Санкт-Петербургский политехнический университет Петра Великого
Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе

Дисциплина: Теория вероятностей

Тема: Статистическая обработка случайных последовательностей. Идентификация законов распределения.

Выполнил студент гр. 3530901/10001 _____ Д.Л. Симоновский

Преподаватель _____ К.В. Никитин

“22” мая 2023 г.

Санкт-Петербург
2023

Оглавление

1. Статистическая обработка экспериментальных данных

1.1. Выборочная функция распределения

По исходным данным, находящимся в файле *Task_2.txt* построена функция распределения и гистограмма.

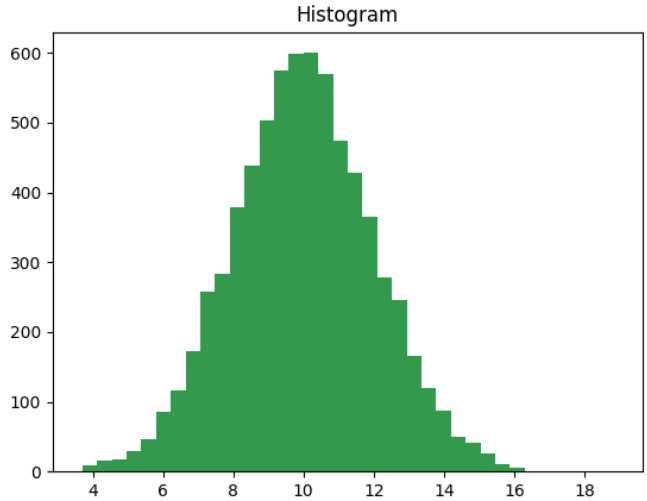


Рис. 1.1.1 Гистограмма распределения

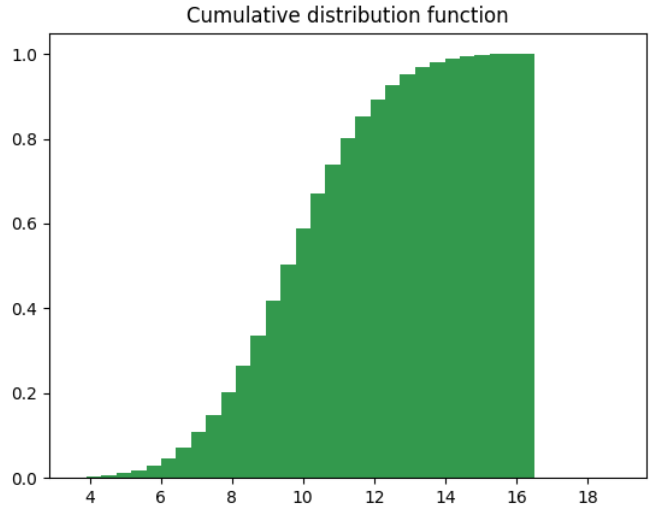


Рис. 1.1.2 Совокупная функция распределения

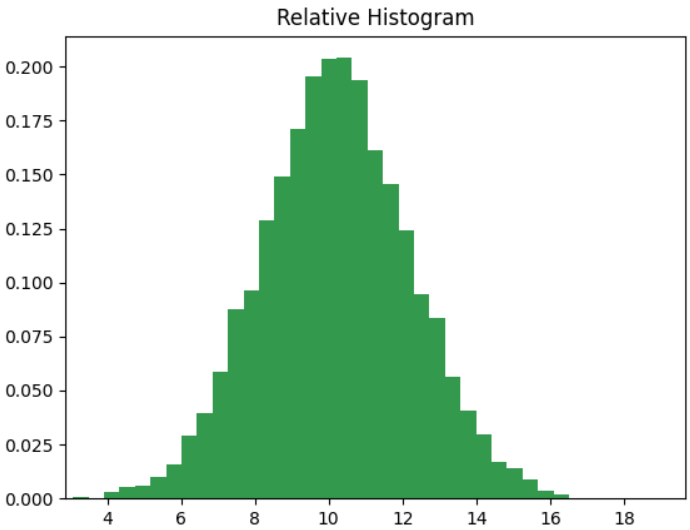


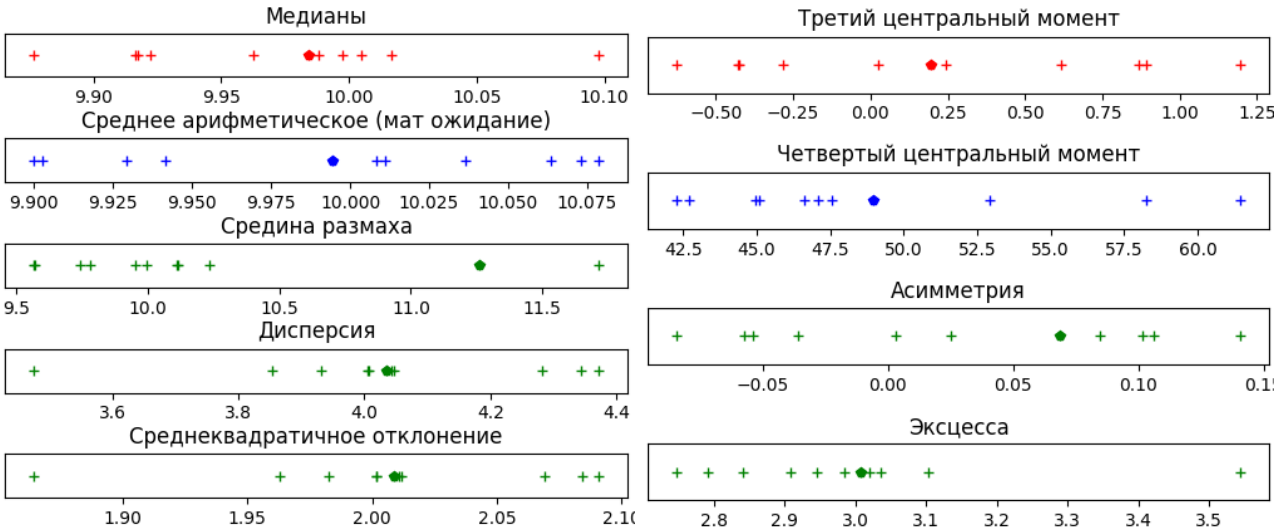
Рис. 1.1.3 Относительная гистограмма

Входные данные были перемешаны, после чего список был поделен на 10 равных частей. Далее по полной выборке и по подвыборкам были посчитаны точечные оценки. Результаты представлены в таблице.

	$\bar{x}$(M(X))	x_{med}	x_{cp}	Variance (s^2)	m_3	m_4	As	Ex
N	9.99442	9.98424	11.262585	4.035378	0.19583	48.96789	0.06878	3.007065
N / 10 (1)	10.0109	9.91712	9.745925	3.853424	0.10405	46.00749	0.01376	3.098382
N / 10 (2)	9.92934	9.87636	10.11656	4.005962	1.34916	44.52259	0.16827	2.774384
N / 10 (3)	10.03639	10.00475	9.997075	3.474467	0.09208	34.13239	0.01422	2.827419
N / 10 (4)	9.899728	9.99735	9.955115	4.344519	-0.56667	55.73525	-0.06258	2.952882
N / 10 (5)	10.06335	9.988235	9.565795	4.006424	-0.55283	47.45719	-0.06894	2.956571
N / 10 (6)	9.941617	9.96253	9.572335	4.047901	-0.63412	49.91667	-0.07786	3.046392
N / 10 (7)	9.902762	9.922195	11.7154	4.281758	0.672247	61.35895	0.075874	3.346829
N / 10 (8)	10.07878	10.01685	9.78214	4.043132	0.091777	50.79762	0.011289	3.107475
N / 10 (9)	10.07297	10.0979	10.109065	3.930781	0.1648001	43.90589	0.021146	2.841614
N / 10 (10)	10.00832	9.91637	10.23747	4.372656	1.4840816	54.96061	0.162308	2.874489

1.2. Определение точечных оценок

Границы интерквантильного промежутка для P = 0.95:
J (значения) = (8.6334, 11.3214)
По номерам точек
J (номера значений) = (1750, 5250)
Графики точечных оценок:



1.3. Интервальные оценки с доверительной вероятностью Q=0.8

Интервальный оценки мат. ожидания и дисперсии

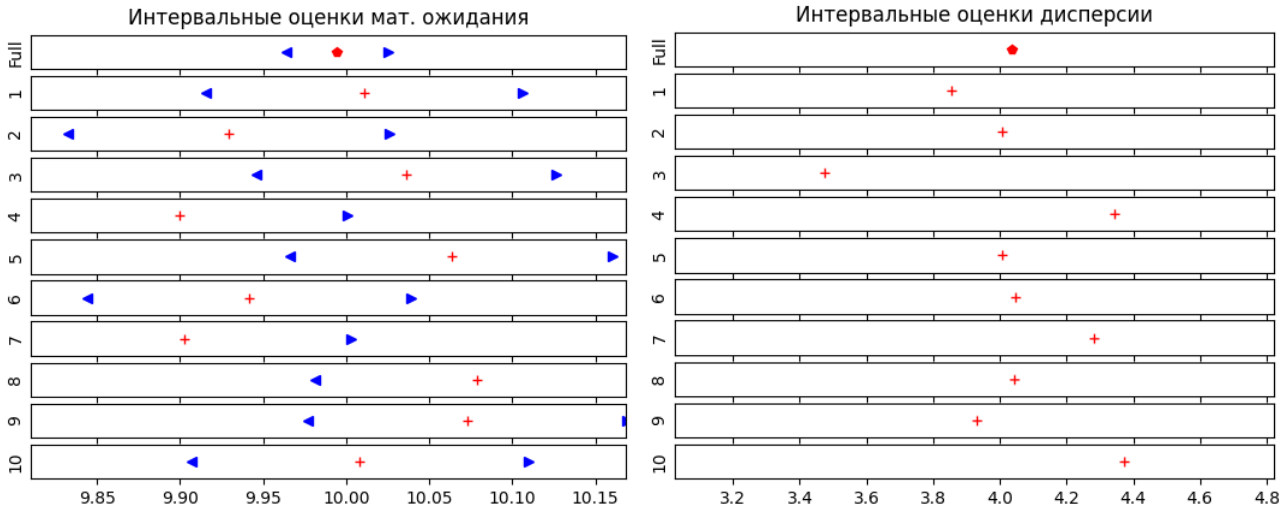
Значения функции распределения Хи-квадрат также посчитаны в MATLAB, с помощью функций:

```
chi2inv(0.9, 7000) = 7,1521e+03
chi2inv(0.1, 7000) = 9,8488e + 03
```

Выборка	Мат. ожидание		Дисперсия	
	Левая	Правая	Левая	Правая
Full	(9.963650239727205, 10.025192811701366),		(2.3154393156810924, 2.3933435289547558),	
1	(9.868696751748313, 10.075783905394545),		(0.26183615739056754, 0.27064577710720916),	
2	(9.84122947416036, 10.039465497268212),		(0.2399321546721422, 0.24800480232142424),	
3	(9.921107949649294, 10.118802050350705),		(0.238622131569758, 0.2466507027802967),	
4	(9.909155194554533, 10.10187611973118),		(0.2267676214259632, 0.23439734120461916),	
5	(9.869356830490045, 10.058353598081386),		(0.21808814274533525, 0.22542583675002226),	
6	(9.94566951846008, 10.139026624397063),		(0.22826723347647626, 0.2359474085169997),	
7	(9.869703003552459, 10.058748996447541),		(0.21820176218191778, 0.22554327897425447),	
8	(9.867124201607371, 10.06425574124977),		(0.23726601096067917, 0.24524895475682656),	
9	(10.019300992921883, 10.207216264220973),		(0.21559935221707807, 0.22285330951279855),	
10	(9.859852024158458, 10.053705775841541)		(0.22944136716111246, 0.2371610465671186)	

Табл. 1.3.1 Таблица интервальных оценок

Графики интервальных оценок мат. ожидания и дисперсии:



1.4. Интервальные оценки интерквантильного промежутка для P = 0.95

Непараметрические толеранные пределы для всей выборки симметричные относительно среднего значения. Кол-во отбрасываемых точек k было найдено с помощью биномиального распределения:

$$k = 366$$

Толерантные пределы всей выборки симметричные относительно среднего значения:

[6.10691, 13.9387]

График

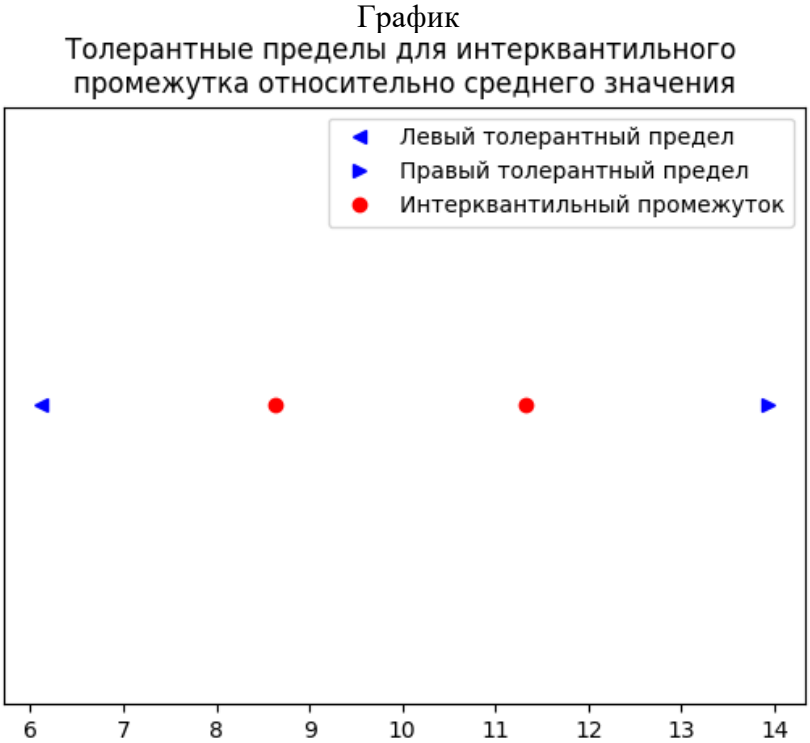


Рис. 1.4.1. Толерантные пределы для интерквантильного промежутка относительно среднего значения

Толерантные пределы всей выборки симметричные относительно нуля

[-13.29, 13.29]

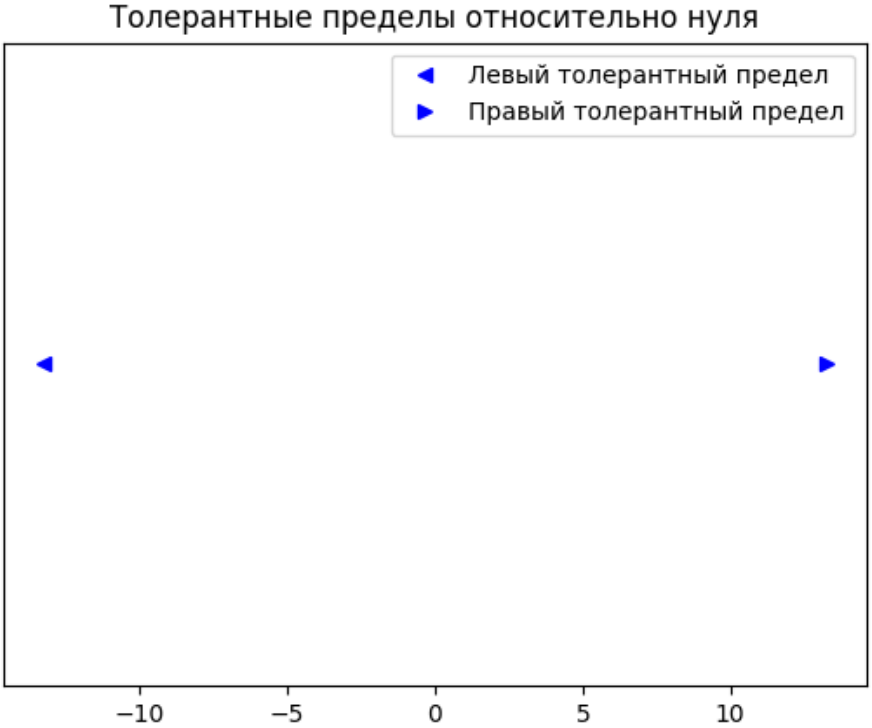
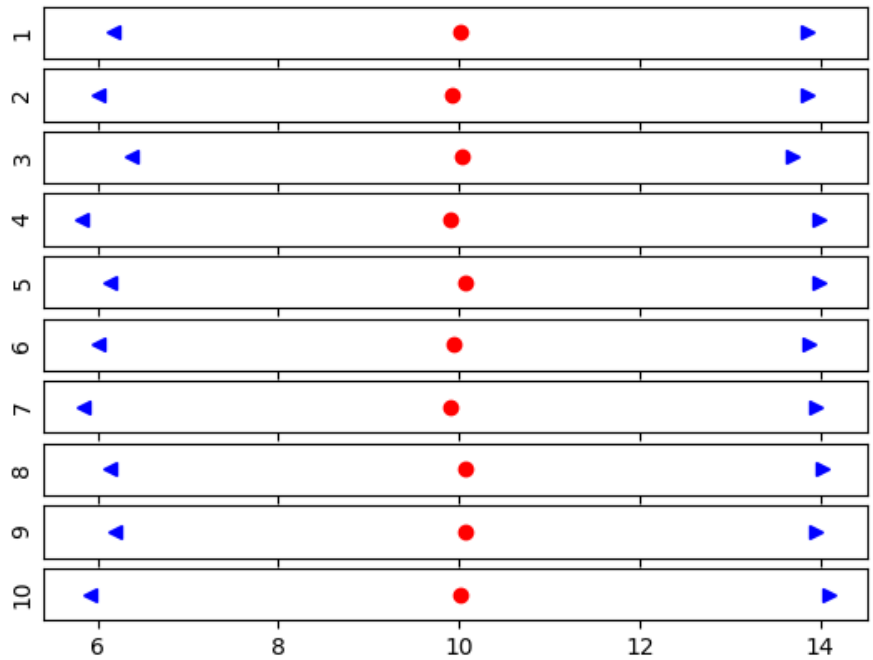


Рис. 1.4.2. Толерантные пределы относительно нуля

Параметрические толерантные пределы для подвыборок:
Таблица результатов:

Номер подвыборки	(Левый, Правый)
1	[5.782609506951021, 14.161871150191837]
2	[5.929786049407495, 13.950908922021076]
3	[6.020357328354483, 14.019552671645517]
4	[6.106531515081526, 13.904499799204187]
5	[6.140215414590589, 13.787495013980841]
6	[6.130493200909731, 13.954202941947413]
7	[6.139590311153239, 13.788861688846762]
8	[5.977473610069071, 13.95390633278807]
9	[6.311498844703993, 13.915018412438862]
10	[6.034876268001687, 13.878681531998312]

Табл. 1.4.1. Параметрически пределы для подвыборок



Красные точки на графиках – это математические ожидания подвыборок.
Как видно из графика толерантного предела интерквантильного промежутка всей выборки толерантные пределы шире, чем интерквантильный промежуток.
А также на всех графиках мат. ожидания лежат посередине толерантного отрезка за исключением толерантного отрезка симметричного относительно нуля.

2. Идентификация закона

2.1. Начальный выбор распределения

В качестве распределений-кандидатов (учитывая точечные показатели и форму гистограммы) были выбраны следующие: Нормальное, Гамма, Лапласа.

2.2. Определение параметров теоретических распределений

```
Для нормального распредления
    c = 9.994421525714285 s = 2.0088252926371784
Для распределения Лапласа
    a = 9.994421525714285 lambda = 0.7040002769561513
Для Гамма-распредления
    k = 24.753179376483732 lambda = 0.4037631438654417
```

Рис. 2.2.1 Метод моментов

```
Для нормального распределения
    c = 9.994421525714262 s = 2.0086817999913538
Для распределения Лапласа
    a = 9.994421525714285 lambda = 0.6261052429673818
Для Гамма-распределения
    k = 23.37606306225524 theta = 0.4275493909773032
```

Рис. 2.2.2. Метод максимального правдоподобия

Для минимизации была написана функция, использующая метод бисекции. Точки, при которых функция принимала положительное и отрицательное значения подбирались вручную.

Графики сравнения результатов:

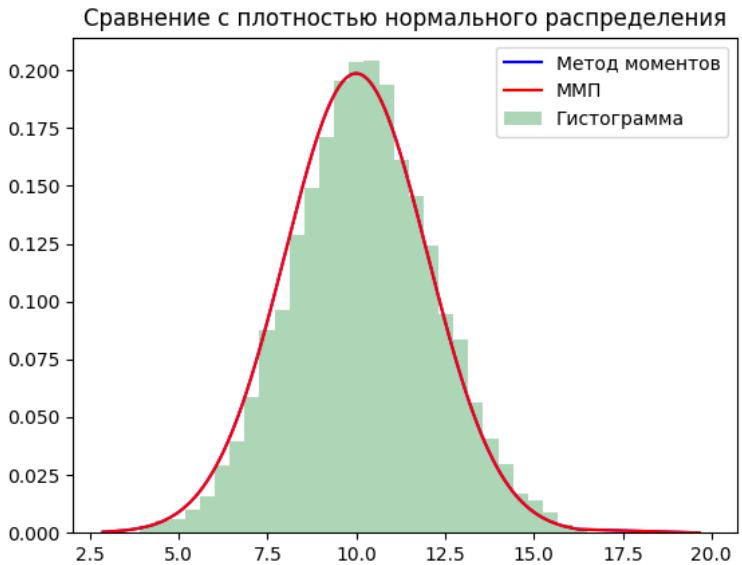


Рис. 2.2.3. Сравнение с плотностью нормального распределения.

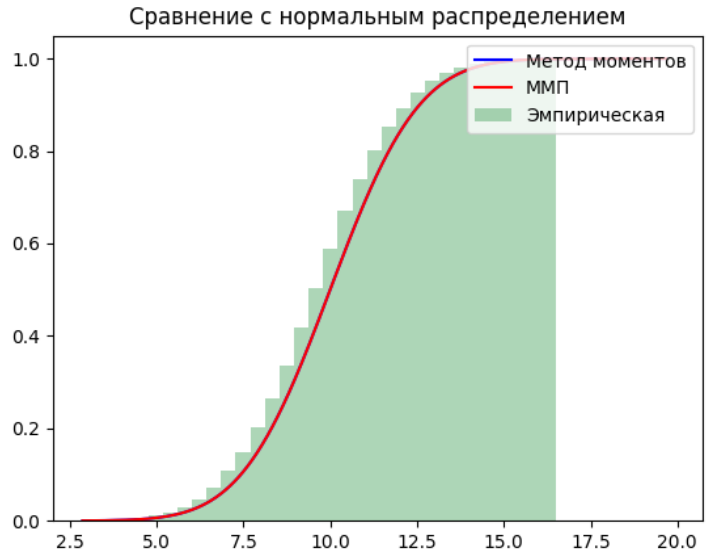


Рис. 2.2.4. Сравнение с нормальным распределением.

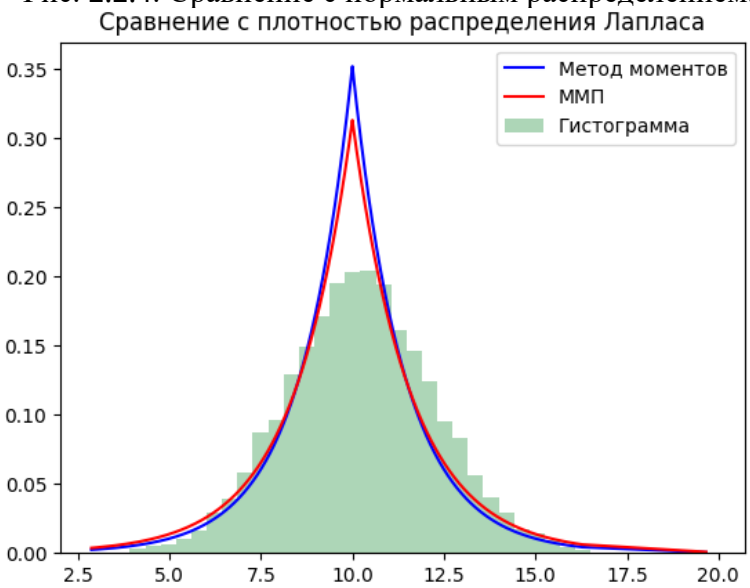


Рис. 2.2.5. Сравнение с плотностью распределения Лапалса.

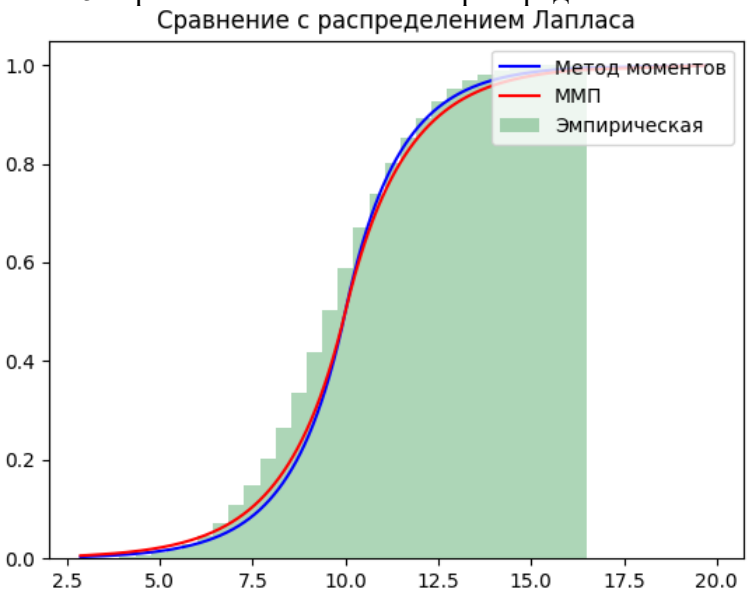


Рис. 2.2.6. Сравнение с распределнием Лапласа

Рис. 2.2.7. Сравнение с плотностью Гамма-распределения

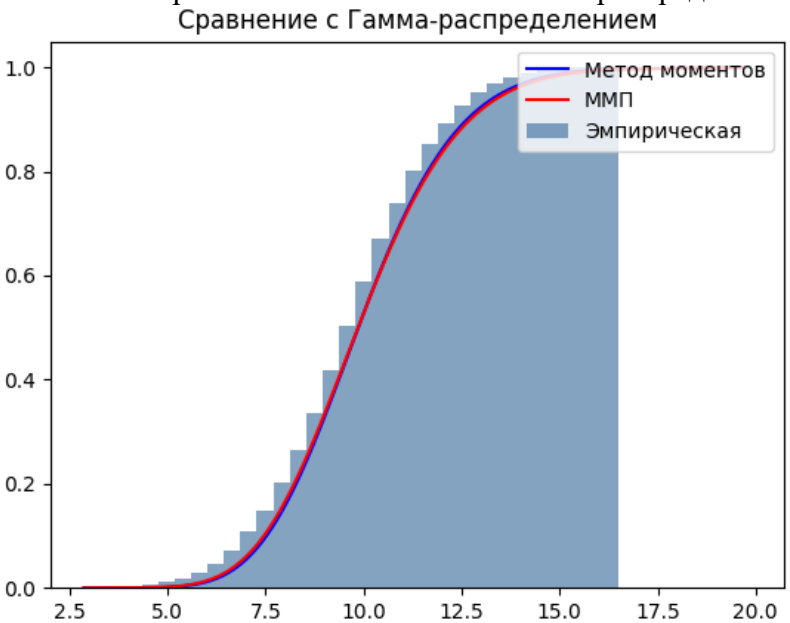


Рис. 2.2.8. Сравнение с Гамма-распределением

На некоторых графиках не видно кривой полученной методом моментов. Это связано с тем, что метод моментов и ММП дали очень близкие значения параметров и одна кривая находится под другой.

2.3. Проверка гипотез

Название	Норм. распределение		Гамма-распределение		Распределение Лапласа	
	Метод моментов	ММП	Метод моментов	ММП	Метод моментов	ММП
Хи-квадрат статистика	59.0763	58.6628	568.968	476.8604	465.9427	3158.577
Хи-квадрат критич. знач.	46.1730					
Хи-квадрат вывод	Близко	Близко	Нет	Нет	Нет	Нет
Колм. - Смирн. статистика	0.008907	0.008896	0.026762	0.031152	0.0598538	0.040475
Колм. - Смирн. критич. знач.		0.0162086				
Колм. - Смирн. вывод	Да	Да	Нет	Нет	Нет	Нет
Мизеса - статистика	0.070566	0.070381	1.70912	2.03717	11.4671	4.69408
Мизеса критич. знач.	0.2415					
Мизеса вывод	Да	Да	Нет	Нет	Нет	Нет

3. Вывод

В ходе данной лабораторной работы был осуществлен анализ выборки случайных величин с вычислением точечных и интервальных оценок. С использованием этих результатов и гистограмм были сформулированы гипотезы о различных распределениях. Затем были применены различные критерии для проверки данных гипотез.

После анализа результатов можно с уверенностью утверждать, что только гипотеза о нормальном распределении была подтверждена двумя критериями из трех. В то время как другие гипотезы не прошли ни одной проверки. Это позволяет сделать вывод о том, что исходная выборка действительно имеет нормальное распределение.

4. Листинг

```
import math
import matplotlib.pyplot as plot
import statistics as stat
import scipy.stats as stats
import numpy as np
import random
import help
from scipy.stats.distributions import chi2

# https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html


# ================================== Подготовка данных(down) ==============================

relay_SDVIG = 0

f = open("Task_2a.txt" 'r')
line = f.readline().split(" ")
data = []
data2 = [[] [] [] [] [] [] [] [] [] []]
numOfPoints = int(line[2])
numOfPointsInOneUnderArray = numOfPoints / 10
for t in f.readline().split(' '):
    if t.replace("." "" 1).isdigit() or (t.startswith("-") and t[1:].replace("." "" 1).isdigit()):
        data.append(float(t))
# создание 10 подвыборок
res = 0
random.shuffle(data)
for i in range(numOfPoints):
    j = int(i // numOfPointsInOneUnderArray)
    res += data[i]
    data2[j].append(data[i])

# сортировка значений
list.sort(data)
for l in data2:
    list.sort(l)
# ==========================  функция распределения и гистограммы ====================
m = 40  # кол-во интервалов
numbers = []
for number in data:
    if number < 200:
        numbers.append(number)
#
for i in data:
    print(i)
min_value = min(numbers)  # минимальное значение в выборке
max_value = max(numbers)  # максимальное значение в выборке
distribution_fun = np.zeros(m)
print(max_value)
h = (max_value + 0.00000000001 * max_value - min_value) / m  # шаг с которым идут интервалы
steps = []  # массив точек с шагом h
for t in range(1 m+1):
    steps.append(t * h + min_value)

index = 0
print(numbers)
for value in numbers:
    if value > steps[index]:
        p = int(abs(steps[index] - value) / h) + 1
        for i in range(1 p):
            distribution_fun[index + i] = distribution_fun[index]
        distribution_fun[index] = distribution_fun[index - 1]
        index += p
    distribution_fun[index - 1] += 1
print(distribution_fun)
plot.title("Cumulative distribution function")
plot.xlim([min(numbers) max(numbers)])  # CHANGE
plot.bar(steps distribution_fun / len(numbers) h color=(0.2 0.6 0.3 1.0))
plot.show()
plot.close()
plot.title("Histogram")
plot.xlim([min(numbers) max(numbers)])  # CHANGE
```

```
plot.hist(numbers steps color=(0.2 0.6 0.3 1.0))
plot.show()
plot.close()
# !!!!!!!!!Для относительной гистограммы
index = 0
for_relative = np.zeros(m)
steps = []  # массив точек с шагом h
for t in range(1 m+1):
    steps.append(min_value+t * h)
for value in numbers:
    if value > steps[index]:
        p = int(abs(steps[index] - value) // h) + 1
        for_relative[index] = for_relative[index] / (h * len(numbers))
        index += p
    for_relative[index] += 1
for_relative[m - 1] = for_relative[m - 1] / (h * len(numbers))


# Проверка площади под гистограммой
ssss_____ = 0
for v in for_relative:
    ssss_____ += v * h
print('Area under an histogram : ' str(ssss_____))
# Конец проверки площади



plot.bar(steps for_relative width=h color=(0.2 0.6 0.3 1.0))
plot.title("Relative Histogram")
plot.xlim([min(numbers) max(numbers)])  # CHANGE
plot.show()
plot.close()
plot.bar(steps for_relative width=h)
plot.title("Относительная гистограмма (до 22)")
plot.xlim([min(numbers) max(numbers)])  # CHANGE
plot.show()
plot.close()
# !!!!!!!!!!!!!!Относительная гистограмма построена

max_value = max(data)
# ================== ТОЧЕЧНЫЕ ОЦЕНКИ ========================
print("================== ТОЧЕЧНЫЕ ОЦЕНКИ ==========================")
empty = np.zeros(11)
median = [stat.median(data)]  # медианы
mean = [stat.mean(data)]  # среднее арифметическое (мат. ожидание)
mid_range = [(min_value + max_value) / 2]  # средина размаха
dispersion = [help.dispersion(data mean[0])]  # дисперсия s^2
root_of_dispersion = [math.sqrt(dispersion[0])]  # корень из дисперсии s
third_central_moment = [help.central_moment(data 3 mean[0])]  # 3-ий центральный момент
fourth_central_moment = [help.central_moment(data 4 mean[0])]  # 4-ый центральный момент
asymmetry = [help.asymmetry(third_central_moment[0] root_of_dispersion[0])]  # асимметрия
kurtosis = [help.kurtosis(fourth_central_moment[0] dispersion[0])]  # эксцесса

interquantile_interval = help.interquantile_interval(numOfPoints 0.5)  # интерквантильный интервал

index = 1
for n in data2:
    median.append(stat.median(n))
    mean.append(stat.mean(n))
    mid_range.append((min(n) + max(n)) / 2)
    dispersion.append(help.dispersion(n mean[index]))
    root_of_dispersion.append((math.sqrt(dispersion[index])))
    third_central_moment.append(help.central_moment(n 3 mean[index]))
    fourth_central_moment.append(help.central_moment(n 4 mean[index]))
    asymmetry.append(third_central_moment[index] / pow(root_of_dispersion[index] 3))
    kurtosis.append(help.kurtosis(fourth_central_moment[index] dispersion[index]))
    index += 1
print('\tMin: ' min_value ' Max: ' max_value)
print('\tx_med :' median)
print('\tM[x] :' mean)
print('\tx_cp :' mid_range)
print('\ts^2 :' dispersion)
print('\ts :' root_of_dispersion)
```

```
print('\t◦μ_3 :' third_central_moment)
print('\t◦μ_4 :' fourth_central_moment)
print('\tAs :' asymmetry)
print('\tEx :' kurtosis)
print('\tJ (номера значений) :' interquantile_interval)
print('\tJ (значения) :'
      "(" + str(data[interquantile_interval[0]]) + " " + str(data[interquantile_interval[1] - 1]) + ")")

# ==================== ГРАФИКИ ТОЧЕЧНЫХ ПОКАЗАТЕЛЕЙ ========================
plot.figure()

ax1 = plot.subplot(9 1 1)
ax1.set_ylim(-0.1 0.1)
ax1.set_yticks([])
ax1.set_yticklabels([])
plot.title('Медианы')
plot.plot(median empty 'r+')
plot.plot(median[0] 0 'rp')

ax2 = plot.subplot(9 1 3)
ax2.set_yticklabels([])
ax2.set_yticks([])
plot.title('Среднее арифметическое (мат ожидание)')
plot.plot(mean empty 'b+')
plot.plot(mean[0] 0 'bp')

ax3 = plot.subplot(9 1 5)
ax3.set_yticks([])
ax3.set_yticklabels([])
plot.title('Средина размаха')
plot.plot(mid_range empty 'g+')
plot.plot(mid_range[0] 0 'gp')

ax4 = plot.subplot(9 1 7)
ax4.set_yticks([])
ax4.set_yticklabels([])
plot.title('Дисперсия')
plot.plot(dispersion empty 'g+')
plot.plot(dispersion[0] 0 'gp')

ax5 = plot.subplot(9 1 9)
ax5.set_yticks([])
ax5.set_yticklabels([])
plot.title('Среднеквадратичное отклонение')
plot.plot(root_of_dispersion empty 'g+')
plot.plot(root_of_dispersion[0] 0 'gp')

plot.show()
plot.close()

plot.figure()
ax1 = plot.subplot(7 1 1)
ax1.set_ylim(-0.1 0.1)
ax1.set_yticks([])
ax1.set_yticklabels([])
plot.title('Третий центральный момент')
plot.plot(third_central_moment empty 'r+')
plot.plot(third_central_moment[0] 0 'rp')

ax2 = plot.subplot(7 1 3)
ax2.set_yticklabels([])
ax2.set_yticks([])
plot.title('Четвертый центральный момент')
plot.plot(fourth_central_moment empty 'b+')
plot.plot(fourth_central_moment[0] 0 'bp')

ax3 = plot.subplot(7 1 5)
ax3.set_yticks([])
ax3.set_yticklabels([])
plot.title('Асимметрия')
plot.plot(asymmetry empty 'g+')
plot.plot(asymmetry[0] 0 'gp')
```

```
ax4 = plot.subplot(7 1 7)
ax4.set_yticks([])
ax4.set_yticklabels([])
plot.title('Эксцесса')
plot.plot(kurtosis empty 'g+')
plot.plot(kurtosis[0] 0 'gp')
plot.show()
plot.close()
# ==================== ГРАФИКИ ТОЧЕЧНЫХ ПОКАЗАТЕЛЕЙ НАЧЕРЧЕНЫ =================


# ======================!!! Часть 1.4 . Интервальные оценки !!!==================
print("======================!!! Часть 1.4 . Интервальные оценки !!!===============")
Q = 0.8  # доверительная вероятность

left_chi2inv = chi2.ppf((1 + Q) / 2 df=11999)
right_chi2inv = chi2.ppf((1 - Q) / 2 df=11999)
tinv = 1.2816  # посчитано в MATLAB функцией tinv(0.9 n-1) 0.9 = (1+q)/2 где q=0.8 CHANGE
mean_interval = [help.mean_interval(numOfPoints mean[0] root_of_dispersion[0] tinv)]
dispersion_interval = [help.dispersion_interval(numOfPoints dispersion[0] left_chi2inv right_chi2inv)]

for i in range(1 11):
    mean_interval.append(help.mean_interval(numOfPointsInOneUnderArray mean[i] root_of_dispersion[i]
tinv))
    dispersion_interval.append(help.dispersion_interval(numOfPointsInOneUnderArray dispersion[i]
left_chi2inv right_chi2inv))
print("\t Интервальные оценки для мат. ожидания" + str(mean_interval))
print("\t Интервальные оценки для дисперсии" + str(dispersion_interval))
# ================== Чертим ИНТЕРВАЛЬНЫЕ ОЦЕНКИ МАТ ОЖИДАНИЯ М ДИСПЕРСИИ
==================================
# Для мат. ожидания
plot.figure()
axes = [plot.subplot(11 1 1)]
axes[0].set_yticks([])
axes[0].set_ylabel('Full')
plot.title('Интервальные оценки мат. ожидания')
plot.setp(axes[0].get_xticklabels() visible=False)
plot.plot(mean[0] 0 'rp')
plot.plot(mean_interval[0][0] 0 'b<')
plot.plot(mean_interval[0][1] 0 'b>')

for i in range(1 11):
    axes.append(plot.subplot(11 1 i + 1 sharex=axes[0]))
    axes[i].set_yticks([])
    axes[i].set_ylabel(str(i))
    if i < 10: plot.setp(axes[i].get_xticklabels() visible=False)
    plot.plot(mean[i] 0 'r+')
    plot.plot(mean_interval[i][0] 0 'b<')
    plot.plot(mean_interval[i][1] 0 'b>')
mat_razmach = max(mean) - min(mean)
axes[0].set_xlim([min(mean) - 0.5 * mat_razmach max(mean) + 0.5 * mat_razmach])  # CHANGE
plot.show()
plot.close()

# Для дисперсии

plot.figure()
axes = [plot.subplot(11 1 1)]
axes[0].set_yticks([])
axes[0].set_ylabel('Full')
plot.title('Интервальные оценки дисперсии')
plot.setp(axes[0].get_xticklabels() visible=False)
plot.plot(dispersion[0] 0 'rp')
plot.plot(dispersion_interval[0][0] 0 'b<')
plot.plot(dispersion_interval[0][1] 0 'b>')

for i in range(1 11):
    axes.append(plot.subplot(11 1 i + 1 sharex=axes[0]))
    axes[i].set_yticks([])
    axes[i].set_ylabel(str(i))
    if i < 10: plot.setp(axes[i].get_xticklabels() visible=False)
    plot.plot(dispersion[i] 0 'r+')
    plot.plot(dispersion_interval[i][0] 0 'b<')
```

```
    plot.plot(dispersion_interval[i][1] 0 'b>')
disp_razmach = max(dispersion) - min(dispersion)
axes[0].set_xlim([min(dispersion) - 0.5 * disp_razmach max(dispersion) +0.5 * disp_razmach])
plot.show()
plot.close()
# ================== графики ИНТЕРВАЛЬНЫХ ОЦЕНКИ МАТ ОЖИДАНИЯ М ДИСПЕРСИИ напечатаны!
==========================


# ============================ ТОЛЕРАНТНЫЕ ПРЕДЕЛЫ ================================
print("============================ ТОЛЕРАНТНЫЕ ПРЕДЕЛЫ ================================")
p = 0.95  # вероятность для интерквантильного промежутка
q = 0.8  # доверительная вероятность
tolerant_interval_average = [0 0]  # массив для толерантных пределов

k = help.find_k(numOfPoints p q)  # кол-во отбрасываемых точек
print("\tПредел k : " + str(k) + "  Значение биномиального распределения : " + str(
    stats.binom.cdf(numOfPoints - k numOfPoints p)))
# Для всей выборки относительно среднего арифметического
if k % 2 == 0:
    left_lim = int(k / 2)
    right_lim = int(numOfPoints - k / 2)
    tolerant_interval_average[0] tolerant_interval_average[1] = data[left_lim] data[right_lim]
else:
    left_lim = int((k - 1) / 2)
    right_lim = int(numOfPoints - (k - 1) / 2)
    tolerant_interval_average[0] tolerant_interval_average[1] = data[left_lim] data[right_lim]

# Для всей выборки относительно нуля
# Для этого возьмем модули отрицательных значений и пересортируем выборку
data_abs = np.sort(abs(np.array(data)))
tolerant_interval_zero = [-data_abs[numOfPoints - k + 1] data_abs[numOfPoints - k + 1]]
print("\tТолерантные пределы для всей выборки относительно среднего: " + str(tolerant_interval_average))
print("\tТолерантные пределы для всей выборки относительно нуля" + str(tolerant_interval_zero))

# ЧЕРТИМ
plot.title("Толерантные пределы для интерквантильного \nпромежутка относительно среднего значения")
plot.yticks([])
plot.plot(tolerant_interval_average[0] 0 'b<')
plot.plot(tolerant_interval_average[1] 0 'b>')
plot.plot(data[interquantile_interval[0]] 0 'ro')
plot.plot(data[interquantile_interval[1]] 0 'ro')
plot.legend(("Левый толерантный предел" "Правый толерантный предел" "Интерквантильный промежуток")
loc='upper right')
plot.show()
plot.close()

plot.title("Толерантные пределы относительно нуля")
plot.yticks([])
plot.plot(tolerant_interval_zero[0] 0 'b<')
plot.plot(tolerant_interval_zero[1] 0 'b>')
plot.legend(("Левый толерантный предел" "Правый толерантный предел") loc='upper right')
plot.show()
plot.close()

# Считаем параметрические толерантные пределы подвыборок
k_tolerant_multiplier = 1.96
parametric_tolerant_interval = [[0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0]]
for i in range(10):
    parametric_tolerant_interval[i][0] = mean[i + 1] - k_tolerant_multiplier * root_of_dispersion[i + 1]
    parametric_tolerant_interval[i][1] = mean[i + 1] + k_tolerant_multiplier * root_of_dispersion[i + 1]
print("\tПараметрические толерантные интервалы для подвыборок:")
print("\t\t" + str(parametric_tolerant_interval))

axes = []
plot.title("Параметрические толерантные пределы для подвыборок")
for i in range(10):
    if i == 0:
        axes.append(plot.subplot(10 1 i + 1))
    else:
        axes.append(plot.subplot(10 1 i + 1 sharex=axes[0]))
    axes[i].set_yticks([])
    axes[i].set_ylabel(str(i + 1))
```

```
    if i < 9: plot.setp(axes[i].get_xticklabels() visible=False)
    plot.plot(parametric_tolerant_interval[i][0] 0 'b<')
    plot.plot(parametric_tolerant_interval[i][1] 0 'b>')
    plot.plot(mean[i + 1] 0 'ro')
plot.show()
plot.close()

# ============================ ЧАСТЬ 2 ======================================
# ========================== МЕТОД МОМЕНТОВ ====================================
print("==========================МЕТОД МОМЕНТОВ==========================")

# Для нормального распредления
print("\tДля нормального распредления")
print("\t\tc = " + str(mean[0]) + " s = " + str(root_of_dispersion[0]))

a_for_laplace_moment_method = mean[0]
laplace_lambda_moment_method = math.sqrt(2 / dispersion[0])
print("\tДля распределения Лапласа")
print("\t\ta = " + str(a_for_laplace_moment_method) + " lambda = " + str(laplace_lambda_moment_method))

k_for_gamma_moment_method = (mean[0] ** 2) / dispersion[0]
theta_for_gamma_moment_method = dispersion[0] / mean[0]
print("\tДля Гамма-распредления")
print("\t\tk = " + str(k_for_gamma_moment_method) + " lambda = " + str(theta_for_gamma_moment_method))
#
k_for_chi_square_method = mean[0]
print("\tДля распределения Хи-квадрат")
print("\t\tk = " + str(k_for_chi_square_method))
#
lamda_for_exp_moment = 1/mean[0]
print("\tДля экспоненциального распределения")
print("\t\tlambda = " + str(lamda_for_exp_moment))
#
disp_for_lognorm_moment = root_of_dispersion[0]
mu_for_lognorm_moment = mean[0]
print("\tДля логнормального распределения")
print("\t\tdisp = " + str(disp_for_lognorm_moment))
print("\t\tmu = " + str(mu_for_lognorm_moment))
#
disp_for_relay_moment = mean[0] * np.sqrt(2/np.pi)
print("\tДля распределения Рэлея")
print("\t\tdisp = " + str(disp_for_relay_moment))

n_for_student_moment = (2 * dispersion[0]) / (dispersion[0] - 1)
print("\tДля распределения Стьюдента")
print("\t\tn = " + str(n_for_student_moment))

# ================================== ММП ================================================
print("==========================ММП==========================")

# Для нормального распределения
c_for_normal_mmp = 1 / numOfPoints * sum(data)
dispersion_for_normal_mmp = 1 / numOfPoints * sum((np.array(data) - c_for_normal_mmp) ** 2)
s_for_normal_mmp = math.sqrt(dispersion_for_normal_mmp)
print("\tДля нормального распределения")
print("\t\tc = " + str(c_for_normal_mmp) + " s = " + str(s_for_normal_mmp))

# Для распределения Лапласа
a_for_laplace_mmp = mean[0]
laplace_lambda_mmp = numOfPoints * (1 / sum(abs(np.array(data) - a_for_laplace_mmp)))
print("\tДля распределения Лапласа")
print("\t\ta = " + str(a_for_laplace_mmp) + " lambda = " + str(laplace_lambda_mmp))
#
# # Для Гамма-распределения
# # Числовые значения которые нужно посчитать
for_optimize1 = 0
for_optimize2 = 0
square_sum = 0
for v in data:
    if v > 0:
        square_sum += v * v
        for_optimize1 += v
        for_optimize2 += np.log(v)
```

```
for_optimize3 = for_optimize1
for_optimize1 = np.log(for_optimize1 / numOfPoints)
for_optimize4 = for_optimize2
for_optimize2 = for_optimize2 / numOfPoints
c_mmp = for_optimize1 - for_optimize2
#
#Достаем градиент Гамма-функции и ищем ее минимум
gamma_gradient = help.gammaGradient(c_mmp).gamma_gradient
k_for_gamma_mmp = help.fmin_bisection(gamma_gradient 0.5 100 1e-14)
theta_for_gamma_mmp = for_optimize3 / (k_for_gamma_mmp * numOfPoints)
print("\tДля Гамма-распределения")
print("\t\tk = " + str(k_for_gamma_mmp) + " theta = " + str(theta_for_gamma_mmp))
#
#
# # # Для Хи-квадрат-распределения
# # # Числовые значения которые нужно посчитать
log_of_sums = for_optimize4/2
# #
chi_gradient = help.chiGradient(log_of_sums numOfPoints).chi_gradient
k_for_chi_square_method_mmp = help.fmin_bisection(chi_gradient 1 10 1e-14)
print("\tДля Хи-квадрат-распределения")
print("\t\tk = " + str(k_for_chi_square_method_mmp))
#
# # Для экспоненциального распределения
lambda_for_exp_mmp = mean[0]
lambda_for_exp_mmp = 1 / mean[0]
print("\tДля экспоненциального распределения")
print("\t\tlambda = " + str(lambda_for_exp_mmp))
#
#
mu_for_lognorm_mmp = for_optimize2
buffer = 0
for v in data:
    if v > 0:
        buffer += (np.log(v) - mu_for_lognorm_mmp)*(np.log(v) - mu_for_lognorm_mmp)
disp_for_lognorm_mmp = buffer/(numOfPoints * 2)
print("\tДля логнормального распределения")
print("\t\tdisp = " + str(disp_for_lognorm_mmp))
print("\t\tmu = " + str(mu_for_lognorm_mmp))
#
mu_for_lognorm_matlab = 0.5791
disp_for_lognorm_matlab = 1.9999
#
disp_for_relay_mmp = np.sqrt(square_sum/(numOfPoints*2))
disp_for_relay_mmp = np.sqrt(square_sum/(numOfPoints*2))
print("\tДля распределения Рэлея")
print("\t\tdisp = " + str(disp_for_relay_mmp))

mu_for_student_mmp = 0.0267811
sigma_for_student_mmp = 1.03866
nu_for_student_mmp = 5.55669

# ======================= Построим финции распределения и плотности вместе с гистограммой

# Для нормального распределения
# https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html#scipy.stats.norm
plot.title("Сравнение с плотностью нормального распределения")
#plot.xlim([0 100])
plot.bar(steps for_relative width=h color=(0.2 0.6 0.3 0.4))
plot.plot(data stats.norm.pdf(np.array(data) loc=mean[0] scale=root_of_dispersion[0]) 'b')
plot.plot(data stats.norm.pdf(np.array(data) loc=c_for_normal_mmp scale=s_for_normal_mmp) 'r')
plot.legend(("Метод моментов" "ММП" "Гистограмма") loc='upper right')
plot.show()
plot.close()

plot.title("Сравнение с нормальным распределением")
# plot.xlim([0 1000])
plot.bar(steps distribution_fun / numOfPoints width=h color=(0.2 0.6 0.3 0.4))
plot.plot(data stats.norm.cdf(np.array(data) loc=mean[0] scale=root_of_dispersion[0]) 'b')
plot.plot(data stats.norm.cdf(np.array(data) loc=c_for_normal_mmp scale=s_for_normal_mmp) 'r')
plot.legend(("Метод моментов" "ММП" "Эмпирическая") loc='upper right')
plot.show()
plot.close()
```

```
# Для распределения Лапласа

plot.title("Сравнение с плотностью распределения Лапласа")
# plot.xlim([0 1000])
plot.bar(steps for_relative width=h color=(0.2 0.6 0.3 0.4))
plot.plot(data
          stats.laplace.pdf(np.array(data) loc=a_for_laplace_moment_method scale=1 /
laplace_lambda_moment_method)
          'b')
plot.plot(data stats.laplace.pdf(np.array(data) loc=a_for_laplace_mmp scale=1 / laplace_lambda_mmp) 'r')
plot.legend(("Метод моментов" "ММП" "Гистограмма") loc='upper right')
plot.show()
plot.close()

plot.title("Сравнение с распределением Лапласа")
# plot.xlim([0 1000])
plot.bar(steps distribution_fun / numOfPoints width=h color=(0.2 0.6 0.3 0.4))
plot.plot(data stats.laplace.cdf(np.array(data) loc=mean[0] scale=1 / laplace_lambda_moment_method) 'b')
plot.plot(data stats.laplace.cdf(np.array(data) loc=a_for_laplace_mmp scale=1 / laplace_lambda_mmp) 'r')
plot.legend(("Метод моментов" "ММП" "Эмпирическая") loc='upper right')
plot.show()
plot.close()

# # Для Гамма-распределения
# # https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gamma.html
plot.title("Сравнение с плотностью Гамма-распределения")

plot.bar(steps for_relative width=h color=(0.2 0.4 0.6 0.6))
plot.plot(data stats.gamma.pdf(np.array(data) k_for_gamma_moment_method
scale=theta_for_gamma_moment_method) 'b')
plot.plot(data stats.gamma.pdf(np.array(data) k_for_gamma_mmp scale=theta_for_gamma_mmp) 'r')
plot.legend(("Метод моментов" "ММП" "Гистограмма") loc='upper right')
plot.show()
plot.close()
#
plot.title("Сравнение с Гамма-распределением")
plot.bar(steps distribution_fun / numOfPoints width=h color=(0.2 0.4 0.6 0.6))
plot.plot(data stats.gamma.cdf(np.array(data) k_for_gamma_moment_method
scale=theta_for_gamma_moment_method) 'b')
plot.plot(data stats.gamma.cdf(np.array(data) k_for_gamma_mmp scale=theta_for_gamma_mmp) 'r')
plot.legend(("Метод моментов" "ММП" "Эмпирическая") loc='upper right')
plot.show()
plot.close()


# ========================== ПРОВЕРКА ГИПОТЕЗ ==========================================
print("========================== ПРОВЕРКА ГИПОТЕЗ ==============================")
_nk = np.empty(m)
index = 0
for val in distribution_fun:
    if index == 0:
        _nk[index] = val
    else:
        _nk[index] = val - distribution_fun[index - 1]
    index += 1

# ============== Хи-квадрат=========================================================
print("============== Хи-квадрат статистика====================")
print("\tКритическое значение = 45.0763 ( 46.1730 для одного параметра )")  # Значение получено в MATLAB
CHANGE

print("\tДля нормального распределения")
index = 0
chi2_stat = 0

y_pdf x_pdf = np.histogram(data bins=100)

for i in range(m):
    if i == 0:
        ___Pk = stats.norm.cdf(steps[index] loc=mean[0] scale=root_of_dispersion[0]) -\
                stats.norm.cdf(min_value loc=mean[0] scale=root_of_dispersion[0])
    else:
```

```
        ___Pk = stats.norm.cdf(steps[index] loc=mean[0] scale=root_of_dispersion[0]) -\
                stats.norm.cdf(steps[index - 1] loc=mean[0] scale=root_of_dispersion[0])
    chi2_stat += (numOfPoints * ___Pk - _nk[index]) / (numOfPoints * ___Pk)
    index += 1
print("\t\tДля метода моментов = " + str(chi2_stat))

index = 0
chi2_stat = 0
for i in range(m):
    if i == 0:
        ___Pk = stats.norm.cdf(steps[index] loc=c_for_normal_mmp
scale=s_for_normal_mmp)/stats.norm.cdf(min_value loc=c_for_normal_mmp scale=s_for_normal_mmp)
    else:
        ___Pk = stats.norm.cdf(steps[index] loc=c_for_normal_mmp scale=s_for_normal_mmp) -
stats.norm.cdf(steps[index - 1] loc=c_for_normal_mmp scale=s_for_normal_mmp)
    chi2_stat += (numOfPoints * ___Pk - _nk[index]) ** 2 / (numOfPoints * ___Pk)
    index += 1
print("\t\tДля ММП = " + str(chi2_stat))

print("\tДля распределения Лапласа")
index = 0
chi2_stat = 0
for i in range(m):
    if i == 0:
        ___Pk = stats.laplace.cdf(steps[index] loc=a_for_laplace_moment_method
                                  scale=1 / laplace_lambda_moment_method) - \
                stats.laplace.cdf(min_value loc=a_for_laplace_moment_method scale=1 /
laplace_lambda_moment_method)
    else:
        ___Pk = stats.laplace.cdf(steps[index] loc=a_for_laplace_moment_method
                                  scale=1 / laplace_lambda_moment_method) - \
                stats.laplace.cdf(steps[index - 1] loc=a_for_laplace_moment_method
                                  scale=1 / laplace_lambda_moment_method)
    chi2_stat += (numOfPoints * ___Pk - _nk[index]) ** 2 / (numOfPoints * ___Pk)
    index += 1
print("\t\tДля метода моментов = " + str(chi2_stat))

index = 0
chi2_stat = 0
for i in range(m):
    if i == 0:
        ___Pk = stats.laplace.cdf(steps[index] loc=a_for_laplace_mmp scale=1 / laplace_lambda_mmp) - \
                stats.laplace.cdf(min_value loc=a_for_laplace_mmp scale=1 / laplace_lambda_mmp)
    else:
        ___Pk = stats.laplace.cdf(steps[index] loc=a_for_laplace_mmp scale=1 / laplace_lambda_mmp) - \
                stats.laplace.cdf(steps[index - 1] loc=a_for_laplace_mmp scale=1 / laplace_lambda_mmp)
    chi2_stat += (numOfPoints * ___Pk - _nk[index]) ** 2 / (numOfPoints * ___Pk)
    index += 1
print("\t\tДля ММП = " + str(chi2_stat))

print("\tДля Гамма-распределения")
# # Здесь мы начинаем цикл от 4 так как иначе будут взяты отрицательные значения которые в гамма
распределении
# # отсутсвуют а значит ___Pk будет ноль и мы получим деление на ноль
index = 0
chi2_stat = 0
for i in range(0 m):
    ___Pk = stats.gamma.cdf(steps[index] k_for_gamma_moment_method scale=theta_for_gamma_moment_method)
- \
            stats.gamma.cdf(steps[index - 1] k_for_gamma_moment_method
scale=theta_for_gamma_moment_method)
    chi2_stat += (numOfPoints * ___Pk - _nk[index]) ** 2 / (numOfPoints * ___Pk)
    index += 1
print("\t\tДля метода моментов = " + str(chi2_stat))
index = 0
chi2_stat = 0
for i in range(0 m):
    ___Pk = stats.gamma.cdf(steps[index] k_for_gamma_mmp scale=theta_for_gamma_mmp) - \
            stats.gamma.cdf(steps[index - 1] k_for_gamma_mmp scale=theta_for_gamma_mmp)
    chi2_stat += (numOfPoints * ___Pk - _nk[index]) ** 2 / (numOfPoints * ___Pk)
    index += 1
print("\t\tДля ММП = " + str(chi2_stat))
#
```

```
# ============== КОЛМАГОРОВА - СМИРНОВА=================================================================
print("============== статистика КОЛМАГОРОВА - СМИРНОВА ====================")
___Dcrit = np.sqrt(- (np.log(0.5 * 0.05) / (2 * numOfPoints))) - 1 / (6 * numOfPoints)
print("\tКритическое значение = " + str(___Dcrit))

print("\tДля нормального распределения")
___D = 0
index = 1
for val in data:
    _____ddd = abs(stats.norm.cdf(val loc=mean[0] scale=root_of_dispersion[0]) - index / numOfPoints)
    if _____ddd > ___D: ___D = _____ddd
    index += 1
print("\t\tДля метода моментов = " + str(___D))

___D = 0
index = 1
for val in data:
    _____ddd = abs(stats.norm.cdf(val loc=c_for_normal_mmp scale=s_for_normal_mmp) - index /
numOfPoints)
    if _____ddd > ___D: ___D = _____ddd
    index += 1
print("\t\tДля ММП = " + str(___D))

print("\tДля распределения Лапласа")
___D = 0
index = 1
for val in data:
    _____ddd = abs(stats.laplace.cdf(val loc=a_for_laplace_moment_method
                                     scale=1 / laplace_lambda_moment_method) - index / numOfPoints)
    if _____ddd > ___D: ___D = _____ddd
    index += 1
print("\t\tДля метода моментов = " + str(___D))

___D = 0
index = 1
for val in data:
    _____ddd = abs(stats.laplace.cdf(val loc=a_for_laplace_mmp scale=1 / laplace_lambda_mmp) - index /
numOfPoints)
    if _____ddd > ___D: ___D = _____ddd
    index += 1
print("\t\tДля ММП = " + str(___D))
print("\tДля Гамма-распределения")
___D = 0
index = 1
for val in data:
    _____ddd = abs(stats.gamma.cdf(val k_for_gamma_moment_method
                                   scale=theta_for_gamma_moment_method) - index / numOfPoints)
    if _____ddd > ___D: ___D = _____ddd
    index += 1
print("\t\tДля метода моментов = " + str(___D))
___D = 0
index = 1
for val in data:
    _____ddd = abs(stats.gamma.cdf(val k_for_gamma_mmp scale=theta_for_gamma_mmp) - index / numOfPoints)
    if _____ddd > ___D: ___D = _____ddd
    index += 1
print("\t\tДля ММП = " + str(___D))
#


# ======================= критерий Мизеса ================================
print("============== статистика Мизеса ====================")
print("\tКритическое значение = 0.2415")  # Значение взято из таблицы

print("\tДля нормального распределения")
___w = 0
index = 1
for val in data:
    ___w += (stats.norm.cdf(val loc=mean[0] scale=root_of_dispersion[0]) - (2 * index - 1) / (2 *
numOfPoints)) ** 2
    index += 1
___w = 1 / (12 * numOfPoints) + ___w
```

```
print("\t\tДля метода моментов = " + str(___w))

___w = 0
index = 1
for val in data:
    ___w += (stats.norm.cdf(val loc=c_for_normal_mmp scale=s_for_normal_mmp) - (2 * index - 1) / (
            2 * numOfPoints)) ** 2
    index += 1
___w = 1 / (12 * numOfPoints) + ___w
print("\t\tДля ММП = " + str(___w))

print("\tДля распределения Лапласа")
___w = 0
index = 1
for val in data:
    ___w += (stats.laplace.cdf(val loc=a_for_laplace_moment_method
                               scale=1 / laplace_lambda_moment_method) - (2 * index - 1) / (2 *
numOfPoints)) ** 2
    index += 1
___w = 1 / (12 * numOfPoints) + ___w
print("\t\tДля метода моментов = " + str(___w))

___w = 0
index = 1
for val in data:
    ___w += (stats.laplace.cdf(val loc=a_for_laplace_mmp
                               scale=1 / laplace_lambda_mmp) - (2 * index - 1) / (2 * numOfPoints)) ** 2
    index += 1
___w = 1 / (12 * numOfPoints) + ___w
print("\t\tДля ММП = " + str(___w))

print("\tДля Гамма-распределения")
___w = 0
index = 1
for val in data:
    ___w += (stats.gamma.cdf(val k_for_gamma_moment_method
                             scale=theta_for_gamma_moment_method) - (2 * index - 1) / (2 * numOfPoints))
** 2
    index += 1
___w = 1 / (12 * numOfPoints) + ___w
print("\t\tДля метода моментов = " + str(___w))
___w = 0
index = 1
for val in data:
    ___w += (stats.gamma.cdf(val k_for_gamma_mmp scale=theta_for_gamma_mmp) - (2 * index - 1) / (
            2 * numOfPoints)) ** 2
    index += 1
___w = 1 / (12 * numOfPoints) + ___w
print("\t\tДля ММП = " + str(___w))
```