Министерство образования Республики Беларусь

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Факультет прикладной математики и информатики

**Кафедра**

Шнитко Денис Иванович

Отчет по лабораторным работам по курсу

“Имитационное и статистическое моделирование”

студента 4 курса группы

*Минск 2010*

**Лабораторная работа 1.**

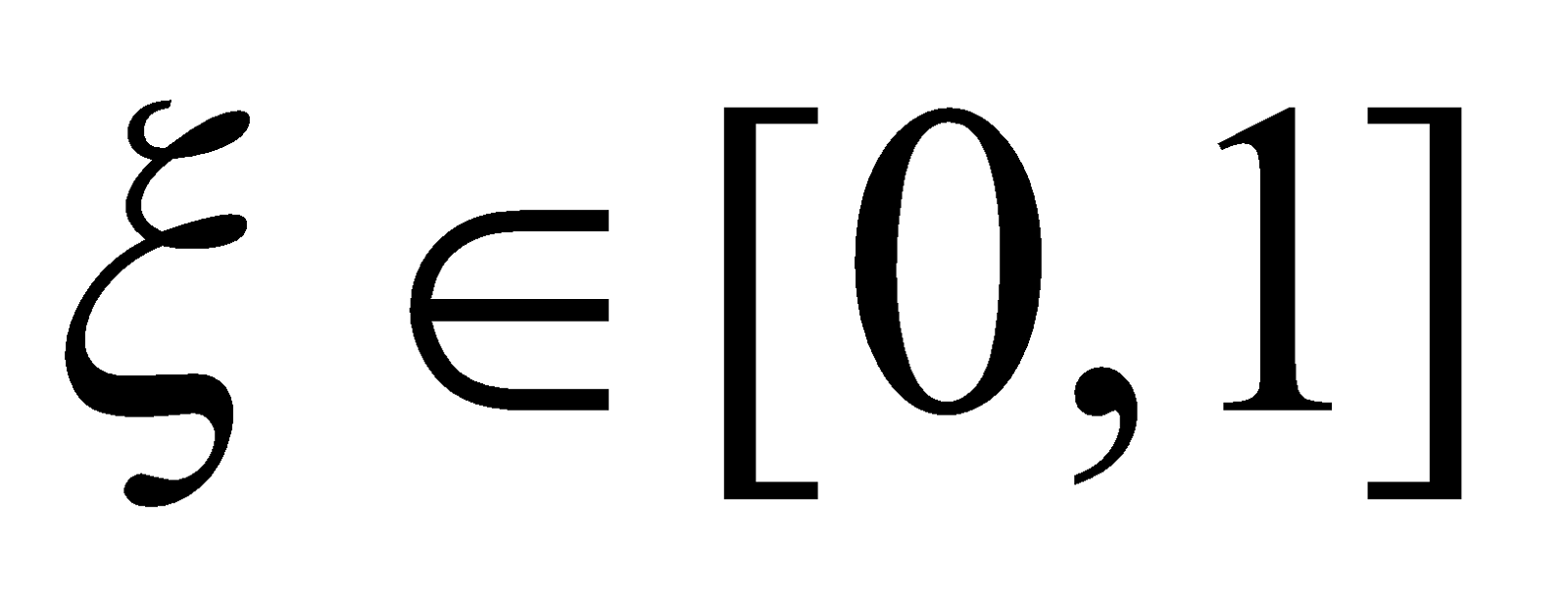
**Условие:**

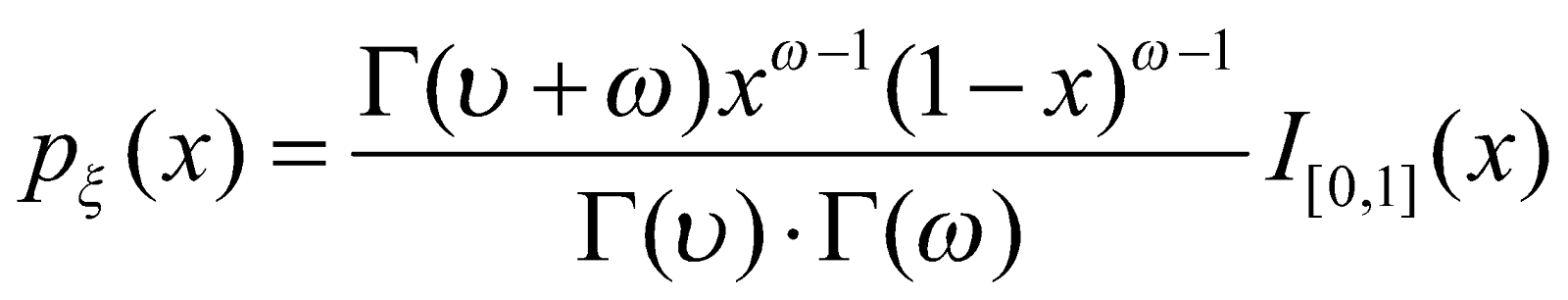
равнить по точности и быстродействию методы моделирования n=1000 реализаций ***CB ξ~ β(v,w)***

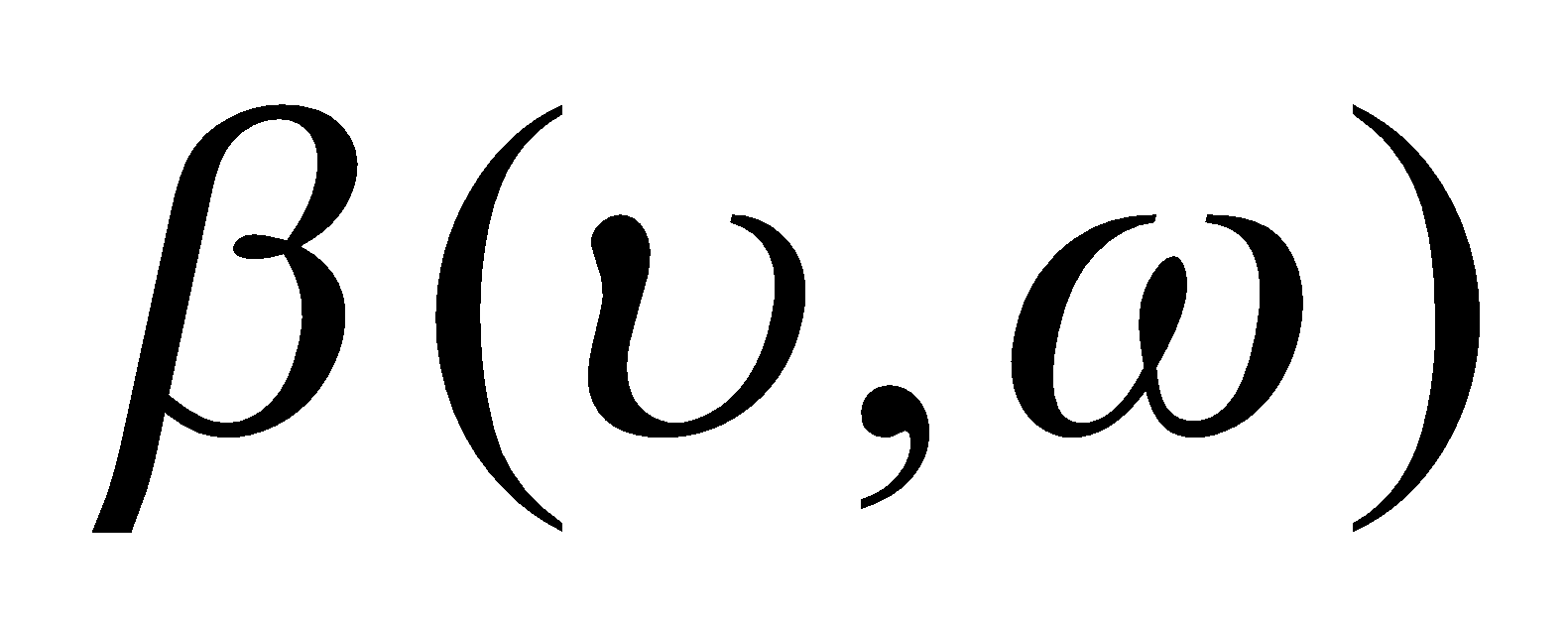
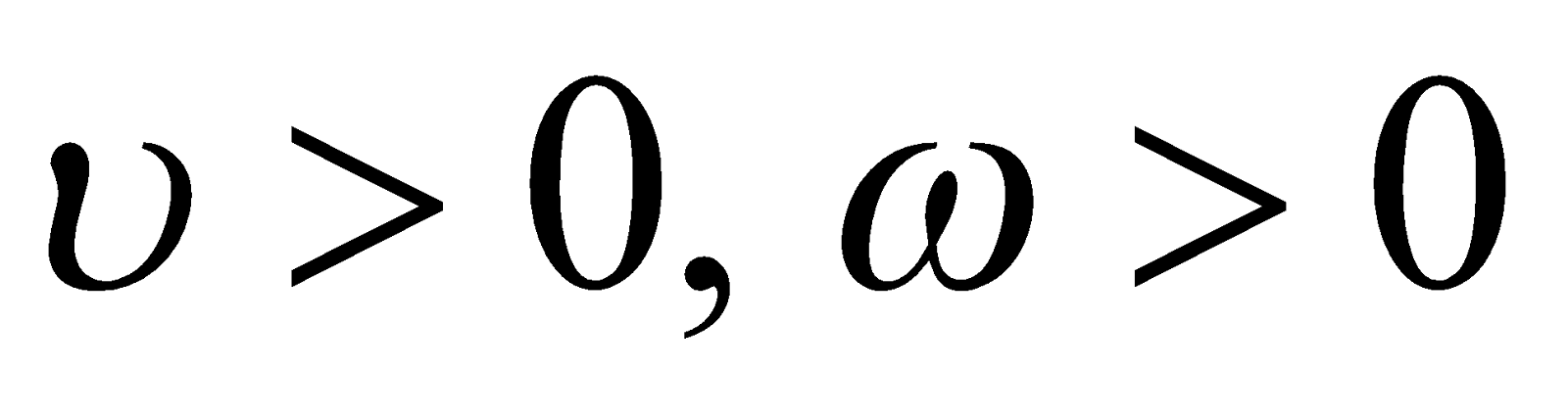
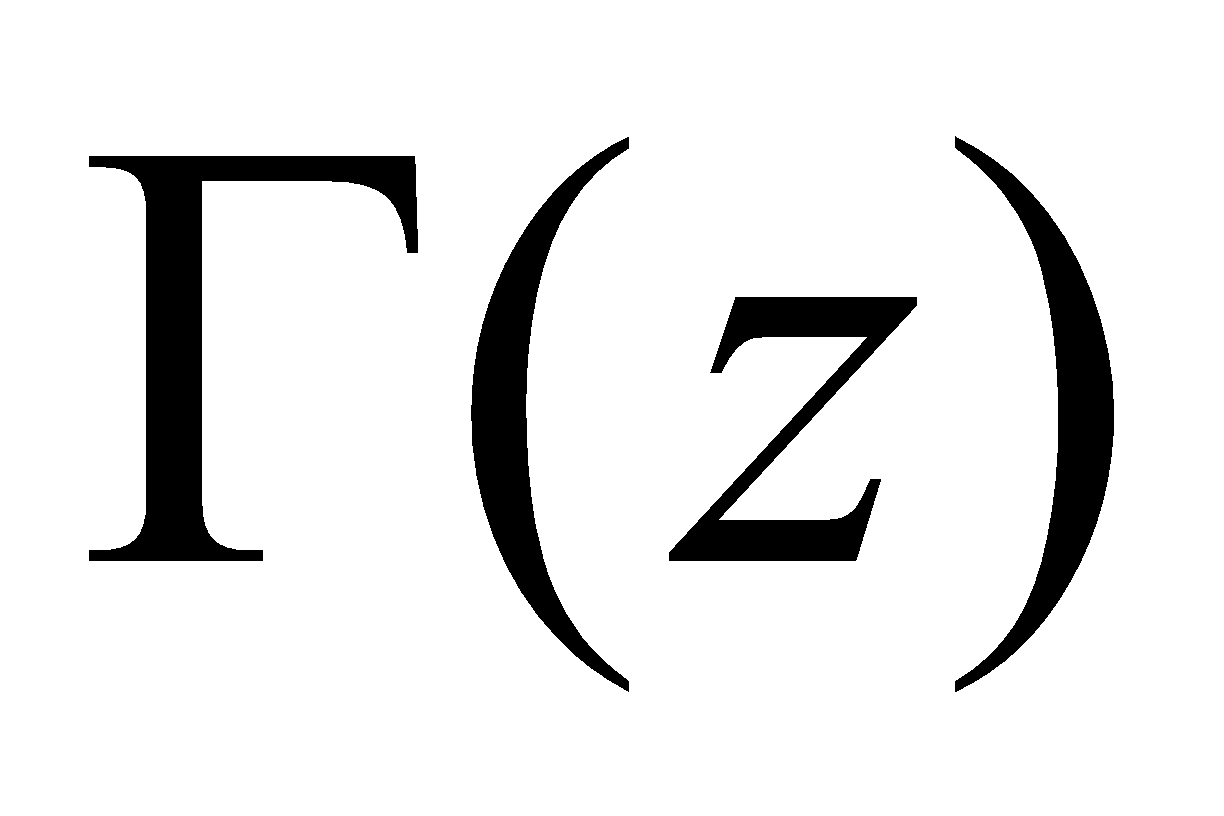
В качестве параметвров были взяты v = 2 и w = 4. Для проверки точности моделирования использовался критерий серий.

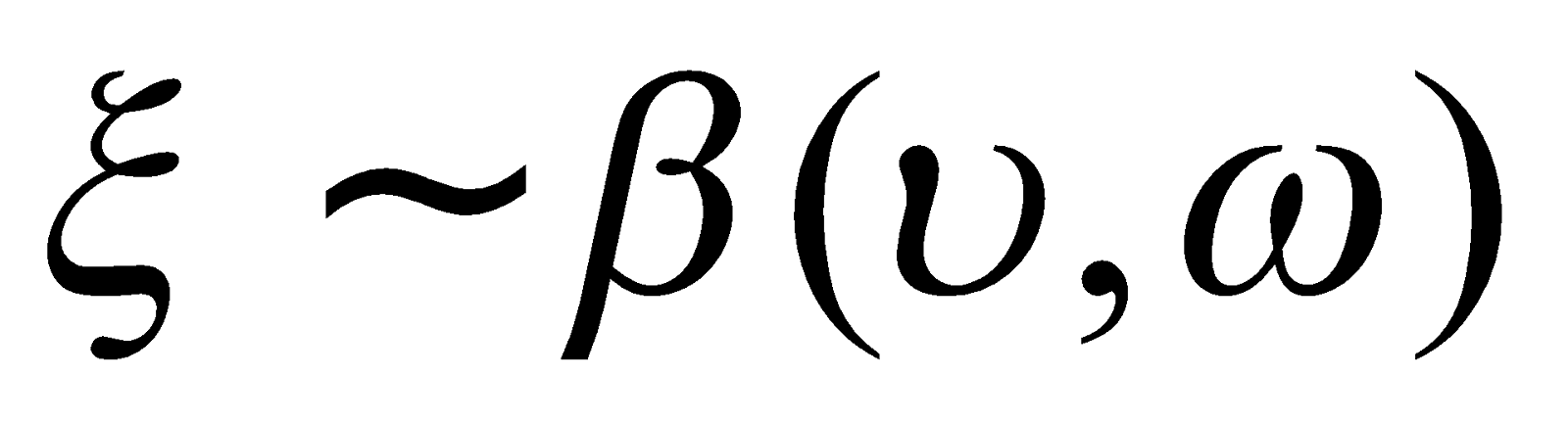
**Теория:**

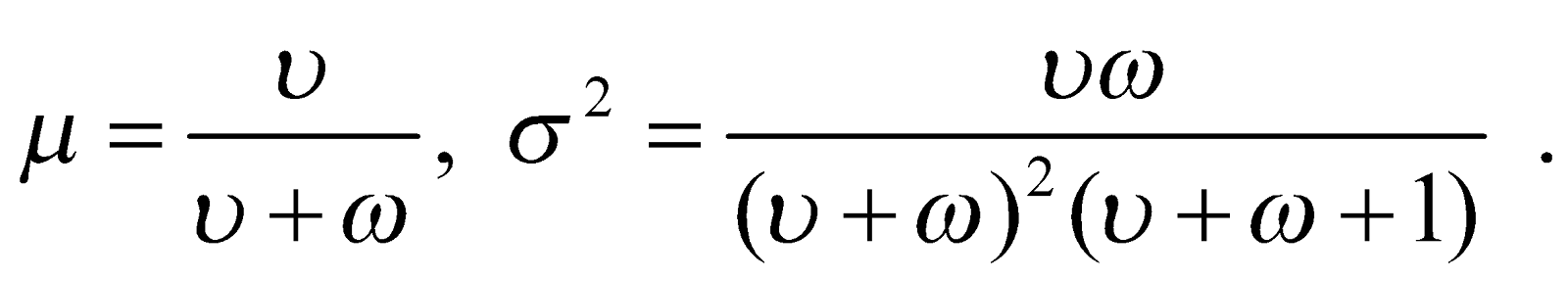
**Бета Распределение:**

НСВ с плотностью распределения

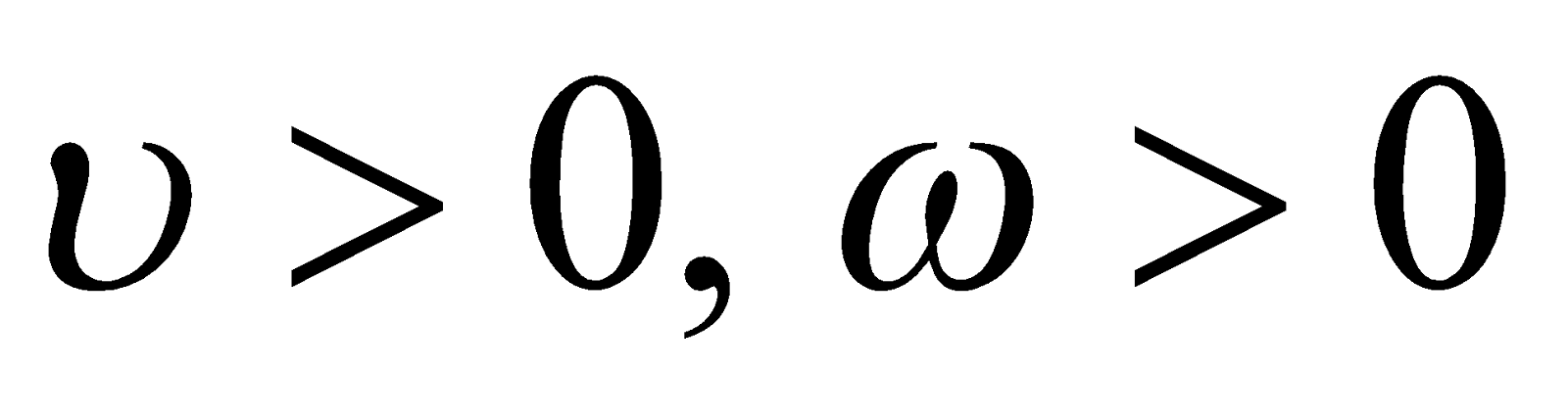
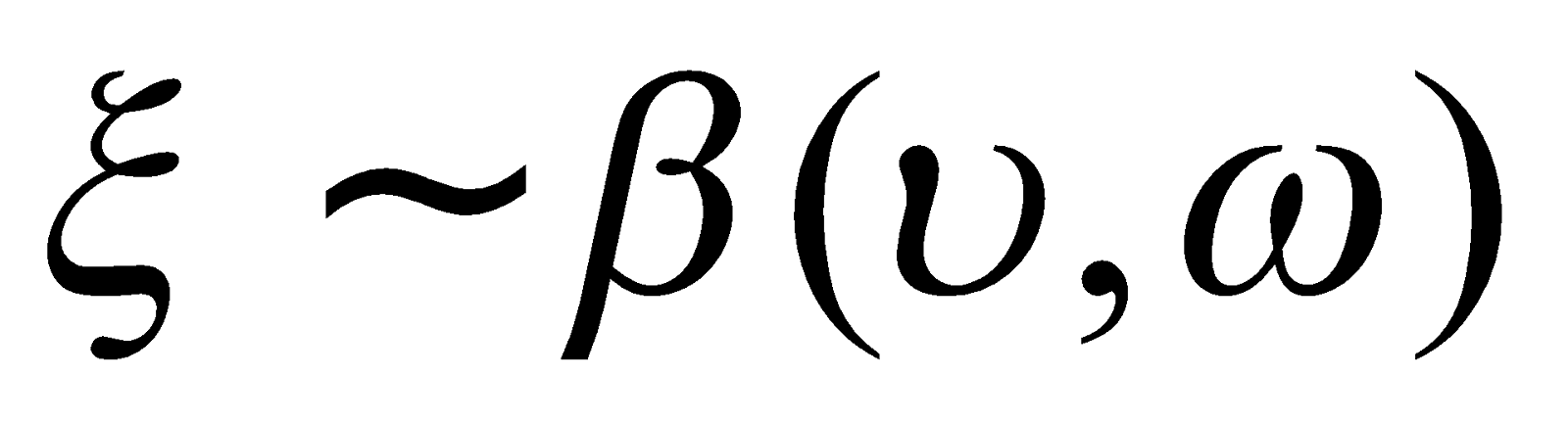
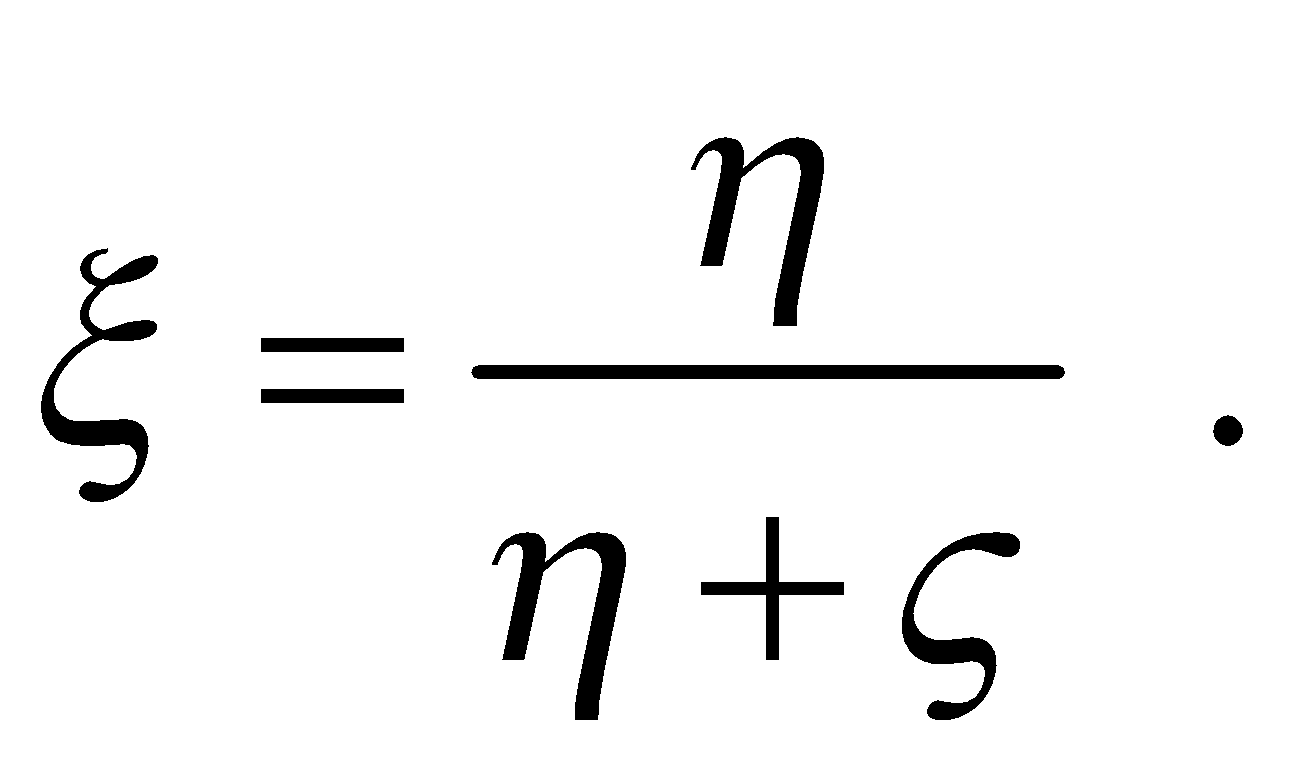
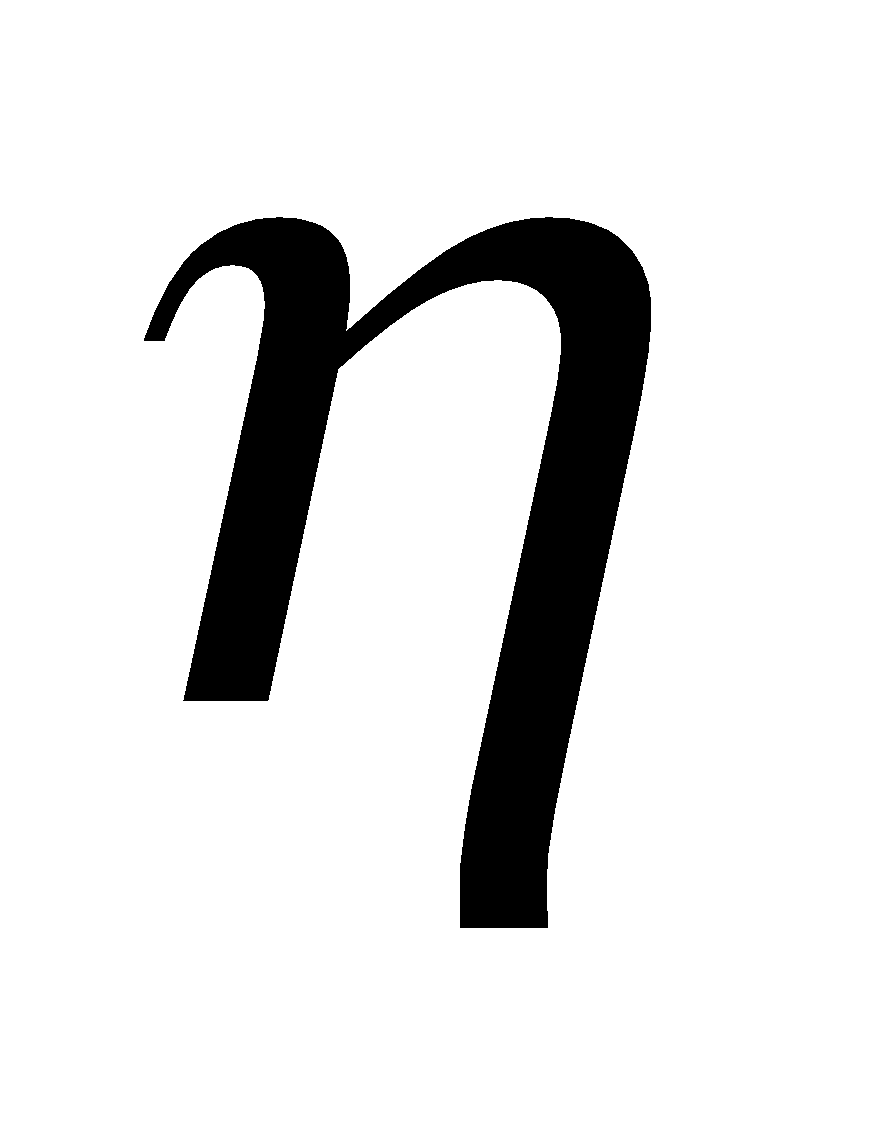
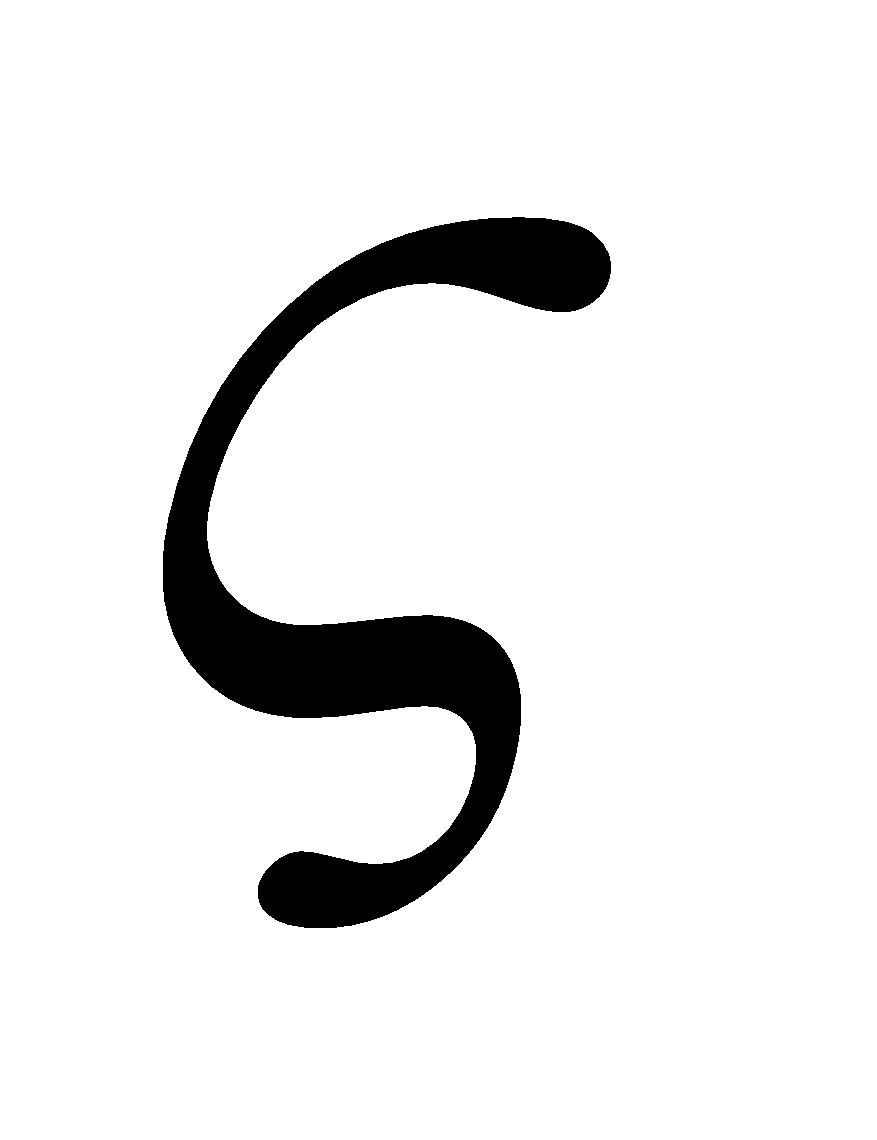
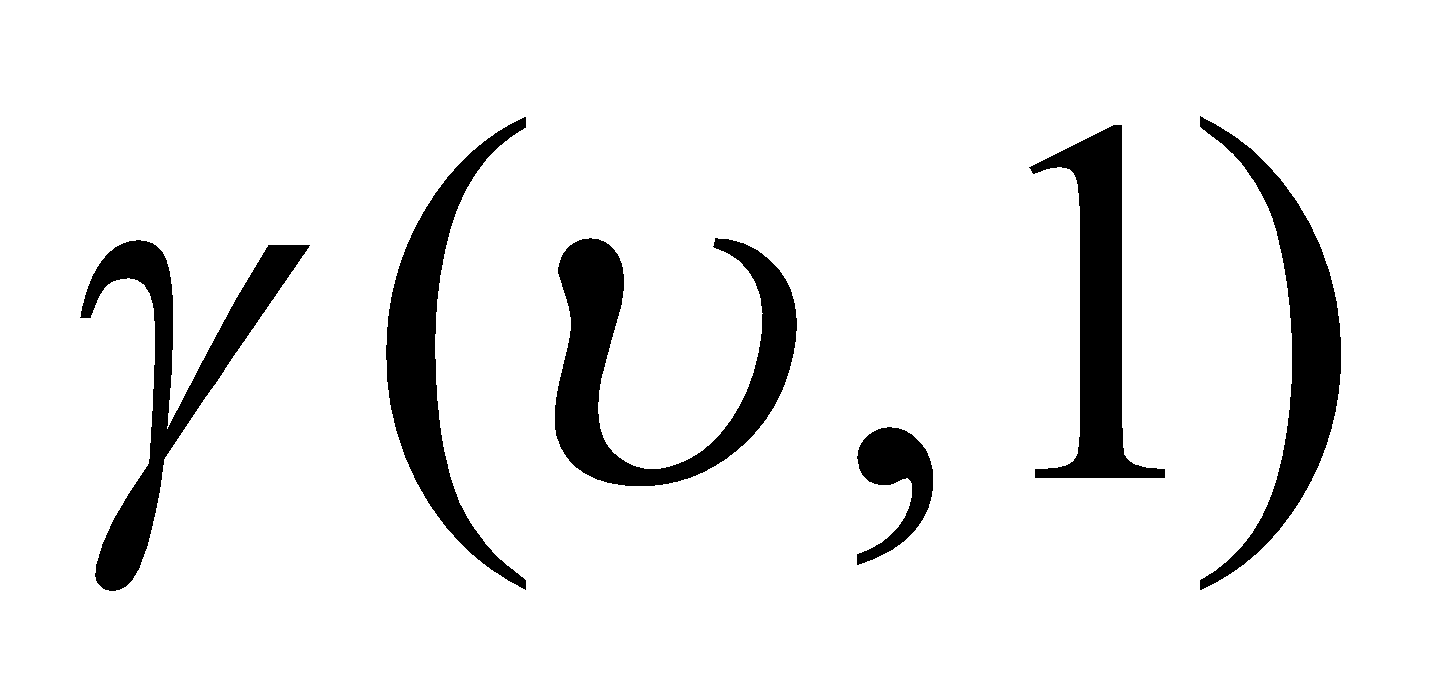
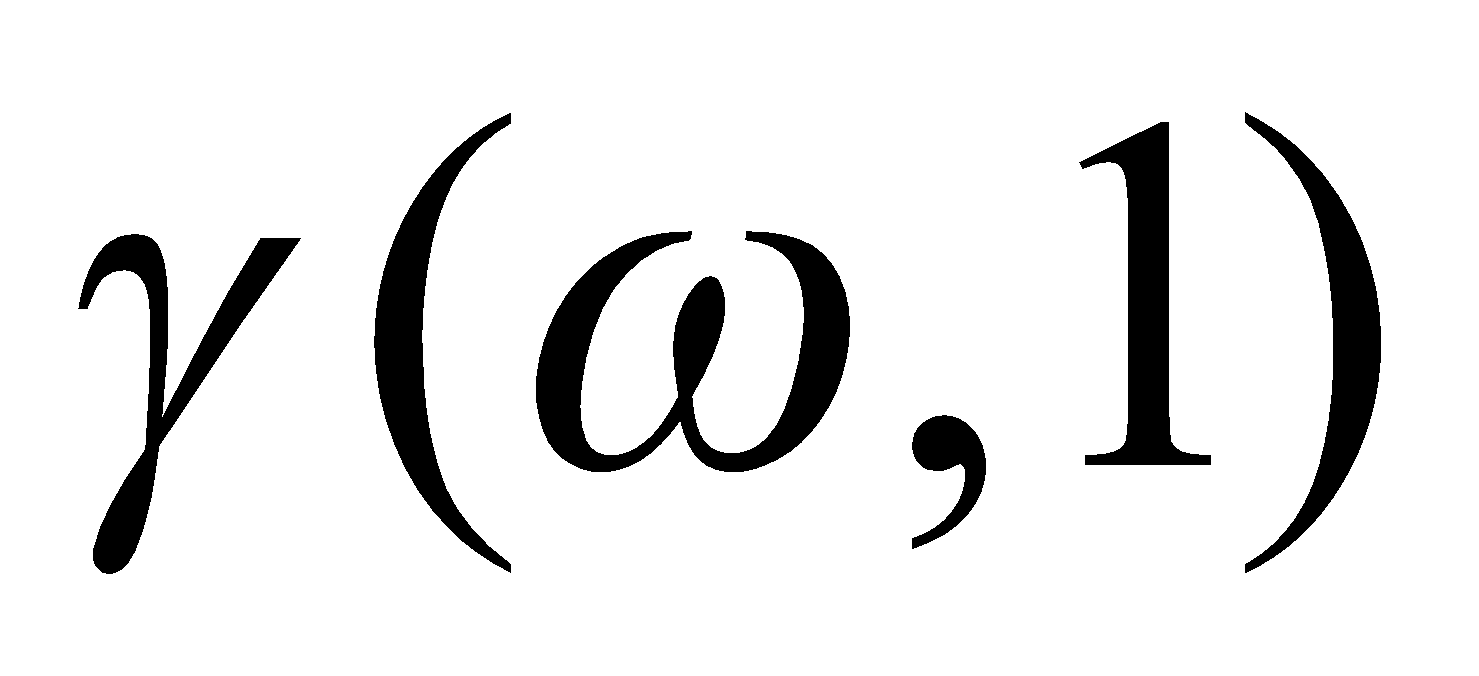
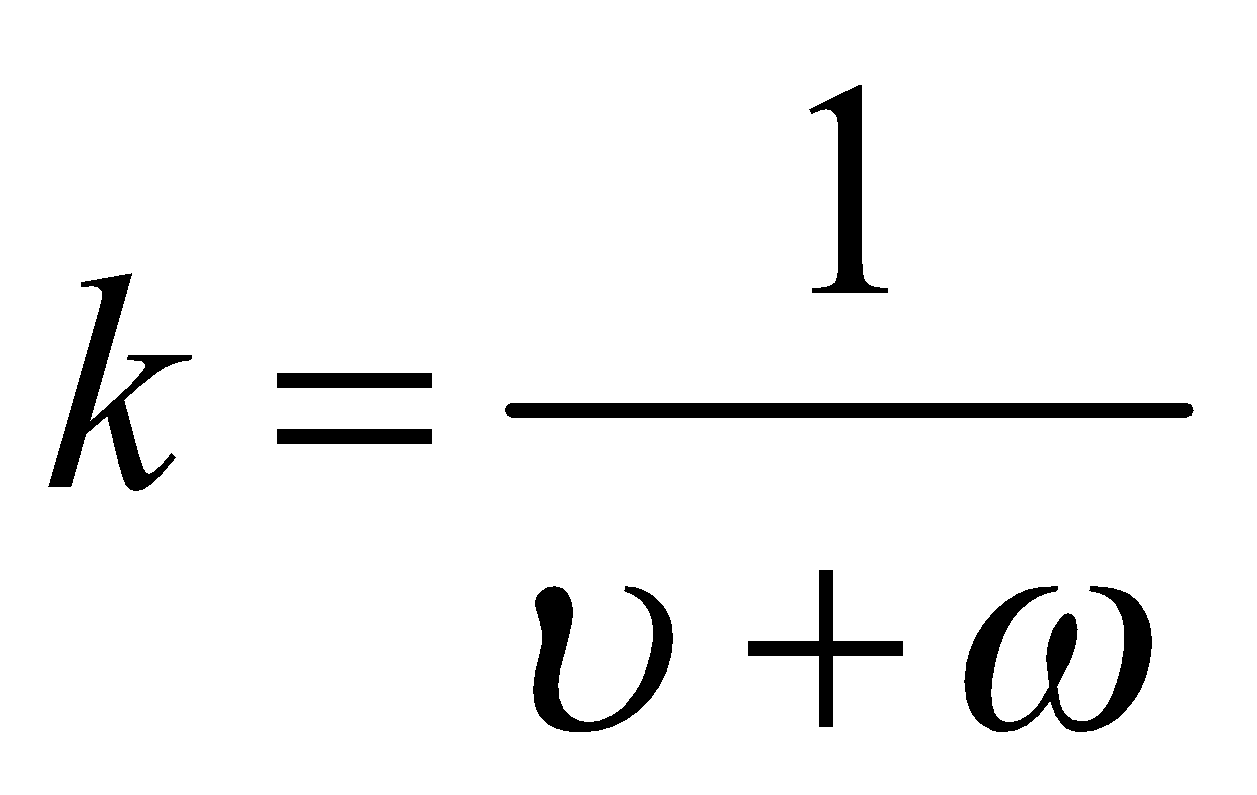
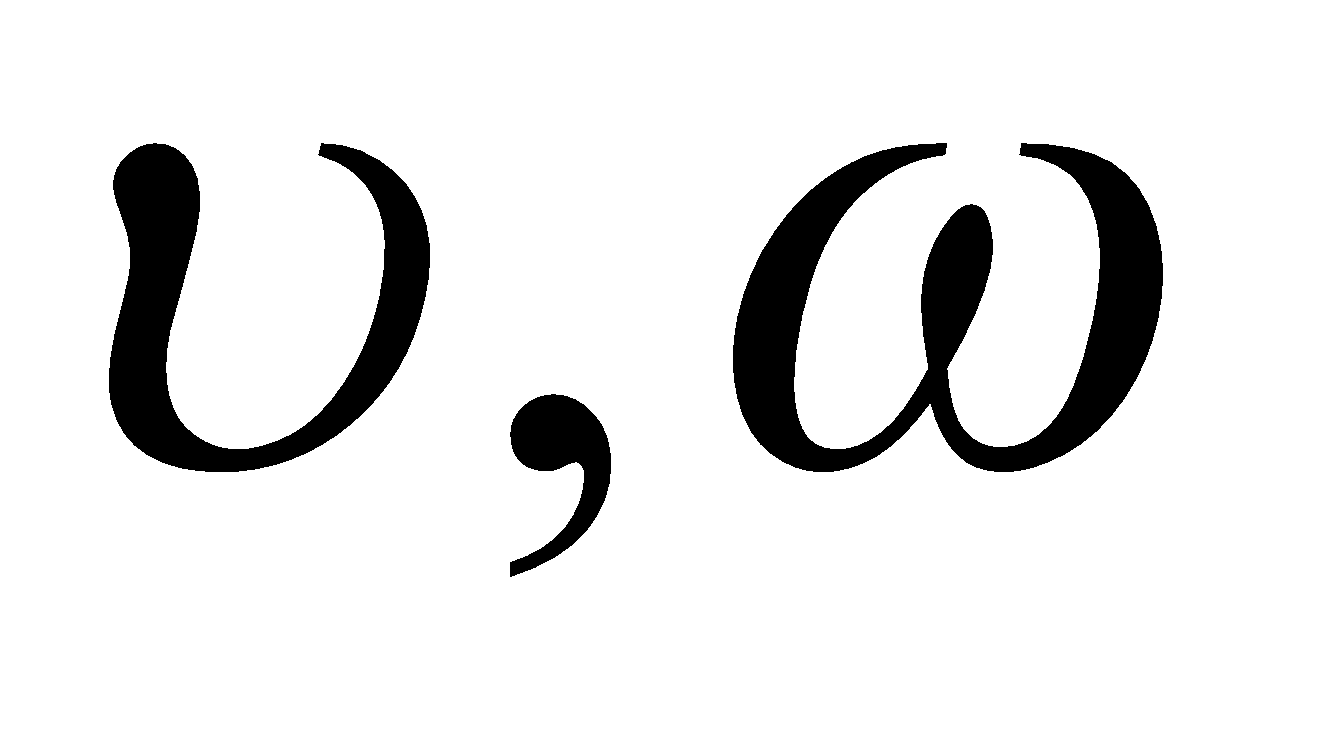


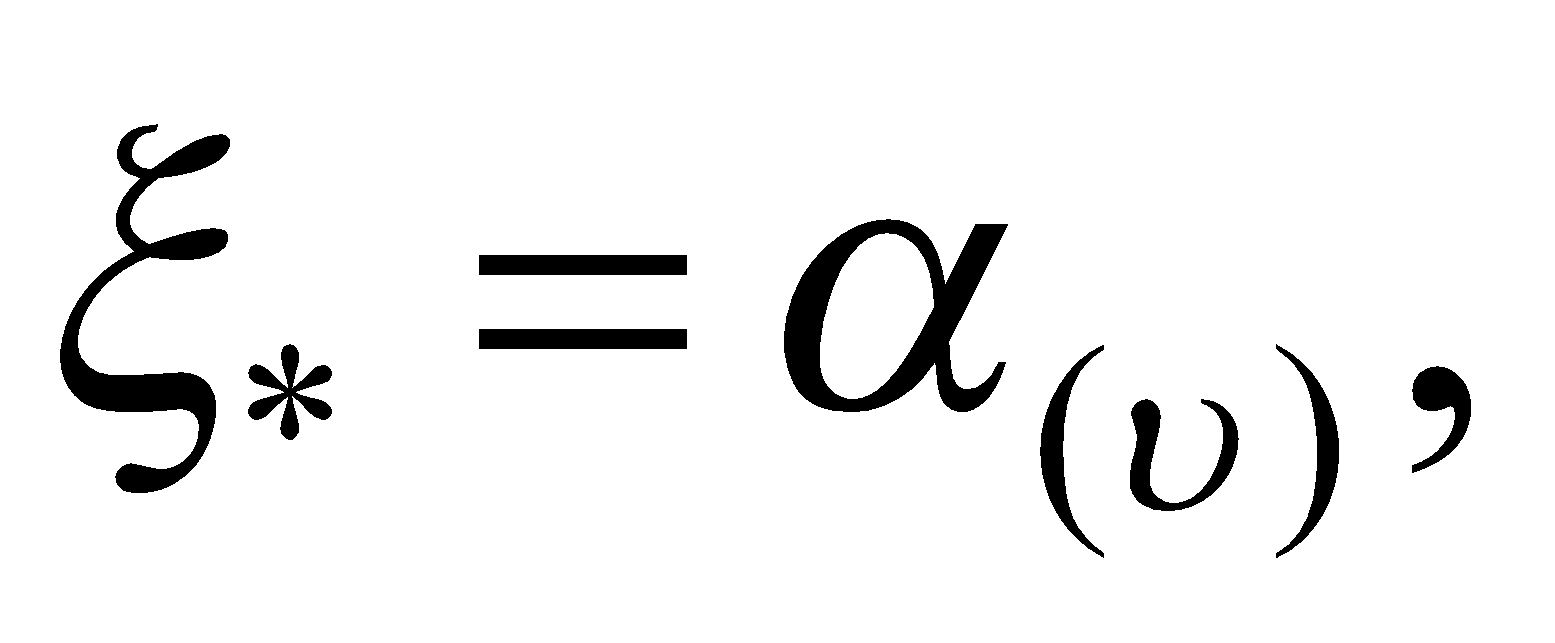
имеет бета-распределение  с параметрами  (где  - гамма-функция Эйлера).

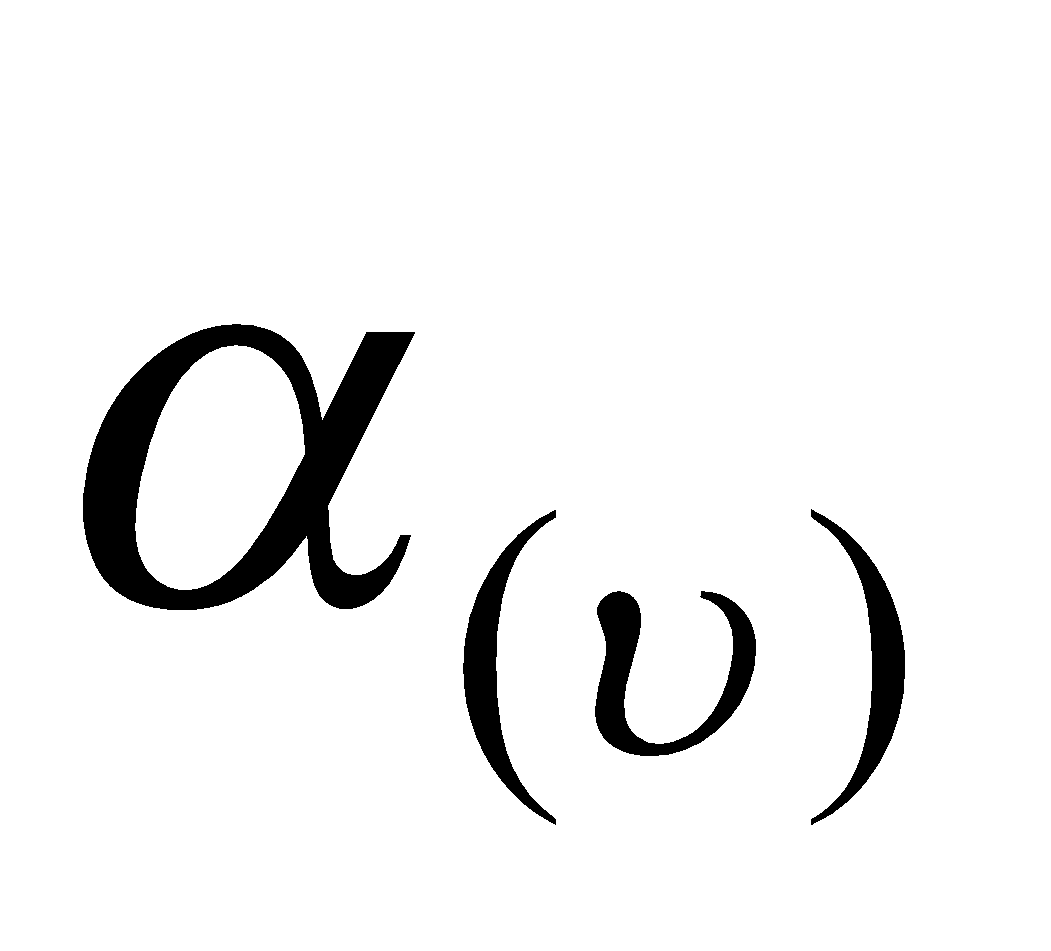
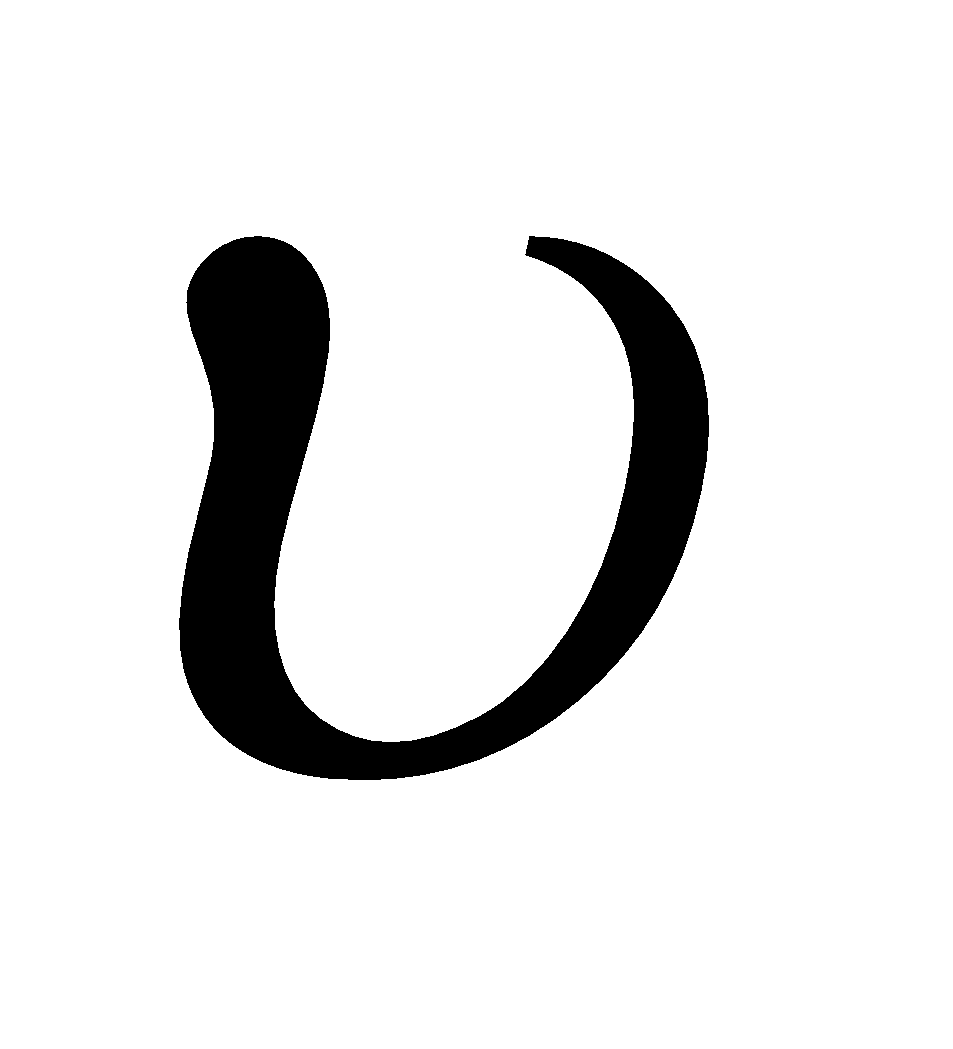
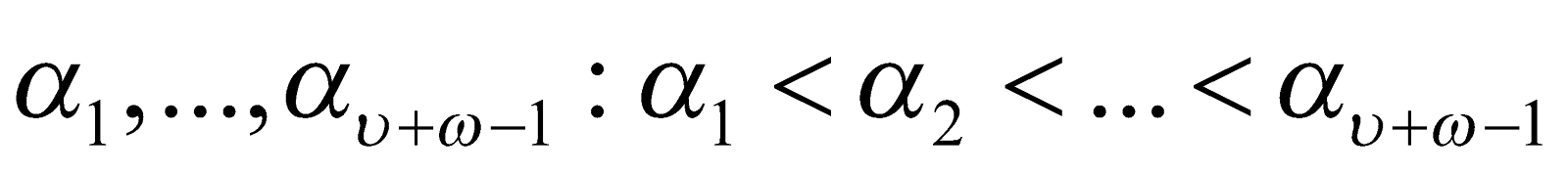
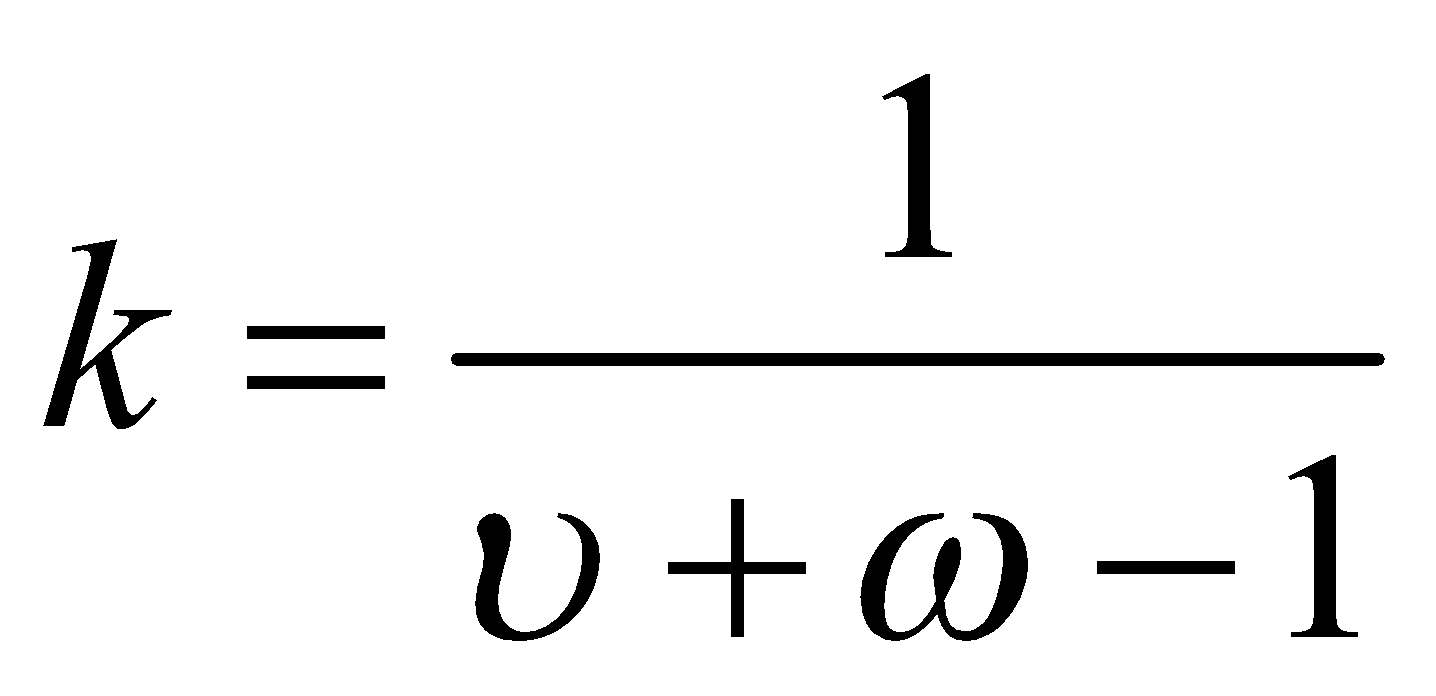
Среднее значение и дисперсия  равны:

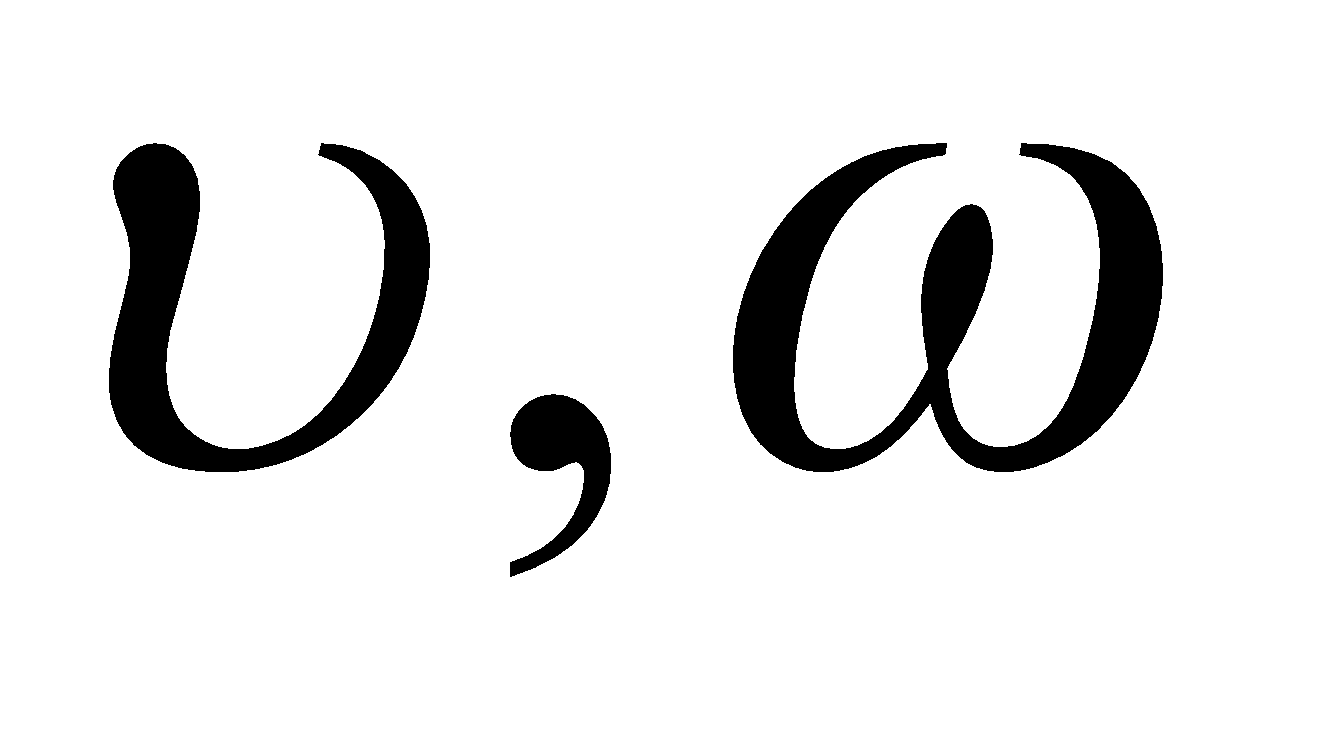


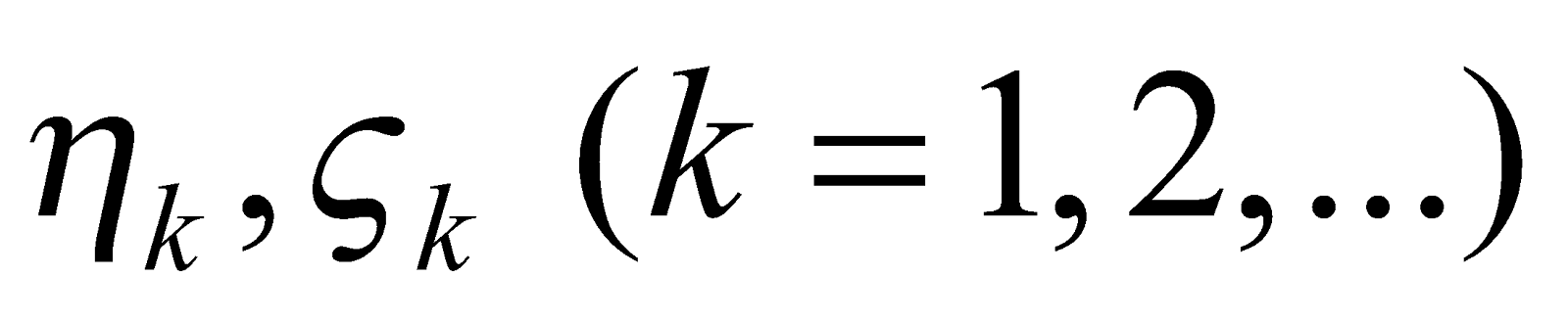
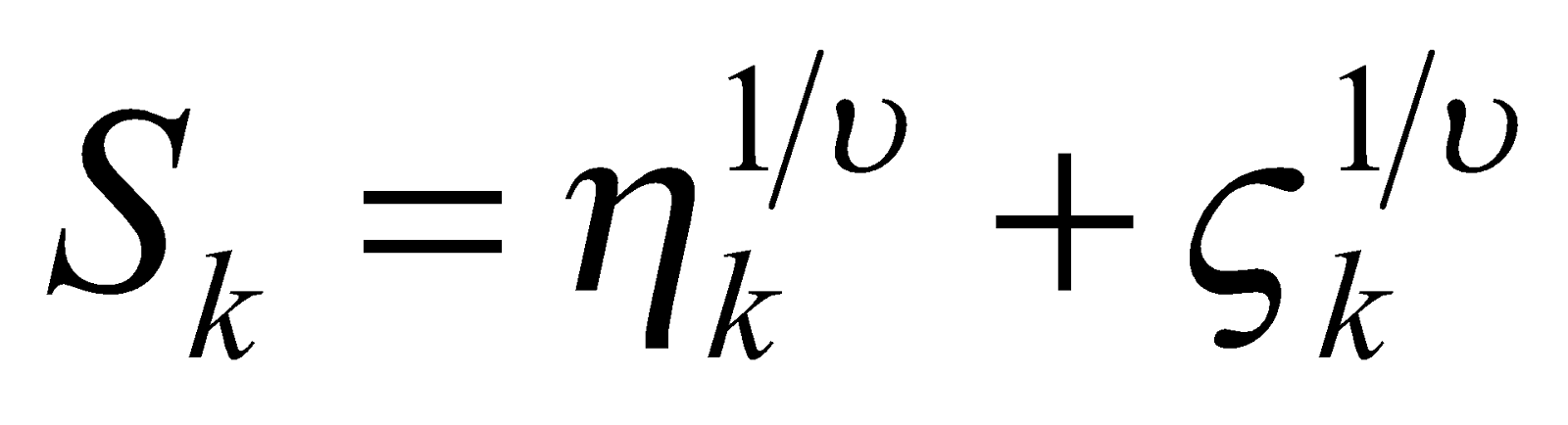
**Алгоритмы моделирования:**

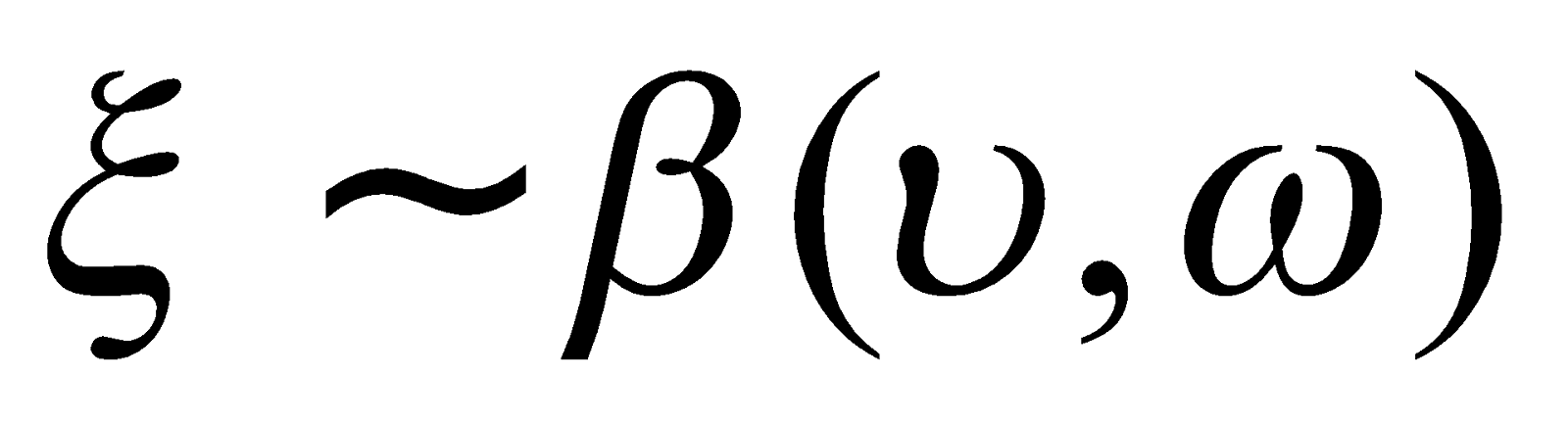
1. Для целых значений  алгоритм моделирования  определяется формулой  , где  и  - случайные величины, имеющие гамма-распределения  и  соответственно Коэффициент использования БСВ .
2. Другой алгоритм для целочисленных значений параметров  основан на методе функционального преобразования и описывается формулой:

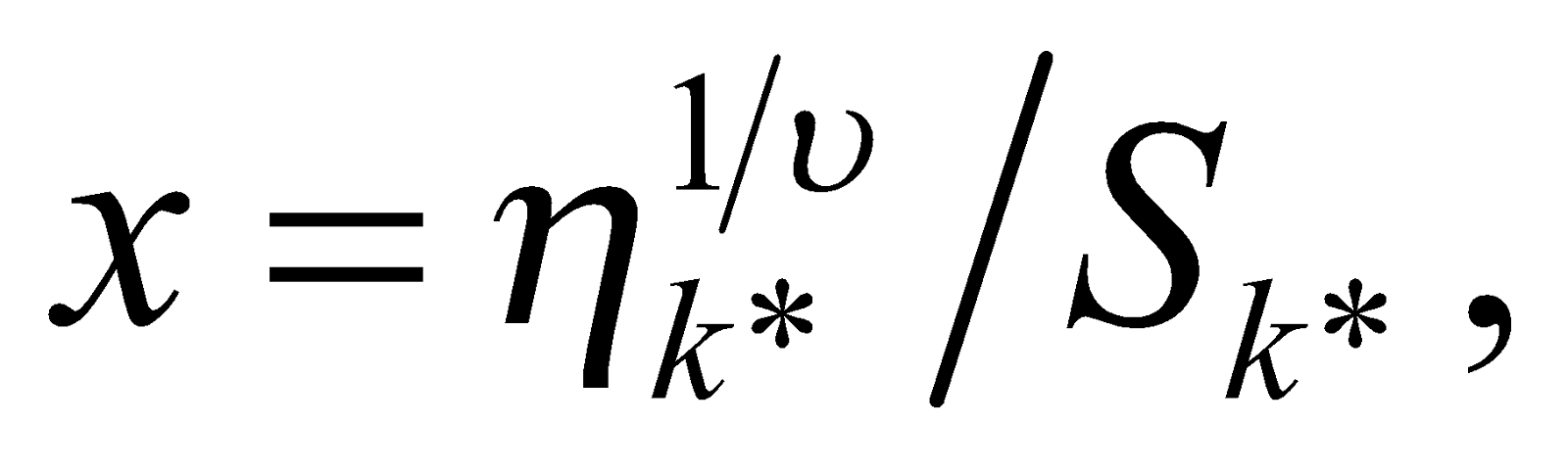


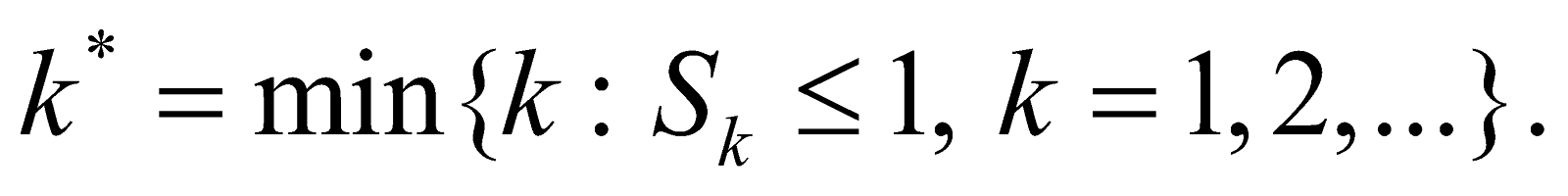
где  - -я порядковая статистика, соответствующая последовательности независимых БСВ . Коэффициент использования БСВ .

1. Для нецелых параметров  используется алгоритм (метод Йонка), составленный из следующих шагов:
2. Моделирование пар независимых реализаций БСВ:

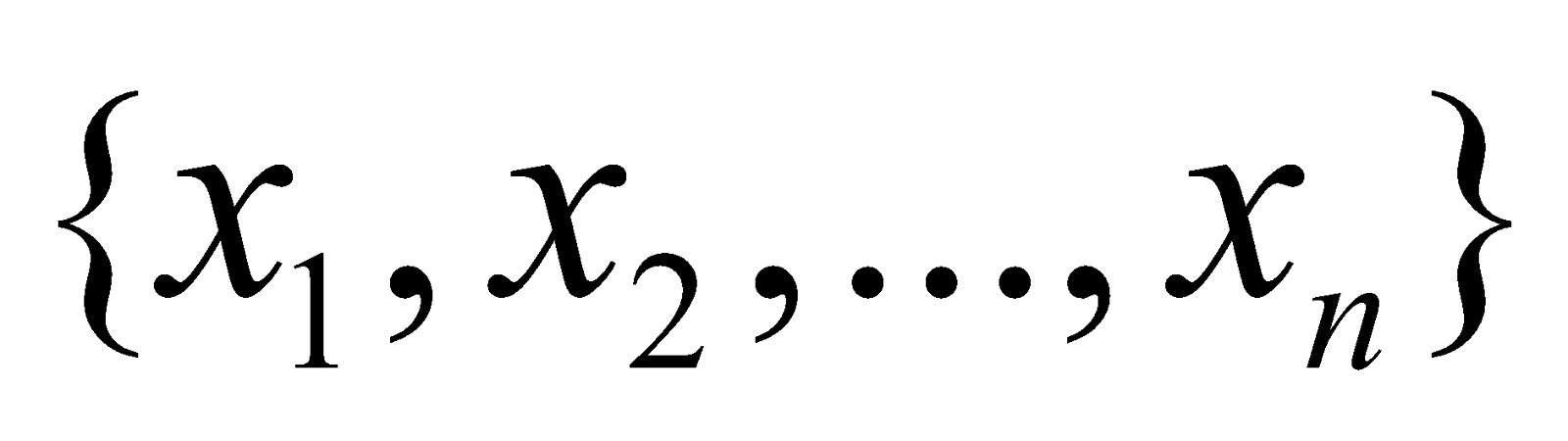
 и вычисление величин .

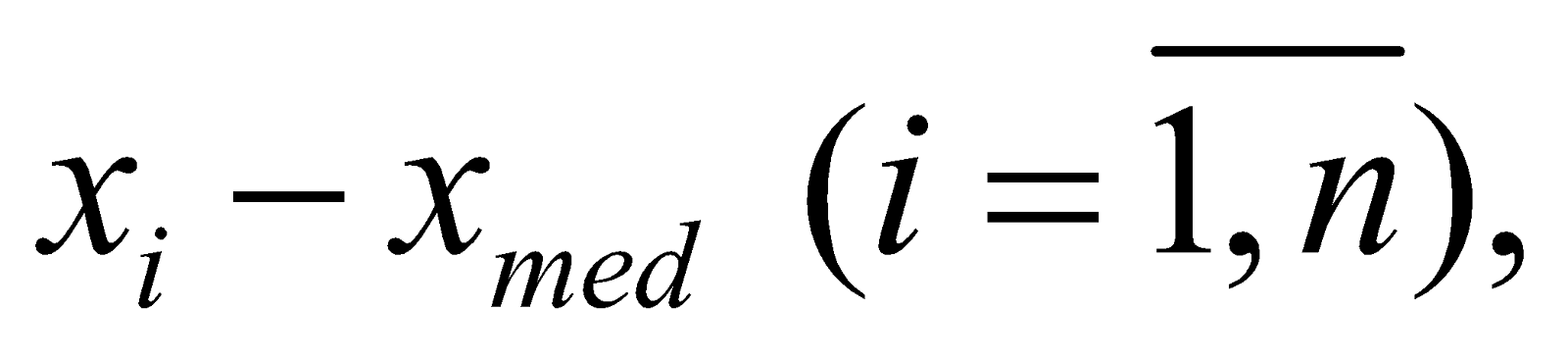
1. Принятие решения о том, что реализацией СВ  является величина вида:

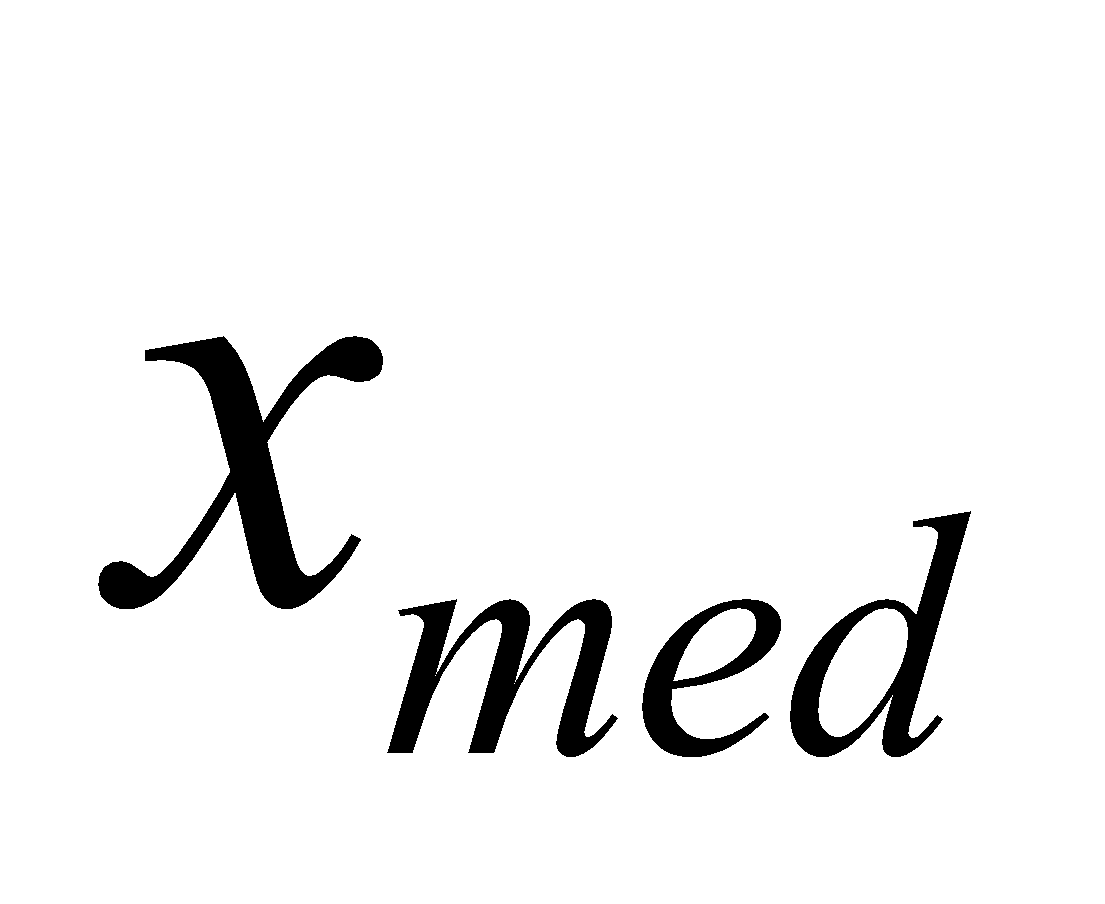
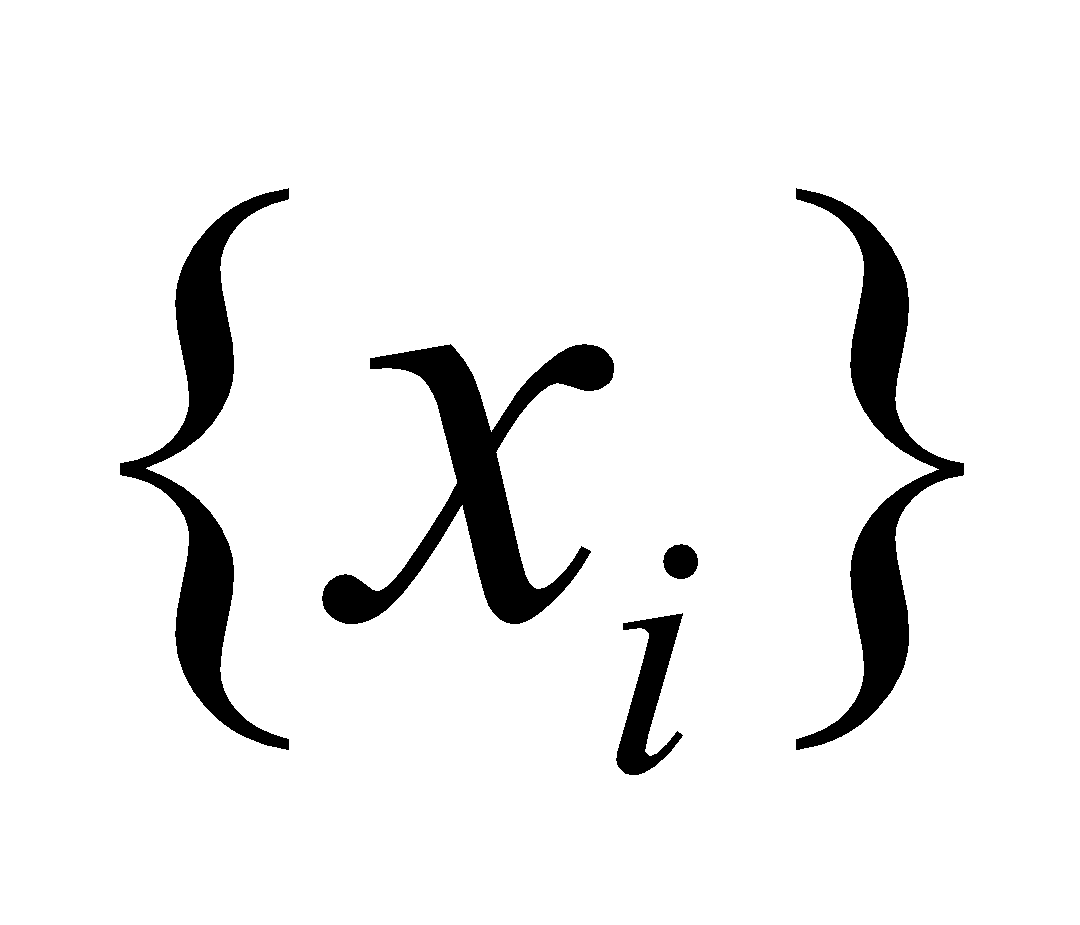


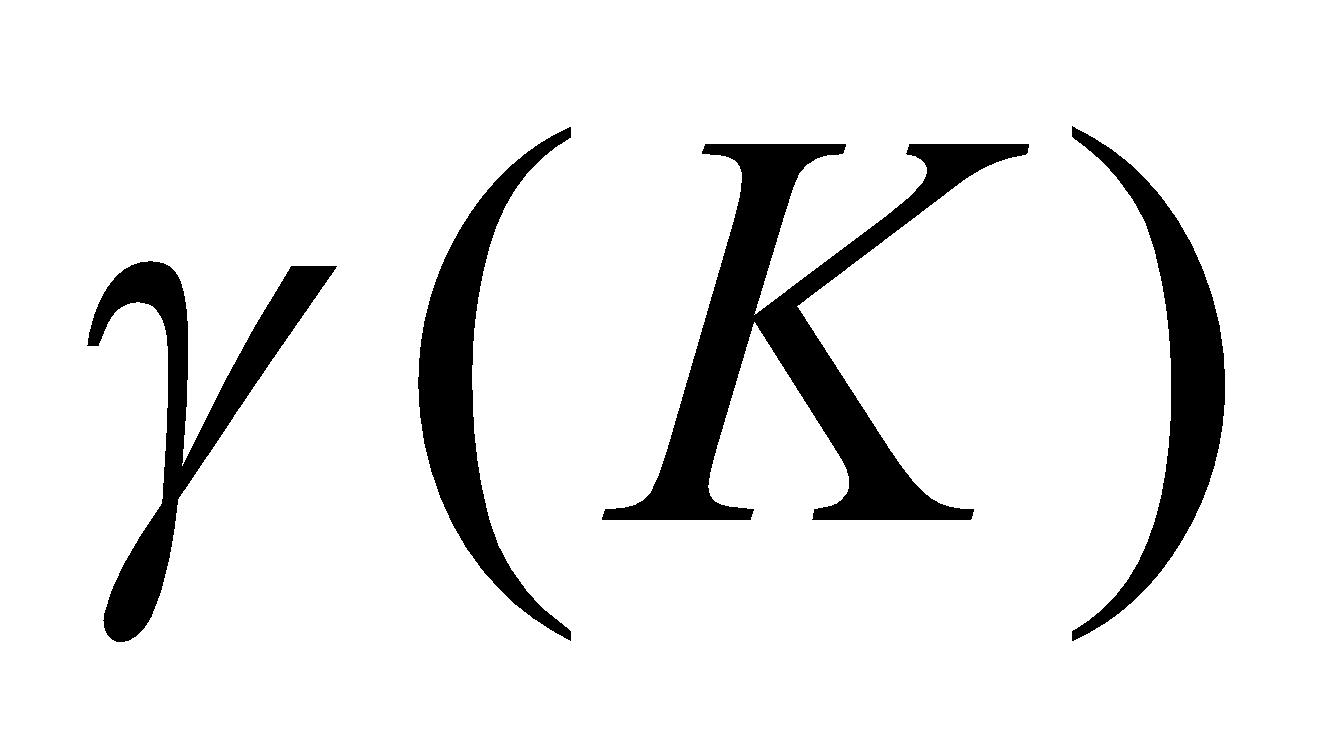
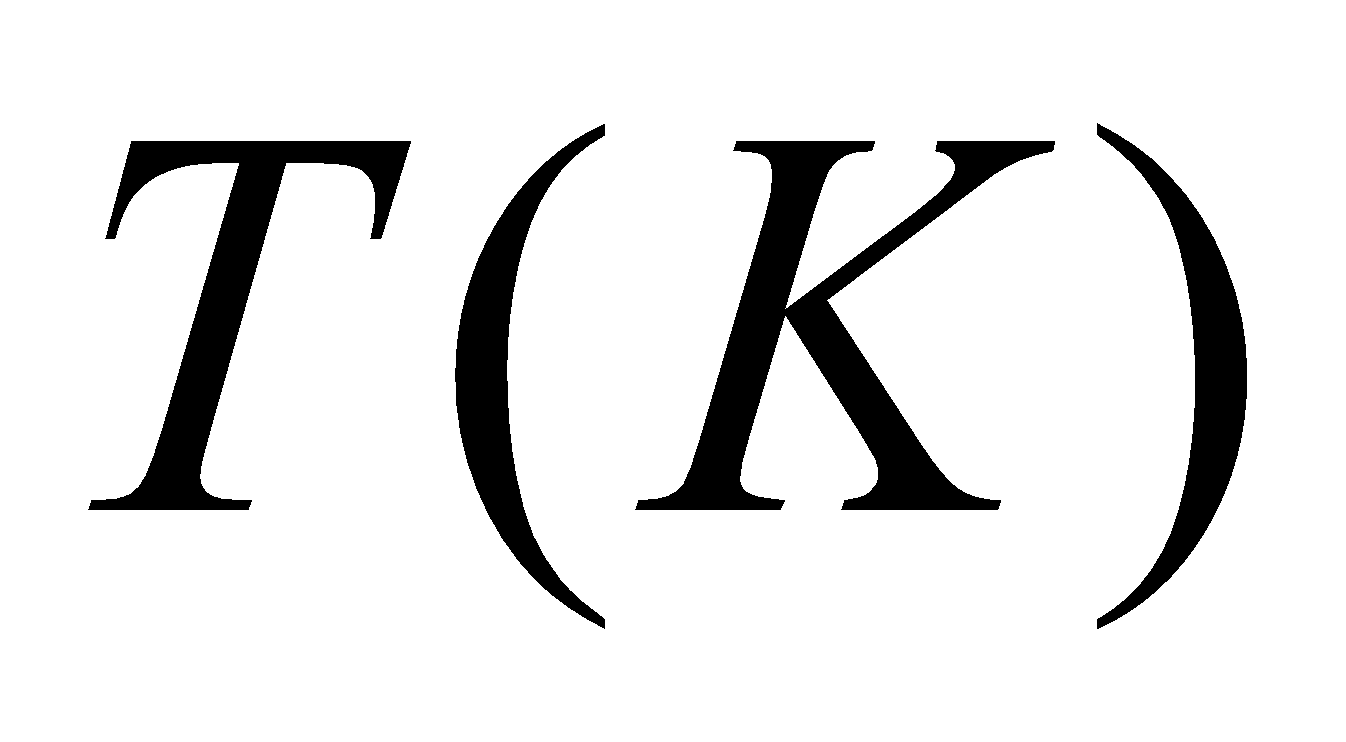
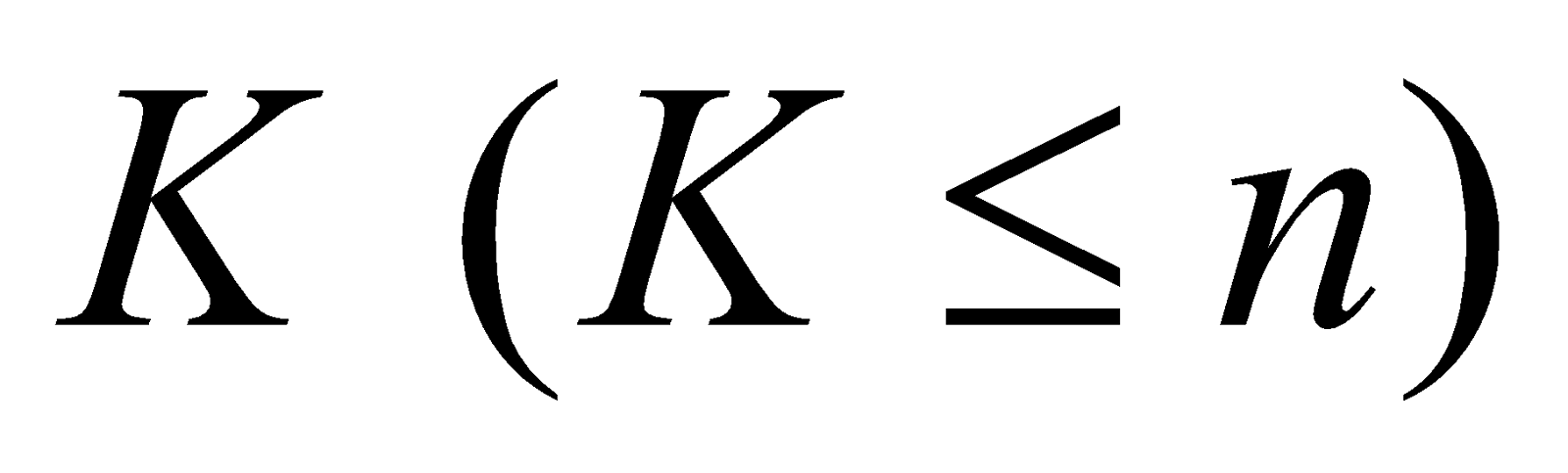
где 

**Критерий Серий:**

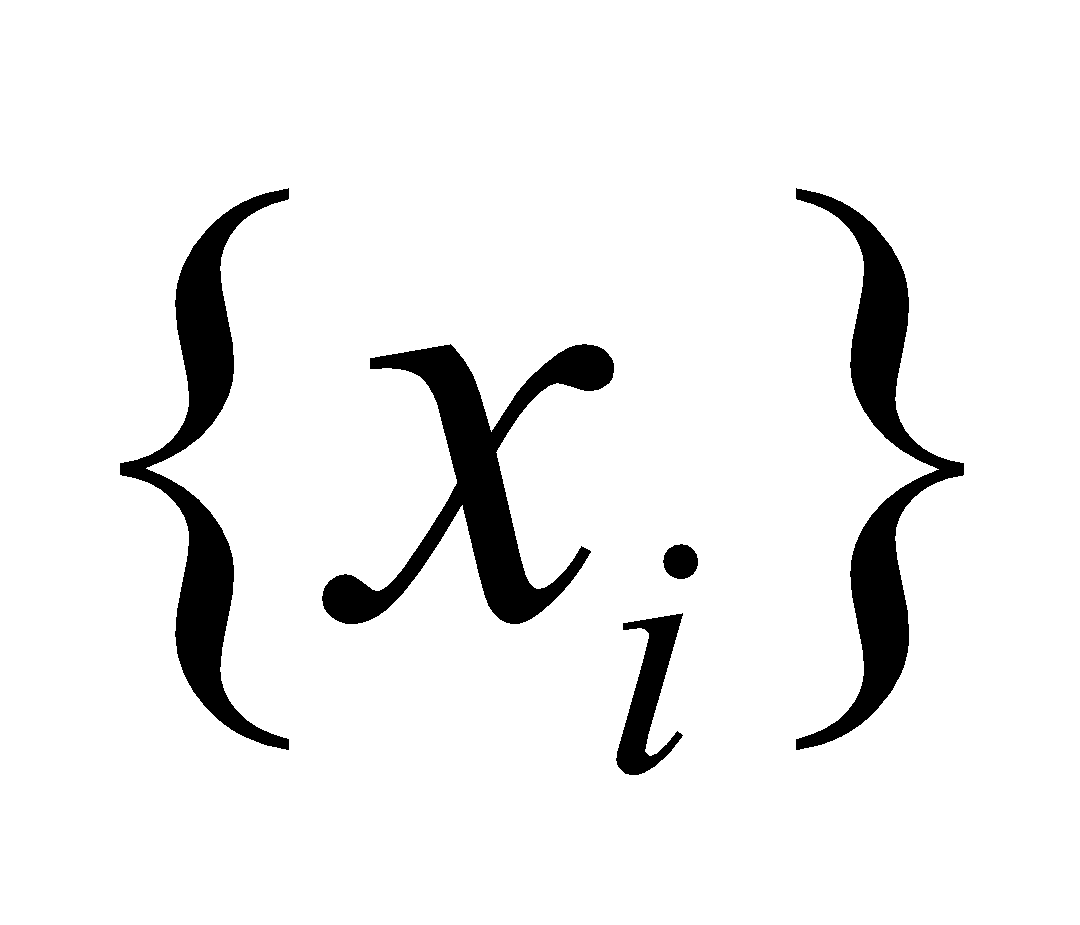
Критерий серий предназначен для проверки гипотезы о случайности выборки . Критерий основан на исследовании знаковой последовательности разностей:



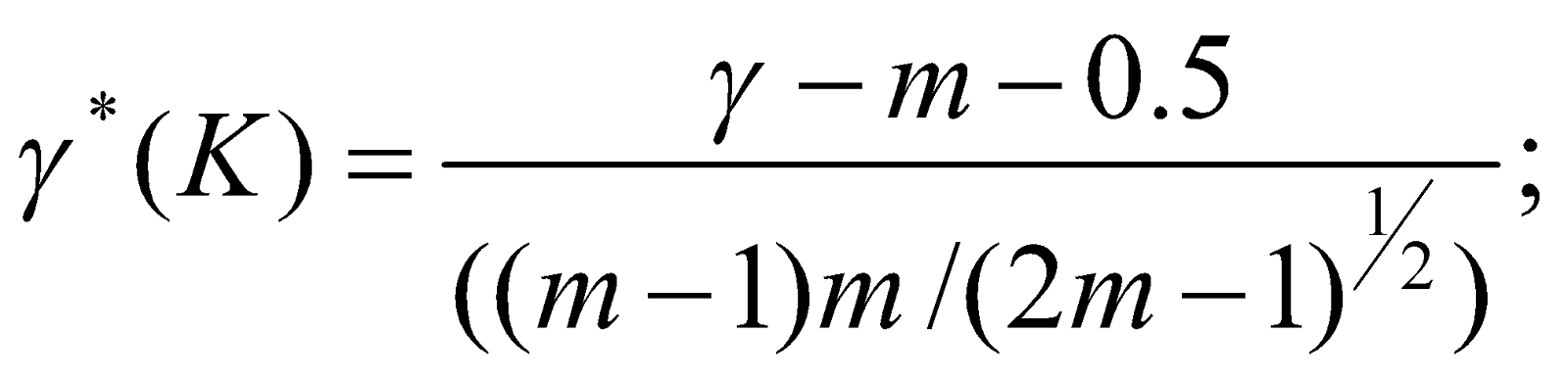
где  - медиана выборки . Знаковая последовательность состоит из знаков “+”, “-”, соответствующих разностям и характеризуется:

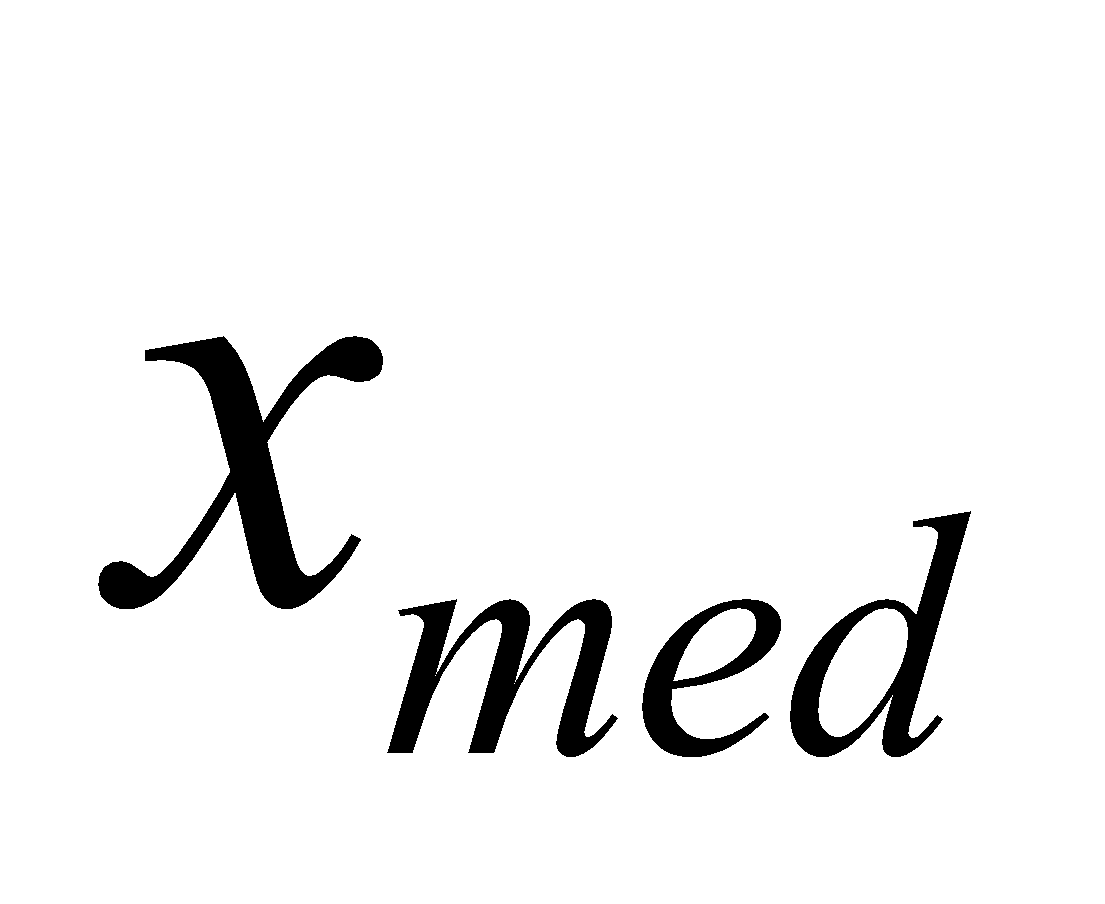
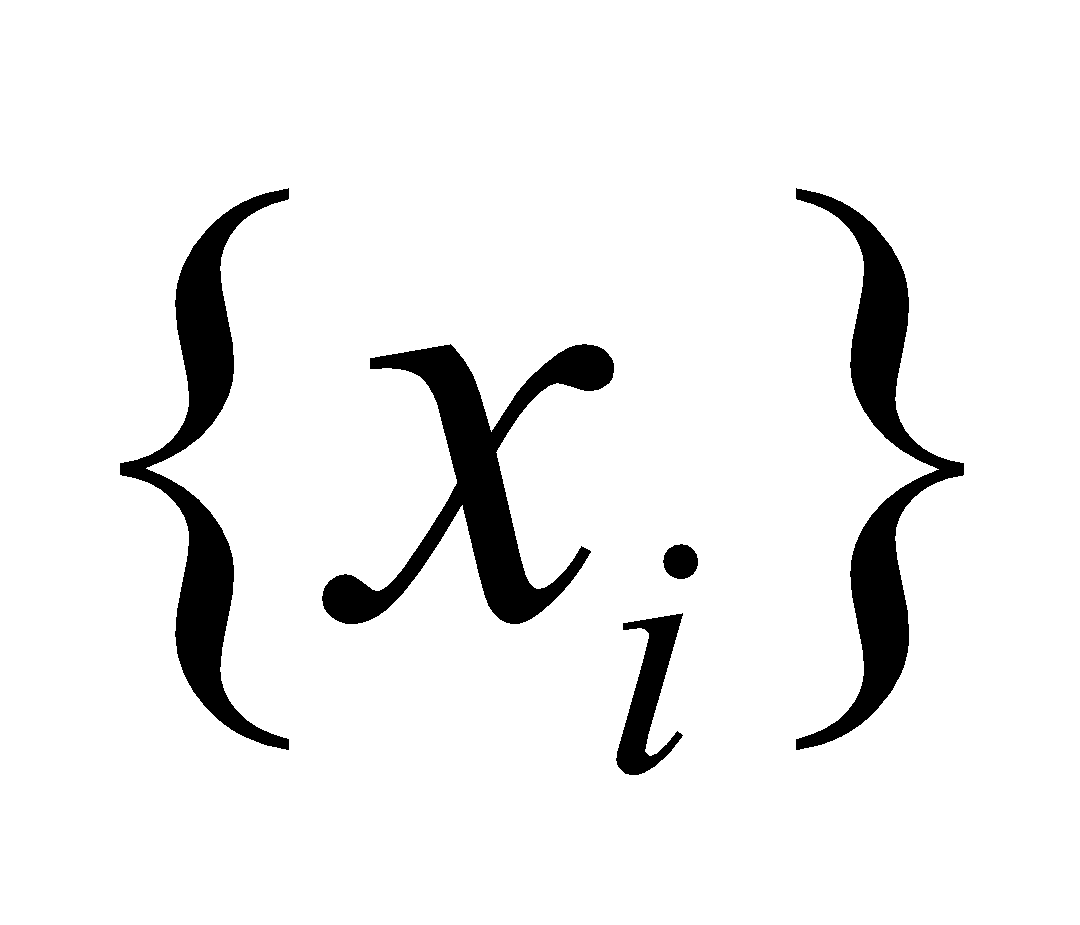
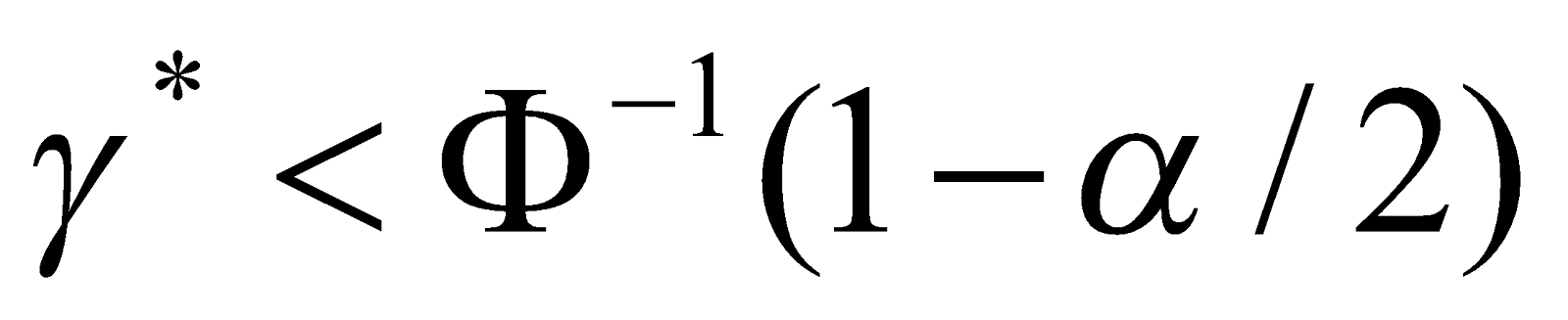
1. общим числом серий ,
2. протяжённостью самой длинной среди , где - число элементов знаковой последовательности.

Под “серией” понимается последовательность подряд идущих одинаковых знаков.

Очевидно, если  - случайная выборка, то знаковая последовательность не должна содержать слишком длинных серий, а общее число серий не должно быть слишком малым.

**Правило в случае *К*** **> 20** основано на использовании статистики:



где *m* – число наблюдений, меньших (больших) . Известно, что статистика (20) распределена асимптотически (при *К →* ∞) по стандартному нормальному закону. Отсюда правило принятия решения следующее: гипотеза о случайности выборки  принимается, если  и отвергается в противном случае (α - заданный уровень значимости).

**Код программы:**

import numpy as np

from scipy import stats

import math

import matplotlib.pyplot as plt

N = 1000

def series\_test(arr):

    med = np.median(np.sort(arr))

    new\_arr = arr[arr != med]

    max\_lenth = 1

    lenth = 1

    count = 1

    m = 0

    for i in range(1, new\_arr.size):

        if(new\_arr[i] < med):

            m+=1

        if new\_arr[i] > med and new\_arr[i-1] < med or new\_arr[i] < med and new\_arr[i-1] > med:

            count+=1

    return (count-m-0.5)/((m-1)\*m/math.sqrt(2\*m-1))

def gamma\_distr(a):

    rng = np.random.default\_rng()

    for i in range(N):

        sum = 0.0

        for j in range(a):

            sum += np.log(rng.random())

        yield sum\*(-1)

def beta\_distr(v, w2):

    gamma1 = list(gamma\_distr(v))

    gamma2 = list(gamma\_distr(w))

    for i in range(N):

        yield gamma1[i]/(gamma1[i]+gamma2[i])

v, w = 2, 4

print('y\'(K) = {}'.format(np.average(np.array([series\_test(np.array(list(beta\_distr(v, w)))) for i in range(100)]))))

**Результаты:**

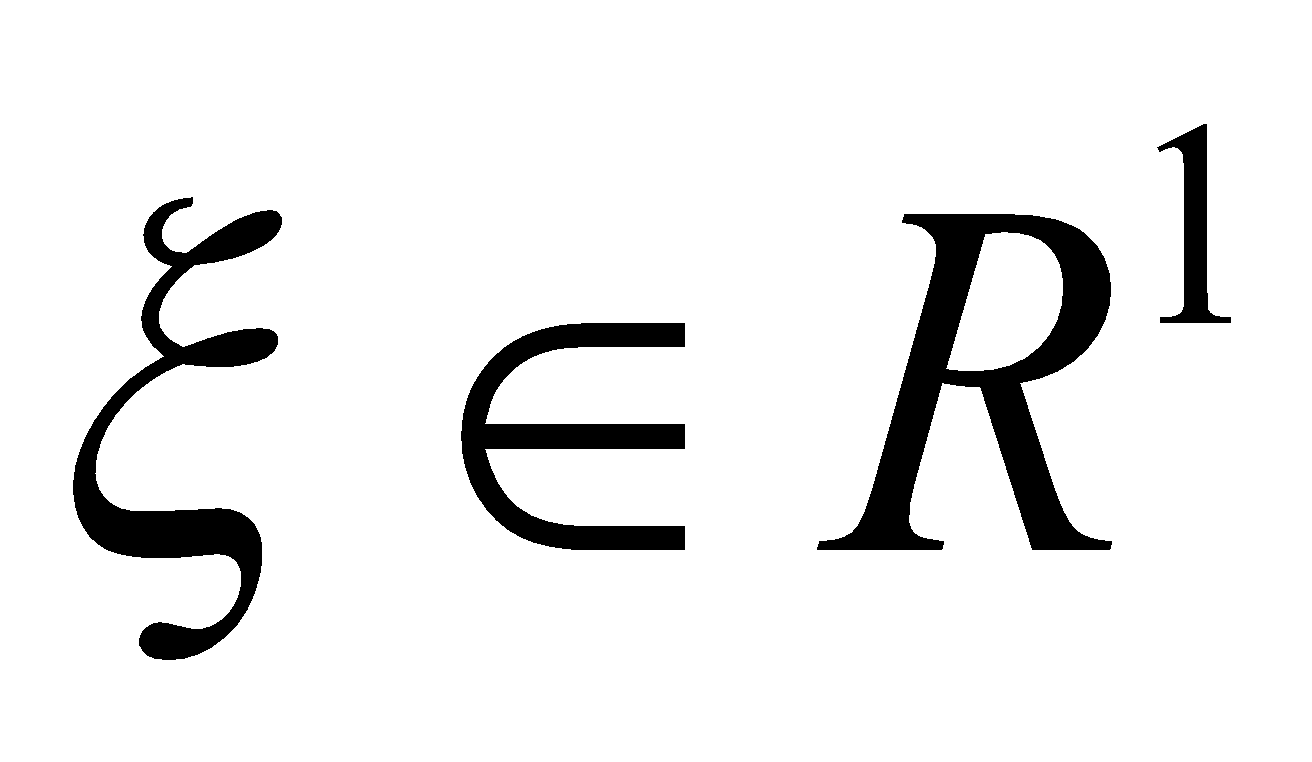
**Лабораторная работа 2.**

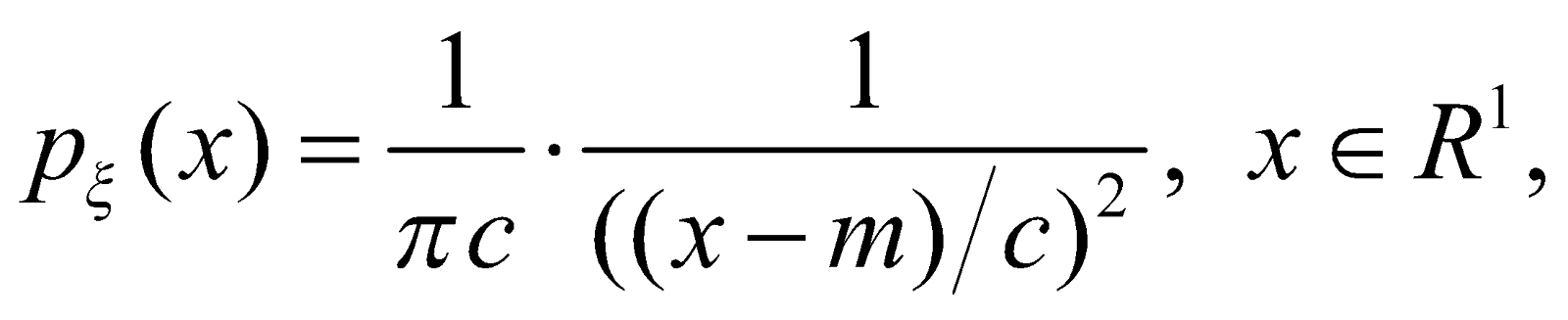
**Условие:**

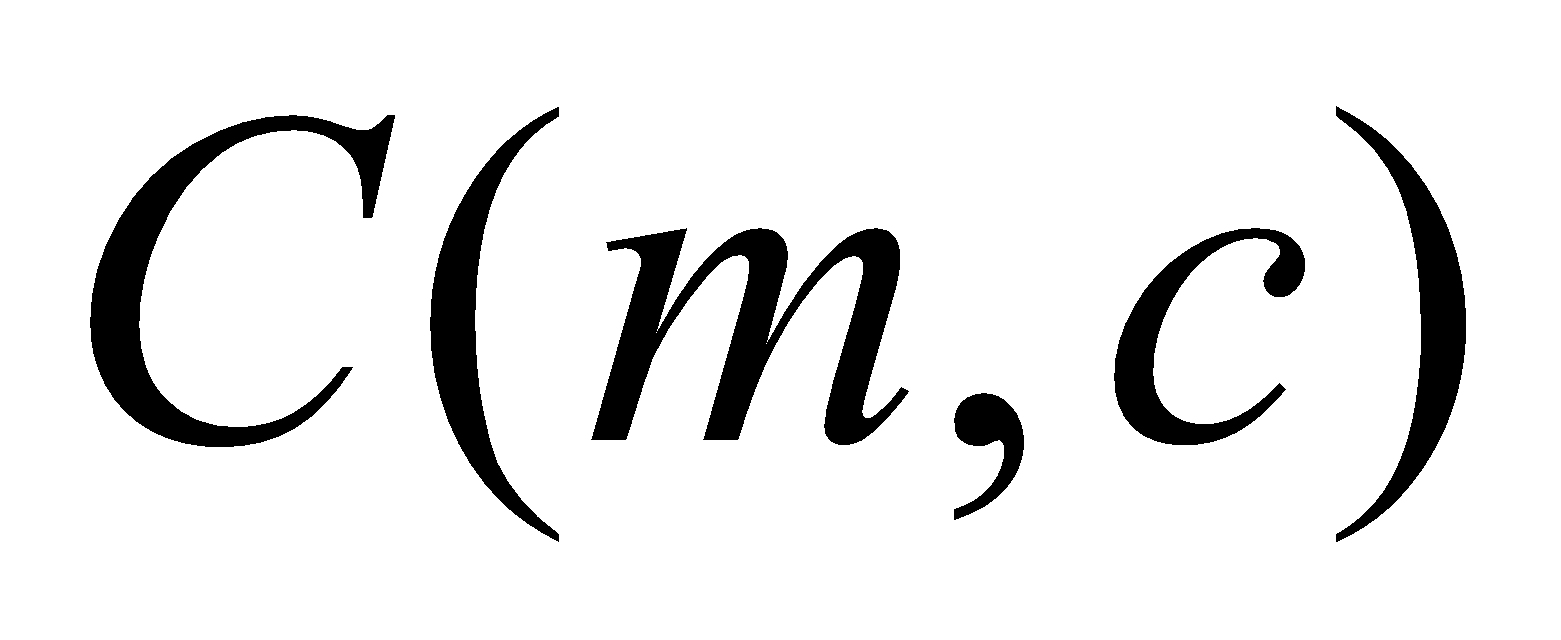
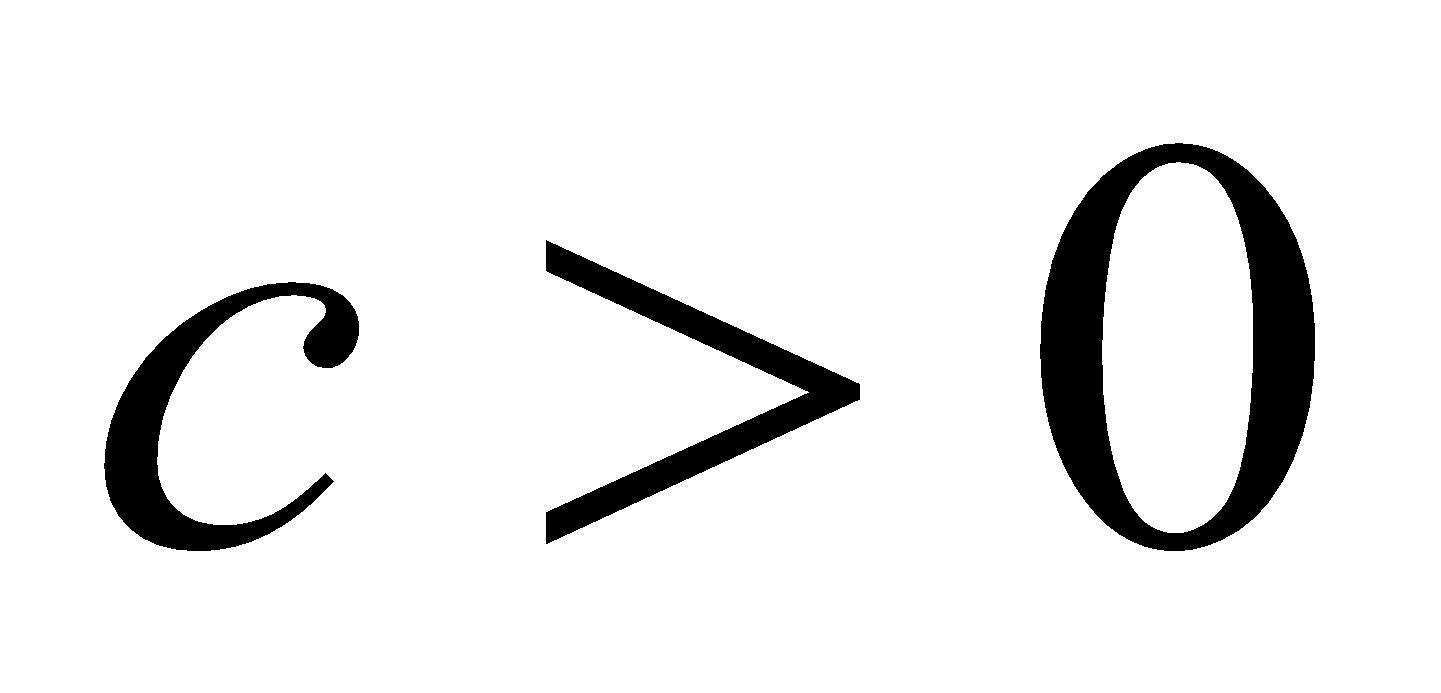
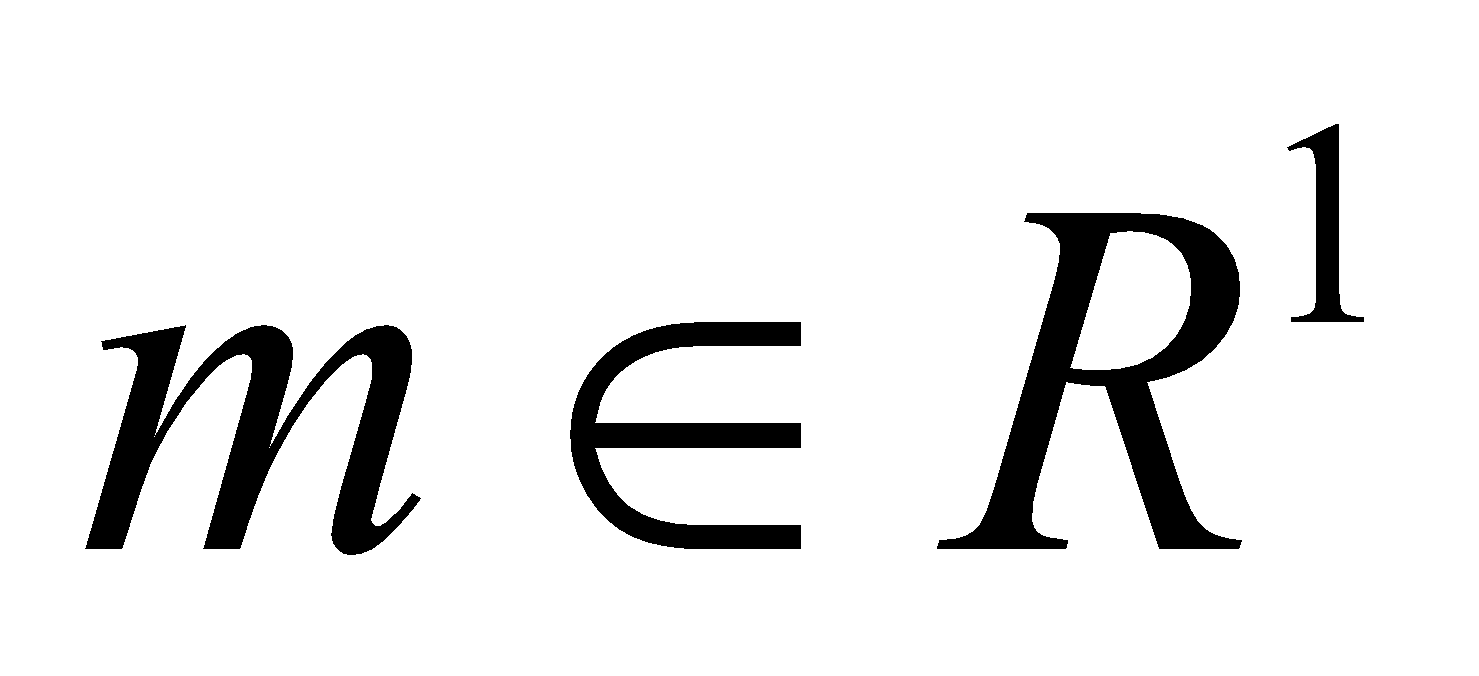
Сравнить по точности и быстродействию методы моделирования n реализаций CB ξ~ C(m,c). Положить: m=0, c=1.

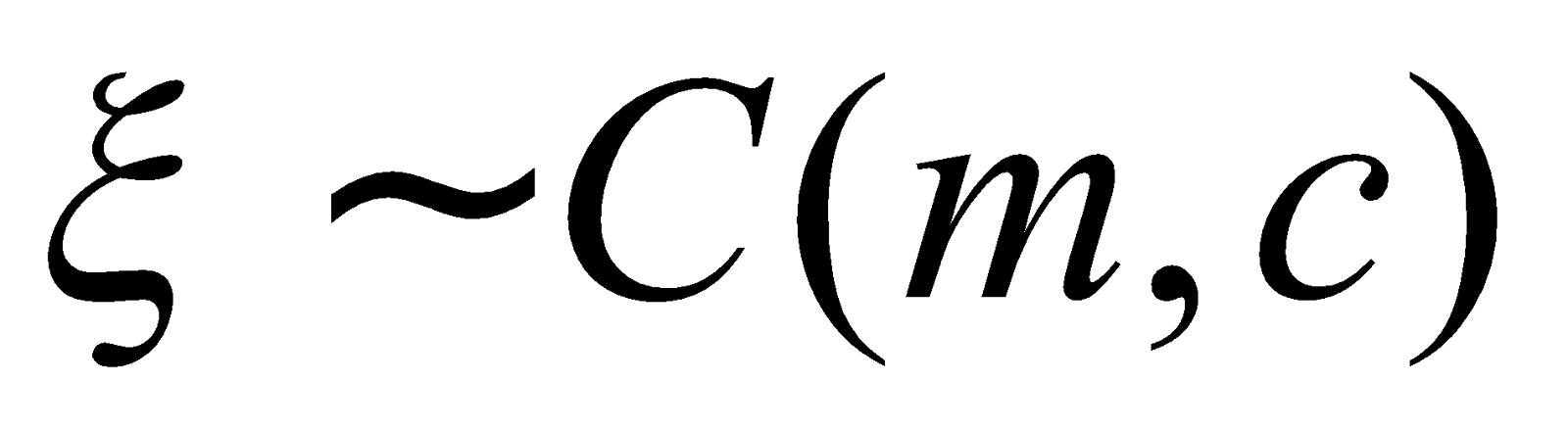
**Теория:**

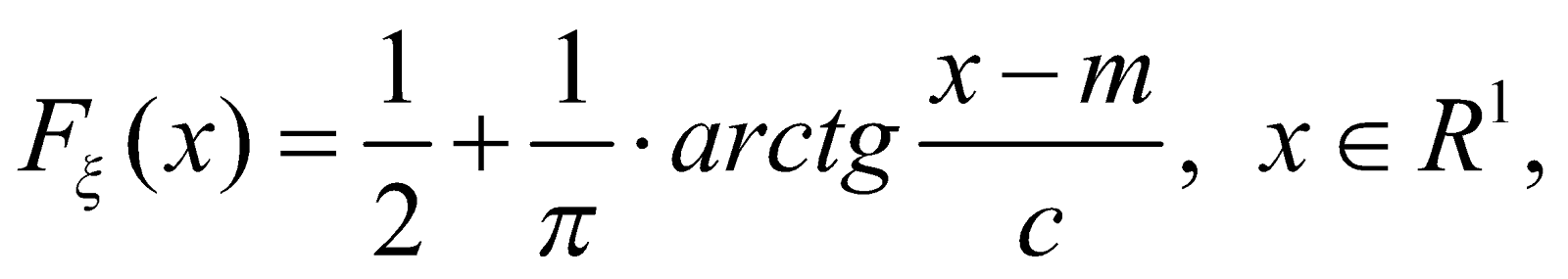
**Распределение Коши:**

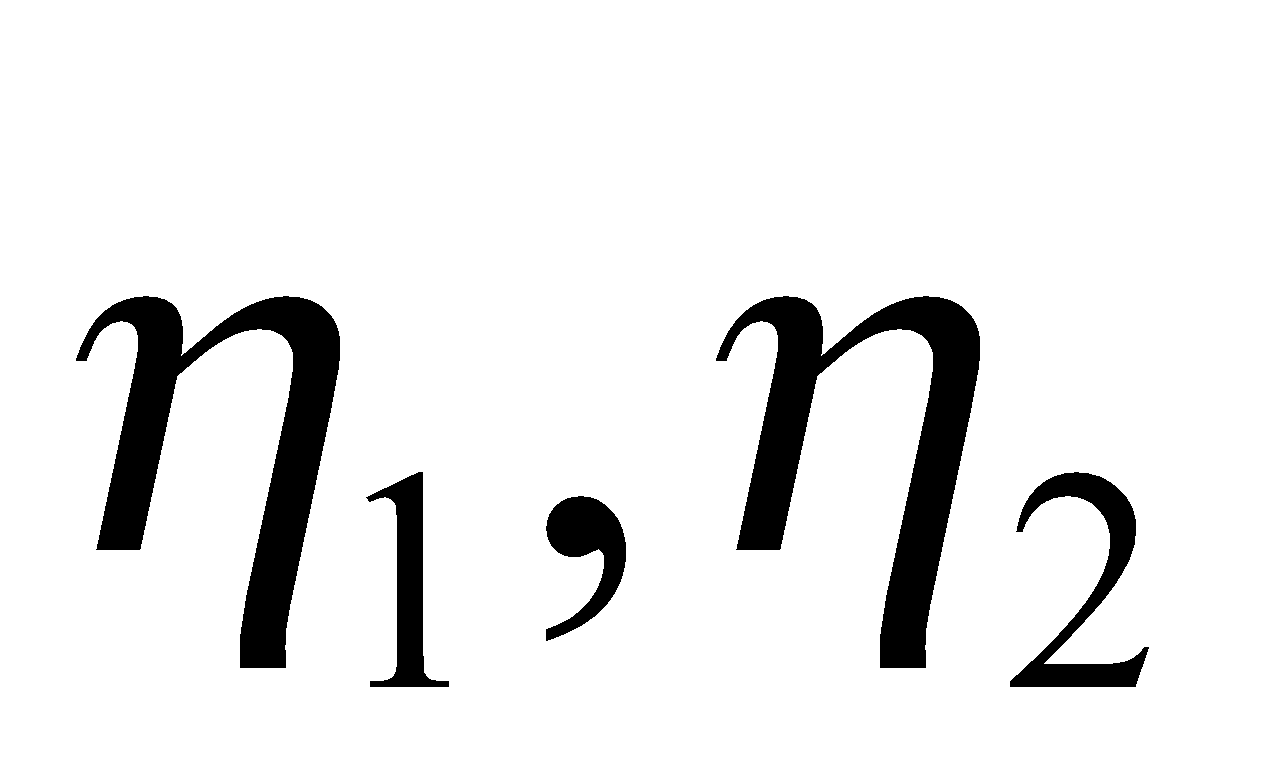
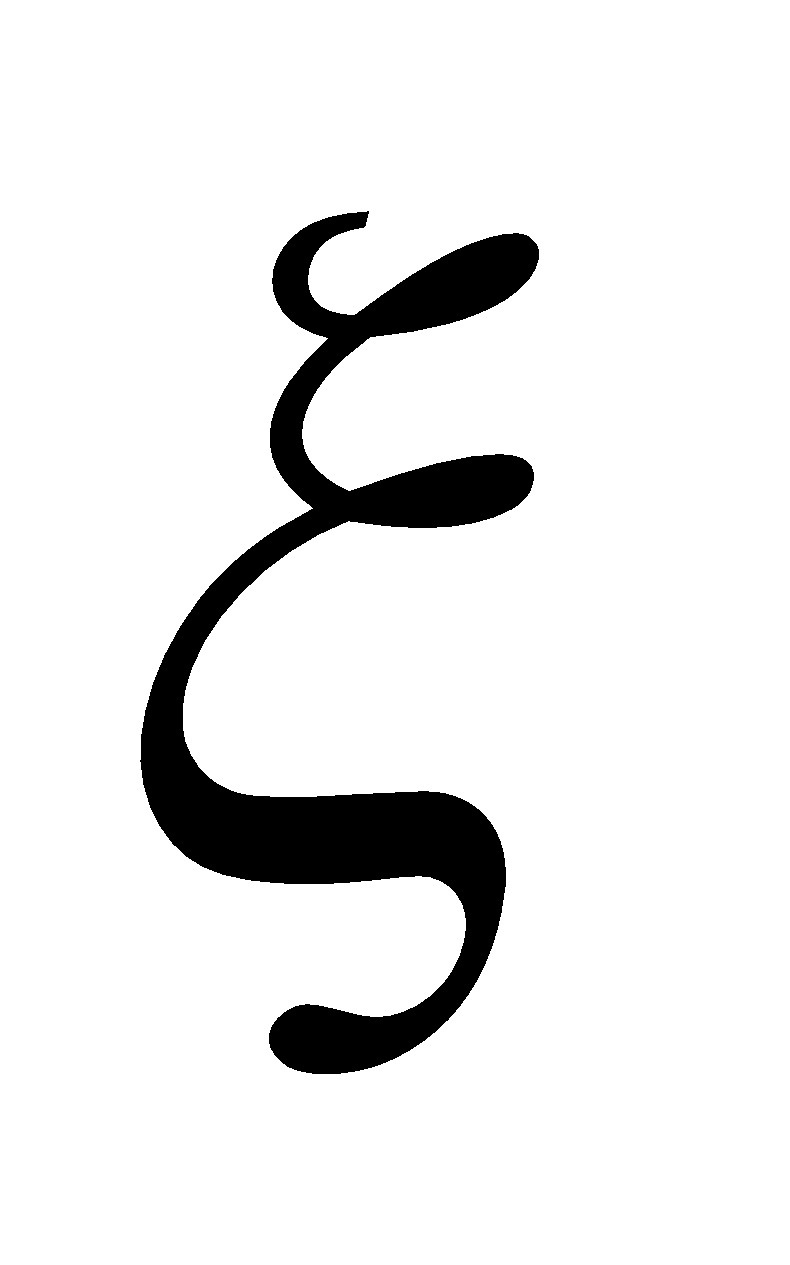
НСВ  с плотностью распределения

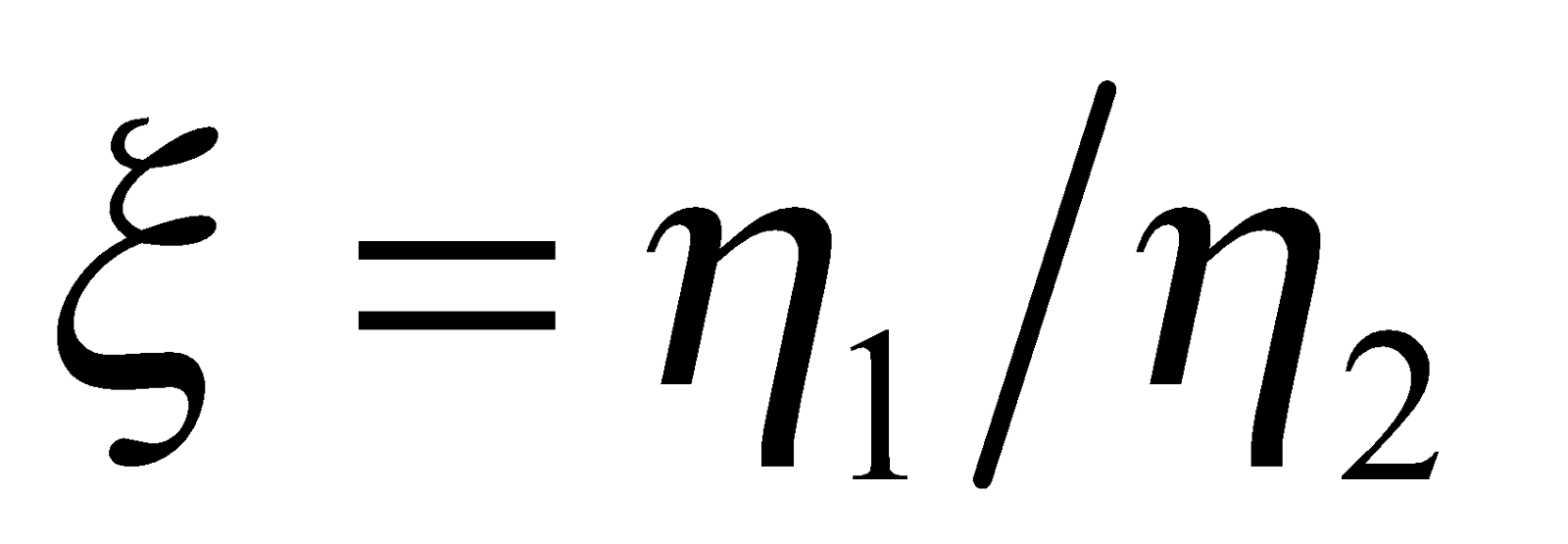


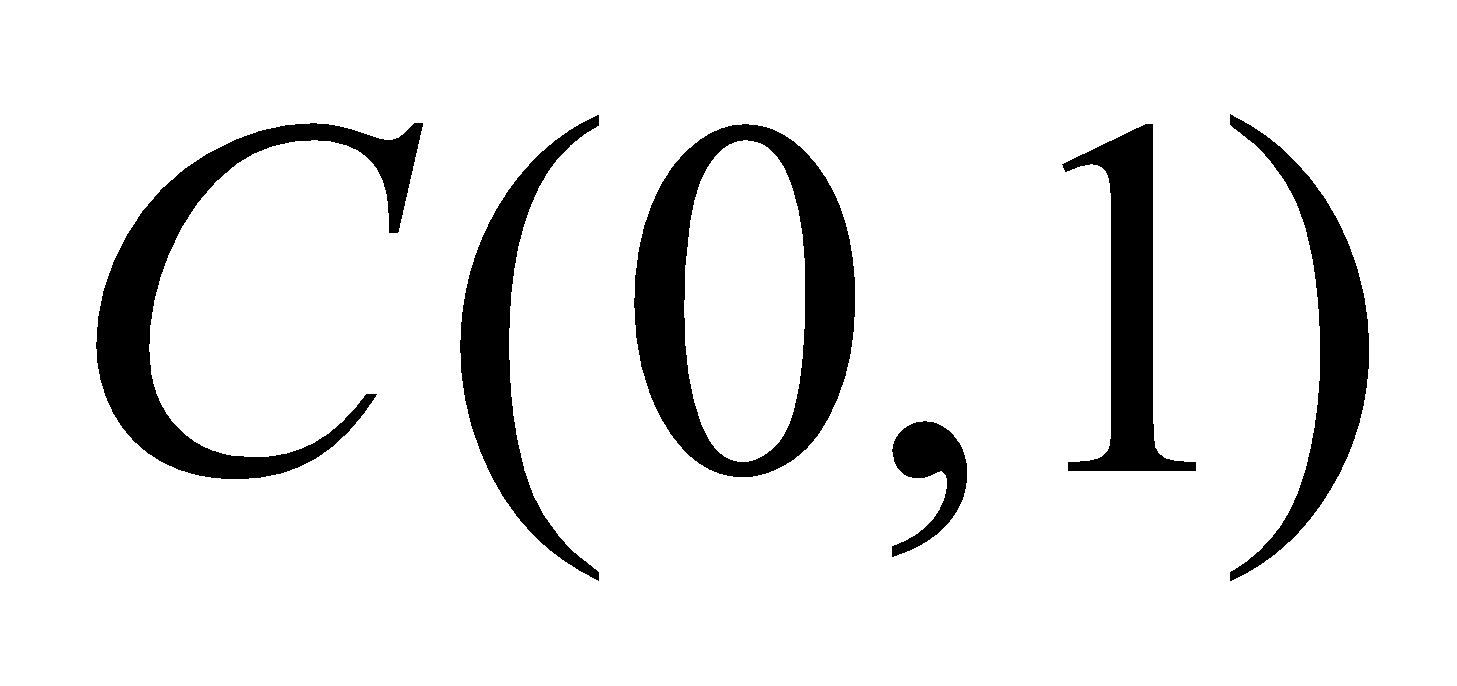
имеет распределение Коши  с параметрами: - параметр масштаба;  - параметр положения (мода, медиана).

Функция распределения СВ  имеет вид:

 (39)

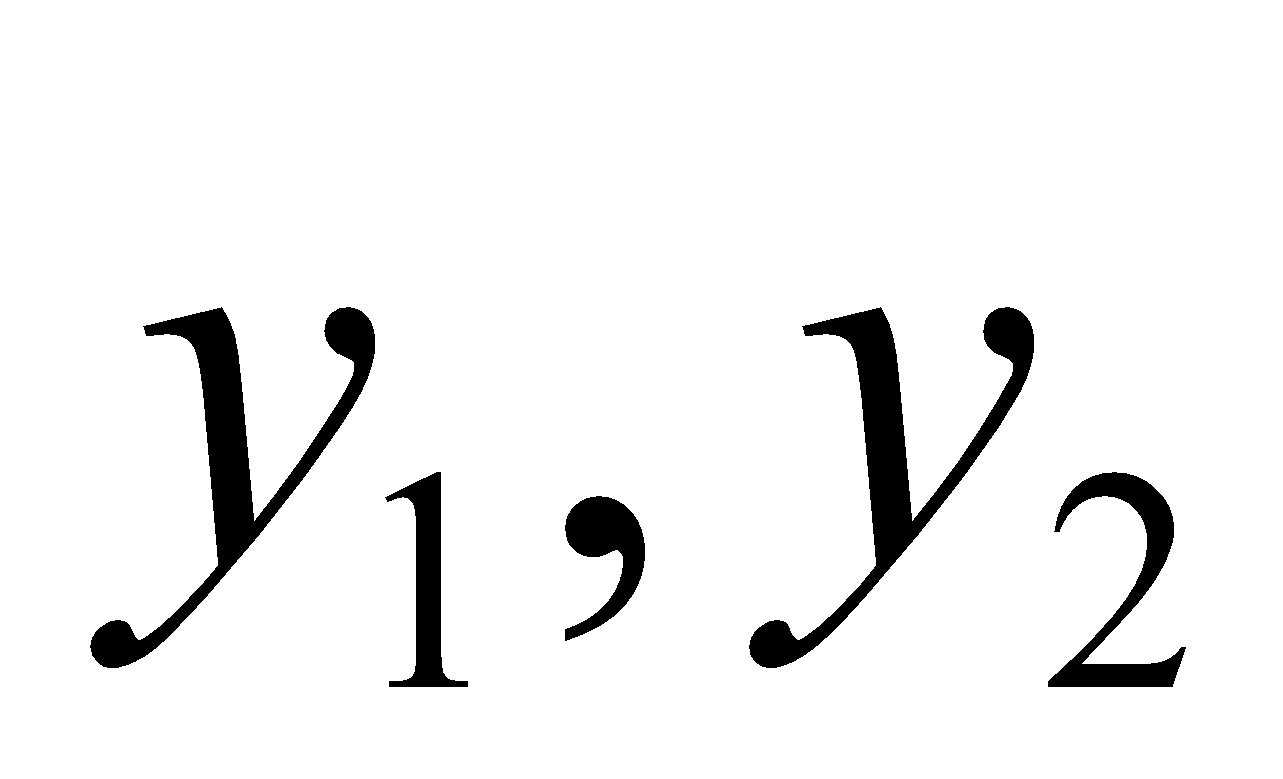
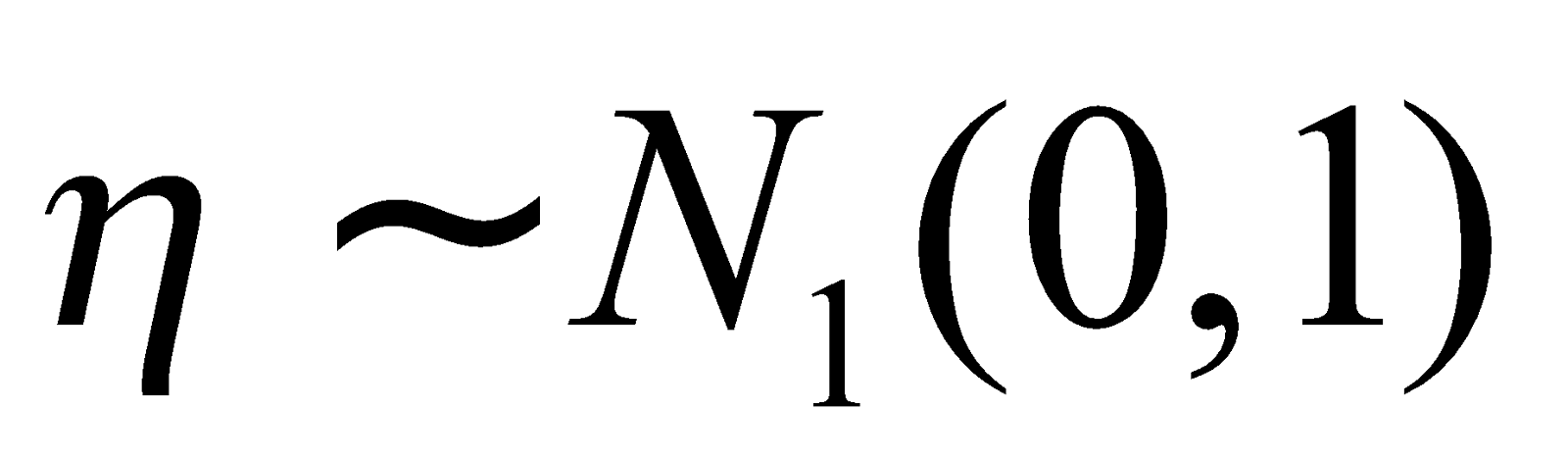
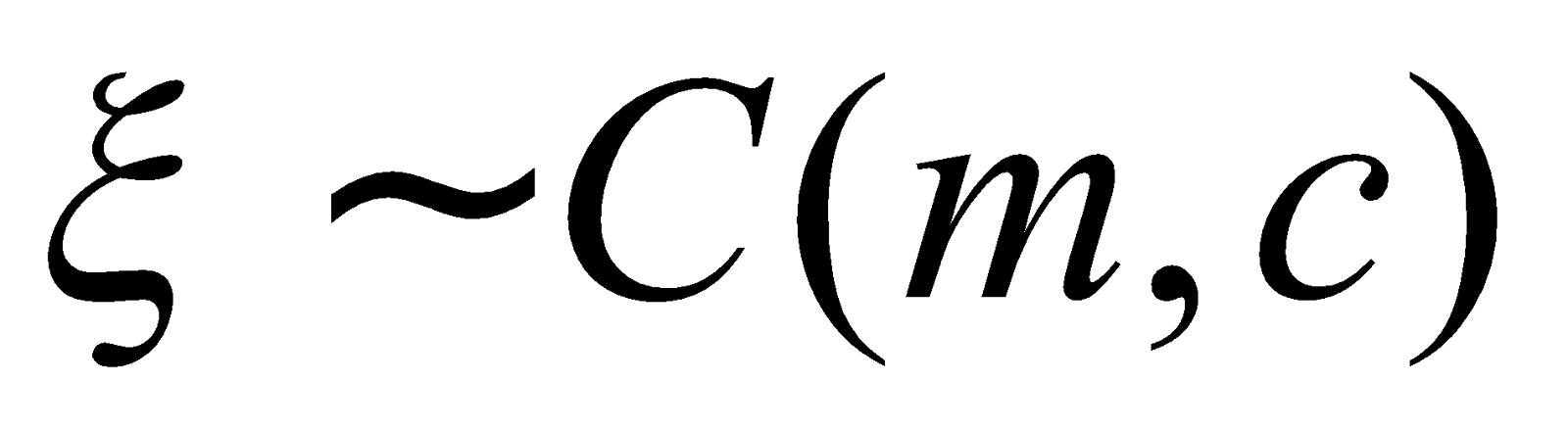
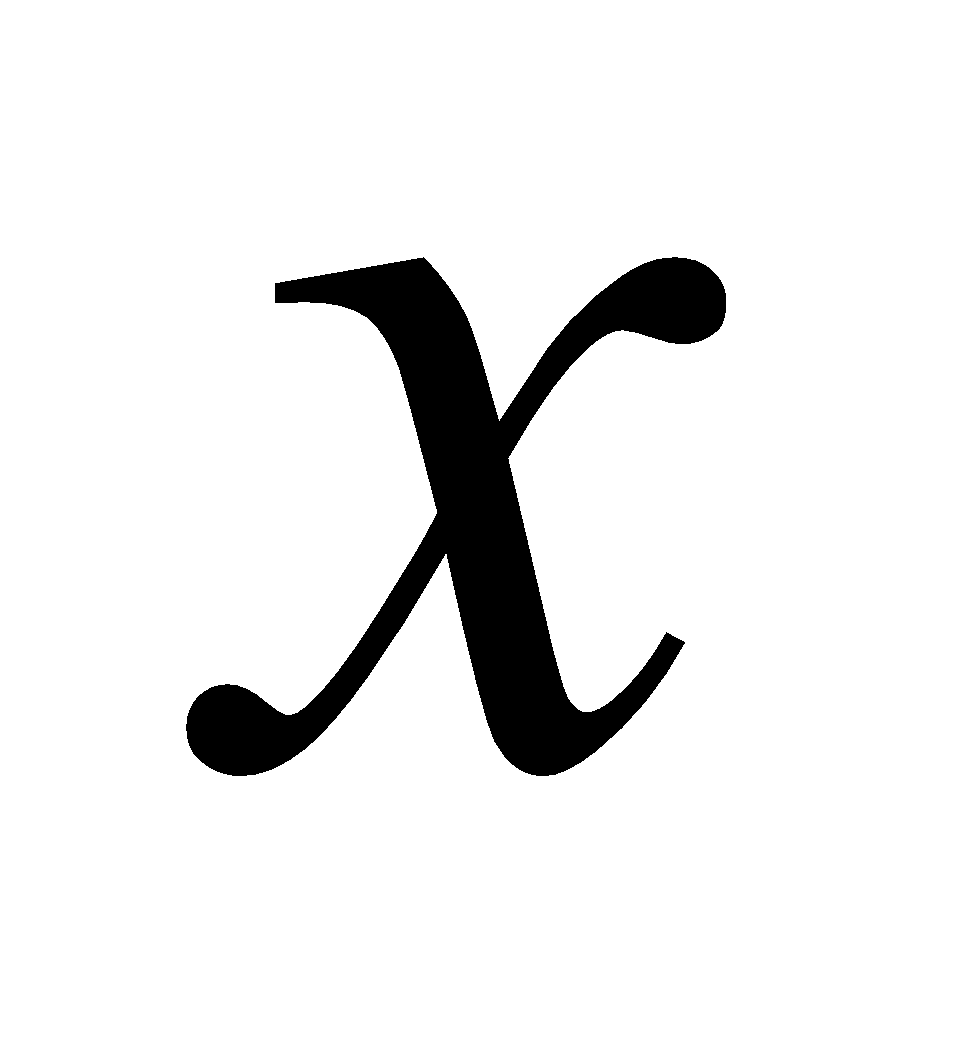
Известно, что если  - независимые стандартные гаусовские величины, то СВ  вида:

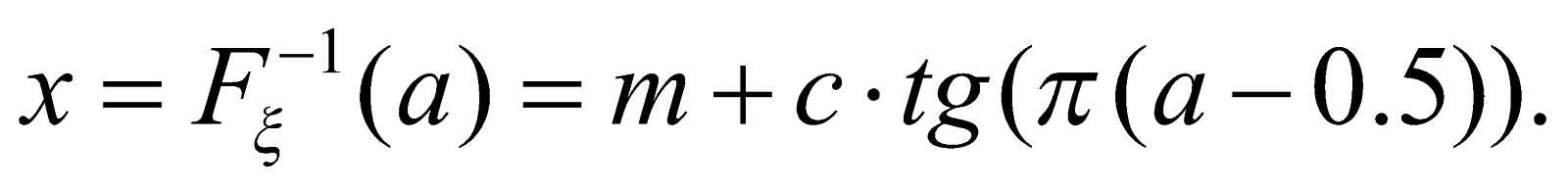


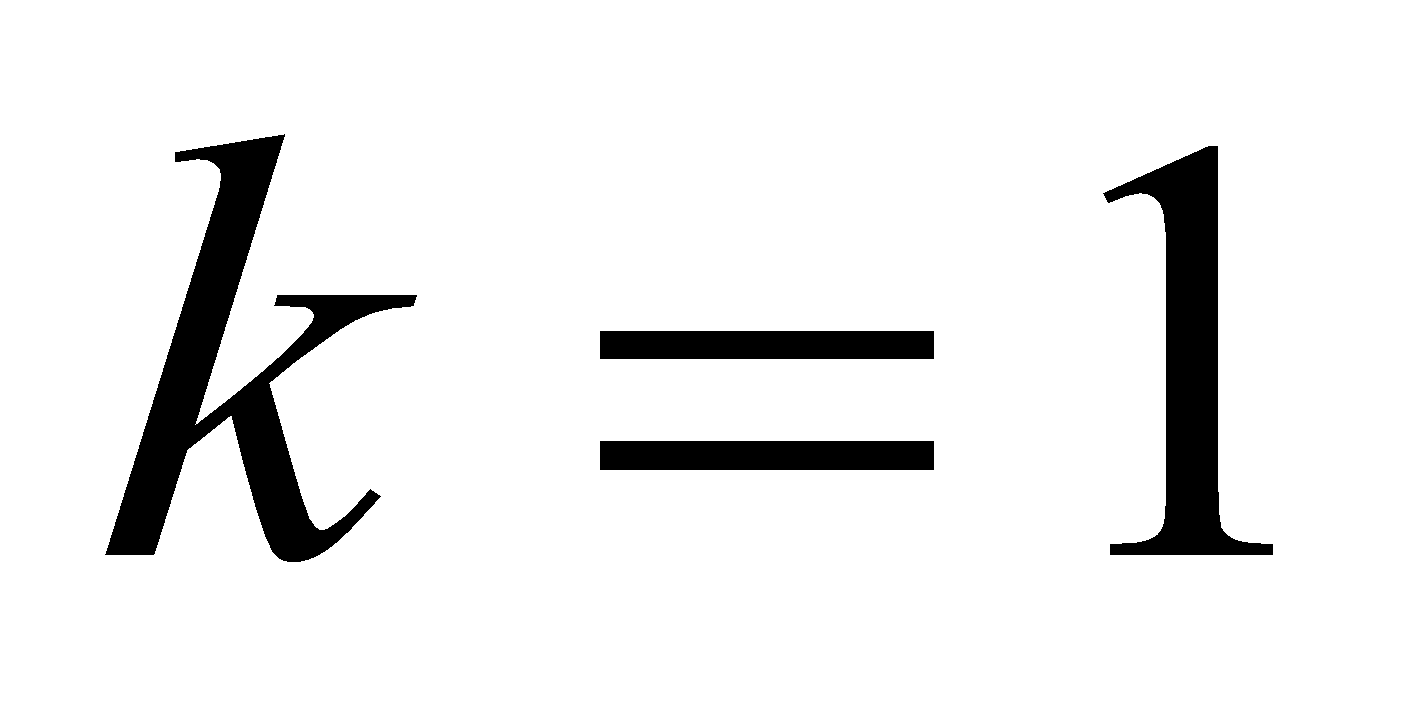
имеет распределение Коши .

**Алгоритмы моделирования**

Первый алгоритм основывается на формуле (39) и состоит из двух шагов:

1. моделирование независимых реализаций  СВ ;
2. принятие решения о том, что реализацией СВ  является величина :



Коэффициент использования БСВ .

**Код программы:**

import numpy as np

from scipy import stats

import math

import matplotlib.pyplot as plt

N = 1000

def series\_test(arr):

    med = np.median(np.sort(arr))

    new\_arr = arr[arr != med]

    max\_lenth = 1

    lenth = 1

    count = 1

    m = 0

    for i in range(1, new\_arr.size):

        if(new\_arr[i] < med):

            m+=1

        if new\_arr[i] > med and new\_arr[i-1] < med or new\_arr[i] < med and new\_arr[i-1] > med:

            count+=1

    return (count-m-0.5)/((m-1)\*m/math.sqrt(2\*m-1))

def normal\_distr():

    rng1 = np.random.default\_rng()

    rng2 = np.random.default\_rng()

    for i in range(N):

        yield math.sqrt(-2\*np.log(rng1.random()))\*math.cos(2\*math.pi\*rng2.random())

def cauchy\_dist():

    normal1 = list(normal\_distr())

    normal2 = list(normal\_distr())

    for i in range(N):

        yield math.tan(math.pi\*(normal1[i]/normal2[i]-0.5))

print('y\'(K) = {}'.format(np.average(np.array([series\_test(np.array(list(cauchy\_dist()))) for i in range(100)]))))

**Результаты:**

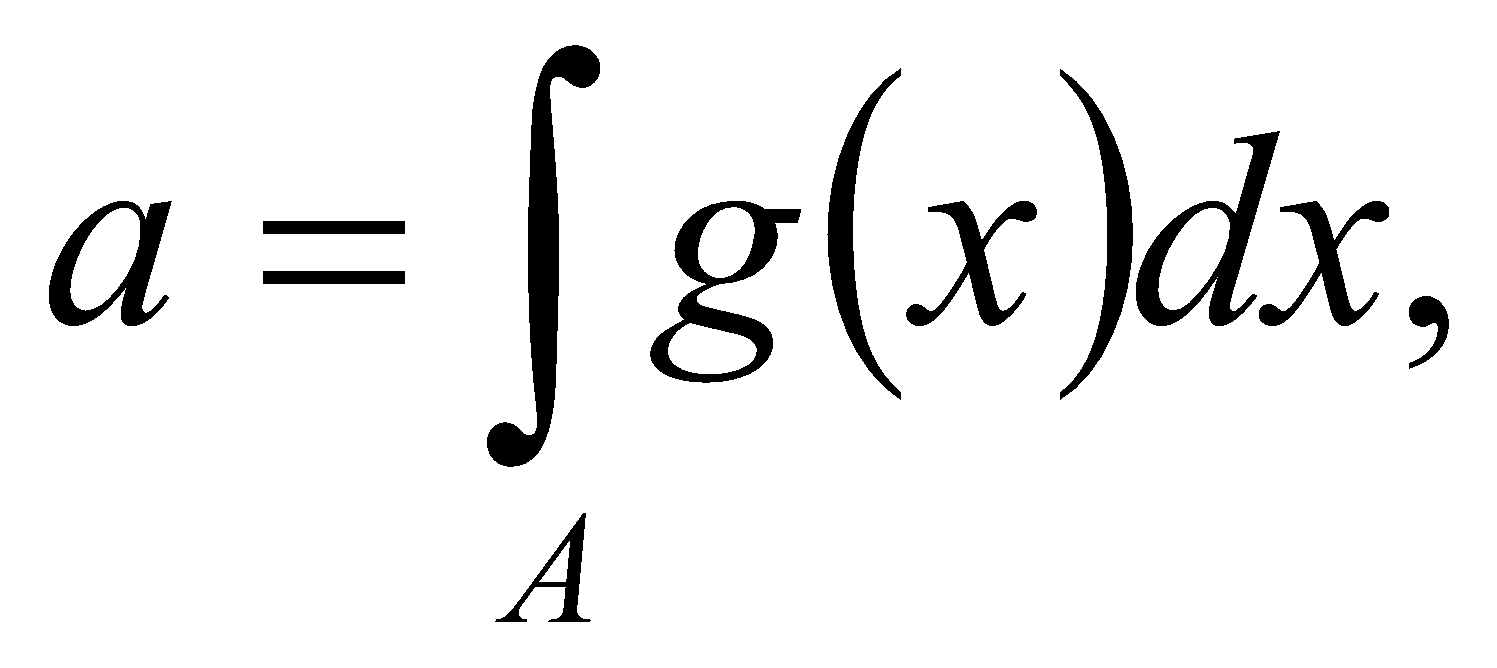
**Лабораторная работа 3.**

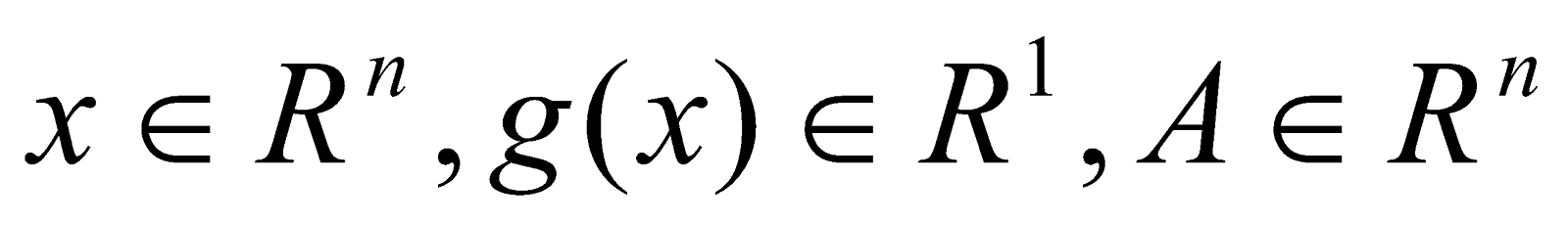
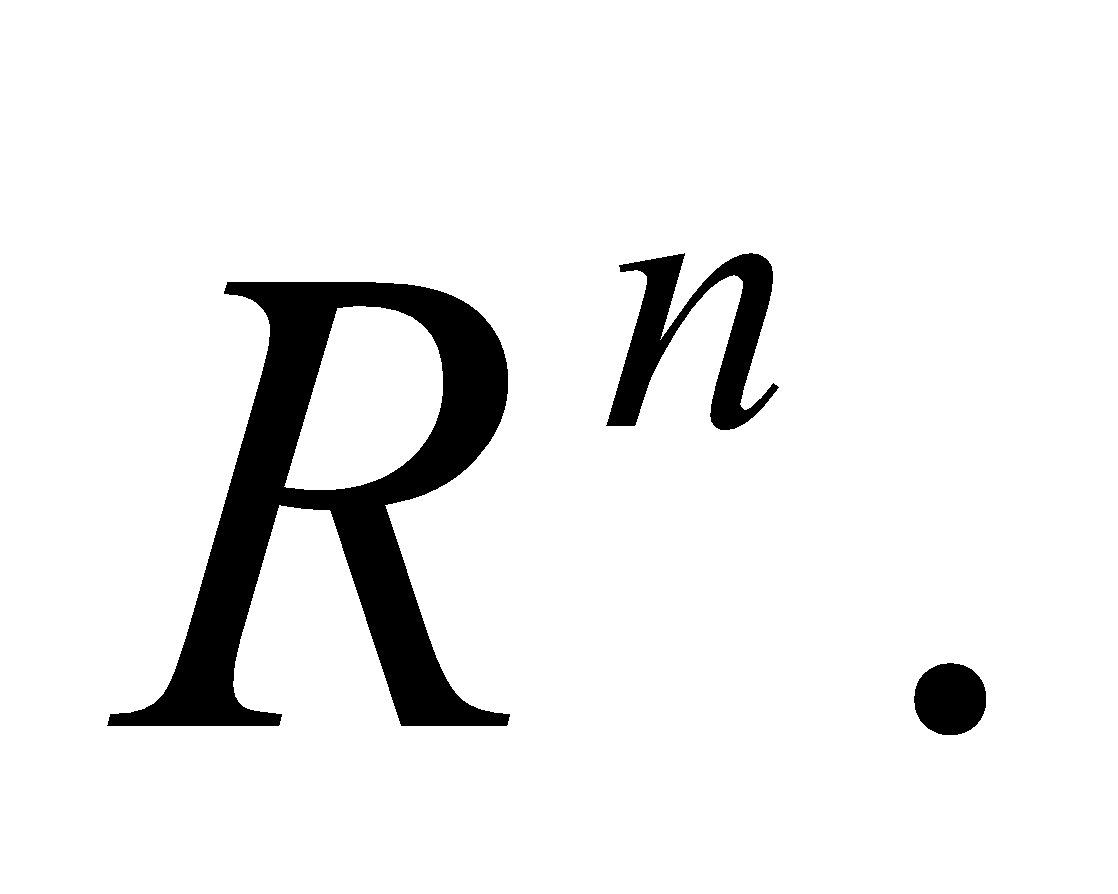
**Условие:**

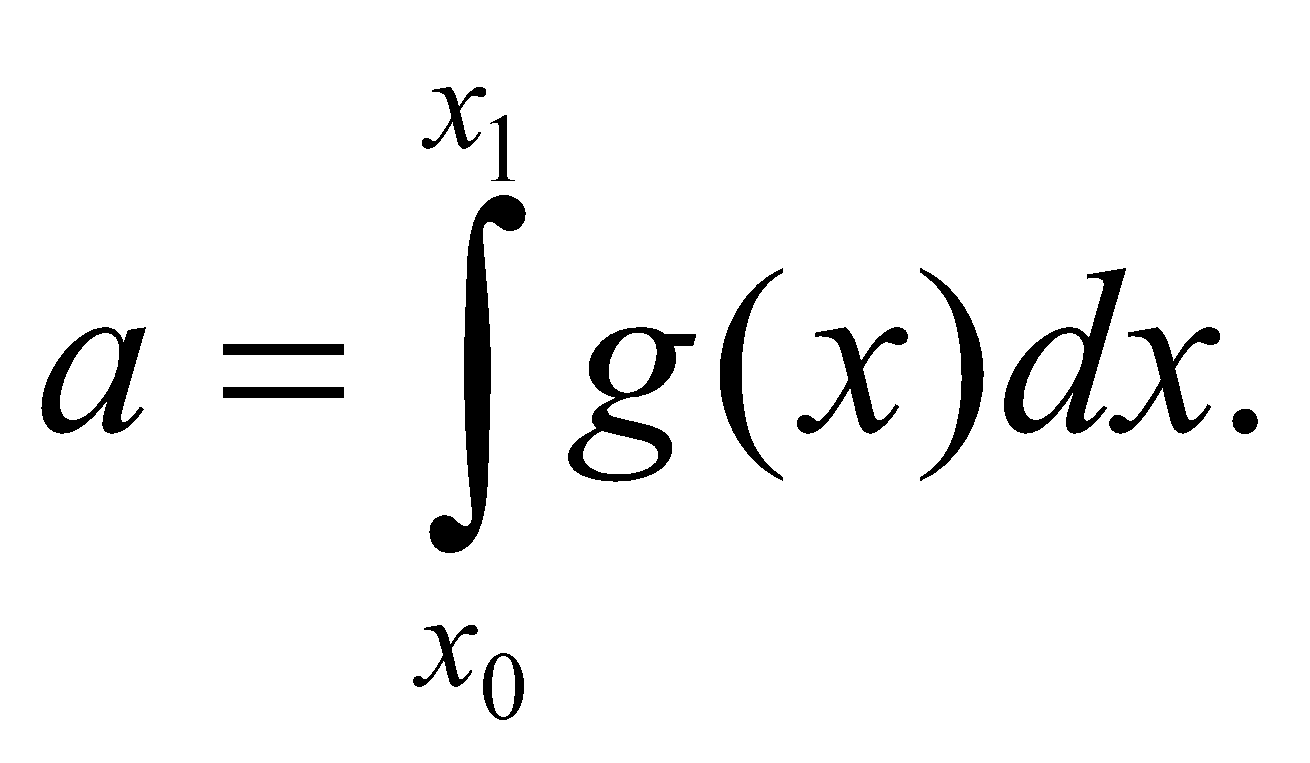
Вычислить интеграл

**Теория:**

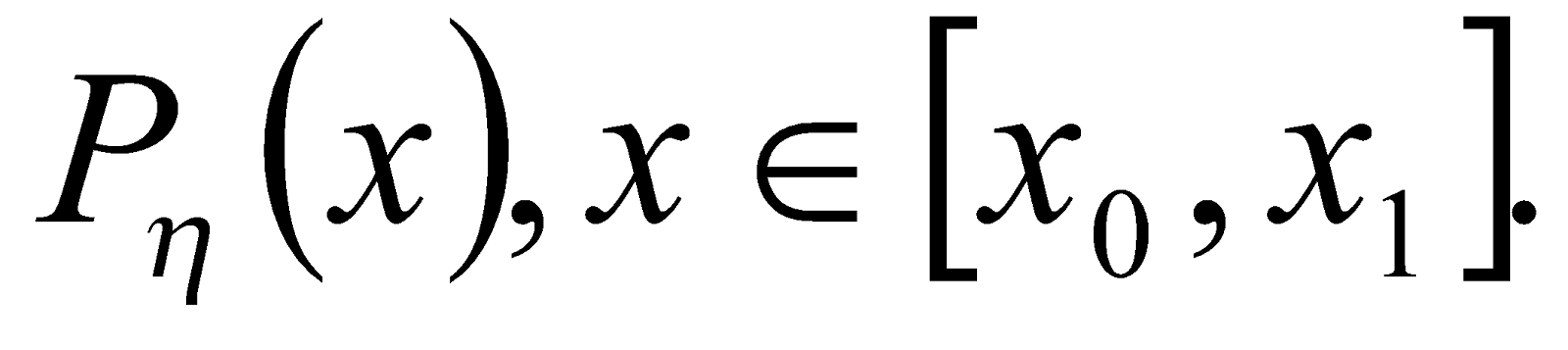
Рассмотрим задачу приближенного вычисления интеграла



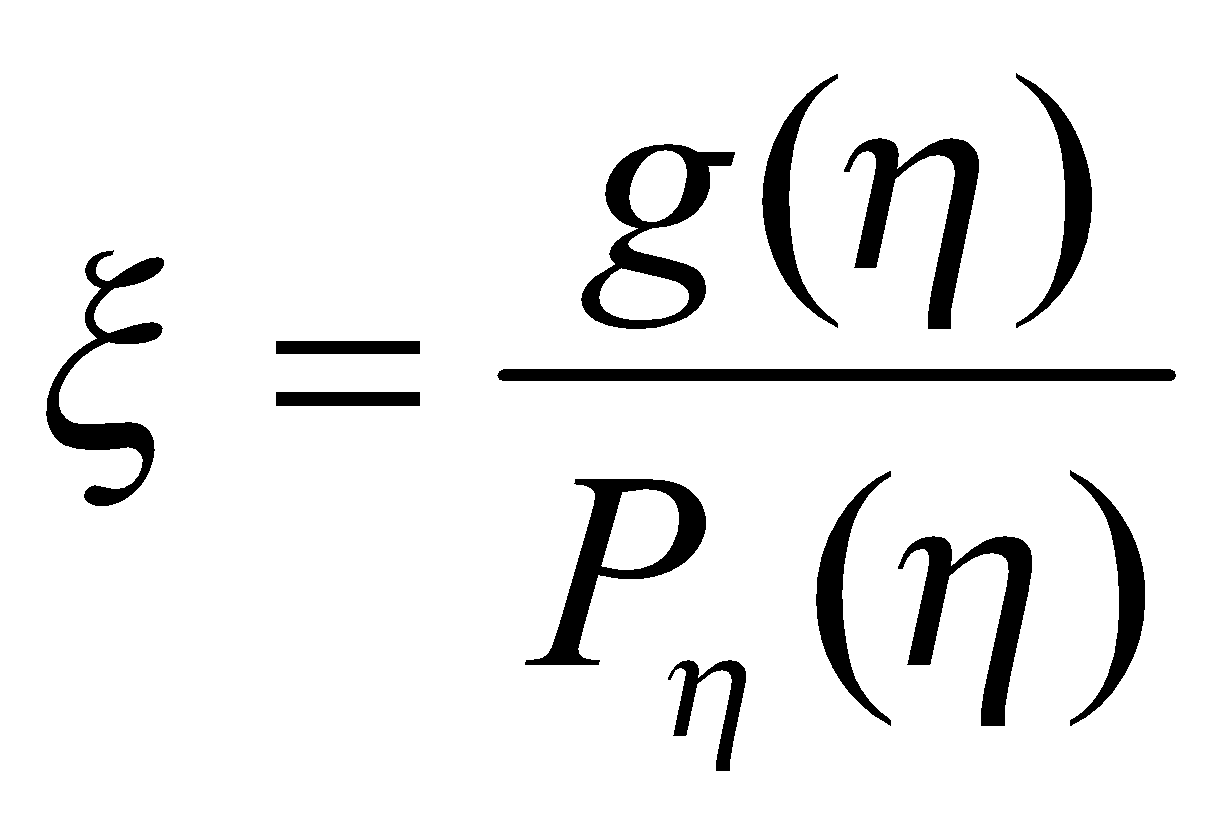
где - подмножество из  При n=1 имеем определенный интеграл вида

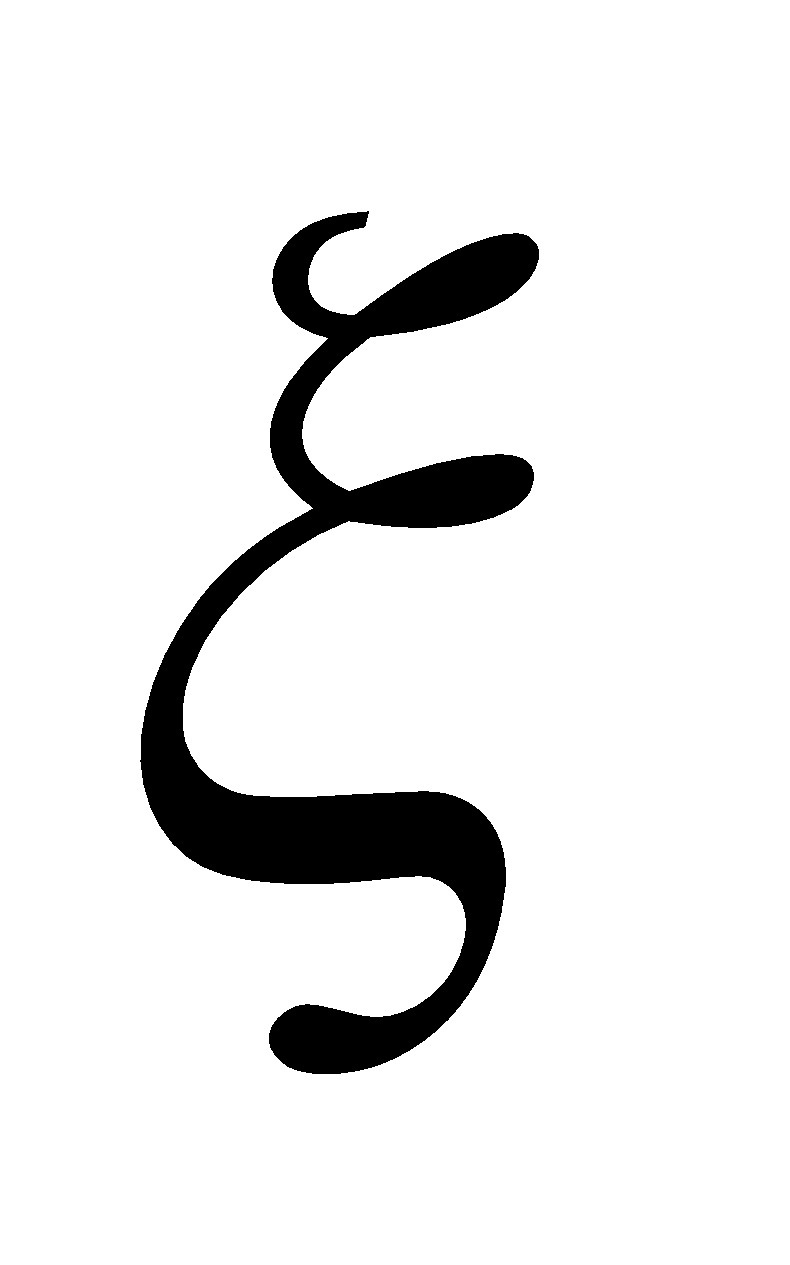


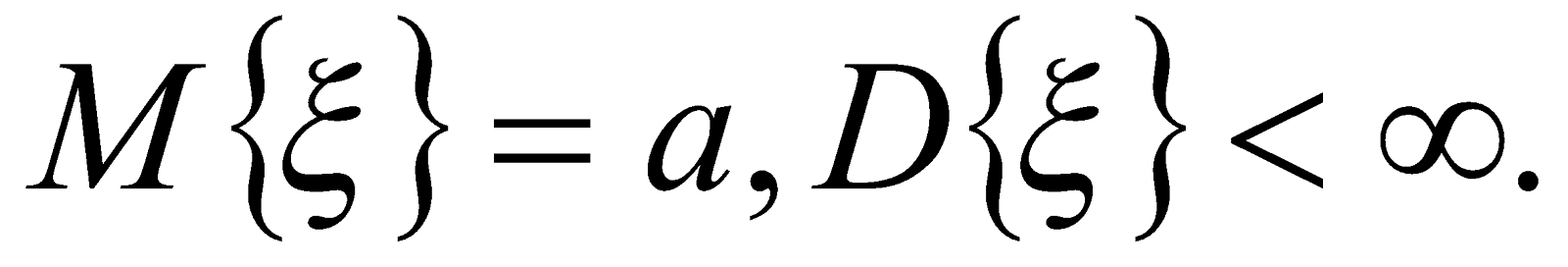
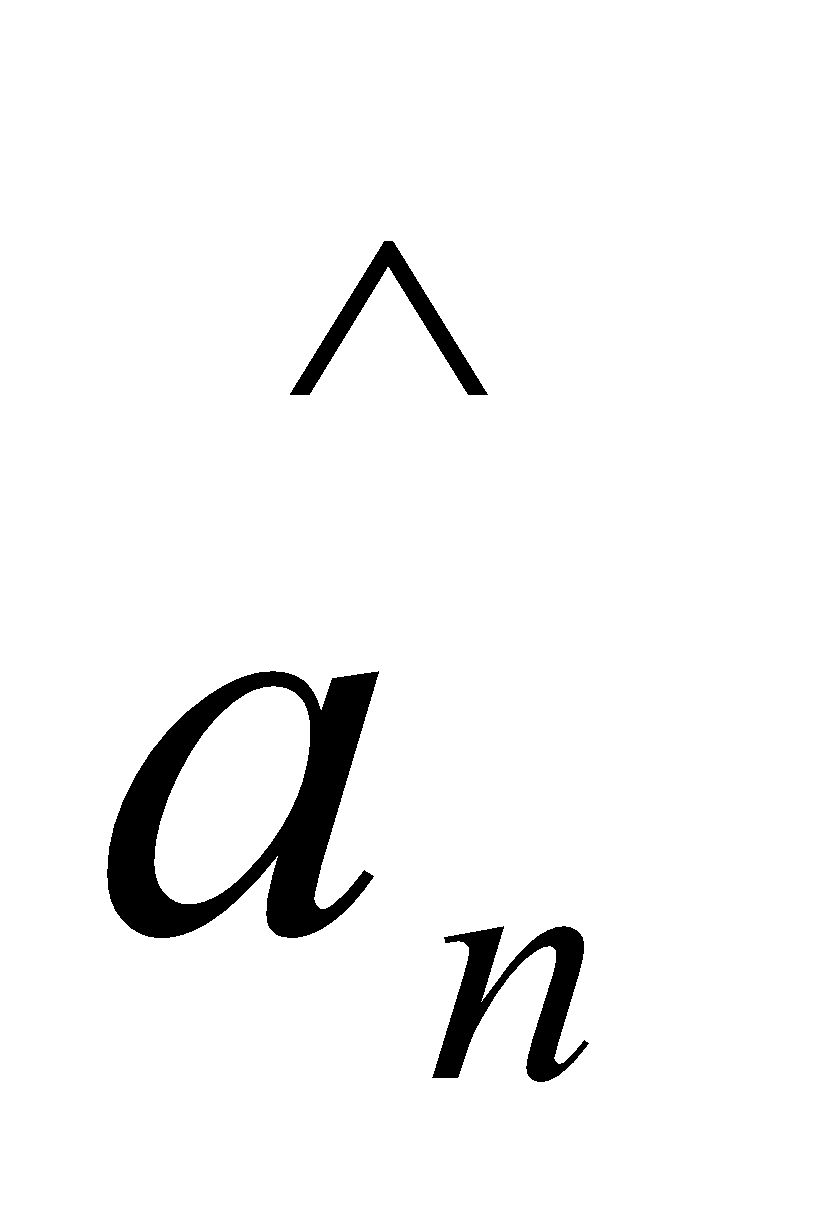
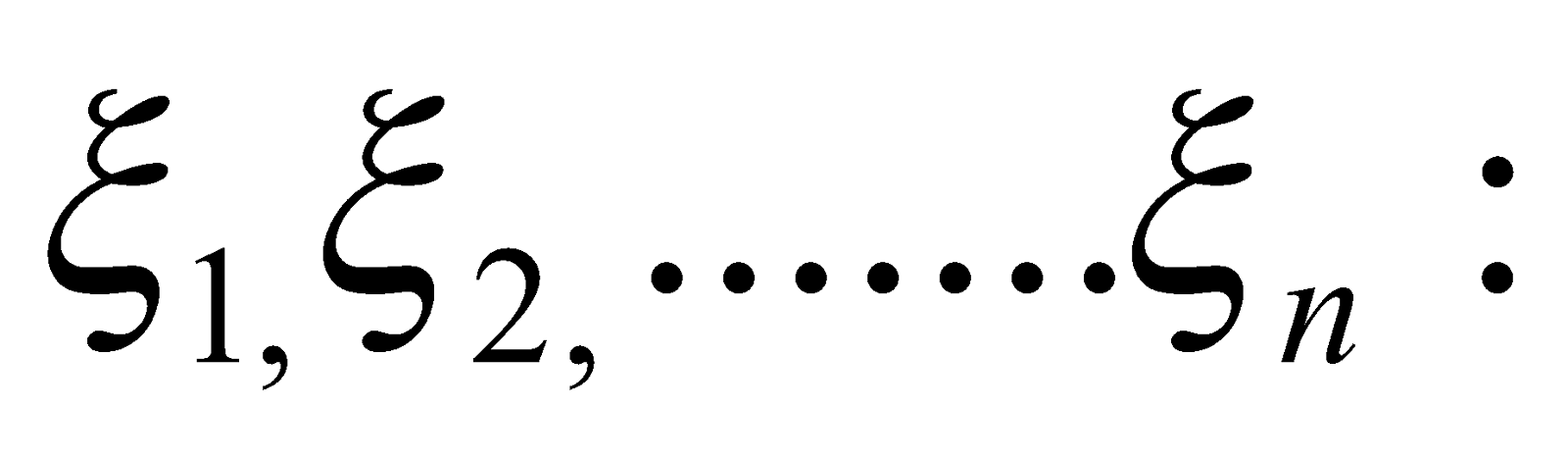
Заметим, что схема вычислений как многомерных , так и одномерных интегралов , абсолютно аналогична.

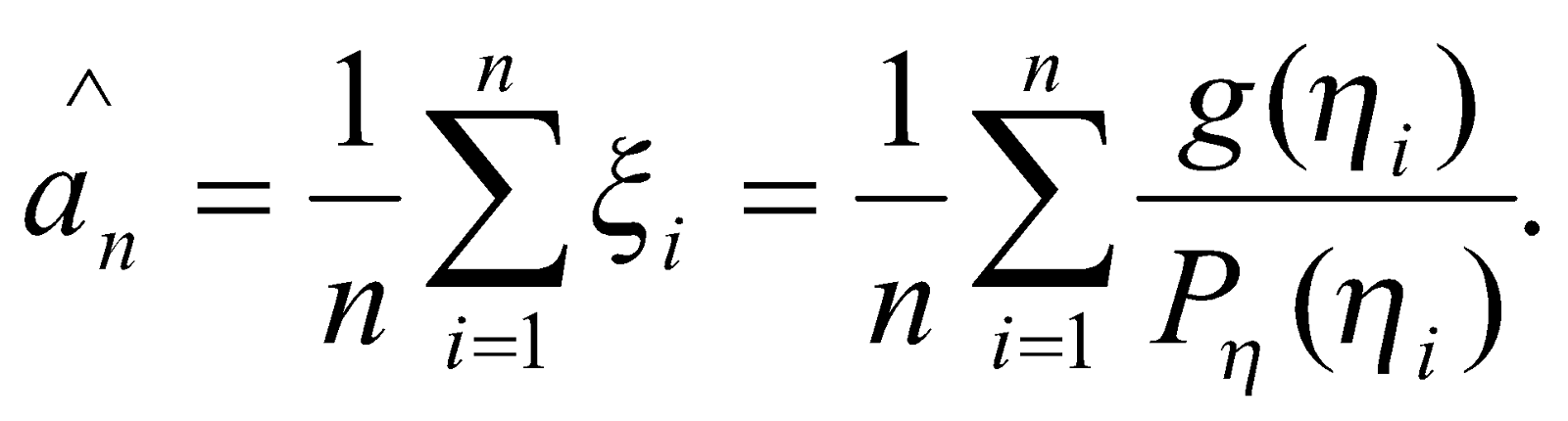
Пусть η – произвольная случайная величина с плотностью распределения вероятностей 

Предполагается только, что существуют моменты случайных величин, встречающиеся ниже. Рассмотрим случайную величину, являющуюся функциональным преобразованием случайной величины η.



В случае с этой задачей значение  принимает значение:

Можно показать, что  Поэтому в качестве приближенного значения интеграла можно использовать статистическую оценку , построенную в выборке из n независимых случайных величин 



**Код программы:**

import numpy as np

from scipy import stats

import math

import matplotlib.pyplot as plt

def f(x):

    return x\*x\*math.sin(x)\*math.sin(x)

def integral(n):

    rng = np.random.default\_rng()

    sum = 0.0

    for i in range(n):

        sum += f(rng.random()\*math.pi)

    return sum\*math.pi/n

print('integral: {}'.format(integral(1000)))

x\_array = np.arange(10, 1000)

y\_array = np.array([integral(x) for x in x\_array])

y\_array2 = np.array([abs(4.38231461665252 - integral(x)) for x in x\_array])

y\_array3 = np.array([integral(1000) for i in range(1000)])

plt.title("Integral")

plt.xlabel("Count of dots")

plt.ylabel("Integral value")

plt.plot(x\_array, y\_array)

plt.show()

plt.title("Integral")

plt.xlabel("Count of dots")

plt.ylabel("Difference")

plt.plot(x\_array, y\_array2)

plt.show()

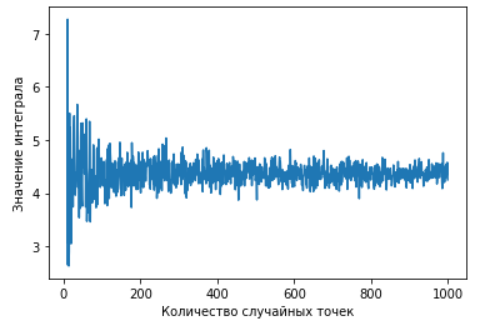
plt.hist(y\_array3,bins=20)

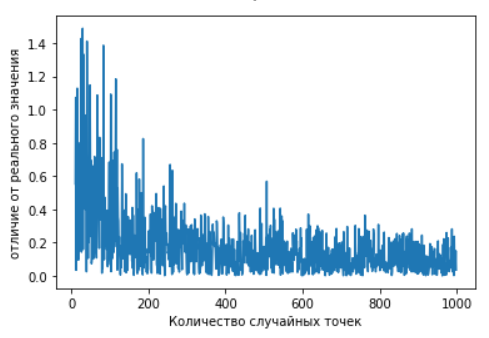
**Результат:**

при n = 1000, где n – количество выбираемых случайных точек

реальное значение интеграла = 4.38231461665252

График зависимости значения интеграла от количества случайных точек.





Гистограмма распределения значений ингерала при тысячи повторений.

