МИНИСТЕРСТВО ОБРАЗОВАНИЕ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ ИМ. В. Н. КАРАЗИНА

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК

**КУРСОВАЯ РАБОТА**

**ПО КУРСУ «ЯЗЫКИ ПРИКЛАДНОГО ПРОГРАММИРОВАНИЧ»**

**НА ТЕМУ: «МОДЕЛИРОВАНИЕ БИНОМИАЛЬНОГО РАСПРЕДЕЛЕНИЯ»**

Выполнил

студент КС-41

Мишенин А.С.

Проверил

преп. Рудичев Д.В.

ХАРЬКОВ – 2017

Оглавление

[Введение 2](#_Toc502704229)

[Теоретические сведенья 3](#_Toc502704230)

[Используемые методы моделированием случайной величины 5](#_Toc502704231)

[Алгоритмы и их реализация 8](#_Toc502704233)

[Требования к программному продукту 14](#_Toc502704235)

[Особенности программной реализации 15](#_Toc502704236)

[Инструкция по эксплуатации 16](#_Toc502704237)

[Заключение 23](#_Toc502704238)

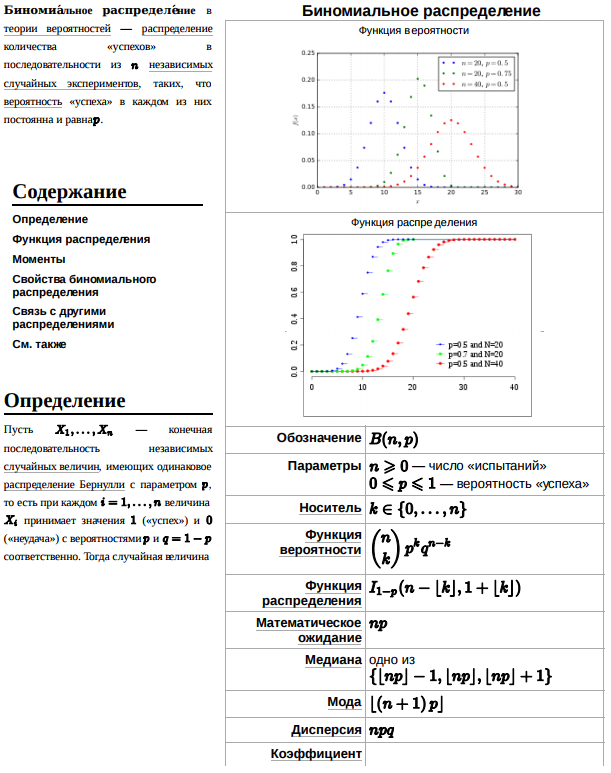
[Список литературы 24](#_Toc502704239)

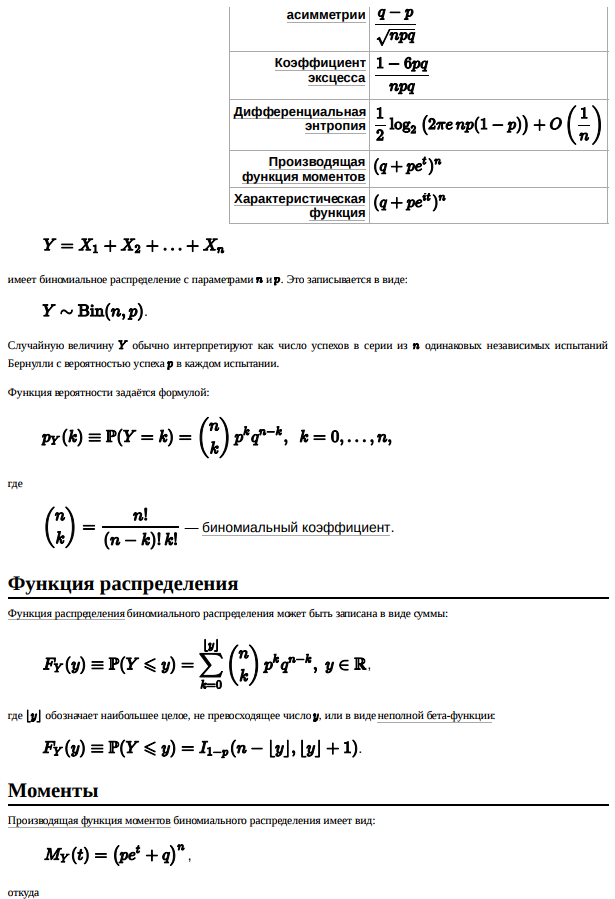
# Введение

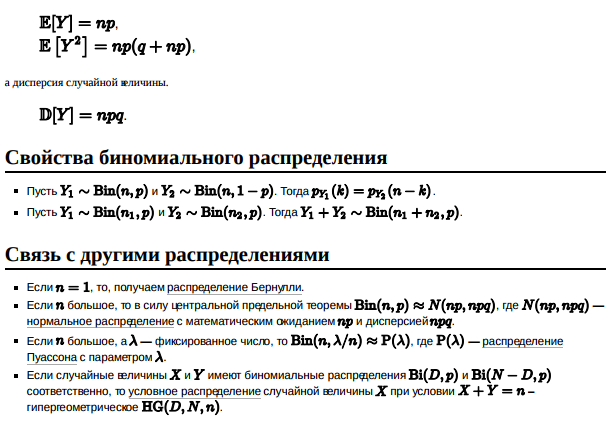
Цель работы – получение практических навыков в разработке программного обеспечения с применением генератора случайных величин, распределенных по заданному распределению, оформление документации, а также закрепление и контроль знаний и навыков, полученных во время изучения курса «Компьютерное моделирование стохастических процессов»,

В данной работе освещены некоторые теоретически сведенья о биномиальном распределении, а также некоторых методах прям его моделирования. А именно: метод обратных функций, метод Неймана, метод Метрополиса. Разработаны алгоритмы реализации данных методов. Предложена программа, выполняющая генерацию случайной величины с биномиальным распределением, описан порядок работы с программой, подведены выводы о проделанной работе.

# Теоретические сведенья







# Используемые методы моделированием случайной величины

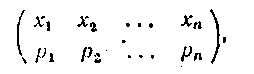
Метод обратных функций

Пусть непрерывная случайная величина (СВН) задана своим законом распределения:



, где – плотность распределения вероятностей, а - функция распределения вероятностей. Доказано, что случайная величина распределена равномерно на интервале (0,1). Отсюда следует, что искомое значение **y** может быть определено из уравнения: которое эквивалентно уравнению: где *y* – значение случайной величины *.* Решение уравнения можно записать в общем виде через обратную функцию  Основной недостаток метода заключается в том, что интеграл не всегда является берущимся, а уравнение не всегда решается аналитическими методами.



Частным случаем реализации метода обратных функций является метод интервалов, который используется для моделирования дискретных случайных величин.

**(1)**

Рассмотрим дискретную случайную величину ξ с рас­пределением (1), где *pi =* P{ξ = *xi*}. Для того чтобы вычислить значе­ния этой величины разделим интервал 0 ≤ у < 1 на интервалы Δi такие, что дли­на Δi фавна *рi .*

*Случайная величина ξ, опреде­ленная формулой ξ = xi*, *когда* γ∈Δi , (2)*имеет распределение вероятностей* (1).

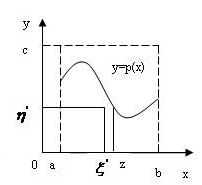
*Доказательство* занимает одну строку:

P{ξ = *xi*} =Р{γ∈Δi } = длина Δi = рi.

Для практической реализации формулы (2) удобно в накопителе ЭВМ расположить подряд значе­ния х1, x2, ..., xn и p1, p1+p2, p1+p2+p3, ..., 1. Для того чтобы вычислить очередное значение ξ, находим очередное γ. Затем сравниваем γс p1*.* Если γ< p1, то ξ = x1; если γ≥ p1, то сравниваем γ с p1+p2.Если γ < p1+p2, то ξ = x2; если γ ≥ p1+p2, то сравниваем γ с p1+p2+p3, и т.д.

Метод Неймана

Метод Неймана, так же как метод обратной функции, является методом, позволяющим получить значения СВ в соответствии с заданным законом распределения. Этот метод является достаточно универсальным он применим для моделирования всех СВ, значения которых не выходят за пределы ограниченного интервала (a,b), а также для СВ, законы распределения которых можно аппроксимировать усеченными.

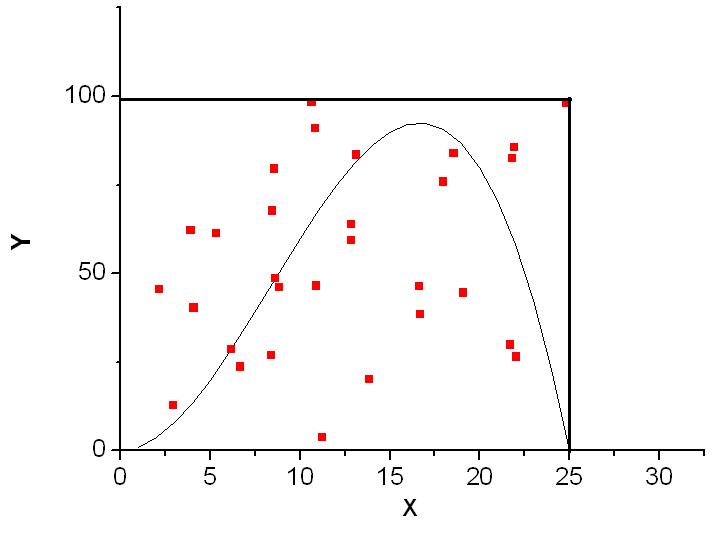
Рассмотрим случайную величину , определенную на конечном интервале  с ограниченной плотностью  (рис). Пусть независимые случайные числа и Случайная величина , определенная условием  имеет плотность вероятностей, равную 

Эффективность метода равна вероятности попадания точи  под кривую  т.е.  Эффективность метода Неймана будет наибольшей, если выбрать наименьшее возможное  т.е. положить  Впрочем, это очевидно также из геометрических соображений.

Другими словами, с геометрической точки зрения алгоритм можно определить так:

* ограничим функцию прямоугольником (*n*-мерным параллелепипедом в случае многих измерений), площадь которого можно легко вычислить; *любая сторона прямоугольника содержит хотя бы 1 точку графика функции, но не пересекает его;*



* «набросаем» в этот прямоугольник (параллелепипед) некоторое количество точек ( штук), координаты которых будем выбирать случайным образом;



* определим число точек ( штук), которые попадут под график функции;



* площадь области, ограниченной функцией и осями координат, даётся выражением



Большие промежутки между фактическим графиком функции и зоной, ограничивающей его, могут привести к значительному снижению эффективности функции. Для повышения эффективности используется модернизация метода Неймана: функцию приближают сверху линейной, либо более высоких порядков, функцией.

Метод Метрополиса

Алгоритм Метрополиса генерирует случайное блуждание точек, распределенных в соответствии с требуемым распределением вероятностей. При этом значения случайной величины не должны выходить за значения интервала (a,b), т.е. вероятность выпадения СВ за пределами интервала=0. На каждой итерации алгоритма принимается значение СВ.

На интервале произвольным образом выбирается начальная точка х0. Для абстрагирования от этого конкретного значения, алгоритм запускают на n-ное количество шагов «вхолостую». На каждом шаге алгоритма новое выбранное значение зависит только от предыдущего . Для получения нового значения к прибавляют новое значение шага , которое может иметь как положительный, так и отрицательный знак и по модулю не превышает длину отрезка (a,b). Если впоследствии этих преобразований точка покинула границы интервала, принимается СВ (=). Иначе проделывается проверка следующих условий.



if (p() < p()) then



=



elseif < p()/p() then



=



else

=



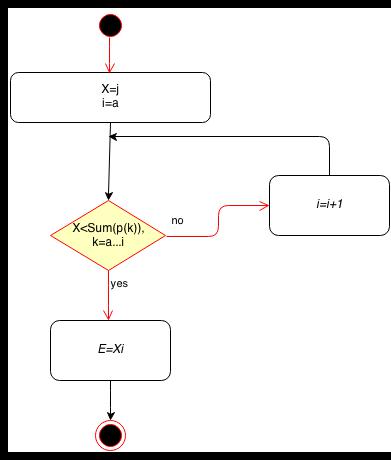
end if

# Алгоритмы и их реализация

Метод обратных функций

Логорифмическое распрделение принадлежик к классу дисретных распределений, поэтому для реализации метода обратных функций используется метод интервалов, описанный в разделе *Используемые методы моделированием случайной величины*.

Алгоритм моделирования СВ методом интервалов:



Обозначения: j- СВ, подчиняющаяся равномерному закону распределения,

Е – СВ, подчиняющаяся искомому закону распределения

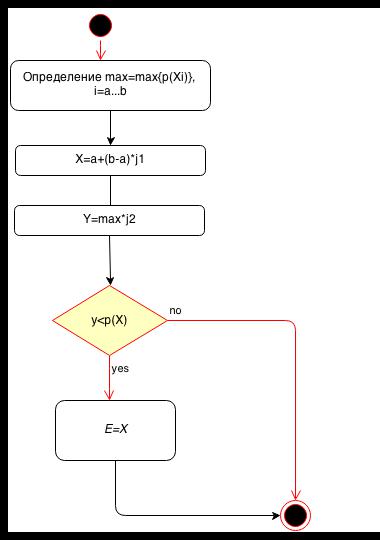
Программная реализация:

1. **const** factorial = (n) => n === 0 ? 1 : n \* factorial(n - 1);
2. **const** binomial = (n, k, p) => (factorial(n) / (factorial(k) \* factorial(n - k))) \* Math.pow(p, k) \* Math.pow(1 - p, n - k);
4. module.exports.inverseTransformSampling = (event, context, callback) => {
5. **const** {n, p} = event;
6. **const** target = Math.random();
7. let current = 0;
8. let k = 0;
9. while ((target >= current) && (k <= n)) {
10. current += binomial(n, k, p);
11. k++;
12. }
14. **const** response = {
15. statusCode: 200,
16. body: JSON.stringify({
17. value: k
18. }),
19. };
21. callback(**null**, response);
22. };

Метод Неймана

Для реализации метода Неймана используется ограничение максимальным значением функции вероятности распределения. Больше о методе в разделе *Используемые методы моделированием случайной величины.*

Алгоритм моделирования СВ методом Неймана:



Обозначения: j1,j2- СВ, подчиняющиеся равномерному закону распределения,

Е – СВ, подчиняющаяся искомому закону распределения

а- левая граница интервала, b-правая граница интервала

max – максимальное значение функции вероятности искомого распределения

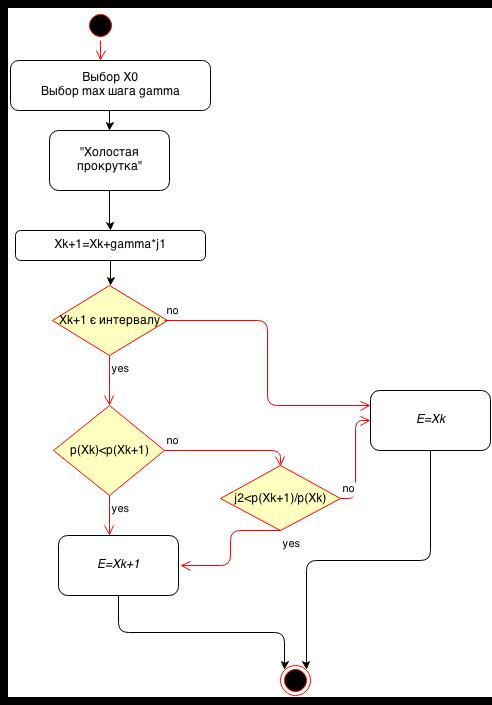
p(Xi)-вероятность выпадение СВ Xi

Программная реализация:

1. **const** factorial = (n) => n === 0 ? 1 : n \* factorial(n - 1);
2. **const** binomial = (n, k, p) => (factorial(n) / (factorial(k) \* factorial(n - k))) \* Math.pow(p, k) \* Math.pow(1 - p, n - k);
4. module.exports.neumannMethod = (event, context, callback) => {
5. **const** {n, p} = event;
6. **const** c = binomial(n, Math.ceil(n / 2), p);
7. let point = {
8. x: **null**,
9. y: **null**
10. };
12. **do** {
13. point.x = Math.round(Math.random() \* n);
14. point.y = Math.random() \* c;
15. } while (point.y >= binomial(n, point.x, p));
17. **const** response = {
18. statusCode: 200,
19. body: JSON.stringify({
20. value: point.x,
21. }),
22. };
24. callback(**null**, response);
25. };

Метод Метрополиса

Больше о методе в разделе *Используемые методы моделированием случайной величины.*

Алгоритм моделирования СВ методом Метрополиса:

Обозначения: j1,j2- СВ, подчиняющиеся равномерному закону распределения,

Е – СВ, подчиняющаяся искомому закону распределения

gamma – раз мер прыжка. Не должен превышать длину отрезка.

p(Xi)-вероятность выпадение СВ Xk

Программная реализация:

1. **const** factorial = (n) => n === 0 ? 1 : n \* factorial(n - 1);
2. **const** binomial = (n, k, p) => (factorial(n) / (factorial(k) \* factorial(n - k))) \* Math.pow(p, k) \* Math.pow(1 - p, n - k);
4. module.exports.metropolisMethod = (event, context, callback) => {
5. **const** {n, p} = event;
6. let current = Math.round(Math.random() \* n);
7. let next = 0;
9. **for** (let i = 0; i < Math.ceil(Math.random() \* 50); i++) {
10. **if** (Math.random() < 0.5) {
11. next = current - 1;
12. } **else** {
13. next = current + 1;
14. }
16. **if** ((next < 0) || (next > n)) **continue**;
17. **else** **if** (binomial(n, current, p) < binomial(n, next, p)) {
18. current = next;
19. } **else** **if** (Math.random() < binomial(n, next, p) / binomial(n, current, p)) {
20. current = next;
21. }
22. }
24. **const** response = {
25. statusCode: 200,
26. body: JSON.stringify({
27. value: current,
28. }),
29. };
31. callback(**null**, response);
32. };

# Требования к программному продукту

Разработанный программный продукт должен реализовывать следующий функционал:

* Программа должна осуществлять генерацию случайной величины с биномиальным распределением вероятности следующими методами: методом обратных функций, методом Неймана, методом Метрополиса, аналитически.
* Результаты текущего расчета должны быть представлены в графическом (график) и интегральном (таблица) виде.
* Результаты расчетов могут быть сохранены в файлах.
* Поля ввода данных должны быть защищены.
* Присутствует возможность прерывания расчета.
* Присутствует индикатор расчета.
* Интерфейс ПП должен быть эргономичен.

# Особенности программной реализации

Выроб языка программирования и среды разработки

Для реализации данной работы в рамках курса Языки прикладного программирования, можно было выбрать один из 3х языков программирования: Ada, Fortran, C#.

Однако, из-за нехватки времени для выполнения работы, мной было принято решение разработать программный продукт, использованием того набора технологий, с которым я очень хорошо знаком: React.js (клиентская часть приложения), AWS Lambda (серверная часть приложения), в качестве дополнительных библиотек, я применил MateriaUI (дизайн фреймворк Material Design), Recharts (для графиков).

В качестве среды разработки, я пользовался Intellij IDEA.

Особенности архитектуры приложения

Вычисление разделено на итерации, после каждой из которых, мы получаем промежуточный результат. При каждой итерации клиентская часть посылает запрос AWS Lambda функции под именем calculate запрос, она в свою очередь вызывает 4 других AWS Lambda функции (каждая из них отвечает за свой метод). Функция calculate собирает результаты, трансформирует их и выдает клиентскому приложению, которое в свою очередь отображает результат.

AWS Lambda “Naumann”

AWS Lambda “Metropolis”

AWS Lambda “Analitic”

AWS Lambda “Inverse Function”

AWS Lambda “Calculate”

Клиентское приложение

Илл. 1. Архитектура приложения.

# Инструкция по эксплуатации

Требования к браузеру

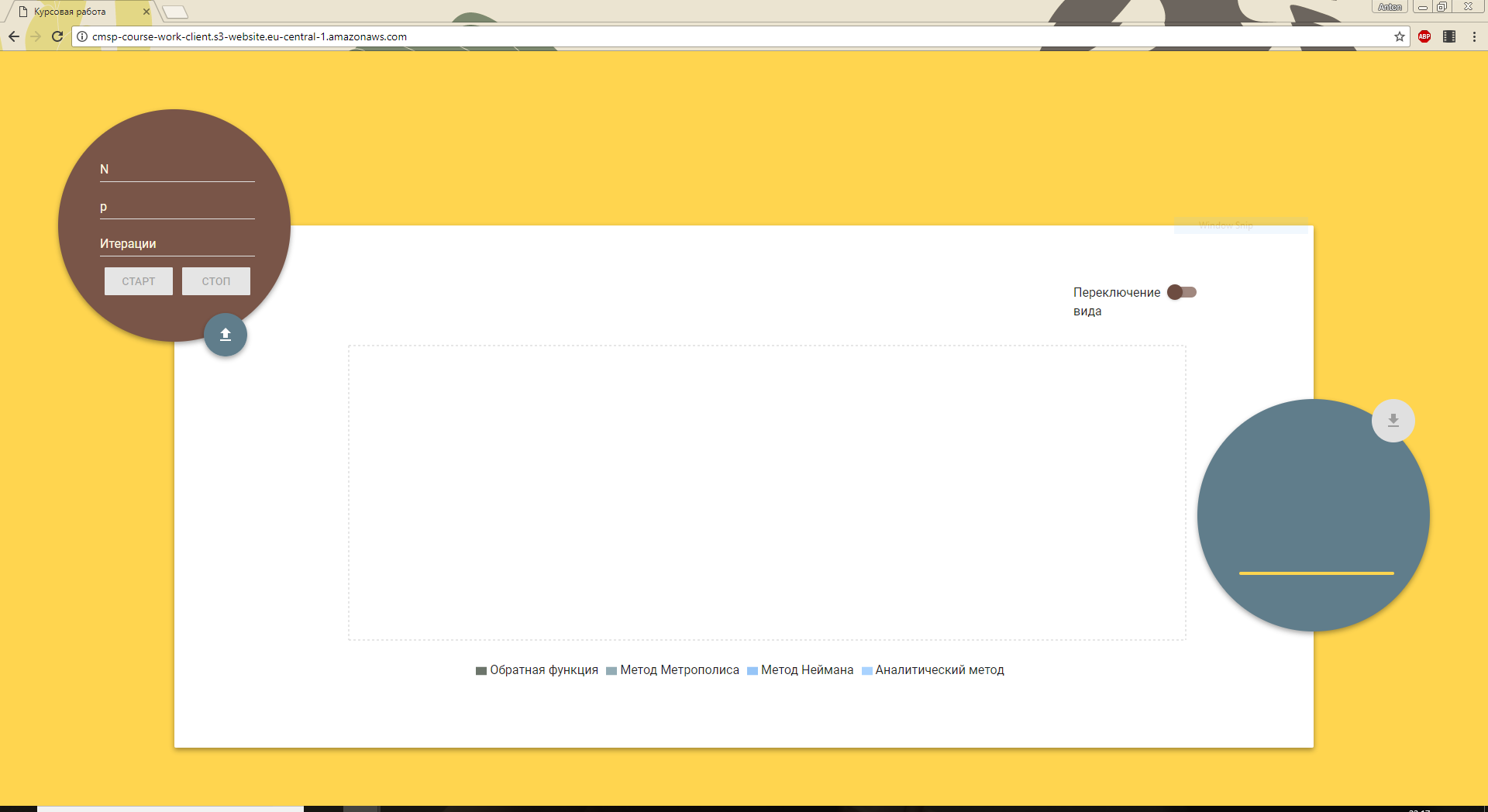
|  |  |
| --- | --- |
| Chrome | > 61 |
| Firefox | > 57 |
| Edge | > 16 |
| Safari | > 10.1 |

Адрес интернет страницы

<http://cmsp-course-work-client.s3-website.eu-central-1.amazonaws.com/>

Описание интерфейса

При входе на интернет страницу, мы видим интерфейс изображённый на рисунке.



3

2

1

Рис. 1. Интерфейс при входе

По своей смысловой нагрузке, интерфейс разделено на три части:

* ввод данных / загрузка файла / старт и стоп вычисления (1)
* отображение результатов вычислений (2)
* вывод характеристик распределения / прогресса вычисления / скачивание файла (3)

1. ввод данных



Кнопки СТАРТ/СТОП

Загрузка файла

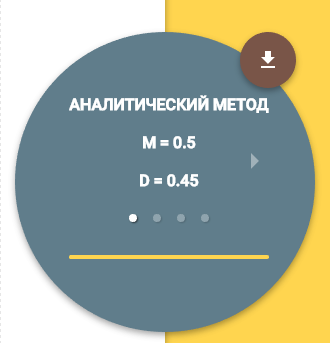
Количество итераций для вычислений

Вероятность успеха, параметр биномиального распределения

Количество испытаний, параметр биномиального распределения

Рис. 2. Панель ввода данных

1. вывод данных о распределении



Кнопка скачивание

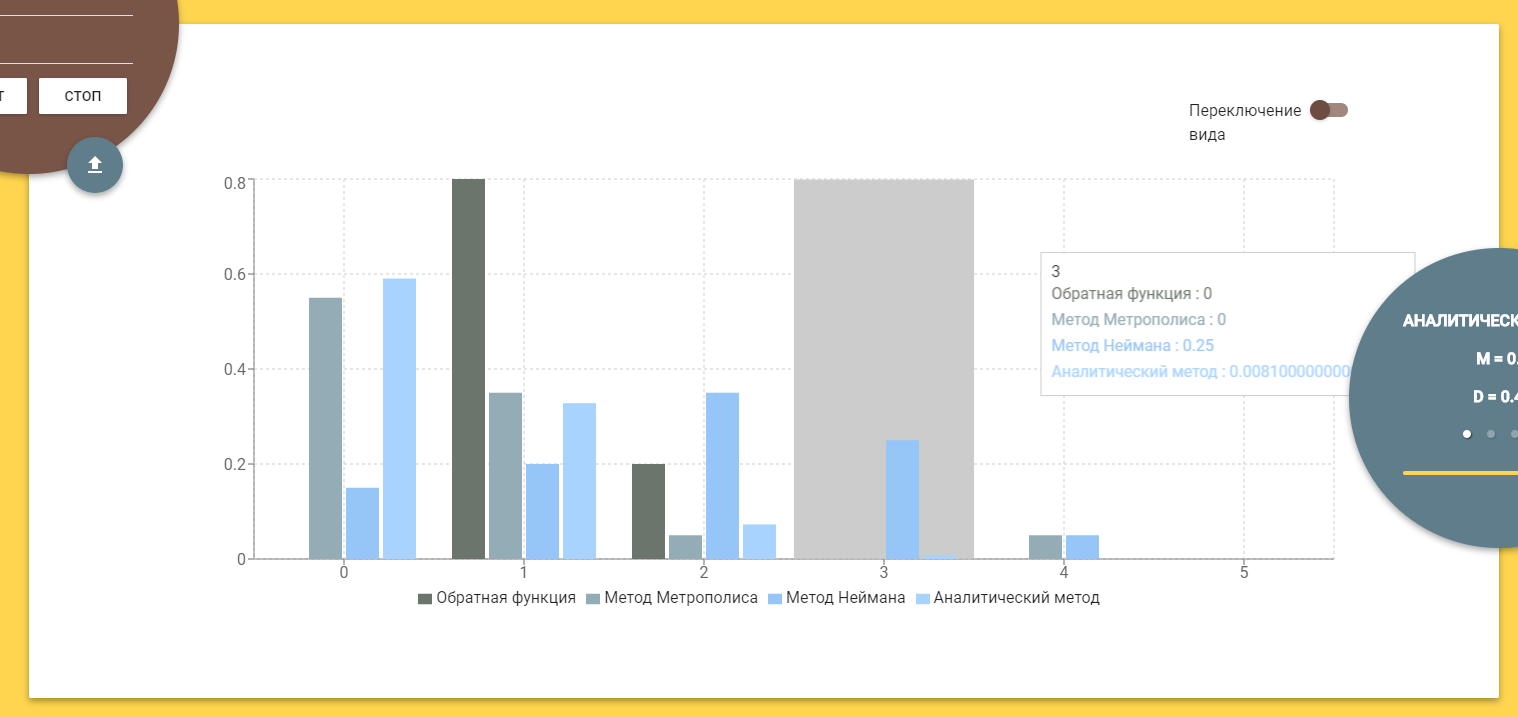
Слайдер с параметрами распределения относительно метода расчёта

Индикатор прогресса

Рис. 3. Панель данных о распределении

1. отображение результатов расчёта

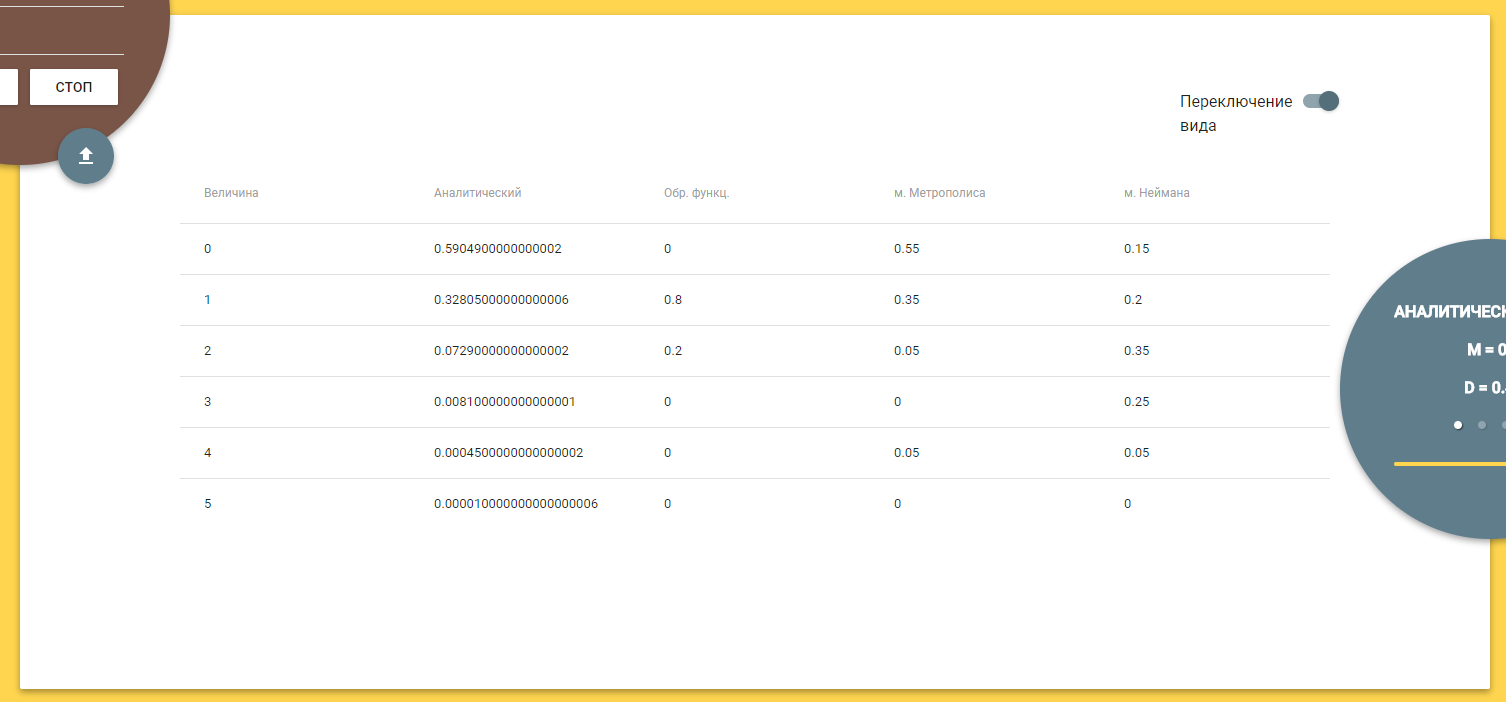
Переключатель вида



Гистограмма

Рис. 4. Отображение результатов с видом на гистограмму

Переключатель вида



Таблица

Рис. 5. Отображение результатов с видом на таблицу

Последовательность действий пользователя

1. Ввести данные в область ввода данных



Рис. 6. Введенные данные для расчётов

1. Нажать кнопку старт
2. Исследовать диаграмму, таблицу, данные о распределении.

Примеры

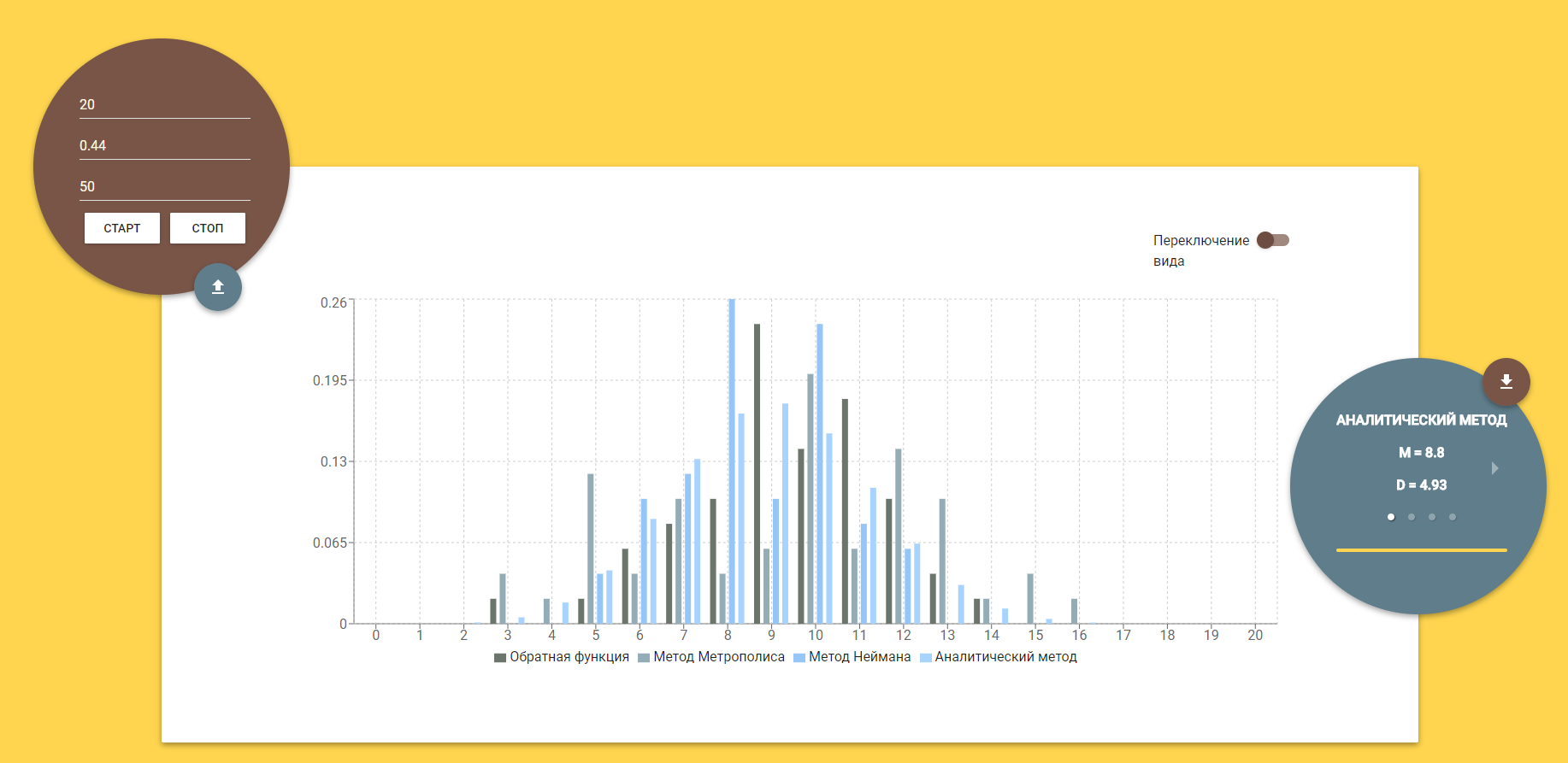


Рис. 7. Вводные данные и гистограмма

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Величина | Аналитический | Обр. функц. | м. Метрополиса | м. Неймана |
| 0 | 0.000009199421971583316 | 0 | 0 | 0 |
| 1 | 0.00014456234526773784 | 0 | 0 | 0 |
| 2 | 0.0010790546486056144 | 0 | 0 | 0 |
| 3 | 0.005086971914855038 | 0.02 | 0.04 | 0 |
| 4 | 0.016986852644248075 | 0 | 0.02 | 0 |
| 5 | 0.042709800934109435 | 0.02 | 0.12 | 0.04 |
| 6 | 0.08389425183485782 | 0.06 | 0.04 | 0.1 |
| 7 | 0.13183382431191945 | 0.08 | 0.1 | 0.12 |
| 8 | 0.16832354354111137 | 0.1 | 0.04 | 0.26 |
| 9 | 0.17633895037640243 | 0.24 | 0.06 | 0.1 |
| 10 | 0.15240723568246206 | 0.14 | 0.2 | 0.24 |
| 11 | 0.1088623112017586 | 0.18 | 0.06 | 0.08 |
| 12 | 0.06415100481532204 | 0.1 | 0.14 | 0.06 |
| 13 | 0.031018068262353506 | 0.04 | 0.1 | 0 |
| 14 | 0.012185669674496021 | 0.02 | 0.02 | 0 |
| 15 | 0.003829781897698749 | 0 | 0.04 | 0 |
| 16 | 0.0009403482338099607 | 0 | 0.02 | 0 |
| 17 | 0.00017384589196486669 | 0 | 0 | 0 |
| 18 | 0.00002276553347158968 | 0 | 0 | 0 |
| 19 | 0.0000018828636705826053 | 0 | 0 | 0 |

Табл. 1. Таблица распределения вероятностей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Аналитический | Обр. функц. | м. Метрополиса | м. Неймана |
| M | 8.8 | 9.42 | 9.44 | 8.62 |
| D | 4.93 | 4.72 | 10.69 | 3.24 |

Табл. 2. Таблица характеристик распределений

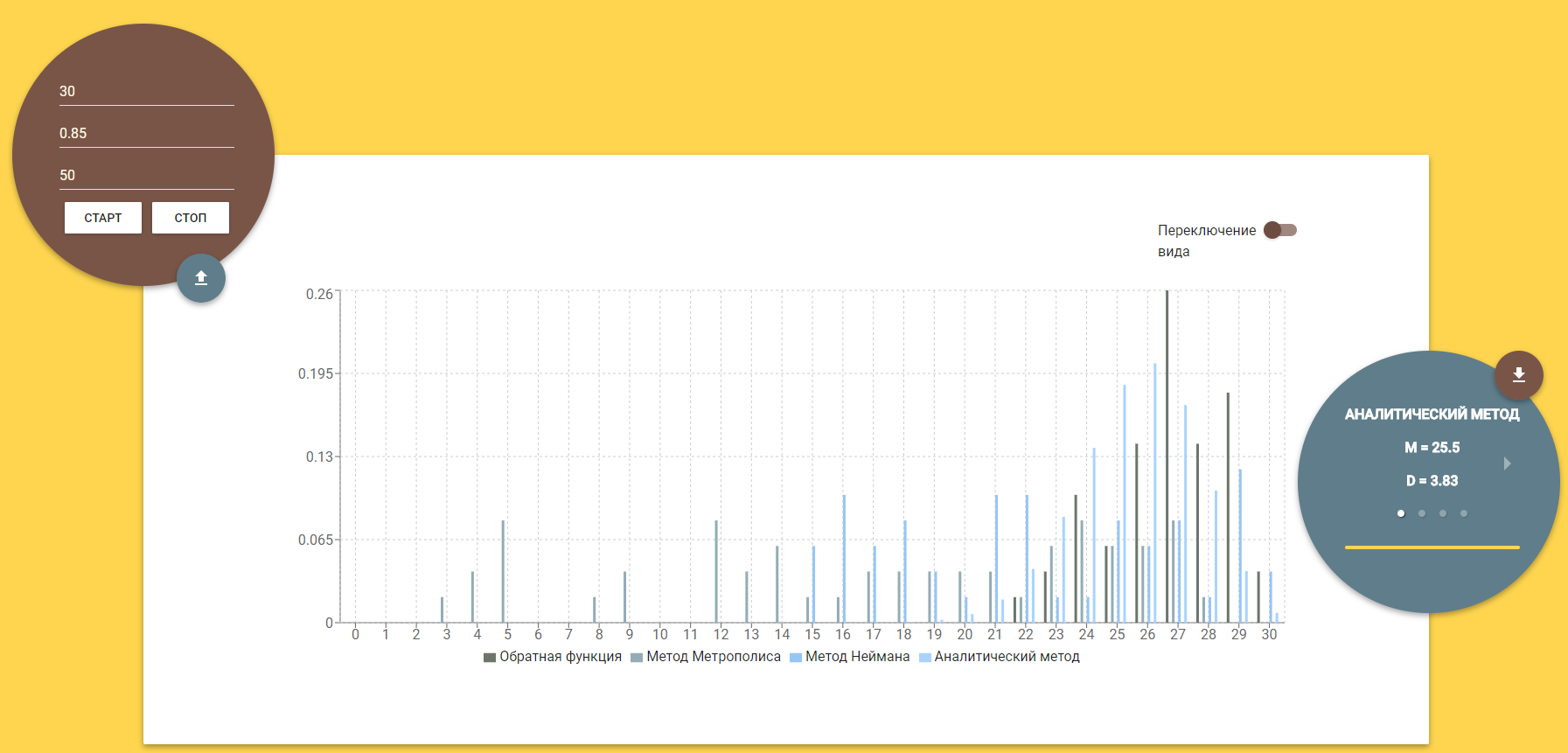


Рис. 8. Вводные данные и гистограмма

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Величина | Аналитический | Обр. функц. | м. Метрополиса | м. Неймана |
| 0 | 1.9175105923288487e-25 | 0 | 0 | 0 |
| 1 | 3.259768006959042e-23 | 0 | 0 | 0 |
| 2 | 2.6784427123846796e-21 | 0 | 0 | 0 |
| 3 | 1.4165985901056743e-19 | 0 | 0.02 | 0 |
| 4 | 5.418489607154202e-18 | 0 | 0.04 | 0 |
| 5 | 1.5966482709081049e-16 | 0 | 0.08 | 0 |
| 6 | 3.769863972977469e-15 | 0 | 0 | 0 |
| 7 | 7.32430714749908e-14 | 0 | 0 | 0 |
| 8 | 1.193251706113392e-12 | 0 | 0.02 | 0 |
| 9 | 1.6528745855052165e-11 | 0 | 0.04 | 0 |
| 10 | 1.9669207567512076e-10 | 0 | 0 | 0 |
| 11 | 2.0265244160466983e-9 | 0 | 0 | 0 |
| 12 | 1.8182427399530093e-8 | 0 | 0.08 | 0 |
| 13 | 1.426621226732361e-7 | 0 | 0.04 | 0 |
| 14 | 9.816512726801242e-7 | 0 | 0.06 | 0 |
| 15 | 0.000005933536581533194 | 0 | 0.02 | 0.06 |
| 16 | 0.000031521913089395094 | 0 | 0.02 | 0.1 |
| 17 | 0.00014710226108384376 | 0 | 0.04 | 0.06 |
| 18 | 0.0006020296240653605 | 0 | 0.04 | 0.08 |
| 19 | 0.002154632338760237 | 0 | 0.04 | 0.04 |
| 20 | 0.006715270789136071 | 0 | 0.04 | 0.02 |
| 21 | 0.018120571970684635 | 0 | 0.04 | 0.1 |
| 22 | 0.04200678047749618 | 0.02 | 0.02 | 0.1 |
| 23 | 0.08279597311506492 | 0.04 | 0.06 | 0.02 |
| 24 | 0.13684334445406562 | 0.1 | 0.08 | 0.02 |
| 25 | 0.18610694845752923 | 0.06 | 0.06 | 0.08 |
| 26 | 0.2028088540883331 | 0.14 | 0.06 | 0.06 |
| 27 | 0.17025928491366232 | 0.26 | 0.08 | 0.08 |
| 28 | 0.10337170869758068 | 0.14 | 0.02 | 0.02 |
| 29 | 0.04039813903123842 | 0.18 | 0 | 0.12 |

Табл. 3. Таблица распределения вероятностей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Аналитический | Обр. функц. | м. Метрополиса | м. Неймана |
| M | 25.5 | 26.88 | 17.4 | 22.32 |
| D | 3.83 | 3.95 | 56.4 | 22.7 |

Табл. 4. Таблица характеристик распределений

# Заключение

В ходе данной работы была спроектирована, закодирована и отлажена программа, выполняющая генерацию случайной величины с биномиальным распределением. Программа также выполняет графическую визуализацию полученных результатов, подсчитывает различные числовые характеристики и поддерживание сохранение/загрузку результатов вычислений. Описан процесс тестирования, и приведены тестовые результаты. Также приводится описание программного продукта, созданного в результате проделанной работы, его интерфейс и процесс работы с программой.

Были разработаны алгоритмы реализации различных методов генерации случайных величин: метода обратных функций, метода Неймана, метода Метрополиса; рассмотрены теоретические сведенья, лежащие в основе данных методов.

Были закреплены знания и навыки, полученные во время изучения курса «Компьютерное моделирование стохастических процессов». Получен ценный опыт по созданию целостных проектов и оформлению документации.

# Список литературы

[1] http://ru.wikipedia.org

[2] http://en.wikipedia.org

[3] Численные методы Монте-Карло, И. М. Соболь. Главная редакция физико-математической литературы изд-ва «Наука», 1973

[4] http://ermak.cs.nstu.ru/mmsa/glava3/glava3\_3.htm