|  |  |  |
| --- | --- | --- |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования  «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическая работа | | |
| по дисциплине «Современные проблемы прикладной математики и наукоёмкого прикладного обеспечения» | | |
| **методы устойчивого оценивания параметра сдвига  распределений непрерывных случайных величин** | | |
|  | Группа: ПММ-42 | Будник Светлана |
|  | Вариант: 10.В | Самсонов Семён |
|  |  | Судеревская Ирина |
|  | Преподаватель | Лисицин Даниил Валерьевич |
|  |  |  |
| Новосибирск, 2024 | | |

Содержание

[1. Цель работы 3](#_Toc181221377)

[2. Ход работы 4](#_Toc181221378)

[2.1. Проверка генерации чистого, засоряющего и засорённого распределений 4](#_Toc181221379)

[2.1.1. Чистое распределение 4](#_Toc181221380)

[2.1.2. Загрязняющее распределение 5](#_Toc181221381)

[2.1.3. Загрязнённое распределение 6](#_Toc181221382)

[2.2. Вычисление оценок параметра сдвига 9](#_Toc181221383)

[2.2.1. О вычислении оценок параметра сдвига 9](#_Toc181221384)

[2.2.2. Оценка параметра сдвига для чистого распределения 10](#_Toc181221385)

[2.2.3. Оценка параметра сдвига симметричного засорения 11](#_Toc181221386)

[2.2.4. Оценка параметра сдвига ассиметричного засорения 12](#_Toc181221387)

[2.3. Графики функций влияния оценок 13](#_Toc181221388)

[3. Выводы 18](#_Toc181221389)

[4. Исходный код программы 19](#_Toc181221390)

# Цель работы

Изучить методы устойчивого оценивания параметра сдвига распределений непрерывных случайных величин.

Вариант задания: 10.В – обобщённое распределение Лапласа со значением параметра  
.

Дисперсия: коэффициент эксцесса: .

# Ход работы

## Проверка генерации чистого, засоряющего и засорённого распределений

### Чистое распределение

Моделирование случайной величины распределения Лапласа происходит по следующей формуле:

где – реализация случайной величины, равномерно распределённой на интервале ; если нет ни одного удовлетворяющего условия элемента в произведении, то произведение равно 1.

Для проверки корректности генератора чистого распределения, сгенерируем несколько выборок разного объёма и сравним выборочные характеристики полученных выборок с их теоретическими значениями (таблица 1).

Таблица 1. Сравнение теоретических и выборочных характеристик для чистого распределения

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Объём выборки** | **Мат ожидание** | | **Медиана** | | **Дисперсия** | | **Асимметрия** | | **Эксцесс** | |
| **Т** | **П** | **Т** | **П** | **Т** | **П** | **Т** | **П** | **Т** | **П** |
| 10000 | 0 | -0.00238 | 0 | -0.00129 | 8 | 8.06043 | 0 | -0.02783 | 0.75 | 0.69796 |
| 100000 | 0 | 0.00082 | 0 | 0.00538 | 8 | 8.01069 | 0 | 0.00379 | 0.75 | 0.73337 |
| 1000000 | 0 | 0.00096 | 0 | 0.00026 | 8 | 8.01254 | 0 | 0.00567 | 0.75 | 0.73396 |

### Засоряющее распределение

Функция плотности случайной величины засоряющего распределения Лапласа принимает следующий вид:

Моделирование случайной величины засоряющего распределения Лапласа происходит по следующей формуле:

Теоретические значения характеристик для засоряющего распределения находятся следующим образом:

1. Матожидание: ;
2. Медиана: ;
3. Коэффициент асимметрии: ;
4. Коэффициент эксцесса: .

Для проверки корректности генератора засоряющего распределения, сгенерируем несколько выборок разного объёма и сравним выборочные характеристики полученных выборок с их теоретическими значениями (таблицы 2 и 3).

Таблица 2. Сравнение теоретических и выборочных характеристик для засоряющего распределения ().

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Объём выборки** | **Мат ожидание** | | **Медиана** | | **Дисперсия** | | **Асимметрия** | | **Эксцесс** | |
| **Т** | **П** | **Т** | **П** | **Т** | **П** | **Т** | **П** | **Т** | **П** |
| 10000 | 4 | 4.07849 | 4 | 4.04918 | 32 | 32.09559 | 0 | -0.00064 | 0.75 | 0.68993 |
| 100000 | 4 | 4.01401 | 4 | 4.02281 | 32 | 31.77302 | 0 | 0.01837 | 0.75 | 0.76731 |
| 1000000 | 4 | 4.00102 | 4 | 4.00004 | 32 | 32.00511 | 0 | 0.00548 | 0.75 | 0.74395 |

Таблица 3. Сравнение теоретических и выборочных характеристик для засоряющего распределения ().

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Объём выборки** | **Мат ожидание** | | **Медиана** | | **Дисперсия** | | **Асимметрия** | | **Эксцесс** | |
| **Т** | **П** | **Т** | **П** | **Т** | **П** | **Т** | **П** | **Т** | **П** |
| 10000 | 0 | 0.07849 | 0 | 0.04918 | 32 | 32.09559 | 0 | -0.00064 | 0.75 | 0.68993 |
| 100000 | 0 | 0.01401 | 0 | 0.02281 | 32 | 31.77302 | 0 | 0.01837 | 0.75 | 0.76731 |
| 1000000 | 0 | 0.00102 | 0 | 0.00004 | 32 | 32.00511 | 0 | 0.00548 | 0.75 | 0.74395 |

### Засорённое распределение

Функция плотности случайной величины засорённого распределения Лапласа принимает следующий вид:

Моделирование случайной величины засорённого распределения Лапласа происходит по следующему алгоритму:

1. Сгенерировать случайное число , равномерно распределённое на интервале . Если , перейти на шаг 2, иначе перейти на шаг 3.
2. Сгенерировать случайное число с плотностью (чистое распределение). Оно и будет искомым числом.
3. Сгенерировать случайное число с плотностью (засоряющее распределение). Оно и будет искомым числом.

Теоретические характеристики засорённого распределения находятся следующим образом:

Для проверки корректности генератора засорённого распределения, сгенерируем несколько выборок разного объёма и сравним выборочные характеристики полученных выборок с их теоретическими значениями (таблицы 4 и 5). Коэффициент зашумления .

Таблица 4. Сравнение теоретических и выборочных характеристик для засорённого распределения ().

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Объём выборки** | **Мат ожидание** | | **Дисперсия** | | **Асимметрия** | | **Эксцесс** | |
| **Т** | **П** | **Т** | **П** | **Т** | **П** | **Т** | **П** |
| 10000 | 0.4 | 0.43938 | 11.84 | 12.28219 | 0.74933 | 0.77483 | 3.22434 | 3.08007 |
| 100000 | 0.4 | 0.39444 | 11.84 | 11.75703 | 0.74933 | 0.76140 | 3.22434 | 3.34871 |
| 1000000 | 0.4 | 0.39887 | 11.84 | 11.81183 | 0.74933 | 0.75457 | 3.22434 | 3.21341 |

Таблица 5. Сравнение теоретических и выборочных характеристик для засорённого распределения ().

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Объём выборки** | **Мат ожидание** | | **Дисперсия** | | **Асимметрия** | | **Эксцесс** | |
| **Т** | **П** | **Т** | **П** | **Т** | **П** | **Т** | **П** |
| 10000 | 0 | 0.01578 | 10.4 | 10.55490 | 0 | 0.01901 | 2.54734 | 2.40814 |
| 100000 | 0 | -0.00456 | 10.4 | 10.36931 | 0 | 0.01925 | 2.54734 | 2.58238 |
| 1000000 | 0 | -0.00027 | 10.4 | 10.38299 | 0 | 0.01286 | 2.54734 | 2.49837 |

Графики функций плотности получившихся распределений при объёме выборки 1000000 изображены на рисунках 1 и 2.

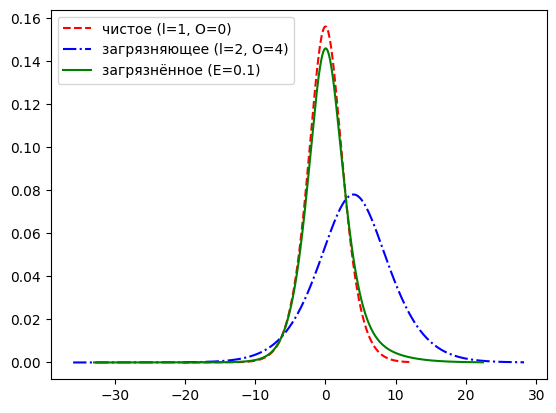


Рисунок 1 – графики функций плотности тестовых распределений для асимметричного зашумления

Изображение выглядит как текст, График, диаграмма, снимок экрана

Автоматически созданное описание

Рисунок 2 – графики функций плотности тестовых распределений для симметричного зашумления

## Вычисление оценок параметра сдвига

### О вычислении оценок параметра сдвига

При вычислении оценок параметра сдвига будут использоваться 3 распределения: чистое, засорённое с симметричным засорением и засорённое с асимметричным засорением. Для каждого из 3 распределений будут сделаны 3 различные выборки размером элементов. Для генерации симметричного засорения будут использоваться параметры . Для генерации ассиметричного засорения: параметры .

Оценку параметра сдвига при помощи среднего арифметического можно провести по следующей формуле:

Оценку параметра сдвига при помощи выборочной медианы можно провести путём подсчёта выборочной медианы. Аналогично и с оценкой по методу усечённого среднего.

Для оценки параметра сдвига по методу максимального правдоподобия, необходимо провести минимизацию следующей функции:

где – функция плотности распределения Лапласа без смещений.

Для оценки параметра сдвига по методу обобщённых радикальных оценок необходимо провести минимизацию следующей функции:

где – параметр, регулирующий степень робастности оценки.

### Оценка параметра сдвига для чистого распределения

Параметры выборки: , , , . Результаты оценок приведены в таблице 6. График функций плотности изображены на рисунке 3.

Таблица 6. Оценка параметра сдвига для чистого распределения

|  |  |  |  |
| --- | --- | --- | --- |
| **Оценка** | **1** | **2** | **3** |
| Среднее арифметическое | 0.031311 | -0.207190 | 0.127137 |
| Медиана | -0.087063 | -0.160677 | 0.017835 |
| Усечённое среднее (0.05) | 0.005833 | -0.235482 | 0.135109 |
| Усечённое среднее (0.10) | -0.024193 | -0.238936 | 0.124555 |
| Усечённое среднее (0.15) | -0.040558 | -0.238717 | 0.119639 |
| Максимальное правдоподобие | -0.008228 | -0.230535 | 0.119671 |
| Обобщённые рад. оценки (0.1) | -0.024548 | -0.238999 | 0.116672 |
| Обобщённые рад. оценки (0.5) | -0.080853 | -0.242184 | 0.096266 |
| Обобщённые рад. оценки (1.0) | -0.123713 | -0.237426 | 0.065782 |

Изображение выглядит как текст, снимок экрана, График, диаграмма

Автоматически созданное описание

Рисунок 3 – график функций плотности чистого распределения

### Оценка параметра сдвига симметричного засорения

Параметры выборки: , , , . Результаты оценок приведены в таблице 7. График функций плотности изображены на рисунке 4.

Таблица 7. Оценка параметра сдвига распределения с симметричным засорением

|  |  |  |  |
| --- | --- | --- | --- |
| **Оценка** | **1** | **2** | **3** |
| Среднее арифметическое | -0.149536 | 0.279051 | -0.155762 |
| Медиана | -0.104019 | 0.131443 | -0.059158 |
| Усечённое среднее (0.05) | -0.132242 | 0.188167 | -0.048394 |
| Усечённое среднее (0.10) | -0.130925 | 0.163511 | -0.047880 |
| Усечённое среднее (0.15) | -0.137261 | 0.142980 | -0.053193 |
| Максимальное правдоподобие | -0.131394 | 0.170082 | -0.068947 |
| Обобщённые рад. оценки (0.1) | -0.121231 | 0.116904 | -0.033999 |
| Обобщённые рад. оценки (0.5) | -0.097883 | 0.035751 | -0.035105 |
| Обобщённые рад. оценки (1.0) | -0.085556 | 0.006322 | -0.061179 |

Изображение выглядит как текст, График, диаграмма, линия

Автоматически созданное описание

Рисунок 4 – график функций плотности распределения с симметричным засорением

### Оценка параметра сдвига ассиметричного засорения

Параметры выборки: , , , . Результаты оценок приведены в таблице 8. График функций плотности изображены на рисунке 5.

Таблица 8. Оценка параметра сдвига распределения с асимметричным засорением

|  |  |  |  |
| --- | --- | --- | --- |
| **Оценка** | **1** | **2** | **3** |
| Среднее арифметическое | 2.650464 | 3.261051 | 2.588238 |
| Медиана | 1.223441 | 1.655123 | 1.439161 |
| Усечённое среднее (0.05) | 2.369304 | 2.949935 | 2.357698 |
| Усечённое среднее (0.10) | 2.194903 | 2.757692 | 2.171514 |
| Усечённое среднее (0.15) | 2.034749 | 2.574047 | 2.002744 |
| Максимальное правдоподобие | 2.015391 | 2.505499 | 2.005297 |
| Обобщённые рад. оценки (0.1) | 1.604532 | 1.950292 | 1.609359 |
| Обобщённые рад. оценки (0.5) | 0.666555 | 0.736931 | 0.747081 |
| Обобщённые рад. оценки (1.0) | 0.272267 | 0.279815 | 0.395158 |

Изображение выглядит как текст, График, диаграмма, линия

Автоматически созданное описание

Рисунок 5 – график функций плотности распределения с асимметричным засорением

## Графики функций влияния оценок

**Функция влияния** – функция, являющаяся одной из мер количественной робастности оценки. Функция влияния определяет воздействие на оценку, оказываемое добавлением к очень большой выборке одного наблюдения в точке . В результате она отражает асимптотическое смещение оценки, вызываемое засорением наблюдений.

Функция влияния среднего арифметического выражается следующей формулой:

Для метода медианы, функция влияния следующая:

Учитывая, что , получаем

График функций влияния среднего арифметического и медианы изображены на рисунке 6.

Изображение выглядит как текст, линия, диаграмма, снимок экрана

Автоматически созданное описание

Рисунок 6 – график функций влияния среднего арифметического и медианы

Функция влияния усечённого среднего имеет следующий вид:

где – квантиль порядка стандартного распределения, то есть,

а – уровень усечения (). Параметр будем находить путём минимизации функционала

График функции влияния усечённого среднего для разных параметров представлен на рисунке 7.

Изображение выглядит как текст, снимок экрана, линия, диаграмма

Автоматически созданное описание

Рисунок 7 – график функции влияния усечённого среднего   
при разных параметрах

Для метода максимального правдоподобия функция влияния задаётся следующим образом:

Производная функции равна

График функции влияния для метода максимального правдоподобия изображён на рисунке 8.

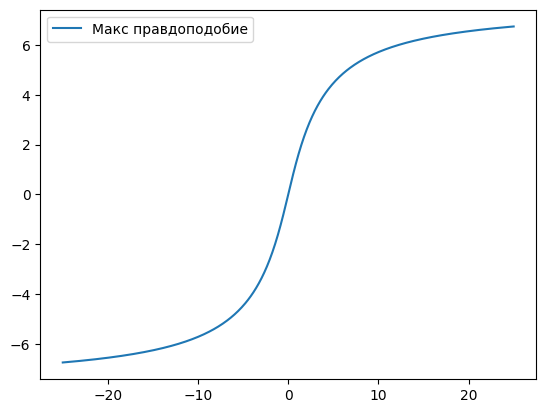


Рисунок 8 – график функции влияния для метода  
максимального правдоподобия

Для метода обобщённых радикальных оценок функция влияния задаётся следующим образом:

График функции влияния для метода обобщённых радикальных оценок с разными параметрами изображён на рисунке 9.

Изображение выглядит как текст, диаграмма, линия, График

Автоматически созданное описание

Рисунок 9 – график функции влияния для метода  
обобщённых радикальных оценок

График всех функций влияния изображён на рисунке 10.

Изображение выглядит как линия, диаграмма, График, скат

Автоматически созданное описание

Рисунок 10 – график всех функций влияния

# Выводы

1. На чистом распределении все оценки показали примерно одинаковые, близкие к истине, результаты. Отклонения от истинного значения возникали вследствие неравномерной генерации выборки случайных значений при сравнительно небольшом объёме выборки.
2. На распределении с симметричным засорением оценки также показали примерно одинаковые результаты.
3. На распределении с ассиметричным засорением лучше всего себя показали оценки на основе медианы и обобщённых радикальных оценок, причём с наибольшим параметром . Причину этому наглядно видно по графикам функций влияния каждого из методов оценки. Функция влияния медианы при значениях и её окрестности (область основного множества значений зашумляющего распределения) постоянна и имеет достаточно низкое значение. Значения функций влияния обобщённых радикальных оценок так же достаточно низкие и при этом стремятся к нулю при удалении от центра (причём чем больше , тем более резко график устремляется к нулю). При этом график функции обобщённых радикальных оценок при находится выше, чем график медианы, поэтому и оценка при этом значении параметра показала результат хуже, чем медиана. Остальные же оценки имеют значения функции влияния в этой области гораздо выше, поэтому они показывают менее верный результат. Хуже всего себя проявляет оценка по среднему арифметическому, поскольку значение её функции влияния не ограничено сверху и линейно растёт по мере удаления от нулевой точки.

# Исходный код программы

[Проект Google Colab](https://drive.google.com/file/d/1xlpupjlTiXtqIkL17IcSUHNizv0eozZp/view?usp=sharing)

# %% [markdown]

# # Вариант 10.В

# %% [markdown]

# ## О чём работа в блокноте

#

# Обобщенное распределение Лапласа со следующим значением параметра $n=4$.

#

# $$

# f(x,n) = f(x,4) = \frac{e^{-|x|}}{(n-1)!\cdot 16} \sum\limits\_{j=0}^{3}\frac{(3+j)!}{(3-j)!\cdot j!} \frac{|x|^{3-j}}{2^{j}} = \

# $$

#

# $$

# \ = e^{-|x|}\cdot\frac{|x|^{3}+6|x|^{2} + 15|x| + 15}{96}.

# $$

#

# Дисперсия: $\sigma^{2}= 8$,

#

# Коэффициент эксцесса: $\large\gamma\_{2} = \frac{3}{4}$.

#

# %% [markdown]

# ## Как пользоваться блокнотом?

#

# В блокноте будут идти вычисления ТОЛЬКО ПО ОДНОЙ ВЫБОРКЕ (все графики, всё прочее строятся только для одной выборки). Поэтому для того, чтобы получить значения на разных выборках, необходимо менять глобальные параметры, расположенные в блоке кода ниже.

#

# Для того, чтобы быстро перезапустить вычисление во всех ячейках, можно нажать `ctrl+F9`, либо через верхнее меню: `Среда выполнения -> выполнить все`.

# %%

# базовые импорты

import math

import statistics

import matplotlib.pyplot as plt

import numpy as np

from scipy import stats as st

from scipy import integrate

from scipy.integrate import quad

from scipy.optimize import minimize

import pandas as pd

# Глобальные параметры

count = 10000 # количество которое хочешь сгенерить

eps = 0.1  # Эпсилон (коэффициент зашумления)

rng = np.random.default\_rng(

    12345 # 12345, 23456, 34567

)  # Генератор рандомных чисел с фиксированным сидом (для повторяемости результатов)

# Параметры для чистого распределения

N\_net = 4  # Параметр распределения

l\_net = 1  # Масштаб (лямбда) НЕ МЕНЯТЬ

O\_net = 0  # Смещение (Тетта) НЕ МЕНЯТЬ

# Параметры для шума

N\_scattener = N\_net  # Параметр распределения

l\_scattener = 2  # Масштаб (лямбда). 1 по умолчанию

O\_scattener = 4  # Смещение (Тетта). 0 по умолчанию

# %% [markdown]

# ## 1 Разработка программы

# %% [markdown]

# ### 1.1. Генерация данных

# %% [markdown]

# > Полное задание пункта

# > 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.

#

# %% [markdown]

# #### Генерация чистого и загрязняющего распределения

#

# Обобщённое распределение Лапласа с параметром $n=4$ задаётся следующим образом:

#

# $$

# f(x) = e^{-|x|}\cdot\frac{|x|^{3}+6|x|^{2} + 15|x| + 15}{96}.

# $$

#

# Дисперсия при этом равна $\sigma^{2}=8$, коэффициент эксцесса: $\large\gamma\_{2}=\frac{3}{4}$.

#

# Моделирование случайных величин осуществляется по следующей формуле:

#

# $$

# x = \ln\left(\frac{\prod\limits\_{i:r\_{i} \leqslant \frac{1}{2}} 2r\_{i}}{\prod\limits\_{i: r\_{i} > \frac{1}{2}}2(1-r\_{i})}\right),

# $$

#

# где $r\_{i}, i=\{1, \dots, n\}= \{1,2,3,4\}$ – \*реализация случайной величины\*, равномерно распределённой на интервале $(0; 1)$ (просто 4 значения с генератора равномерных случайных чисел); если нет ни одного элемента, удовлетворяющего условию в произведении, то оно равно 1.

#

# #### Генерация засоряющего распределения

#

# Засоряющее распределение должно (судя по всему) повторять исходное распределение, но также должно иметь возможность изменять свой масштаб и сдвиг.  Сдвиг-масштабное преобразование осуществляется по следующей формуле:

#

# $$

# x = \theta + \lambda x\_{1},

# $$

#

# где $\theta$ – параметр сдвига, $\lambda$ – параметр масштаба, $x\_{1}$ – значение случайной величины с нулевым сдвигом и единичным масштабом.

#

# Применимо к формуле нашего распределения, получаем:

#

# $$

# x = \theta + \lambda \ln\left(\frac{\prod\limits\_{i:r\_{i} \leqslant \frac{1}{2}} 2r\_{i}}{\prod\limits\_{i: r\_{i} > \frac{1}{2}}2(1-r\_{i})}\right).

# $$

#

#

# %%

def gen\_rand\_laplas(n\_lap: int, theta: float = 0.0, lamb: float = 1.0) -> float:

    """

    Генератор случайной величины, соответствующей обобщенному распределению Лапласа, с возможностью задания сдвига и масштабирования

    Args:

        n\_lap (int): Натуральное число, соответствющее количеству случайных величин. Является параметром обобщённого распределения Лапласа `n`

        theta (float, optional): параметр сдвига распределения тэтта. По умолчанию 0.0

        lamb (float, optional): параметр масштабироваиня сдвига лямбда. По умолчанию 1.0

    Returns:

        float: Случайная величина, соответствующая обобщенному распределению Лапласа.

    """

    global rng

    list\_rand = rng.random((n\_lap,))

    res = 1.0

    for item in list\_rand:

        if item <= (1 / 2):

            res \*= 2 \* item

        else:

            res /= 2 \* (1 - item)

    return np.log(res) \* lamb + theta

def gen\_full\_laplas(

    count: int, theta: float = 0.0, lamb: float = 1.0

) -> np.ndarray[np.float64]:

    """Генератор множества случайных величин по распределению Лапласа

    Args:

        count (int): размер выборки случайных чисел по данному распределению

        theta (float, optional): параметр сдвига распределения тэтта. По умолчанию 0.0

        lamb (float, optional): параметр масштабироваиня сдвига лямбда. По умолчанию 1.0

    Returns:

        ndarray[float64]: множество значений полученной выборки

    """

    global N\_net

    res = [gen\_rand\_laplas(N\_net, theta=theta, lamb=lamb) for \_ in range(count)]

    return np.asarray(res)

print(f"Размер выборки: {count} элементов...")

print("Строим чистое распределение...")

clean = gen\_full\_laplas(count)

print("Строим загрязняющее распределение...")

scattener = gen\_full\_laplas(count, theta=O\_scattener, lamb=l\_scattener)

# %% [markdown]

# #### Генерация засорённого распределения

#

# Засорённое распределение представляет собой смесь двух распределений с плотностью, задаваемой следующей формулой:

#

# $$

# g(x) = (1 - \varepsilon) f(x) + \varepsilon h(x),

# $$

#

# где $0 < \varepsilon < 0.5$ – уровень засорения, $f(x), h(x)$ – плотности чистого и засоряющего распределений соответственно.

#

# Моделирование случайной величины с засорённым распределением можно проводить по следующему алгоритму:

#

# 1. Сгенерировать случайное число $r$, равномерно распределённое на интервале $(0; 1)$. Если $r \leqslant 1 - \varepsilon$, перейти на шаг 2, иначе перейти на шаг 3.

# 2. Сгенерировать случайное число с плотностью $f(x)$ (чистое распределение). Оно и будет искомым числом.

# 3. Сгенерировать случайное число с плотностью $h(x)$ (загрязняющее распределение). Оно и будет искомым числом.

# %%

def get\_scattened\_laplas(

    clean: np.ndarray[np.float64], scattener: np.ndarray[np.float64], eps: float

) -> np.ndarray[np.float64]:

    """Соединяет чистое и загрязняющее распределения в единое загрязнённое

    Args:

        clean (np.ndarray[np.float64]): множество случайных величин из чистого распределения

        scattener (np.ndarray[np.float64]): множество случайных величин из загрязняющего распределения (должно совпадать по размеру с чистым)

        eps (float): коэффициент загрязнения (0 <= `eps` <= 0.5)

    Returns:

        np.ndarray[np.float64]: полученное загрязнённое распределение

    """

    global rng

    res = np.zeros\_like(clean)

    rand = rng.random((clean.size,))

    for ind in range(res.size):

        if rand[ind] <= (1 - eps):

            res[ind] = clean[ind]

        else:

            res[ind] = scattener[ind]

    return res

scattened = get\_scattened\_laplas(clean, scattener, eps)

# %% [markdown]

# #### Вывод графиков

#

#

# %%

def f\_laplas(data: np.ndarray, theta: float = 0.0, lamb: float = 1.0) -> np.ndarray:

    """

    Функция плотности распределения Лапласа от значений X

    Args:

        data (np.ndarray): значения X текущего распределения (желательно должна быть отсортированной)

        theta (float): значение параметра сдвига функции (по умолчанию 0)

        lamb (float): значение параметра масштабирования функции (по умолчанию 1)

    Returns:

        np.ndarray: значения функции плотности распределения Лапласа

    """

    ordered = np.sort(data) # Сортируем последовательность

    ordered = (ordered - theta) / lamb # Убираем сдвиги

    ordered = np.abs(ordered) # Убираем знаки значений

    result = np.exp(-1 \* ordered)

    result \*= (ordered\*\*3 + 6 \* ordered\*\*2 + 15\*ordered + 15) / (96 \* lamb)

    return result

def f\_laplas\_mixed(mixed: np.ndarray) -> np.ndarray:

    """

    Функция плотности смешанного распределения от значений X чистого и загрязняющего распределений

    Args:

        mixed (np.ndarray): значения X смешанного распределения

    Returns:

        np.ndarray: значения функции плотности смешанного распределения Лапласа

    """

    mixed\_sorted = np.sort(mixed) # Сортируем последовательность

    f\_clean = f\_laplas(mixed\_sorted, O\_net, l\_net)

    f\_scattener = f\_laplas(mixed\_sorted, O\_scattener, l\_scattener)

    return (1 - eps) \* f\_clean + eps \* f\_scattener

def plot\_graphics(

    clean: np.ndarray[np.float64],

    scattener: np.ndarray[np.float64],

    scattened: np.ndarray[np.float64],

):

    """Выводит графики сгенерированных распределений на экран (приводя размеры к единичному по графику `clean`)

    Args:

        clean (np.ndarray[np.float64]): массив значений из чистого распределения

        scattener (np.ndarray[np.float64]): массив значений из загрязняющего распределения

        scattened (np.ndarray[np.float64]): массив значений из грязного распределения

    """

    step = 1

    if clean.size > 10000:

        step = int(clean.size / 5000)

    clean\_sorted = np.sort(clean)[::step]

    scattener\_sorted = np.sort(scattener)[::step]

    scattened\_sorted = np.sort(scattened)[::step]

    f\_clean = f\_laplas(clean\_sorted, O\_net, l\_net)

    f\_scattener = f\_laplas(scattener\_sorted, O\_scattener, l\_scattener)

    f\_scattened = f\_laplas\_mixed(scattened\_sorted)

    # Выводим полученные графики

    fig, axes = plt.subplots()

    axes.plot(clean\_sorted, f\_clean, label=f"чистое (l={l\_net}, O={O\_net})", linestyle="--", color="red")

    axes.plot(scattener\_sorted, f\_scattener, label=f"загрязняющее (l={l\_scattener}, O={O\_scattener})", linestyle="-.", color="blue")

    axes.plot(scattened\_sorted, f\_scattened, label=f"загрязнённое (E={eps})", linestyle="-", color="green")

    # axes.plot([10.0, 10.0], [0.0, 1.1], label="Центр", color="purple")

    axes.legend()

    plt.show()

print("Графики функций плотности полученных распределений")

print(f"Чистое распределение: Тетта = {O\_net}, Лямбда = {l\_net}")

print(f"Загрязняющее распределение: Тетта = {O\_scattener}, Лямбда = {l\_scattener}")

plot\_graphics(clean, scattener, scattened)

# %% [markdown]

# ### 1.4. Вычисление математического ожидания, дисперсии, коэффициентов асимметрии и эксцесса случайной величины с засоренным распределением по заданным характеристикам чистого и засоряющего распределений

# %% [markdown]

# Для того, чтобы получить значения данных характеристик для загрязнённого распределения, надо найти их для чистого и загрязняющего. Ищуются они по следующим формулам:

#

# $$

# M = 0 + \theta,

# $$

#

# $$

# D = \sigma^{2} = 2\cdot n \cdot \lambda^{2},

# $$

#

# $$

# \gamma\_{1} = 0, ???

# $$

#

# $$

# \gamma\_{2} = \frac{3}{n}.

# $$

#

# %%

dataframe\_labels = [

    "Мат ожидание М",

    "Дисперсия D",

    "Ассиметрия gamma\_1",

    "Эксцесс gamma\_2",

]

# Теоретические величины

M1\_ = 0 + O\_net            # Матожидание

D1\_ = 2 \* N\_net \* l\_net\*\*2 # Дисперсия

G11\_ = 0                   # Ассиметрия

G12\_ = 3 / N\_net           # Эксцесс

print(f"Для чистого распределения ({count} точек):")

pd.DataFrame(np.array([dataframe\_labels, [M1\_, D1\_, G11\_, G12\_]]).T, columns=["Параметр чистого", "Значение"])

# %%

# Теоретические величины

M2\_ = 0 + O\_scattener                  # Матожидание

D2\_ = 2 \* N\_scattener \* l\_scattener\*\*2 # Дисперсия

G21\_ = 0 # Ассиметрия

G22\_ = 3 / N\_scattener # Эксцесс

print(f"Для загрязняющего распределения ({count} точек):")

pd.DataFrame(np.array([dataframe\_labels, [M2\_, D2\_, G21\_, G22\_]]).T, columns=["Параметр загрязняющего", "Значение"])

# %% [markdown]

# Для вычисления всяких там параметров полученного засорённого распределения можно воспользоваться следующими формулами. Обозначим $M\_{i}, D\_{i}, \gamma\_{2i}, i=1,2$ мат ожидание, дисперсию и коэффициент эксцесса, соответствующие при $i=1$ и $i=2$ плотностям $f(x)$ и $h(x)$. Тогда матожидание, дисперсия, коэффициенты асимметрии и эксцесса случайной величины с засорённым распределением выражается формулами:

#

# $$

# M = (1-\varepsilon)M\_{1} + \varepsilon M\_{2},

# $$

#

# $$

# D = (1 - \varepsilon)(M\_{1}^{2} + D\_{1}) + \varepsilon(M\_{2}^{2} + D\_{2}) - M^{2},

# $$

#

# $$

# \gamma\_{1} = \frac{1}{D^{1.5}} \left[(1 - \varepsilon) \left\{(M\_{1} - M)^{3} + 3(M\_{1} - M)D\_{1}\right\} + \varepsilon \left\{(M\_{2}-M)^{3} + 3(M\_{2}-M)D\_{2} \right\} \right],

# $$

#

# $$

# \gamma\_{2} = \frac{1}{D^{2}} \left\{ (1-\varepsilon) \left[ (M\_{1} - M)^{4} + 6(M\_{1} - M)^{2}D\_{1} + D\_{1}^{2}(\gamma\_{21} + 3) \right] + \ \right.

# $$

#

# $$

# \ + \left. \varepsilon [(M\_{2}-M)^{4} +6(M\_{2}-M)^{2}D\_{2} + D\_{2}^{2}(\gamma\_{22} + 3) ] \right\} - 3.

# $$

# %%

# Теоретические величины

M\_ = (1 - eps) \* M1\_ + eps \* M2\_

D\_ = (1 - eps) \* (M1\_\*\*2 + D1\_) + eps \* (M2\_\*\*2 + D2\_) - M\_\*\*2

G1\_ = (1 / D\_\*\*1.5) \* (

    (1 - eps) \* ((M1\_ - M\_) \*\* 3 + 3 \* (M1\_ - M\_) \* D1\_)

    + eps \* ((M2\_ - M\_) \*\* 3 + 3 \* (M2\_ - M\_) \* D2\_)

)

G2\_ = (1 / D\_\*\*2) \* (

    (1 - eps) \* ((M1\_ - M\_)\*\*4 + 6 \* (M1\_ - M\_)\*\*2 \* D1\_ + D1\_\*\*2 \* (G12\_ + 3))

        + eps \* ((M2\_ - M\_)\*\*4 + 6 \* (M2\_ - M\_)\*\*2 \* D2\_ + D2\_\*\*2 \* (G22\_ + 3))

) - 3

print(f"Для загрязнённого распределения ({count} точек):")

pd.DataFrame(

    np.array([dataframe\_labels, [M\_, D\_, G1\_, G2\_]]).T,

    columns=["Параметр загрязнённого", "Значение"],

)

# %% [markdown]

# ### 1.2. Вычисление выборочных характеристик полученных распределений

#

# Здесь фактически совпадёт со 2 пунктом, поэтому пропускаю

# %% [markdown]

# ## 2 Провести проверку генератора чистого и засорённого распределений

# %% [markdown]

# Выборочные характеристики считаются следующим образом:

#

# |Название|Формула|Функция|

# |---|:---:|---|

# |Среднее арифметическое $M$ | $\frac{1}{N}\sum\limits\_{i=1}^{N}y\_{i}$ | `numpy.mean(array) -> float` |

# |Медиана $m$ | тю-тю (просто средний элемент) | `numpy.median(array) -> float` |

# |Дисперсия $D$ | $\frac{1}{N}\sum\limits\_{i=1}^{N}(y\_{i}-M)^{2}$ | `numpy.var(array) -> float` |

# |Коэф. асимметрии $\gamma\_1$| $\frac{\sum\limits\_{i=1}^{N} (y\_{i} - \bar{y})^{3}}{N \cdot D^{1.5}}$ |`scipy.stats.skew(array) -> float`|

# |Коэф. эксцесса $\gamma\_2$|$\frac{\sum\limits\_{i=1}^{N}(y\_{i} - \hat{y})^{4}}{N\cdot D^{2}} - 3$|`scipy.stats.kurtosis(array) -> float`|

#

# Для чистого и загрязняющего распределений, медиана равна $0$ и $\theta$ соответственно в силу их симметричности. Для загрязнённого распределения, в силу его возможной несимметричности, теоретическую медиану найти неизвестно как.

#

# %%

dataframe\_labels = [

    "Мат ожидание М",

    "Медиана m",

    "Дисперсия D",

    "Ассиметрия gamma\_1",

    "Эксцесс gamma\_2",

]

# Практические величины

PM1\_ = np.mean(clean)      # Матожидание

Pm1\_ = np.median(clean)    # Медиана

PD1\_ = np.var(clean)       # Дисперсия

PG11\_ = st.skew(clean)     # Ассиметрия

PG12\_ = st.kurtosis(clean) # Эксцесс

print(f"Теоретические и практические величины для чистого распределения ({count} точек)")

pd.DataFrame(

    np.array([

        dataframe\_labels,

        [M1\_, M1\_, D1\_, G11\_, G12\_],

        [PM1\_, Pm1\_, PD1\_, PG11\_, PG12\_],

    ]).T,

    columns=["Параметр чистого", "Теоретические", "Практические"],

)

# %%

# Практические величины

PM2\_ = np.mean(scattener)      # Матожидание

Pm2\_ = np.median(scattener)    # Медиана

PD2\_ = np.var(scattener)       # Дисперсия

PG21\_ = st.skew(scattener)     # Ассиметрия

PG22\_ = st.kurtosis(scattener) # Эксцесс

print(f"Теоретические и практические величины для загрязняющего распределения ({count} точек)")

pd.DataFrame(

    np.array([

        dataframe\_labels,

        [M2\_, M2\_, D2\_, G21\_, G22\_],

        [PM2\_, Pm2\_, PD2\_, PG21\_, PG22\_],

    ]).T,

    columns=["Параметр загрязняющего", "Теоретические", "Практические"],

)

# %%

dataframe\_labels = [

    "Мат ожидание М",

    "Дисперсия D",

    "Ассиметрия gamma\_1",

    "Эксцесс gamma\_2",

]

# Практические величины

PM\_ = np.mean(scattened)      # Матожидание

PD\_ = np.var(scattened)       # Дисперсия

PG1\_ = st.skew(scattened)     # Ассиметрия

PG2\_ = st.kurtosis(scattened) # Эксцесс

print(f"Теоретические и практические величины для загрязнённого распределения ({count} точек)")

pd.DataFrame(

    np.array([

        dataframe\_labels,

        [M\_, D\_, G1\_, G2\_],

        [PM\_, PD\_, PG1\_, PG2\_],

    ]).T,

    columns=["Параметр загрязнённого", "Теоретические", "Практические"],

)

# %% [markdown]

# ## 3 Для выборок с разными видами распределений вычислить следующие оценки параметра сдвига:

# %% [markdown]

# ### 3.1. Среднее арифметическое

#

# Само среднее арифметическое выглядит следующим образом:

#

# $$

# M = \frac{1}{N}\sum\limits\_{i=1}^{N}y\_{i}.

# $$

#

# Среднее арифметическое является М-оценкой параметра сдвига $\theta$ с функцией потерь

#

# $$

# \rho(y, \theta) = (y - \theta)^{2}.

# $$

#

# Для нахождения параметра сдвига необходимо решить следующую задачу минимизации:

#

# $$

# Q(\theta) = \frac{1}{N}\sum\limits\_{i=1}^{N} (y\_{i} - \theta)^{2} \rightarrow \min

# $$

#

# В нашем случае (видимо из-за симметричности распределения и того, что по умолчанию матожидание равно 0), для нахождения параметра сдвига по среднему арифметическому достаточно просто посчитать среднее арифметическое.

# %%

def theta\_approximation\_by\_mean(data: np.ndarray) -> float:

    """

    Функция оценки параметра сдвига при помощи среднего арифметического

    Args:

        data (np.ndarray): последовательность, для которой необходимо выполнить оценку

    Returns:

        float: значение оценки параметра

    """

    return np.mean(data)

theta\_by\_mean = theta\_approximation\_by\_mean(scattened)

print(f"Параметр сдвига по методу среднего арифметического: {theta\_by\_mean}")

# %% [markdown]

# ### 3.2. Выборочная медиана

#

# Выборочная медиана также является М-оценкой, но проще найти параметр сдвига через само значение медианы

# %%

def theta\_approximation\_by\_median(data: np.ndarray) -> float:

    """

    Функция оценки параметра сдвига при помощи медианы

    Args:

        data (np.ndarray): последовательность, для которой необходимо выполнить оценку

    Returns:

        float: значение оценки параметра

    """

    return np.median(data)

theta\_by\_median = theta\_approximation\_by\_median(scattened)

print(f"Параметр сдвига по методу медианы: {theta\_by\_median}")

# %% [markdown]

# ### 3.3. Усечённое среднее

#

# Для оценки параметра сдвига при помощи усечённого среднего можно воспользоваться `scipy.stats.trim\_mean(array, alpha: float) -> float`. Необходимо будет также сделать 3 таких оценки с коэффициентами $0.05$, $0.10$, $0.15$.

# %%

def theta\_approximation\_by\_trim\_mean(data: np.ndarray, alpha: float) -> float:

    """

    Функция оценки параметра сдвига при помощи усечённого среднего

    Args:

        data (np.ndarray): последовательность, для которой необходимо выполнить оценку

        alpha (float): процент усечения диапазона (от 0.0 до 0.5)

    Returns:

        float: значение оценки параметра

    """

    return st.trim\_mean(data, alpha)

theta\_by\_trim\_mean\_05 = theta\_approximation\_by\_trim\_mean(scattened, 0.05)

theta\_by\_trim\_mean\_10 = theta\_approximation\_by\_trim\_mean(scattened, 0.10)

theta\_by\_trim\_mean\_15 = theta\_approximation\_by\_trim\_mean(scattened, 0.15)

print(f"Параметр сдвига по методу усечённого среднего (alpha = 0.05): {theta\_by\_trim\_mean\_05}")

print(f"Параметр сдвига по методу усечённого среднего (alpha = 0.10): {theta\_by\_trim\_mean\_10}")

print(f"Параметр сдвига по методу усечённого среднего (alpha = 0.15): {theta\_by\_trim\_mean\_15}")

# %% [markdown]

# ### 3.4. Метод максимального правдоподобия

#

# Для оценки максимального правдоподобия необходимо провести минимизацию функционала вида

#

# $$

# Q(\theta) = \frac{1}{N} \sum\limits\_{i=1}^{N} -\ln f\left(\frac{y-\theta}{\lambda}\right),

# $$

#

# где $f$ – функция плотности распределения Лапласа без смещений и прочего.

# %%

def theta\_approximation\_by\_MLE(data: np.ndarray) -> float:

    """

    Функция оценки параметра сдвига при помощи метода максимального правдоподобия

    Args:

        data (np.ndarray): последовательность, для которой необходимо выполнить оценку

    Returns:

        float: значение оценки параметра

    """

    f = lambda O, x: np.sum(-np.log(f\_laplas(x - O)))

    return minimize(f, x0=0, args=(data), tol=1e-6).x[0]

theta\_by\_MLE = theta\_approximation\_by\_MLE(scattened)

print(f"Параметр сдвига по методу максимального правдоподобия: {theta\_by\_MLE}")

# %% [markdown]

# ### 3.5. Метод обобщённых радикальных оценок

#

# Обобщённые радикальные оценки задаются следующей функцией потерь:

#

# $$

# \rho(y, \theta) = - \frac{1}{f^{\delta}(0)}f^{\delta}\left( \frac{y-\theta}{\lambda} \right),

# $$

#

# где $\delta > 0$ – параметр, регулирующий степень робастности оценки.

# %%

def theta\_approximation\_by\_MGD(data: np.ndarray, delta: float) -> float:

    """

    Функция оценки параметра сдвига при помощи метода обобщённых радикальных оценок

    Args:

        data (np.ndarray): последовательность, для которой необходимо выполнить оценку

        delta (float): параметр робастности оценки

    Returns:

        float: значение оценки параметра

    """

    f0 = 15/96

    f = lambda O, x: np.sum(-1 / (f0\*\*delta) \* f\_laplas(x - O)\*\*delta)

    return minimize(f, x0=0, args=(data), tol=1e-6).x[0]

theta\_by\_MGD\_01 = theta\_approximation\_by\_MGD(scattened, 0.1)

theta\_by\_MGD\_05 = theta\_approximation\_by\_MGD(scattened, 0.5)

theta\_by\_MGD\_10 = theta\_approximation\_by\_MGD(scattened, 1.0)

print(f"Параметр сдвига по методу обобщённых радикальных оценок (параметр = 0.1): {theta\_by\_MGD\_01}")

print(f"Параметр сдвига по методу обобщённых радикальных оценок (параметр = 0.5): {theta\_by\_MGD\_05}")

print(f"Параметр сдвига по методу обобщённых радикальных оценок (параметр = 1.0): {theta\_by\_MGD\_10}")

# %% [markdown]

# ### 3.6. Все оценки в общем виде

# %%

results = [

    [

        "Среднее арифметическое", theta\_approximation\_by\_mean(scattened),

    ],

    [

        "Медиана", theta\_approximation\_by\_median(scattened),

    ],

    [

        "Усечённое среднее (0.05)", theta\_approximation\_by\_trim\_mean(scattened, 0.05),

    ],

    [

        "Усечённое среднее (0.10)", theta\_approximation\_by\_trim\_mean(scattened, 0.10),

    ],

    [

        "Усечённое среднее (0.15)", theta\_approximation\_by\_trim\_mean(scattened, 0.15),

    ],

    [

        "Максимальное правдоподобие", theta\_approximation\_by\_MLE(scattened),

    ],

    [

        "Обобщённые рад. оценки (0.1)", theta\_approximation\_by\_MGD(scattened, 0.1),

    ],

    [

        "Обобщённые рад. оценки (0.5)", theta\_approximation\_by\_MGD(scattened, 0.5),

    ],

    [

        "Обобщённые рад. оценки (1.0)", theta\_approximation\_by\_MGD(scattened, 1.0),

    ],

]

pd.DataFrame(results, columns=["Оценка", "Значение"])

# %% [markdown]

# ## 4 Графики функций влияния

# %% [markdown]

# ### 4.1. График среднего арифметического, медианы

#

# Функция влияния среднего арифметического выражается следующей формулой:

#

# $$

# \text{IF}(y) = y-\theta

# $$

#

# Для метода медианы, функция влияния следующая:

#

# $$

# \text{IF}(y) = \frac{\lambda\cdot \text{sign}(y - \theta)}{2\cdot f(0)}

# $$

#

# Учитывая, что $f(0) = \frac{5}{32}$, получаем:

#

# $$

# \text{IF}(y) = \frac{\lambda\cdot \text{sign}(y - \theta) \cdot 16}{5}

# $$

# %%

def IF\_mean(data: np.ndarray, theta: float) -> np.ndarray:

    """

    Функция влиялия для среднего арифметического

    Args:

        data (np.ndarray): значения X распределения

        theta (float): параметр тетта смещения

    Returns:

        np.ndarray: значения функции влияния

    """

    return data - theta

def IF\_median(data: np.ndarray, theta: float, lamb: float) -> np.ndarray:

    """

    Функция влиялия для медианы

    Args:

        data (np.ndarray): значения X распределения

        theta (float): параметр тетта смещения

        lamd (float): параметр лямбда распределения

    Returns:

        np.ndarray: значения функции влияния

    """

    return 16 \* lamb \* np.sign(data - theta) / 5

x = np.linspace(-5, 5, 500)

fig, axes = plt.subplots()

axes.plot(x, IF\_mean(x, 0), label="Среднее")

axes.plot(x, IF\_median(x, 0, 1), label="Медиана")

axes.legend()

plt.show()

# %% [markdown]

# ### 4.2. Графики для усечённого среднего

#

# Функция влияния усечённого среднего имеет следующий вид:

#

# $$

# \large

# \text{IF}(y) = \frac{1}{1 - 2\alpha} \begin{cases}

# -q,~\frac{y - \theta}{\lambda} < -q \\

# \frac{y - \theta}{\lambda}, \left| \frac{y - \theta}{\lambda} \right| \leqslant q, \\

# q, \frac{y - \theta}{\lambda} > q,

# \end{cases}

# $$

#

# где $q$ – квантиль порядка $1 - \alpha$ стандартного распределения, то есть,

#

# $$

# \int\_{-\infty}^{q} f(y) dy = 1 - \alpha,

# $$

#

# а $\alpha$ – уровень усечения ($\alpha = \{0.05, 0.10, 0.15\}$).

# %%

def IF\_trim\_mean(data: np.ndarray, alpha: float, theta: float, lamb: float) -> np.ndarray:

    """

    Функция влиялия для усечённого среднего

    Args:

        data (np.ndarray): значения X распределения

        alpha (float): параметр усечения (в диапазоне 0 <= alpha < 0.5)

        theta (float): параметр тетта смещения

        lamd (float): параметр лямбда распределения

    Returns:

        np.ndarray: значения функции влияния

    """

    # Определяем параметр q

    right = 1 - alpha

    fn = lambda x: np.exp(-1 \* np.abs(x)) \* (np.abs(x)\*\*3 + 6 \* np.abs(x)\*\*2 + 15\*np.abs(x) + 15) / 96

    fn\_m = lambda q: np.abs(right - quad(fn, -np.inf, q)[0])

    q = minimize(fn\_m, x0=2).x[0]

    # Считаем

    left = 1 / (1 - 2 \* alpha) # Базовый множитель

    x = (data - theta) / lamb # преобразованная переменная y (массив)

    res = np.ndarray(x.shape) # Результат

    for i in range(x.size):

        if x[i] < -q:

            res[i] = -q

        elif x[i] > q:

            res[i] = q

        else:

            res[i] = x[i]

    return left \* res

x = np.linspace(-5, 5, 500)

fig, axes = plt.subplots()

axes.plot(x, IF\_trim\_mean(x, 0.05, 0, 1), label="Усечённое среднее (A = 0.05)")

axes.plot(x, IF\_trim\_mean(x, 0.1, 0, 1), label="Усечённое среднее (A = 0.10)")

axes.plot(x, IF\_trim\_mean(x, 0.15, 0, 1), label="Усечённое среднее (A = 0.15)")

axes.legend()

plt.show()

# %% [markdown]

# ### 4.3. Графики для метода максимального правдоподобия

#

# Для максимального правдоподобия функция влияния задаётся следующим образом:

#

# $$

# \text{IF}(y) = \frac{-\lambda f'(\frac{y-\theta}{\lambda})/f(\frac{y - \theta}{\lambda})}{\int\_{-\infty}^{+\infty}[f'(z)]^{2}/f(z)~dz}

# $$

#

# Производная функции $f$ равна

#

# $$

# \frac{df}{dx} = -xe^{-|x|}\frac{\left( |x|^2 + 3|x| + 3 \right) }{96}.

# $$

#

# %%

def IF\_MLE(data: np.ndarray, theta: float, lamb: float) -> np.ndarray:

    """

    Функция влиялия для метода максимального правдоподобия

    Args:

        data (np.ndarray): значения X распределения

        theta (float): параметр тетта смещения

        lamd (float): параметр лямбда распределения

    Returns:

        np.ndarray: значения функции влияния

    """

    fn = lambda x: np.exp(-1 \* np.abs(x)) \* (np.abs(x)\*\*3 + 6 \* np.abs(x)\*\*2 + 15\*np.abs(x) + 15) / 96

    dfn = lambda x: -1 \* x \* np.exp(-1 \* np.abs(x)) \* (np.abs(x)\*\*2 + 3\*np.abs(x) + 3) / 96

    fndfn = lambda x: dfn(x)\*\*2 / fn(x)

    denumenator = quad(fndfn, -700, 700)[0] # По идее надо -np.inf; np.inf, но там всё ломается

    x = (data - theta) / lamb

    return -lamb / denumenator \* dfn(x) / fn(x)

x = np.linspace(-5, 5, 500)

fig, axes = plt.subplots()

axes.plot(x, IF\_MLE(x, 0, 1), label="Макс правдоподобие")

axes.legend()

plt.show()

# %% [markdown]

# ### 4.4. Графики для метода обобщённых радикальных оценок

#

# Для обобщённых радикальных оценок функция влияния задаётся следующим образом:

#

# $$

# \text{IF}(y) = \frac{-\lambda f'(\frac{y-\theta}{\lambda})f^{\delta - 1}(\frac{y - \theta}{\lambda})}{\int\_{-\infty}^{+\infty}[f'(z)]^{2} f^{\delta - 1}(z)~dz}

# $$

#

# Производная функции $f$ равна

#

# $$

# \frac{df}{dx} = -xe^{-|x|}\frac{\left( |x|^2 + 3|x| + 3 \right) }{96}.

# $$

# %%

def IF\_MGD(data: np.ndarray, delta: float, theta: float, lamb: float) -> np.ndarray:

    """

    Функция влиялия для метода обобщённых радикальных оценок

    Args:

        data (np.ndarray): значения X распределения

        delta (float): параметр регулировки степени робастности оценки

        theta (float): параметр тетта смещения

        lamd (float): параметр лямбда распределения

    Returns:

        np.ndarray: значения функции влияния

    """

    fn = lambda x: np.exp(-1 \* np.abs(x)) \* (np.abs(x)\*\*3 + 6 \* np.abs(x)\*\*2 + 15\*np.abs(x) + 15) / 96

    dfn = lambda x: -1 \* x \* np.exp(-1 \* np.abs(x)) \* (np.abs(x)\*\*2 + 3\*np.abs(x) + 3) / 96

    fndfn = lambda x: dfn(x)\*\*2 \* fn(x)\*\*(delta - 1)

    denumenator = quad(fndfn, -700, 700)[0] # По идее надо -np.inf; np.inf, но там всё ломается

    x = (data - theta) / lamb

    return -lamb / denumenator \* dfn(x) \* fn(x)\*\*(delta - 1)

x = np.linspace(-15, 15, 500)

fig, axes = plt.subplots()

axes.plot(x, IF\_MGD(x, 0.1, 0, 1), label="Рад. оценки (d = 0.1)")

axes.plot(x, IF\_MGD(x, 0.5, 0, 1), label="Рад. оценки (d = 0.5)")

axes.plot(x, IF\_MGD(x, 1.0, 0, 1), label="Рад. оценки (d = 1.0)")

axes.legend()

plt.show()

# %% [markdown]

# ### 4.5. Графики функций влияния всех оценок

# %%

x = np.linspace(-15, 15, 500)

fig, axes = plt.subplots(figsize=(25,15))

axes.plot(x, IF\_mean(x, 0), label="Среднее")

axes.plot(x, IF\_median(x, 0, 1), label="Медиана")

axes.plot(x, IF\_trim\_mean(x, 0.05, 0, 1), label="Усечённое среднее (A = 0.05)")

axes.plot(x, IF\_trim\_mean(x, 0.1, 0, 1), label="Усечённое среднее (A = 0.10)")

axes.plot(x, IF\_trim\_mean(x, 0.15, 0, 1), label="Усечённое среднее (A = 0.15)")

axes.plot(x, IF\_MLE(x, 0, 1), label="Макс правдоподобие")

axes.plot(x, IF\_MGD(x, 0.1, 0, 1), label="Рад. оценки (d = 0.1)")

axes.plot(x, IF\_MGD(x, 0.5, 0, 1), label="Рад. оценки (d = 0.5)")

axes.plot(x, IF\_MGD(x, 1.0, 0, 1), label="Рад. оценки (d = 1.0)")

axes.legend()

plt.show()