)):  
 f.write('%s\t' % MassM\_eta2[i])  
f.write('\n')  
f.close()

Вывод численных и графических результатов эксперимента 3.

f = open('Exp3.txt', 'r')  
MassM\_tau = [ ]  
MassM\_ksi = [ ]  
MassM\_eta1 = [ ]  
MassM\_eta2 = [ ]  
x = [ ]  
nam = f.readline()  
MassM\_tau = list(map(float, nam.split()))  
nam = f.readline()  
MassM\_ksi = list(map(float, nam.split()))  
nam = f.readline()  
MassM\_eta1 = list(map(float, nam.split()))  
nam = f.readline()  
MassM\_eta2 = list(map(float, nam.split()))  
print(MassM\_tau)  
for i in range(1, 16, 1):  
 i = i/10  
 x.append(i)  
  
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)  
ax1.scatter(x, MassM\_tau)  
ax\_(ax1, 'T', 'M(tau)')  
  
ax2.scatter(x, MassM\_ksi)  
ax\_(ax2, 'T', 'M(ksi)')  
  
ax3.scatter(x, MassM\_eta1)  
y = MassM\_eta1[len(MassM\_eta1)-1]  
ax3.set\_ylim(y-0.05, y+0.05)  
ax\_(ax3, 'T', 'M(eta1)')  
  
ax4.scatter(x, MassM\_eta2)  
y = MassM\_eta2[len(MassM\_eta2)-1]  
ax4.set\_ylim(y-0.5, y+0.5)  
ax\_(ax4, 'T', 'M(eta2)')  
plt.show()  
  
w = Tk()  
w.geometry("1200x200")  
x.insert(0, 'T')  
tree = ttk.Treeview(columns=x, show="headings")  
tree.pack(fill=BOTH, expand=1)  
for i in x:  
 tree.heading(i, text=i)  
 tree.column(i, width=1)  
MassM\_tau.insert(0, 'M(tau)')  
MassM\_ksi.insert(0, 'M(ksi)')  
MassM\_eta1.insert(0, 'M(eta1)')  
MassM\_eta2.insert(0, 'M(eta2)')  
table = [MassM\_tau, MassM\_ksi, MassM\_eta1, MassM\_eta2]  
for i in table:  
 tree.insert("", END, values=i)  
w.mainloop()

Эксперимент 4.

f = open('Exp4.txt', 'w')  
# вероятности перехода из 1-го состояния во 2-е увеличены  
a = 5  
b = 55  
h = 5  
MassM\_eta1 = [ ]  
for q in range(a, b, h):  
 MassEta1 = [ ]  
 l1 = q/10  
 for r in range(100):  
 A = potokSimulator(3000, 0.1, l1, 0.2, 1.1, 0.2, 0.3, 0.7, 0.3, 0.7)  
 MassIntervS = A[0]  
 eta1 = 0  
 M\_eta1 = 0  
 count\_eta1 = 0  
 for i in range(len(MassIntervS)):  
 if MassIntervS[i][1] == 1:  
 eta1 = eta1 + MassIntervS[i][0]  
 count\_eta1 = count\_eta1 + 1  
 if count\_eta1 != 0:  
 eta1 = eta1/(count\_eta1)  
 else:  
 eta1 = 0  
 print(' Иммитационная модель не содержит 1-го состояния')  
 MassEta1.append(eta1)  
 if len(MassEta1) != 0:  
 for i in range(len(MassEta1)):  
 M\_eta1 = (M\_eta1 + MassEta1[i])  
 M\_eta1 = (M\_eta1/(len(MassEta1)))  
 else:  
 M\_eta1 = 0  
 MassM\_eta1.append(round(M\_eta1, 4))  
for i in range(len(MassM\_eta1)):  
 f.write('%s\t' % MassM\_eta1[i])  
f.write('\n')  
  
  
#вероятности остаться в 1-м состоянии увеличены  
MassM\_eta1 = [ ]  
for q in range(a, b, h):  
 MassEta1 = [ ]  
 l1 = q/10  
 for r in range(100):  
 A = potokSimulator(3000, 0.1, l1, 0.2, 1.1, 0.2, 0.7, 0.3, 0.7, 0.3)  
 MassIntervS = A[0]  
 eta1 = 0  
 M\_eta1 = 0  
 count\_eta1 = 0  
 for i in range(len(MassIntervS)):  
 if MassIntervS[i][1] == 1:  
 eta1 = eta1 + MassIntervS[i][0]  
 count\_eta1 = count\_eta1 + 1  
 if count\_eta1 != 0:  
 eta1 = eta1/(count\_eta1)  
 else:  
 eta1 = 0  
 print(' Иммитационная модель не содержит 1-го состояния')  
 MassEta1.append(eta1)  
 if len(MassEta1) != 0:  
 for i in range(len(MassEta1)):  
 M\_eta1 = (M\_eta1 + MassEta1[i])  
 M\_eta1 = (M\_eta1/(len(MassEta1)))  
 else:  
 M\_eta1 = 0  
 MassM\_eta1.append(round(M\_eta1, 4))  
for i in range(len(MassM\_eta1)):  
 f.write('%s\t' % MassM\_eta1[i])  
f.write('\n')  
f.close()

Вывод численных и графических результатов эксперимента 4.

f = open('Exp4.txt', 'r')  
MassM\_eta1 = [ ]  
MassM\_eta2 = [ ]  
x = [ ]  
nam = f.readline()  
MassM\_eta1\_1 = list(map(float, nam.split()))  
nam = f.readline()  
MassM\_eta1\_2 = list(map(float, nam.split()))  
for i in range(5, 55, 5):  
 i = i/10  
 x.append(i)  
  
fig, (ax1, ax2) = plt.subplots(2)  
ax1.scatter(x, MassM\_eta1\_1)  
ax1.set\_xlabel("l1")  
ax1.set\_ylabel("M(eta1)", rotation=0, labelpad=20)  
  
ax2.scatter(x, MassM\_eta1\_2)  
ax2.set\_xlabel("l1")  
ax2.set\_ylabel("M(eta1)", rotation=0, labelpad=20)  
plt.show()  
  
w = Tk()  
w.geometry("800x600")  
x.insert(0, 'l1')  
tree1 = ttk.Treeview(columns=x, show="headings")  
tree1.pack(fill=BOTH, expand=1)  
for i in x:  
 tree1.heading(i, text=i)  
 tree1.column(i, width=1)  
MassM\_eta1\_1.insert(0, 'M(eta1)')  
table1 = [MassM\_eta1\_1]  
for i in table1:  
 tree1.insert("", END, values=i)  
tree2 = ttk.Treeview(columns=x, show="headings")  
tree2.pack(fill=BOTH, expand=1)  
for i in x:  
 tree2.heading(i, text=i)  
 tree2.column(i, width=1)  
MassM\_eta1\_2.insert(0, 'M(eta1)')  
table2 = [MassM\_eta1\_2]  
for i in table2:  
 tree2.insert("", END, values=i)  
w.mainloop()

Эксперимент 5.

f = open('Exp5.txt', 'w')  
a = 1  
b = 11  
h = 1  
MassM\_eta2 = [ ]  
for q in range(a, b, h):  
 MassEta2 = [ ]  
 a2 = q/100  
 for r in range(100):  
 A = potokSimulator(3000, 0.1, 1.5, 0.2, 1.1, a2, 0.4, 0.6, 0.6, 0.4)  
 MassIntervS = A[0]  
 eta2 = 0  
 M\_eta2 = 0  
 count\_eta2 = 0  
 for i in range(len(MassIntervS)):  
 if MassIntervS[i][1] == 2:  
 eta2 = eta2 + MassIntervS[i][0]  
 count\_eta2 = count\_eta2 + 1  
 if count\_eta2 != 0:  
 eta2 = eta2/(count\_eta2)  
 else:  
 eta2 = 0  
 print(' Иммитационная модель не содержит 2-го состояния')  
 MassEta2.append(eta2)  
 print(a2)  
 print(MassEta2)  
 if len(MassEta2) != 0:  
 for i in range(len(MassEta2)):  
 M\_eta2 = (M\_eta2 + MassEta2[i])  
 M\_eta2 = (M\_eta2/(len(MassEta2)))  
 else:  
 M\_eta2 = 0  
 MassM\_eta2.append(round(M\_eta2, 4))  
for i in range(len(MassM\_eta2)):  
 f.write('%s\t' % MassM\_eta2[i])  
f.write('\n')  
  
a = 5  
b = 55  
h = 5  
MassM\_eta2 = [ ]  
for q in range(a, b, h):  
 MassEta2 = [ ]  
 a2 = q/10  
 for r in range(100):  
 A = potokSimulator(3000, 0.1, 1.5, 0.2, 1.1, a2, 0.4, 0.6, 0.6, 0.4)  
 MassIntervS = A[0]  
 eta2 = 0  
 M\_eta2 = 0  
 count\_eta2 = 0  
 for i in range(len(MassIntervS)):  
 if MassIntervS[i][1] == 2:  
 eta2 = eta2 + MassIntervS[i][0]  
 count\_eta2 = count\_eta2 + 1  
 if count\_eta2 != 0:  
 eta2 = eta2/(count\_eta2)  
 else:  
 eta2 = 0  
 print(' Иммитационная модель не содержит 2-го состояния')  
 MassEta2.append(eta2)  
 print(a2)  
 print(MassEta2)  
 if len(MassEta2) != 0:  
 for i in range(len(MassEta2)):  
 M\_eta2 = (M\_eta2 + MassEta2[i])  
 M\_eta2 = (M\_eta2/(len(MassEta2)))  
 else:  
 M\_eta2 = 0  
 MassM\_eta2.append(round(M\_eta2, 4))  
for i in range(len(MassM\_eta2)):  
 f.write('%s\t' % MassM\_eta2[i])  
f.write('\n')  
f.close()

Вывод численных и графических результатов эксперимента 5.

f = open('Exp5.txt', 'r')  
MassM\_eta2 = [ ]  
x1 = [ ]  
nam = f.readline()  
MassM\_eta2\_1 = list(map(float, nam.split()))  
nam = f.readline()  
MassM\_eta2\_2 = list(map(float, nam.split()))  
for i in range(1, 11, 1):  
 i = i/100  
 x1.append(i)  
  
fig, (ax1, ax2) = plt.subplots(2)  
ax1.scatter(x1, MassM\_eta2\_1)  
ax1.set\_xlabel("a2")  
ax1.set\_ylabel("M(eta1)", rotation=0, labelpad=20)  
x2 = [ ]  
for i in range(5, 55, 5):  
 i = i/10  
 x2.append(i)  
ax2.scatter(x2, MassM\_eta2\_2)  
ax2.set\_xlabel("a2")  
ax2.set\_ylabel("M(eta1)", rotation=0, labelpad=20)  
plt.show()  
  
w = Tk()  
w.geometry("800x600")  
x1.insert(0, 'a2')  
tree1 = ttk.Treeview(columns=x1, show="headings")  
tree1.pack(fill=BOTH, expand=1)  
for i in x1:  
 tree1.heading(i, text=i)  
 tree1.column(i, width=1)  
MassM\_eta2\_1.insert(0, 'M(eta2)')  
table1 = [MassM\_eta2\_1]  
for i in table1:  
 tree1.insert("", END, values=i)  
x2.insert(0, 'a2')  
tree2 = ttk.Treeview(columns=x2, show="headings")  
tree2.pack(fill=BOTH, expand=1)  
for i in x2:  
 tree2.heading(i, text=i)  
 tree2.column(i, width=1)  
MassM\_eta2\_2.insert(0, 'M(eta2)')  
table2 = [MassM\_eta2\_2]  
for i in table2:  
 tree2.insert("", END, values=i)  
w.mainloop()

# ПРИЛОЖЕНИЕ 2

На рисунке 16 представлен результат работы программы в виде скриншота. На нем показана одна из реализаций полусихронного потока событий второго порядка в схеме с продлевающимся мертвым временем. В реализации кусочно-постоянного случайного процесса  синим и зеленым цветом представлены периоды между событиями, сгенерированные 1-ой и 2-ой случайной величиной соответственно.

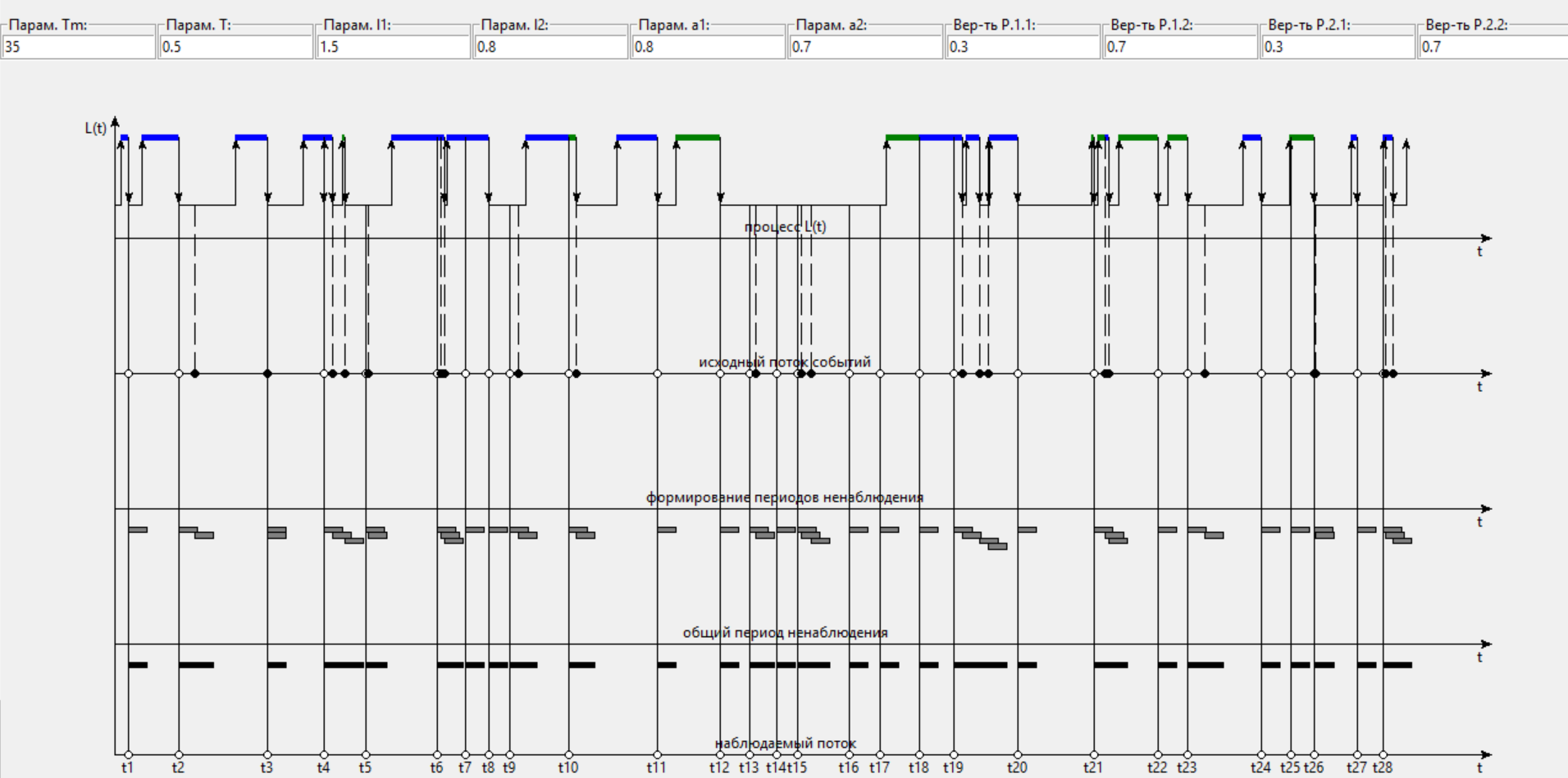


Рис. 16. Скриншот программы