|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | | |
|  | | | |
| Кафедра теоретической и прикладной информатики | | | |
| По дисциплине СППМиНПО | | | |
| Практическое задание №1 | | | |
| «Устойчивые методы оценивания параметров статистических моделей» | | | |
|  | | | |
|  | Группа | ПМм-21 |
| Вариант 2а | Лоточ Е. Ю., Славич А.Н. |
|  |  |
|  |  |
|  |  |
| Преподаватель | Лисицин Д. В. |
| Дата | 24.12.2022 |
|  | | | |
| Новосибирск 2022 | | | |

**Цель:** Изучить методы робастного оценивания параметра сдвига распределений случайных величин.

Вариант 2а. Лёгкий уровень сложности, немного дополненный средним. Обобщённое гауссовское распределение при параметре формы

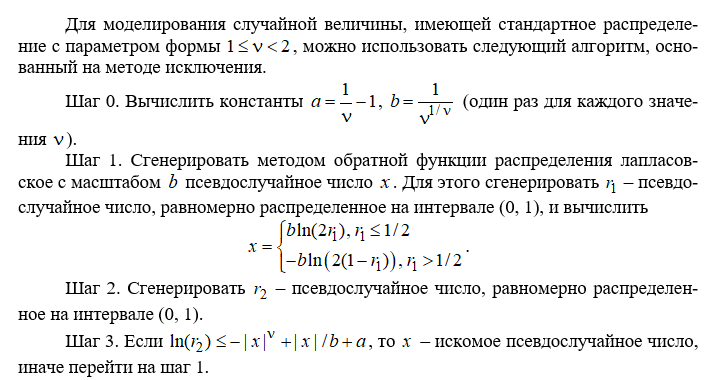
 = 1.5.

Приступим к выполнению работы.

**Задание:** разработать программу, которая реализует генерацию наборов данных с заданным в варианте чистым распределением и засоренным распределением, использовать засоряющие распределения, совпадающие с чистым с точностью до значений параметров сдвига и масштаба, ;

1. Генерация распределения

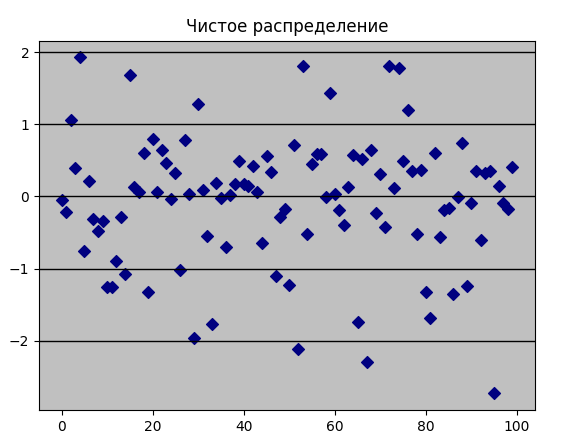
Формируем чистое распределение с помощью следующего алгоритма:



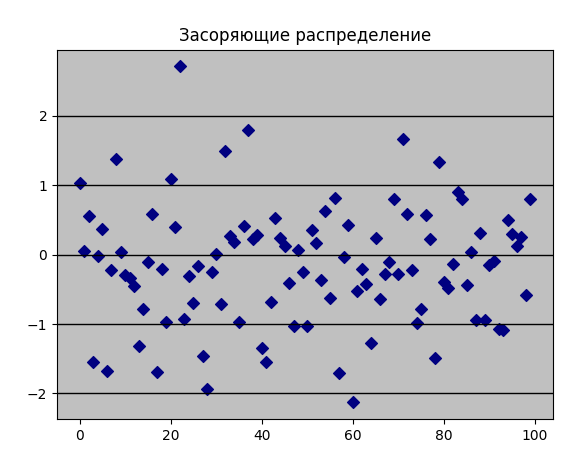
Согласно алгоритму, параметр “b” является масштабом для лаплассавского и в результате вычисления равн 0.763

Параметр “a” является сдвигом лаплассавского и равен (-0.333) при расчёте, учитывая параметр формы.

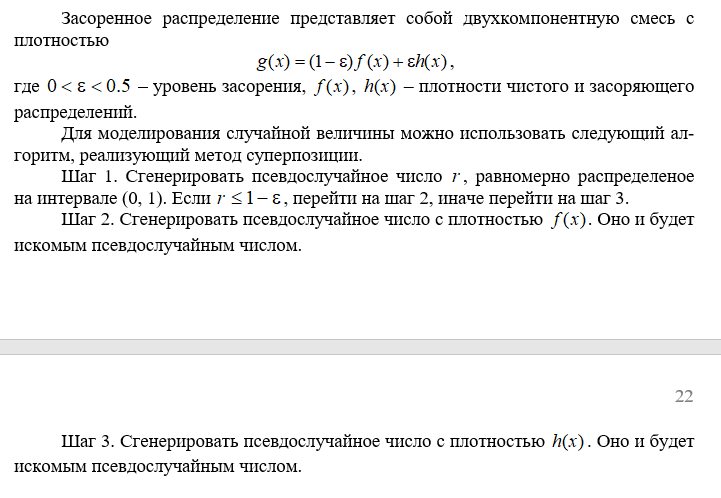
Полученное с помощью данного алгоритма чистое распределение:

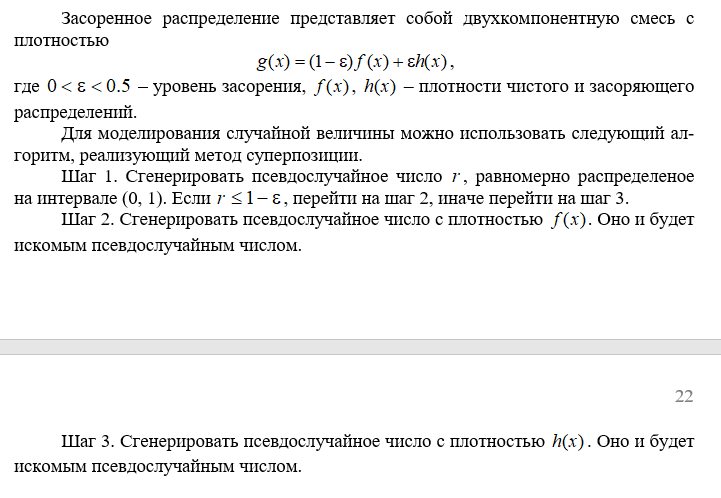


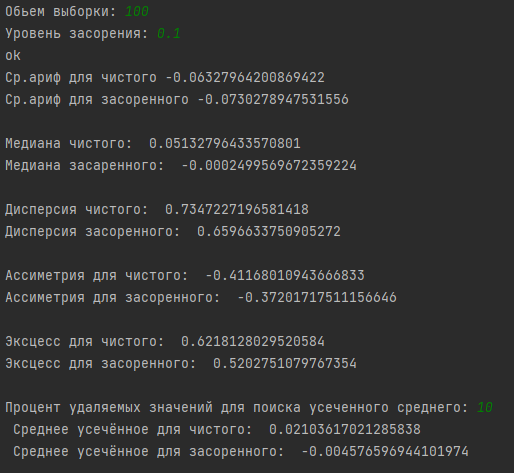
Генерацию наборов данных с заданным в варианте чистым распределением и засоренным распределением, использовать засоряющие распределения, совпадающие с чистым с точностью до значений параметров сдвига и масштаба:



Засорённое распределение получается путём смешивания чистого распределения с засоряющим:

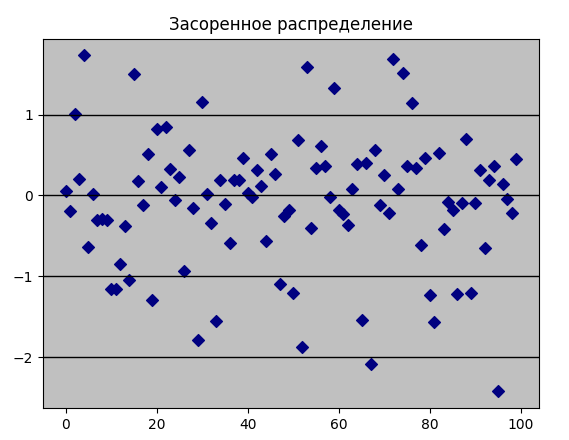






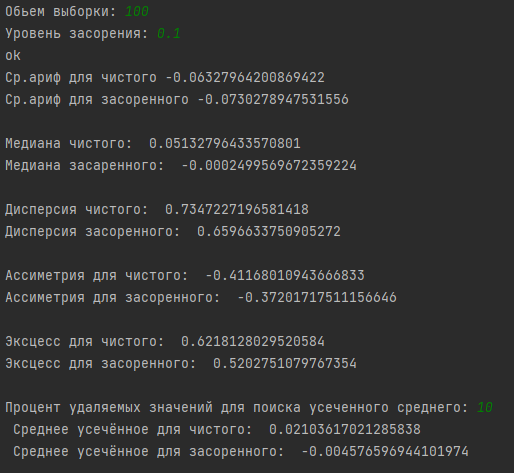
Укажем процент засорения 10%

Результат работы алгоритма засорения:

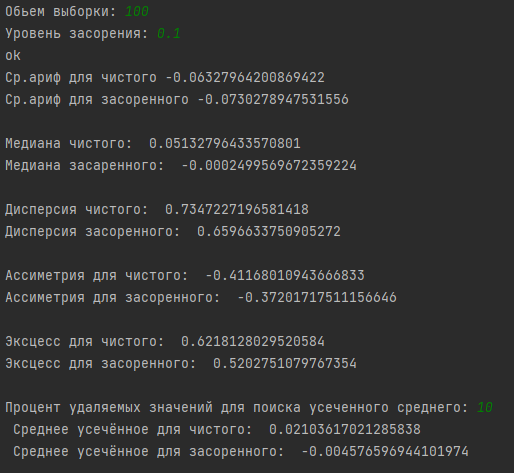


При сравнении полученных рисунков, заметим, что засорённое распределение представляет собой по большей части чистое распределение с приближением значений на указанный процент к значениям засоряющего распределения.

Вычисление выборочных характеристик: среднего арифметического, выборочной медианы, дисперсии, коэффициентов асимметрии, эксцесса представлены на следующем рисунке:

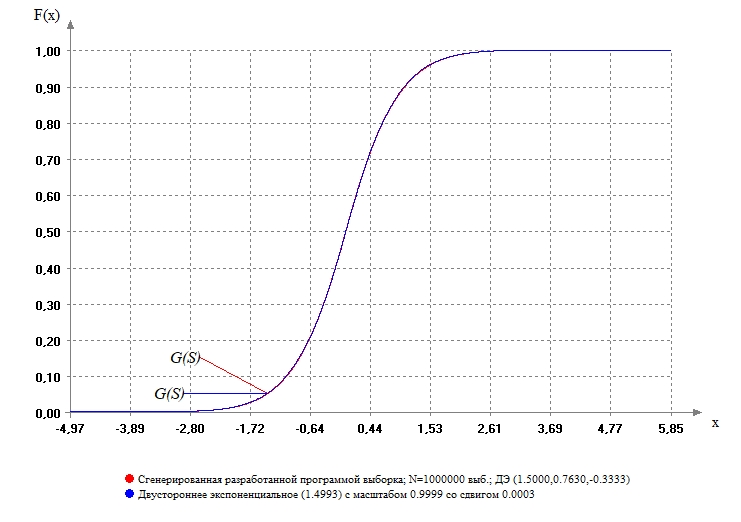


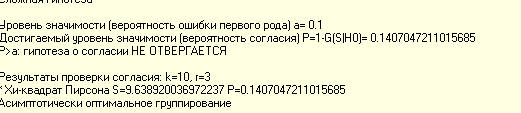
Усечённое среднее представляет собой среднее арифметическое без учёта определённого количества наибольших и наименьших значений выборки. Таким образом, для анализа усеченного среднего предусмотрен ввод процента удаляемых значений.



1. Проверка генератора чистого распределения путём сравнения с их теоретическими значениями на выборке большого объема (N = )

При помощи программы ISW графически сравним “двустороннее экспоненциальное распределение” (Ближайшее к обобщённому Гауссовскому ) и полученное с помощью разработанного генератора чистое распределение.

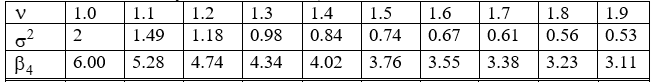




Как можно видеть на графике, двустороннее экспоненциальное распределение незначительно совпадает с полученным законом, что даёт основания считать проверку успешно пройденной. Также можем заметить, что данные графики совпадают при совпадении параметра формы (1.5), параметр масштаба равен 1, параметр формы равен 0. Используем эти данные при оценке устойчивости генератора.

1. Проверка устойчивости оценок, посредством генерации нескольких наборов распределений по рассчитанному закону.

Для удобства проверки результатов оценок будем пользоваться следующими зависимостями:

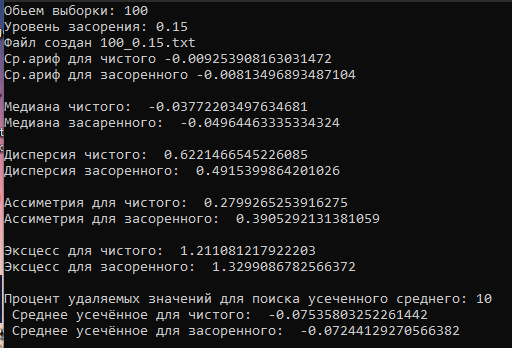
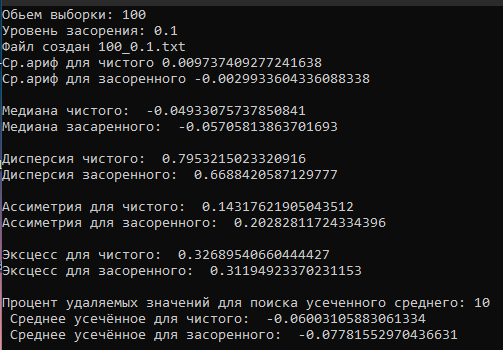
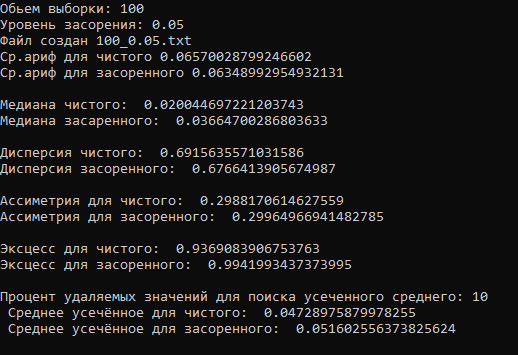


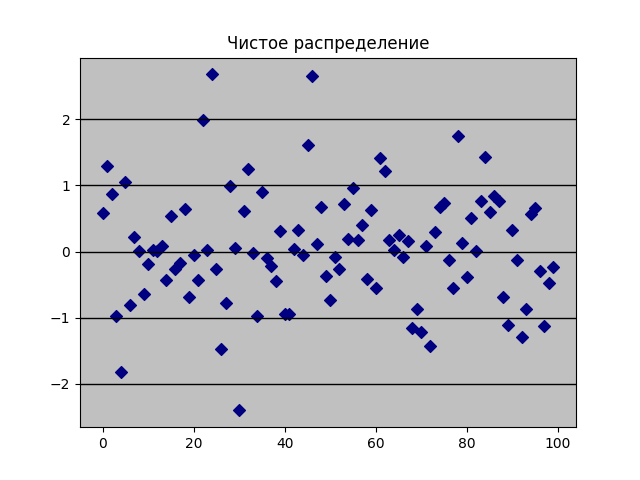
Где

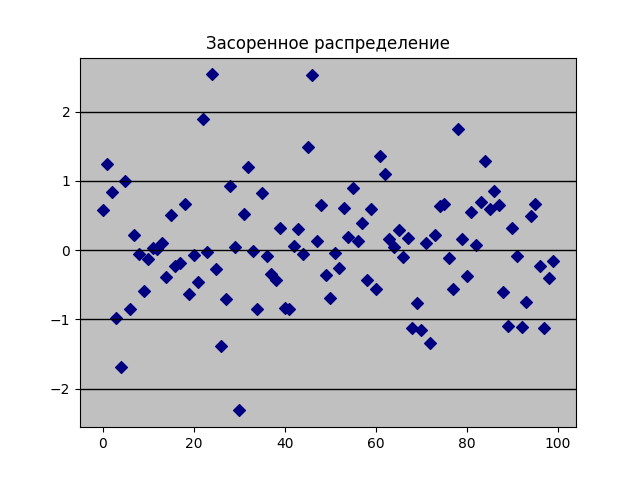
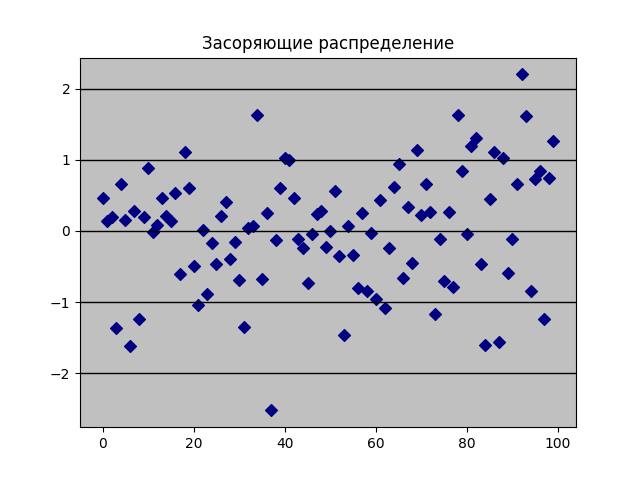
по Пирсону или по Фишеру. Для удобства реализации на языке Python, будем использовать оценку коэффициента эксцесса по Фишеру.

Также, согласно методической литературе, значения медианы должно приближаться к значению параметра сдвига , а ассиметрия стремиться к 0.

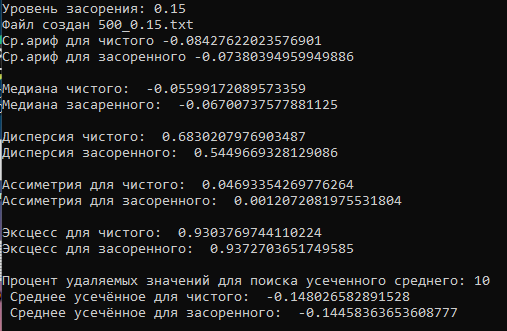
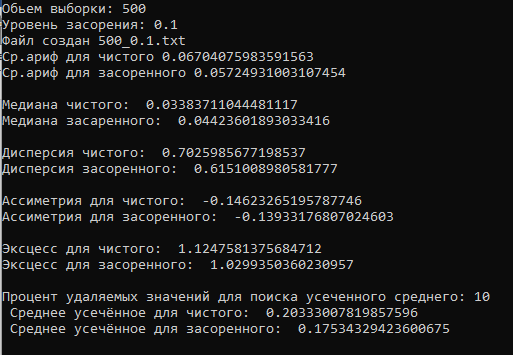
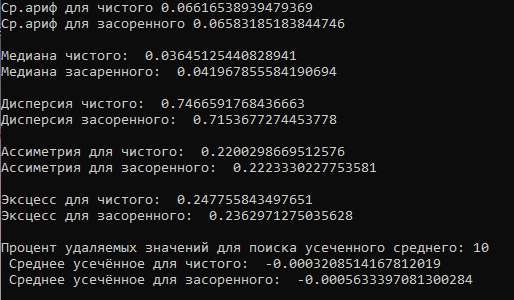
N = 100

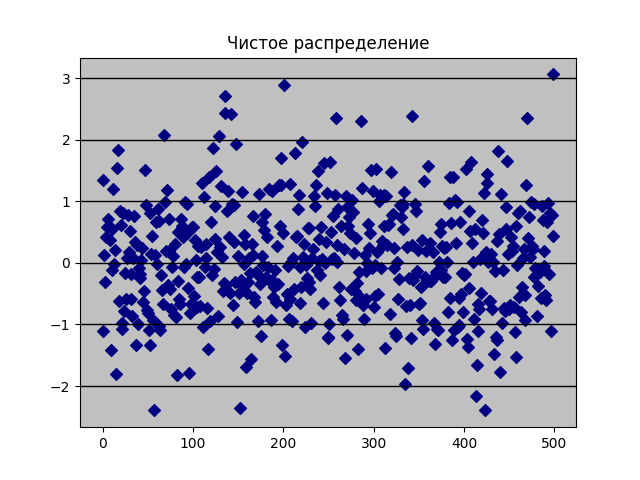


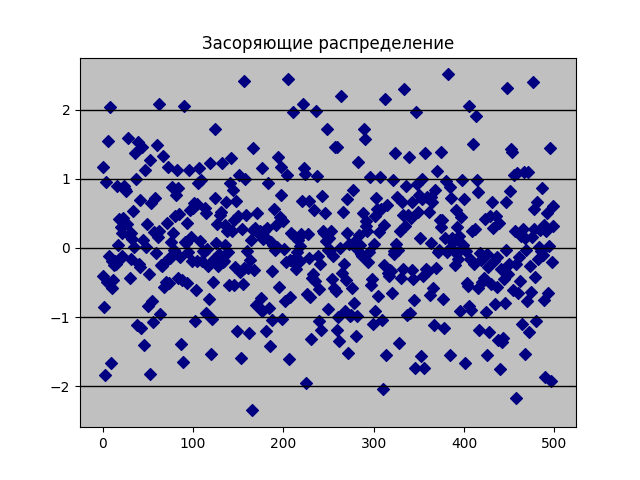


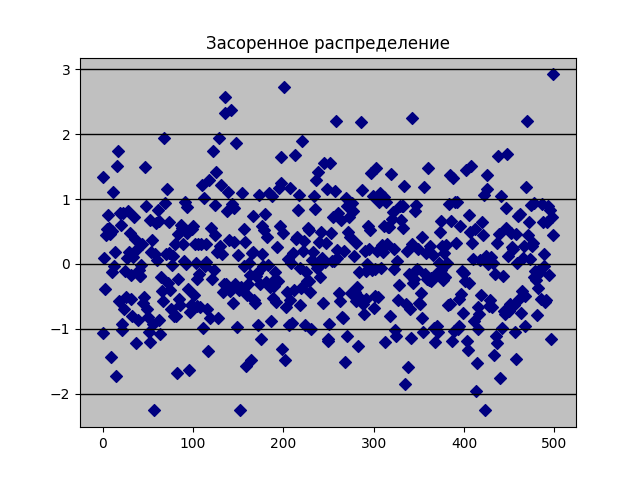


N = 500

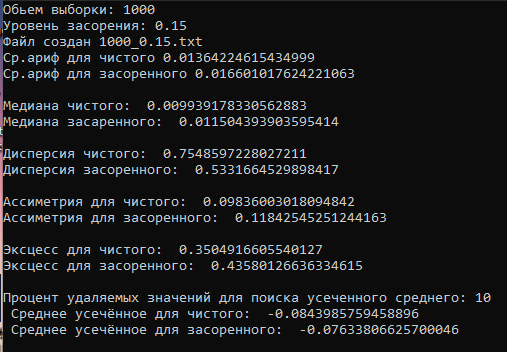
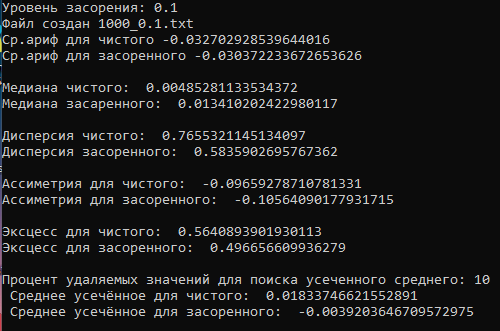
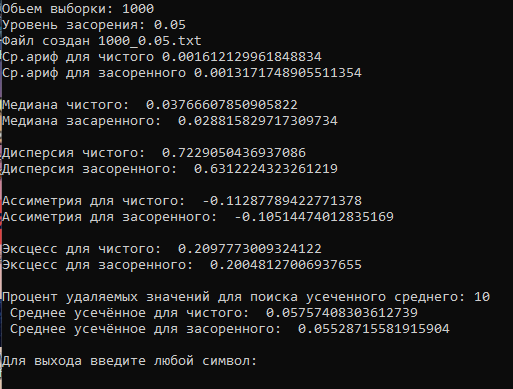


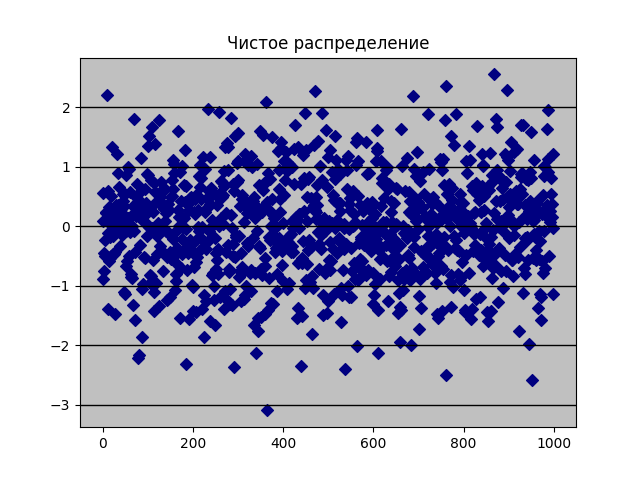


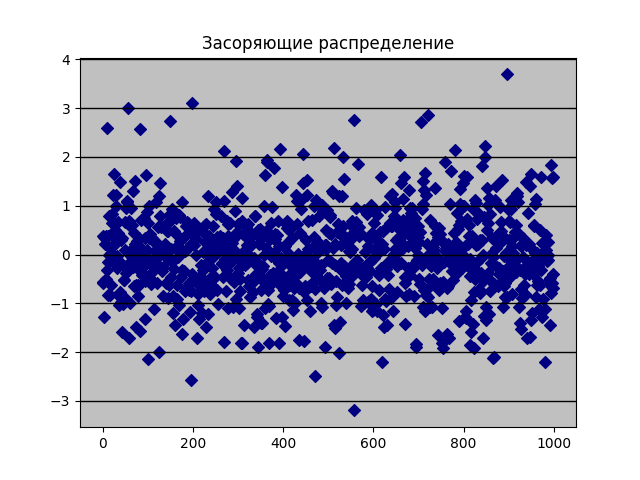


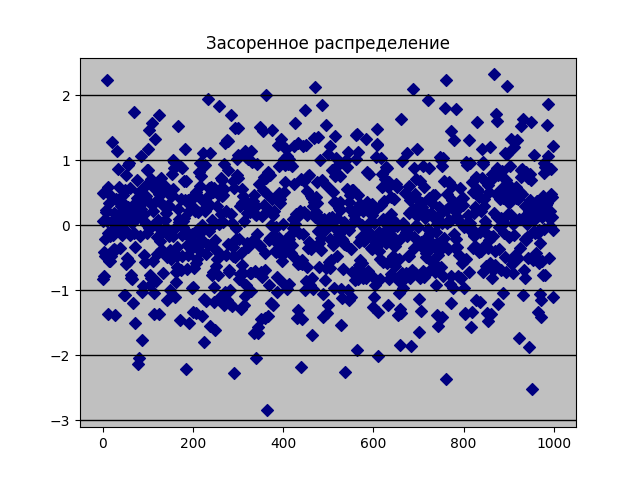


N = 1000

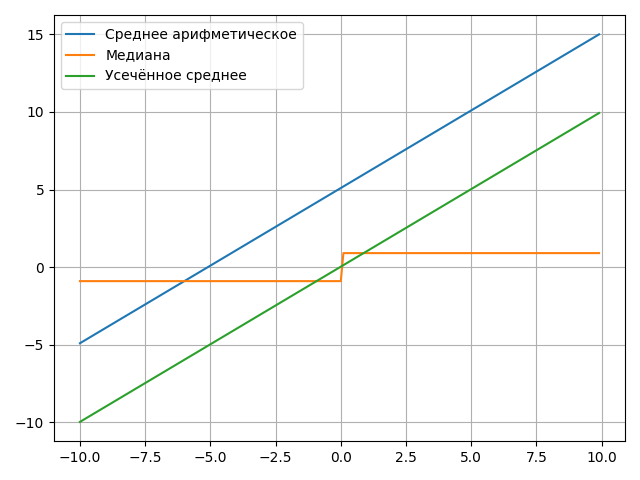








Функции влияния



**Выводы:** Был разработан генератор чистого и засорённого распределения, соответствующего обобщенному гауссовскому распределению. Генератор прошёл проверку на совпадение полученного распределения с табличным. При анализе оценок можно сделать следующие умозаключения:

1. Медиана отражает параметр сдвига и потому в данном случае должна стремиться к 0, при проверке оценок видно, что это условие соблюдается одинаково для любого объема выборки.

2. Табличный параметр дисперсии равен 0.74. При сравнении этого значения с полученными в результате проверки оценок, видим, что все значения действительно близки к табличному с некоторым отклонением, уменьшающимся с ростом объема выборки.

3. Параметр асимметрии должен стремиться к 0, что при полученных оценках соблюдается. Можем сделать вывод, что график остаётся симметричным и при засорении, и при различных объемах выборки.

4. Параметр эксцесса был оценен по методу Фишера. Это значит, что от табличного значения необходимо отнять 3 для получения необходимого в результате проверки значения, в нашем случае, 3.76 – 3 = 0.76. Сверив результаты оценок, видим, что эксцесс, отвечающий за остроту пика графика действительно стремится к заданному значению, причём значение становится более устойчивым с увеличением объема выборки.

5. Оценка усечённого среднего и среднего арифметического также устойчиво близки по значению к заданному параметру сдвига, равному 0, независимо от объема выборки.

6. Засорение выборки с использованием засоряющего распределения, совпадающего с чистым с точностью до значений параметров сдвига и масштаба, особо негативно сказывается на параметре дисперсии, уменьшая его, причём с ростом процента засорения, изменение параметра становится более ощутимым. На других параметрах не имеет однозначного влияния.

Текст программы:

import matplotlib.pyplot as plt  
import numpy as np  
from random import random  
import math  
  
import scipy  
import scipy.stats as sps  
  
v = 1.5 # paramert formi  
a = (1 / v) - 1 # сдвиг  
b = 1 / (v \*\* (1 / v)) # маштаб  
  
clean\_array = []  
clog\_array = []  
n = 0  
  
  
def gauss\_element():  
 global n  
  
 r1 = random()  
 if r1 <= 1 / 2:  
 x = b \* math.log(r1 \* 2)  
 else:  
 x = -b \* math.log(2 \* (1 - r1))  
  
 r2 = random()  
 if math.log(r2) <= -(abs(x) \*\* v) + (abs(x) / b) + a:  
 n = x  
 else:  
 gauss\_element()  
  
i = int(input('Oбьем выборки: '))  
for j in range(i):  
 gauss\_element()  
 clean\_array.append(n)  
 gauss\_element()  
 clog\_array.append(n)  
  
clogged = []  
a\_clog = []  
b\_clog = []  
e = float(input("Уровень засорения: "))  
for j in range(i):  
 a\_clog.append(clean\_array[j] \* (1 - e))  
 b\_clog.append(clog\_array[j] \* e)  
 clogged.append(a\_clog[j] + b\_clog[j])  
  
fig = plt.figure()  
ax = fig.add\_subplot(111)  
ax.set\_facecolor('silver')  
ax.set\_title('Чистое распределение')  
u = np.arange(0, i)  
plt.scatter(u, clean\_array, marker='D').set(facecolor='navy')  
plt.grid(axis='y', linewidth=1, color='black')  
  
fig = plt.figure()  
ax = fig.add\_subplot(111)  
ax.set\_facecolor('silver')  
ax.set\_title('Засоряющие распределение')  
u = np.arange(0, i)  
plt.scatter(u, clog\_array, marker='D').set(facecolor='navy')  
plt.grid(axis='y', linewidth=1, color='black')  
  
  
fig = plt.figure()  
ax = fig.add\_subplot(111)  
ax.set\_facecolor('silver')  
ax.set\_title('Засоренное распределение')  
u = np.arange(0, i)  
plt.scatter(u, clogged, marker='D').set(facecolor='navy')  
plt.grid(axis='y', linewidth=1, color='black')  
  
plt.show()  
  
print("Ср.ариф для чистого", sum(clean\_array) / i) # Average clean  
# print("Ср.ариф засоряющие", sum(clog\_array) / i) # Average clog  
print("Ср.ариф для засоренного", sum(clogged) / i) # Average clogged  
print('')  
  
sum\_clean = sum(clean\_array)  
sum\_clogged = sum(clogged)  
  
M\_clean = np.median(clean\_array)  
M\_clog = np.median(clog\_array)  
M\_clogged = np.median(clogged)  
  
  
print("Медиана чистого: ", M\_clean)  
# print("Засоряющие медиана: ", M\_clog)  
print("Медиана засаренного: ", M\_clogged)  
print('')  
  
D\_clean = np.var(clean\_array)  
D\_clog = np.var(clog\_array)  
D\_clogged = np.var(clogged)  
  
print("Дисперсия чистого: ", D\_clog)  
# print("Дисперсия засоряющая: ", D\_clean)  
print("Дисперсия засоренного: ", D\_clogged)  
print('')  
  
result\_clean = scipy.stats.describe(clean\_array, ddof=1, bias=False)  
result\_clog = scipy.stats.describe(clog\_array, ddof=1, bias=False)  
result\_clogged = scipy.stats.describe(clogged, ddof=1, bias=False)  
  
skewness\_clean = result\_clean.skewness # Ассиметрия чистая  
skewness\_clog = result\_clog.skewness # Ассиметрия засоряющая  
skewness\_clogged = result\_clogged.skewness # Ассиметрия засоренная  
  
kurtosis\_clean = result\_clean.kurtosis # Эксцесс чистая  
kurtosis\_clog = result\_clog.kurtosis # Эксцесс засоряющая  
kurtosis\_clogged = result\_clogged.kurtosis # Эксцесс засоренная  
  
print("Ассиметрия для чистого: ", skewness\_clean)  
# print("Ассиметрия засоряющая: ", skewness\_clog)  
print("Ассиметрия для засоренного: ", skewness\_clogged)  
print('')  
  
print("Эксцесс для чистого: ", kurtosis\_clean)  
# print("Эксцесс засоряющая: ", kurtosis\_clog)  
print("Эксцесс для засоренного: ", kurtosis\_clogged)  
print('')  
  
k = int(input('Процент удаляемых значений для поиска усеченного среднего: '))  
Count = k / 100 \* i # Сколько нужно удалить максимальных элементов  
Sum\_clean = sum(clean\_array)  
Sum\_clog = sum(clog\_array)  
Sum\_clogged = sum(clogged)  
  
kk = i-2\*k # Cколько элементов должно остаться  
while Count > 0:  
 Sum\_clean = Sum\_clean - max(clean\_array) - min(clean\_array)  
 Sum\_clog = Sum\_clog - max(clog\_array) - min(clog\_array)  
 Sum\_clogged = Sum\_clogged - max(clogged) - min(clogged)  
 Count -= 1  
  
T\_clean\_array = Sum\_clean/kk  
T\_clog\_array = Sum\_clog/kk  
T\_clogged = Sum\_clogged/kk  
  
print(" Среднее усечённое для чистого: ", T\_clean\_array)  
# print(" засоряющая: ", T\_clog\_array)  
print(" Среднее усечённое для засоренного: ", T\_clogged)  
  
#Расчёт функций влияния  
  
def IF\_med(y,tetta,lambd,coef):  
 IF\_result = []  
 for i in y:  
 u = (lambd \*\* 2 \* np.sign(i - tetta)) / (2 \* coef)  
 IF\_result = np.append(IF\_result, u)  
 return IF\_result  
  
def IF\_avg(y,tetta):  
 IF\_result = []  
 for i in y:  
 u = i - tetta  
 IF\_result = np.append(IF\_result, u)  
 return IF\_result  
  
def IF\_avgc(y,tetta, Count, k, lambd):  
 IF\_result = []  
 ff = 1/1-2\*Count  
 for i in y:  
 ff2 = (i - tetta) / lambd  
 if ff2 <= -k:  
 u = -k  
 if k >= ff2 > -k:  
 u = ff2  
 if ff2 > k:  
 u = k  
 u = ff \* u  
 IF\_result = np.append(IF\_result, u)  
 return IF\_result  
  
coef = v/(2\*math.gamma(1/v))  
a = -10  
b = 10  
h = 0.1  
lambd = 1  
x\_in = np.arange(a, b, h)  
tetta = sum\_clogged  
AVG = IF\_avg(x\_in, tetta)  
tetta = M\_clogged  
MED = IF\_med(x\_in, tetta, lambd, coef)  
tetta = T\_clogged  
AVGC = IF\_avgc(x\_in,tetta,Count,k,lambd)  
  
plt.plot(x\_in, AVG, label="Среднее арифметическое")  
plt.plot(x\_in, MED, label="Медиана")  
plt.plot(x\_in, AVGC, label="Усечённое среднее")  
plt.legend()  
plt.grid(True)  
plt.tight\_layout()  
plt.show()