МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ ИМ. В. Н. КАРАЗИНА

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК

КАФЕДРА МОДЕЛИРОВАНИЯ СИСТЕМ И ТЕХНОЛОГИЙ

**КУРСОВАЯ РАБОТА**

**ПО КУРСУ**

**«КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СТОХАСТИЧЕСКИХ ПРОЦЕССОВ»**

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

**Выполнил**

студент КС-41

Мишенин А.С.

**Проверил**

профессор Лазурик В.Т.

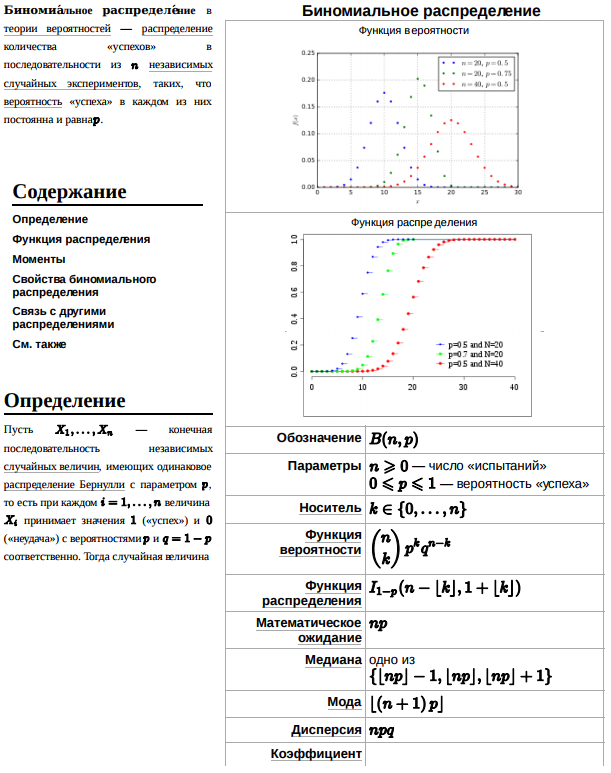
ХАРЬКОВ – 2017

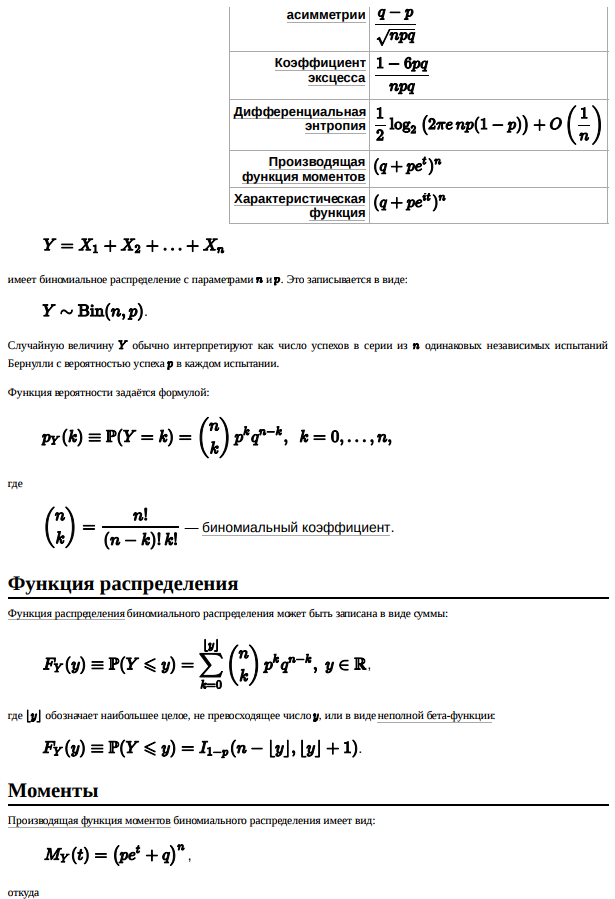
# Введение

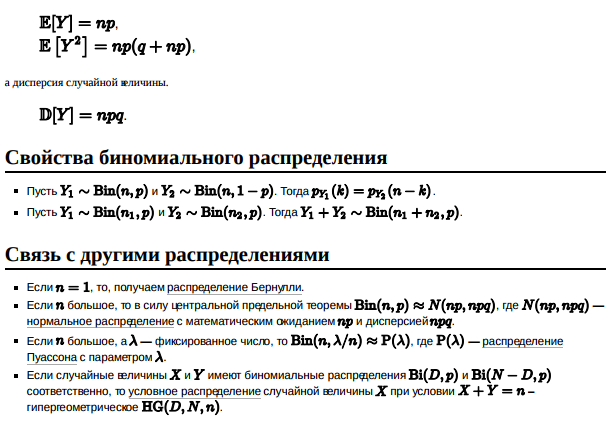
Цель работы – получение практических навыков в разработке программного обеспечения с применением генератора случайных величин, распределенных по заданному распределению, оформление документации, а также закрепление и контроль знаний и навыков, полученных во время изучения курса «Компьютерное моделирование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

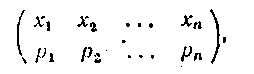
Метод обратных функций

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).

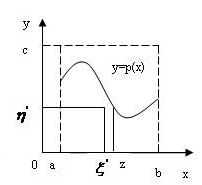
*Доказательство* занимает одну строку:

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.

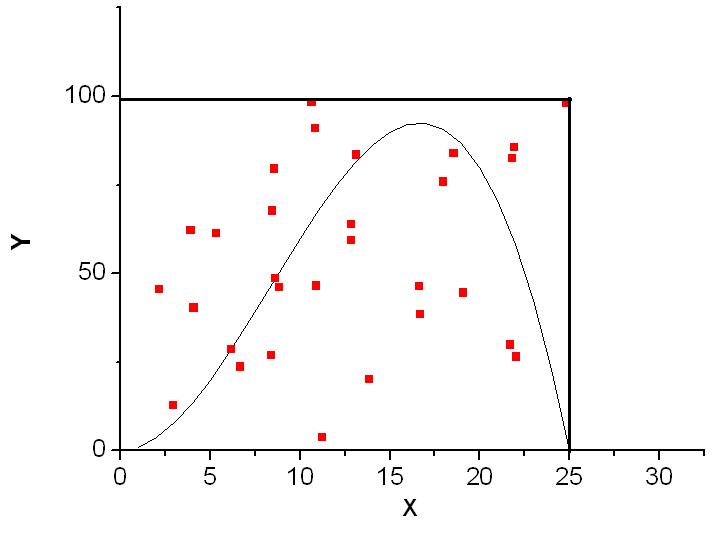
Рассмотрим случайную величину , определенную на конечном интервале  с ограниченной плотностью  (рис). Пусть независимые случайные числа и Случайная величина , определенная условием  имеет плотность вероятностей, равную 

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

=



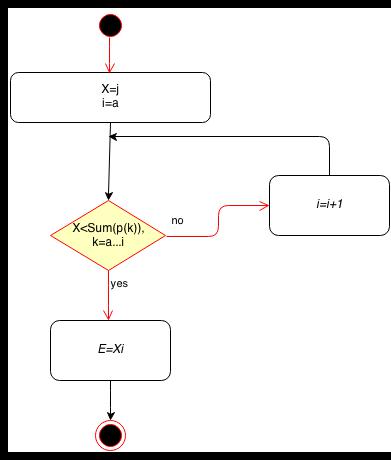
end if

# 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

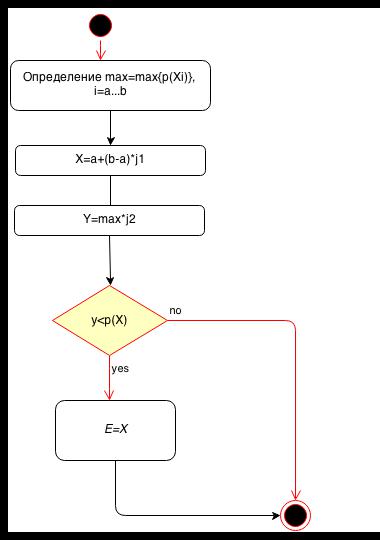
Программная реализация:

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

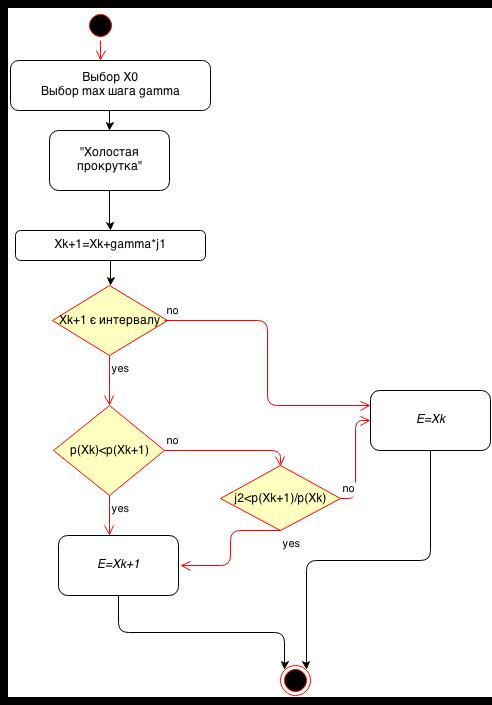
p(Xi)-вероятность выпадение СВ Xi

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. };

# Работа с программой