МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра прикладной математики и   
экономико-математических методов

Направление Прикладная математика и информатика

**ОТЧЕТ**

**по дисциплине «Имитационное моделирование»**

студента (-ки) Широков Александр Анатольевич

(Ф.И.О. полностью)

Курс 4 Группа ПМ-1701

Форма обучения очная

Форма представления на кафедру выполненных заданий:

отчёт в письменной и электронной форме

|  |  |
| --- | --- |
| Оценка по результатам текущего контроля (КТ3) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись преподавателя)* |

Санкт-Петербург

2020 г.

# Оглавление

[Оглавление 2](#_Toc54711082)

[Введение 4](#_Toc54711083)

[**1. Модель. Имитационная модель. Предмет имитационного моделирования** 5](#_Toc54711084)

[**1.1 Имитационная модель** 5](#_Toc54711085)

[**1.2 Имитационное моделирование** 5](#_Toc54711086)

[**2. Этапы построения имитационной модели** 5](#_Toc54711087)

[**3 Генераторы случайных чисел** 6](#_Toc54711088)

[**3.1 Равномерное распределение** 6](#_Toc54711089)

[**3.2 Метод обратного преобразования** 8](#_Toc54711090)

[**3.3 Алгоритмические генераторы случайных чисел** 9](#_Toc54711091)

[**3.3.1 Метод серединных квадратов** 9](#_Toc54711092)

[**3.3.2 Метод серединных произведений** 12](#_Toc54711093)

[**3.3.3 Линейный конгруэнтный датчик** 14](#_Toc54711094)

[**3.4 Моделирование нормально распределённых случайных величин** 16](#_Toc54711095)

[**3.4.1 Преобразование Бокса-Мюллера** 16](#_Toc54711096)

[**3.4.2 Преобразование через центральную предельную теорему** 18](#_Toc54711097)

[**3.5 Статистическое моделирование. Метод Монте-Карло** 19](#_Toc54711098)

[**3.5.1 Интегрирование с помощью метода Монте-Карло** 19](#_Toc54711099)

[**3.5.2 Классический эксперимент Полиа** 22](#_Toc54711100)

[**3.5.3 Моделирование выигрыша при игре в карты** 23](#_Toc54711101)

[**3.6 Генерация случайных событий** 24](#_Toc54711102)

[**4. Имитационное моделирование** 24](#_Toc54711103)

[**4.1 Процессное (дискретно-событийное) моделирование** 24](#_Toc54711104)

[**4.2 Моделирование систем массового обслуживания** 24](#_Toc54711105)

[**4.2.1 Моделирование СМО банка в Python** 24](#_Toc54711106)

[**4.2.2 Моделирование СМО банка в AnyLogic** 26](#_Toc54711107)

[**4.3 Работа с презентациями в AnyLogic** 26](#_Toc54711108)

[**4.4 Моделирование СМО “Call Center” с отказами** 29](#_Toc54711109)

[**4.5 Моделирование процесса производства мороженого** 30](#_Toc54711110)

[**4.6 Моделирование метро на учебном примере** 30](#_Toc54711111)

[**4.7 Индивидуальное задание: моделирование многоканальной СМО – кинотеатр «Художественный»** 31](#_Toc54711112)

[**4.8 Индивидуальное задание: моделирование работы станции метро Чернышевская** 31](#_Toc54711113)

[**4.9 Работа с файлами в AnyLogic** 32](#_Toc54711114)

[**4.10 Реализация разложения в ряд экспоненты и косинуса** 34](#_Toc54711115)

[**4.11 Модели SIR** 34](#_Toc54711116)

[**5 Агентные модели в дискретном пространстве** 36](#_Toc54711117)

[**5.1 Жизнь Конвея** 36](#_Toc54711118)

# Введение

Курсовая работа содержит описание методов решения некоторых задач численного анализа, их программную реализацию, решение тестовых задач и анализ полученных результатов (сравнение с точным решением или с решением, полученным встроенной функцией, или оценка точности полученного решения).

Название одного из методов, который описан более подробно, вынесено на титульный лист в качестве темы курсовой работы.

**Цель исследования** – изучить и продемонстрировать принцип работы методов решения различных задач численного анализа, в том числе итерационного метода решения системы линейных алгебраических уравнений в среде Wolfram Mathematica.

**Поставленные задачи:**

1. Познакомиться с несколькими методами решения задач численного анализа
2. Описать постановки решаемых задач и алгоритмы реализации используемых методов
3. Продемонстрировать программную реализацию методов в среде Wolfram Mathematica
4. Решить тестовые задачи и сравнить полученные результаты с точными или с результатами, полученными с использованием встроенных функций в различных математических пакетах.

**1. Модель. Имитационная модель. Предмет имитационного моделирования**

**1.1 Имитационная модель**

**Имитация** – компьютерный эксперимент с моделью реальной системы для получения информации, которая переносится на реальную систему. (07.09.2020)

**Имитационное (программное) моделирование** – *имитация*, при котором математическая модель исследуемой системы представляет собой алгоритм функционирования системы, реализуемый программно.

**1.2 Имитационное моделирование**

**Метод имитационного моделирования** – экспериментальный метод исследования реальной системы по её имитационной модели, который сочетает особенности экспериментального подхода и использования компьютерной техники. Особенностью имитационного моделирования является тот факт, что имитационная модель позволяет воспроизводить моделируемые объекты с сохранением их *логической структуры* и с сохранением *поведенческих свойств*, *динамики взаимодействия*.

Составляющие в описании имитационной модели:

1. Статистическое описание системыm описание её структуры. При разработке ИМ необходимо выполнять структурный анализ моделируемых процессов;
2. Динамическое описание системы или описание динамики взаимодействия её элементов. При его составлении фактически требуется построение функциональной модели моделируемых динамических процессов;
3. Динамика в имитационных моделях реализуется с помощью механизма продвижения модельного времени;

**2. Этапы построения имитационной модели**

Перед построением имитационной модели необходимо выделить некоторые этапы её построения – наметить план, по которому будет производиться имитационный эксперимент.

Выделяют следующие этапы построение имитационной модели:

1. Формирование целей исследования;
2. Сбор и обработка информации о моделируемой системе и протекающих в них процессах;
3. Построение концептуальной модели:
   * Выделить изучаемую системы из более общей окружающей её системы на основе данных;
   * Определить основные и вспомогательные переменные, характеризующие элементы системы;
   * Построить диаграммы причинно-следственных связей;
   * Построить диаграммы потоков;
   * Определить начальные условия, задать параметры;
   * Задать зависимости

**3 Генераторы случайных чисел**

В основе статистического моделирования лежит генерация случайных чисел, которые должны быть *равномерно распределены* в интервале . Если генератор выдаст числа, смещённые в какую-то часть интервала (одни числа будут выпадать чаще других), то результат задачи, основанный на статистическом подходе, может оказаться абсолютно неверным.

**3.1 Равномерное распределение**

Пусть случайная величина имеет непрерывное равномерное распределение на отрезке : . Математическое ожидание и дисперсия такой случайной величины равны, соответственно:

Непрерывное равномерное распределение называют *стандартным* с математическим ожиданием и дисперсией:

Поэтому для последовательности, состоящий из случайных чисел выборочное среднее   и выборочная дисперсия должна быть равна (если это действительно равномерно распределённые случайные числа в интервале от 0 до 1).

Для перехода к произвольному интервалу используется следующее преобразование, показанное ниже.

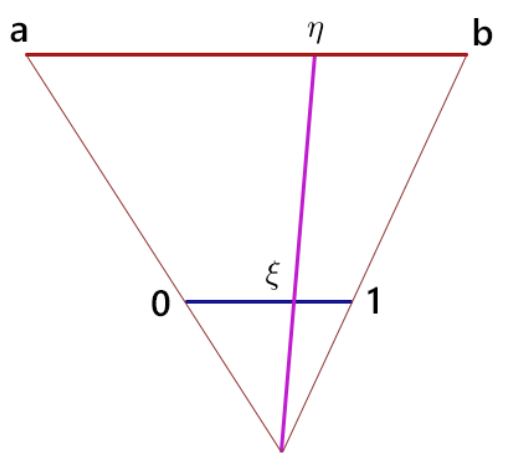


Рисунок 1 – схема перевода числа из интервала в интервал

После данного преобразования – случайное число, равномерно распределённое на , а – случайное число из интервала .

Следовательно, за эталон генератора случайных чисел (в дальнейшем – ГСЧ) принят такой генератор, который порождает *последовательность* случайных чисел с равномерным законом распределения в интервале .

Поэтому генерация произвольного случайного числа состоит из двух этапов:

* генерация равномерно распределённого случайного от до ;
* преобразование нормализованных случайных чисел в числа , которые распределены по необходимому закону распределения или в необходимом интервале.

Более того, имея генератор и зная функцию обратную к функции распределения, можно построить генератор выборки **любого непрерывного распределения** (не обязательно равномерного) с помощью *метода обратного преобразования (метод обратной функции)*.

**3.2 Метод обратного преобразования**

Метод обратного преобразования – способ генерации случайных величин с заданной функцией распределения, путём модификации работы генератора равномерно распределённых чисел.

Пусть является функцией произвольного распределения. Покажем, как, имея генератор выборки из *стандартного непрерывного равномерного распределения*, получить выборку из распределения, задаваемого функцией распределения .

Если функция строго возрастает на всей области определения, то она биективна, следовательно, имеет обратную функцию .

Пусть – выборка из стандартного непрерывного равномерного распределения. Тогда - выборка из нужного нам распределения.

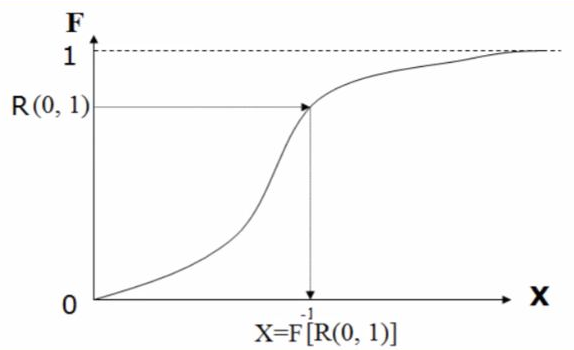


Рисунок 2 – метод обратного преобразования

Например, пусть требуется сгенерировать выборку из экспоненциального распределения с параметром . Запишем функцию распределения экспоненциальной величины:

Функция распределения строго возрастает, поэтому у неё есть обратная функция. Если – выборка из стандартного непрерывного распределения, то выражается следующим образом:

Получили искомую выборку из экспоненциального распределения.

**3.3 Алгоритмические генераторы случайных чисел**

Числа, генерируемые с помощью алгоритмических ГСЧ, всегда являются *псевдослучайными*, так как каждое последующее сгенерированное число **зависит** от предыдущего:

Последовательности, составленные из таких чисел, образуют петли, так как обязательно существует цикл, повторяющийся бесконечное число раз.

**Период** – повторяющийся цикл.

Достоинством данных ГСЧ является быстродействие, генераторы не требуют ресурсов памяти, но числа нельзя в полной мере называть случайными, так как между ними существуют зависимость.

Различают следующие алгоритмические методы получения ГСЧ:

* метод серединных квадратов
* метод серединных произведений
* линейный конгруэнтный метод
* метод перемешивания

Из данных методов будем реализовывать первые три метода, используя язык программирования Python и Wolfram Mathematica.

**3.3.1 Метод серединных квадратов**

Пусть есть некоторое четырёхзначное число . Это число возводится в квадрат и заносится в . Из берётся середина (четыре средних цифры) – новое случайное число – и записывается в . Затем процедура повторяется. В качестве случайного числа берётся число – с приписанным нулём и десятичной точкой.

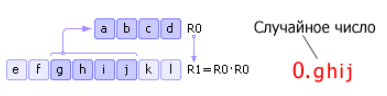


Рисунок 3 – метод серединных квадратов

В общем случае берётся некоторое число разрядности и из середины полученного квадрата , доведённого до разрядности добавлением нулей в начале числа, берётся новое число разрядности (размера) . Новое число используется как мантисса числа, и используется для повторного возведения в квадрат для продолжения ряда.

Недостатки метода:

* если на некотором шаге равен нулю, то генератор вырождается
* генератор повторяет последовательность через шагов (в лучшем случае), где – размерность числа , – основание системы счисления.

Данный способ был предложен Джоном фон Нейманом в 1946 году, но способ оказался ненадёжным и от него отказались.

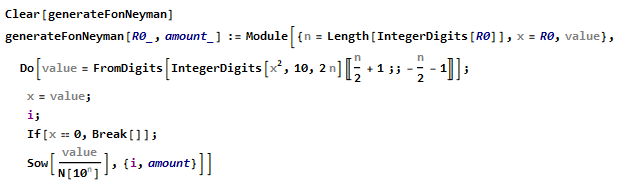


Рисунок 4 – реализация метода серединных квадратов в Wolfram Mathematica

Проверить, что случайные числа сгенерировались равномерно, можно с помощью нескольких методов:

* выборочное математическое ожидание и выборочная дисперсия выборки должны быть примерно равна и
* гистограмма выборки должна быть похожа на плотность

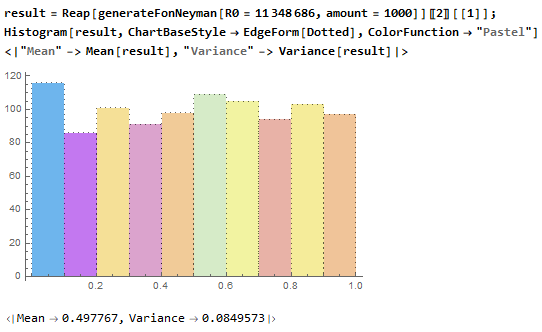


Рисунок 5 – визуализация результатов ГСЧ с помощью метода серединных квадратов

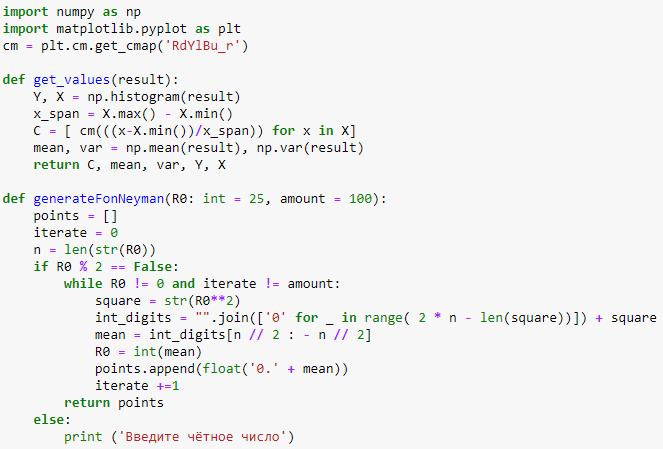


Рисунок 6 – реализация метод серединных квадратов в Python

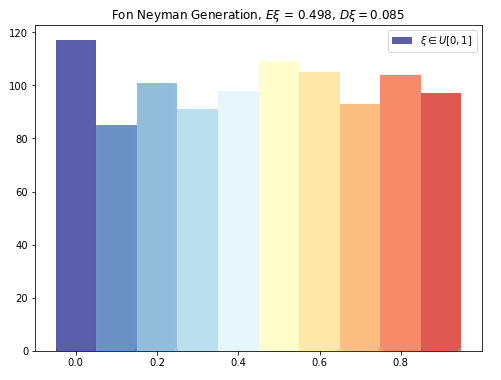


Рисунок 7 – визуализация результатов ГСЧ с помощью метода серединных квадратов в Python

**3.3.2 Метод серединных произведений**

Данный метод является улучшением метода серединных квадратов. Начальное число умножается на , из полученного результат извлекается середина и умножается на .

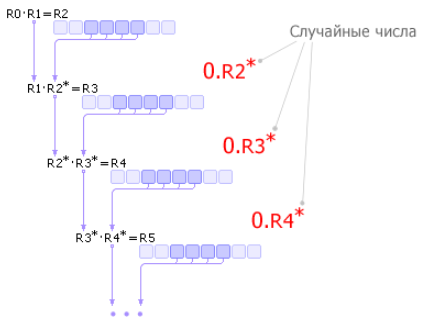


Рисунок 8 – алгоритм метода серединных произведений

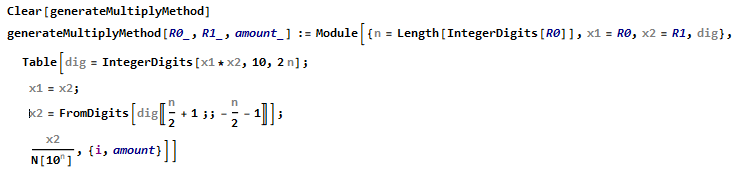


Рисунок 9 – реализация метода серединных произведений в Wolfram Mathematica

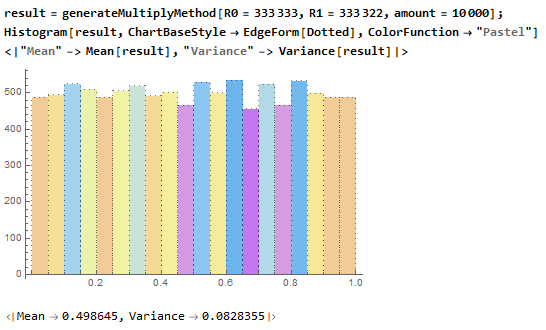


Рисунок 10 - визуализация результатов ГСЧ с помощью метода серединных произведений

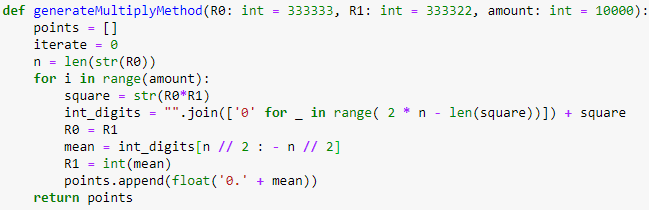


Рисунок 11 - реализация метода серединных произведений на языке Python

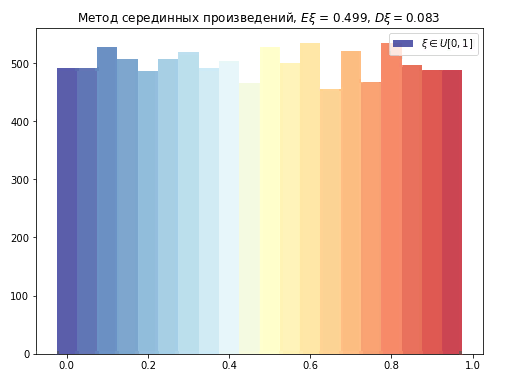


Рисунок 12 - визуализация результатов ГСЧ с помощью метода серединных произведений в Python

**3.3.3 Линейный конгруэнтный датчик**

Линейный конгруэнтный метод является одним из простейших методов, имитирующих случайные числа. В этом методе используется остаток от деления:

где – модуль, – множитель, – приращение , а – начальное значение .

Для качественного генератора требуется подобрать подходящие коэффициенты. Число должно быть довольно большим. Деление является медленной операцией, для двоичной вычислительной машины логичным будет выбор .

Одним из требований к линейным конгруэнтным последовательностям является как можно большая длина периода. Длина периода зависит от значений .

Теорема, которая будет ниже, позволяет определить, возможно ли достижение периода максимальной длины для конкректных значений .

**Теорема 1.** Линейная конгруэнтная последовательность, определённая числами и имеет период длиной тогда и только тогда:

* числа и взаимно простые;
* кратно для каждого просто , являющегося делителем ;
* кратно , если кратно

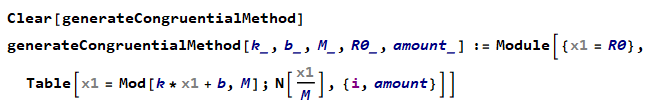


Рисунок 13 – реализация линейного конгруэнтного метода на языке Wolfram Mathematica

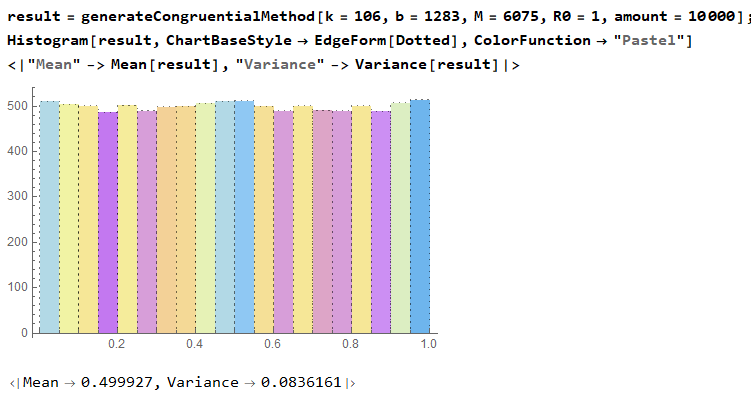


Рисунок 14 – визуализация линейного конгруэнтного метода на языке Wolfram Mathematica

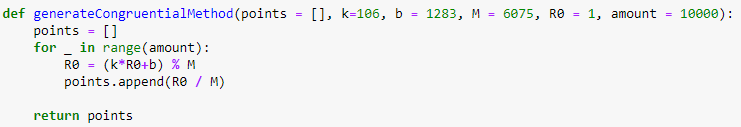


Рисунок 15 – реализация линейного конгруэнтного метода на Python

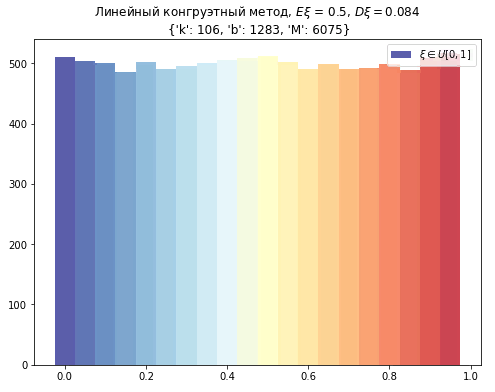


Рисунок 16 – визуализация линейного конгруэнтного метода на Python

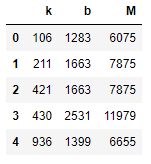


Рисунок 17 – часть таблицы коэффициентов, предназначенных для работы линейного конгруэнтного метода

Для каждого метода был найден период последовательности и выяснилось, что у каждой последовательности он равен количеству чисел в списке, то есть не наблюдается в цикле. Это означает, что мы выбрали хорошие числа для генерации случайных величин и можем использовать данные генераторы для получения других распределений.

**3.4 Моделирование нормально распределённых случайных величин**

**3.4.1 Преобразование Бокса-Мюллера**

Пусть и – независимые случайные величины, равномерно распределённые на интервале .

Вычислим и по формулам:

Тогда и – независимые и распределены нормально с математическим ожиданием и дисперсией : .

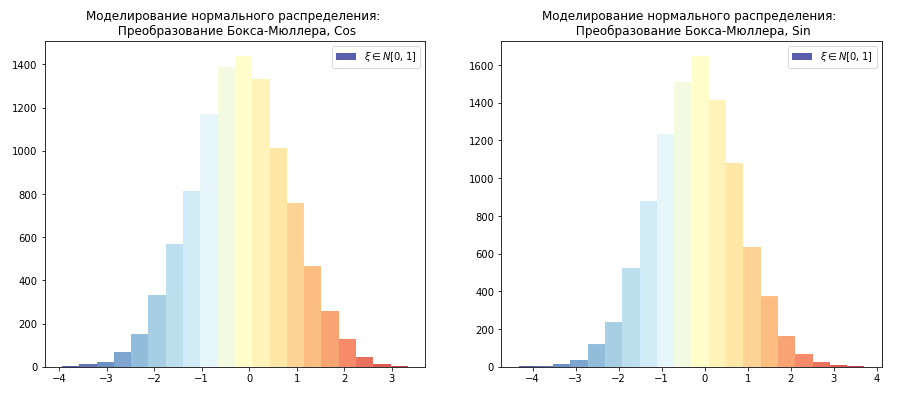


Рисунок 18 – преобразование Бокса-Мюллера для моделирования нормально распределённой случайной величины

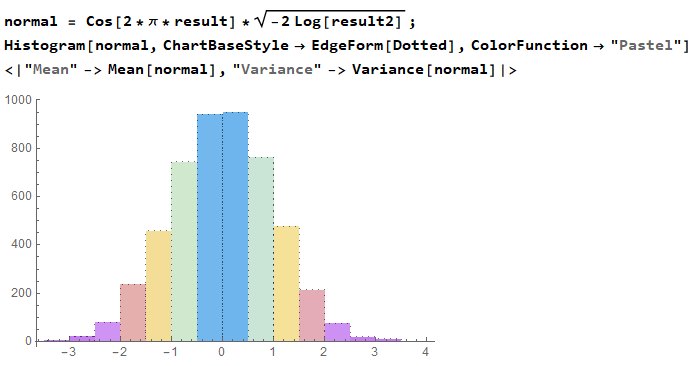


Рисунок 19 – преобразование Бокса-Мюллера для моделирования нормально распределённой случайной величины

**3.4.2 Преобразование через центральную предельную теорему**

Ещё один способ моделирования нормальной распределённой случайной величины заключается в применении центральной предельной теоремы. Освежим её формулировку.

Классическая центральная предельная теорема утверждает, что сумма независимых одинаково распределённых случайных величин имеет распределение, близкое к , что эквивалентно:

Соответственно, чтобы сгенерировать нормально распределённую случайную величину необходимо выбрать случайным образом из получившейся равномерно распределённой выборки подмножество длиной, меньшей и посчитать следующую статистику:

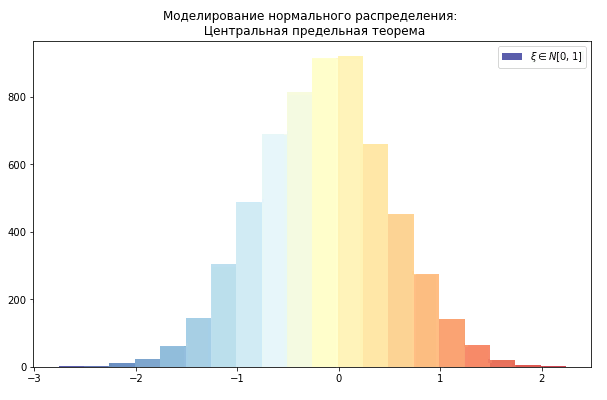


Рисунок 20 – моделирование нормально распределённой случайной величины с помощью центральной предельной теоремы

**3.5 Статистическое моделирование. Метод Монте-Карло**

**Статистическое моделирование** – это машинное воспроизведение функционирования вероятностных моделей, исследование детерминированных процессов, заданных в виде математических моделей с логическими элементами с помощью статистических испытаний на ЭВМ.

*Особенность статистического моделирования –* случайное задание исходных данных с известными законами распределения и вероятностное оценивание характеристик исследуемых процессов.

Имитацию используют тогда, когда другие методы применить невозможно.

Рассмотрим метод Монте-Карло на примере вычисления интеграла, значение которого аналитическим способом найти не удаётся.

**3.5.1 Интегрирование с помощью метода Монте-Карло**

Найдём значение интеграла с помощью метода Монте-Карло:

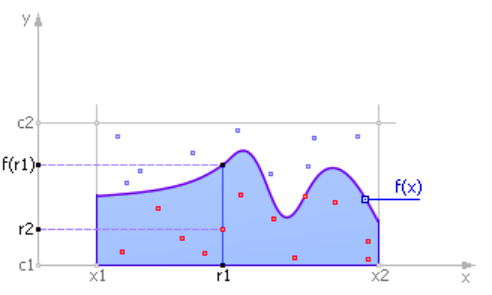


Рисунок 21 – определение значения интеграла методом Монте-Карло

Ограничим кривую сверху, справа и слева. Вычислить значение интеграла функции – найти площадь под графиком. Случайным образом распределяем точки в прямоугольнике поиска. Обозначим за через количество точек, принятых для испытаний (попавших в прямоугольник). За – количество точек под кривой, попавших в закрашенную площадь под функцией (точки изображены красным цветом). Тогда количество точек, попавших под кривую по отношению к общему числу точек пропорционально площади под кривой (величине интеграла) по отношению к площади испытуемого прямоугольника:

Рассуждения верны, чем большее число испытуемых точек мы возьмём.

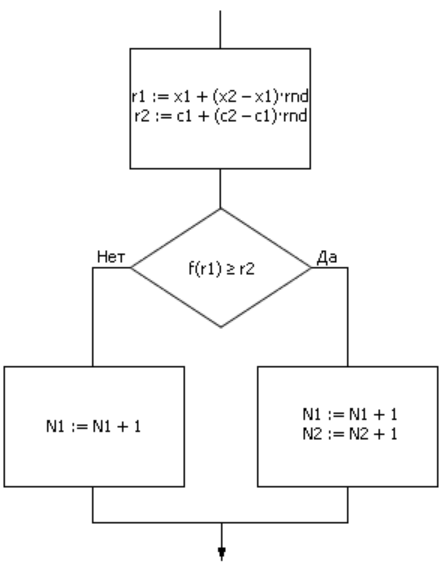


Рисунок 22 – алгоритм реализации метод Монте-Карло

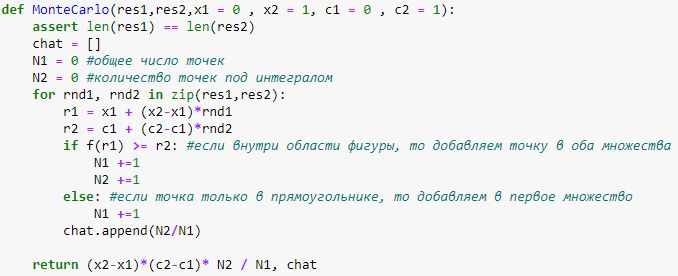


Рисунок 23 – реализация метода Монте-Карло в Python



Рисунок 24 – вычисление интеграла методом Монте Карло и соответствующий график зависимости отклонения от количества элементов в выборке

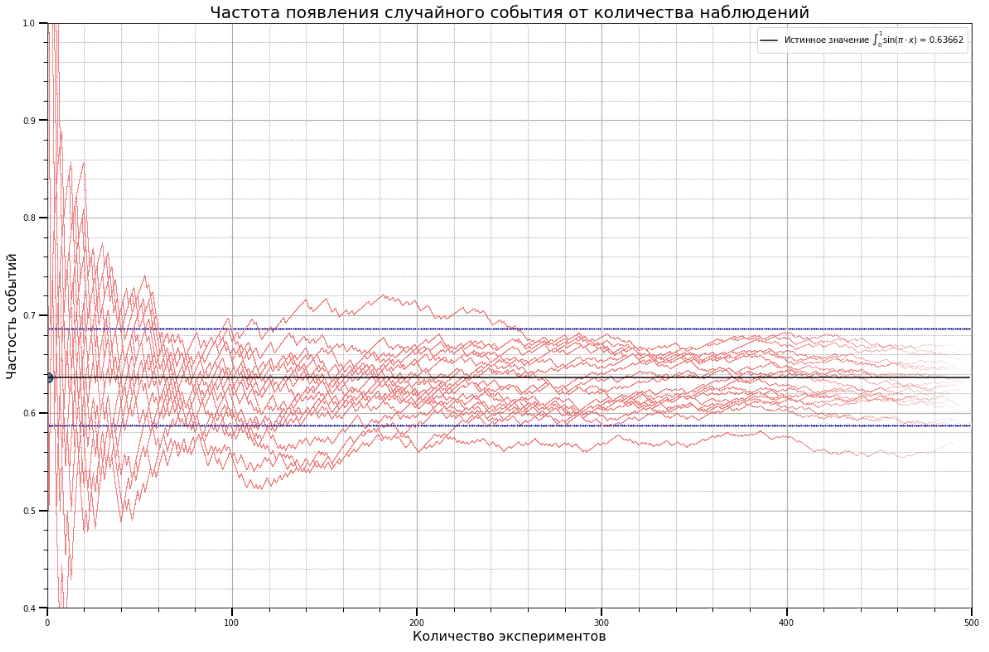


Рисунок 25 – иллюстрация скорости схождения экспериментально получаемой частости к теоретической вероятности

**3.5.2 Классический эксперимент Полиа**

В чёрном ящике находится один белый и один чёрный шарик. Вынимается случайно из коробки один шарик и кладётся в ящик 2 шарика того цвета, который мы вынули. Требуется определить с помощью метода Монте-Карло, в каком отношении делятся чёрные и белые шарики в зависимости от количества экспериментов. (14.09.2020)

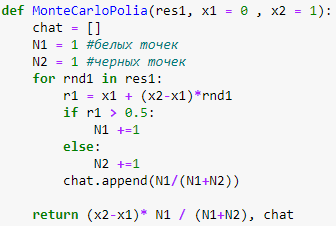


Рисунок 26 – реализация классического эксперимента Полиа

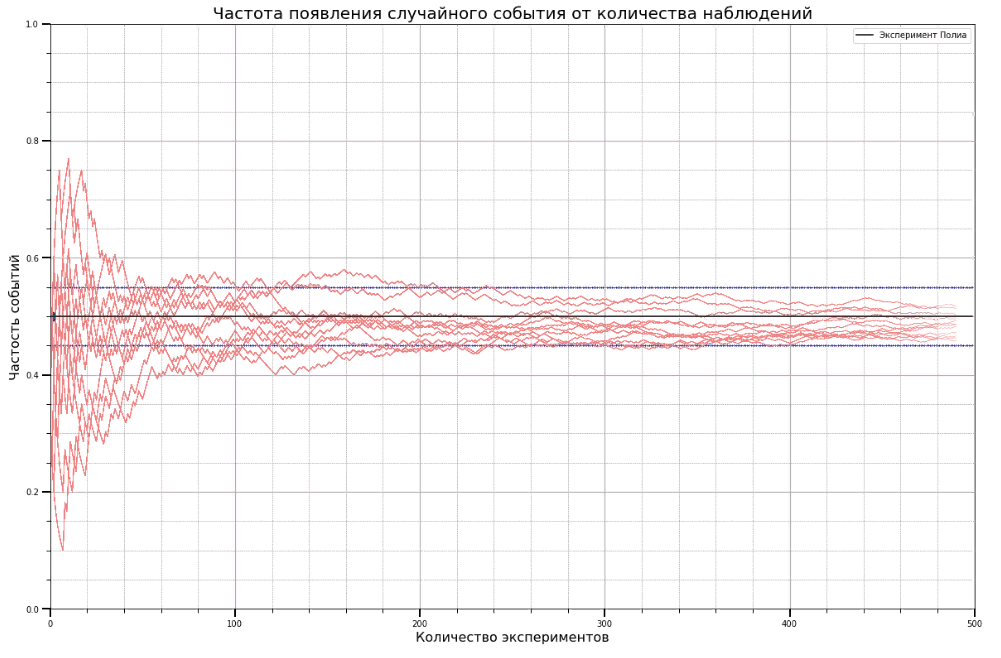


Рисунок 27 – частота появления события, смоделированная с помощью метода Монте-Карло. Видно, что в конечном счёте отношение стремится к равенству белых и чёрных шариков

**3.5.3 Моделирование выигрыша при игре в карты**

Нам предлагают сыграть в игру. Если мы угадываем цвет карты, то мы получаем выигрыш , если же мы не угадываем, то начальный проигрыш увеличивается в 2 раза. Требуется узнать, каково математическое ожидание выигрыша такой игры?

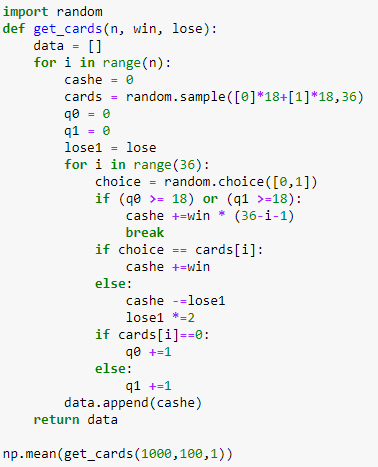


Рисунок 28 – реализация моделирования данной игры – мы проигрываем огромную сумму

**3.6 Генерация случайных событий**

Случайное событие подразумевает, что у некоторого события есть несколько исходов и то, который из исходов произойдёт в очередной раз, определяется только его вероятностью.

**4. Имитационное моделирование**

Имеются три подхода в имитационном моделировании:

* системная динамика
* дискретно-событийное моделирование
* агентное моделирование

**4.1 Процессное (дискретно-событийное) моделирование**

Дискретно-событийное моделирование предполагает представление моделируемой системы в виде процесса, то есть, последовательности операций. Джоном Гордоном было разработано программное обеспечение GPSS.

**Модельное время** – необходимость представления параллельных по времени процессов в виде последовательного алгоритма приводит к введению глобальной переменной , называемой системным модельным временем.

При разработке ИМ рассматривают *реальное, модельное и машинное* время, отражающее затраты времени для проведения имитации.

Возможны следующие *способы изменения* модельного времени:

* изменение МВ с **постоянным шагом**. При имитации первым способом интервал времени моделирования разбивается на интервалы фиксированной длины ;
* для реализации принципа фиксированных интервалов ММ имитируемой системы должна быть преобразована к виду, допускающему рекуррентные вычисления.

**4.2 Моделирование систем массового обслуживания**

**4.2.1 Моделирование СМО банка в Python**

Реализуем одноканальную систему массового обслуживания с отказами, в которую поступают заявки через промежуток времени , где – случайная величина с равномерным законом распределения. Время обслуживания заявки является случайной величиной с показательным законом распределения.



Рисунок 29 – реализация СМО в Python

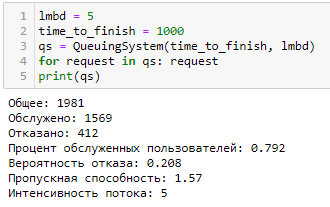


Рисунок 30 – результаты моделирования системы массового обслуживания

**4.2.2 Моделирование СМО банка в AnyLogic**

Построение модели происходит в соответствии с инструкцией, выданной заранее. Было необходимо вывод статистики доли времени работы и простоя обработчика, средней длины очереди и среднего времени ожидания. В результате получилась следующая модель:

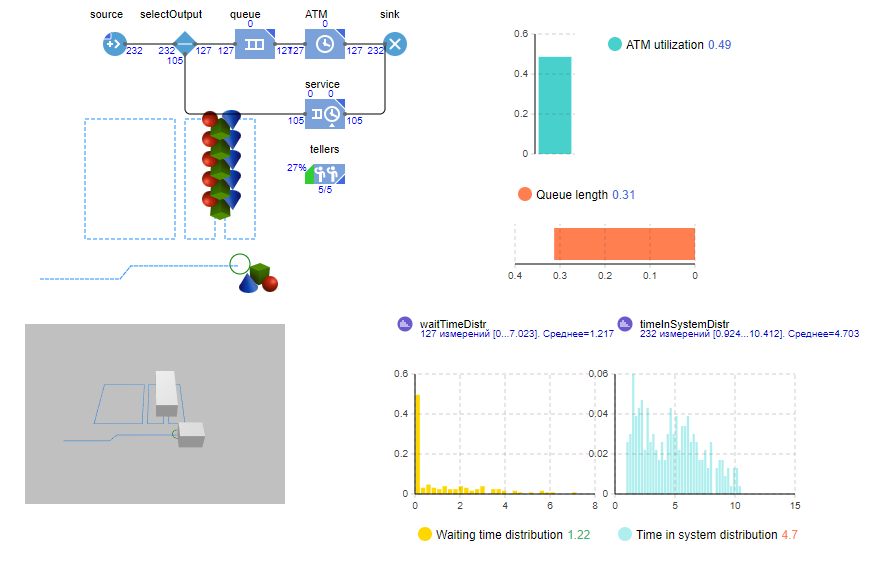


Рисунок 31 – реализация СМО с очередью в AnyLogic

**4.3 Работа с презентациями в AnyLogic**

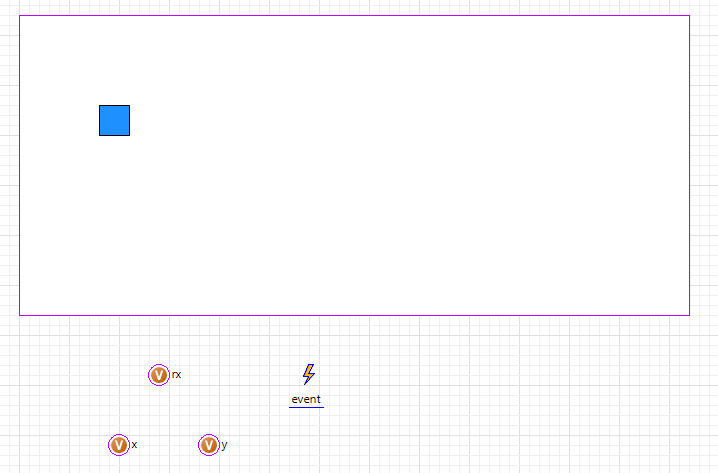


Рисунок 32 – макет реализации движения по окружности

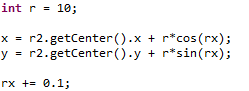


Рисунок 33 – реализация движения по окружности в AnyLogic

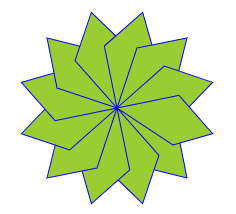


Рисунок 34 – моделирование 12-ти одинаковых фигур с помощью кода

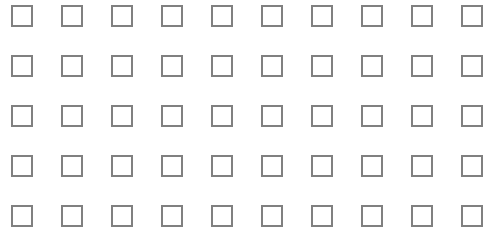


Рисунок 35 – реализация выстроенных в ряд прямоугольника с помощью index



Рисунок 36 – код для построения данной фигуры

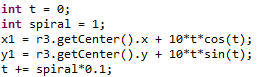


Рисунок 37 – код для движения по спирали

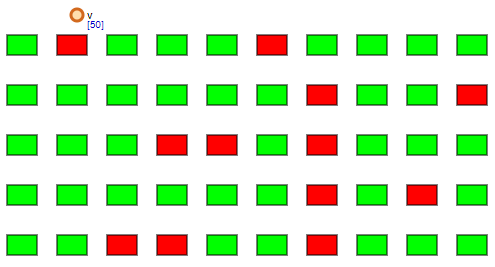


Рисунок 38 – генерация цвета прямоугольника в зависимости от значения случайной переменной

Цвет заливки: .

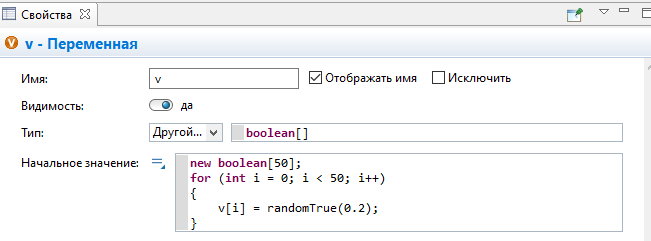


Рисунок 39 – массив в AnyLogic со случайным значением (0,1)

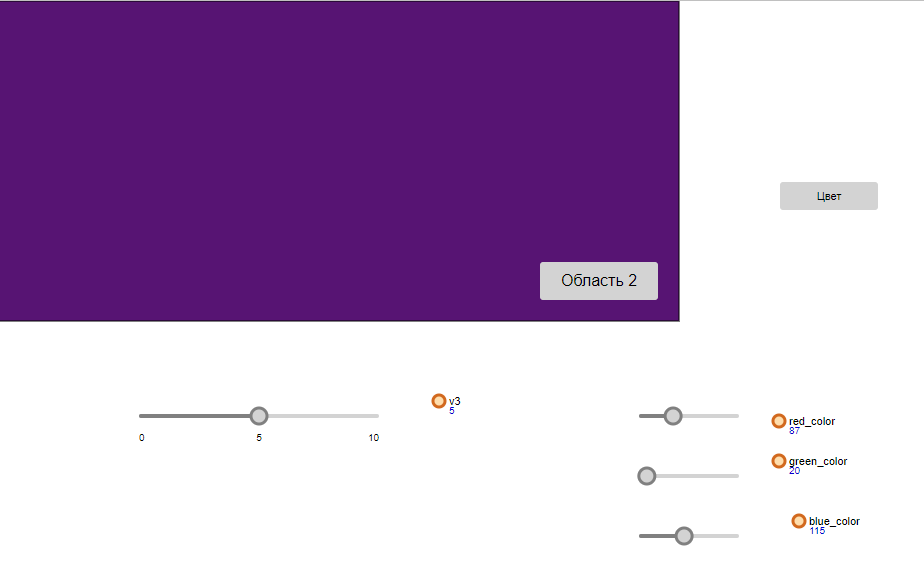


Рисунок 40 – изменение цвета прямоугольника слайдерами RGB

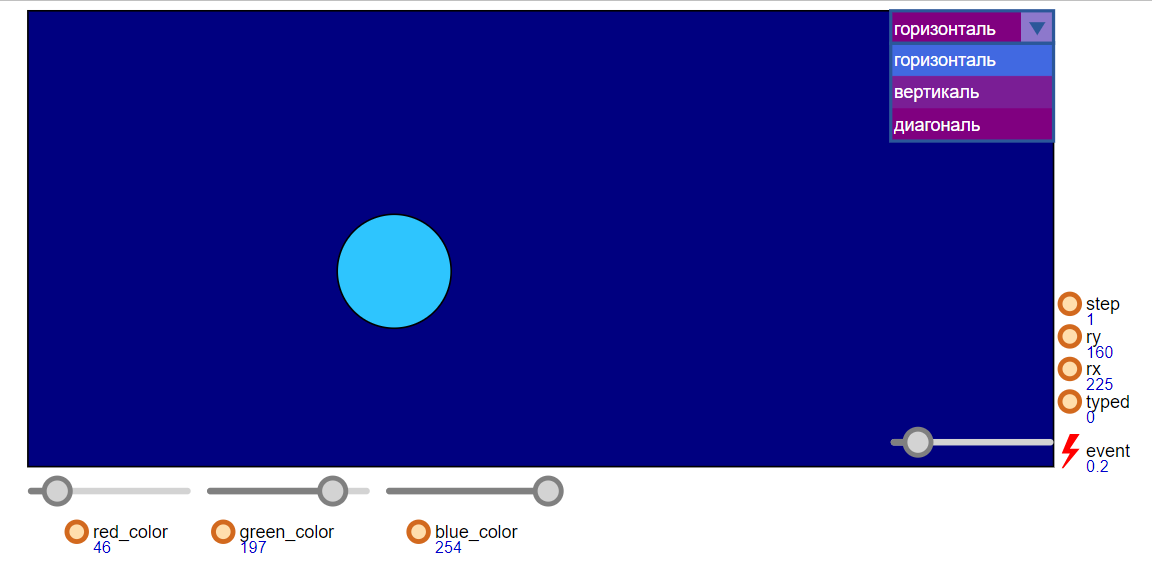


Рисунок 41 – реализация выбора движения: горизонталь, вертикаль, диагональ; цвета круга и скорости движения

**4.4 Моделирование СМО “Call Center” с отказами**

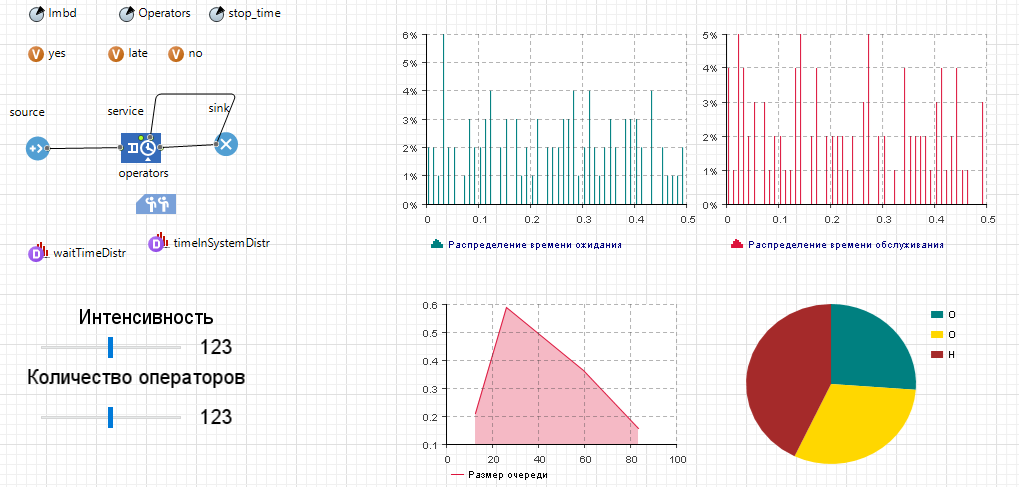


Рисунок 41 – модель СМО Call Center в AnyLogic

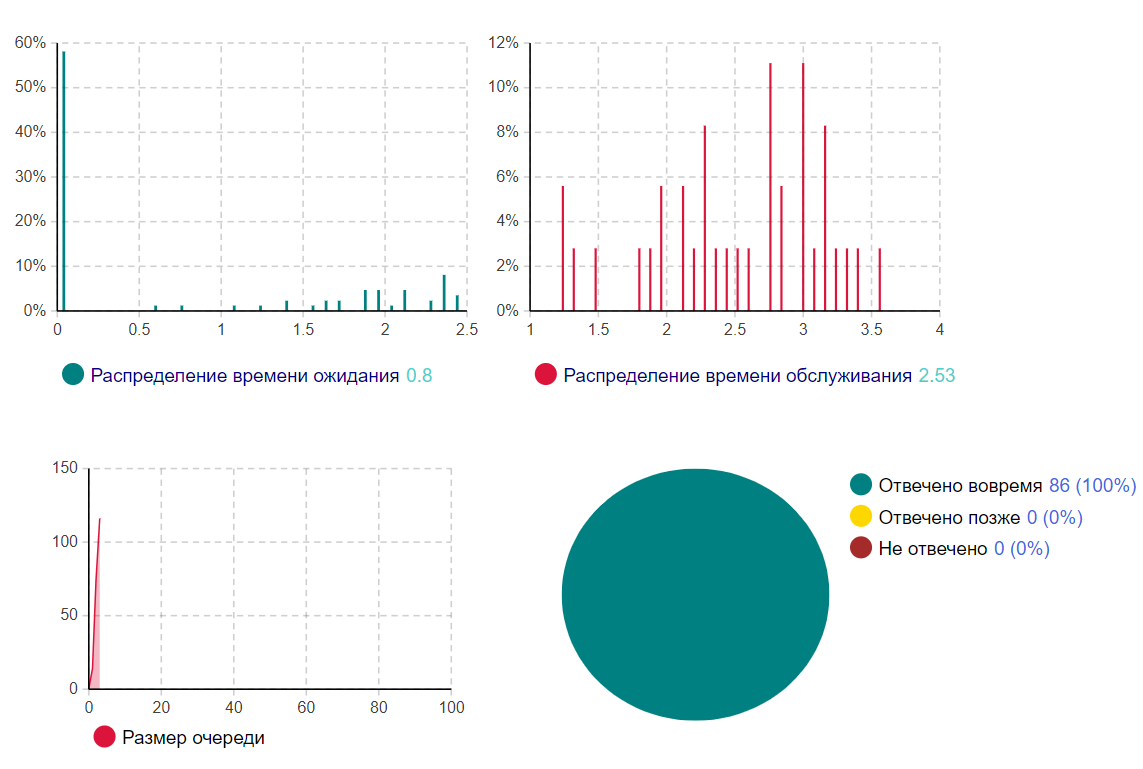


Рисунок 42 – описательные статистики моделирования СМО

**4.5 Моделирование процесса производства мороженого**

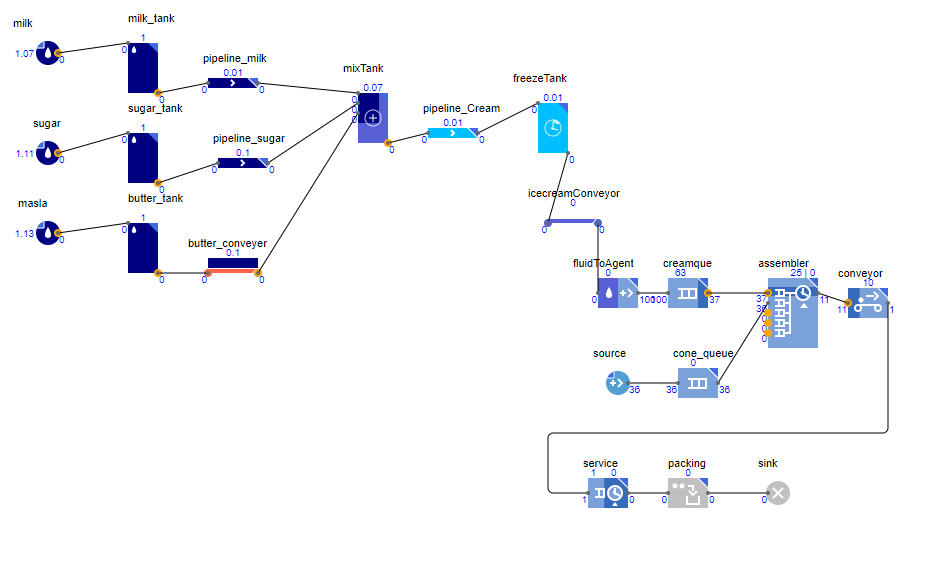


Рисунок 43 – финальная модель производства мороженого в AnyLogic

**4.6 Моделирование метро на учебном примере**

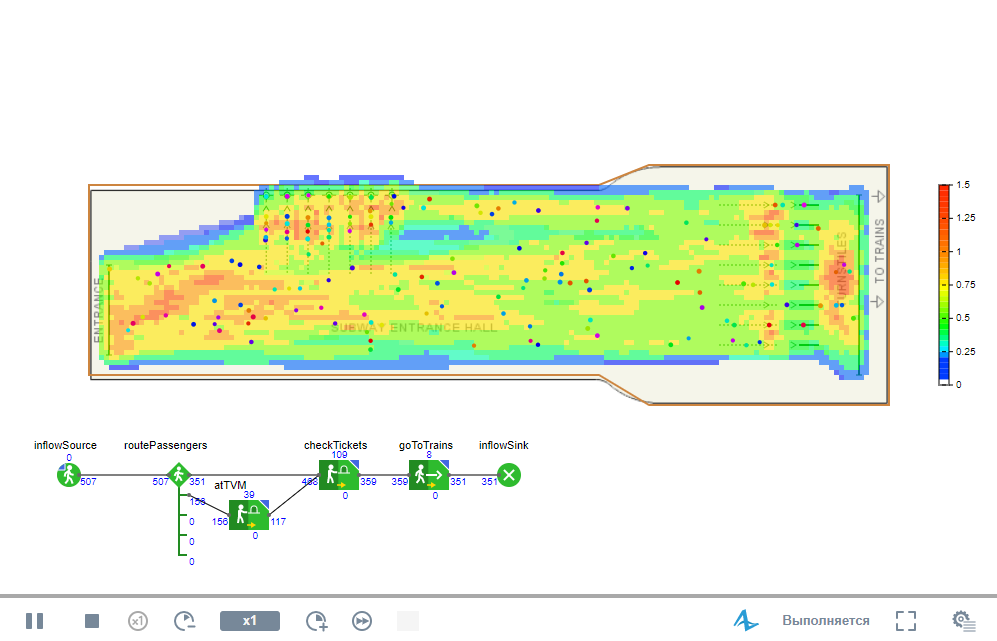


Рисунок 44 – визуализация моделирования метро с помощью AnyLogic

**4.7 Индивидуальное задание: моделирование многоканальной СМО – кинотеатр «Художественный»**

В рамках данного индивидуального задания необходимо было реализовать многоканальную систему массового обслуживания. В качестве СМО был выбран кинотеатр художественный и целью данного эксперимента являлось понимание, возможно ли функционирование кинотеатра в эпоху пандемии. Для этого были сформированы два сервиса – покупка билетов и размещение людей в зале кинотеатра за столами, расположенных на 1.5 метров. Необходимо было определить, при каких значениях параметра интенсивности входного потока людей и при каком значении времени сидения за столом для ожидания фильма кинотеатр может функционировать. Оказалось, что для интенсивности человека в минуту и при среднем времени ожидания около 6 минут, в кинотеатре не возникает очередей и люди спокойно могут размещаться в зале.

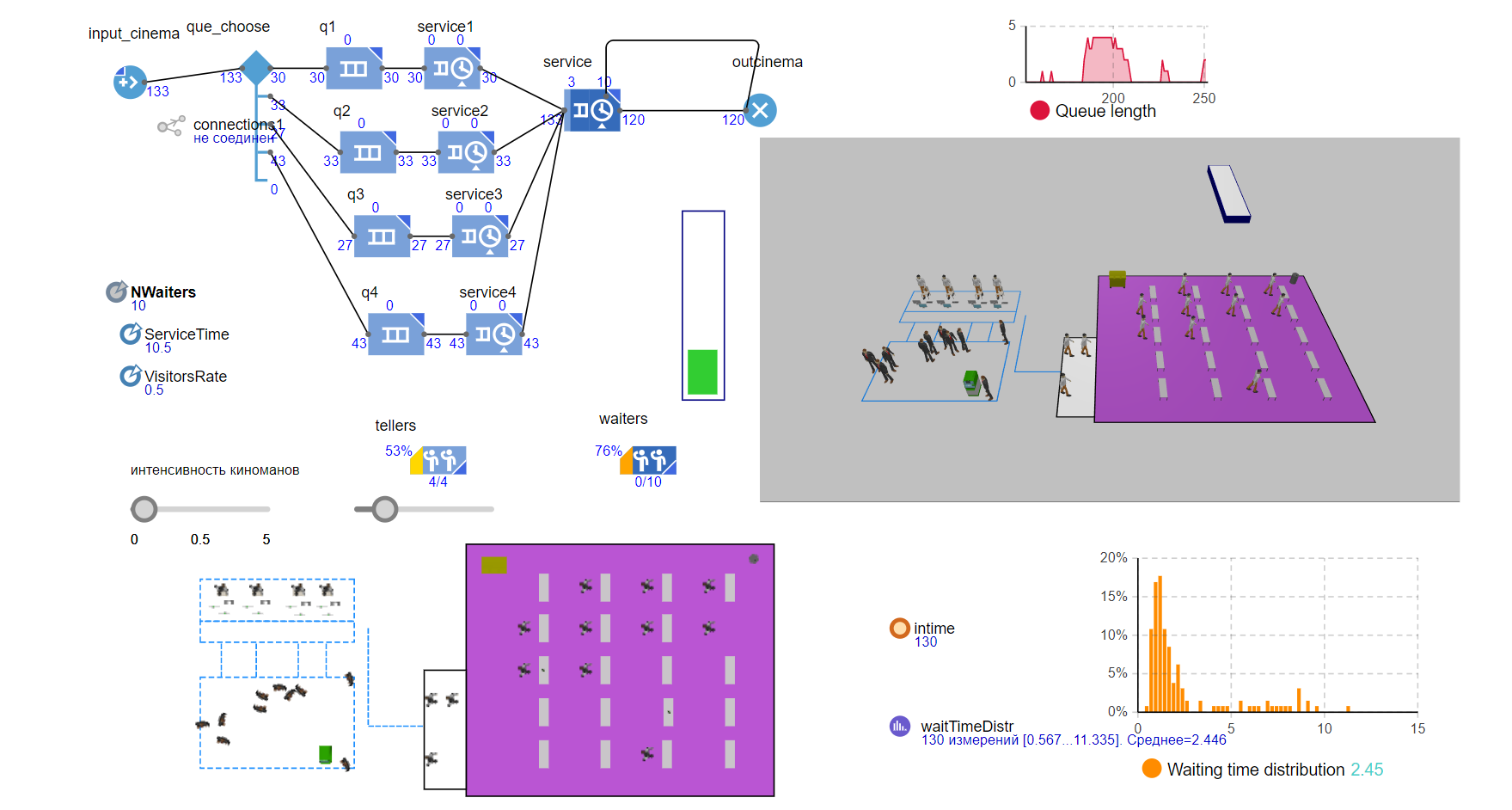


Рисунок 45 – реализация СМО кинотеатра художественный с помощью AnyLogic

**4.8 Индивидуальное задание: моделирование работы станции метро Чернышевская**

В данном задании необходимо было реализовать моделирование и имитацию работы станции метро Чернышевская с помощью библиотеки моделирования потоков, а после выявить наиболее востребованные и нагруженные источники данной станции метро.

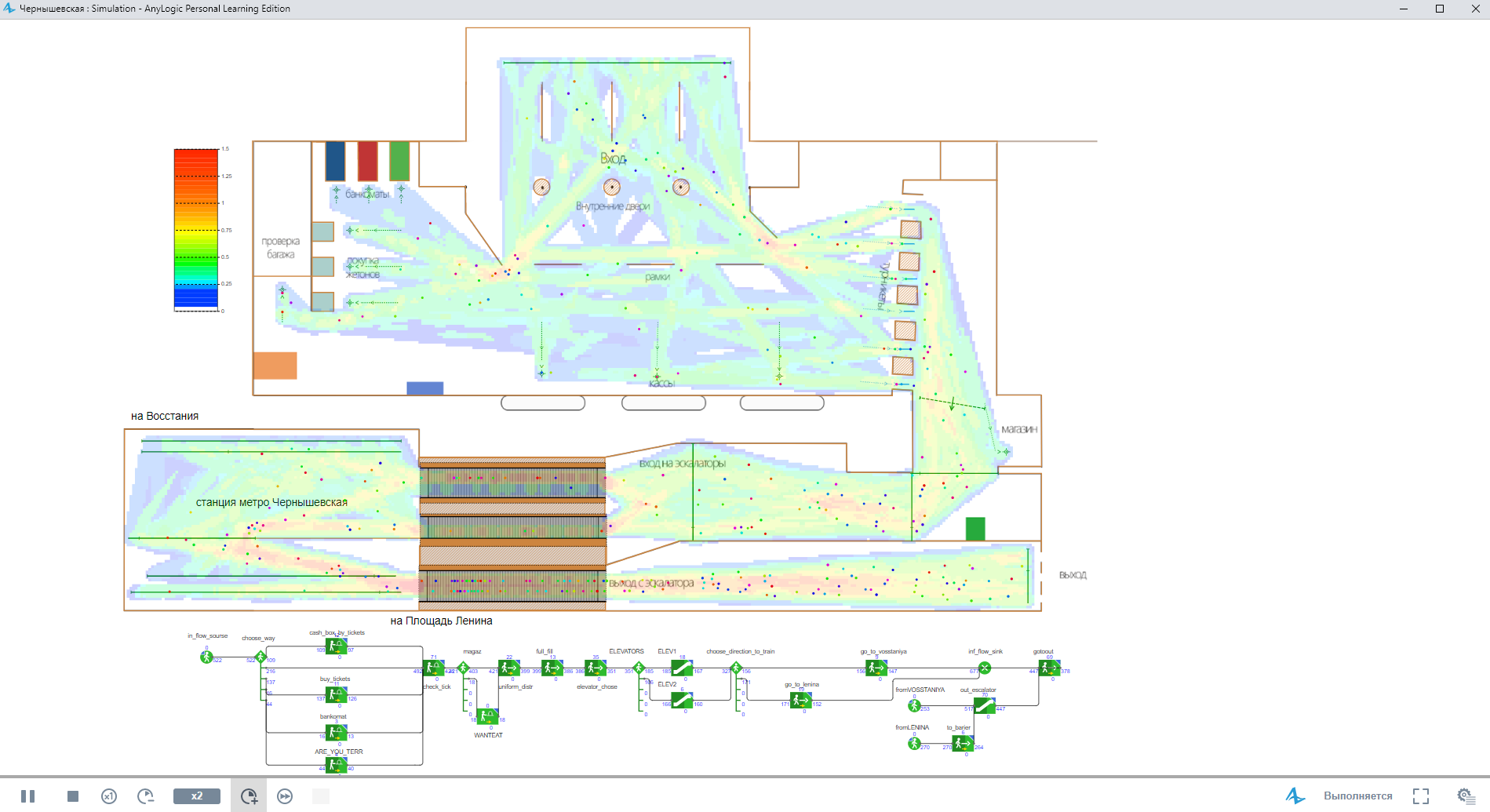


Рисунок 46 – реализация моделирования станции метро Чернышевская с помощью AnyLogic

**4.9 Работа с файлами в AnyLogic**

Необходимо было реализовать сначала сбор статистики некоторой простой системы массового обслуживания в Excel файл, а затем нарисовать фигуры с координатами из другого Excel файла.

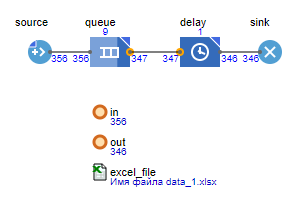


Рисунок 47 – реализация работы с файлами в AnyLogic

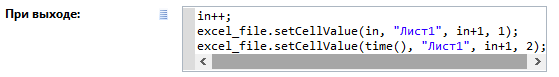


Рисунок 48 – обработка записи в Excel файл

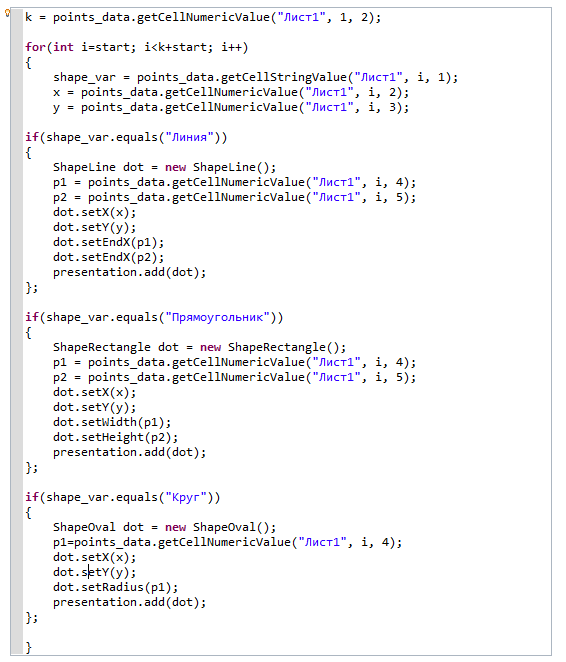


Рисунок 49 – обработка координат фигур из файла

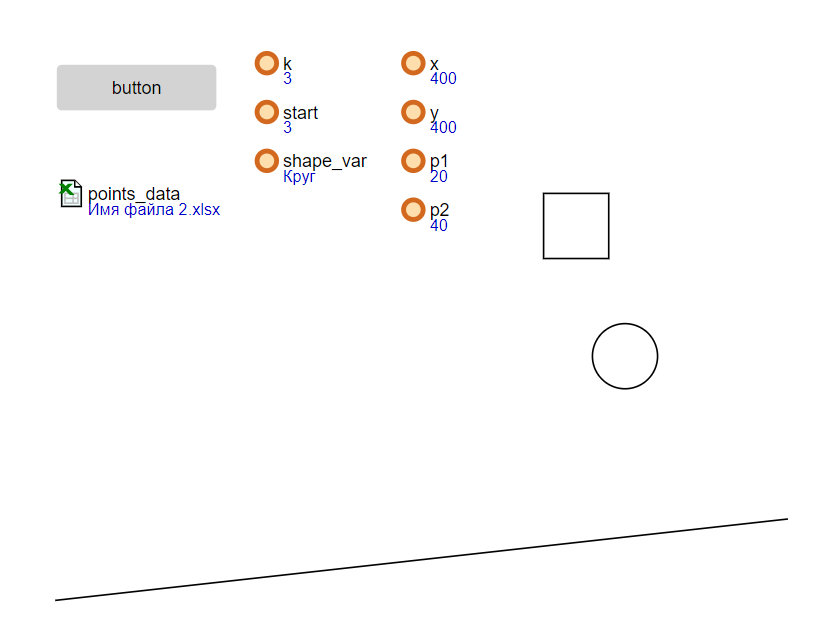


Рисунок 50 – визуализация работы программы

**4.10 Реализация разложения в ряд экспоненты и косинуса**

С помощью блок схем можно последовательно задать наглядную демонстрацию выполнения алгоритма. Необходимо было реализовать блок схемы, которые раскладывали в ряд Тейлора функции косинуса и экспоненты.

Данные диаграммы выглядят следующим образом.

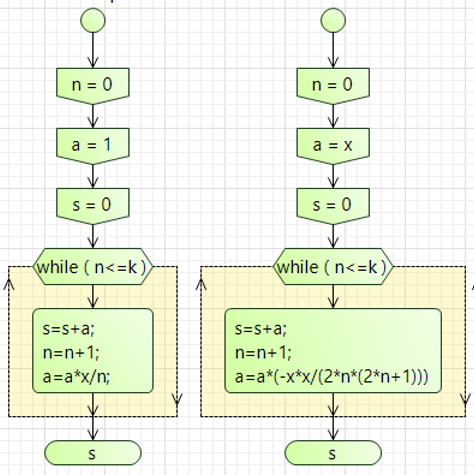


Рисунок 51 – реализация блок-схем с помощью диаграмм действий

**4.11 Модели SIR**

SIR-модель – это модель поведения эпидемий в мезомасштабных сетях, получившая своё название от кратких названий компонентов, входящих внутри данной системы: **S –** количество здоровых (ещё не болевших) в момент времени , **I –** количество больных в момент времени ,  **–** количество иммунитетных к болезни в момент времени .

Пусть общее количество человек фиксировано:

Данная модель описывается системой дифференциальных уравнений:

где – вероятность заразиться здоровому при контакте с больным, – доля выздоровевших/умерших от болезни ( – среднее время выздоровления).

В данной системе дифференциальных уравнений предполагается, что все люди равномерно распределены по пространству.

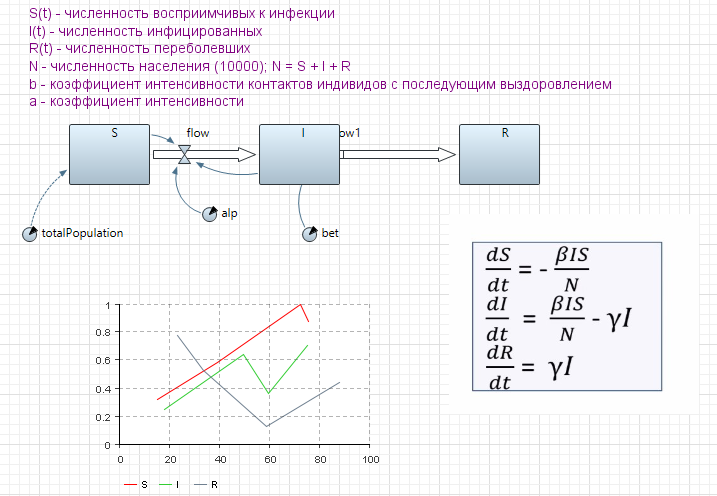


Рисунок 52 – реализация базовой модели SIR

Модель SIER записывается следующим образом:

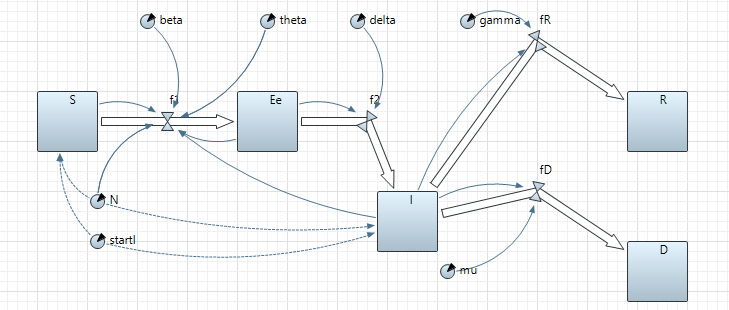


Рисунок 53 – реализация модели SIER

**5 Агентные модели в дискретном пространстве**

**5.1 Жизнь Конвея**

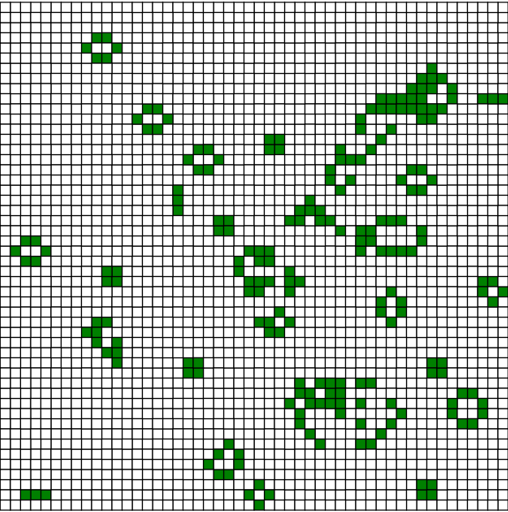


Рисунок 54 – реализация жизни Конвея в AnyLogic

**5.2 Модель Шеллинга**

Модель поведения агента в данной реализации следующая:

* агент, который имеет только одного соседа, переедет, если этот сосед другого цвета
* агент, имеющий двух соседей, не будет переезжать, если хотя бы один из них того же цвета, что и он
* агент, проживающий по соседству от трёх до пяти человек, не будет переезжать, если хотя бы два и них будут его цвета
* агент с соседями от шести до восьми человек не будет переезжать, если хотя бы три из них будут одного цвета

Шеллинг программировал своих агентом таким образом, чтобы каждый житель удовлетворён, если хотя бы 37.5 его соседей принадлежат к тому же классу.

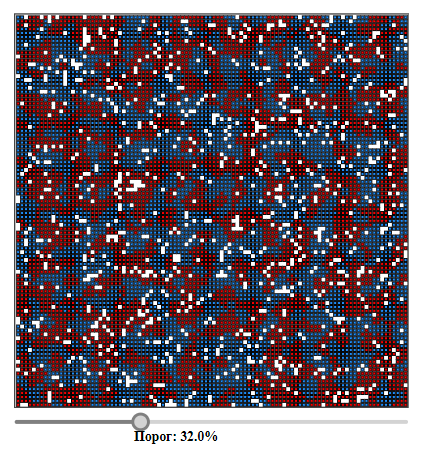


Рисунок 55 – реализация модели Шеллинга, порог 32 процента

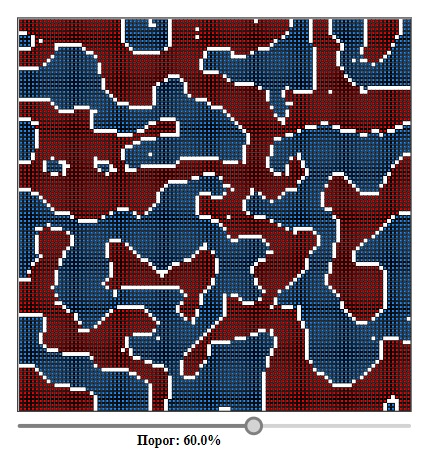


Рисунок 56 – порог 60 процентов – формируются общины

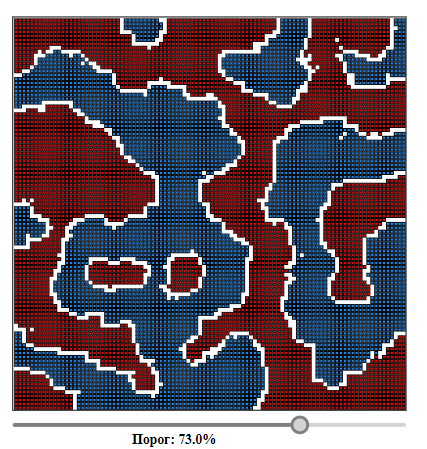


Рисунок 57 – порог 73 процента