**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

Кафедра теории вероятностей и математической статистики

**Крагель Алины Олеговны**

**Моделирование непрерывной случайной величины**

Отчет по лабораторной работе №3

(«Имитационное и статистическое моделирование»)

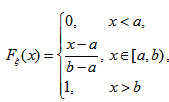
Студентки 4 курса 9 группы

**Преподаватель**

*Гайдук Антон Николаевич*

# Теоретическая часть

## Равномерное распределение

НСВ  имеет равномерное распределение на интервале [a, b), обозначаемое R(a, b), если функция и плотность распределения  определяются соотношениями:

Алгоритм моделирования СВ  основан на методе обратной функции. Обратная функция для находится при решении уравнения  относительно x:



Далее в соответствии с указанным методом алгоритм моделирования реализации СВ  включает два шага:

* Моделирование реализации  БСВ.
* Принятие решения о том, что реализацией  является величина :

## Нормальное распределение

НСВ  может иметь одномерное нормальное (гауссово) распределение с параметрами: средним значением  и дисперсией  (обозначается ). Функция распределения  имеет вид:

Распределение  называется стандартным нормальным распределением, а НСВ  ~  -стандартной нормальной (гауссовской) величиной.

Случайные величины  и  связаны соотношением:

 (1)

Алгоритм моделирования реализуется методом суммирования, основанном на центральной предельной теореме: если -независимые БСВ, то при  случайная величина

распределена асимптотически нормально, так что .

На практике приемлемая точность аппроксимации стандартной гауссовской СВ достигается при N = 12.

Таким образом, алгоритм моделирования ~ состоит из следующих шагов:

* Моделирование N=12 реализацией  БСВ.
* Принятие решения о том, что реализацией СВ  является величина x, равная

## Логнормальное распределение

НСВ  может иметь логарифмически-нормальное распределение (логнормальное распределение) с параметрами: средним значением  и дисперсией  (обозначается *L*. Функция распределения *L* имеет вид:

Очевидно, СВ  имеет распределение *L*, если СВ распределена по нормальному закону . Эта связь распределений лежит в основе моделирующего алгоритма.

Алгоритм моделирования ~*L*основан на методе функциональных преобразований и состоит из следующих шагов:

* Моделирование z реализации стандартной гауссовской СВ.
* Вычисление реализации x СВ ~ *L*по формуле.

## Экспоненциальное распределение

НСВ  с функцией распределения, определяемой соотношением:

имеет экспоненциальное распределение , где -параметр распределения.

Алгоритм моделирования СВ  основа на методе обратной функции. Обратная функция для  находится при решении уравнения относительно x:

Далее в соответствии с методом обратной функции алгоритм моделирования СВ состоит из двух шагов:

* Моделирование реализации БСВ a.
* Вычисление реализации x СВ : 

## Логистическое распределение

НСВ  может иметь имеет логистическое распределение  с параметрами  - среднее значение, (где  - стандартное отклонение СВ ). Тогда функция распределения закона  имеет вид:

Алгоритм моделирования основан на методе обратной функции. Обратная для  функция имеет вид:

(2)

Для моделирования реализации СВ выполняются следующие действия.

* Моделируется реализация  БСВ.
* Вычисляется значение  по формуле (2).

## Распределение Лапласа

НСВ  может иметь распределение Лапласа , где  параметр распределения. Функция распределения имеет вид:



Алгоритм моделирования основан на методе обратной функции. Обратная для функция имеет вид:

(3)

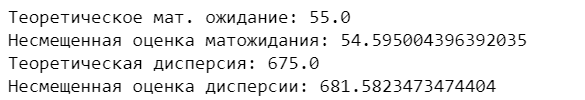
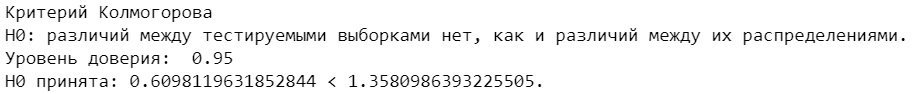
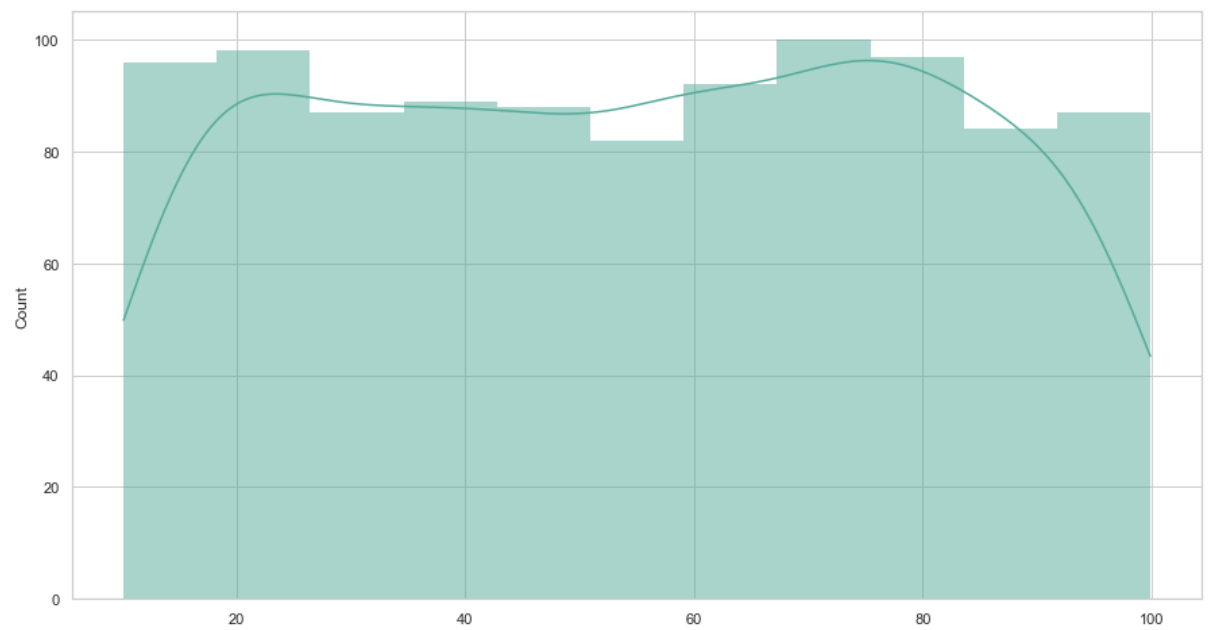
. (4)

Для моделирования реализации x СВ выполняются следующие действия:

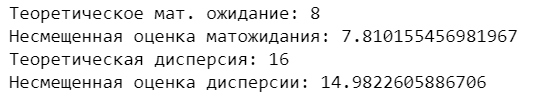
* Моделируется реализация y БСВ.
* Принимается решение о том, что реализацией СВ является величина x, вычисляемая по формуле (3), если y < 0.5; но формуле (4), если .

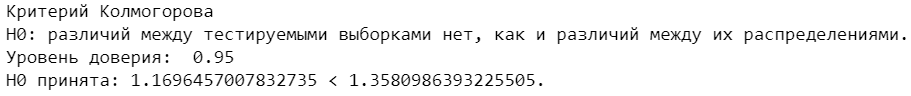
# Результаты эксперимента

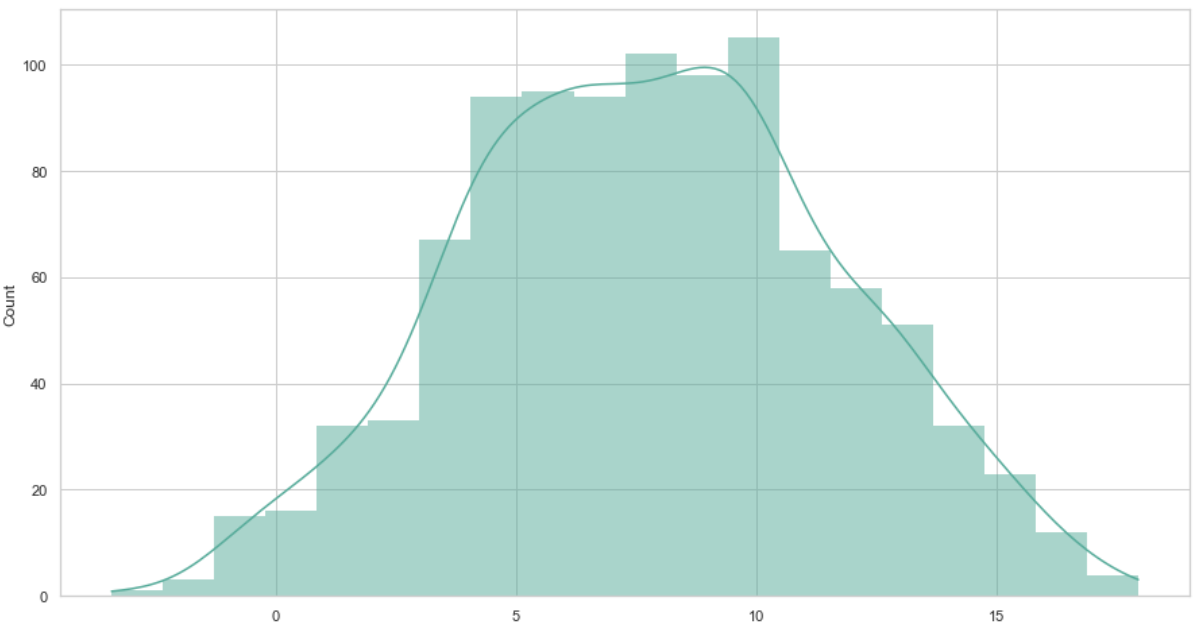
## Равномерное распределение

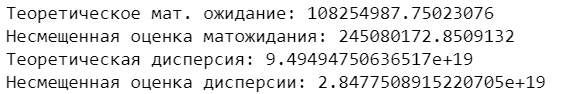
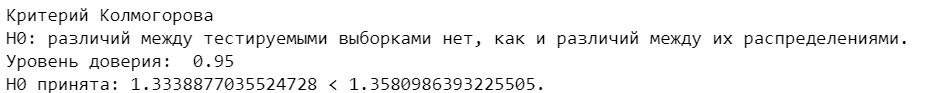
## Нормальное распределение

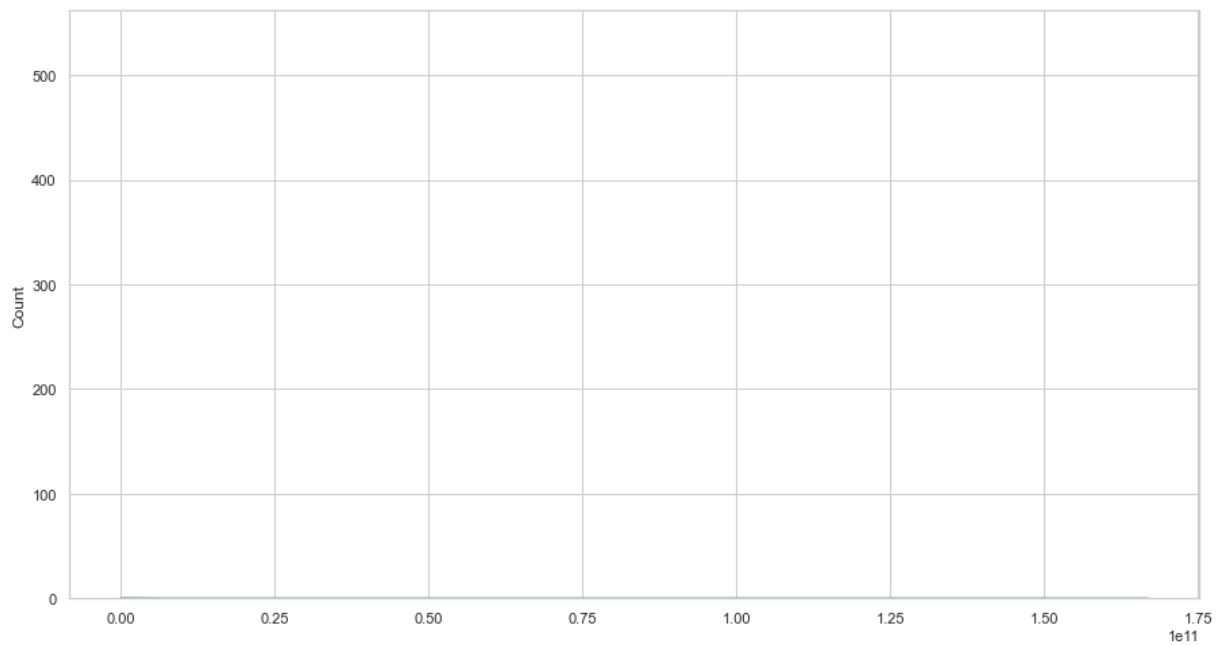




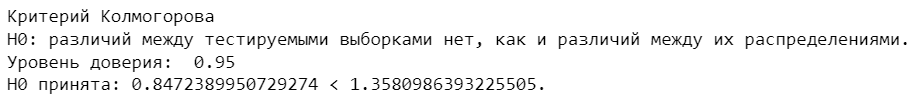


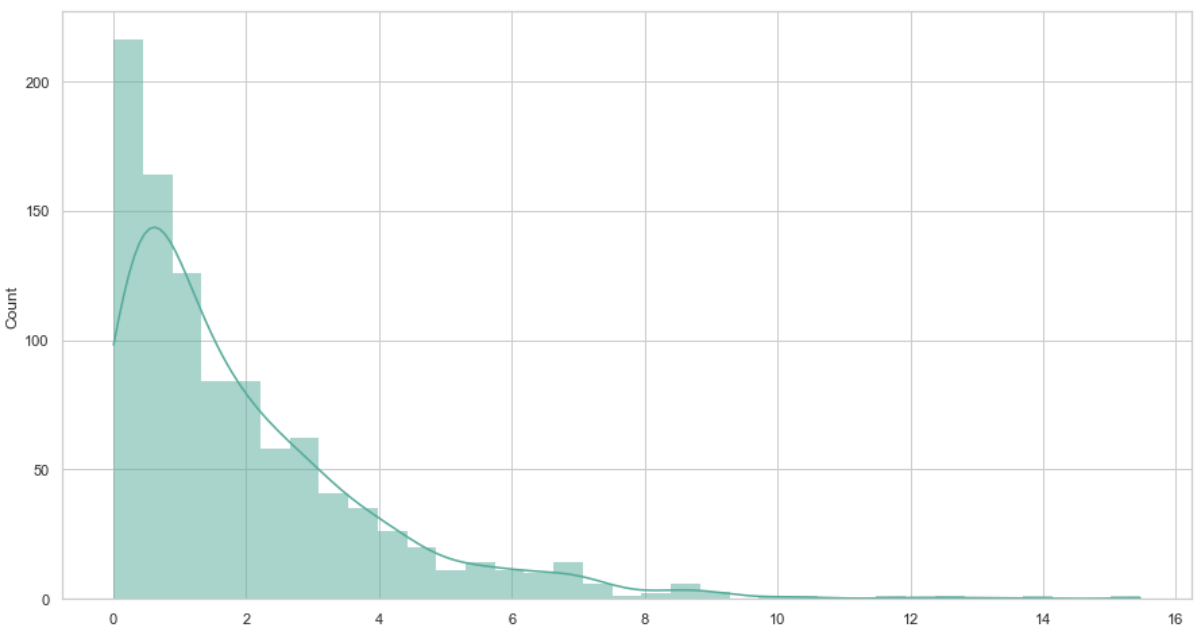
## Логнормальное распределение

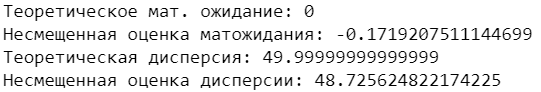


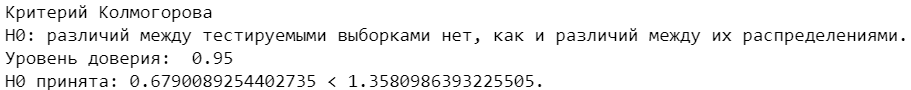
## Экспоненциальное распределение

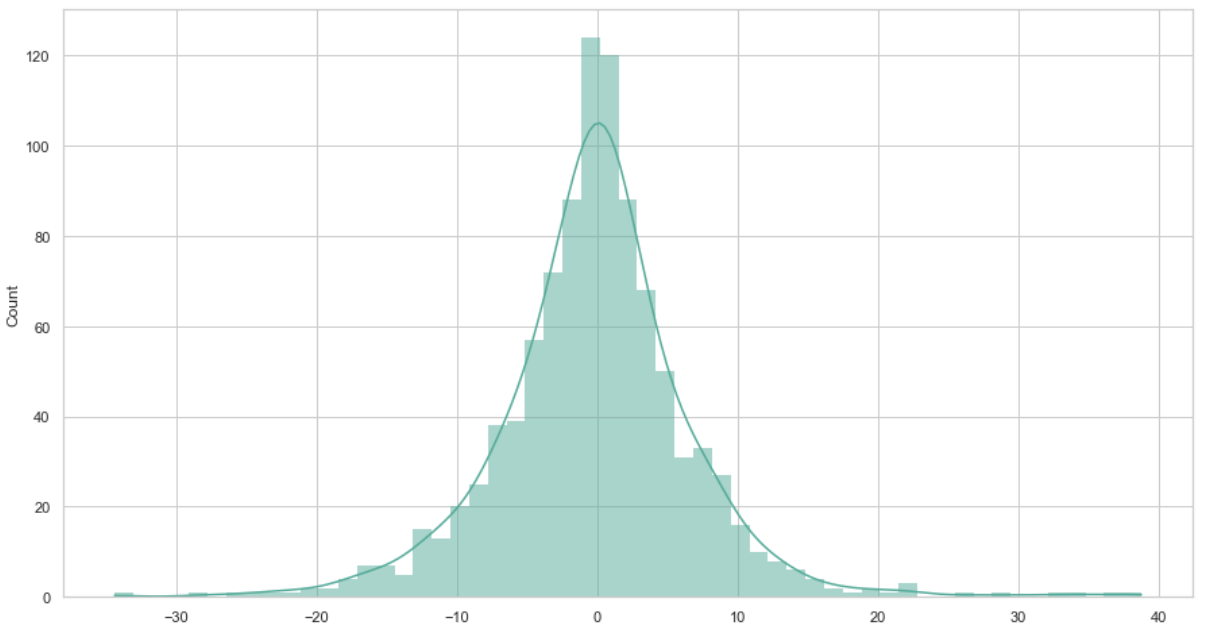




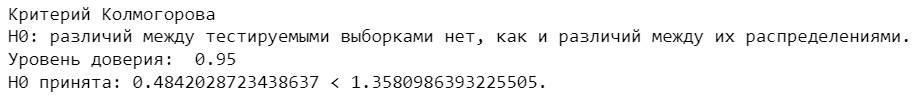
## Распределение Лапласа

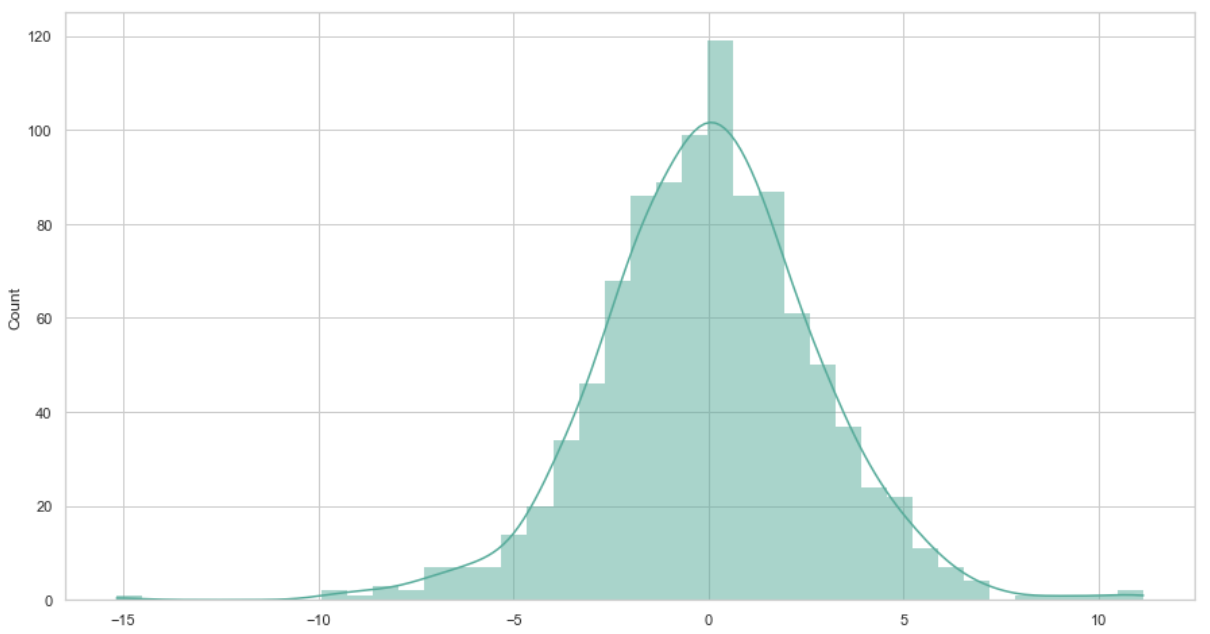






## Логистическое распределение





# Выводы

Результаты тестирования позволяют сделать вывод, что построенная с помощью генератора Маклорена-Марсальи на обновленных параметрах последовательность действительно могут служить непрерывной случайной величиной.

Были построены непрерывные случайные величины с равномерным, нормальным, логнормальным, экспоненциальным и логистическим законами распределения, а также с распределением Лапласа. Для них были подсчитаны статистики математического ожидания и дисперсии, построены гистограммы. На основе теста Колмогорова все последовательности относятся к тому или иному распределению.

*#!/usr/bin/env python  
# coding: utf-8  
# # S&SM  
# ## L3. Alina Kragel, gr. 9  
# In[1]:*import numpy as np  
import seaborn as sns  
from matplotlib import pyplot as plt  
import math  
from math import gcd  
from math import factorial as fac  
from scipy.stats import norm, chi2, kstwobign  
from scipy.special import erf  
  
  
*# In[2]:*sns.set(rc = {**'figure.figsize'**:(15,8)})  
sns.set\_theme(style=**'whitegrid'**, palette=**'dark:#5A9\_r'**)  
  
  
*# ##### Все новое - хорошо забытое старое  
# In[3]:*class LCG:  
 def \_\_init\_\_(self, x0, a, c, M):   
 self.a = a  
 self.c = c  
 self.M = M  
 self.x = x0  
   
 def \_\_call\_\_(self):  
 self.x = (self.x \* self.a + self.c) % self.M  
 return self.x / self.M  
   
 def get\_period(self):  
 s = list()  
 while True:  
 x = self()  
 if x in s:  
 break   
 s.append(x)  
 return len(s)  
  
  
*# In[4]:*class MMG:  
 def \_\_init\_\_(self, g1, g2, k):  
 assert k > 0  
 self.k = k  
 self.g1 = g1  
 self.g2 = g2  
 self.v = [g1() for \_ in range(k)]  
   
 def rand(self):  
 s = int(self.g2() \* self.k)  
 rx = self.v[s]  
 self.v[s] = self.g1()  
 return rx  
  
  
*# Более оптимальные параметры для генератора Макларена-Марсальи  
# In[5]:*MMG = MMG(LCG(2 \*\* 17, 473109257, 5732841, 2 \*\* 32), LCG(2 \*\* 9, 12425, 5813, 2 \*\* 16), 100)  
  
  
*# In[6]:*np.random.seed(42)  
  
  
*# Критерий Колмогорова  
# In[7]:*def ks\_test(samples, cdf, alpha=0.05, \*\*kwargs):  
 n = len(samples)  
 empirical\_cdf = np.arange(n) / n  
 theoretical\_cdf = np.array([cdf(x, \*\*kwargs) for x in sorted(samples)])  
 Dn = np.max(np.abs(theoretical\_cdf - empirical\_cdf))  
 ks\_value = np.sqrt(n) \* Dn  
  
 significance\_level = 1 - alpha  
 critical\_value = kstwobign.ppf(significance\_level)  
  
 print(  
 **f"Критерий Колмогорова"**,  
 **"**\n**H0: различий между тестируемыми выборками нет, как и различий между их распределениями."**,  
 **"**\n**Уровень доверия: "**, significance\_level  
 )  
 if ks\_value < critical\_value:  
 print(**f"H0 принята:** {ks\_value} **<** {critical\_value}**."**)  
 else:  
 print(**f"H0 не принята:** {ks\_value} **>=** {critical\_value}**."**)  
  
  
*# Генерация выборки на ГММ  
# In[8]:*def generate\_samples(generate\_sample, n=1000, \*\*kwargs):  
 return np.array([generate\_sample(\*\*kwargs) for \_ in range(n)])  
  
  
*# #### Равномерное распределение на ГММ и встроенном генераторе  
# In[9]:*def uniform\_sample(a=0, b=1):  
 x = MMG.rand()  
 return (b - a) \* x + a  
  
  
*# In[10]:*def uniform\_cdf(x, a=0, b=1):  
 return 0 if x < a else 1 if x > b else (x - a) / (b - a)  
  
  
*# In[11]:*a = 10  
b = 100  
  
  
*# In[12]:*uniform = generate\_samples(uniform\_sample, a=a, b=b)  
  
  
*# In[13]:*print(**"Теоретическое мат. ожидание:"**, (a + b) / 2)  
print(**"Несмещенная оценка матожидания:"**, uniform.mean())  
print(**"Теоретическая дисперсия:"**, (b - a) \*\* 2 / 12)  
print(**"Несмещенная оценка дисперсии:"**, uniform.var())  
  
  
*# In[14]:*ks\_test(uniform, uniform\_cdf, a=a, b=b)  
  
  
*# In[15]:*sns.histplot(uniform, kde=True, linewidth=0)  
  
  
*# #### Одномерное нормальное распределение на ГММ и встроенном генераторе  
# In[16]:*def normal\_sample(N=12, loc=0, scale=1):  
 sum = 0   
 for i in range(0, N):  
 sum += np.random.rand()  
 return loc + (12 / N) \*\* 0.5 \* (sum - N / 2) \* scale  
  
  
*# In[17]:*def normal\_cdf(x, loc=0, scale=1):  
 return 0.5 \* (1 + erf((x - loc) / (scale \* 2 \*\* 0.5)))  
  
  
*# In[18]:*N = 24  
m = 8  
s2 = 16  
s = s2 \*\* 0.5  
  
  
*# In[19]:*normal = generate\_samples(normal\_sample, loc=m, scale=s)  
  
  
*# In[20]:*print(**"Теоретическое мат. ожидание:"**, m)  
print(**"Несмещенная оценка матожидания:"**, normal.mean())  
print(**"Теоретическая дисперсия:"**, s2)  
print(**"Несмещенная оценка дисперсии:"**, normal.var())  
  
  
*# In[21]:*ks\_test(normal, normal\_cdf, loc=m, scale=s)  
  
  
*# In[22]:*sns.histplot(normal, kde=True, linewidth=0)  
  
  
*# #### Логнормальное распределение на ГММ и встроенном генераторе  
# In[23]:*def lognormal\_sample(N=12, loc=0, scale=1):  
 sum = 0  
 for i in range(0, 12):  
 sum += MMG.rand()  
 sum -= 6  
 return np.exp(loc + scale \* sum)  
  
  
*# In[24]:*def lognormal\_cdf(x, loc=0, scale=1):  
 return 0.5 \* (1 + erf((np.log(x) - loc) / (scale \* 2 \*\* 0.5)))  
  
  
*# In[25]:*N = 20  
m = 14  
s2 = 9  
s = s2 \*\* 0.5  
  
  
*# In[26]:*lognormal = generate\_samples(lognormal\_sample, loc=m, scale=s)  
  
  
*# In[27]:*print(**"Теоретическое мат. ожидание:"**, np.exp(m + s2 / 2))  
print(**"Несмещенная оценка матожидания:"**, lognormal.mean())  
print(**"Теоретическая дисперсия:"**, (np.exp(s2) - 1) \* np.exp(2 \* m + s2))  
print(**"Несмещенная оценка дисперсии:"**, lognormal.var())  
  
  
*# In[28]:*ks\_test(lognormal, lognormal\_cdf, loc=m, scale=s)  
  
  
*# In[29]:*sns.histplot(lognormal, kde=True, linewidth=0)  
  
  
*# #### Экспоненциальное распределение на ГММ и встроенном генераторе  
# In[30]:*def exponential\_sample(a=1):  
 return -np.log(MMG.rand()) / a  
  
  
*# In[31]:*def exponential\_cdf(x, a=1):  
 return 1 - np.exp(-a \* x)  
  
  
*# In[32]:*a = 0.5  
  
  
*# In[33]:*exponential = generate\_samples(exponential\_sample, a=a)  
  
  
*# In[34]:*print(**"Теоретическое мат. ожидание:"**, 1 / a)  
print(**"Несмещенная оценка матожидания:"**, exponential.mean())  
print(**"Теоретическая дисперсия:"**, 1 / a \*\* 2)  
print(**"Несмещенная оценка дисперсии:"**, exponential.var())  
  
  
*# In[35]:*ks\_test(exponential, exponential\_cdf, a=a)  
  
  
*# In[36]:*sns.histplot(exponential, kde=True, linewidth=0)  
  
  
*# #### Распределение Лапласа на ГММ и встроенном генераторе  
# In[37]:*def laplace\_sample(a):  
 n = MMG.rand()  
 if n < 0.5:  
 return math.log(2 \* n) / a  
 else:  
 return -math.log(2 \* (1 - n)) / a  
  
  
*# In[38]:*def laplace\_cdf(x, a):  
 if x < 0:  
 return 0.5 \* np.exp(a \* x);  
 else:  
 return 1 - 0.5 \* np.exp(-a \* x);  
  
  
*# In[39]:*a = 0.2  
mu = 0  
  
  
*# In[40]:*laplace = generate\_samples(laplace\_sample, a=a)  
  
  
*# In[41]:*print(**"Теоретическое мат. ожидание:"**, mu)  
print(**"Несмещенная оценка матожидания:"**, laplace.mean())  
print(**"Теоретическая дисперсия:"**, 2 / a \*\* 2)  
print(**"Несмещенная оценка дисперсии:"**, laplace.var())  
  
  
*# In[42]:*ks\_test(laplace, laplace\_cdf, a=a)  
  
  
*# In[43]:*sns.histplot(laplace, kde=True, linewidth=0)  
  
  
*# #### Логистическое распределение на ГММ и встроенном генераторе  
# In[44]:*def logistic\_sample(loc=0, scale=1):  
 x = MMG.rand()  
 return loc + scale \* np.log(x / (1 - x))  
  
  
*# In[45]:*def logistic\_cdf(x, loc=0, scale=1):  
 return 1 / (1 + np.exp(-(x - loc) / scale))  
  
  
*# In[46]:*mu = 0  
k = 1.5  
  
  
*# In[47]:*logistic = generate\_samples(logistic\_sample, loc=mu, scale=k)  
  
  
*# In[48]:*print(**"Теоретическое мат. ожидание:"**, mu)  
print(**"Несмещенная оценка матожидания:"**, logistic.mean())  
print(**"Теоретическая дисперсия:"**, np.pi \*\* 2 \* k \*\* 2 / 3)  
print(**"Несмещенная оценка дисперсии:"**, logistic.var())  
  
  
*# In[49]:*ks\_test(logistic, logistic\_cdf, loc=mu, scale=k)  
  
  
*# In[50]:*sns.histplot(logistic, kde=True, linewidth=0)