Курсовая работа по курсу математической статистики

Сааков А.С. СКБ182
Версия от 03.11.2020

Содержание

1. Домашнее задание. Вероятностные распределения

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import maxwell
import scipy.stats as sts
from scipy.stats import geom
from random import random
from collections import Counter
import copy
import math
from math import *
from random import *
import pandas as pd
import calendar
import statsmodels.api as sm

plt.style.use('ggplot')  # Красивые графики
plt.rcParams['figure.figsize'] = (15, 5)  # Размер картинок
```

1.1. Геометрическое распределение

1.1.1. Описание основных характеристик распределения

Функция вероятности дискретного распределения: $P_{\xi}(x) = pq^x, x \in \{0, 1, 2, \dots\}$
Математическое ожидание:

$$M\xi = \sum_{k=1}^{\infty} kpq^{k-1} = p\sum_{k=1}^{\infty} kq^{k-1} = p\sum_{k=1}^{\infty} \frac{dq^k}{dq} = p\frac{d}{dq}(\sum_{k=1}^{\infty} q^k) = p\frac{d}{dq}(\frac{q}{1-q}) = p\frac{1}{(1-q)^2} = \frac{1}{p}$$

Дисперсия:

$$D\xi = M(\xi - M\xi)^2 = M\xi^2 - (M\xi)^2 = M(\xi(\xi-1) + \xi) - M\xi^2 = M(\xi(\xi-1)) + M\xi - (M\xi)^2 = M(\xi(\xi-1)) + M\xi(1-\xi)$$

$$M(\xi(\xi-1)) = p\sum_{k=1}^{\infty} k^2q^{k-1} = pq\sum_{k=0}^{\infty} \frac{d^2q^k}{dq^2} = pq\frac{d^2}{dq^2}(\sum_{k=0}^{\infty} q^k) = pq\frac{d^2}{dq^2}(\frac{1}{1-q}) = pq\frac{2}{(1-q)^3} = \frac{2q}{p^2}$$

$$D\xi = M\xi^2 + M\xi - (M\xi)^2 = \frac{2q}{p^2} + \frac{1}{p} - \frac{1}{p^2} = \frac{2q-1}{p^2} + \frac{1}{p} = \frac{2q-1+p}{p^2} = \frac{2q-1+1-q}{p^2} = \frac{q}{p^2}$$

In [2]:

```
for p in [0.1, 0.4, 0.6, 0.9]:
    geom_rv = sts.geom(p)
    sample = geom_rv.rvs(1000)
    plt.hist(sample, density = True, label='p = {}'.format(p))
    plt.legend()
    plt.show()
print('Рис. 1: 1.1.1, Гистограмма вероятностей дискретного распределения')
```

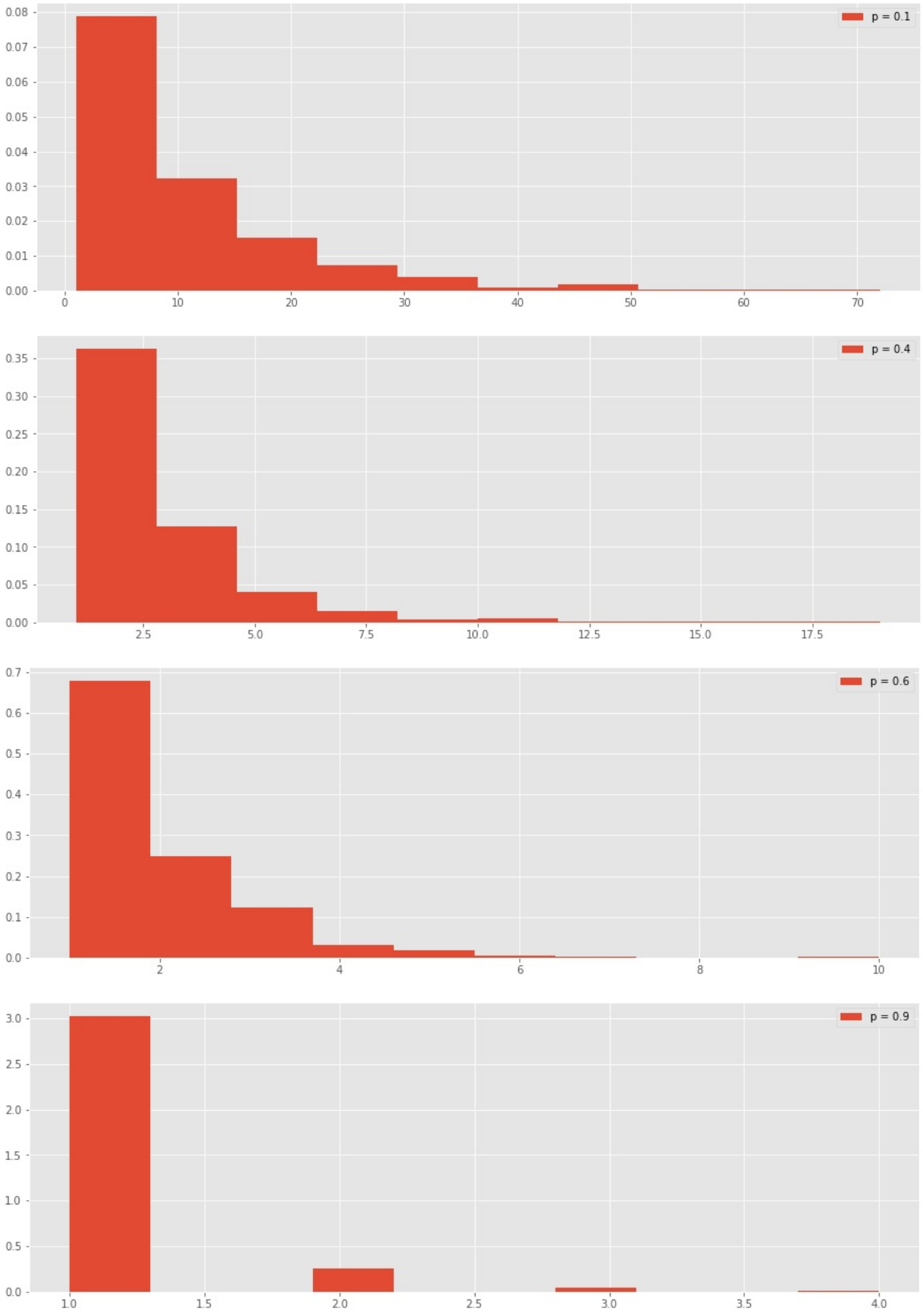


Рис. 1: 1.1.1, Гистограмма вероятностей дискретного распределения

Мода M_0 - значение во множестве наблюдений, которое встречается наиболее часто, для дискретной случайной величины определяется с помощью гистограммы вероятностей.
Из гистограмм видно, что $M_0 = 1$

In [3]:

```
for p in [0.1, 0.4, 0.6, 0.9]:
    n = np.arange(0, 8, 1)
    plt.step(n, 1-(1-p)**(n+1), label='p = {}'.format(p))
    plt.legend()
plt.show()
print('Рис. 1: 1.1.1, Гистограмма вероятностей дискретного распределения')
```

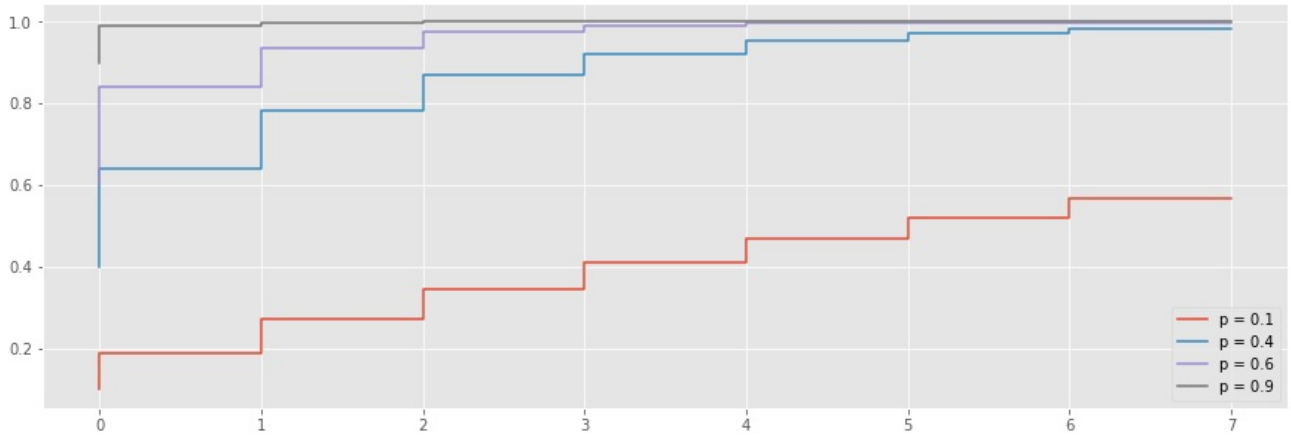


Рис. 1: 1.1.1, Гистограмма вероятностей дискретного распределения

Медиана Me находится из уравнения $P_\xi(x) = 0.5$

$$\begin{cases} p + qp + q^2p + \ldots + q^{Me-1}p \geq \frac{1}{2} \\ q^{Me-1}p + q^{Me}p + q^{Me+1}p + \ldots \geq \frac{1}{2} \end{cases}$$

$$\begin{cases} p\frac{1-q^{Me}}{1-q} \geq \frac{1}{2} \\ q^{Me-1}p\frac{1}{1-q} \geq \frac{1}{2} \end{cases}$$

$$\begin{cases} 1 - q^{Me} \geq 2^{-1} \\ q^{Me-1} \geq 2^{-1} \end{cases}$$

$$\begin{cases} q^{Me} \leq 2^{-1} \\ q^{Me-1} \geq 2^{-1} \end{cases}$$

$$\begin{cases} Me \cdot log_2 q \leq -1 \\ (Me-1) log_2 q \geq -1 \end{cases}$$

Отсюда $-\frac{1}{log_2 q} \leq Me \leq 1 - \frac{1}{log_2 q}$

Примеры событий, которые могут быть описаны выбранными случайными величинами

Типичные интерпретации геометрического распределения: описывает количество испытаний n до первого успеха при вероятности наступления успеха в каждом испытании p. Если n подразумевается номер испытания, в котором наступил успех, то геометрическое распределение будет описываться следующей формулой:

$$Geom_p(n) = q^{n-1}p$$

Геометрическое распределение считается дискретной версией экспоненциального распределения.
Предположим, что эксперименты Бернулли проводятся через равные промежутки времени. Тогда геометрическая случайная величина X - это время, измеренное в дискретных единицах, которое проходит до того, как мы добьемся первого успеха. . Но если мы хотим смоделироватьвремя, прошедшее до того, как данное событие произойдет в непрерывном времени, то подходящим распределением для использования будетэкспоненциальное распределение. С математической точки зрения геометрическое распределение обладает тем же свойством без памяти,которым обладает экспоненциальное распределение: в экспоненциальном случае вероятность того, что событие произойдет в течениезаданного временного интервала, не зависит от того, сколько времени уже прошло, а событие не произошло; в геометрическом случаевероятность того, что событие произойдет в данный момент (дискретное) времени, не зависит от того, что произошло раньше, потому чтоэксперимент Бернулли, проведенный в каждый момент времени, не зависит от предыдущих испытаний.Геометрическое распределение полезно для определения вероятности успеха при ограниченном количестве испытаний, что очень применимо креальному миру, в котором неограниченные испытания редки. Поэтому неудивительно, что различные сценарии хорошо моделируютсягеометрическими распределениями:

- В спорте, особенно в бейсболе, геометрическое распределение полезно для анализа вероятности того, что отбивающий получит удар,прежде чем он получит три удара; здесь цель - добиться успеха за 3 испытания.
- При анализе затрат и выгод, например, когда компания решает, финансировать ли исследовательские испытания, которые в случае успехапринесут компании некоторую предполагаемую прибыль, цель состоит в том, чтобы достичь успеха до того, как затраты превысятпотенциальную выгоду.
- В тайм-менеджменте цель состоит в том, чтобы выполнить задачу за установленный промежуток времени.Другие приложения, подобные вышеупомянутым, также легко создаются. Фактически, геометрическое распределение применяется наинтуитивном уровне в повседневной жизни на регулярной основе.

1.1.3 Описание способа моделирования выбранных случайных величин

Существует такой способ реализации метода обратных функций, при котором трудоемкость по крайней мере формально не зависит от p. Действительно, накопленная вероятность $s_{n+1} = p_0 + \ldots + p_n$ для геометрического распределения имеет вид

$$s_{n+1} = \sum_{i=0}^{n} p(1-p)^i = 1 - (1-p)^{n+1}$$

Поэтому событие $\{\xi = n\}$ приобретает вид

$$\{\xi = n\} = \{s_n < \alpha \le s_{n+1}\} = \{1-(1-p)^n < \alpha \le 1-(1-p)^{n+1}\} = \{(1-p)^{n+1} \le 1-\alpha < (1-p)^n\} = \{(n+1)ln(1-p) \le ln(1-\alpha) < n \cdot ln(1-p)\} = \{n < \frac{ln(1-\alpha)}{ln(1-p)} \le$$

и тем самым

$$\xi = [\frac{ln(1-\alpha)}{ln(1-p)}]$$

Эту же формулу можно получить по-другому. Пусть v - случайная величина, имеющая показательное распределение с параметром λ и $\xi = [n]$. Тогда при $n \ge 0$

$$P(\xi = n) = P(n \le v < n+1) = e^{-n\lambda} - e^{-(n+1)\lambda} = (1-e^{-\lambda})e^{-n\lambda}.$$

Поскольку случайная величина $\frac{-ln(1-\alpha)}{\lambda}$ имеет показательное распределение с параметров λ, то взяв $\lambda = -ln(1-p)$, приходим к формуле $\xi = [\frac{ln(1-\alpha}{$

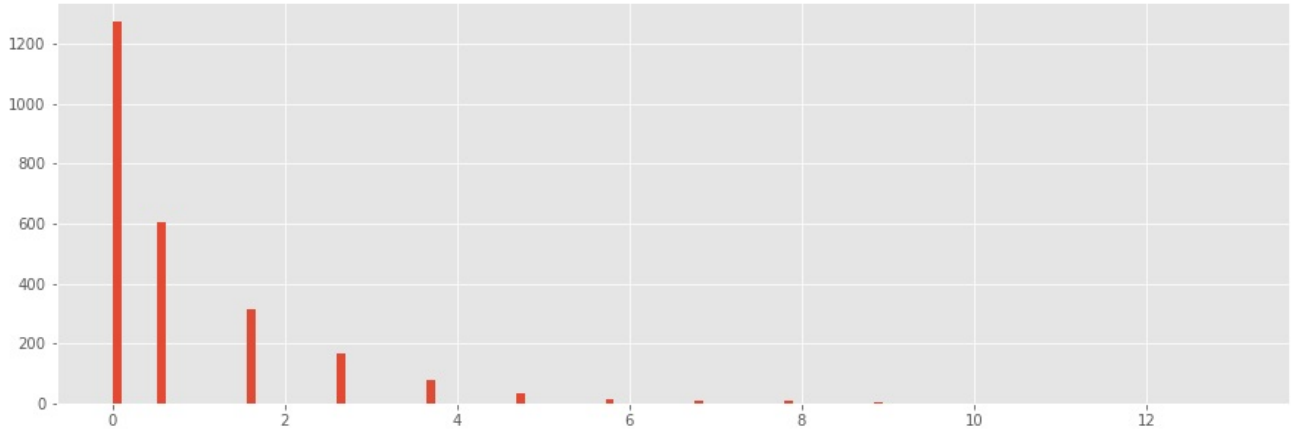
In [4]:

```
def sample_(N=2500, scale = 0.5):
    for x in range(N):
        je = np.log(random())//np.log(1-scale)#Генерирование случайных чисел по формуле из справочника
    return je
def Geom(n, p=0.5):
    x=[sample_(scale=p) for x in range(n)]
    print(x)
    return x
plt.hist(Geom(2500,0.5),25, width = 0.1)
plt.show()
```

```
[4.0, 0.0, 2.0, 3.0, 0.0, 0.0, 1.0, 5.0, 1.0, 0.0, 2.0, 0.0, 3.0, 1.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 2.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 2.0, 2.0, 0.0, 0.0, 0.0, 1.0, 1.0,
1.0, 0.0, 2.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 4.0, 1.0, 0.0, 0.0, 0.0, 3.0, 3.0,
4.0, 0.0, 2.0, 0.0, 4.0, 1.0, 3.0, 0.0, 2.0, 3.0, 1.0, 0.0, 6.0, 0.0, 1.0, 0.0, 1.0, 1.0, 3.0, 0.0,
1.0, 2.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 5.0, 0.0, 0.0, 4.0, 0.0, 0.0, 0.0, 1.0, 0.0, 2.0,
0.0, 1.0, 1.0, 0.0, 0.0, 3.0, 4.0, 2.0, 0.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.0, 1.0, 2.0, 0.0, 0.0, 3.0,
```

0.0, 3.0, 2.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 2.0, 1.0, 1.0, 0.0, 0.0, 1.0, 2.0, 2.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0, 0.0, 0.0, 2.0, 6.0, 2.0, 3.0, 0.0, 0.0, 7.0, 0.0, 2.0, 0.0, 0.0, 1.0, 3.0, 1.0, 0.0,
0.0, 3.0, 0.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0, 2.0, 0.0, 0.0, 3.0, 2.0, 0.0,
1.0, 4.0, 0.0, 0.0, 3.0, 0.0, 0.0, 3.0, 1.0, 3.0, 0.0, 1.0, 0.0, 0.0, 0.0, 4.0, 1.0, 0.0, 2.0, 1.0,
0.0, 6.0, 0.0, 0.0, 1.0, 3.0, 1.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 2.0,
1.0, 0.0, 2.0, 1.0, 0.0, 0.0, 2.0, 1.0, 1.0, 0.0, 3.0, 0.0, 3.0, 0.0, 0.0, 1.0, 2.0, 1.0, 0.0, 3.0,
3.0, 1.0, 1.0, 0.0, 3.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 2.0, 0.0, 1.0, 0.0, 0.0, 0.0, 5.0,
1.0, 1.0, 0.0, 6.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 8.0, 0.0,
0.0, 2.0, 0.0, 2.0, 0.0, 6.0, 1.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.0, 1.0, 4.0, 0.0, 2.0, 4.0, 3.0, 2.0, 1.0, 2.0, 0.0, 0.0, 0.0, 3.0,
0.0, 0.0, 0.0, 0.0, 0.0, 3.0, 2.0, 0.0, 1.0, 0.0, 0.0, 0.0, 3.0, 0.0, 0.0, 1.0, 5.0, 2.0, 0.0, 0.0,
4.0, 3.0, 0.0, 0.0, 1.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 3.0, 0.0, 1.0, 1.0, 1.0, 5.0, 0.0, 1.0, 0.0,
13.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 3.0, 2.0, 1.0, 2.0, 0.0, 2.0, 0.0, 2.0, 1.0, 1.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 1.0, 2.0, 0.0, 1.0, 0.0, 2.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 3.0, 1.0,
2.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 3.0, 2.0, 1.0, 0.0, 2.0, 0.0, 0.0, 5.0,
0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 1.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0, 1.0, 0.0, 0.0, 0.0, 4.0, 1.0, 1.0,
0.0, 0.0, 1.0, 0.0, 2.0, 1.0, 0.0, 0.0, 0.0, 2.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.0, 1.0, 0.0, 4.0, 2.0,
0.0, 0.0, 0.0, 1.0, 1.0, 5.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0, 1.0, 0.0, 3.0, 0.0, 0.0, 5.0, 0.0, 1.0,
2.0, 0.0, 0.0, 4.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 2.0, 3.0, 0.0,
0.0, 6.0, 0.0, 0.0, 0.0, 0.0, 4.0, 1.0, 0.0, 2.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 5.0, 3.0, 1.0, 0.0,
0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 3.0, 0.0, 0.0, 4.0, 0.0, 0.0, 3.0, 1.0, 7.0, 0.0, 1.0, 5.0, 1.0,
3.0, 1.0, 0.0, 4.0, 3.0, 0.0, 0.0, 2.0, 0.0, 2.0, 5.0, 0.0, 0.0, 2.0, 0.0, 0.0, 1.0, 1.0, 2.0, 0.0,
0.0, 0.0, 1.0, 0.0, 0.0, 3.0, 0.0, 0.0, 3.0, 2.0, 1.0, 4.0, 2.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 2.0, 1.0, 1.0, 0.0, 0.0, 0.0, 2.0, 1.0, 0.0, 2.0, 1.0, 2.0, 2.0, 1.0, 1.0, 0.0,
2.0, 1.0, 0.0, 3.0, 2.0, 0.0, 0.0, 4.0, 0.0, 3.0, 1.0, 1.0, 0.0, 0.0, 2.0, 1.0, 1.0, 3.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 2.0, 0.0, 0.0, 1.0, 4.0, 3.0, 0.0, 6.0, 0.0, 1.0, 0.0, 1.0, 1.0,
0.0, 1.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 2.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0,
0.0, 2.0, 0.0, 0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 3.0, 1.0, 3.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.0,
0.0, 2.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 4.0, 7.0, 1.0, 2.0, 4.0, 4.0, 0.0, 1.0, 0.0, 0.0, 0.0, 2.0,
1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 2.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 2.0,
0.0, 0.0, 0.0, 1.0, 3.0, 2.0, 2.0, 3.0, 3.0, 0.0, 0.0, 2.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0,
0.0, 0.0, 0.0, 7.0, 0.0, 1.0, 0.0, 0.0, 2.0, 2.0, 0.0, 0.0, 4.0, 3.0, 0.0, 0.0, 1.0, 0.0, 2.0, 0.0,
3.0, 2.0, 5.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 3.0, 0.0, 2.0,
0.0, 0.0, 1.0, 0.0, 2.0, 0.0, 4.0, 2.0, 1.0, 0.0, 2.0, 6.0, 5.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0,
2.0, 1.0, 7.0, 8.0, 6.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 3.0, 1.0, 0.0, 0.0, 2.0, 3.0, 0.0, 2.0,
1.0, 0.0, 2.0, 0.0, 1.0, 0.0, 0.0, 0.0, 3.0, 0.0, 0.0, 1.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0,
2.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0, 3.0, 0.0, 1.0, 1.0, 2.0, 1.0,
0.0, 0.0, 1.0, 3.0, 2.0, 0.0, 1.0, 0.0, 3.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 3.0, 0.0, 0.0, 1.0,
0.0, 0.0, 0.0, 0.0, 1.0, 2.0, 0.0, 0.0, 4.0, 0.0, 2.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 5.0, 1.0, 5.0, 1.0, 0.0, 1.0, 1.0, 5.0, 1.0, 0.0, 4.0, 1.0, 3.0, 1.0, 0.0,
1.0, 2.0, 0.0, 0.0, 0.0, 1.0, 4.0, 2.0, 2.0, 0.0, 1.0, 1.0, 1.0, 1.0, 4.0, 0.0, 1.0, 0.0, 0.0, 1.0,
1.0, 5.0, 0.0, 3.0, 0.0, 1.0, 0.0, 2.0, 3.0, 0.0, 0.0, 1.0, 2.0, 0.0, 0.0, 4.0, 0.0, 0.0, 0.0, 6.0,
0.0, 1.0, 0.0, 0.0, 1.0, 3.0, 2.0, 1.0, 4.0, 1.0, 0.0, 4.0, 5.0, 0.0, 0.0, 2.0, 1.0, 0.0, 0.0, 4.0,
1.0, 1.0, 0.0, 1.0, 2.0, 0.0, 1.0, 5.0, 1.0, 2.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 3.0, 0.0, 0.0, 8.0,
0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 4.0, 6.0, 2.0, 0.0, 2.0, 0.0, 0.0, 1.0, 4.0, 0.0, 0.0, 0.0, 0.0, 1.0,
0.0, 1.0, 1.0, 2.0, 0.0, 0.0, 0.0, 2.0, 1.0, 5.0, 1.0, 1.0, 2.0, 0.0, 1.0, 0.0, 0.0, 0.0, 5.0, 2.0,
1.0, 1.0, 0.0, 3.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0,
2.0, 0.0, 0.0, 2.0, 1.0, 1.0, 0.0, 0.0, 2.0, 0.0, 1.0, 0.0, 2.0, 0.0, 8.0, 1.0, 0.0, 0.0, 1.0, 1.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 3.0, 0.0, 1.0, 1.0, 2.0, 0.0, 0.0, 1.0, 1.0, 0.0, 3.0, 1.0, 2.0, 0.0,
1.0, 1.0, 0.0, 0.0, 2.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 2.0, 0.0, 0.0, 1.0, 2.0, 4.0, 0.0, 2.0,
0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 2.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 4.0, 1.0, 0.0,
0.0, 0.0, 0.0, 1.0, 2.0, 4.0, 8.0, 3.0, 0.0, 2.0, 1.0, 2.0, 0.0, 1.0, 0.0, 0.0, 0.0, 3.0, 0.0, 3.0,
0.0, 2.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 2.0, 1.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0, 2.0,
4.0, 1.0, 4.0, 3.0, 0.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0,
2.0, 3.0, 1.0, 0.0, 5.0, 0.0, 0.0, 4.0, 0.0, 1.0, 2.0, 0.0, 2.0, 0.0, 1.0, 2.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 2.0, 1.0, 2.0, 3.0, 0.0, 3.0, 1.0, 0.0,
1.0, 0.0, 4.0, 0.0, 0.0, 0.0, 1.0, 3.0, 0.0, 0.0, 0.0, 4.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 9.0, 1.0,
6.0, 1.0, 2.0, 2.0, 3.0, 1.0, 0.0, 1.0, 3.0, 1.0, 2.0, 1.0, 1.0, 4.0, 1.0, 2.0, 1.0, 2.0, 1.0, 3.0,
1.0, 0.0, 0.0, 3.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 3.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0,
2.0, 0.0, 0.0, 0.0, 0.0, 3.0, 0.0, 3.0, 1.0, 3.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 3.0, 0.0, 0.0, 0.0, 2.0, 0.0, 4.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 2.0, 0.0, 1.0, 1.0, 0.0, 0.0,
2.0, 1.0, 0.0, 2.0, 3.0, 2.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 3.0,
1.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 3.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 2.0, 2.0, 1.0, 0.0,
0.0, 2.0, 2.0, 2.0, 0.0, 4.0, 2.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0,
1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 2.0, 1.0, 0.0, 1.0, 2.0, 2.0, 0.0, 1.0, 4.0, 1.0, 1.0, 0.0, 0.0, 0.0,
6.0, 1.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 1.0, 2.0, 0.0, 1.0, 1.0, 2.0, 2.0, 2.0, 0.0, 0.0, 1.0, 0.0,
1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.0, 0.0, 0.0, 1.0, 0.0, 2.0, 0.0, 1.0, 6.0, 0.0, 5.0,
0.0, 3.0, 0.0, 0.0, 1.0, 3.0, 0.0, 0.0, 1.0, 2.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 3.0, 0.0,
4.0, 0.0, 1.0, 2.0, 2.0, 0.0, 1.0, 0.0, 5.0, 3.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 2.0,
1.0, 5.0, 1.0, 2.0, 1.0, 3.0, 2.0, 0.0, 1.0, 1.0, 3.0, 3.0, 0.0, 0.0, 1.0, 2.0, 1.0, 1.0, 0.0, 0.0,
1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 2.0, 2.0, 0.0, 3.0, 1.0, 3.0, 0.0, 0.0, 0.0, 0.0, 2.0, 1.0, 2.0, 1.0,
0.0, 0.0, 1.0, 0.0, 7.0, 2.0, 4.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 3.0, 2.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 2.0, 0.0, 1.0, 0.0, 2.0, 0.0, 0.0, 2.0, 2.0, 3.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 2.0, 2.0, 0.0, 2.0,
2.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 4.0, 1.0, 2.0, 1.0, 0.0, 0.0, 2.0, 0.0, 1.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.0,
2.0, 0.0, 0.0, 1.0, 0.0, 1.0, 2.0, 3.0, 2.0, 0.0, 1.0, 4.0, 2.0, 0.0, 1.0, 1.0, 3.0, 0.0, 1.0, 0.0,
1.0, 0.0, 0.0, 2.0, 3.0, 0.0, 1.0, 3.0, 2.0, 1.0, 1.0, 0.0, 1.0, 3.0, 0.0, 2.0, 1.0, 0.0, 2.0, 3.0,
3.0, 5.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 4.0, 0.0, 3.0, 5.0, 1.0, 1.0, 0.0, 1.0, 2.0, 2.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 4.0, 1.0, 0.0, 1.0, 2.0, 2.0,
2.0, 1.0, 2.0, 0.0, 1.0, 0.0, 1.0, 1.0, 2.0, 5.0, 1.0, 3.0, 0.0, 3.0, 0.0, 0.0, 1.0, 1.0, 1.0, 4.0,
1.0, 1.0, 0.0, 2.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 2.0, 0.0, 2.0, 0.0, 3.0, 1.0, 0.0, 1.0, 1.0, 3.0,

```
0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 4.0, 3.0, 0.0, 9.0, 0.0, 2.0, 0.0, 2.0, 2.0, 1.0, 3.0, 1.0, 0.0, 2.0,
0.0, 4.0, 0.0, 3.0, 0.0, 0.0, 0.0, 2.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 2.0, 3.0, 0.0, 0.0, 1.0, 2.0,
1.0, 1.0, 0.0, 2.0, 3.0, 1.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.0, 0.0, 1.0, 2.0, 0.0, 2.0, 0.0, 1.0, 1.0,
6.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 3.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 2.0, 1.0, 0.0, 0.0, 4.0,
0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 3.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 1.0,
1.0, 2.0, 0.0, 0.0, 0.0, 2.0, 1.0, 0.0, 3.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 2.0, 3.0, 1.0, 2.0,
0.0, 0.0, 0.0, 3.0, 4.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 8.0, 1.0, 3.0, 1.0, 0.0, 1.0,
3.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 3.0, 0.0, 2.0, 0.0,
1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 5.0, 2.0, 0.0, 0.0, 0.0,
3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 5.0, 2.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 2.0, 0.0, 1.0, 0.0,
4.0, 2.0, 0.0, 1.0, 0.0, 7.0, 2.0, 0.0, 1.0, 2.0, 0.0, 3.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 2.0, 0.0,
0.0, 3.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 3.0, 2.0, 0.0, 0.0, 4.0, 1.0, 1.0, 0.0, 1.0, 0.0, 2.0, 3.0,
1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 2.0, 2.0, 2.0, 0.0,
0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 2.0, 1.0, 2.0, 0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 3.0, 0.0, 2.0, 1.0, 0.0, 2.0, 1.0, 3.0, 1.0, 0.0, 2.0, 3.0, 0.0, 0.0, 1.0,
1.0, 0.0, 2.0, 0.0, 3.0, 3.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 3.0, 1.0, 1.0, 2.0, 1.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 1.0, 1.0, 3.0, 0.0, 0.0, 4.0, 0.0, 1.0, 3.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 1.0, 1.0,
1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 4.0, 1.0, 2.0, 2.0, 3.0, 2.0, 0.0, 0.0,
0.0, 0.0, 4.0, 0.0, 0.0, 0.0, 0.0, 4.0, 0.0, 2.0, 0.0, 2.0, 0.0, 3.0, 0.0, 1.0, 2.0, 0.0, 3.0, 0.0,
0.0, 4.0, 0.0, 1.0, 2.0, 2.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.0, 1.0, 0.0,
2.0, 1.0, 0.0, 3.0, 0.0, 2.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 2.0, 0.0,
2.0, 0.0, 3.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 2.0, 0.0, 1.0, 2.0, 0.0, 0.0, 0.0, 0.0,
4.0, 1.0, 1.0, 2.0, 2.0, 0.0, 0.0, 3.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0,
0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 2.0, 1.0, 1.0, 0.0, 0.0, 0.0, 3.0, 4.0, 0.0, 0.0, 0.0, 4.0, 2.0,
4.0, 2.0, 1.0, 0.0, 0.0, 2.0, 8.0, 1.0, 3.0, 0.0, 0.0, 0.0, 3.0, 3.0, 0.0, 1.0, 1.0, 0.0, 1.0, 2.0,
1.0, 0.0, 0.0, 0.0, 2.0, 1.0, 0.0, 4.0, 1.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0, 0.0, 2.0, 1.0, 1.0, 1.0, 0.0, 2.0, 3.0, 2.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
4.0, 0.0, 0.0, 0.0, 0.0, 3.0, 0.0, 1.0, 0.0, 4.0, 1.0, 0.0, 1.0, 0.0, 0.0, 3.0, 0.0, 1.0, 2.0, 2.0,
1.0, 0.0, 4.0, 0.0, 3.0, 3.0, 0.0, 0.0, 2.0, 1.0, 0.0, 2.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 3.0,
1.0, 0.0, 1.0, 0.0, 1.0, 2.0, 1.0, 2.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 3.0, 2.0, 2.0, 0.0,
3.0, 0.0, 2.0, 0.0, 1.0, 0.0, 0.0, 2.0, 3.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 4.0,
0.0, 0.0, 2.0, 0.0, 2.0, 0.0, 0.0, 2.0, 0.0, 1.0, 1.0, 0.0, 1.0, 8.0, 0.0, 3.0, 0.0, 3.0, 1.0, 0.0,
0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 1.0, 3.0, 0.0, 1.0, 0.0, 1.0, 4.0, 0.0, 1.0, 0.0, 0.0, 5.0, 0.0,
1.0, 6.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.0, 4.0, 0.0, 0.0,
0.0, 2.0, 2.0, 0.0, 0.0, 3.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0]
```



1.2. Распределение Максвелла

1.2.1. Описание основных характеристик распределения

Математическое ожидание:

$$M\xi = \int_0^\infty x\sqrt{\frac{2}{\pi}}\frac{x^2}{\lambda^3}e^{-\frac{x^2}{2\lambda^2}}dx = \sqrt{\frac{2}{\pi}}\frac{1}{\lambda^3}\int_0^\infty x^3 e^{\frac{-x^2}{2\lambda^2}}dx = 2\lambda^4 \cdot \sqrt{\frac{2}{\pi}}\frac{1}{\lambda^3} = 2\lambda\sqrt{\frac{2}{\pi}}$$

Дисперсия:

$$D\xi = M(\xi - M\xi)^2 = M\xi^2 - (M\xi)^2 = M(\xi(\xi-1)+\xi) - (M\xi)^2 = M(\xi(\xi-1)) + M\xi - (M\xi)^2 = M(\xi(\xi-1)) + M\xi(1-(M\xi))$$

$$M(\xi(\xi-1)) = \int_0^\infty x^2 f(x)dx = \int_0^\infty x^2\sqrt{\frac{2}{\pi}}\frac{x^2}{\lambda^3}e^{-\frac{x^2}{2\lambda^2}} = \sqrt{\frac{2}{\pi}}\frac{1}{\lambda^3}\int_0^\infty x^4 e^{-\frac{x^2}{2\lambda^2}} = \sqrt{\frac{2}{\pi}}\frac{1}{\lambda^3}\cdot 3\lambda^4\sqrt{\frac{\lambda^2\pi}{2}} = 3\lambda^2$$

$$D\xi = M\xi^2 - (M\xi)^2 = 3\lambda^2 - 4\lambda^2\cdot\frac{2}{\pi} = \frac{3\pi-8}{\pi}\lambda^2$$

Все вычисленные интегралы можно найти в таблице интегралов, взятой из курса физики и представленной ниже.

$$\int \frac{dx}{x} = \ln x$$

$$\int x^m dx = \frac{1}{m+1} x^{m+1}, m \neq 1$$

$$\int_0^\infty x^n \cdot e^{-x} dx = n!$$

$$\int_0^\infty x^n \cdot e^{-\alpha x} dx = \frac{n!}{\alpha^{n+1}}$$

$$\int_0^\infty x^{1/2} \cdot e^{-\alpha x} dx = \frac{\sqrt{\pi}}{2} \alpha^{-3/2}$$

$$\int_0^\infty x^{3/2} \cdot e^{-\alpha x} dx = \frac{3}{4} \sqrt{\pi} \cdot \alpha^{-5/2}$$

$$\int_0^\infty e^{-\alpha x^2} dx = \frac{1}{2} \sqrt{\frac{\pi}{\alpha}}$$

$$\int_0^\infty x \cdot e^{-\alpha x^2} dx = \frac{1}{2\alpha}$$

$$\int_0^\infty x^2 \cdot e^{-\alpha x^2} dx = \frac{\sqrt{\pi}}{4} \alpha^{-3/2}$$

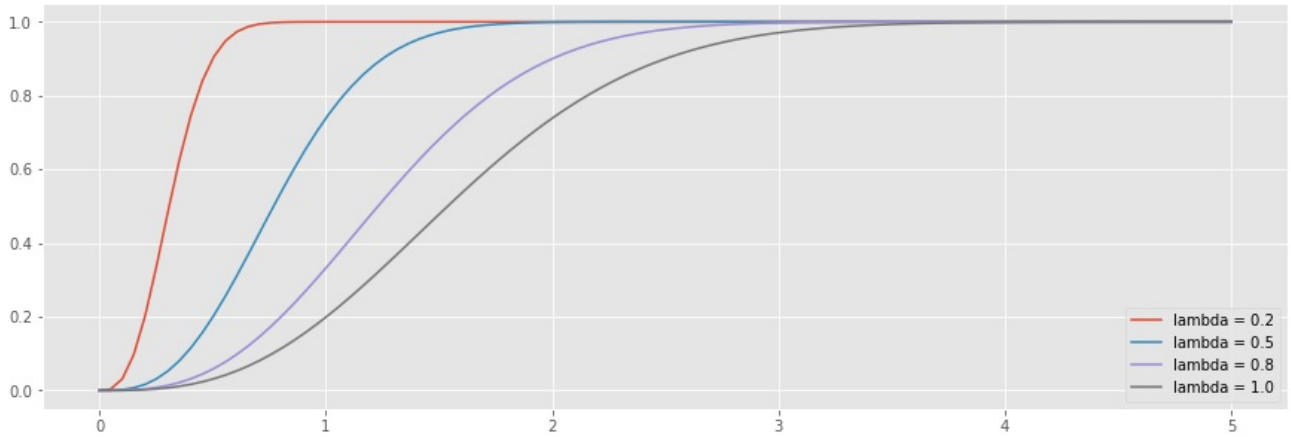
$$\int_0^\infty x^3 \cdot e^{-\alpha x^2} dx = \frac{1}{2\alpha^2}$$

$$\int_0^\infty x^4 \cdot e^{-\alpha x^2} dx = \frac{3}{8} \sqrt{\pi} \cdot \alpha^{-5/2}$$

In [5]:

```
for lambd in [0.2,0.5,0.8,1.0]:
    maxwell_rv = sts.maxwell(scale = lambd)
    x = np.linspace(0,5,100)
    cdf = maxwell_rv.cdf(x)
    plt.plot(x, cdf, label = 'lambda = {}'.format(lambd))
    plt.legend()
print('Рис.4: График функции распределения')
```

Рис.4: График функции распределения

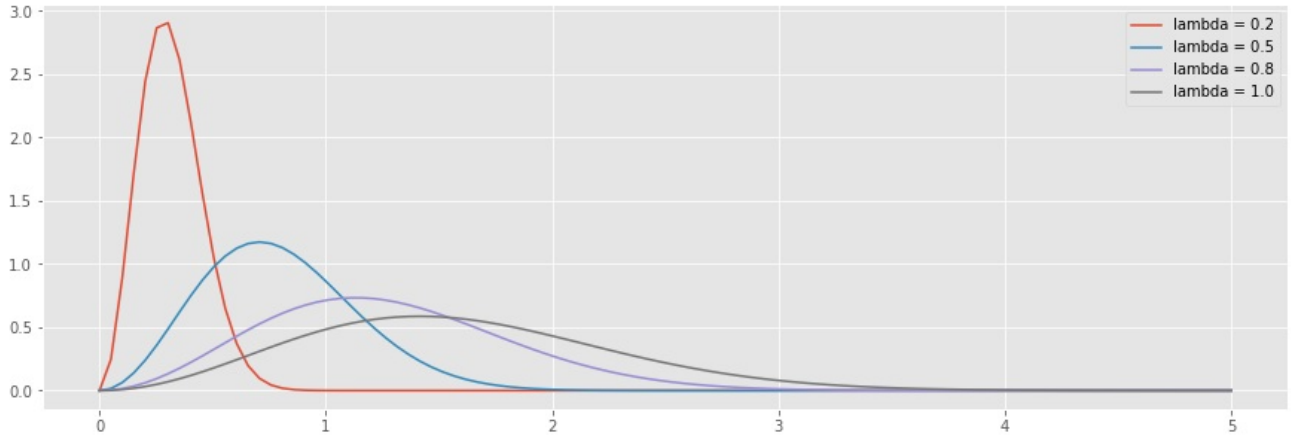


In [6]:

```
for lambd in [0.2,0.5,0.8,1.0]:
    maxwell_rv = sts.maxwell(scale = lambd)
    x = np.linspace(0,5,100)
    pdf = maxwell_rv.pdf(x)
    k = max(pdf)
    print('Значение моды по У:', k)
    plt.plot(x, pdf, label ='lambda = {}'.format(lambd))
    plt.legend()
print('\n')
print('Рис.5: График плотности вероятности распределения')
```

```
Значение моды по У: 2.9061680461442085
Значение моды по У: 1.1741012992781867
Значение моды по У: 0.7333399979540686
Значение моды по У: 0.5870506496390934
```

Рис.5: График плотности вероятности распределения



Исходя из графика видно, что $M_0 = \lambda\sqrt{2}$ - максимум функции плотности вероятности распределения

Медиана

$$\int_0^{Me} \sqrt{\frac{2}{\pi}} \frac{1}{\lambda^3} x^2 e^{-\frac{-x^2}{2\lambda^2}} dx = \frac{1}{2}$$

$$\int_0^{Me} x^2 e^{-\frac{-x^2}{2\lambda^2}} dx = \frac{\lambda^3}{2}\sqrt{\frac{\pi}{2}}$$

$$(-\lambda^2 e^{-\frac{x^2}{2\lambda^2}} x + \lambda^3 \sqrt{\frac{\pi}{2}}) \Big|_0^{Me} = \frac{\lambda^3}{2}\sqrt{\frac{\pi}{2}}$$

$$-Me\lambda^2 e^{\frac{-Me^2}{2\lambda^2}} = -\frac{\lambda^3}{2}\sqrt{\frac{\pi}{2}}$$

$$Me \cdot e^{\frac{-Me^2}{2\lambda^2}} = \frac{\lambda^2}{2}\sqrt{\frac{\pi}{2}}$$

$$Me \approx 1,5383\lambda$$

1.2.2. Примеры событий, которые могут быть описаны выбранными случайными величинами

Впервые распределение было определено и использовалось для описания скоростей частиц в идеализированных газах, где частицы свободноперемещаются внутри стационарного контейнера, не взаимодействуя друг с другом, за исключением очень коротких столкновений, в которыхони обмениваются энергией и импульсом друг с другом или со своим тепловым окружением. Термин «частица» в этом контексте относится только к газообразным частицам (атомам или молекулам), и предполагается, что система частиц достигла термодинамического равновесия.Энергии таких частиц следуют так называемой статистике Максвелла – Больцмана, а статистическое распределение скоростей выводится путем приравнивания энергии частиц к кинетической энергии.Распределение Максвелла – Больцмана в основном применяется к скоростям частиц в трех измерениях, но оказывается, что оно зависит только от скорости (величины скорости) частиц. Распределение вероятности скорости частицы указывает, какие скорости более вероятны: частица будет иметь скорость, выбранную случайным образом из распределения, и с большей вероятностью будет находиться в одном диапазоне скоростей, чем в другом.

При тепловом равновесии (T=const) $u_{\text{к в}}$ молекул газа остается постоянной и равной $u = \sqrt{\frac{3kT}{m}}$

Это объясняется тем, что в газе устанавливается стационарное статическое распределение молекул по значениям скоростей, называемое распределением Максвелла:

$f(u) = \frac{dN(u)}{Ndu} = 4\pi(\frac{m}{2\pi kT})^{\frac{3}{2}} \cdot u^2 \cdot e^{-\frac{mu^2}{2kT}}$

В теории вероятностей рассматривается распределения Максвелла, в котором $x = u$ и $\frac{1}{\lambda^2} = \frac{m}{kT}$

1.2.3. Описание способа моделирования выбранных случайных величин

Центральная предельная теорема - довольно неожиданный результат, связывающий выборочное среднее n независимых и одинаково распределенных (независимых и одинаково распределенных) случайных величин и нормального распределения. Точнее сказать

Пусть $X_1, X_2, \ldots, X_n - n$ независимых и одинаково распределенных случайных величин с $M(X_i) = \mu$ и $D(X_i) = \sigma^2$, и пусть $S_n = \frac{X_1 + X_2 + \ldots + X_n}{n}$ - среднее по выборке. Тогда S_n аппроксимирует нормальное распределение со средним значением μ и дисперсией $\frac{\sigma^2}{n}$ для больших n (т.е. $S_n \approx N(\mu, \frac{\sigma^2}{n})$)

Удивительный результат состоит в том, что X_n может быть любым распределением. Это не ограничивается только нормальными распределениями. Мы также можем определить стандартное нормальное распределение в терминах S_n, сдвигая и масштабируя его:

$$N(0, 1) \approx \frac{S_n - \mu}{\frac{\sigma}{\sqrt{n}}} = \frac{\sqrt{n}(S_n - \mu)}{\sigma} \quad (1)$$

Попробуем использовать центральную предельную теорему для выборки из N (0,1). Сначала давайте определим нашу независимую и одинаково распределенную переменную X_n так, чтобы она имела распределение Бернулли с p = 0,5, которое мы можем интуитивно представить как подбрасывание несмещенной монеты:

$$P(X_n = k) = \begin{cases} p = 0.5 & \text{если k = 1} \\ 1 - p = 0.5 & \text{если k = 0} \end{cases}$$

Напомним, распределение Бернулли тесно связано с биномиальным распределением, обозначенным через B(n, p) как Bernoulli(p) = B(n = 1, p). Биномиальное распределение можно интуитивно представить как подсчет количества орлов в n бросках монеты (т.е. в испытаниях Бернулли). Если n = 1, это сводится к распределению Бернулли (или единственному подбрасыванию монеты).

Давайте теперь определим нашу выборку среднего для n бросков нашей несмещенной монеты:

$$S_n = \frac{X_1 + X_2 + \ldots + X_n}{n} = \frac{B(n, p = 0.5)}{n}$$

Мы знаем, что распределение Бернулли имеет μ = 12 (мы ожидаем, что половина наших бросков будет орлом), и $\sigma^2 = p(1 - p) = 0.25$

Сдвигая и масштабируя это, чтобы получить наше стандартное нормальное распределение с использованием уравнения (1), мы получаем:

$$N(0, 1) \approx \frac{\sqrt{n}(S_n - \mu)}{\sigma} = \frac{\sqrt{n}(\frac{X_1 + X_2 + \ldots + X_n}{n} - 0.5)}{\sqrt{0.25}} = 2\sqrt{n}(\frac{X_1 + X_2 + \ldots + X_n}{n} - 0.5) \quad (2)$$

Теоретически это должно дать нам уравнение для грубой имитации стандартного нормального распределения.

In [7]:

```
def Maxwell(n,lambd = 1):
    x = [sample_(scale = lambd) for x in range(n)]
    y=[sample_(scale = lambd) for x in range(n)]
    z=[sample_(scale = lambd) for x in range(n)]
    l = []
    print(x)
    for i in range(n):
        l.append(np.sqrt(x[i]**2+y[i]**2+z[i]**2))
    return l
# Our sample function of N(0,1) using Equation (2)
def sample_(N = 3000,scale = 1):
    return scale*2.0*np.sqrt(N)*(sum(randint(0,1)for x in range(N))/N-0.5)
plt.hist(Maxwell(3000,5),30, width = 0.1)
plt.show()
```

```
[1.0954451150103333, -11.137025335938368, -3.651483716701101, 5.294651389216631, 6.572670690062, -1.
4605934866804342, 1.4605934866804646, 0.7302967433402019, 2.3734644158557017, 1.4605934866804646, -8
.033264176742433, -7.302967433402202, -0.7302967433402323, 2.7386127875258333, -0.9128709291752676,
-5.842373946721767, -4.0166320883712325, 1.4605934866804646, -10.771876964268268, 4.381780460041333,
3.286335345031, -2.0083160441856007, -2.7386127875258333, -2.5560386016907675, 2.1908902300206665, -
0.18257418583506568, 1.4605934866804646, 2.9211869733608684, -10.954451150103333, -1.825741858350565
6, 6.390096504226964, -8.2158383625775, -4.9295030175465, 7.120393247567166, -3.4689095308660653, 2.
9211869733608684, 1.8257418583505352, 4.746928831711465, 7.302967433402202, 5.294651389216631, -4.92
95030175465, 8.580986734247631, -4.564354645876398, 1.4605934866804646, 5.112077203381535, -4.381780
460041333, -0.36514837167010095, -4.9295030175465, 3.468909530866035, 6.937819061732131, -5.11207720
3381565, 7.668115805072333, -4.381780460041333, -5.659799760886732, -0.36514837167010095, 14.6059348
66804404, -1.8257418583505656, 2.1908902300206665, 0.18257418583503526, -8.763560920082666, 2.008316
0441856314, -4.0166320883712325, 4.199206274206298, 6.024948132556833, -0.18257418583506568, -4.0166
320883712325, -1.8257418583505656, -4.564354645876398, 0.7302967433402019, -6.207522318391868, -2.73
86127875258333, 0.5477225575051666, -2.921186973360899, -2.1908902300206665, 3.651483716701131, -1.8
257418583505656, -1.8257418583505656, 7.485541619237297, -4.0166320883712325, -2.921186973360899, -0
.7302967433402323, 1.6431676725155, -2.0083160441856007, 0.36514837167013137, -3.8340579025361663, 2
.7386127875258333, 2.3734644158557017, -2.1908902300206665, -4.9295030175465, -2.921186973360899, 5.
842373946721798, 0.912870929175298, 5.4772255750516665, -1.0954451150103333, 2.3734644158557017, 2.7
386127875258333, -1.4605934866804342, -3.651483716701101, 3.651483716701131, -0.18257418583506568, -
4.9295030175465, 4.564354645876368, 8.946135105917701, 4.746928831711465, -2.1908902300206665, 3.468
909530866035, 4.9295030175465, -2.921186973360899, -2.0083160441856007, -4.564354645876398, -3.10376
1159195934, 3.651483716701131, -1.4605934866804342, 0.36514837167013137, -2.921186973360899, 3.83405
79025361663, 4.0166320883712014, 6.755244875897035, -1.0954451150103333, 4.199206274206298, -4.74692
8831711434, 1.2780193008453686, 6.937819061732131, 12.232470450948702, -2.7386127875258333, 2.373464
4158557017, 5.659799760886702, -0.36514837167010095, -4.564354645876398, -9.128709291752767, -3.4689
095308660653, 3.651483716701131, -2.0083160441856007, 8.763560920082666, 3.651483716701131, 4.199206
274206298, 0.5477225575051666, 0.7302967433402019, 2.1908902300206665, -0.5477225575051666, 2.190890
2300206665, 0.5477225575051666, 6.937819061732131, -3.103761159195934, 2.0083160441856314, 0.9128709
29175298, 8.033264176742465, 4.9295030175465, 4.564354645876368, 3.286335345031, 1.2780193008453686,
1.4605934866804646, 4.381780460041333, -0.5477225575051666, 0.5477225575051666, -4.564354645876398,
-1.8257418583505656, -7.668115805072333, -2.1908902300206665, -4.381780460041333, 6.390096504226964,
-2.1908902300206665, 5.112077203381535, -5.2946513892166, -3.8340579025361663, -5.842373946721767, -
5.4772255750516665, 4.381780460041333, 0.5477225575051666, -5.112077203381565, -4.9295030175465, -0.
36514837167010095, -11.684747893443534, -4.9295030175465, -0.7302967433402323, 2.0083160441856314, 0
.912870929175298, -0.9128709291752676, -5.842373946721767, 7.668115805072333, -3.651483716701101, -7
.850689990907369, -5.4772255750516665, -1.8257418583505656, 1.6431676725155, 2.0083160441856314, -3.
8340579025361663, 7.302967433402202, 1.2780193008453686, 0.0, -3.651483716701101, -1.6431676725155,
1.2780193008453686, -1.0954451150103333, 5.294651389216631, 2.3734644158557017, 2.1908902300206665,
0.18257418583503526, -11.684747893443534, 3.1037611591959644, -5.4772255750516665, -1.27801930084539
9, -2.1908902300206665, -3.8340579025361663, -2.1908902300206665, 1.4605934866804646, 3.468909530866
035, 0.36514837167013137, 2.556038601690798, 0.0, 2.7386127875258333, -3.286335345031, -1.2780193008
45399, -1.8257418583505656, -1.6431676725155, 1.6431676725155, -2.5560386016907675, 2.73861278752583
33, 9.859006035093, -1.8257418583505656, 1.6431676725155, 5.294651389216631, -8.033264176742433, 3.8
340579025361663, 2.9211869733608684, 0.18257418583503526, -4.564354645876398, -6.390096504226934, 2.
1908902300206665, -6.024948132556833, -7.120393247567166, 3.1037611591959644, 0.18257418583503526, -
1.6431676725155, -5.2946513892166, -2.7386127875258333, 4.564354645876368, 2.0083160441856314, -4.74
6928831711434, -0.36514837167010095, -2.3734644158557323, 4.0166320883712014, 2.7386127875258333, -3
.286335345031, 2.1908902300206665, 0.18257418583503526, 9.493857663422869, -5.842373946721767, 1.278
0193008453686, -7.485541619237267, 4.564354645876368, 7.668115805072333, -2.921186973360899, -7.3029
67433402202, 2.3734644158557017, -1.0954451150103333, 0.7302967433402019, 0.7302967433402019, -0.365
14837167010095, 6.390096504226964, -3.286335345031, -3.103761159195934, -1.4605934866804342, -1.2780
19300845399, 7.668115805072333, 2.556038601690798, 7.120393247567166, -0.7302967433402323, -4.564354
645876398, -7.302967433402202, -0.36514837167010095, -6.755244875897035, -9.311283477587834, 0.0, -7
.302967433402202, -8.763560920082666, -0.7302967433402323, 1.0954451150103333, -1.8257418583505656,
-5.112077203381565, 0.36514837167013137, 1.2780193008453686, -6.024948132556833, 2.9211869733608684,
3.651483716701131, 2.3734644158557017, 2.3734644158557017, -0.7302967433402323, -5.659799760886732,
-11.137025335938368, 1.4605934866804646, -2.3734644158557323, -3.651483716701101, 4.199206274206298,
-8.763560920082666, 2.9211869733608684, 4.746928831711465, -7.302967433402202, 4.564354645876368, 3.
8340579025361663, -0.9128709291752676, 3.468909530866035, -2.0083160441856007, 2.7386127875258333, 2
.1908902300206665, -5.112077203381565, -1.6431676725155, -3.8340579025361663, 12.780193008453868, -3
.651483716701101, 2.3734644158557017, -1.0954451150103333, 3.651483716701131, -2.5560386016907675, -
1.8257418583505656, -2.3734644158557323, -4.381780460041333, -8.946135105917701, -0.3651483716701009
5, 3.8340579025361663, -8.946135105917701, -2.3734644158557323, 0.36514837167013137, 4.9295030175465
, 1.6431676725155, 11.502173707608499, -6.390096504226934, -1.0954451150103333, 7.302967433402202, 8
.398412548412535, 10.041580220928035, -3.4689095308660653, -5.2946513892166, -2.1908902300206665, -7
```

.485541619237267, 8.946135105917701, 1.2780193008453686, 0.36514837167013137, -5.842373946721767, 2.
1908902300206665, -4.9295030175465, 0.7302967433402019, 3.286335345031, 1.0954451150103333, 5.659799
760886702, 1.8257418583505352, 6.024948132556833, -0.9128709291752676, 3.468909530866035, -6.5726706
90062, 12.962767194288903, -2.5560386016907675, 3.651483716701131, -2.1908902300206665, 3.2863353450
31, -2.5560386016907675, 3.1037611591959644, -1.8257418583505656, 6.024948132556833, -3.103761159195
934, 0.7302967433402019, 7.850689990907369, -2.921186973360899, -2.921186973360899, -5.6597997608867
32, 11.319599521773464, -1.278019300845399, -4.746928831711434, 11.319599521773464, 2.00831604418563
14, 4.564354645876368, 8.2158383625775, 2.9211869733608684, 3.8340579025361663, 13.875638123464201,
-9.859006035093, -0.9128709291752676, 2.3734644158557017, 0.18257418583503526, 1.8257418583505352, 4
.381780460041333, 1.4605934866804646, -4.199206274206268, -9.859006035093, -1.6431676725155, -0.3651
4837167010095, -3.651483716701101, 1.4605934866804646, 4.564354645876368, -1.278019300845399, -12.59
7618822618832, -5.659799760886732, -2.3734644158557323, 14.240786495134332, -2.921186973360899, 2.00
83160441856314, -5.842373946721767, 5.659799760886702, -1.4605934866804342, 1.0954451150103333, -1.6
431676725155, 0.5477225575051666, 5.294651389216631, 2.3734644158557017, -6.755244875897035, 7.85068
9990907369, -5.842373946721767, 13.693063937629166, -2.7386127875258333, 4.564354645876368, 2.556038
601690798, 4.0166320883712014, 6.024948132556833, -1.4605934866804342, -10.2241544067631, 0.18257418
583503526, -5.112077203381565, -8.2158383625775, 2.9211869733608684, -1.8257418583505656, 12.2324704
50948702, 2.3734644158557017, 0.18257418583503526, 9.128709291752797, 4.0166320883712014, 2.37346441
58557017, 1.8257418583505352, 3.8340579025361663, 0.7302967433402019, -2.0083160441856007, 0.0, -2.7
386127875258333, -0.7302967433402323, 2.7386127875258333, -6.207522318391868, 3.651483716701131, -1.
6431676725155, -2.0083160441856007, 2.1908902300206665, 5.659799760886702, -9.676431849257934, 4.381
780460041333, -10.406728592598165, 0.7302967433402019, 5.4772255750516665, 0.0, -0.18257418583506568
, -5.2946513892166, 3.468909530866035, -3.651483716701101, 2.7386127875258333, 6.755244875897035, -0
.18257418583506568, -5.659799760886732, -3.651483716701101, -11.867322079278601, 0.5477225575051666,
-1.0954451150103333, 4.9295030175465, 10.954451150103333, 3.1037611591959644, 8.763560920082666, 6.9
37819061732131, -2.7386127875258333, -5.4772255750516665, -2.0083160441856007, -3.103761159195934, 2
.1908902300206665, -0.18257418583506568, -4.564354645876398, -2.7386127875258333, 6.572670690062, -7
.485541619237267, 1.2780193008453686, 3.468909530866035, -3.8340579025361663, 3.8340579025361663, -4
.199206274206268, 12.597618822618832, 0.7302967433402019, -2.7386127875258333, -2.0083160441856007,
-8.033264176742433, -1.4605934866804342, 0.7302967433402019, -4.199206274206268, -4.199206274206268,
-7.485541619237267, -9.493857663422869, 7.302967433402202, -2.5560386016907675, -1.278019300845399,
-1.8257418583505656, -11.867322079278601, 5.294651389216631, 3.286335345031, 0.912870929175298, -0.9
128709291752676, -1.6431676725155, -0.36514837167010095, 2.9211869733608684, -1.0954451150103333, -1
.278019300845399, 4.746928831711465, -5.659799760886732, -5.2946513892166, -1.4605934866804342, -3.2
86335345031, 8.763560920082666, 7.302967433402202, 5.842373946721798, 3.651483716701131, 3.651483716
701131, 2.7386127875258333, -13.875638123464201, 5.842373946721798, -1.4605934866804342, -5.11207720
3381565, 3.286335345031, 0.7302967433402019, 0.36514837167013137, 0.7302967433402019, 2.373464415855
7017, 6.207522318391868, -3.286335345031, -2.1908902300206665, 3.1037611591959644, 4.9295030175465,
-2.7386127875258333, 0.912870929175298, -1.0954451150103333, 5.4772255750516665, 2.7386127875258333,
-1.4605934866804342, 5.112077203381535, -1.6431676725155, -7.668115805072333, -0.36514837167010095,
-7.485541619237267, -8.763560920082666, 8.946135105917701, 4.746928831711465, -2.3734644158557323, -
0.9128709291752676, 1.4605934866804646, 2.9211869733608684, -2.7386127875258333, 2.9211869733608684,
-0.5477225575051666, -0.18257418583506568, -3.4689095308660653, 0.7302967433402019, -3.8340579025361
663, -4.746928831711434, 5.842373946721798, 4.564354645876368, -4.0166320883712325, -3.4689095308660
653, 0.18257418583503526, -4.564354645876398, 0.36514837167013137, -2.0083160441856007, 4.0166320883
712014, -7.120393247567166, 2.3734644158557017, 2.0083160441856314, -4.564354645876398, -0.182574185
83506568, 8.033264176742465, -0.18257418583506568, -4.0166320883712325, 3.651483716701131, 6.7552448
75897035, -0.36514837167010095, -1.4605934866804342, 8.033264176742465, -2.1908902300206665, 0.18257
418583503526, 0.18257418583503526, 0.36514837167013137, -2.1908902300206665, 4.746928831711465, 0.54
77225575051666, 2.7386127875258333, 4.9295030175465, 0.912870929175298, -5.659799760886732, -6.20752
2318391868, 2.0083160441856314, 9.859006035093, 2.9211869733608684, -7.485541619237267, -6.937819061
7321005, 4.381780460041333, 8.033264176742465, -10.5893027784332, -2.7386127875258333, 15.1536574243
0957, -0.7302967433402323, 5.112077203381535, -2.5560386016907675, -9.859006035093, -5.1120772033815
65, -7.668115805072333, -0.5477225575051666, 5.4772255750516665, 0.5477225575051666, 11.137025335938
368, 6.024948132556833, -4.381780460041333, -1.4605934866804342, 7.120393247567166, 5.11207720338153
5, -8.398412548412535, -5.842373946721767, -4.564354645876398, -4.0166320883712325, 4.381780460041333
, -3.8340579025361663, 8.398412548412535, 3.8340579025361663, -4.564354645876398, 3.651483716701131
, 4.199206274206298, 2.0083160441856314, 1.2780193008453686, -5.2946513892166, 2.556038601690798, 4.
9295030175465, -2.921186973360899, -2.3734644158557323, -0.9128709291752676, -0.9128709291752676, 0.
0, 2.9211869733608684, -4.199206274206268, 4.381780460041333, 0.18257418583503526, -2.921186973360899
, 6.755244875897035, -1.0954451150103333, -1.8257418583505656, 1.4605934866804646, 1.095445115010333
33, 2.7386127875258333, 3.286335345031, 1.4605934866804646, -0.5477225575051666, 8.398412548412535,
9.311283477587834, 4.0166320883712014, 0.0, 2.7386127875258333, -2.1908902300206665, -1.278019300845
399, 1.4605934866804646, -8.946135105917701, 1.4605934866804646, 11.137025335938368, -1.825741858350
5656, 2.9211869733608684, 3.1037611591959644, 4.9295030175465, 3.286335345031, 1.2780193008453686, 6
.024948132556833, -3.286335345031, 3.651483716701131, 1.2780193008453686, -2.921186973360899, -5.477
2255750516665, 0.18257418583503526, 4.199206274206298, 4.0166320883712014, 1.0954451150103333, -2.00
83160441856007, 7.850689990907369, 2.0083160441856314, 6.207522318391868, 1.6431676725155, 2.1908902
300206665, 4.564354645876368, -0.5477225575051666, -2.5560386016907675, 5.112077203381535, -3.834057
9025361663, -1.0954451150103333, 6.937819061732131, 2.3734644158557017, -2.3734644158557323, -4.9295
030175465, -9.493857663422869, -3.8340579025361663, 4.746928831711465, 6.755244875897035, -0.1825741
8583506568, 9.859006035093, -4.381780460041333, -0.7302967433402323, 4.746928831711465, 0.3651483716
7013137, 0.5477225575051666, -5.842373946721767, -2.7386127875258333, 2.0083160441856314, 4.38178046
0041333, -2.0083160441856007, 3.286335345031, -2.921186973360899, -0.18257418583506568, -4.564354645
876398, 0.18257418583503526, 4.9295030175465, 3.286335345031, 8.580986734247631, -3.286335345031, -1
0.5893027784332, -0.9128709291752676, 2.0083160441856314, 2.0083160441856314, 3.8340579025361663, 6.
207522318391868, -3.286335345031, 3.468909530866035, 2.7386127875258333, -4.199206274206268, 3.46890
9530866035, -12.049896265113667, 9.676431849257964, 3.651483716701131, 2.0083160441856314, -2.008316
0441856007, 6.937819061732131, -2.921186973360899, -3.651483716701101, 1.6431676725155, -0.912870929
1752676, 0.0, 9.859006035093, -2.1908902300206665, 10.22415440676313, 2.9211869733608684, -12.597618
822618832, -0.9128709291752676, 4.0166320883712014, 1.4605934866804646, 3.468909530866035, -3.286335
345031, 3.1037611591959644, 1.4605934866804646, 1.4605934866804646, -1.8257418583505656, 3.834057902
5361663, 2.1908902300206665, 0.0, 4.9295030175465, -2.1908902300206665, -4.9295030175465, 8.03326417

6742465, -1.6431676725155, 9.676431849257964, 0.5477225575051666, -2.921186973360899, 0.182574185835
03526, -3.4689095308660653, 2.9211869733608684, -0.7302967433402323, -3.286335345031, 9.128709291752
797, -11.319599521773434, -0.18257418583506568, -2.0083160441856007, 2.1908902300206665, 0.547722557
5051666, 1.8257418583505352, -3.286335345031, 0.18257418583503526, -2.5560386016907675, 6.9378190617
32131, 1.0954451150103333, 2.9211869733608684, -4.564354645876398, -0.36514837167010095, -1.46059348
66804342, 1.4605934866804646, -2.1908902300206665, 4.564354645876368, -2.1908902300206665, -8.215838
3625775, 2.1908902300206665, 3.651483716701131, -3.8340579025361663, 3.468909530866035, -4.381780460
041333, 6.572670690062, 11.684747893443534, 8.946135105917701, 2.9211869733608684, -2.190890230020666
65, -5.112077203381565, 6.572670690062, -4.9295030175465, 5.294651389216631, -3.286335345031, -2.373
4644158557323, 0.7302967433402019, 2.556038601690798, -0.18257418583506568, 1.4605934866804646, 1.09
54451150103333, -4.564354645876398, -11.502173707608499, -0.18257418583506568, -9.859006035093, -1.2
78019300845399, 3.651483716701131, 1.2780193008453686, -11.137025335938368, -8.2158383625775, 2.9211
869733608684, 4.564354645876368, 0.5477225575051666, -3.4689095308660653, 0.36514837167013137, 1.278
0193008453686, -13.327915565959035, -0.18257418583506568, -2.5560386016907675, 3.468909530866035, -2
.1908902300206665, 7.850689990907369, -1.0954451150103333, 10.406728592598165, -8.033264176742433, -
5.2946513892166, -4.199206274206268, 5.842373946721798, -4.0166320883712325, -5.842373946721767, -1.
0954451150103333, -2.5560386016907675, 2.0083160441856314, -0.9128709291752676, -2.921186973360899,
2.3734644158557017, 9.311283477587834, 3.1037611591959644, -10.406728592598165, 3.8340579025361663,
-3.8340579025361663, -9.128709291752767, 6.937819061732131, -1.4605934866804342, 1.8257418583505352,
6.572670690062, 1.2780193008453686, 6.755244875897035, 1.6431676725155, 3.468909530866035, -0.365148
37167010095, 4.0166320883712014, 1.2780193008453686, 10.406728592598165, 0.0, -3.103761159195934, 3.
651483716701131, 0.7302967433402019, 1.4605934866804646, -8.2158383625775, 2.7386127875258333, -12.2
32470450948702, 1.2780193008453686, -0.5477225575051666, 5.659799760886702, 7.668115805072333, -6.20
7522318391868, -1.0954451150103333, 1.0954451150103333, 5.4772255750516665, 4.746928831711465, 5.477
2255750516665, -2.1908902300206665, 6.937819061732131, 6.572670690062, -1.278019300845399, -2.008316
0441856007, 0.912870929175298, -0.18257418583506568, -2.3734644158557323, -3.103761159195934, 3.8340
579025361663, 2.1908902300206665, -7.120393247567166, -4.199206274206268, 6.572670690062, 0.18257418
583503526, 10.771876964268298, 5.659799760886702, 1.4605934866804646, -10.406728592598165, -4.564354
645876398, 1.6431676725155, -4.381780460041333, 0.18257418583503526, 4.381780460041333, -0.547722557
5051666, -7.120393247567166, 5.112077203381535, -0.18257418583506568, -9.128709291752767, 0.36514837
167013137, -4.0166320883712325, -1.0954451150103333, 0.18257418583503526, -4.199206274206268, 6.3900
96504226964, 1.6431676725155, 11.86732207927857, -1.8257418583505656, 2.3734644158557017, -3.8340579
025361663, -5.2946513892166, 0.5477225575051666, -10.954451150103333, -6.572670690062, -3.8340579025
361663, 2.0083160441856314, 1.6431676725155, 0.36514837167013137, -1.2780193008453399, -10.5893027784
332, -0.9128709291752676, -1.2780193008453399, -5.4772255750516665, 3.651483716701131, 2.921186973360
8684, 8.763560920082666, -9.493857663422869, 4.9295030175465, 0.18257418583503526, -7.48554161923726
7, 2.3734644158557017, 1.0954451150103333, 6.937819061732131, 14.240786495134332, -1.095445115010333
3, -5.112077203381565, 3.468909530866035, 1.0954451150103333, 0.0, 1.0954451150103333, 0.54772255750
51666, -6.9378190617321005, 4.0166320883712014, -0.5477225575051666, 2.556038601690798, -4.016632088
3712325, -3.4689095308660653, 4.9295030175465, 3.286335345031, -6.207522318391868, 5.659799760886702
, -2.3734644158557323, 1.0954451150103333, -2.0083160441856007, 10.5893027784332, 11.502173707608499
, 2.0083160441856314, -4.746928831711434, 8.398412548412535, -6.9378190617321005, 6.024948132556833,
8.763560920082666, -5.4772255750516665, 3.468909530866035, 0.5477225575051666, -8.946135105917701, 0
.5477225575051666, 1.8257418583505352, 4.0166320883712014, -1.4605934866804342, -10.771876964268268,
-7.302967433402202, -0.18257418583506568, 1.8257418583505352, -3.4689095308660653, -5.65979976088673
2, -1.8257418583505656, 5.294651389216631, -3.8340579025361663, -3.286335345031, 3.286335345031, -3.
651483716701101, -6.390096504226934, 7.120393247567166, -5.112077203381565, 8.2158383625775, -2.0083
160441856007, -2.921186973360899, -7.485541619237267, -2.3734644158557323, -2.1908902300206665, 0.0,
2.0083160441856314, 3.286335345031, -9.128709291752767, 5.294651389216631, 6.755244875897035, -7.850
689990907369, 0.5477225575051666, 1.6431676725155, -1.0954451150103333, 4.564354645876368, 7.8506899
90907369, 2.1908902300206665, -1.6431676725155, -0.9128709291752676, -10.406728592598165, -1.2780193
00845399, -0.9128709291752676, 2.3734644158557017, -4.746928831711434, 4.199206274206298, 3.28633534
5031, 11.502173707608499, 4.564354645876368, 7.302967433402202, -4.564354645876398, -3.4689095308660
653, 5.112077203381535, 2.3734644158557017, 1.8257418583505352, 6.390096504226964, 0.0, 0.5477225575
051666, 2.7386127875258333, -0.9128709291752676, -2.7386127875258333, -6.572670690062, 0.36514837167
013137, 3.651483716701131, -8.5809867342476, 3.468909530866035, -3.286335345031, -4.0166320883712325
, -6.390096504226934, -0.9128709291752676, 5.659799760886702, -9.859006035093, -4.0166320883712325,
2.3734644158557017, 2.1908902300206665, 2.1908902300206665, -8.2158383625775, 0.36514837167013137, 4
.199206274206298, -2.1908902300206665, -4.381780460041333, 1.6431676725155, -6.572670690062, -3.1037
61159195934, -0.18257418583506568, 0.18257418583503526, 5.842373946721798, 2.556038601690798, -4.746
928831711434, -1.6431676725155, -3.286335345031, -9.859006035093, -1.278019300845399, -4.92950301754
65, -5.4772255750516665, -1.8257418583505656, -6.9378190617321005, -0.5477225575051666, -2.373464415
8557323, 4.199206274206298, 1.8257418583505352, 8.763560920082666, 2.0083160441856314, -2.9211869733
60899, 7.668115805072333, -5.4772255750516665, -9.859006035093, 4.564354645876368, 0.912870929175298
, -0.36514837167010095, 1.6431676725155, -1.2780193008453399, -2.7386127875258333, -5.2946513892166,
-3.8340579025361663, -7.302967433402202, -6.9378190617321005, 1.6431676725155, 19.352863698515870, -8
.2158383625775, -4.199206274206268, 2.7386127875258333, 1.2780193008453686, 0.36514837167013137, 4.0
166320883712014, 4.199206274206298, 4.9295030175465, 4.564354645876368, -3.651483716701101, 4.564354
645876368, 5.659799760886702, -0.18257418583506568, 0.5477225575051666, -2.1908902300206665, -0.1825
7418583506568, 0.912870929175298, -2.0083160441856007, 11.86732207927857, -2.0083160441856007, -5.84
2373946721767, 4.9295030175465, 0.0, 2.3734644158557017, 0.5477225575051666, -0.36514837167010095, -
0.7302967433402323, 1.8257418583505352, 2.556038601690798, -4.199206274206268, -7.302967433402202, 7
.302967433402202, 7.485541619237297, 6.207522318391868, 4.199206274206298, 0.912870929175298, -3.286
335345031, -0.5477225575051666, -2.0083160441856007, -8.398412548412535, -7.302967433402202, -2.1908
902300206665, 0.36514837167013137, -3.4689095308660653, 9.859006035093, 7.120393247567166, -0.730296
7433402323, -0.36514837167010095, 0.5477225575051666, -5.659799760886732, -16.066528353484866, 3.834
0579025361663, 9.676431849257964, 6.024948132556833, -0.5477225575051666, -2.5560386016907675, 7.485
541619237297, -4.0166320883712325, 1.4605934866804646, -0.7302967433402323, -3.4689095308660653, 4.1
992062742062298, 0.0, 10.406728592598165, 3.468909530866035, -0.9128709291752676, 1.2780193008453686,
1.6431676725155, -0.18257418583506568, 2.9211869733608684, -6.572670690062, 3.8340579025361663, 4.01
66320883712014, -2.3734644158557323, -3.286335345031, -5.659799760886732, 5.842373946721798, 0.73029
67433402019, 3.651483716701131, -9.128709291752767, -4.381780460041333, 10.954451150103333, 3.651483
716701131, -5.4772255750516665, 2.0083160441856314, 2.7386127875258333, 4.9295030175465, -3.46890953

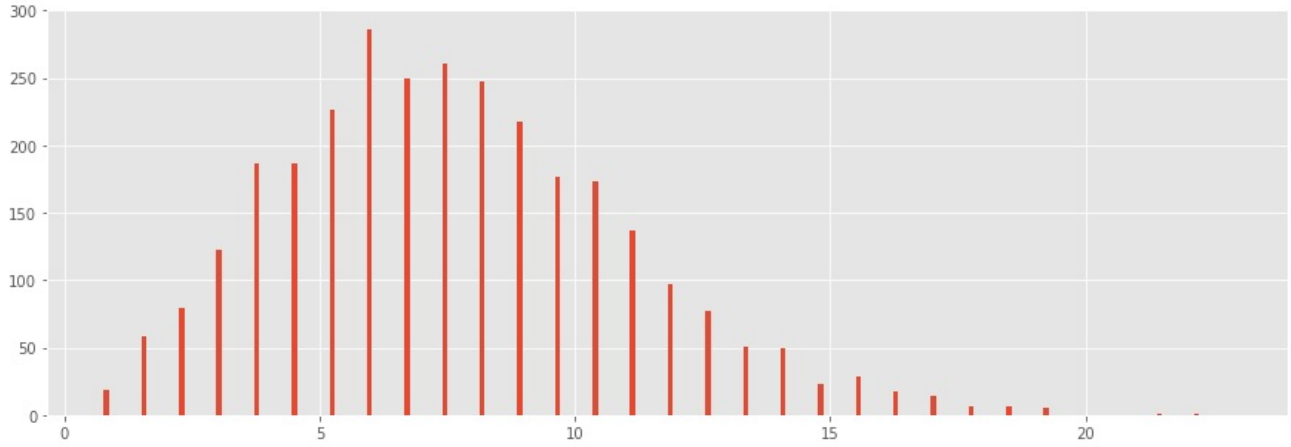
08660653, 8.580986734247631, -3.103761159195934, -0.9128709291752676, 9.128709291752797, 11.50217370
7608499, -4.746928831711434, -5.112077203381565, -10.954451150103333, 4.199206274206298, -4.38178046
0041333, 4.564354645876368, 10.406728592598165, -3.8340579025361663, 5.842373946721798, 1.8257418583
505352, -1.278019300845399, 0.36514837167013137, -3.103761159195934, -2.921186973360899, 0.365148371
67013137, -2.921186973360899, -6.207522318391868, 0.0, 2.9211869733608684, 1.0954451150103333, -4.92
95030175465, 6.572670690062, -3.286335345031, 2.3734644158557017, -0.36514837167010095, 6.2075223183
91868, -9.859006035093, -2.1908902300206665, 4.564354645876368, -3.4689095308660653, -0.182574185835
06568, -2.3734644158557323, 0.912870929175298, -8.398412548412535, 7.120393247567166, 1.643167672515
5, 3.651483716701131, 4.564354645876368, 4.0166320883712014, -4.0166320883712325, 7.302967433402202,
5.294651389216631, 2.1908902300206665, -2.921186973360899, 4.564354645876368, 0.7302967433402019, -0
.5477225575051666, -0.5477225575051666, -7.120393247567166, 6.207522318391868, 5.1120077203381535, -2
.0083160441856007, 2.9211869733608684, 0.18257418583503526, 6.755244875897035, 8.763560920082666, 2.
0083160441856314, -2.921186973360899, -4.381780460041333, -8.5809867342476, -1.0954451150103333, -10
.2241544067631, 8.946135105917701, 0.5477225575051666, 0.18257418583503526, -5.1120077203381565, -2.0
083160441856007, -0.7302967433402323, 4.381780460041333, -8.5809867342476, 8.398412548412535, -3.286
335345031, 1.0954451150103333, -0.9128709291752676, -8.2158383625775, -0.5477225575051666, 0.0, 3.65
1483716701131, 3.8340579025361663, -6.390096504226934, -2.3734644158557323, 8.763560920082666, 7.850
689990907369, 3.468909530866035, -1.6431676725155, 6.390096504226964, -6.572670690062, 4.56435464587
6368, -5.842373946721767, 11.137025335938368, 1.8257418583505352, 5.294651389216631, 1.6431676725155
, 0.7302967433402019, 1.8257418583505352, 8.763560920082666, -16.7968250968251, -9.859006035093, -12
.597618822618832, 8.398412548412535, -3.651483716701101, -3.103761159195934, -4.381780460041333, -2.
7386127875258333, -5.4772255750516665, -9.311283477587834, -0.7302967433402323, -0.7302967433402323,
6.572670690062, 6.207522318391868, -1.0954451150103333, -5.2946513892166, -4.746928831711434, -3.103
761159195934, 2.3734644158557017, -9.859006035093, 2.3734644158557017, 3.468909530866035, -4.0166320
883712325, 4.564354645876368, -5.659799760886732, -1.8257418583505656, 8.2158383625775, 5.2946513892
16631, -2.0083160441856007, -0.7302967433402323, 1.6431676725155, -2.1908902300206665, -7.3029674334
02202, -1.8257418583505656, 6.937819061732131, -0.18257418583506568, 1.0954451150103333, 3.651483716
701131, -3.103761159195934, 1.6431676725155, -3.103761159195934, 10.406728592598165, -3.834057902536
1663, -4.0166320883712325, -6.207522318391868, -1.0954451150103333, 5.842373946721798, 2.00831604418
56314, -9.859006035093, -4.746928831711434, 2.3734644158557017, 7.120393247567166, -0.73029674334023
23, -0.18257418583506568, -3.286335345031, 1.4605934866804646, -2.5560386016907675, -5.6597997608867
32, -2.1908902300206665, 0.36514837167013137, -11.867322079278601, 4.564354645876368, -4.19920627420
6268, -1.278019300845399, -4.746928831711434, -2.3734644158557323, -1.0954451150103333, 3.1037611591
959644, 1.2780193008453686, -8.398412548412535, 3.8340579025361663, 2.3734644158557017, 3.4689095308
66035, 1.0954451150103333, 4.199206274206298, 3.286335345031, 5.294651389216631, -6.390096504226934,
-2.0083160441856007, 0.912870929175298, -6.390096504226934, -0.5477225575051666, -6.024948132556833,
-2.7386127875258333, -2.1908902300206665, -1.278019300845399, -2.0083160441856007, 4.746928831711465
, -5.1120077203381565, 3.651483716701131, 10.22415440676313, -0.9128709291752676, -4.0166320883712325
, 6.390096504226964, 1.4605934866804646, -3.8340579025361663, 3.468909530866035, -8.033264176742433,
-4.9295030175465, -11.502173707608499, -3.8340579025361663, -7.120393247567166, 1.8257418583505352,
0.7302967433402019, -8.2158383625775, 0.0, 2.7386127875258333, 4.381780460041333, 1.0954451150103333
, -3.286335345031, 1.8257418583505352, -2.0083160441856007, -9.128709291752767, -12.049896265113667,
-5.842373946721767, -2.0083160441856007, 0.36514837167013137, -2.5560386016907675, 6.024948132556833
, -5.842373946721767, 0.36514837167013137, -5.4772255750516665, 6.755244875897035, -3.83405790253616
63, -10.771876964268268, -4.0166320883712325, -0.7302967433402323, -8.2158383625775, -2.373464415855
7323, 2.1908902300206665, -7.668115805072333, 2.9211869733608684, 1.2780193008453686, 1.643167672515
5, -5.2946513892166, -1.4605934866804342, -5.112077203381565, -4.381780460041333, 0.5477225575051666
, -5.4772255750516665, -2.921186973360899, 1.4605934866804646, -1.6431676725155, -4.0166320883712325
, -1.8257418583505656, 3.1037611591959644, 3.1037611591959644, 12.415044636783737, 1.095445115010333
3, -5.112077203381565, -1.6431676725155, 3.286335345031, -2.5560386016907675, -0.5477225575051666, 2
.7386127875258333, 0.7302967433402019, 3.651483716701131, 3.8340579025361663, 1.4605934866804646, -6
.390096504226934, -6.207522318391868, -3.286335345031, 5.842373946721798, 3.286335345031, -2.5560386
016907675, -8.946135105917701, -5.4772255750516665, 4.381780460041333, 4.199206274206298, -6.9378190
617321005, -2.3734644158557323, -1.4605934866804342, -1.0954451150103333, -7.485541619237267, -3.468
9095308660653, 8.763560920082666, 1.0954451150103333, 0.7302967433402019, -2.921186973360899, -6.755
244875897035, -4.564354645876398, -4.9295030175465, -3.103761159195934, 8.580986734247631, -8.763560
920082666, 1.4605934866804646, 5.294651389216631, 5.112077203381535, -6.207522318391868, 1.825741858
3505352, 7.850689990907369, 11.502173707608499, -1.278019300845399, 2.3734644158557017, -1.095445115
0103333, -10.954451150103333, -6.024948132556833, -7.850689990907369, 0.912870929175298, -0.54772255
75051666, 4.0166320883712014, -1.6431676725155, 8.033264176742465, 0.18257418583503526, -6.024948132
556833, -0.7302967433402323, -1.8257418583505656, -8.5809867342476, 8.763560920082666, 3.10376115919
59644, 1.8257418583505352, -15.701379981814767, 0.912870929175298, -1.278019300845399, 5.65979976088
6702, -1.278019300845399, -2.3734644158557323, -2.7386127875258333, -8.5809867342476, -5.11207720338
1565, 1.0954451150103333, 5.659799760886702, -2.0083160441856007, 8.763560920082666, -1.825741858350
5656, -10.5893027784332, -4.564354645876398, 3.1037611591959644, -0.5477225575051666, 0.547722557505
1666, 1.0954451150103333, -4.564354645876398, -0.7302967433402323, 1.4605934866804646, -8.9461351059
17701, 5.842373946721798, -2.921186973360899, 3.1037611591959644, 2.3734644158557017, 0.0, -3.103761
159195934, -4.564354645876398, -6.207522318391868, 2.0083160441856314, -10.771876964268268, 6.572670
690062, 5.4772255750516665, 1.8257418583505352, -2.0083160441856007, -3.103761159195934, -1.27801930
0845399, -2.921186973360899, -0.36514837167010095, 5.842373946721798, 1.4605934866804646, -1.0954451
150103333, 1.6431676725155, -0.5477225575051666, 1.6431676725155, 3.468909530866035, 0.5477225575051
666, -4.746928831711434, -3.8340579025361663, -2.3734644158557323, -0.36514837167010095, 6.390096504
226964, 1.0954451150103333, 1.4605934866804646, 9.676431849257964, -2.5560386016907675, 6.7552448758
97035, -3.103761159195934, 2.0083160441856314, 1.6431676725155, 1.4605934866804646, 6.20752231839186
8, -14.7885090526395, -0.5477225575051666, -0.36514837167010095, -5.4772255750516665, 10.77187696426
8298, -4.0166320883712325, 2.556038601690798, -1.8257418583505656, 0.0, 0.18257418583503526, 5.29465
1389216631, -1.4605934866804342, 2.0083160441856314, 2.7386127875258333, 10.771876964268298, 1.46059
34866804646, 12.415044636783737, 10.954451150103333, 0.36514837167013137, -3.8340579025361663, -1.09
54451150103333, -1.0954451150103333, -1.4605934866804342, -1.278019300845399, 0.912870929175298, -1.
0954451150103333, -1.6431676725155, -4.564354645876398, 5.112077203381535, 4.199206274206298, -6.390
096504226934, -0.7302967433402323, -1.4605934866804342, -8.946135105917701, -16.249102539319935, 0.1
8257418583503526, 5.4772255750516665, -0.36514837167010095, -4.0166320883712325, 3.468909530866035,
-3.651483716701101, 2.7386127875258333, -7.668115805072333, -0.18257418583506568, 1.6431676725155, 0

.5477225575051666, 7.120393247567166, -6.390096504226934, -7.302967433402202, 4.199206274206298, -3.
651483716701101, -2.1908902300206665, 6.024948132556833, -4.746928831711434, -6.390096504226934, -6.
390096504226934, -3.8340579025361663, -0.5477225575051666, -2.7386127875258333, 0.5477225575051666,
-2.5560386016907675, 3.286335345031, -2.921186973360899, 2.1908902300206665, 4.381780460041333, 7.48
5541619237297, -7.485541619237267, 0.912870929175298, 7.668115805072333, -8.946135105917701, -0.5477
225575051666, -3.651483716701101, 0.5477225575051666, -6.390096504226934, -2.7386127875258333, 1.460
5934866804646, -1.0954451150103333, -8.5809867342476, -5.112077203381565, 2.3734644158557017, -1.095
4451150103333, -4.381780460041333, -4.381780460041333, 0.912870929175298, 9.128709291752797, -2.1908
902300206665, 0.36514837167013137, -0.36514837167010095, 3.286335345031, 2.556038601690798, -8.03326
4176742433, -3.103761159195934, -5.112077203381565, -0.36514837167010095, -0.9128709291752676, 10.40
6728592598165, -6.390096504226934, 5.842373946721798, -10.041580220928035, 4.199206274206298, -7.120
393247567166, 3.8340579025361663, 2.3734644158557017, 0.5477225575051666, 7.668115805072333, -2.0083
160441856007, -1.6431676725155, -3.8340579025361663, 9.676431849257964, -1.278019300845399, -0.91287
09291752676, 4.564354645876368, 10.5893027784332, -2.921186973360899, 0.36514837167013137, 4.1992062
74206298, 9.128709291752797, 1.6431676725155, 0.18257418583503526, -8.398412548412535, 0.54772255750
51666, 8.580986734247631, -2.921186973360899, 0.0, -4.9295030175465, 2.7386127875258333, -8.94613510
5917701, -2.1908902300206665, 2.0083160441856314, -11.867322079278601, -1.8257418583505656, 0.912870
929175298, 10.041580220928035, 1.0954451150103333, -6.207522318391868, 0.0, -2.5560386016907675, 2.1
908902300206665, 8.033264176742465, -1.278019300845399, -4.0166320883712325, -8.946135105917701, -3.
286335345031, -22.4566248577118, -2.921186973360899, 5.4772255750516665, 6.024948132556833, 0.547722
5575051666, 0.912870929175298, -5.659799760886732, -3.8340579025361663, -7.120393247567166, -7.12039
3247567166, -3.103761159195934, -5.4772255750516665, 0.912870929175298, -4.199206274206268, 0.365148
37167013137, -0.9128709291752676, -1.8257418583505656, -5.112077203381565, 2.9211869733608684, 1.278
0193008453686, 4.746928831711465, -4.746928831711434, 0.7302967433402019, -7.120393247567166, 5.1120
77203381535, 6.207522318391868, 5.4772255750516665, 9.311283477587834, 12.232470450948702, 4.3817804
60041333, 3.1037611591959644, 1.2780193008453686, 2.1908902300206665, -8.2158383625775, -2.190890230
0206665, -1.4605934866804342, 3.8340579025361663, -6.572670690062, -3.651483716701101, 0.0, 2.738612
7875258333, -7.120393247567166, -3.286335345031, 7.120393247567166, -0.5477225575051666, -6.75524487
5897035, -8.763560920082666, -0.9128709291752676, 6.024948132556833, -4.746928831711434, -5.65979976
0886732, 13.327915565959035, 3.286335345031, 8.398412548412535, 1.2780193008453686, -3.8340579025361
663, 1.6431676725155, 1.0954451150103333, 0.36514837167013137, -6.755244875897035, -4.01663208837123
25, 3.468909530866035, 4.381780460041333, -12.232470450948702, 1.8257418583505352, -2.92118697336089
9, 2.7386127875258333, -7.302967433402202, 0.7302967433402019, -6.755244875897035, -6.20752231839186
8, 0.5477225575051666, -3.103761159195934, -10.5893027784332, -8.2158383625775, -1.4605934866804342,
5.659799760886702, -0.5477225575051666, 7.850689990907369, -3.103761159195934, 1.6431676725155, 2.00
83160441856314, -0.5477225575051666, 6.207522318391868, -1.4605934866804342, -2.7386127875258333, 3.
286335345031, 0.0, 5.294651389216631, 1.6431676725155, 4.199206274206298, 3.1037611591959644, 4.1992
06274206298, 0.5477225575051666, 2.556038601690798, -2.5560386016907675, 7.850689990907369, -8.94613
5105917701, -9.859006035093, 1.0954451150103333, -1.0954451150103333, -0.7302967433402323, 0.1825741
8583503526, 10.954451150103333, 5.4772255750516665, -8.2158383625775, -12.049896265113667, -0.365148
37167010095, 1.8257418583505352, 2.0083160441856314, 8.580986734247631, -6.207522318391868, 2.373464
4158557017, 1.8257418583505352, 3.1037611591959644, 3.1037611591959644, 5.4772255750516665, -2.73861
27875258333, 1.6431676725155, 0.18257418583503526, -4.199206274206268, -8.763560920082666, -1.825741
8583505656, -0.9128709291752676, 5.842373946721798, -1.0954451150103333, 5.294651389216631, -3.83405
79025361663, 5.659799760886702, -4.381780460041333, -2.1908902300206665, 6.755244875897035, 3.651483
716701131, -6.755244875897035, -4.564354645876398, 6.572670690062, 5.112077203381535, -1.46059348668
04342, 1.6431676725155, -6.572670690062, 0.0, 2.1908902300206665, 2.3734644158557017, -8.76356092008
2666, -4.746928831711434, -3.286335345031, -9.128709291752767, -12.597618822618832, -0.5477225575051
666, 12.049896265113667, 2.556038601690798, 4.199206274206298, -0.9128709291752676, -2.3734644158557
323, 7.668115805072333, -3.4689095308660653, -5.659799760886732, 2.1908902300206665, -5.294651389216
6, -6.572670690062, -1.8257418583505656, 4.9295030175465, -6.024948132556833, 6.024948132556833, 5.8
42373946721798, -6.207522318391868, -1.0954451150103333, -8.398412548412535, -7.485541619237267, -1.
6431676725155, -3.4689095308660653, 3.8340579025361663, -4.381780460041333, 9.311283477587834, 3.103
7611591959644, -2.7386127875258333, -6.024948132556833, -0.7302967433402323, -4.564354645876398, 3.1
037611591959644, 4.9295030175465, 0.7302967433402019, 8.398412548412535, 1.8257418583505352, 4.38178
0460041333, -2.5560386016907675, -2.5560386016907675, -0.9128709291752676, 0.0, -6.207522318391868,
-2.921186973360899, -8.398412548412535, -4.746928831711434, -2.5560386016907675, -3.4689095308660653
, 7.850689990907369, -1.6431676725155, -2.5560386016907675, -11.502173707608499, 0.0, -2.92118697336
0899, -6.390096504226934, -6.207522318391868, -1.4605934866804342, 0.5477225575051666, 3.83405790253
61663, 7.850689990907369, -0.5477225575051666, -5.2946513892166, -0.9128709291752676, -1.27801930084
5399, 9.676431849257964, -1.8257418583505656, 1.8257418583505352, -4.199206274206268, -5.65979976088
6732, -0.36514837167010095, 9.493857663422869, -1.0954451150103333, 3.651483716701131, 0.18257418583
503526, -5.112077203381565, -1.4605934866804342, 5.4772255750516665, 3.286335345031, -10.40672859259
8165, -2.0083160441856007, -3.103761159195934, 5.4772255750516665, 2.0083160441856314, 4.19920627420
6298, 3.468909530866035, 8.946135105917701, -8.946135105917701, 3.8340579025361663, 0.0, -5.47722557
50516665, 1.0954451150103333, -5.4772255750516665, 8.398412548412535, 1.0954451150103333, 1.46059348
66804646, 4.9295030175465, 2.0083160441856314, -3.651483716701101, -1.0954451150103333, -0.912870929
1752676, -5.659799760886732, -7.120393247567166, 2.0083160441856314, 1.8257418583505352, 3.286335345
031, 6.937819061732131, -1.0954451150103333, -4.381780460041333, -1.8257418583505656, -8.03326417674
2433, -4.564354645876398, -5.112077203381565, -7.668115805072333, -4.9295030175465, -7.4855416192372
67, -0.7302967433402323, -6.024948132556833, -5.659799760886732, 3.1037611591959644, 0.0, 0.0, 4.564
354645876368, 1.8257418583505352, 1.0954451150103333, 1.0954451150103333, 0.7302967433402019, 2.0083
160441856314, 0.5477225575051666, -8.398412548412535, -2.7386127875258333, -5.112077203381565, 7.850
689990907369, 4.746928831711465, 4.9295030175465, -4.0166320883712325, -4.746928831711434, -3.651483
716701101, 7.485541619237297, 1.2780193008453686, 7.302967433402202, -0.9128709291752676, 7.12039324
7567166, 7.120393247567166, -10.041580220928035, -0.5477225575051666, 4.381780460041333, -3.28633534
5031, -5.842373946721767, -0.36514837167010095, -3.8340579025361663, -6.9378190617321005, 2.73861278
75258333, -1.6431676725155, -7.302967433402202, 10.406728592598165, 1.0954451150103333, 3.6514837167
01131, -0.5477225575051666, -2.1908902300206665, 1.4605934866804646, 2.7386127875258333, -2.92118697
3360899, -4.0166320883712325, -2.921186973360899, 0.912870929175298, -0.18257418583506568, -1.278019
300845399, 1.8257418583505352, -4.9295030175465, 5.659799760886702, -1.6431676725155, -3.10376115919
5934, -5.842373946721767, 3.651483716701131, 1.4605934866804646, -6.572670690062, 2.0083160441856314
, -6.024948132556833, 0.7302967433402019, -8.5809867342476, 0.5477225575051666, 5.842373946721798, -

0.36514837167010095, 0.7302967433402019, -5.2946513892166, -6.207522318391868, 2.1908902300206665, 5
.294651389216631, 3.1037611591959644, -5.2946513892166, 3.286335345031, -0.36514837167010095, -0.912
8709291752676, 0.7302967433402019, -4.199206274206268, 2.556038601690798, 3.1037611591959644, -2.190
8902300206665, -4.564354645876398, -4.0166320883712325, -0.7302967433402323, 6.572670690062, 2.55603
8601690798, -4.381780460041333, -1.0954451150103333, 1.4605934866804646, -4.381780460041333, 5.29465
1389216631, 1.0954451150103333, -1.4605934866804342, -4.199206274206268, -0.36514837167010095, -1.64
31676725155, 4.9295030175465, -12.049896265113667, 6.572670690062, 4.0166320883712014, 6.57267069006
2, 3.286335345031, 6.937819061732131, -1.8257418583505656, 0.7302967433402019, -2.7386127875258333,
4.564354645876368, -6.390096504226934, -5.659799760886732, -4.381780460041333, 5.294651389216631, 8.
763560920082666, -4.0166320883712325, -1.8257418583505656, -2.7386127875258333, -6.207522318391868,
7.668115805072333, -4.381780460041333, -5.842373946721767, 0.0, -1.4605934866804342, -1.460593486680
4342, -5.842373946721767, -4.564354645876398, -0.9128709291752676, -0.36514837167010095, -1.27801930
0845399, 6.937819061732131, -7.302967433402202, 2.9211869733608684, -2.1908902300206665, 2.556038601
690798, 0.18257418583503526, -4.199206274206268, -0.9128709291752676, 5.294651389216631, 1.095445115
0103333, 0.912870929175298, 2.9211869733608684, 10.041580220928035, -3.286335345031, -6.024948132556
833, 3.8340579025361663, 9.493857663422869, 5.112077203381535, -3.103761159195934, 0.730296743340201
9, 4.199206274206298, -0.18257418583506568, 1.0954451150103333, 8.2158383625775, -7.668115805072333,
2.556038601690798, 7.485541619237297, 3.651483716701131, 2.556038601690798, 8.763560920082666, -0.73
02967433402323, 7.120393247567166, 6.024948132556833, 4.0166320883712014, 1.8257418583505352, -4.016
6320883712325, 0.0, -0.18257418583506568, 6.937819061732131, 1.6431676725155, -0.9128709291752676, 2
.9211869733608684, 12.597618822618832, -2.5560386016907675, 1.8257418583505352, 9.859006035093, -0.7
302967433402323, 11.86732207927857, 7.302967433402202, 9.128709291752797, -0.18257418583506568, 1.82
57418583505352, -1.0954451150103333, 5.842373946721798, -7.302967433402202, 5.294651389216631, 1.278
0193008453686, -1.4605934866804342, 1.6431676725155, -0.9128709291752676, 10.771876964268298, -4.199
206274206268, -3.103761159195934, 0.7302967433402019, -1.4605934866804342, -5.2946513892166, 3.65148
3716701131, -4.0166320883712325, 6.755244875897035, -0.18257418583506568, -2.1908902300206665, 2.008
3160441856314, -8.2158383625775, 10.771876964268298, -5.2946513892166, -0.18257418583506568, 0.54772
25575051666, -2.0083160441856007, -6.390096504226934, 3.286335345031, -4.199206274206268, 5.29465138
9216631, 3.651483716701131, 4.564354645876368, 4.0166320883712014, -3.4689095308660653, 0.5477225575
051666, -1.4605934866804342, -2.921186973360899, 1.6431676725155, 2.9211869733608684, 1.825741858350
5352, 2.556038601690798, 0.0, 3.8340579025361663, -6.390096504226934, -1.0954451150103333, 9.3112834
77587834, -2.7386127875258333, 6.937819061732131, 4.564354645876368, 5.842373946721798, -4.199206274
206268, 8.033264176742465, 1.2780193008453686, 2.3734644158557017, -3.4689095308660653, 1.0954451150
103333, 4.9295030175465, -3.4689095308660653, 5.659799760886702, -2.1908902300206665, -0.91287092917
52676, -14.058212309299266, -1.4605934866804342, 5.842373946721798, 1.0954451150103333, -0.365148371
67010095, -9.311283477587834, 1.6431676725155, -1.8257418583505656, 2.3734644158557017, 3.1037611591
959644, 2.9211869733608684, -0.36514837167010095, 0.912870929175298, 2.7386127875258333, -1.27801930
0845399, 1.0954451150103333, -0.36514837167010095, -5.842373946721767, -6.572670690062, 5.4772255750
516665, 3.651483716701131, -0.9128709291752676, -1.8257418583505656, 2.3734644158557017, -0.36514837
167010095, -8.763560920082666, 1.2780193008453686, 1.4605934866804646, 4.564354645876368, 2.00831604
41856314, -3.4689095308660653, -1.278019300845399, -4.381780460041333, -6.024948132556833, 4.1992062
74206298, 1.0954451150103333, -11.502173707608499, 5.842373946721798, -2.0083160441856007, 2.9211869
733608684, -6.572670690062, 1.6431676725155, -13.875638123464201, -7.302967433402202, -8.58098673424
76, 0.5477225575051666, -4.9295030175465, 2.9211869733608684, -5.659799760886732, 1.2780193008453686
, 1.8257418583505352, -0.5477225575051666, 4.0166320883712014, 5.294651389216631, 4.746928831711465,
-6.207522318391868, -2.7386127875258333, -0.36514837167010095, -11.137025335938368, -2.3734644158557
323, -10.771876964268268, -2.921186973360899, -0.18257418583506568, 3.1037611591959644, -1.278019300
845399, -7.668115805072333, 2.7386127875258333, 1.2780193008453686, 3.468909530866035, 2.55603860169
0798, 3.468909530866035, -1.2780193008845399, 3.286335345031, 3.1037611591959644, 0.5477225575051666,
13.145341380124, 2.9211869733608684, 1.4605934866804646, -6.572670690062, -0.36514837167010095, 1.27
80193008453686, -3.4689095308660653, 4.199206274206298, -0.36514837167010095, 1.4605934866804646, 1.
2780193008453686, -0.18257418583506568, -6.9378190617321005, 0.912870929175298, 2.9211869733608684,
-0.18257418583506568, 2.9211869733608684, -3.103761159195934, 0.7302967433402019, -1.460593486680434
2, -1.4605934866804342, -4.0166320883712325, 1.4605934866804646, 5.112077203381535, -2.0083160441856
007, -1.6431676725155, -2.1908902300206665, -10.2241544067631, 0.5477225575051666, 8.763560920082666
, -7.302967433402202, 0.7302967433402019, 3.468909530866035, -5.659799760886732, 6.024948132556833,
-0.7302967433402323, -1.4605934866804342, -2.921186973360899, -0.18257418583506568, 1.6431676725155,
3.286335345031, 2.7386127875258333, 7.120393247567166, -1.8257418583505656, 2.1908902300206665, -11.
867322079278601, 7.302967433402202, 0.36514837167013137, -4.381780460041333, -4.564354645876398, 4.1
99206274206298, -3.8340579025361663, 2.3734644158557017, 1.6431676725155, -5.4772255750516665, -9.12
8709291752767, 2.3734644158557017, -0.5477225575051666, 3.286335345031, -3.286335345031, -2.73861278
75258333, -7.302967433402202, 0.36514837167013137, -3.4689095308660653, -6.755244875897035, 0.365148
37167013137, -5.842373946721767, 14.423360680969369, -0.36514837167010095, 0.7302967433402019, 1.643
1676725155, -0.5477225575051666, -6.9378190617321005, 9.676431849257964, 8.2158383625775, -7.1203932
47567166, 1.6431676725155, 6.755244875897035, 2.3734644158557017, -2.1908902300206665, 5.65979976088
6702, -6.572670690062, 7.668115805072333, -3.4689095308660653, 1.6431676725155, 3.651483716701131, 4
.9295030175465, -3.8340579025361663, 2.9211869733608684, 0.912870929175298, 7.120393247567166, 5.842
373946721798, 2.9211869733608684, 4.199206274206298, -8.398412548412535, -6.572670690062, 0.91287092
9175298, -2.3734644158557323, -1.0954451150103333, 3.651483716701131, -2.7386127875258333, 5.2946513
89216631, 7.120393247567166, -1.8257418583505656, -3.8340579025361663, -0.7302967433402323, 1.460593
4866804646, -7.850689990907369, 7.302967433402202, 8.580986734247631, 4.564354645876368, 3.468909530
866035, 1.2780193008453686, 4.199206274206298, -3.103761159195934, -0.7302967433402323, 2.9211869733
608684, -3.651483716701101, -4.199206274206268, 2.0083160441856314, 0.36514837167013137, 4.016632088
3712014, -1.4605934866804342, -3.4689095308660653, -4.0166320883712325, 10.22415440676313, 9.8590060
35093, 1.0954451150103333, 4.9295030175465, -1.6431676725155, -4.0166320883712325, 5.294651389216631
, -2.3734644158557323, 2.7386127875258333, 10.041580220928035, -5.112077203381565, 7.668115805072333
, -0.7302967433402323, 6.024948132556833, -7.302967433402202, -3.651483716701101, -1.095445115010333
3, -7.668115805072333, -4.9295030175465, -0.5477225575051666, 12.415044636783737, 0.1825741858350352
6, 0.0, -1.8257418583505656, 2.7386127875258333, 2.3734644158557017, -5.659799760886732, -6.39009650
4226934, 4.199206274206298, -1.278019300845399, -9.311283477587834, -3.651483716701101, 0.5477225575
051666, 2.9211869733608684, 4.746928831711465, 13.693063937629166, 2.9211869733608684, -2.9211869733
60899, -3.8340579025361663, 8.580986734247631, 7.302967433402202, -7.485541619237267, 5.842373946721
798, 6.207522318391868, 1.6431676725155, 3.8340579025361663, 6.024948132556833, 1.8257418583505352,

-5.4772255750516665, 2.3734644158557017, 0.7302967433402019, 12.232470450948702, -2.3734644158557323
, -2.5560386016907675, -2.7386127875258333, -1.6431676725155, -6.024948132556833, 2.9211869733608684
, 6.937819061732131, -7.120393247567166, 3.8340579025361663, -0.7302967433402323, -4.9295030175465,
-4.9295030175465, -4.0166320883712325, -1.278019300845399, -6.9378190617321005, 4.381780460041333, 5
.842373946721798, 5.112077203381535, -0.9128709291752676, 0.7302967433402019, 10.22415440676313, 3.8
340579025361663, 10.954451150103333, 3.468909530866035, -2.7386127875258333, 2.3734644158557017, -4.
0166320883712325, 1.4605934866804646, -8.5809867342476, 3.468909530866035, -8.2158383625775, 5.84237
3946721798, 3.468909530866035, 0.5477225575051666, -7.120393247567166, -4.381780460041333, -5.477225
5750516665, -3.651483716701101, -5.2946513892166, 1.2780193008453686, -1.6431676725155, -2.556038601
6907675, -3.286335345031, -6.572670690062, -0.36514837167010095, -1.0954451150103333, -10.0415802209
28035, -4.199206274206268, -7.485541619237267, -4.9295030175465, 3.468909530866035, 1.6431676725155,
-1.4605934866804342, -4.199206274206268, 0.7302967433402019, 1.2780193008453686, -3.8340579025361663
, -7.120393247567166, 6.755244875897035, 9.493857663422869, -1.4605934866804342, -8.398412548412535,
-1.8257418583505656, -11.319599521773434, 5.842373946721798, 11.502173707608499, 1.8257418583505352,
0.36514837167013137, -3.651483716701101, 6.390096504226964, 4.381780460041333, 2.9211869733608684, 4
.9295030175465, 14.058212309299236, 6.572670690062, -2.921186973360899, 8.763560920082666, 0.1825741
8583503526, 0.912870929175298, -1.8257418583505656, -1.6431676725155, -12.597618822618832, 6.7552448
75897035, -2.0083160441856007, 0.912870929175298, 1.0954451150103333, 6.937819061732131, -2.37346441
58557323, -3.651483716701101, -6.9378190617321005, -3.8340579025361663, -3.103761159195934, -0.73029
67433402323, -1.8257418583505656, -2.5560386016907675, 2.556038601690798, 5.294651389216631, -6.5726
70690062, 3.651483716701131, 2.9211869733608684, -6.755244875897035, -2.3734644158557323, -1.4605934
866804342, 5.294651389216631, 4.564354645876368, 6.937819061732131, 2.556038601690798, -5.1120772033
81565, -2.921186973360899, 12.415044636783737, -3.286335345031, -7.302967433402202, 2.190890230020666
65, 0.5477225575051666, 4.0166320883712014, -1.6431676725155, -2.1908902300206665, 3.286335345031, -
8.033264176742433, 3.468909530866035, -3.103761159195934, -1.0954451150103333, -2.1908902300206665,
-4.0166320883712325, -3.286335345031, -0.9128709291752676, -4.564354645876398, 0.36514837167013137,
-4.9295030175465, 0.5477225575051666, 2.1908902300206665, 3.286335345031, -0.36514837167010095, 1.09
54451150103333, -0.9128709291752676, 0.912870929175298, 2.1908902300206665, 0.18257418583503526, -2.
3734644158557323, -0.9128709291752676, -3.103761159195934, 0.5477225575051666, 9.128709291752797, -0
.5477225575051666, 1.4605934866804646, 0.912870929175298, 1.8257418583505352, 6.937819061732131, 3.8
340579025361663, 7.302967433402202, -3.651483716701101, 4.564354645876368, 4.9295030175465, 1.460593
4866804646, -0.18257418583506568, 9.493857663422869, -4.564354645876398, 4.746928831711465, 4.199206
274206298, -0.9128709291752676, -3.4689095308660653, -1.0954451150103333, -3.4689095308660653, 0.365
14837167013137, -7.302967433402202, -5.2946513892166, 1.2780193008453686, 2.3734644158557017, 1.2780
193008453686, 0.0, 0.912870929175298, -6.024948132556833, -5.4772255750516665, 9.859006035093, 7.120
393247567166, -0.9128709291752676, -4.0166320883712325, 1.0954451150103333, 3.1037611591959644, -5.8
42373946721767, 2.0083160441856314, -2.3734644158557323, -3.286335345031, 2.3734644158557017, 4.1992
06274206298, 8.2158383625775, 5.842373946721798, -1.4605934866804342, -4.9295030175465, 0.7302967433
402019, -2.5560386016907675, -0.36514837167010095, 0.18257418583503526, 5.842373946721798, 2.1908902
300206665, -5.112077203381565, 6.207522318391868, -7.485541619237267, -5.112077203381565, 1.27801930
08453686, -6.9378190617321005, -13.693063937629166, -2.1908902300206665, -3.286335345031, 0.0, -4.38
1780460041333, -4.9295030175465, -2.7386127875258333, -8.398412548412535, -1.6431676725155, 0.182574
18583503526, -1.4605934866804342, -2.1908902300206665, 0.912870929175298, -0.5477225575051666, 1.278
0193008453686, -2.5560386016907675, -2.5560386016907675, 1.4605934866804646, -9.128709291752767, -7.
120393247567166, 6.207522318391868, -9.311283477587834, 0.7302967433402019, 2.556038601690798, -10.7
71876964268268, -9.493857663422869, -10.041580220928035, -6.390096504226934, -1.0954451150103333, -0
.36514837167010095, -4.199206274206268, 6.937819061732131, -6.390096504226934, 6.755244875897035, -4
.564354645876398, 1.6431676725155, 3.8340579025361663, 7.302967433402202, 6.207522318391868, 10.7718
76964268298, -3.286335345031, 0.7302967433402019, 1.4605934866804646, -6.755244875897035, -2.5560386
016907675, 3.8340579025361663, 3.468909530866035, -0.36514837167010095, 1.8257418583505352, 3.651483
716701131, -0.7302967433402323, 2.556038601690798, -15.336231610144665, -1.0954451150103333, 0.0, 8.
033264176742465, -0.18257418583506568, 7.485541619237297, 3.468909530866035, 6.390096504226964, -4.3
81780460041333, 14.058212309299236, -2.3734644158557323, 2.1908902300206665, 0.7302967433402019, 1.8
257418583505352, 4.9295030175465, 1.8257418583505352, 2.0083160441856314, -7.485541619237267, 3.1037
611591959644, -5.659799760886732, 0.18257418583503526, 5.842373946721798, 6.207522318391868, 2.55603
8601690798, 1.2780193008453686, 13.51048975179407, 0.7302967433402019, -2.5560386016907675, 0.912870
929175298, -7.850689990907369, 6.937819061732131, -8.5809867342476, 8.2158383625775, 3.1037611591959
644, -4.9295030175465, -1.278019300845399, 0.5477225575051666, -1.6431676725155, -5.2946513892166, 2
.3734644158557017, 4.9295030175465, 9.859006035093, 2.0083160441856314, 2.556038601690798, 3.2863353
45031, 2.3734644158557017, -3.103761159195934, 12.415044636783737, -5.2946513892166, 1.8257418583505
352, 4.199206274206298, 7.302967433402202, -1.278019300845399, -4.746928831711434, 12.232470450948702
, -9.128709291752767, -2.5560386016907675, 3.651483716701131, 1.8257418583505352, -1.460593486680434
42, 2.7386127875258333, -0.9128709291752676, -3.651483716701101, -2.5560386016907675, 0.547722557505
1666, 0.7302967433402019, 3.651483716701131, 2.0083160441856314, -5.659799760886732, -2.738612787525
8333, 4.746928831711465, 2.0083160441856314, -5.659799760886732, -11.867322079278601, 0.0, -7.485541
619237267, 5.4772255750516665, 1.0954451150103333, 5.659799760886702, 3.286335345031, 7.485541619237
297, 4.564354645876368, -6.207522318391868, 3.1037611591959644, 3.286335345031, -5.2946513892166, 0.
912870929175298, 0.5477225575051666, 0.5477225575051666, 8.2158383625775, 0.5477225575051666, 2.0083
160441856314, -1.8257418583505656, 4.381780460041333, -1.278019300845399, 3.651483716701131, 17.5271
21840165332, -16.066528353484866, 5.112077203381535, -1.0954451150103333, 4.199206274206298, 0.0, 4.
0166320883712014, -9.128709291752767, 0.18257418583503526, -2.921186973360899, 4.9295030175465, 4.74
6928831711465, 8.398412548412535, 2.9211869733608684, -0.36514837167010095, -3.4689095308660653, -0.
18257418583506568, 0.0, 2.9211869733608684, -8.033264176742433, 2.9211869733608684, 0.91287092917529
8, -4.199206274206268, -2.921186973360899, 9.676431849257964, -0.18257418583506568, -14.605934866804
434, -4.746928831711434, -1.0954451150103333, -4.199206274206268, 1.2780193008453686, 4.199206274206
298, -9.493857663422869, -8.033264176742433, 1.6431676725155, -5.4772255750516665, 6.207522318391868
, -6.9378190617321005, 6.390096504226964, 2.556038601690798, 4.0166320883712014, -2.1908902300206665
, 2.3734644158557017, -3.4689095308660653, -0.7302967433402323, 3.468909530866035, 11.31959952177346
4, 2.556038601690798, -2.7386127875258333, 5.842373946721798, 9.676431849257964, -8.946135105917701,
2.9211869733608684, -2.921186973360899, -2.0083160441856007, -4.199206274206268, 8.763560920082666,
-0.18257418583506568, -3.651483716701101, 7.485541619237297, -7.120393247567166, -7.850689990907369,
5.4772255750516665, -0.5477225575051666, 5.842373946721798, 7.850689990907369, -0.9128709291752676,
8.763560920082666, -3.651483716701101, 3.468909530866035, 4.564354645876368, 0.18257418583503526, 2.

```
3734644158557017, -2.7386127875258333, 3.8340579025361663, 1.8257418583505352, -1.0954451150103333,
0.18257418583503526, 2.1908902300206665, -6.024948132556833, -2.5560386016907675, 4.0166320883712014
, 3.286335345031, 5.842373946721798, 3.468909530866035, 3.8340579025361663, -3.286335345031, 4.92950
30175465, 3.8340579025361663, -4.9295030175465, -3.8340579025361663, 2.1908902300206665, 4.564354645
876368, 7.668115805072333, 4.746928831711465, -0.7302967433402323, 4.746928831711465, -0.18257418583
506568, 0.18257418583503526, 6.572670690062, 0.912870929175298, -2.7386127875258333, 1.2780193008453
686, -3.651483716701101, 1.8257418583505352, -3.4689095308660653, -4.199206274206268, -10.0415802209
28035, 3.651483716701131, 1.4605934866804646, 3.468909530866035, 1.4605934866804646, -8.946135105917
701, 5.294651389216631, -5.2946513892166, 7.485541619237297, 8.398412548412535, -2.7386127875258333,
-0.9128709291752676, -1.278019300845399, -0.5477225575051666, 1.2780193008453686, -2.556038601690767
5, 12.415044636783737, -4.0166320883712325, 7.668115805072333, -7.120393247567166, -0.91287092917526
76, 1.2780193008453686, -1.0954451150103333, -0.9128709291752676, -0.36514837167010095, -2.008316044
1856007, 2.1908902300206665, -5.112077203381565, -1.0954451150103333, -8.033264176742433, 3.65148371
6701131, -3.8340579025361663, -1.4605934866804342, -10.771876964268268, 11.137025335938368, 1.460593
4866804646, -4.9295030175465, -0.18257418583506568, -5.112077203381565, -4.0166320883712325, 4.56435
4645876368, 11.137025335938368, -2.921186973360899, -3.103761159195934, -2.7386127875258333, -5.2946
513892166, -6.390096504226934, -7.668115805072333, 0.912870929175298, 5.4772255750516665, -0.3651483
7167010095, 4.381780460041333, -2.921186973360899, -1.4605934866804342, -3.103761159195934, 1.278019
3008453686, -0.9128709291752676, -1.6431676725155, 3.286335345031, 0.5477225575051666, -4.0166320883
712325, 4.199206274206298, -5.2946513892166, 2.0083160441856314, -1.4605934866804342, -6.93781906173
21005, -1.278019300845399, 4.9295030175465, 9.676431849257964, -5.2946513892166, -3.103761159195934,
5.4772255750516665, 6.755244875897035, 1.8257418583505352, 2.556038601690798, 5.659799760886702, 2.9
2118697336086 84, -3.4689095308660653, -11.502173707608499, 6.572670690062, 4.199206274206298, 3.1037
611591959644, -10.406728592598165, -2.3734644158557323]
```



2 Домашнее задание. Основные понятия математической статистики

2.1 Геометрическое распределение

2.1.1 Моделирование выбранных случайных величин

In [75]:

```
# Создание случайной величины с геометрическим распределением, зависящим
# от параметра p
p = 0.5
geom_rv = sts.geom(p)
```

In [9]:

```
#Генерация выборки объема n = 5 с выводом
for n in [5]:
    means_5 = []
    for i in range(5):
        sample = geom_rv.rvs(n)
        means_5.append(sample)
        print(sample)
```

```
[2 2 3 1 1]
[6 1 2 6 4]
[1 3 3 2 2]
[2 1 3 3 1]
[2 1 1 1 5]
```

In [10]:

```
#Генерация выборки объема n = 10 с выводом
for n in [10]:
    means_10 = []
    for i in range(5):
        sample = geom_rv.rvs(n)
        means_10.append(sample)
        print(sample)
```

```
[1 1 1 1 3 4 1 1 3 1]
[4 1 2 3 1 4 1 1 2 2]
[2 1 2 7 1 1 1 1 1 3]
[2 3 1 1 1 1 1 3 3 4]
[1 4 2 1 2 1 2 1 1 2]
```

In [11]:

```
#Генерация выборки объема n = 100 ,без вывода
for n in [100]:
    means_100 = []
    for i in range(5):
        sample = geom_rv.rvs(n)
        means_100.append(sample)
```

In [12]:

```
#Генерация выборки объема n = 1000 ,без вывода
for n in [1000]:
    means_1000 = []
    for i in range(5):
        sample = geom_rv.rvs(n)
        means_1000.append(sample)
```

In [13]:

```
#Генерация выборки объема n = 100000 ,без вывода
for n in [100000]:
    means_100000 = []
    for i in range(5):
        sample = geom_rv.rvs(n)
        means_100000.append(sample)
```

2.1.2 Построение эмпирической функции распределения

In [14]:

```
#n=5
for a in range(5):
    b=means_5[a]
    b=sorted(b)
    print('Empirical distribution function F5(x) for sample',a+1,':')
    for i in range(4):
        if(i==0):
            n=0.
            g=1
            print(n,', x <=',b[i])
        if(b[i+1]==b[i]):
            g+=1
        else:
            n=round(n+0.2*g,1)
            g=1
            print(n,',',b[i],'< x <=',b[i+1])
        if(i==3):
            n=1.
            print(n,', x >',b[i+1])
```

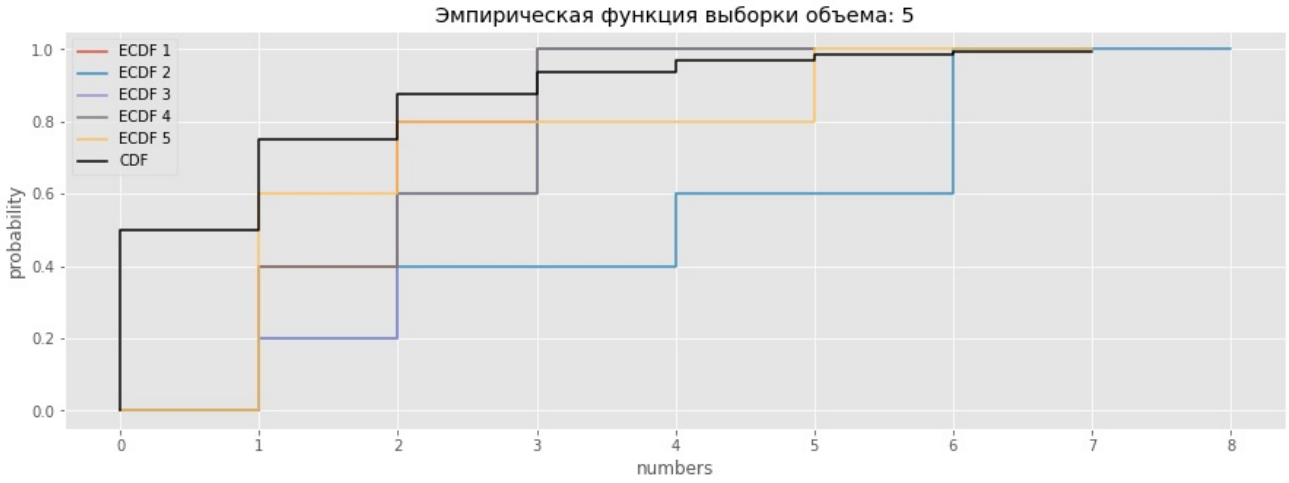
```
Empirical distribution function F5(x) for sample 1 :
0.0 , x <= 1
0.4 , 1 < x <= 2
0.8 , 2 < x <= 3
1.0 , x > 3
Empirical distribution function F5(x) for sample 2 :
0.0 , x <= 1
0.2 , 1 < x <= 2
0.4 , 2 < x <= 4
0.6 , 4 < x <= 6
1.0 , x > 6
Empirical distribution function F5(x) for sample 3 :
0.0 , x <= 1
0.2 , 1 < x <= 2
0.6 , 2 < x <= 3
1.0 , x > 3
Empirical distribution function F5(x) for sample 4 :
0.0 , x <= 1
0.4 , 1 < x <= 2
0.6 , 2 < x <= 3
1.0 , x > 3
Empirical distribution function F5(x) for sample 5 :
0.0 , x <= 1
0.6 , 1 < x <= 2
0.8 , 2 < x <= 5
1.0 , x > 5
```

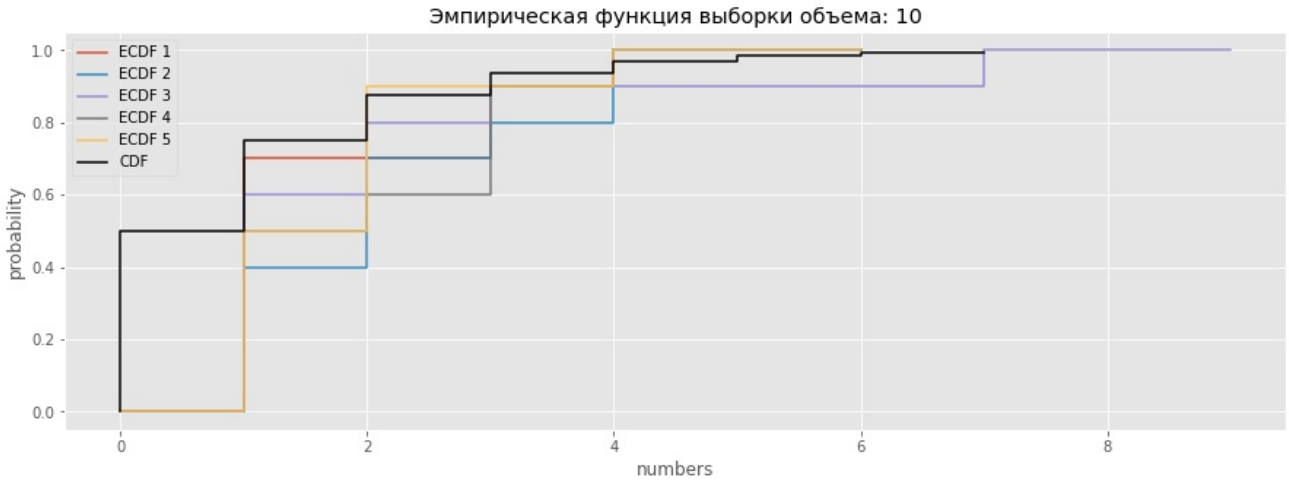
In [15]:

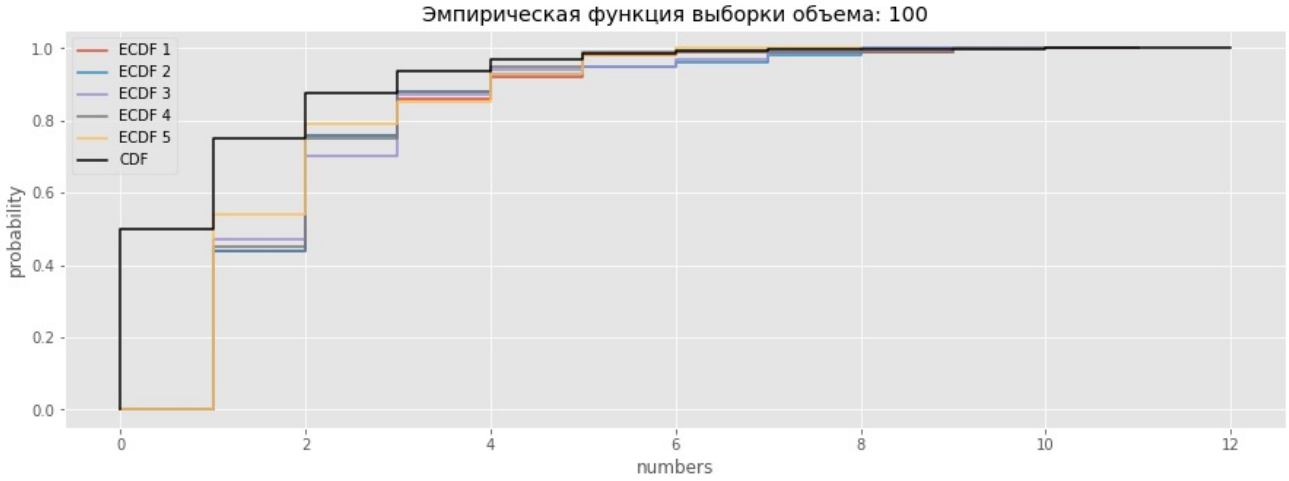
```
#n=5
for a in range(5):
    b=means_5[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range(b[v-1]):
        N.append(b.count(i))
        x1=[]
        y1=[]
        t=0
    for i in range(b[v-1]):
        t+=N[i]
        x1.append(i)
        y1.append(t/v)
        x1.append(i+1)
        y1.append(t/v)
    x1.append(b[v-1])
    y1.append(1)
    x1.append(b[v-1]+2)
    y1.append(1)
    plt.plot(x1,y1,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
n=np.arange(0,8,1)#Построение
plt.step(n,1-(1-p)**(n),'k-', label='CDF')#теоретическойфункции
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
print("\n")
#n=10
for a in range(5):
```

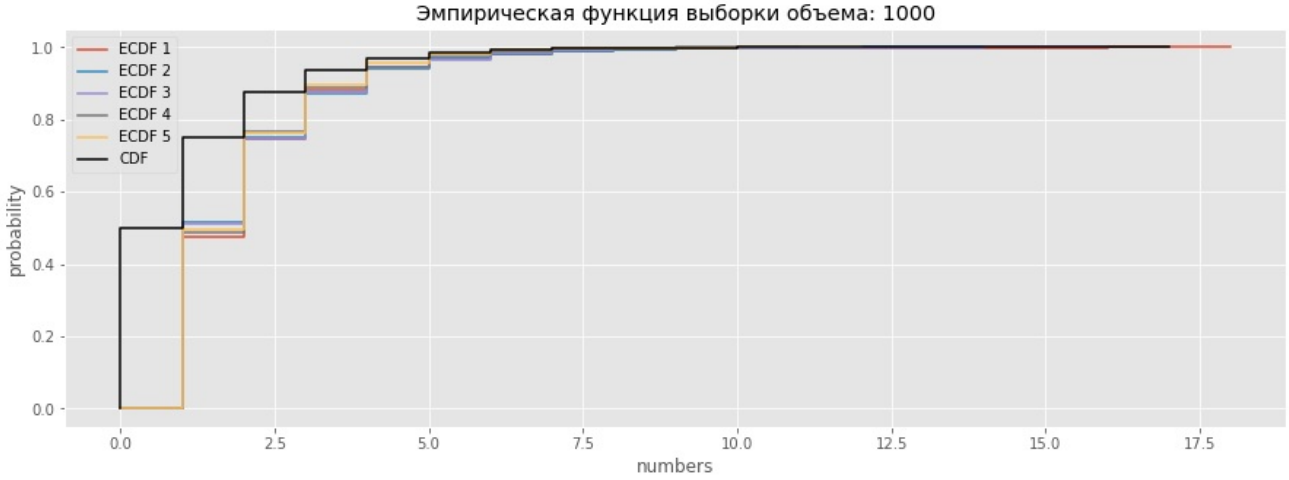
```
for a in range(5):
    b=means_10[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range(b[v-1]):
        N.append(b.count(i))
        x2=[]
        y2=[]
        t=0
    for i in range(b[v-1]):
        t+=N[i]
        x2.append(i)
        y2.append(t/v)
        x2.append(i+1)
        y2.append(t/v)
    x2.append(b[v-1])
    y2.append(1)
    x2.append(b[v-1]+2)
    y2.append(1)
    plt.plot(x2,y2,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
n=np.arange(0,8,1)#Построение
plt.step(n,1-(1-p)**(n),'k-', label='CDF')#теоретическойфункции
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
print("\n")
#n=100
for a in range(5):
    b=means_100[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range(b[v-1]):
        N.append(b.count(i))
        x3=[]
        y3=[]
        t=0
    for i in range(b[v-1]):
        t+=N[i]
        x3.append(i)
        y3.append(t/v)
        x3.append(i+1)
        y3.append(t/v)
    x3.append(b[v-1])
    y3.append(1)
    x3.append(b[v-1]+2)
    y3.append(1)
    plt.plot(x3,y3,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
n=np.arange(0,13,1)#Построение
plt.step(n,1-(1-p)**(n),'k-', label='CDF')#теоретическойфункции
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
print("\n")
#n=1000
for a in range(5):
    b=means_1000[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range(b[v-1]):
        N.append(b.count(i))
        x4=[]
        y4=[]
        t=0
    for i in range(b[v-1]):
        t+=N[i]
        x4.append(i)
        y4.append(t/v)
        x4.append(i+1)
        y4.append(t/v)
    x4.append(b[v-1])
    y4.append(1)
    x4.append(b[v-1]+2)
    y4.append(1)
    plt.plot(x4,y4,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
```

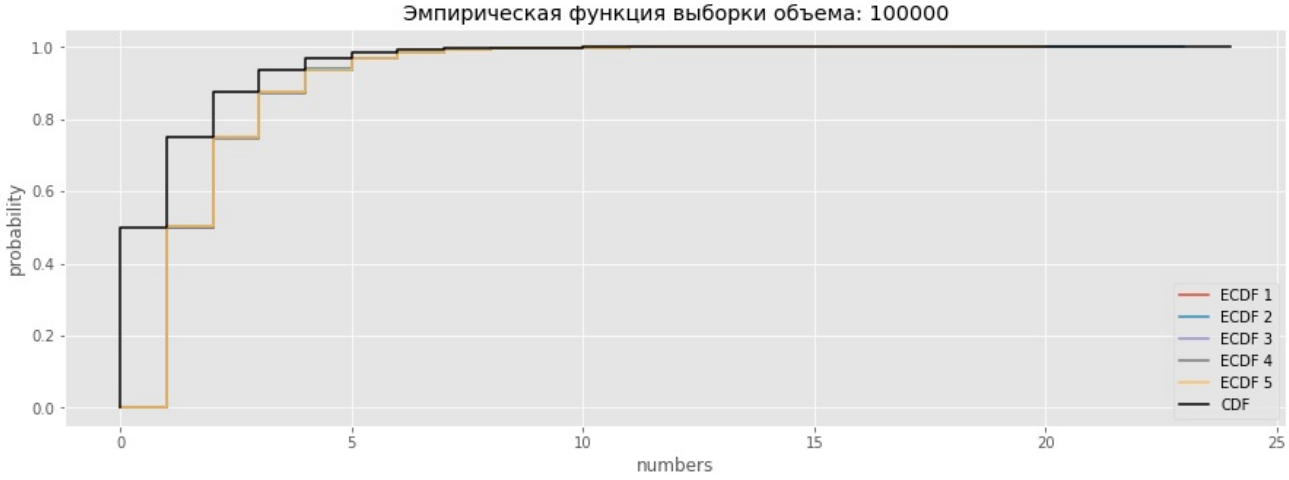
```
plt.title("Эмпирическая функция выборки объема: "+str(v))

n=np.arange(0,18,1)#Построение
plt.step(n,1-(1-p)**(n), 'k-', label='CDF')#теоретическойфункции
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
print("\n")
#n=100000
for a in range(5):
    b=means_100000[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range(b[v-1]):
        N.append(b.count(i))
        x5=[]
        y5=[]
        t=0
    for i in range(b[v-1]):
        t+=N[i]
        x5.append(i)
        y5.append(t/v)
        x5.append(i+1)
        y5.append(t/v)
    x5.append(b[v-1])
    y5.append(1)
    x5.append(b[v-1]+2)
    y5.append(1)
    plt.plot(x5,y5,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
n=np.arange(0,25,1)#Построение
plt.step(n,1-(1-p)**(n), 'k-', label='CDF')#теоретическойфункции
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
print("\n")
```






Эмпирическая функция выборки объема: 100
ECDF 1
ECDF 2
ECDF 3
ECDF 4
ECDF 5
CDF
probability
numbers
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
12


Эмпирическая функция выборки объема: 1000
ECDF 1
ECDF 2
ECDF 3
ECDF 4
ECDF 5
CDF
probability
numbers
0.0
0.2
0.4
0.6
0.8
1.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Эмпирическая функция выборки объема: 100000
ECDF 1
ECDF 2
ECDF 3
ECDF 4
ECDF 5
CDF
probability
numbers
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25

Пусть $X = (X_1, \ldots, X_n)$- выборка из дискретного распределения $\sigma(\xi)$ Величина скачка в точке j есть

$$\Delta\hat{F}_n(j) = \hat{F}_n(j) - \hat{F}_n(j-0) = \frac{v}{n},$$

$j = 1, \ldots, N$

Здесь $P\{\Delta\hat{F}_n(j) = 0\} = P(v_j = 0) = (1 - p_j)^n$ что мало при больших n, т.е. в большой выборке скачок в точке j наверняка будет иметь место. Более того, так как $P\{\cup_{j=1}^{N}\{\Delta\hat{F}_n(j) = 0\}\} \leq \sum_{j=1}^{N}(1 - p_j)^n \to 0,$ при $n \to \infty$, то в больших выборках с вероятностью, близкой к 1, скачки э.ф.р. $F_n(x)$ будут иметь место во всех точках 1,2,...,N, а случайными будут лишь величины этих скачков.

Если же теоретическая функция распределения $F_\xi = F(x)$ непрерывна, то с вероятностью 1 все элементы выборки $X = (X_1, \ldots, X_n)$ будут различны, и случайными теперь будут точки скачков, величины же скачков неслучайны и равны $\frac{1}{n}$

Таким образом, для выборок из дискретных и непрерывных распределений характер соответствующих эмпирических функций распределениябудет различным, что можно заметить на получившихся графиков для дискретного и непрерывного распределения. Тем не менее в любомслучае э.ф.р. $\hat{F}_n(x)$ с увеличением объема выборки n сближается в каждой точке x с теоретической функцией распределения F(x).

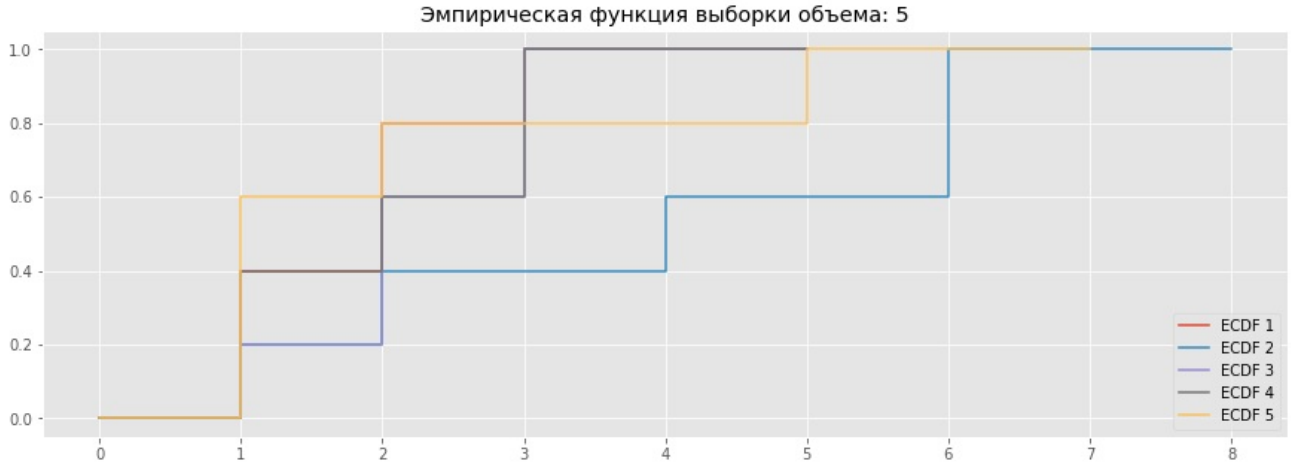
Максимальная точная верхняя граница разности пары эмпирических функций распределения - наибольшая разность между значениямивероятности двух функций в одной точке.

Неудачная попытка рассчитать верхнюю границу разности э.ф.р. выборок размера n = 5, так как у меня получилось 0, что неверно.

In [16]:

```
s=0
for a in range(5):
    b=means_5[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range(b[v-1]):
        N.append(b.count(i))
        x1=[]
        y1=[]
        t=0
        x_=[]
        y_=[]
    for i in range(b[v-1]):
        t+=N[i]
        x1.append(i)
        y1.append(t/v)
        x1.append(i+1)
        y1.append(t/v)
        if(t!=0):
            sch=0
            if(len(y1)>len(y_)):
                for j in y_:
                    if(math.fabs(y1[sch]-y_[sch])>s):
                        s=math.fabs(y1[sch]-y_[sch])
                    sch+=1
            else:
                for j in y1:
                    if(math.fabs(y1[sch]-y_[sch]>s)):
                        s=math.fabs(y1[sch]-y_[sch])
                    sch+=1
        y_=copy.copy(y1)
    x1.append(b[v-1])
    y1.append(1)
    x1.append(b[v-1]+2)
    y1.append(1)
    #x_ = copy.copy(x1)
    #y_ = copy.copy(y1)
    plt.plot(x1,y1,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
print("Верхняя граница разности э.ф.р. выборок размера n = 5 :",s)
```

```
Верхняя граница разности э.ф.р. выборок размера n = 5 : 0
```



Верхняя граница разности э.ф.р. выборок размера n = 5 : 0.600
Верхняя граница разности э.ф.р. выборок размера n = 10 : 0.400
С увеличением объема выборки верхняя граница разности уменьшается, что очевидно.

2.1.3 Построение вариационного ряда выборки

Определение:
Пусть $X = (X_1, \ldots, X_n)$- выборка из некоторого распределения $\sigma(\xi)$
Произвольной реализации $x = (x_1, \ldots, x_n)$ этой выборки можно поставить в соответствие упорядоченную последовательность

$$x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(n)}$$

располагая $x_1, \ldots, x_n$ в порядке их возрастания, так что $x_{(1)} = min\{x_1, \ldots, x_n\}, x_{(2)}$ - второе по величине значение, $x_{(n)} = max\{x_1, \ldots, x_n\}$
Обозначим через $X_{(k)}$ случайную величину, которая для каждой реализации выборки X принимает значение $x_{(k)}, k = 1, \ldots, n$. Так по выборке X определяют новую последовательность случайных величин $X_{(1)}, \ldots, X_{(n)}$, называемых порядковыми статистиками выборки. Из определения порядковых статистик следует, что они упорядочены по возрастанию их значений, т.е. они образуют возрастающую последовательность

$$X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(n)},$$

которая называется вариационным рядом выборки X.

In [17]:

```
#Вариационный ряд для выборки объема n=5 с выводом
for a in range(5):
    b=means_5[a]
    b=sorted(b)
    print(b)
```

```
[1, 1, 2, 2, 3]
[1, 2, 4, 6, 6]
[1, 2, 2, 3, 3]
[1, 1, 2, 3, 3]
[1, 1, 1, 2, 5]
```

In [18]:

```
#Вариационный ряд для выборки объема n=10 с выводом
for a in range(5):
    b=means_10[a]
    b=sorted(b)
    print(b)
```

```
[1, 1, 1, 1, 1, 1, 1, 3, 3, 4]
[1, 1, 1, 1, 2, 2, 2, 3, 4, 4]
[1, 1, 1, 1, 1, 1, 2, 2, 3, 7]
[1, 1, 1, 1, 1, 2, 3, 3, 3, 4]
[1, 1, 1, 1, 1, 2, 2, 2, 2, 4]
```

In [19]:

```
#Вариационный ряд для выборки объема n=100 без вывода
for a in range(5):
    b=means_100[a]
    b=sorted(b)
```

In [20]:

```
#Вариационный ряд для выборки объема n=1000 без вывода
for a in range(5):
    b=means_1000[a]
    b=sorted(b)
```

In [21]:

```
#Вариационный ряд для выборки объема n=100000 без вывода
for a in range(5):
    b=means_100000[a]
    b=sorted(b)
```

Определение:
α- квантиль случайной величины ξ с функцией распределения $F(x) = P\{\xi < x\}$ — это любое число x_α, удовлетворяющее двум условиям:
$1) F(x_\alpha) \leq \alpha \; 2) F(x_\alpha + 0) \geq \alpha.$
Исходя из того, что при больших выборках э.ф.р. стремится к теоритической функции распределения, эмпирические квантили так же стремятся к теоритическим по определению. Пусть F(x) - функция распределения. Тогда квантильная функция:
$F^{-1}(r) = min\{x \in N_+ : F(x) \geq r\} for r \in (0; 1)$
$F^{-1}(r) = [\frac{ln(1-r)}{ln(1-p}]$

In [22]:

```
k = 1
for b in [means_5[a], means_10[a],means_100[a],means_1000[a],means_100000[a]]:
    if(k==1):
        print('n = 5')
    if(k==2):
        print('n = 10')
    if(k==3):
        print('n = 100')
    if(k==4):
        print('n = 1000')
    if(k==5):
        print('n = 100000')
    print(np.quantile(b, 0.1))
    k += 1
```

```
n = 5
1.0
n = 10
1.0
n = 100
1.0
n = 1000
1.0
n = 100000
1.0
```

In [23]:

```
#Сравнение
np.log(1-0.1)//np.log(1-p)
```

Out[23]:

```
0.0
```

In [24]:

```
k = 1
for b in [means_5[a], means_10[a],means_100[a],means_1000[a],means_100000[a]]:
    if(k==1):
        print('n = 5')
    if(k==2):
        print('n = 10')
    if(k==3):
        print('n = 100')
    if(k==4):
        print('n = 1000')
    if(k==5):
        print('n = 100000')
    print(np.quantile(b, 0.5))
    k += 1
```

```
n = 5
1.0
n = 10
1.5
n = 100
1.0
n = 1000
2.0
n = 100000
1.0
```

In [25]:

```
#Сравнение
geom.median(p)
```

Out[25]:

```
1.0
```

In [26]:

```
#Сравнение
np.log(1-0.5)//np.log(1-p)
```

Out[26]:

```
1.0
```

In [27]:

```
k = 1
for b in [means_5[a], means_10[a],means_100[a],means_1000[a],means_100000[a]]:
    if(k==1):
        print('n = 5')
    if(k==2):
        print('n = 10')
    if(k==3):
        print('n = 100')
    if(k==4):
        print('n = 1000')
    if(k==5):
        print('n = 100000')
    print(np.quantile(b, 0.7))
    k += 1
```

```
n = 5
1.7999999999999998
n = 10
2.0
n = 100
2.0
n = 1000
2.0
n = 100000
2.0
```

In [28]:

```
#Сравнение
np.log(1-0.7)//np.log(1-p)
```

Out[28]:

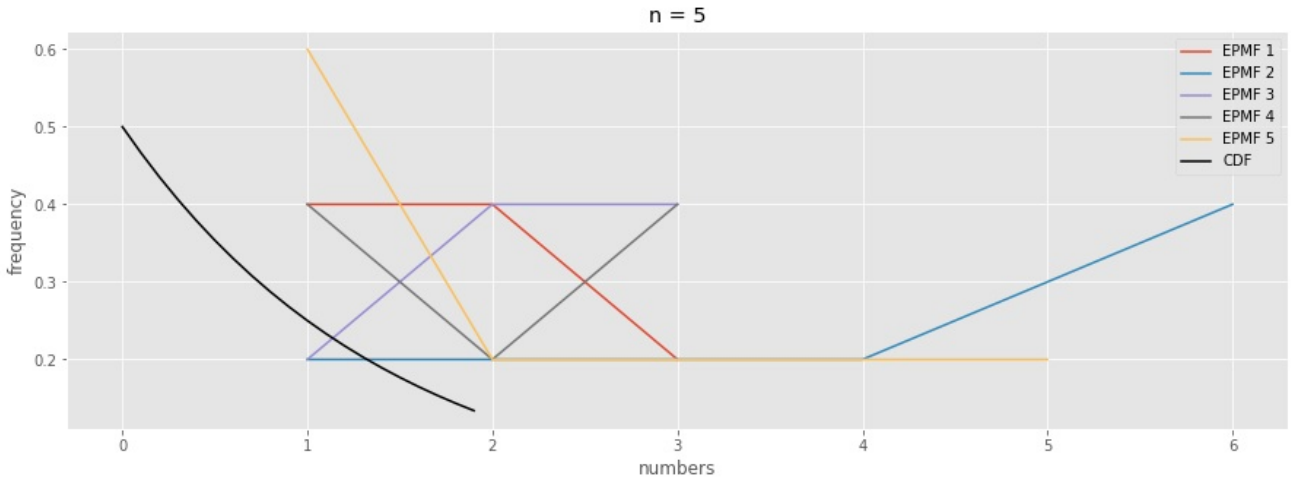
```
1.0
```

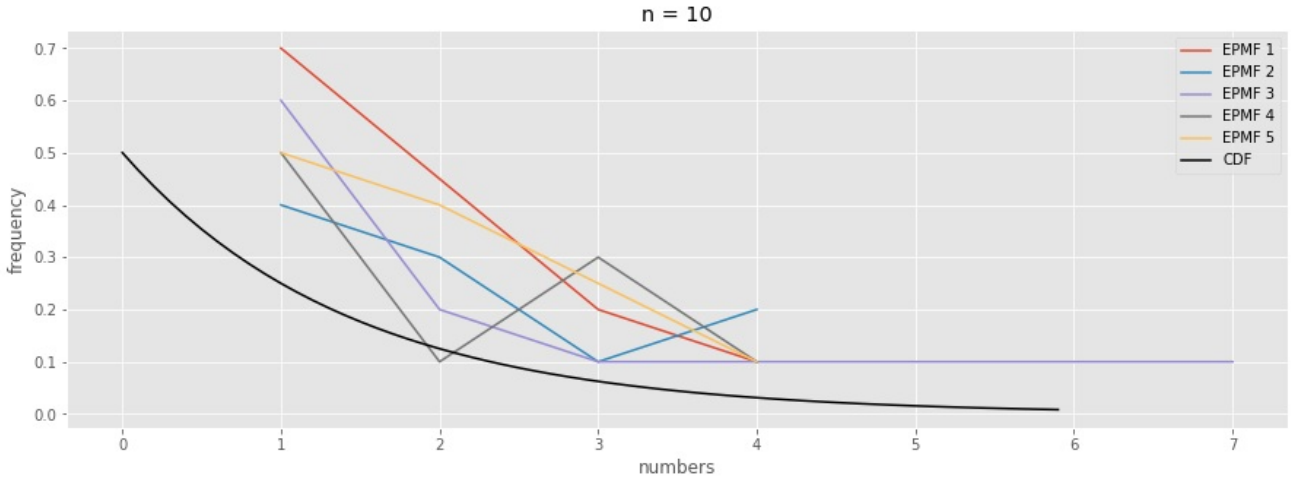
2.1.4 Построение гистограммы и полигона частот

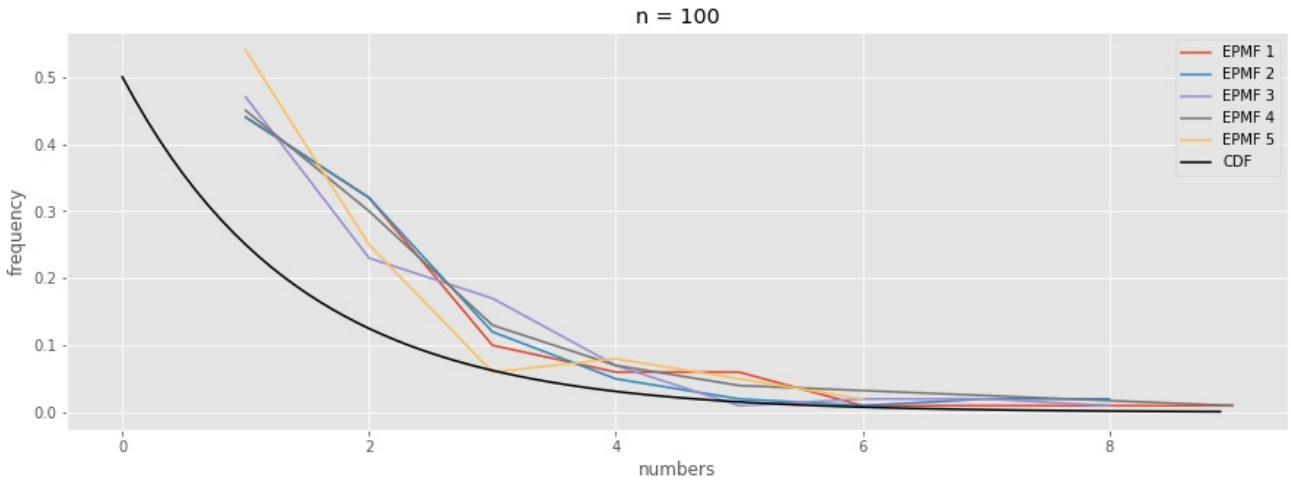
In [29]:

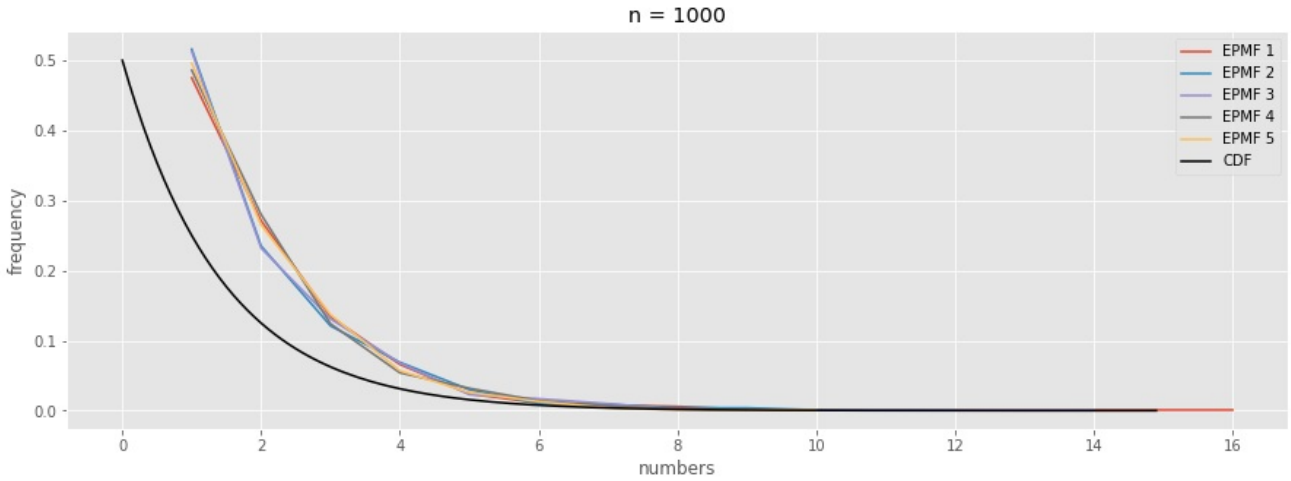
```
#n=5
for a in range(5):
    b=means_5[a]
    b=sorted(b)
    x=[]
    y=[]
    c=Counter(b)
    for i in c:
        x.append(i)
        y.append(b.count(i)/5.0)
    plt.plot(x,y,label="EPMF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("n = 5")
n=np.arange(0,2,0.1)#Построение
plt.plot(n,p*(1-p)**(n),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=10
for a in range(5):
    b=means_10[a]
    b=sorted(b)
    x=[]
    y=[]
    c=Counter(b)
    for i in c:
        x.append(i)
        y.append(b.count(i)/10.0)
    plt.plot(x,y,label="EPMF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("n = 10")
n=np.arange(0,6,0.1)#Построение
plt.plot(n,p*(1-p)**(n),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=100
for a in range(5):
    b=means_100[a]
    b=sorted(b)
```

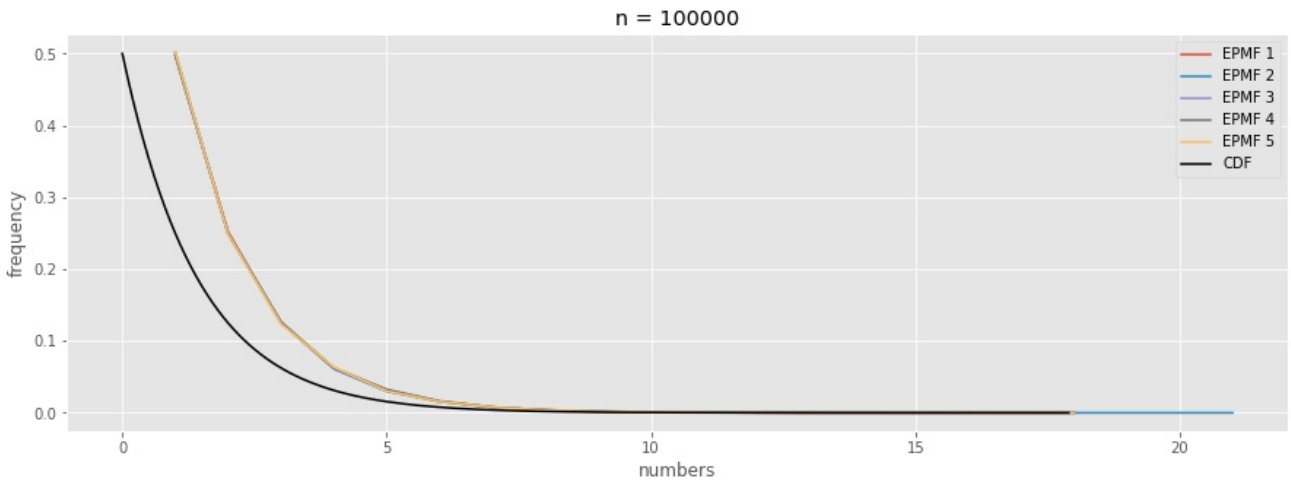
```
    b=sorted(b)
    x=[]
    y=[]
    c=Counter(b)
    for i in c:
        x.append(i)
        y.append(b.count(i)/100.0)
    plt.plot(x,y,label="EPMF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("n = 100")
n=np.arange(0,9,0.1)#Построение
plt.plot(n,p*(1-p)**(n),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=1000
for a in range(5):
    b=means_1000[a]
    b=sorted(b)
    x=[]
    y=[]
    c=Counter(b)
    for i in c:
        x.append(i)
        y.append(b.count(i)/1000.0)
    plt.plot(x,y,label="EPMF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("n = 1000")
n=np.arange(0,15,0.1)#Построение
plt.plot(n,p*(1-p)**(n),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=100000
for a in range(5):
    b=means_100000[a]
    b=sorted(b)
    x=[]
    y=[]
    c=Counter(b)
    for i in c:
        x.append(i)
        y.append(b.count(i)/100000.0)
    plt.plot(x,y,label="EPMF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("n = 100000")
n=np.arange(0,18,0.1)#Построение
plt.plot(n,p*(1-p)**(n),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```




n = 10
EPMF 1
EPMF 2
EPMF 3
EPMF 4
EPMF 5
CDF
frequency
numbers

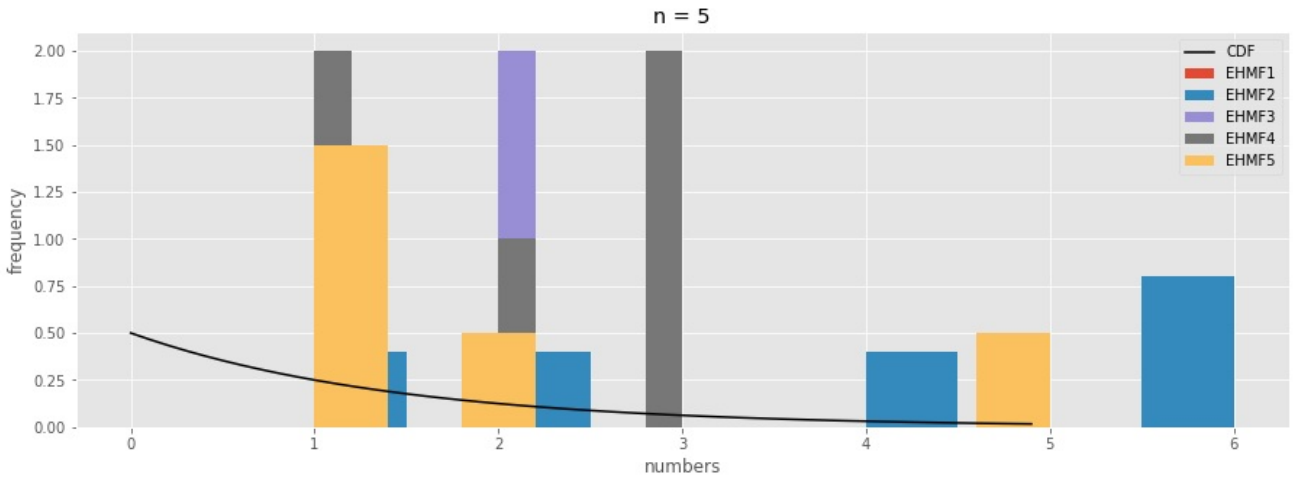

n = 100
EPMF 1
EPMF 2
EPMF 3
EPMF 4
EPMF 5
CDF
frequency
numbers

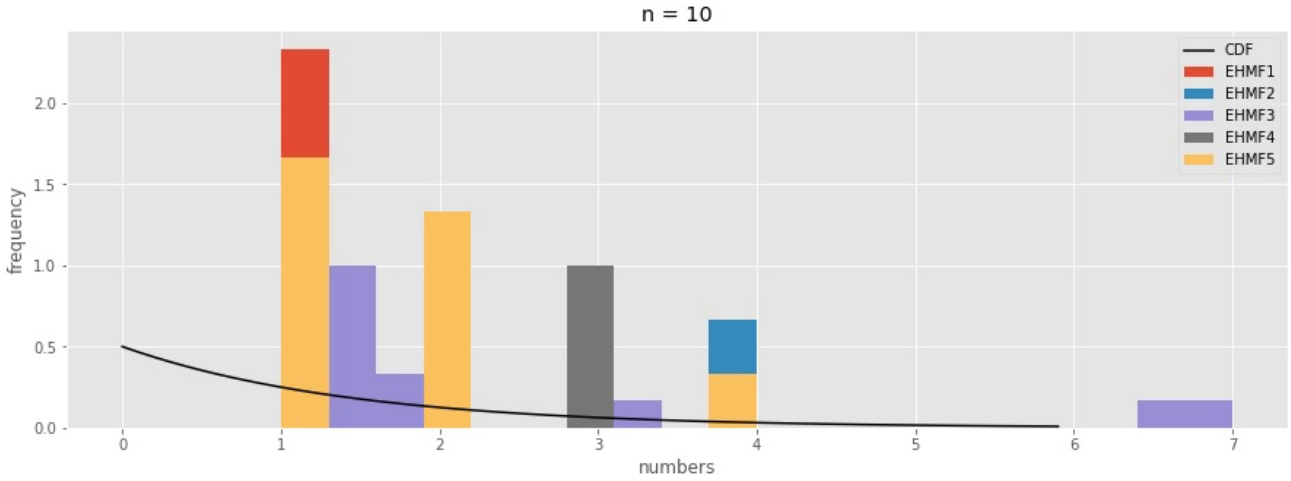

n = 1000
EPMF 1
EPMF 2
EPMF 3
EPMF 4
EPMF 5
CDF
frequency
numbers

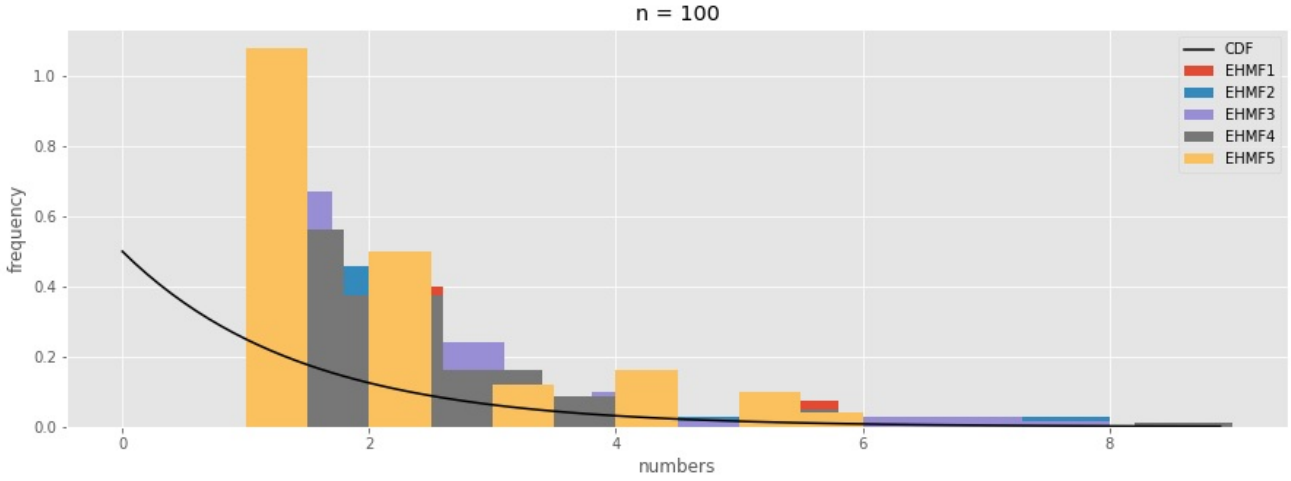

n = 100000
EPMF 1
EPMF 2
EPMF 3
EPMF 4
EPMF 5
CDF
frequency
numbers

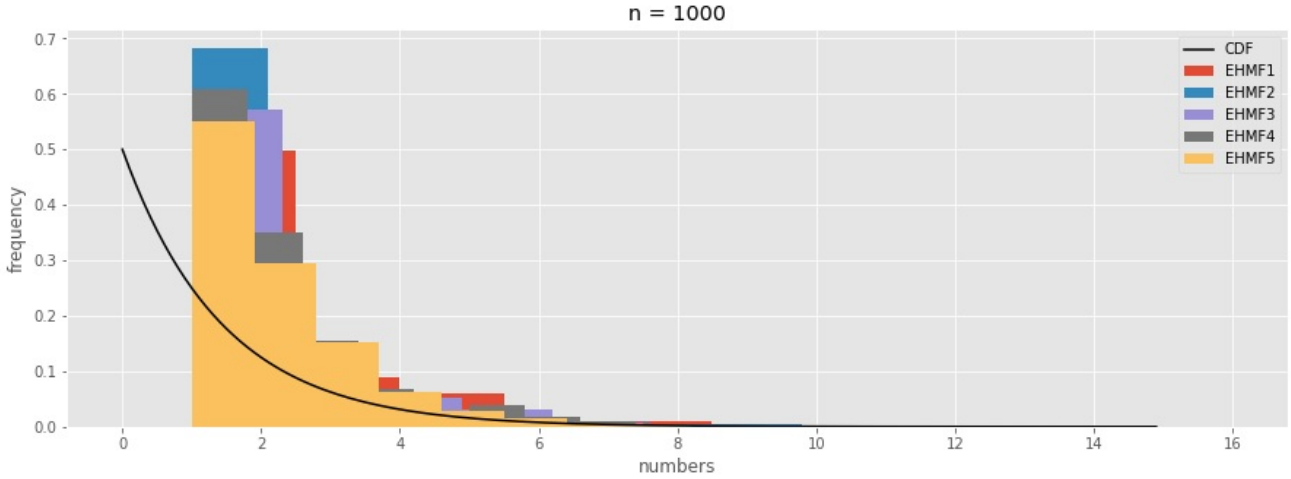
In [30]:

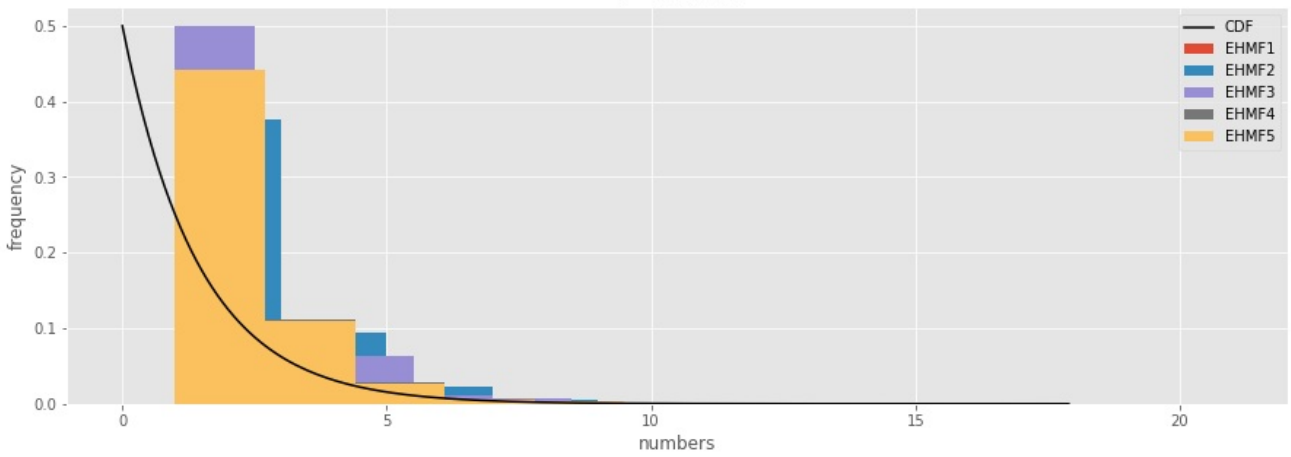
```
#n=5
for a in range(5):
    plt.hist(means_5[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
n=np.arange(0,5,0.1)#Построение
plt.plot(n,p*(1-p)**(n),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.title("n = 5")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=10
for a in range(5):
    plt.hist(means_10[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
n=np.arange(0,6,0.1)#Построение
plt.plot(n,p*(1-p)**(n),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.title("n = 10")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=100
for a in range(5):
    plt.hist(means_100[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
n=np.arange(0,9,0.1)#Построение
plt.plot(n,p*(1-p)**(n),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.title("n = 100")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=1000
for a in range(5):
    plt.hist(means_1000[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
n=np.arange(0,15,0.1)#Построение
plt.plot(n,p*(1-p)**(n),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.title("n = 1000")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=100000
for a in range(5):
    plt.hist(means_100000[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
n=np.arange(0,18,0.1)#Построение
plt.plot(n,p*(1-p)**(n),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.title("n = 100000")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```




n = 10
CDF
EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
frequency
numbers


n = 100
CDF
EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
frequency
numbers


n = 1000
CDF
EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
frequency
numbers


n = 100000
CDF
EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
frequency
numbers

Если наблюдаемая в эксперименте случайная величина ξ дискретна и принимает значения $a_1, a_2, \ldots$, то более наглядное представление о ее законе распределения дадут относительные частоты $v_r^* = \frac{v_r}{n}$, где v_r - число элементов выборки $X = (X_1, \ldots, X_n)$, принявших значение a_r: $v_r = \sum_{j=1}^{n} I(X_j = a_r), r = 1, 2, \ldots$, т.е. v_r^* сближается с ростом n с теоретической вероятностью $P\{\xi = a_r\}$, и потому, по крайней мере для больших выборок, относительные частоты v_r^* можно рассматривать в качестве приближенных значений (оценок) для неизвестных вероятностей $P\{\xi = a_r\}$.
Наглядным представлением данных является полигон частот, который представляет собой ломаную с вершинами в точках $(a_r; v_r), r = 1, 2, \ldots$
Можно рассматривать также статистический ряд {{$(a_r; v_r)$}}.
На графиках выше наглядно подтверждаются наши теоретические знания.

2.2 Распределение Максвелла

2.2.1 Моделирование выбранных случайных величин

In [31]:

```
# Создание случайной величины с распределением Максвелла, зависящим
# от параметра lambda
lambd=1.0
maxwell_rv=sts.maxwell(scale=lambd)
```

In [32]:

```
#Генерация выборки объема n = 5 с выводом
for n in[5]:
    means__5=[]
    for i in range(5):
        sample=maxwell_rv.rvs(n)
        means__5.append(sample)
        print(sample)
```

```
[1.81532796 1.24706856 1.90436227 2.96600488 1.32430324]
[0.90130832 2.40574367 1.32340085 1.37806689 0.23917746]
[0.85574397 0.896143   1.9855106  1.93659757 2.18636079]
[0.87263024 1.44555386 0.70580869 1.37176173 2.0228521 ]
[1.57346381 2.93557089 1.17503094 0.98464114 2.38578258]
```

In [33]:

```
#Генерация выборки объема n = 10 с выводом
for n in[10]:
    means__10=[]
    for i in range(5):
        sample=maxwell_rv.rvs(n)
        means__10.append(sample)
        print(sample)
```

```
[1.26636245 1.7807016  2.9419484  1.19330058 1.13967485 1.18530103
 1.35675794 2.8040828  2.56665123 1.39780498]
[1.89867962 0.8087531  1.2622001  1.42572836 0.62763719 1.65591953
 2.16348264 2.21258969 0.72110622 1.32946195]
[0.65774779 1.24607998 1.43685579 2.05247404 1.63119034 1.05587414
 1.22291086 1.52417635 1.83152899 0.84995274]
[0.8943084  1.74874446 2.20149536 1.31877633 1.02998226 1.26469672
 2.94863991 1.64189344 1.5184008  0.6959279 ]
[0.5376665  1.69858016 2.60588875 1.81281287 1.71742912 2.57790948
 1.43968108 1.81886915 1.91245001 2.73423483]
```

In [34]:

```
#Генерация выборки объема n = 100 без вывода
for n in[100]:
    means__100=[]
    for i in range(5):
        sample=maxwell_rv.rvs(n)
        means__100.append(sample)
```

In [35]:

```
#Генерация выборки объема n = 1000 без вывода
for n in[1000]:
    means__1000=[]
    for i in range(5):
        sample=maxwell_rv.rvs(n)
        means__1000.append(sample)
```

In [36]:

```
#Генерация выборки объема n = 100000 без вывода
for n in[100000]:
    means__100000=[]
    for i in range(5):
        sample=maxwell_rv.rvs(n)
        means__100000.append(sample)
```

In [37]:

```
#Вернёмся к медиане и убедимся, что в пункте 1.2.1 она была найдена верно
maxwell.median()
```

Out[37]:

```
1.5381722544550522
```

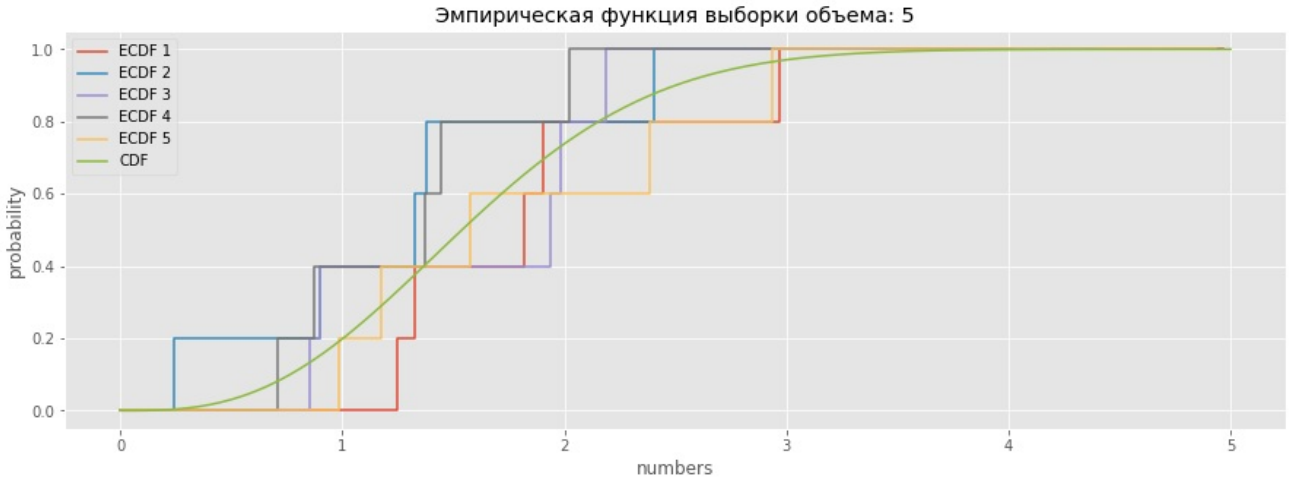
2.2.2 Построение эмпирической функции распределения

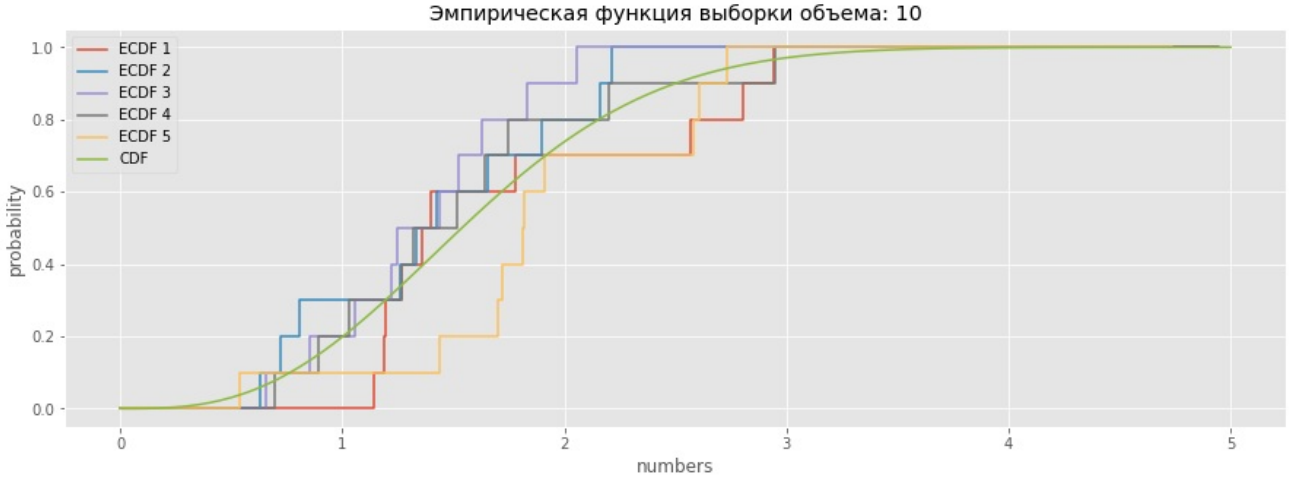
In [38]:

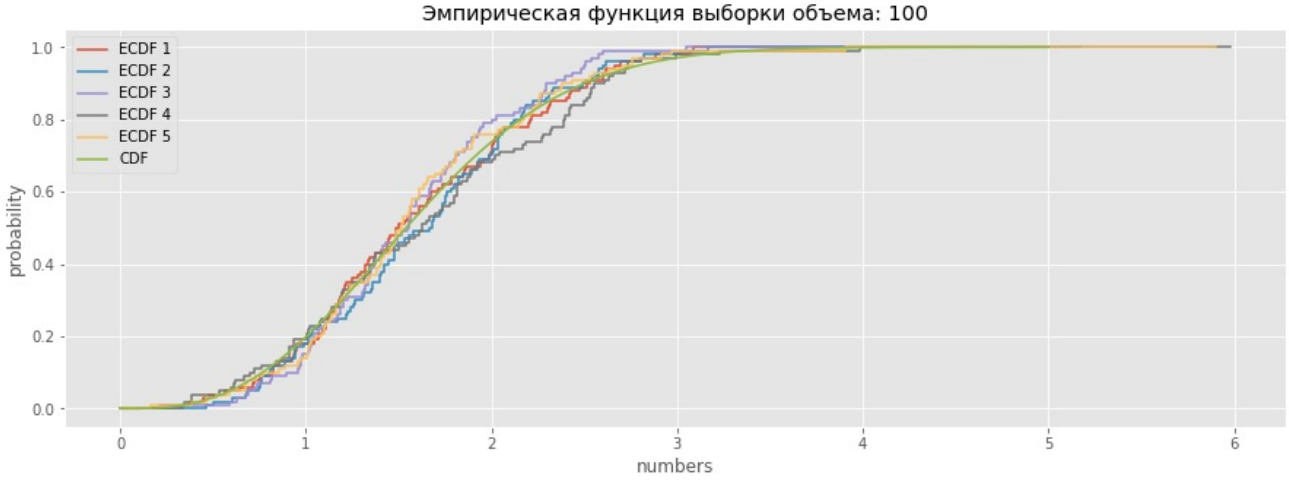
```
#n=5
for a in range(5):
    b=means__5[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range((v-1)):
        N.append(b.count(b[i]))
        x=[]
        y=[]
        t=0
        x.append(0.0)
        y.append(0.0)
        x.append(b[0])
        y.append(0.0)
    for i in range((v-1)):
        t+=N[i]
        x.append(b[i])
        y.append(float(t/v))
        x.append(b[i+1])
        y.append(float(t/v))
    x.append(b[v-1])
    y.append(1)
    x.append(b[v-1]+2)
    y.append(1)
    plt.plot(x,y,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
x=np.linspace(0,5,100)#Построение
cdf=maxwell_rv.cdf(x)#теоретической функции
plt.plot(x,cdf,label='CDF')#распределения
plt.legend()
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
print("\n")
#n=10
for a in range(5):
    b = means__10[a]
    b = sorted(b)
    v = len(b)
    N = []
    for i in range((v-1)):
        N.append(b.count(b[i]))
        x=[]
        y=[]
        t=0
        x.append(0.0)
        y.append(0.0)
        x.append(b[0])
        y.append(0.0)
    for i in range((v-1)):
        t+=N[i]
        x.append(b[i])
        y.append(float(t/v))
        x.append(b[i+1])
        y.append(float(t/v))
    x.append(b[v-1])
    y.append(1)
    x.append(b[v-1]+2)
```

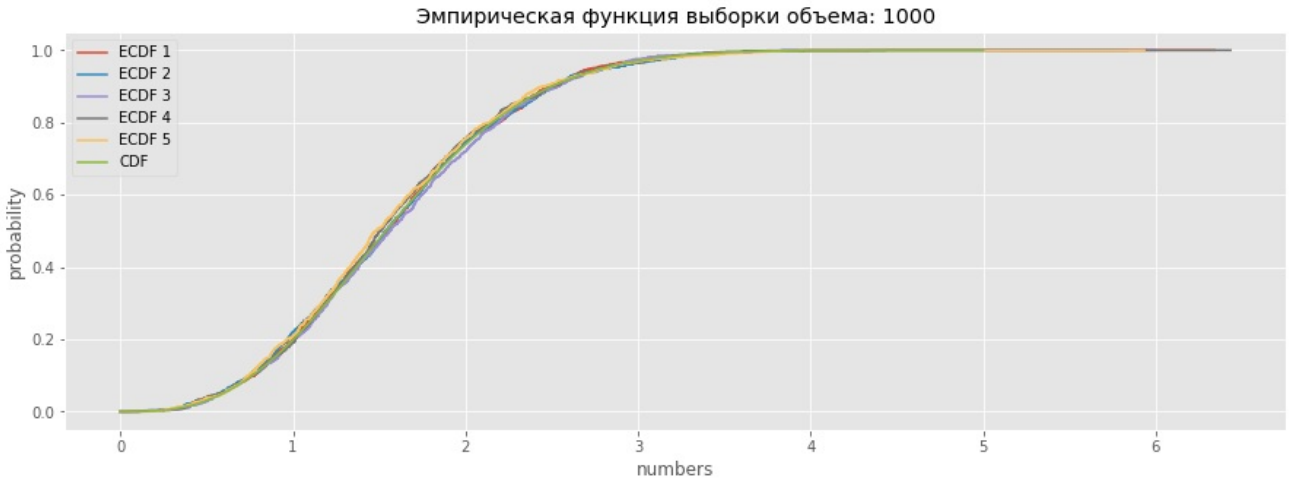
```
    y.append(1)

    plt.plot(x,y,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
x=np.linspace(0,5,100)#Построение
cdf=maxwell_rv.cdf(x)#теоретической функции
plt.plot(x,cdf,label='CDF')#распределения
plt.legend()
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
print("\n")
#n=100
for a in range(5):
    b=means__100[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range((v-1)):
        N.append(b.count(b[i]))
        x=[]
        y=[]
        t=0
        x.append(0.0)
        y.append(0.0)
        x.append(b[0])
        y.append(0.0)
    for i in range((v-1)):
        t+=N[i]
        x.append(b[i])
        y.append(float(t/v))
        x.append(b[i+1])
        y.append(float(t/v))
    x.append(b[v-1])
    y.append(1)
    x.append(b[v-1]+2)
    y.append(1)
    plt.plot(x,y,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
x=np.linspace(0,5,100)#Построение
cdf=maxwell_rv.cdf(x)#теоретической функции
plt.plot(x,cdf,label='CDF')#распределения
plt.legend()
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
print("\n")
#n=1000
for a in range(5):
    b=means__1000[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range((v-1)):
        N.append(b.count(b[i]))
        x=[]
        y=[]
        t=0
        x.append(0.0)
        y.append(0.0)
        x.append(b[0])
        y.append(0.0)
    for i in range((v-1)):
        t+=N[i]
        x.append(b[i])
        y.append(float(t/v))
        x.append(b[i+1])
        y.append(float(t/v))
    x.append(b[v-1])
    y.append(1)
    x.append(b[v-1]+2)
    y.append(1)
    plt.plot(x,y,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
x=np.linspace(0,5,100)#Построение
cdf=maxwell_rv.cdf(x)#теоритической функции
plt.plot(x,cdf,label='CDF')#распределения
plt.legend()
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
```

```
print("\n")
```


Эмпирическая функция выборки объема: 5
ECDF 1
ECDF 2
ECDF 3
ECDF 4
ECDF 5
CDF
probability
numbers


Эмпирическая функция выборки объема: 10
ECDF 1
ECDF 2
ECDF 3
ECDF 4
ECDF 5
CDF
probability
numbers


Эмпирическая функция выборки объема: 100
ECDF 1
ECDF 2
ECDF 3
ECDF 4
ECDF 5
CDF
probability
numbers



2.2.3 Построение вариационного ряда выборки

In [39]:

```
#Вариационный ряд для выборки объема n=5 с выводом
for a in range(5):
    b=means__5[a]
    b=sorted(b)
    print(b)
```

```
[1.2470685591624504, 1.3243032409827724, 1.81532796453434, 1.9043622661494823, 2.9660048838677433]
[0.23917745542919205, 0.9013083202772701, 1.323400853813342, 1.3780668910363136, 2.405743673315854]
[0.8557439745838243, 0.8961430028515255, 1.9365975712097148, 1.9855106028094072, 2.1863607864926924]
[0.7058086904402556, 0.8726302384455435, 1.3717617301905698, 1.44555385597942, 2.02285209747939]
[0.984641138172079, 1.1750309353714015, 1.5734638080034873, 2.3857825813602567, 2.935570894480867]
```

In [40]:

```
#Вариационный ряд для выборки объема n=10 с выводом
for a in range(5):
    b=means__10[a]
    b=sorted(b)
    print(b)
```

```
[1.1396748537973511, 1.1853010283847474, 1.193300580721856, 1.2663624548257295, 1.3567579439438262,
1.3978049781213124, 1.7807016011851546, 2.56665123131523, 2.8040828014749173, 2.9419484021294124]
[0.6276371930782778, 0.7211062191171548, 0.8087531042396046, 1.2622001049199714, 1.329461954375532,
1.4257283613190956, 1.655919533282689, 1.8986796248625102, 2.163482642604677, 2.212589691477496]
[0.65774778847095, 0.8499527411663018, 1.0558741356809438, 1.2229108593395108, 1.2460799809756997, 1
.4368557912413, 1.5241763544555065, 1.6311903404068715, 1.831528986714442, 2.0524740394388683]
[0.6959279027350386, 0.8943083994020349, 1.0299822619136827, 1.264696716643979, 1.3187763303352453,
1.5184007980158851, 1.6418934445778848, 1.7487444584840308, 2.2014953551128187, 2.9486399107505172]
[0.5376665033110863, 1.4396810825984496, 1.6985801565913599, 1.7174291249219489, 1.8128128696770405,
1.8188691543104352, 1.9124500084986826, 2.577909477943119, 2.6058887514499225, 2.7342348308520763]
```

In [41]:

```
#Вариационный ряд для выборки объема n=100 без вывода
for a in range(5):
    b=means__100[a]
    b=sorted(b)
```

In [42]:

```
#Вариационный ряд для выборки объема n=1000 без вывода
for a in range(5):
    b=means__1000[a]
    b=sorted(b)
```

In [43]:

```
#Вариационный ряд для выборки объема n=100000 без вывода
for a in range(5):
    b=means__100000[a]
    b=sorted(b)
```

Возникли сложности при вычислении теоретических значений квантилей, однако был найден справочник:"Справочник по вероятностным распределениям" Р.Н.Вадзинский. В нём была найдена таблица для приближенного решения уравнения $x_\alpha = \lambda m_\alpha$, где $x_\alpha = \lambda m_\alpha$ - квантиль порядка α распределения Максвелла

In [44]:

```
k=1
for b in [means__5[a],means__10[a],means__100[a],means__1000[a],means__100000[a]]:
    if(k==1):
        print('n = 5')
    if(k==2):
        print('n = 10')
    if(k==3):
        print('n = 100')
    if(k==4):
        print('n = 1000')
    if(k==5):
        print('n = 100000')
    print(np.quantile(b,0.1))
    k+=1
```

```
n = 5
1.060797057051808
n = 10
1.3494796246697134
n = 100
0.8474203296645143
n = 1000
0.7425919843364756
n = 100000
0.7683893753535961
```

Сравнение со значением (с теоретическим) из таблицы:
$\alpha \approx 0.76$

In [45]:

```
k=1
for b in [means__5[a],means__10[a],means__100[a],means__1000[a],means__100000[a]]:
    if(k==1):
        print('n = 5')
    if(k==2):
        print('n = 10')
    if(k==3):
        print('n = 100')
    if(k==4):
        print('n = 1000')
    if(k==5):
        print('n = 100000')
    print(np.quantile(b,0.5))
    k+=1
```

```
n = 5
1.5734638080034873
n = 10
1.8158410119937378
n = 100
1.5147399868792055
n = 1000
1.4818944071588231
n = 100000
1.5439501605820434
```

In [46]:

```
#Сравнение
maxwell.median()
```

Out[46]:

```
1.5381722544550522
```

In [47]:

```
k=1
for b in [means__5[a],means__10[a],means__100[a],means__1000[a],means__100000[a]]:
    if(k==1):
        print('n = 5')
    if(k==2):
        print('n = 10')
    if(k==3):
        print('n = 100')
    if(k==4):
        print('n = 1000')
    if(k==5):
        print('n = 100000')
    print(np.quantile(b,0.7))
    k+=1
```

```
n = 5
2.2233188266889026
n = 10
2.1120878493320134
n = 100
1.8040855355096013
n = 1000
1.8805065025128846
n = 100000
1.9225891361298735
```

Сравнение со значением (с теоретическим) из таблицы:
$\alpha \approx 1.92$

С увеличением объема выборки э.ф.р. стремится к теоритической функции распределения, следовательно, эмпирические квантили так жестремятся к теоритическим по определению. Что и видно выше.

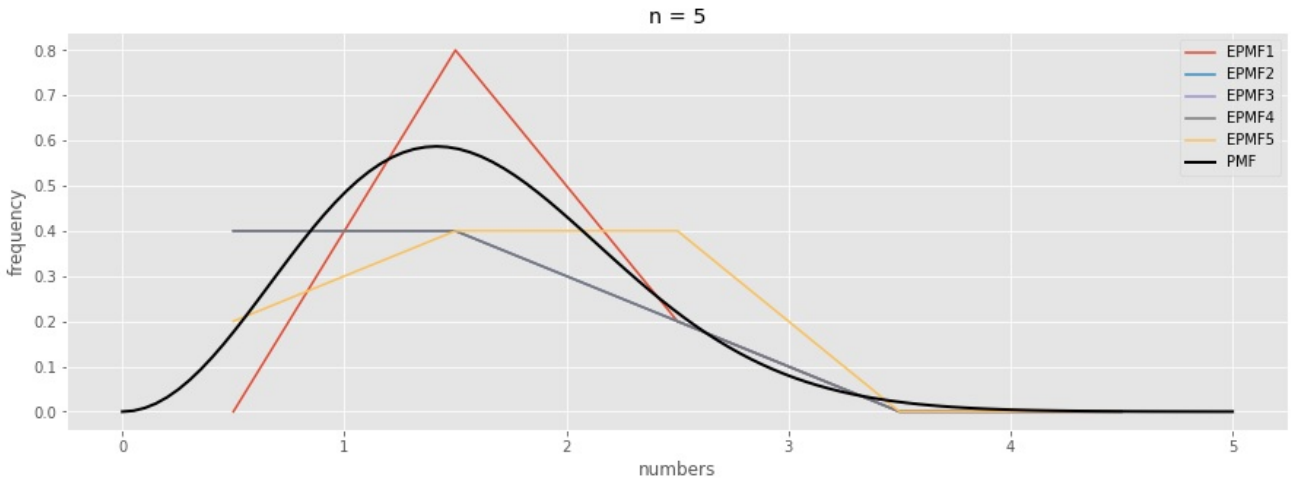
2.2.4 Построение гистограммы и полигона частот

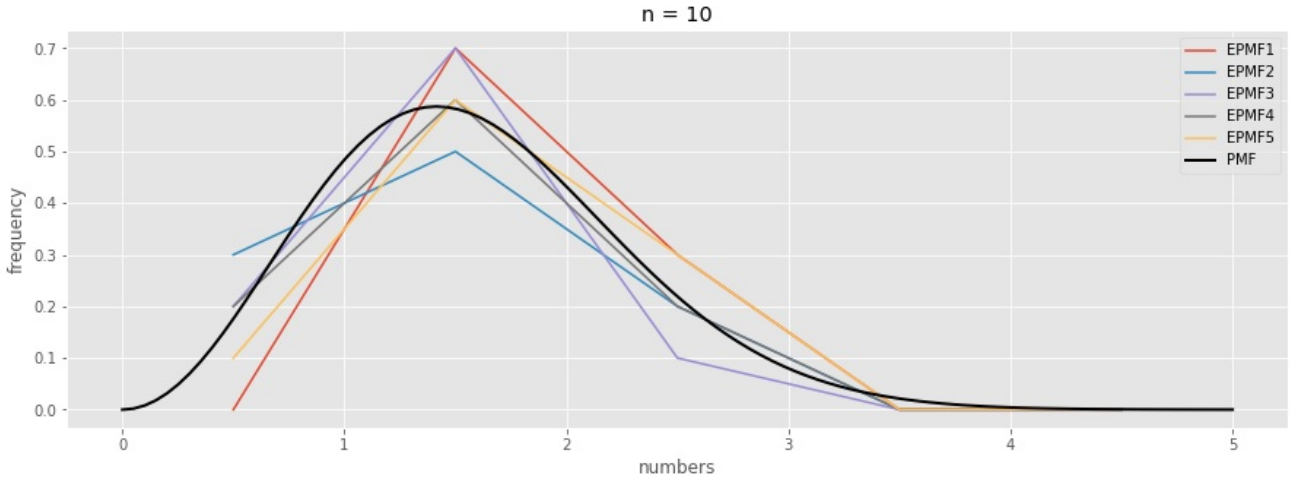
In [48]:

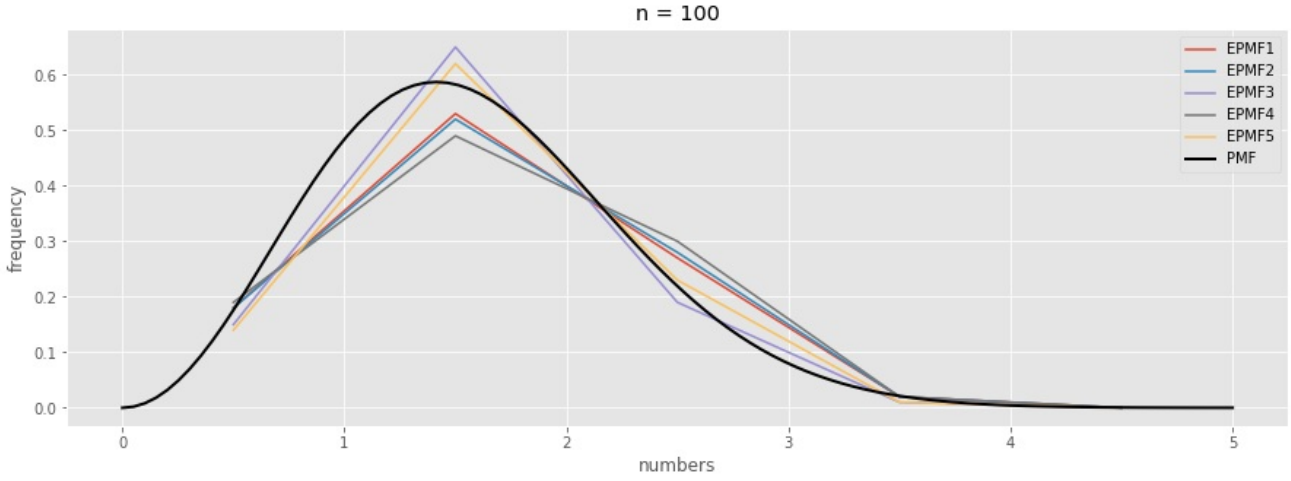
```
#n=5
for a in range(5):
    b=means__5[a]
    mas=list(range(1,6))
    p=[0,0,0,0,0]
    for i in range(5):
        mas[i]=b[i]
        if mas[i]>0 and mas[i]<1:
            p[0]=p[0]+1
        if mas[i]>1 and mas[i]<2:
            p[1]=p[1]+1
        if mas[i]>2 and mas[i]<3:
            p[2]=p[2]+1
        if mas[i]>3 and mas[i]<4:
            p[3]=p[3]+1
        if mas[i]>4 and mas[i]<5:
            p[4]=p[4]+1
    print()
    dob=[]
    bod=[]
    keks=0.5
    for i in range(5):
        dob.append(keks)
        bod.append(p[i]/5.0)
        keks+=1
    plt.plot(dob,bod,label='EPMF'+str(a+1))
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 5")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=10
for a in range(5):
    b=means__10[a]
    mas=list(range(1,11))
    p=[0,0,0,0,0]
    for i in range(10):
        mas[i]=b[i]
        if mas[i]>0 and mas[i]<1:
```

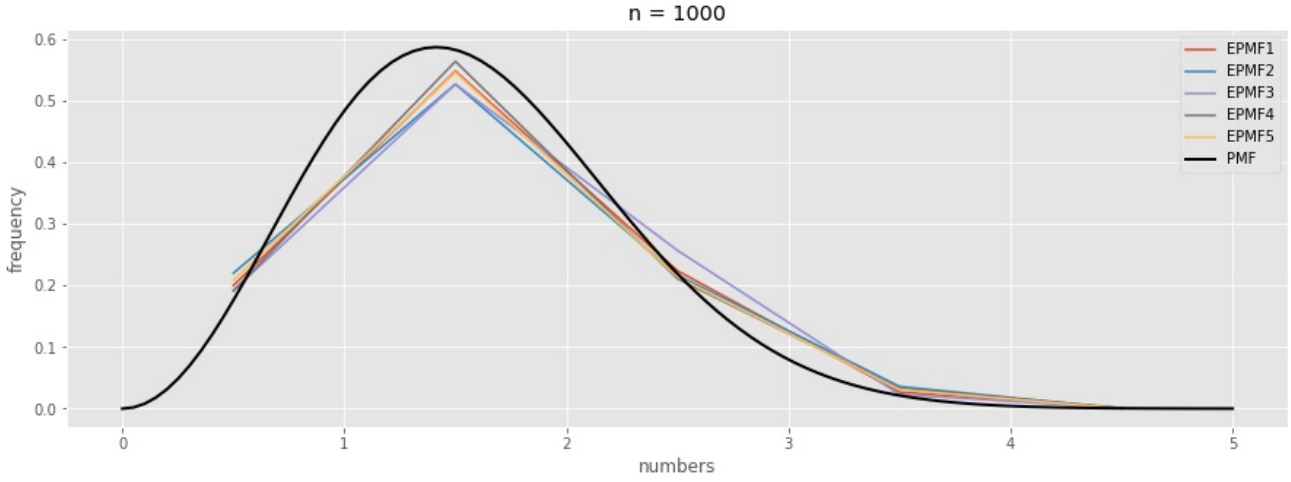
```
            p[0]=p[0]+1

        if mas[i]>1 and mas[i]<2:
            p[1]=p[1]+1
        if mas[i]>2 and mas[i]<3:
            p[2]=p[2]+1
        if mas[i]>3 and mas[i]<4:
            p[3]=p[3]+1
        if mas[i]>4 and mas[i]<5:
            p[4]=p[4]+1
    print()
    dob=[]
    bod=[]
    keks=0.5
    for i in range(5):
        dob.append(keks)
        bod.append(p[i]/10.0)
        keks+=1
    plt.plot(dob,bod,label='EPMF'+str(a+1))
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 10")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=100
for a in range(5):
    b=means__100[a]
    mas=list(range(1,101))
    p=[0,0,0,0,0]
    for i in range(100):
        mas[i]=b[i]
        if mas[i]>0 and mas[i]<1:
            p[0]=p[0]+1
        if mas[i]>1 and mas[i]<2:
            p[1]=p[1]+1
        if mas[i]>2 and mas[i]<3:
            p[2]=p[2]+1
        if mas[i]>3 and mas[i]<4:
            p[3]=p[3]+1
        if mas[i]>4 and mas[i]<5:
            p[4]=p[4]+1
    print()
    dob=[]
    bod=[]
    keks=0.5
    for i in range(5):
        dob.append(keks)
        bod.append(p[i]/100.0)
        keks+=1
    plt.plot(dob,bod,label='EPMF'+str(a+1))
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 100")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=1000
for a in range(5):
    b=means__1000[a]
    mas=list(range(1,1001))
    p=[0,0,0,0,0]
    for i in range(1000):
        mas[i]=b[i]
        if mas[i]>0 and mas[i]<1:
            p[0]=p[0]+1
        if mas[i]>1 and mas[i]<2:
            p[1]=p[1]+1
        if mas[i]>2 and mas[i]<3:
            p[2]=p[2]+1
        if mas[i]>3 and mas[i]<4:
            p[3]=p[3]+1
        if mas[i]>4 and mas[i]<5:
            p[4]=p[4]+1
    print()
    dob=[]
    bod=[]
    keks=0.5
    for i in range(5):
        dob.append(keks)
```

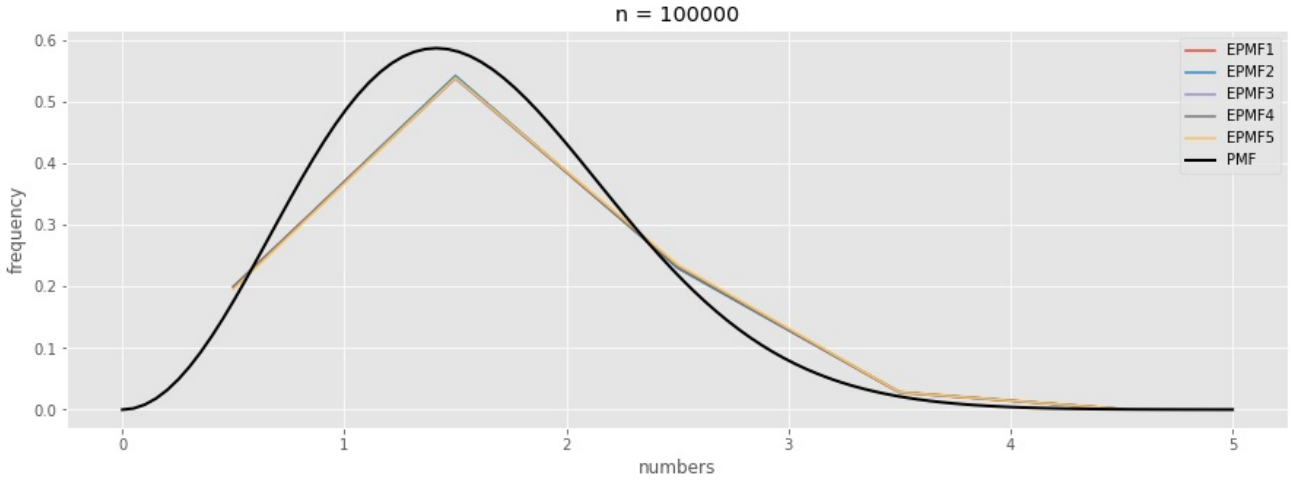
```
        bod.append(p[i]/1000.0)

        keks+=1
    plt.plot(dob,bod,label='EPMF'+str(a+1))
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 1000")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=100000
for a in range(5):
    b=means__100000[a]
    mas=list(range(1,100001))
    p=[0,0,0,0,0]
    for i in range(100000):
        mas[i]=b[i]
        if mas[i]>0 and mas[i]<1:
            p[0]=p[0]+1
        if mas[i]>1 and mas[i]<2:
            p[1]=p[1]+1
        if mas[i]>2 and mas[i]<3:
            p[2]=p[2]+1
        if mas[i]>3 and mas[i]<4:
            p[3]=p[3]+1
        if mas[i]>4 and mas[i]<5:
            p[4]=p[4]+1
    print()
    dob=[]
    bod=[]
    keks=0.5
    for i in range(5):
        dob.append(keks)
        bod.append(p[i]/100000.0)
        keks+=1
    plt.plot(dob,bod,label='EPMF'+str(a+1))
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 100000")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```




n = 10
EPMF1
EPMF2
EPMF3
EPMF4
EPMF5
PMF
frequency
numbers

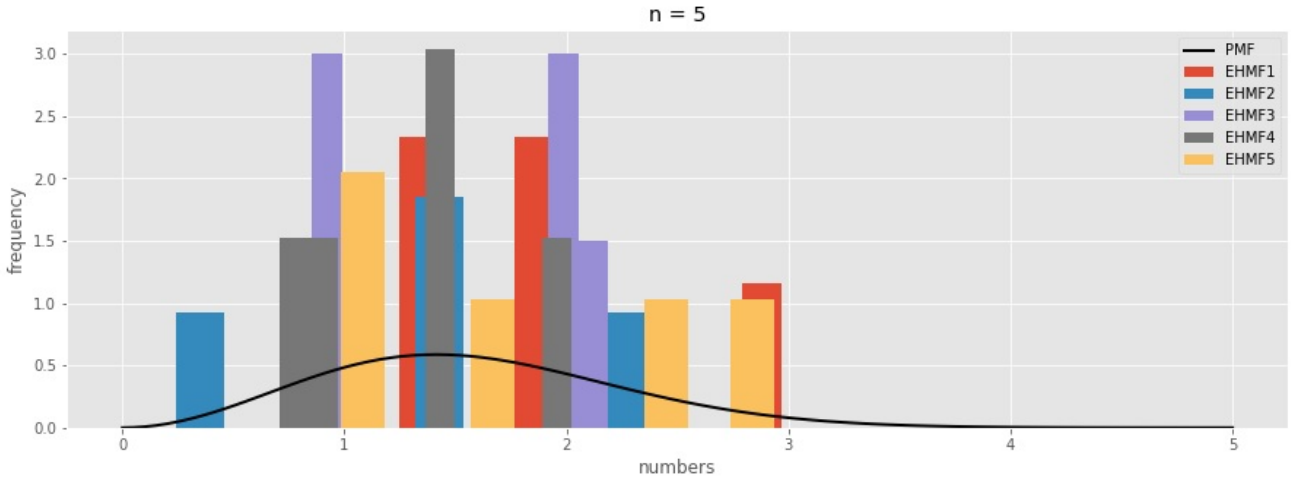

n = 100
EPMF1
EPMF2
EPMF3
EPMF4
EPMF5
PMF
frequency
numbers

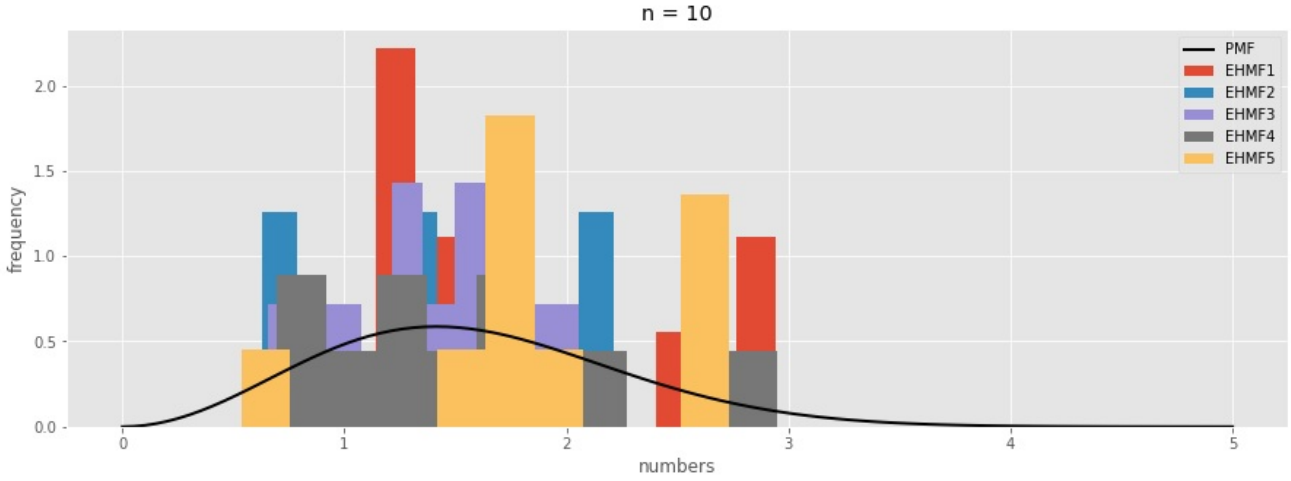

n = 1000
EPMF1
EPMF2
EPMF3
EPMF4
EPMF5
PMF
frequency
numbers

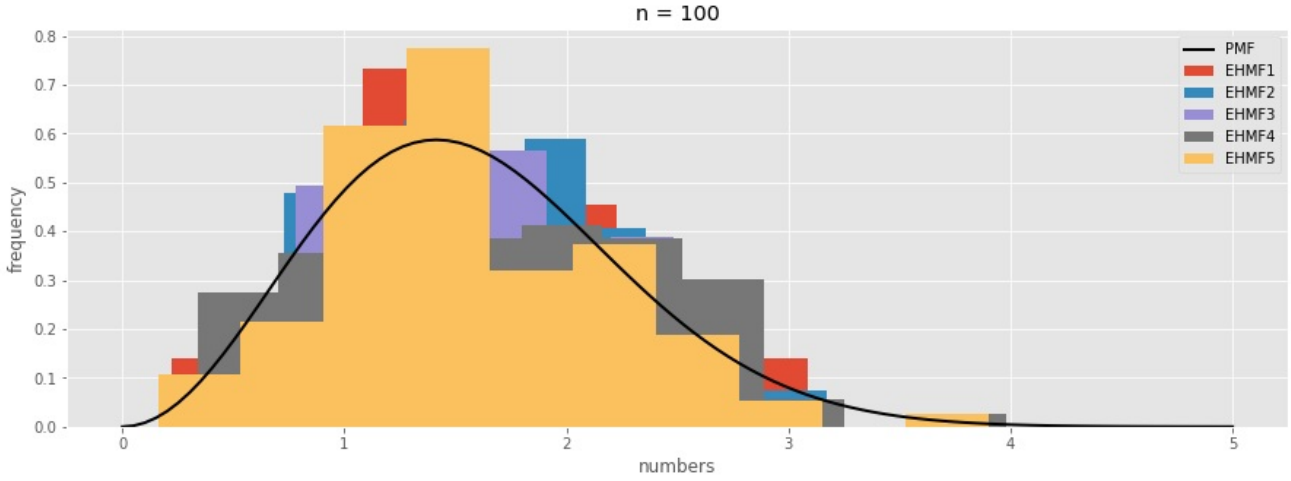


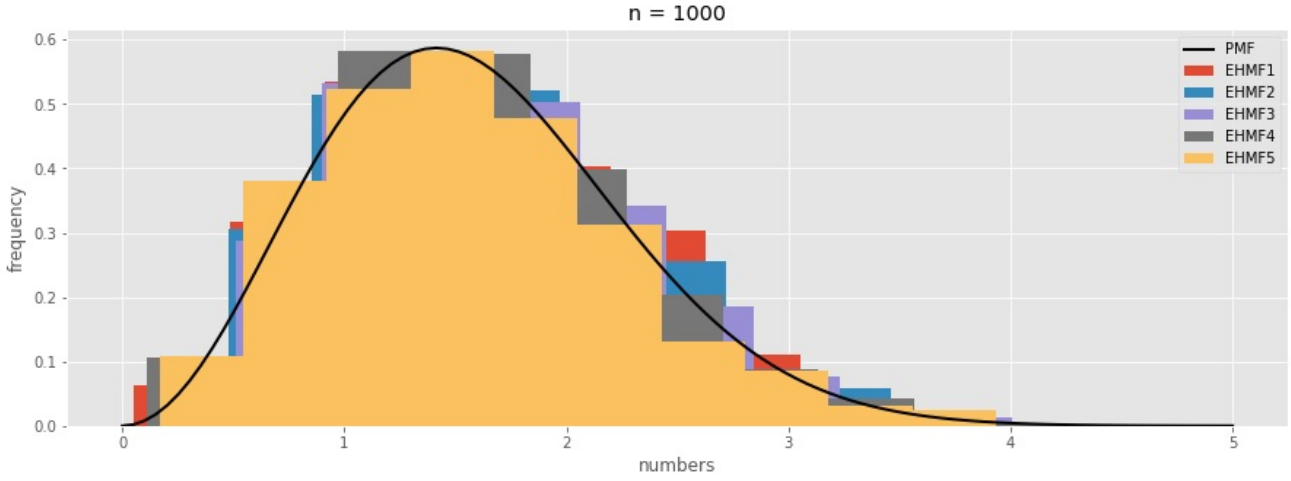
In [49]:

```
#n=5
for a in range(5):
    plt.hist(means__5[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 5")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=10
for a in range(5):
    plt.hist(means__10[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 10")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=100
for a in range(5):
    plt.hist(means__100[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 100")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
#n=1000
for a in range(5):
    plt.hist(means__1000[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 1000")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```


n = 5
PMF
EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
frequency
numbers


n = 10
PMF
EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
frequency
numbers


n = 100
PMF
EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
frequency
numbers


n = 1000
PMF
EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
frequency
numbers

Для непрерывной случайной величины ξ, обладающей непрерывной плотностью f(x), также можно построить по соответствующей выборке $X = (X_1, \ldots, X_n)$ статистический аналог $\hat{f}_n(x)$ для плотности f(x), который называется гистограммой. Для этого используется методгруппировки, в соответствии с которым область Δ возможных значений ξ разбивается на некоторое число N непересекающихся интервалов $\Delta_1, \ldots, \Delta_N$ (так что $\Delta = \bigcup_{r=1}^{N} \Delta_r$, подсчитывают числа $v_1, \ldots, v_N$ наблюдений $X_1, \ldots, X_n$, попавших в соответствующие интервалы: $v_r = \sum_{j=1}^{N} I(X_j \in \Delta_r), r = 1, \ldots, N$ (так что $\sum_{r=1}^{N} v_r = n$,
и строят кусочно-постоянную функцию

$$\hat{f}_n(x) = \frac{v_r}{n\,|\Delta_r|}$$

при x $\in \Delta_r, r = 1, \ldots, N$
Здесь $|\Delta_r|$ - длина интервала Δ_r. То, что построенная по такому правилу гистограмма $\hat{f}_n(x)$ действительно "похожа" на теоретическую плотность f(x), следует из закона больших чисел, согласно которому при $n \to \infty$ относительная частота $\frac{v_r}{n}$ сближается с теоретической вероятностью

$$P\{\xi \in \Delta_r\} = \int_{\Delta_r} f(x)dx$$

Но этот интеграл по теореме о среднем равен $f(a_r)\,|\Delta_r|$ где a_r - некоторая внутренняя точка интервала Δ_r(при малом Δ_r в качестве a_r можно взять, например, середину интервала), Таким образом, при больших n и достаточно "мелком" разбиении $\{\Delta_r\}\hat{f}_n(x) \approx f(a_r)$ при $x \in \Delta_r$ т.е. гистограмма $\hat{f}_n(x)$ будет достаточно хорошо приближать график плотности f(x), следовательно, $\hat{f}_n(x)$ можно рассматривать в качестве статистического аналога (оценки) для f(x).Наряду с гистограммой, в качестве приближения для неизвестной теоретической плотности f(x) можно использовать кусочно-линейный график называемый полигоном частот. Он также считается статистическим аналогом теоретической плотности. Данные на полигоне частот и гистограммах подтверждают теоретические знания: с увеличением объема выборки полигон частот и гистограммы практически совпадают с теоретической плотностью f(x).

3 Домашнее задание. Оценки

3.1 Нахождение выборочного среднего и выборочной дисперсии геометрического распределения

Наиболее важными характеристиками случайной величины ξ являются ее моменты $\alpha_k = M\xi_k$, а также цетральные моменты $\mu^k = M(\xi - \alpha 1)^k$ (когда они существуют). Их статистическими аналогами, вычисляемыми по соответсвующей выборке $X = (X_1, \ldots, X_n)$, являются выборочные моменты соотаетсвенно обычные:

$$\hat{\alpha}_k = \frac{1}{n}\sum_{i=1}^{n} X_i^k$$

и центральные:

$$\hat{\mu}_k = \frac{1}{n}\sum_{i=1}^{n} (X_i - \hat{\alpha}_1)^k$$

$\hat{\alpha}_1$ (принято обозначать, как $\bar{X}$) называют выборочным средним, μ^2 - выборочной дисперсией. Таким образом, выборочное среднее и выборочная дисперсия являются статистическими аналогами теоритических среднего (математического ожидания) $M\xi$ и дисперсии $D\xi$, когда они существуют.

Выборочное среднее, относящийся к выборке X, считается как:

$$\bar{X} = \hat{\alpha}_1 = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Выборочная дисперсия, относящийся к выборке X, подсчитывается как:

$$S^2 = \hat{\mu}_2 = \frac{1}{n}\sum_{i=1}^{n} (X_i - \bar{X})^2$$

Найдем математическое ожидание и дисперсию выборочного среднего и выборочной дисперсии:

$$M\bar{X} = \frac{1}{n}\sum_{i=1}^{n} MX_i = M\xi = \alpha_1$$

$$D\bar{X} = \frac{1}{n^2}\sum_{i=1}^{n} DX_i = \frac{1}{n}D\xi = \frac{\mu_2}{n}$$

Для выборочной дисперсии введем обозначение: $Y_i = X_i - \alpha_1$:

$$S^2 = \hat{\mu}_2 = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y})^2 = \frac{1}{n}\sum_{i=1}^{n} Y_i^2 - \bar{Y}^2$$

Поскольку $MY_i = 0, MY_i^2 = \mu_2$ и $MY_iY_j = MY_jY_i = 0, (i \neq j),$ то:

$$M\bar{Y}^2 = \frac{1}{n^2}\sum_{i,j=1}^{n} MY_iY_j = \frac{1}{n^2}\sum_{i=1}^{n} MY_i^2 = \frac{\mu_2}{n}$$

Отсюда следует, что

$$MS^2 = \frac{n-1}{n}\mu_2$$

Перейдём к вычислению DS^2

$$(S^2)^2 = \frac{1}{n^2}(\sum_{i=1}^{n} Y_i^2)^2 - \frac{2}{n}\bar{Y}^2\sum_{i=1}^{n} Y_i^2 + \bar{Y}^4$$

Так как случайные величины $Y_1, \ldots, Y_n$ независимы и $MY_i = 0$, то в правой части равенства

$$M\bar{Y}^4 = \frac{1}{n^4}(n\mu_4 + 3n(n-1)\mu_2^2) = \frac{\mu_4 + 3(n-1)\mu_2^2}{n^3}$$

Аналогично находим

$$\frac{1}{n^2}M(\sum_{i=1}^{n} Y_i^2) = \frac{\mu_4 + (n-1)\mu_2^2}{n} = M(\bar{Y}^2\sum_{i=1}^{n} Y_i^2)$$

С учётом этих соотношений по формуле

$$DS^2 = M(S^2)^2 - (MS^2)^2$$

получим

$$DS^2 = \frac{\mu_4 - \mu_2^2}{n} - \frac{2(\mu_4 - 2\mu_2^2)}{n^2} + \frac{\mu_4 - 3\mu_2^2}{n^3} = \frac{(n-1)^2}{n^3}(\mu_4 - \frac{n-3}{n-1}\mu_2^2)$$

Аналогично можно находить моменты и более высоких порядков, хотя с увеличением порядка вид формул и их вывод усложняются.

Теперь рассмотрим свойства выборочных среднего и дисперсии при неограниченном возрастании объема выборки n, которые дадут нам ответ на вопрос, оценками каких параметров рапределений они являются. Чтобы подчеркнуть зависимость моментов $\hat{\alpha}_k, \hat{\mu}_k$ от объема выборки, будем в дальнейшем приписывать дополнительный индекс n: $\hat{\alpha}_{nk}, \hat{\mu}_{nk}$

$$M\hat{\alpha}_{nk} = \frac{1}{n}\sum_{i=1}^{n} MX_i^k = M\xi^k = \hat{\alpha}_k$$

$$D\hat{\alpha}_{nk} = \frac{1}{n^2}\sum_{i=1}^{n} DX_i^k = \frac{1}{n}D\xi^k = \frac{1}{n}(M\xi^{2k} - (M\xi^k)^2) = \frac{\alpha_{2k} - \alpha_k^2}{n}$$

На основании неравенства Чебышева, отсюда следует, что для любого $\epsilon > 0$ при $n \to \infty$

$$P|\hat{\alpha}_{nk} - \alpha_k| < \epsilon \to 1$$

т.е. выборочный момент $\hat{\alpha}_{nk}$ сходится по вероятности при $n \to \infty$ к соответствующему теоретическому моменту α_k . Таким образом, $\hat{\alpha}_{nk}$ можно использовать в качестве оценки α_k, когда объем выборки достаточно велик. Аналогичное утверждение справедливо и для центральных моментов:

$$P|\mu_{nk} - \mu_k| < \epsilon \to 1$$

т.е. μ_{nk} можно использовать в качестве оценки μ_k , когда объем выборки достаточно велик.

1. Оценка $\hat{\theta}(X)$ параметра θ называется несмещенной, если:

$$E(\hat{\theta}(X)) = \theta$$

1. Оценка $\hat{\theta}(X) = \hat{\theta}_n(X_1, \ldots, X_n)$ параметра θ называется состоятельной, если при $n \to \infty$ соблюдается:

$$\hat{\theta}_n(X_1, \ldots, X_n) \xrightarrow{p} \theta$$

При этом для проверки состоятельности достаточно убедиться, что соблюдены следующие два условия:

$$\lim_{n\to\infty} E(\hat{\theta}_n(X_1, \ldots, X_n)) = \theta$$

$$\lim_{n\to\infty} Var(\hat{\theta}_n(X_1, \ldots, X_n)) = 0$$

Выборочное среднее является несмещенной оценкой для теоретического математического ожидания.

$$\hat{\alpha_1} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

$$M\hat{\alpha_1} = M(\frac{1}{n}\sum_{i=1}^{n} x_i) = \frac{1}{n} \cdot n \cdot MX = MX$$

$$M\hat{\alpha_1} = MX$$

Выборочное среднее является состоятельной оценкой для теоретического математического ожидания.

$$\hat{\alpha_1} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

$$\lim_{n\to\infty} \hat{\alpha_1} = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} X_i = \lim_{n\to\infty} \frac{1}{n} \cdot n \cdot MX = MX$$

Выборочная дисперсия S^2 является состоятельной и несмещенной оценкой для теоретической дисперсии.

$$S^2 = \hat{\mu_2} = \frac{1}{n}\sum_{i=1}^{n} X_i^2 - \frac{1}{n}\sum_{i=1}^{n} (\hat{\alpha_1})^2 = MX^2 - \frac{1}{n} n (\hat{\alpha_1})^2 = \hat{\alpha_2} - (\hat{\alpha_1})^2$$

$$MS^2 = M(\hat{\alpha_2} - (\hat{\alpha_1})^2) = M(\frac{1}{n}\sum_{i=1}^{n} X_i^2) - M((\frac{1}{n}\sum_{i=1}^{n} X_i))^2 = MX^2 - (MX)^2 = DX$$

Продолжим исследование свойств выборочных моментов для больших выборок и рассмотрим теперь асимптотическое поведение их выборочных распределений.
Если распределение случайной величины v_n сходится при $n \to \infty$ к распределению случайной величины v и при этом $\zeta(v) = N(\mu, \sigma^2)$, то будем писать $\zeta(v_n) \to N(\mu, \sigma^2)$. Далее иногда будем говорить, что случайная величина v_n асимптотически нормальна $N(\mu_n, \sigma_n^2)$, и записывать это следующим образом:
$\zeta(v_n) \approx N(\mu_n, \sigma_n^2)$, если $\zeta(\frac{v_n - \mu_n}{\sigma_n}) \to N(0, 1)$.
Найдем сначала асимптотические распределения выборочных моментов $\hat{\alpha}_{nk}$. Величина $n\hat{\alpha}_{nk} = \sum_{i=1}^{n} X_i^k$ является суммой независимых одинаково распределенных случайных величин. Если конечен момент $\alpha_{2k} = M\xi^{2k}$, то к этой сумме можно применить центральную предельную теорему теории вероятностей. Так как $MX_i^k = \alpha^k, DX_i^k = \alpha_{2k} - \alpha_k^2$, то величина

$$\frac{n\hat{\alpha}_{nk} - n\alpha_k}{\sqrt{n(\alpha_{2k} - \alpha_k^2)}} = \frac{\hat{\alpha}_{nk} - \alpha_k}{\sqrt{\frac{\alpha_{2k} - \alpha_k^2}{n}}}$$

асимтотически нормальна $N(0, 1)$. Таким образом справедлива следующая теорема:

Если конечен теоритический момент α_{2k} , то при $n \to \infty$ выборочный момент $\hat{\alpha}_{nk}$ асимптотически нормален $N(\alpha_k, \frac{\alpha_{2k} - \alpha_k^2}{n})$

Из теоремы следует, что если существует теоретическая дисперсия, то выборочное среднее $\hat{\alpha}_{n1}$ асимптотически нормально $N(\alpha_1, \frac{\mu_2}{n})$
Из теоремы об асимптотической нормальности функций от выборочных моментов следует, что асимптотически нормальными являются и центральные выборочные моменты $\hat{\mu}_{nk}$, поскольку они являются непрерывными функциями (многочленами) от обычных выборочных моментов.

3.1.1. Геометрическое распределение

Выборочное среднее

In [50]:

```
#n=5
vs_5 = []
for i in range(5):
    vs_5.append(np.mean(means_5[i]))
print(vs_5)
#n=10
vs_10 = []
for i in range(5):
    vs_10.append(np.mean(means_10[i]))
print(vs_10)
#n=100
vs_100 = []
for i in range(5):
    vs_100.append(np.mean(means_100[i]))
print(vs_100)
#n=1000
vs_1000 = []
for i in range(5):
    vs_1000.append(np.mean(means_1000[i]))
print(vs_1000)
#n=100000
vs_100000 = []
for i in range(5):
    vs_100000.append(np.mean(means_100000[i]))
print(vs_100000)
```

```
[1.8, 3.8, 2.2, 2.0, 2.0]
[1.7, 2.1, 2.0, 2.0, 1.7]
[2.07, 2.1, 2.11, 2.01, 1.91]
[2.028, 1.988, 1.991, 1.952, 1.919]
[2.00096, 1.99073, 2.00076, 2.00731, 1.99558]
```

$M\xi = \frac{1}{p}$, при p = 0.5 $M\xi = 2$

In [77]:

```
#Сравнение
p = 0.5
geom.stats(p, moments = 'm')
```

Out[77]:

```
array(2.)
```

Выборочная дисперсия

In [55]:

```
#n=5
vd_5 = []
for i in range(5):
    vd_5.append(round(np.var(means_5[i]),6))
print(vd_5)
#n=10
vd_10 = []
for i in range(5):
    vd_10.append(round(np.var(means_10[i]),6))
print(vd_10)
#n=100
vd_100 = []
for i in range(5):
    vd_100.append(round(np.var(means_100[i]),6))
print(vd_100)
#n=1000
vd_1000 = []
for i in range(5):
    vd_1000.append(round(np.var(means_1000[i]),6))
print(vd_1000)
#n=100000
vd_100000 = []
for i in range(5):
    vd_100000.append(round(np.var(means_100000[i]),6))
print(vd_100000)
```

```
[0.56, 4.16, 0.56, 0.8, 2.4]
[1.21, 1.29, 3.2, 1.2, 0.81]
[1.9651, 2.31, 2.1779, 1.7099, 1.6819]
[2.119216, 2.071856, 2.094919, 1.631696, 1.496439]
[1.989759, 1.969944, 1.996059, 1.996237, 1.99118]
```

$$D\xi = \frac{q}{p^2} = \frac{1-p}{p^2}$$

При p = 0.5, $D\xi = 2$

In [78]:

```
#Сравнение
geom.stats(p,moments = 'v')
```

Out[78]:

```
array(2.)
```

Как видно из полученных значений, чем больше объём выборки, тем менее отличаются выборочное среднее от теоретического математического ожидания и выборочная дисперсия от теоретической дисперсии

3.1.2 Распределения Максвелла

Выборочное среднее

In [57]:

```
#n=5
vs__5 = []
for i in range(5):
    vs__5.append(round(np.mean(means__5[i]),6))
print(vs__5)
#n=10
vs__10 = []
for i in range(5):
    vs__10.append(round(np.mean(means__10[i]),6))
print(vs__10)
#n=100
vs__100 = []
for i in range(5):
    vs__100.append(round(np.mean(means__100[i]),6))
print(vs__100)
#n=1000
vs__1000 = []
for i in range(5):
    vs__1000.append(round(np.mean(means__1000[i]),6))
print(vs__1000)
#n=100000
vs__100000 = []
for i in range(5):
    vs__100000.append(round(np.mean(means__100000[i]),6))
print(vs__100000)
```

```
[1.851413, 1.249539, 1.572071, 1.283721, 1.810898]
[1.763259, 1.410556, 1.350879, 1.526287, 1.885552]
[1.591401, 1.627762, 1.5569, 1.651824, 1.57553]
[1.591306, 1.589926, 1.612018, 1.580167, 1.568099]
[1.59339, 1.593304, 1.595136, 1.595854, 1.602313]
```

In [79]:

```
#Сравнение
2*np.sqrt(2/np.pi)
```

Out[79]:

```
1.5957691216057308
```

$$M\xi = 2\lambda\sqrt{\frac{2}{\pi}}$$

При $\lambda =$ 1.0 $M\xi = 2 \cdot \sqrt{\frac{2}{\pi}} \approx 1.5957691216057308$

Выборочная дисперсия

In [59]:

```
#n=5
vd__5 = []
for i in range(5):
    vd__5.append(round(np.var(means__5[i]),6))
print(vd__5)
#n=10
vd__10 = []
for i in range(5):
    vd__10.append(round(np.var(means__10[i]),6))
print(vd__10)
#n=100
vd__100 = []
for i in range(5):
    vd__100.append(round(np.var(means__100[i]),6))
print(vd__100)
#n=1000
vd__1000 = []
for i in range(5):
    vd__1000.append(round(np.var(means__1000[i]),6))
print(vd__1000)
#n=100000
vd__100000 = []
for i in range(5):
    vd__100000.append(round(np.var(means__100000[i]),6))
print(vd__100000)
```

```
[0.3779, 0.500176, 0.330233, 0.216647, 0.547757]
[0.471182, 0.298791, 0.1685, 0.398861, 0.380779]
[0.418308, 0.364353, 0.297598, 0.528752, 0.407944]
[0.455744, 0.476013, 0.450745, 0.450872, 0.466015]
[0.452269, 0.449961, 0.453451, 0.454812, 0.455364]
```

In [80]:

```
#Сравнение
(3*np.pi-8)/np.pi
```

Out[80]:

```
0.45352091052967 45
```

$$D\xi = \frac{3\pi - 8}{\pi} \cdot \lambda$$

При $\lambda =$ 1.0 $D\xi = \frac{3\pi-8}{\pi} \approx 0.4535209105296745$

Как видно из полученных значений, чем больше объём выборки, тем менее отличаются выборочное среднее от теоретического математического ожидания и выборочная дисперсия от теоретической дисперсии.

3.2 Построение доверительного интервала для выборочного среднего и выборочной дисперсии

Определение: γ - доверительным интервалом для g называется такой случайный интервал $(T_1(X), T_2(X))$, $T_1(X) < T_2(X)$, который содержит внутри себя (накрывает) неизвестное значение g с вероятностью, не меньшей γ:

$$P\{T_1(X) < g < T_2(X)\} \geq \gamma$$

Здесь $T_1(X)$ и $T_2(X)$ - некоторые статистики (функции от выборки), называемые соответственно нижней и верхней доверительными границами, а γ - задаваемый заранее доверительный уровень, который обычно выбирается близким к 1. Длина доверительного интервала характеризует точность локализации оцениваемой характеристики g, а величина γ является показателем надежности доверительного интервала.
В сформулированной ранее теореме [Если конечен теоритический момент α_{2k} , то при $n \to \infty$ выборочный момент $\hat{\alpha}_{nk}$ асимптотически нормален $N(\alpha_k, \frac{\alpha_{2k}-\alpha_k^2}{n})$] можно заменить асимптотическую дисперсию $\frac{\alpha_{2k}-\alpha_k^2}{n}$ ее оценкой $\frac{\hat{\alpha}_{n,2k}-\hat{\alpha}_{nk}^2}{n}$. Это дает искомый асимптотический γ-доверительный интервал для момента α_k вида:

$$(\hat{\alpha}_{nk} \mp c_\gamma \sqrt{\frac{\hat{\alpha}_{n,2k} - \hat{\alpha}_{nk}^2}{n}})$$

Полагая здесь k = 1, получим соответствующий интервал для теоретического среднего $\alpha_1 = M\xi$:

$$(\bar{X} \mp \frac{c_\gamma S}{\sqrt{n}})$$

Чтобы построить асимптотический γ-доверительный интервал для теоретической дисперсии $\mu_2 = D\xi$, надо просто воспользоваться результатом теоремы об асимптотической нормальности выборочной дисперсии [$\zeta(\frac{\sqrt{n}(S^2-\mu_2)}{\sqrt{\hat{\mu}_{n4}-S^4}}) \to N(0, 1)$]: искомый интервал есть

$$(S^2 \mp c_\gamma \sqrt{\frac{\hat{\mu}_{n4} - S^4}{n}})$$

3.2.1 Геометрическое распределение

Положим $\gamma = 0.95$ и найдем доверительный интервал для выборочного среднего.

$$\Phi(c_\gamma) = \frac{\gamma}{2} = 0.475$$

Из таблицы значений функции Лапласа $c_\gamma \approx 1.96$

In [61]:

```
#n=5
print('n = 5')
for i in range(5):
    print('(', vs_5[i], '-+ 1.96 *', np.sqrt(vd_5[i]/5), ') = (', vs_5[i], '-+', round(1.96*np.sqrt(vd_5[i]/5), 6
), ')')
#n=10
print('n = 10')
for i in range(5):
    print('(', vs_10[i], '-+ 1.96 *', np.sqrt(vd_10[i]/10), ')= (', vs_10[i], '-+', round(1.96*np.sqrt(vd_10[i]/1
0), 6), ')')
#n=100
print('n = 100')
for i in range(5):
    print('(', vs_100[i], '-+ 1.96 *', np.sqrt(vd_100[i]/100), ') = (', vs_100[i], '-+', round(1.96*np.sqrt(vd_10
0[i]/100), 6), ')')
#n=1000
print('n = 1000')
for i in range(5):
    print('(', vs_1000[i], '-+ 1.96 *', np.sqrt(vd_1000[i]/1000), ') = (', vs_1000[i], '-+', round(1.96*np.sqrt(v
d_1000[i]/1000), 6), ')')
#n=100000
print('n = 100000')
for i in range(5):
    print('(', vs_100000[i], '-+ 1.96 *', np.sqrt(vd_100000[i]/100000), ') = (', vs_100000[i], '-+', round(1.96*n
p.sqrt(vd_100000[i]/100000), 6), ')')
```

```
n = 5
( 1.8 -+ 1.96 * 0.33466401061363027 ) = ( 1.8 -+ 0.655941 )
( 3.8 -+ 1.96 * 0.9121403400793104 ) = ( 3.8 -+ 1.787795 )
( 2.2 -+ 1.96 * 0.33466401061363027 ) = ( 2.2 -+ 0.655941 )
( 2.0 -+ 1.96 * 0.4 ) = ( 2.0 -+ 0.784 )
( 2.0 -+ 1.96 * 0.6928203230275509 ) = ( 2.0 -+ 1.357928 )
n = 10
( 1.7 -+ 1.96 * 0.3478505426185217 )= ( 1.7 -+ 0.681787 )
( 2.1 -+ 1.96 * 0.35916569992135944 )= ( 2.1 -+ 0.703965 )
( 2.0 -+ 1.96 * 0.565685424949238 )= ( 2.0 -+ 1.108743 )
( 2.0 -+ 1.96 * 0.34641016151377546 )= ( 2.0 -+ 0.678964 )
( 1.7 -+ 1.96 * 0.28460498941515416 )= ( 1.7 -+ 0.557826 )
n = 100
( 2.07 -+ 1.96 * 0.14018202452525788 ) = ( 2.07 -+ 0.274757 )
( 2.1 -+ 1.96 * 0.15198684153570663 ) = ( 2.1 -+ 0.297894 )
( 2.11 -+ 1.96 * 0.14757709849431247 ) = ( 2.11 -+ 0.289251 )
( 2.01 -+ 1.96 * 0.13076314465475355 ) = ( 2.01 -+ 0.256296 )
( 1.91 -+ 1.96 * 0.12968808734806755 ) = ( 1.91 -+ 0.254189 )
n = 1000
( 2.028 -+ 1.96 * 0.04603494324966634 ) = ( 2.028 -+ 0.090228 )
( 1.988 -+ 1.96 * 0.04551764493029049 ) = ( 1.988 -+ 0.089215 )
( 1.991 -+ 1.96 * 0.045770285120370395 ) = ( 1.991 -+ 0.08971 )
( 1.952 -+ 1.96 * 0.040394257017551394 ) = ( 1.952 -+ 0.079173 )
( 1.919 -+ 1.96 * 0.03868383383275241 ) = ( 1.919 -+ 0.07582 )
n = 100000
( 2.00096 -+ 1.96 * 0.004460671474116874 ) = ( 2.00096 -+ 0.008743 )
( 1.99073 -+ 1.96 * 0.004438405118958836 ) = ( 1.99073 -+ 0.008699 )
( 2.00076 -+ 1.96 * 0.004467727610318248 ) = ( 2.00076 -+ 0.008757 )
( 2.00731 -+ 1.96 * 0.004467926812292251 ) = ( 2.00731 -+ 0.008757 )
( 1.99558 -+ 1.96 * 0.0044622639993617584 ) = ( 1.99558 -+ 0.008746 )
```

*Округлено до 6 знаков после запятой.

3.2.2 Распределение Максвелла

Положим $\gamma = 0.95$ и найдем доверительный интервал для выборочного среднего.

$$\Phi\,(c_\gamma) = \frac{\gamma}{2} = 0.475$$

Из таблицы значений функции Лапласа $c_\gamma \approx 1.96$

In [62]:

```
#n=5
print('n = 5')
for i in range(5):
    print('(', vs__5[i], '-+ 1.96 *', np.sqrt(vd__5[i]/5), ') = (', vs__5[i], '-+', round(1.96*np.sqrt(vd__5[i]/5
), 6), ')')
#n=10
print('n = 10')
for i in range(5):
    print('(', vs__10[i], '-+ 1.96 *', np.sqrt(vd__10[i]/10), ')= (', vs__10[i], '-+', round(1.96*np.sqrt(vd__10[
i]/10), 6), ')')
#n=100
print('n = 100')
for i in range(5):
    print('(', vs__100[i], '-+ 1.96 *', np.sqrt(vd__100[i]/100), ') = (', vs__100[i], '-+', round(1.96*np.sqrt(vd
__100[i]/100), 6), ')')
#n=1000
print('n = 1000')
for i in range(5):
    print('(', vs__1000[i], '-+ 1.96 *', np.sqrt(vd__1000[i]/1000), ') = (', vs__1000[i], '-+', round(1.96*np.sqr
t(vd__1000[i]/1000), 6), ')')
#n=100000
print('n = 100000')
for i in range(5):
    print('(', vs__100000[i], '-+ 1.96 *', np.sqrt(vd__100000[i]/100000), ') = (', vs__100000[i], '-+', round(1.9
6*np.sqrt(vd__100000[i]/100000), 6), ')')
```

```
n = 5
( 1.851413 -+ 1.96 * 0.2749181696432595 ) = ( 1.851413 -+ 0.53884 )
( 1.249539 -+ 1.96 * 0.31628341720678305 ) = ( 1.249539 -+ 0.619915 )
( 1.572071 -+ 1.96 * 0.2569953306968825 ) = ( 1.572071 -+ 0.503711 )
( 1.283721 -+ 1.96 * 0.20815715217114208 ) = ( 1.283721 -+ 0.407988 )
( 1.810898 -+ 1.96 * 0.3309854981717477 ) = ( 1.810898 -+ 0.648732 )
n = 10
( 1.763259 -+ 1.96 * 0.2170672706789303 )= ( 1.763259 -+ 0.425452 )
( 1.410556 -+ 1.96 * 0.17285572018304746 )= ( 1.410556 -+ 0.338797 )
( 1.350879 -+ 1.96 * 0.1298075498574717 )= ( 1.350879 -+ 0.254423 )
( 1.526287 -+ 1.96 * 0.19971504700447584 )= ( 1.526287 -+ 0.391441 )
( 1.885552 -+ 1.96 * 0.1951355938828178 )= ( 1.885552 -+ 0.382466 )
n = 100
( 1.591401 -+ 1.96 * 0.06467673461145051 ) = ( 1.591401 -+ 0.126766 )
( 1.627762 -+ 1.96 * 0.06036166001693459 ) = ( 1.627762 -+ 0.118309 )
( 1.5569 -+ 1.96 * 0.054552543478741666 ) = ( 1.5569 -+ 0.106923 )
( 1.651824 -+ 1.96 * 0.07271533538394773 ) = ( 1.651824 -+ 0.142522 )
( 1.57553 -+ 1.96 * 0.06387049397022071 ) = ( 1.57553 -+ 0.125186 )
n = 1000
( 1.591306 -+ 1.96 * 0.02134816151334817 ) = ( 1.591306 -+ 0.041842 )
( 1.589926 -+ 1.96 * 0.021817722154248827 ) = ( 1.589926 -+ 0.042763 )
( 1.612018 -+ 1.96 * 0.021230755992192082 ) = ( 1.612018 -+ 0.041612 )
( 1.580167 -+ 1.96 * 0.021233746725436847 ) = ( 1.580167 -+ 0.041618 )
( 1.568099 -+ 1.96 * 0.02158738057291806 ) = ( 1.568099 -+ 0.042311 )
n = 100000
( 1.59339 -+ 1.96 * 0.0021266617032334972 ) = ( 1.59339 -+ 0.004168 )
( 1.593304 -+ 1.96 * 0.002121228417686318 ) = ( 1.593304 -+ 0.004158 )
( 1.595136 -+ 1.96 * 0.0021294388932298573 ) = ( 1.595136 -+ 0.004174 )
( 1.595854 -+ 1.96 * 0.0021326321764429984 ) = ( 1.595854 -+ 0.00418 )
( 1.602313 -+ 1.96 * 0.0021339259593528546 ) = ( 1.602313 -+ 0.004182 )
```

*Округлено до 6 знаков после запятой.

3.3 Нахождение оптимальности рассматриваемых оценок

Для построения теории оптимального оценивания прежде всего надо договориться о мере точности оценок, т.е. уточнить смысл приближенного равенства $T(X) \approx g$. Если статистика T(x) используется для оценивания g, то одной из разумных мер расхождения между ними является $(T(X) - g)^2$, или квадратичная ошибка. Но так как это величина случайная используется среднеквадратичная ошибка (с. к. о.) $\Delta(T) = M(T(X) - g)^2$.
Определение: Оценка минимизирующая с. к. о. в данном классе оценок T_g называется оптимальной в среднеквадратичном смысле и обозначается T^* :

$$T^* = argmin_{T \in T_g} \Delta(T)$$

Пусть требуется оценить заданную параметрическую функцию $\tau(\theta)$ в модели $F = F(x; \theta), \theta \in \Theta$ по соответствующей выборке $X = (X_1, \ldots, X_n)$. Обозначим τ_τ класс всех несмещенных оценок $T = T(X)$ для $\tau(\theta)$ и предположим, что он не пуст. Дополнительно предположим, что дисперсии всех оценок из класса τ_τ конечны: $D\theta T = M_\theta(T - \tau(\theta))2 < \infty$, в этом случае мерой точности оценок является их дисперсия.
Утверждение: Для несмещенных оценок среднеквадратичное отклонение совпадает с ее дисперсией, а для смещенной оценки больше ее дисперсии.
Доказательство:

$$M\theta(T - \tau)^2 = M(T - MT + MT - \tau)^2 = M(T - MT)^2 + M(MT - \tau)^2 + 2M((T - MT)(MT - \tau)) = DT + b^2 + 0;$$

$$b^2 = 0 \Leftrightarrow MT = \tau$$

Теорема Рао-Блэкуэлла-Колмогорова: Оптимальная оценка, если она существует, является функцией от достаточной статистики.
По определению достаточная статистика T = T(X) называется полной, если для всякой функции $\varphi(T)$ из того, что

$$M_\theta \varphi(T) = 0, \ \forall \theta$$

следует $\varphi(t) \equiv 0$ на всем множестве значений статистики T.
Теорема: Если существует полная достаточная статистика, то всякая функция от нее является оптимальной оценкой своего математического ожидания.
Итак, пусть существует полная достаточная статистика T = T(X) и требуется оценить заданную параметрическую функцию $\tau(\theta)$. Тогда:
1)Если существует какая-то несмещенная оценка $\tau(\theta)$, то существует и несмещенная оценка, являющаяся функцией от T; можно так же сказать, что если нет несмещенныхоценок вида H(T), то класс несмещенных оценок τ_τ для $\tau(\theta)$ пуст;
2)оптимальная (н.о.р.м.д.) оценка когда она существует, всегда является функцией от T и она однозначно определяется уравнением $M_\theta H(T) = \tau(\theta)$
3)оптимальную оценку τ^* можно искать по формуле:

$$\tau^* = H(T) = M_\theta(T_1 \mid T)$$

исходя из любой несмещенной оценки T_1 функции $\tau(\theta)$.

3.3.1 Геометрическое распределение

Найдем оптимальную оценку для параметра $p = \theta_1$:
Рассмотрим произвольную из сгенерированных выборку $X = (X_1, \ldots, X_m)$ из геометрического распределения. Для параметра θ_1 множество несмещенных оценок не пусто.

$$\bar{x} = M_1 = \sum_{k=1}^{\infty} pq^{k-1}k = p\sum_{k=1}^{\infty} q^{k-1}k = \frac{1}{p}$$

$$D = M_c^2 = M(x - \bar{x})^2 = p\sum_{k=1}^{\infty} q^{k-1}k^2 = \frac{q}{p^3} = M_1\frac{q}{p^2}$$

По методу моментов оценка $\hat{\theta}_1 = \ldots = \ldots = \frac{1}{\theta_1}$

$$M = \ldots = \frac{1}{\theta_1}$$

$$D\xi = \frac{1 - \theta_1}{\theta_1^2}$$

Запишем логарифмическую функцию правдоподобия:

$$lnL(p) = ln[p^n\prod_{i=1}^{n}(1-p)^{x_i-1}] = ln[p^n] + ln[\prod_{i=1}^{n}(1-p)^{x_i-1}] = nlnp + \sum_{i=1}^{n} ln((1-p)^{x_i-1}) = nlnp + \sum_{i=1}^{n}(x_i-1)ln(1-p) = nlnp + ln(1-p)\sum_{i=1}^{n}(x_i-1) = nlnp + \sum_{i=1}^{n}(x_i-1)ln($$

Условия экстремума:

$$\frac{dlnL}{dp} = (nlnp + ln(1-p)\sum_{i=1}^{n} x_i - nln(1-p)) = n\cdot\frac{1}{p} + \frac{-1}{1-p}\sum_{i=1}^{n} x_i - n\frac{-1}{1-p} = 0,$$

$$n\cdot\frac{1}{p} + \frac{1}{p-1}\sum_{i=1}^{n} x_i - n\cdot\frac{1}{p-1} = 0,$$

Преобразуем:

$$\frac{1}{p-1}(\sum_{i=1}^{n} x_i - n) = -n\cdot\frac{1}{p}$$

$$-\frac{p-1}{p} = \frac{\sum_{i=1}^{n} x_i - n}{n}$$

$$\frac{1}{p} - 1 = \frac{1}{n}\sum_{i=1}^{n} x_i - 1$$

$$\frac{1}{p} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$p = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} x_i}$$

Таким образом, в качестве оценки получаем:

$$\hat{\theta}_1 = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} x_i} = \frac{1}{\theta}$$

3.3.2 Распределение Максвелла

Для простоты предположим, что все частоты $p_i, i = 1, n$ равны единице.
Запишем функцию максимального правдоподобия для закона Максвелла.

$$L(\lambda) = (\frac{2}{\pi})^{\frac{n}{2}} \frac{\prod_{i=1}^{n} x_i^2}{\lambda^{3n}} e^{-\frac{1}{2\lambda^2}\sum_{i=1}^{n} x_i^2}$$

$$ln(L(\lambda)) = 2\sum_{i=1}^{n} lnx_j - 3nln\lambda - \frac{n}{2} ln\frac{2}{\pi} - \frac{1}{2\lambda^2}\sum_{i=1}^{n} x_i^2$$

$$\frac{\partial(lnL(\lambda))}{\partial\lambda} = -\frac{3n}{\lambda} + \frac{\sum_{i=1}^{n} x_i^2}{\lambda^3} = 0$$

Переходя к статистическому ряду (не все p_i равны 1, $i = 1, n$), получим уравнение для нахождения λ:

$$\lambda = \sqrt{\frac{\sum_{i=1}^{m} p_i x_i^2}{3n}}$$

Оценка методом моментов:
Поскольку по выборке оценивается лишь один параметр, то для нахождения λ используемся оценку математического ожидания.

$$M[X] = \bar{X} = 2\lambda\sqrt{\frac{2}{\pi}},$$

где $\bar{X} = \frac{1}{n}\sum_{i=1}^{m} x_i p_i$, $\sum_{i=1}^{m} p_i = n$
Отсюда $\lambda = \sqrt{\frac{\pi}{8}}\bar{X}$

3.4 Работа с данными

В данном задании проведем анализ реальных данных. Воспользуемся нетипичной интерпретацией геометрического распределения - игра бейсбол. В бейсболе геометрическое распределение полезно для анализа вероятности того, что отбивающий получит удар, прежде чем он получит три удара; здесь цель - добиться успеха за 3 испытания.
Рассмотрим финальную серию чемпионата МЛБ-2020 между Лос Анджелес Доджерс и Тампа-Бэй Рэйс. Проанализируем статистику бэттеров первых 5 матчей, чтобы оценить какие бэттеры должны чаще выходить на поле в последнем матче серии и сравним с тем, как на самом деле это было. Почему принято решение выбрать именно такой подход? Всё из-за того, что чем выше процент отбивания у бэттера, тем выше вероятность того, что произойдет успех. Докажем это:
Пусть вероятность того, что бэттер отобьёт удар равна p. Тогда рассчитаем вероятность успеха:

$$P(X = 0) + P(X = 1) + P(X = 2) = q^0 p + q^1 p + q^2 p = p + (1 - p)p + (1 - p)^2 p = p + p - p^2 + p - 2p^2 + p^3 = p(p^2 - 3p + 3)$$

Пройдемся по циклу и убедимся, что при увеличении p увеличивается и вероятность успеха.

In [63]:

```
for p in [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]:
    print(p*(p**2-3*p+3))
```

```
0.271
0.488
0.6569999999999999
0.784
0.875
0.9359999999999999
0.9730000000000002
0.9919999999999999
0.9989999999999999
```

Очевидно, что с ростом вероятности p растет и вероятность успеха.
Рассмотрим статистические показатели, которые нам пригодятся:
AB - At Bats = PA - BB - IBB - HBP - CI - SF - SH (На бите): Выходы на биту бэттера, за исключением уоков, ударов мячом, пожертвованных ударов, умышленных помех со стороны защиты или других препятствий.
H - Hits (Хиты): общее количество хитов (1B, 2B, 3B и HR). Хит - удар, давший возможность выйти на базу. При этом защита не совершила ошибку.
BA - Batting average = H / AB (он же AVG - средний коэффициент результативности отбивания): число хитов, деленное на число выходов на биту.

In [64]:

```
match_1 = pd.read_csv('/home/alexander/Рабочий стол/Курсовая/Матч-1.csv',',',parse_dates=['Игрок'])
match_2 = pd.read_csv('/home/alexander/Рабочий стол/Курсовая/Матч-2.csv',',',parse_dates=['Игрок'])
match_3 = pd.read_csv('/home/alexander/Рабочий стол/Курсовая/Матч-3.csv',',',parse_dates=['Игрок'])
match_4 = pd.read_csv('/home/alexander/Рабочий стол/Курсовая/Матч-4.csv',',',parse_dates=['Игрок'])
match_5 = pd.read_csv('/home/alexander/Рабочий стол/Курсовая/Матч-5.csv',',',parse_dates=['Игрок'])
```

In [65]:

```
match_1
```

Out[65]:

	Игрок	Команда	AB	H	BA
0	Adames W.	TAM	2	-	0.000
1	Arozarena R.	TAM	3	-	0.000
2	Barnes A.	LAD	4	-	0.000
3	Bellinger C.	LAD	4	1	0.250
4	Betts M.	LAD	4	2	0.500
5	Brosseau M.	TAM	1	1	1.000
6	Choi Ji-Man	TAM	-	-	0.000
7	Diaz Y.	TAM	4	1	0.250
8	Hernandez E.	LAD	2	1	0.500
9	Kiermaier K.	TAM	3	2	0.667
10	Lowe B.	TAM	4	-	0.000
11	Margot M.	TAM	4	1	0.250
12	Meadows A.	TAM	2	-	0.000
13	Muncy M.	LAD	4	2	0.500
14	Pederson J.	LAD	2	-	0.000
15	Renfroe H.	TAM	2	-	0.000
16	Seager C.	LAD	2	-	0.000
17	Taylor C.	LAD	3	2	0.667
18	Turner J.	LAD	4	1	0.250
19	Wendle J.	TAM	4	1	0.250
20	Zunino M.	TAM	3	-	0.000
21	Смит У.	LAD	5	1	0.200

In [66]:

```
match_2
```

Out[66]:

	Игрок	Команда	AB	H	BA
0	Adames W.	TAM	4	1	0.167
1	Arozarena R.	TAM	3	1	0.167
2	Barnes A.	LAD	1	-	0.000
3	Bellinger C.	LAD	3	-	0.143
4	Betts M.	LAD	3	-	0.286
5	Brosseau M.	TAM	2	-	0.333
6	Choi Ji-Man	TAM	3	1	0.333
7	Diaz Y.	TAM	1	1	0.400
8	Hernandez E.	LAD	1	-	0.333
9	Kiermaier K.	TAM	4	-	0.286
10	Lowe B.	TAM	5	2	0.222
11	Margot M.	TAM	3	2	0.429
12	Meadows A.	TAM	3	1	0.200
13	Muncy M.	LAD	3	-	0.286
14	Pederson J.	LAD	1	-	0.000
15	Phillips B.	TAM	-	-	0.000
16	Pollock A.	LAD	2	-	0.000
17	Renfroe H.	TAM	-	-	0.000
18	Rios E.	LAD	2	-	0.000
19	Seager C.	LAD	4	2	0.333
20	Taylor C.	LAD	4	1	0.429
21	Turner J.	LAD	4	1	0.250
22	Wendle J.	TAM	3	1	0.286
23	Zunino M.	TAM	4	-	0.000
24	Смит У.	LAD	4	1	0.222

In [67]:

```
match_3
```

Out[67]:

	Игрок	Команда	AB	H	BA
0	Adames W.	TAM	3	1	0.222
1	Arozarena R.	TAM	4	1	0.200
2	Barnes A.	LAD	3	1	0.125
3	Bellinger C.	LAD	4	1	0.182
4	Betts M.	LAD	5	2	0.333
5	Choi Ji-Man	TAM	4	-	0.143
6	Hernandez E.	LAD	1	-	0.250
7	Kiermaier K.	TAM	2	-	0.222
8	Lowe B.	TAM	4	-	0.154
9	Margot M.	TAM	3	1	0.400
10	Meadows A.	TAM	4	1	0.222
11	Muncy M.	LAD	4	2	0.364
12	Pederson J.	LAD	3	1	0.167
13	Seager C.	LAD	3	1	0.333
14	Taylor C.	LAD	4	-	0.273
15	Tsutsugo Y.	TAM	1	-	0.000
16	Turner J.	LAD	5	2	0.308
17	Wendle J.	TAM	3	-	0.200
18	Zunino M.	TAM	2	-	0.000
19	Перес М.	TAM	-	-	0.000
20	Смит У.	LAD	4	-	0.154

In [68]:

```
match_4
```

Out[68]:

	Игрок	Команда	AB	H	BA
0	Adames W.	TAM	4	1	0.231
1	Arozarena R.	TAM	4	3	0.357
2	Bellinger C.	LAD	4	-	0.133
3	Betts M.	LAD	5	-	0.235
4	Brosseau M.	TAM	2	1	0.400
5	Choi Ji-Man	TAM	-	-	0.143
6	Diaz Y.	TAM	3	-	0.250
7	Hernandez E.	LAD	4	1	0.250
8	Kiermaier K.	TAM	4	2	0.308
9	Lowe B.	TAM	4	1	0.176
10	Margot M.	TAM	2	-	0.333
11	Meadows A.	TAM	2	-	0.182
12	Muncy M.	LAD	4	1	0.333
13	Pederson J.	LAD	2	2	0.375
14	Phillips B.	TAM	1	1	1.000
15	Pollock A.	LAD	2	1	0.250
16	Renfroe H.	TAM	4	1	0.167
17	Seager C.	LAD	5	4	0.500
18	Taylor C.	LAD	5	1	0.250
19	Tsutsugo Y.	TAM	1	-	0.000
20	Turner J.	LAD	5	4	0.444
21	Wendle J.	TAM	1	-	0.182
22	Zunino M.	TAM	2	-	0.000
23	Смит У.	LAD	4	1	0.176

In [69]:

```
match_5
```

Out[69]:

	Игрок	Команда	AB	H	BA
0	Adames W.	TAM	4	-	0.176
1	Arozarena R.	TAM	4	1	0.333
2	Barnes A.	LAD	2	-	0.100
3	Bellinger C.	LAD	4	1	0.158
4	Betts M.	LAD	5	1	0.227
5	Brosseau M.	TAM	-	-	0.400
6	Choi Ji-Man	TAM	-	-	0.143
7	Diaz Y.	TAM	3	2	0.364
8	Hernandez E.	LAD	1	-	0.222
9	Kiermaier K.	TAM	3	2	0.375
10	Lowe B.	TAM	4	-	0.143
11	Margot M.	TAM	3	2	0.400
12	Meadows A.	TAM	2	-	0.154
13	Muncy M.	LAD	3	2	0.389
14	Pederson J.	LAD	2	1	0.400
15	Renfroe H.	TAM	1	-	0.143
16	Seager C.	LAD	3	1	0.471
17	Taylor C.	LAD	4	-	0.200
18	Tsutsugo Y.	TAM	1	-	0.000
19	Turner J.	LAD	4	-	0.364
20	Wendle J.	TAM	4	-	0.133
21	Zunino M.	TAM	2	-	0.000
22	Перес М.	TAM	-	-	0.000
23	Смит У.	LAD	4	-	0.143

In [70]:

```
itog = pd.read_csv('/home/alexander/Рабочий стол/Курсовая/Итого.csv',',',parse_dates=['Игрок'])
itog
```

Out[70]:

	Игрок	Команда	AB	H	BA
0	Adames W.	TAM	17	3	0,176470588235294
1	Arozarena R.	TAM	18	6	0,333333333333333
2	Barnes A.	LAD	10	1	0,1
3	Bellinger C.	LAD	19	3	0,157894736842105
4	Betts M.	LAD	22	5	0,227272727272727
5	Brosseau M.	TAM	5	2	0,4
6	Choi Ji-Man	TAM	7	1	0,142857142857143
7	Diaz Y.	TAM	11	4	0,363636363636364
8	Hernandez E.	LAD	9	2	0,222222222222222
9	Kiermaier K.	TAM	16	6	0,375
10	Lowe B.	TAM	21	3	0,142857142857143
11	Margot M.	TAM	15	6	0,4
12	Meadows A.	TAM	13	2	0,153846153846154
13	Muncy M.	LAD	18	7	0,388888888888889
14	Pederson J.	LAD	10	4	0,4
15	Phillips B.	TAM	1	1	1
16	Pollock A.	LAD	4	1	0,25
17	Rios E.	LAD	2	0	0
18	Seager C.	LAD	17	8	0,470588235294118
19	Taylor C.	LAD	20	4	0,2
20	Turner J.	LAD	22	8	0,363636363636364
21	Смит У.	LAD	21	3	0,142857142857143
22	Renfroe H.	TAM	7	1	0,142857142857143
23	Wendle J.	TAM	15	2	0,133333333333333
24	Zunino M.	TAM	13	0	0
25	Tsutsugo Y.	TAM	3	0	0

А теперь попробуем спрогнозировать кто из бэттеров будет подходить к бите больше в своей команде, анализируя приведенные выше статистические данные по итогам первых 5 матчей финальной серии. Выведем данные в порядке убывания.

In [71]:

```
prognoz_TAM = pd.read_csv('/home/alexander/Рабочий стол/Курсовая/Итого_TAM.csv',',',parse_dates=['Игрок'])
prognoz_TAM
```

Out[71]:

	Игрок	Команда	AB	H	BA	Комментарий
0	Margot M.	TAM	15	6	0,4	Лучший BA, один из лучших AB
1	Arozarena R.	TAM	18	6	0,333333333333333	Один из лидеров по AB и отличный BA
2	Adames W.	TAM	17	3	0,176470588235294	Один из лидеров по AB и средний BA
3	Lowe B.	TAM	21	3	0,142857142857143	Лучший AB, но значимо хуже BA
4	Kiermaier K.	TAM	16	6	0,375	Очень хороший BA и неплохой AB
5	Wendle J.	TAM	15	2	0,133333333333333	Примерно равные средние показатели
6	Meadows A.	TAM	13	2	0,153846153846154	Примерно равные средние показатели
7	Diaz Y.	TAM	11	4	0,363636363636364	Отличный BA, но все же низкий AB
8	Zunino M.	TAM	13	0	0	Приличный AB, но худший BA
9	Choi Ji-Man	TAM	7	1	0,142857142857143	Низкие AB и BA
10	Renfroe H.	TAM	7	1	0,142857142857143	Низкие AB и BA
11	Brosseau M.	TAM	5	2	0,4	Один из худших AB, но отличный BA
12	Tsutsugo Y.	TAM	3	0	0	Очень низкие AB и BA
13	Phillips B.	TAM	1	1	1	Подходил к бите всего лишь раз

А теперь посмотрим, сколько в итоге было подходов к бите у игроков TAM. Выведем данные в порядке убывания.

In [72]:

```
match_6_TAM = pd.read_csv('/home/alexander/Рабочий стол/Курсовая/Матч-6_TAM.csv',',',parse_dates=['Игрок'])
match_6_TAM
```

Out[72]:

	Игрок	Команда	AB	H	BA
0	Margot M.	TAM	4	0	0.316
1	Arozarena R.	TAM	4	2	0.364
2	Adames W.	TAM	4	0	0.143
3	Lowe B.	TAM	3	0	0.125
4	Kiermaier K.	TAM	3	1	0.368
5	Wendle J.	TAM	3	0	0.111
6	Meadows A.	TAM	3	1	0.188
7	Zunino M.	TAM	3	1	0.063
8	Choi Ji-Man	TAM	2	0	0.111
9	Diaz Y.	TAM	1	0	0.333
10	Renfroe H.	TAM	1	0	0.125
11	Brosseau M.	TAM	1	0	0.333
12	Phillips B.	TAM	0	0	1.000

In [73]:

```
prognoz_LAD = pd.read_csv('/home/alexander/Рабочий стол/Курсовая/Итого_LAD.csv',',',parse_dates=['Игрок'])
prognoz_LAD
```

Out[73]:

	Игрок	Команда	AB	H	BA	Комментарий
0	Turner J.	LAD	22	8	0,363636363636364	Лучший AB, один из лучших BA
1	Betts M.	LAD	22	5	0,227272727272727	Лучший AB, высокий BA
2	Seager C.	LAD	17	8	0,470588235294118	Высокий AB, лучший BA
3	Muncy M.	LAD	18	7	0,388888888888889	Высокий AB, один из лучших BA
4	Taylor C.	LAD	20	4	0,2	Высокий AB, неплохой BA
5	Смит У.	LAD	21	3	0,142857142857143	Высокий AB, неплохой BA
6	Bellinger C.	LAD	19	3	0,157894736842105	Высокий AB, неплохой BA
7	Pederson J.	LAD	10	4	0,4	Средний AB, высокий BA
8	Hernandez E.	LAD	9	2	0,222222222222222	Средние показатели
9	Barnes A.	LAD	10	1	0,1	Средние показатели
10	Pollock A.	LAD	4	1	0,25	Низкий AB, неплохой BA
11	Rios E.	LAD	2	0	0	Всего лишь 2 подхода к бите

А теперь посмотрим, сколько в итоге было подходов к бите у игроков LAD. Выведем данные в порядке убывания.

In [74]:

```
match_6_LAD = pd.read_csv('/home/alexander/Рабочий стол/Курсовая/Матч-6_LAD.csv',',',parse_dates=['Игрок'])
match_6_LAD
```

Out[74]:

	Игрок	Команда	AB	H	BA
0	Betts M.	LAD	4	2	0.269
1	Muncy M.	LAD	4	-	0.318
2	Turner J.	LAD	3	-	0.320
3	Seager C.	LAD	3	-	0.400
4	Taylor C.	LAD	3	1	0.217
5	Смит У.	LAD	3	1	0.167
6	Bellinger C.	LAD	3	-	0.136
7	Barnes A.	LAD	3	1	0.154
8	Pollock A.	LAD	2	-	0.167
9	Hernandez E.	LAD	1	-	0.200
10	Pederson J.	LAD	-	-	0.400
11	Rios E.	LAD	-	-	0.000

Оценка того, кто больше из игроков будет подходить к бите в своей команде оказалось достаточно точной, несмотря на то, что данные были взяты лишь по 5 последним играм, что подтверждает важность этих показателей у игроков. Конечно же, по данным за такой короткий период корректировку могли внести такие ситуации как, например, травмы игроков, из-за которых они не смогли бы участвовать в 6 финальном матче или же участвовать в нём не в полной мере.
Обычно в бейсболе статистику игроков оценивают по всему прошедшему сезону, что даст более высокую точность.

Как можно заметить, рассмотренные статистические показатели у игроков команды LAD лучше, чем у игроков команды TAM. Отсюда неудивительно, что LAD выиграли не только последний матч у TAD со счетом 3:1, но и всю финальную серию со счетом 4:2 и стали чемпионами MLB-2020.

Ссылка на данные: https://www.scoreboard.com/ru/baseball/usa/mlb-2020/ (https://www.scoreboard.com/ru/baseball/usa/mlb-2020/)

Задачки раз

Имеется выборка $X = (X_1, \ldots, X_n)$ из распределения ξ. Случайная величина ξ имеет распределение Бернулли с параметром $p = \theta$. Проверьте, является ли оценка $\hat{\theta}$ параметра θ несмещенной и состоятельной, если:

1. $\hat{\theta} = \bar{X}$.
2. $\hat{\theta} = \bar{X} + \frac{1}{n}$.
3. $\hat{\theta} = \frac{1}{10}X_1 + \frac{1}{5}X_2 + \frac{7}{10}X_3$.
4. $\hat{\theta} = \frac{2^{n-1}}{2^n-1}X_1 + \frac{2^{n-2}}{2^n-1}X_2 + \ldots + \frac{2^{n-n}}{2^n-1}X_n$.
5. $\hat{\theta} = \frac{1}{n}\left(\frac{1}{X_1+\gamma} + \frac{1}{X_2+\gamma} + \ldots + \frac{1}{X_n+\gamma}\right)$. Проверьте, существует ли такое значение параметра γ, при котором данная оценка будет несмещенной.
6. Проверьте, является ли несмещенной и состоятельной оценка дисперсии $\hat{Var}(\xi) = \frac{X_1(1-X_1)+\ldots+X_n(1-X_n)}{n}$. Как изменится ответ для оценки $\hat{Var}(\xi) = \frac{X_1(1-X_1)+\ldots+X_n(1-X_n)}{n-5}$?
7. $\hat{\theta} = \frac{\gamma}{n}X_2 + \ldots + \frac{\gamma}{n}X_n$, где n - четное. Найдите параметр γ, при котором данная оценка будет несмещенной и состоятельной.
8. Самостоятельно придумайте несмещенную и состоятельную оценку для 100-го начального момента $E(\xi^{100})$.

Решение

1. Оценка является несмещенной, поскольку:

$$E(\hat{\theta}) = E(\bar{X}) = \frac{n * E(\xi)}{n} = \theta$$

Оценка является состоятельной, так как:

$$\lim_{n\to\infty} E(\hat{\theta}) = \lim_{n\to\infty} E(\bar{X}) = \lim_{n\to\infty} \theta = \theta$$

$$\lim_{n\to\infty} Var(\hat{\theta}) = \lim_{n\to\infty} Var(\bar{X}) = \lim_{n\to\infty} \frac{n}{n^2} Var(\xi) = 0$$

1. Оценка является смещенной, поскольку:

$$E(\hat{\theta}) = E(\bar{X} + \frac{1}{n}) = \frac{n * E(\xi)}{n} = \theta + \frac{1}{n} \neq \theta$$

Оценка является состоятельной, так как:

$$\lim_{n\to\infty} E(\hat{\theta}) = \lim_{n\to\infty} \theta + \frac{1}{n} = \theta$$

$$\lim_{n\to\infty} Var(\hat{\theta}) = \lim_{n\to\infty} Var(\bar{X} + \frac{1}{n}) = \lim_{n\to\infty} \frac{n}{n^2} Var(\xi) = 0$$

1. Оценка является несмещенной:

$$E(\hat{\theta}) = E(\frac{1}{10}X_1 + \frac{1}{5}X_2 + \frac{7}{10}X_3) = \frac{1}{10}E(\xi) + \frac{1}{5}E(\xi) + \frac{7}{10}E(\xi) = E(\xi) = \theta$$

Оценка не является состоятельной, так как:

$$\lim_{n\to\infty} Var(\hat{\theta}) = \lim_{n\to\infty} Var(\frac{1}{10}X_1 + \frac{1}{5}X_2 + \frac{7}{10}X_3) = \lim_{n\to\infty} Var(\frac{1}{10}\xi) + Var(\frac{1}{5}\xi) + Var(\frac{7}{10}\xi) = = \lim_{n\to\infty}\left(\frac{1}{100} + \frac{1}{25} + \frac{7}{10}\right)\theta(1-\theta) \neq 0$$

1. Заметим, что последовательность коэффициентов в выражении для оценки параметра θ формирует геометрическую прогрессию со знаменателем 0.5. Используя формулу для суммы членов геометрической прогрессии нетрудно показать, что оценка является несмещенной:

$$E(\hat{\theta}) = \frac{2^{n-1}}{2^n-1}E(X_1) + \frac{2^{n-2}}{2^n-1}E(X_2) + \ldots + \frac{2^{n-n}}{2^n-1}E(X_n) = = \theta\left(\frac{2^{n-1}}{2^n-1} + \frac{2^{n-2}}{2^n-1} + \ldots + \frac{2^{n-n}}{2^n-1}\right) = \frac{\frac{2^{n-1}}{2^n-1}(1-0.5^n)}{1-0.5}\theta = \theta$$

Для проверки состоятельности достаточно убедиться, что дисперсия оценки стремится к нулю:

$$\lim_{n\to\infty} Var(\hat{\theta}) = \lim_{n\to\infty}\left(\left(\frac{2^{n-1}}{2^n-1}\right)^2 + \ldots + \left(\frac{2^{n-n}}{2^n-1}\right)^2\right)\theta(1-\theta) = = \lim_{n\to\infty}\left(\left(2^{n-1}\right)^2 + \left(2^{n-2}\right)^2 + \ldots + \left(2^{n-n}\right)^2\right)\frac{\theta(1-\theta)}{(2^n-1)^2} = = \lim_{n\to\infty}\left(4^{n-1} + 4^{n-2} + \ldots + 4^{n-n}\right)\frac{\theta(1-\theta}{(2^n-1)}$$

Заметим, что знаменатель последовательности в скобках составляет 0.25. Откуда в итоге получаем, что такая оценка не является состоятельной:

$$\lim_{n\to\infty} Var(\hat{\theta}) = \lim_{n\to\infty}\left(4^{n-1}+4^{n-2}+\ldots+4^{n-n}\right)\frac{\theta(1-\theta)}{(2^n-1)^2} = = \lim_{n\to\infty}\frac{4^{n-1}(1-0.25^n)}{1-0.25} * \frac{\theta(1-\theta)}{(2^n-1)^2} = \frac{\theta(1-\theta)}{3} \neq 0$$

1. В первую очередь обратим внимание, что при $\gamma \notin \{0, -1\}$:

$$E\left(\frac{1}{\xi+\gamma}\right) = \theta\frac{1}{1+\gamma} + (1-\theta)\frac{1}{\gamma} = \frac{(1-\theta)+\gamma}{\gamma}$$

Очевидно, что оценка будет несмещенной при:

$$\frac{(1-\theta)+\gamma}{\gamma} = \theta$$

Однако, данное уравнение имеет решение для γ лишь при $\theta = 1$. Следовательно, искомого значения параметра γ не существует.

1. Оценка является несмещенной, поскольку:

$$E(\hat{Var}(\xi)) = E(\frac{X_1(1-X_1)+\ldots+X_n(1-X_n)}{n}) = \frac{1}{n}\left(E(X_1)E(1-X_1)+\ldots+E(X_n)E(1-X_n)\right) = = \frac{1}{n}(\underbrace{p(1-p)+\ldots+p(1-p)}_{n\text{ раз}}) = p(1-p) = Var(\xi)$$

Пользуясь свойством независимости убедимся в состоятельности оценки:

$$\lim_{n\to\infty} Var(\hat{Var}(\xi)) = \lim_{n\to\infty}\frac{1}{n^2}(Var(X_1-X_1^2)+\ldots+Var(X_n-X_n^2)) = = \lim_{n\to\infty}\frac{1}{n}Var(\xi-\xi^2) = 0$$

Последнее равенство следует из того, что $Var(\xi-\xi^2)$ является константой.

Для оценки $\hat{Var}(\xi) = \frac{X_1(1-X_1)+\ldots+X_n(1-X_n)}{n-5}$ нетрудно показать, что сохранится свойство состоятельности, однако будет нарушена несмещенность.

1. Нетрудно догадаться, что оценка будет несмещенной и состоятельной при $\gamma = 2$.
2. Легко проверить, что $E(\xi) = E(\xi^{100}) = p$, в связи с чем подойдет любая состоятельная и несмещенная оценка для θ из предыдущих пунктов.

Задачки два

Имеется выборка $X = (X_1, \ldots, X_n)$ из распределения ξ. Случайная величина ξ имеет равномерное распределение. Проверьте, является ли оценка $\hat{\theta}$ параметра θ несмещенной и состоятельной, если:

1. $\xi \sim U(0, \theta)$ и $\hat{\theta} = \max(X_1, \ldots, X_n)$. Если данная оценка не является несмещенной, то исправьте ситуацию предложив альтернативную оценку, а затем проверьте её состоятельность.

Решение

1. Нетрудно догадаться, что при $x \in \{0, \theta\}$ справедливо $F_{\hat{\theta}}(x) = F_{\xi}(x)^n = \left(\frac{x}{\theta}\right)^n$. Откуда, при $x \in \{0, \theta\}$ получаем функцию плотности $f_{\hat{\theta}}(x) = \frac{nx^{n-1}}{\theta^n}$.

   Отсюда несложно найти математическое ожидание рассматриваемой оценки и убедиться в том, что она не является несмещенной:

$$E(\hat{\theta}) = \int_0^{\theta} x\frac{nx^{n-1}}{\theta^n}dx = \int_0^{\theta}\frac{nx^n}{\theta^n}dx = \frac{nx^{n+1}}{(n+1)\theta^n}\Big|_0^{\theta} = \frac{n}{n+1}\theta \neq \theta$$

Однако очевидно, что оценка $\hat{\theta_2} = \frac{n+1}{n}\hat{\theta}$ будет являться несмещенной. Для проверки состоятельности оценки $\hat{\theta_2}$ достаточно убедиться, что её дисперсия стремится к нулю. Найдем выражение для дисперсии этой оценки:

$$\hat{Var}(\theta_2) = \left(\frac{n+1}{n}\right)^2 Var(\hat{\theta}) = \left(\frac{n+1}{n}\right)(E(\hat{\theta}^2) - E(\hat{\theta})^2) = = \left(\frac{n+1}{n}\right)^2\left(\int_0^{\theta} x^2\frac{nx^{n-1}}{\theta^n}dx - \left(\frac{n}{n+1}\theta\right)^2\right) = \frac{1}{n(n+2)}\theta^2$$

Состоятельность следует из того, что:

$$\lim_{n\to\infty}\hat{Var}(\theta_2) = \lim_{n\to\infty}\frac{1}{n(n+2)}\theta^2 = 0$$

4 Литература

[1] "Справочник по вероятностным распределениям" Р.Н.Вадзинский
https://fileskachat.com/view/10838_b741e0be3370efed892ccfe2b6c1358f.html
(https://fileskachat.com/view/10838_b741e0be3370efed892ccfe2b6c1358f.html)
[2] "Введение в математическую статистику" (Ивченко Г.И., Медведев Ю.И.)
http://bookre.org/reader?file=1221378&pg=101 (http://bookre.org/reader?file=1221378&pg=101)
[3] Power Maxwell distribution:
https://arxiv.org/pdf/1807.01200.pdf (https://arxiv.org/pdf/1807.01200.pdf)
[4] Geometric Distributionhttps:
https://brilliant.org/wiki/geometric-distribution/ (https://brilliant.org/wiki/geometric-distribution/)
[5]The Maxwell Distributionhttps:
https://randomservices.org/random/special/Maxwell.html (https://randomservices.org/random/special/Maxwell.html)
[6]The Geometric Distributionhttps:
https://randomservices.org/random/bernoulli/Geometric.html (https://randomservices.org/random/bernoulli/Geometric.html)
[7] Sampling from a Normal Distribution
http://bjlkeng.github.io/posts/sampling-from-a-normal-distribution/ (http://bjlkeng.github.io/posts/sampling-from-a-normal-distribution/)
[8] "Моделирование распределений В.В.Некруткин"
https://clck.ru/RHRdy (https://clck.ru/RHRdy)
[9] Сайт со статистикой бейсбола
https://www.scoreboard.com/ru/baseball/usa/mlb-2020/ (https://www.scoreboard.com/ru/baseball/usa/mlb-2020/)

In [ ]: