Курс "Практикум по математической статистике"

3 курс ФПМИ МФТИ, осень 2020

Домашнее задание 1. Свойства оценок

Дедлайн --- 5 октября 9:00

Это первое обязательное домашнее задание нашего курса. Мы предлагаем выполнять задания прямо в этом ноутбуке. Пожалуйста, не стирайте условия задач.

Информация о выполнении и курсе в целом есть в этой папке.

В этом и последующих заданиях вам потребуется выполнять генерацию случайных величин из некоторого распределения. Для этого вам понадобится библиотека **scipy.stats**. Мы настоятельно рекомендуем для генерации выборок использовать именно эту библиотеку.

Каждая задача оценивается в **10** баллов.

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sps
import seaborn as sns

sns.set(style='darkgrid')
%matplotlib inline
```

Зафиксируем `seed` для воспроизводимости.

In [2]:

```
np.random.seed(42)
```

Задача 1

Сгенерируйте выборку $X_1, \ldots, X_N$ из равномерного распределения на отрезке $[0, \theta]$ для $N = 10^4$.

In [3]:

```
N = int(1e4)   # use this
loc = 0
scale = 1
samples = sps.uniform.rvs(loc=loc, scale=scale, size=N)
```

In [4]:

```
samples
```

Out[4]:

```
array([0.37454012, 0.95071431, 0.73199394, ..., 0.94670792, 0.39748799,
       0.2171404 ])
```

Для всех $n \leqslant N$ посчитайте оценки параметра θ из теоретической задачи: $2\overline{X},\ \overline{X} + X_{(n)}/2,\ (n+1)X_{(1)},\ X_{(1)} + X_{(n)},\ \frac{n+1}{n}X_{(n)}$. Используйте векторные операции.

Подсказка: Могут быть полезными функции `np.arange`, `np.cumsum`, `np.maximum.accumulate` и `np.minimum.accumulate`

In [5]:

```
def PlotAbsLoss(estimates, theta, ylim : float = 0.5):
        plt.figure(figsize=(14, 9))

        for name in estimates:
            plt.plot(np.arange(1, len(estimates[name]) + 1), np.abs(estimates[name] - th
eta))

        plt.legend([name for name in estimates])
        plt.xlabel("n")
        plt.ylabel("Absolute loss")
        plt.ylim(0, ylim)

        plt.show()

def BestEstimate(estimates, theta):
        ests = {key : np.abs(estimates[key] - theta)[-1] for key in estimates}
        name = min(ests, key=ests.get)

        return (name, ests[name])

def RunExperiments(estimates, params, dist_name : str):
    for i in range(len(params)):
        print(dist_name, " with parameter ", params[i])
        name , loss = BestEstimate(estimates[i], params[i])
        print("for param = {} best estimate {} with abs loss {}".format(params[i], name,
loss))
        PlotAbsLoss(estimates[i], params[i])
```

In [6]:

```
def CalcUniformEstimates(samples):
    minimums = np.minimum.accumulate(samples)
    maximums = np.maximum.accumulate(samples)
    ns = np.arange(1, len(samples) + 1)
    means = np.cumsum(samples) / ns

    return {
        "2 * mean" : 2 * means ,
        "mean + X(n)/2" : means + maximums / 2 ,
        "(n+1) * X(1)" : (ns + 1) * minimums ,
        "X(1) + X(n)" : minimums + maximums ,
        "(n+1)/n * X(n)" : (ns + 1) / ns * maximums
    }
```

In [7]:

```
estimates = CalcUniformEstimates(samples)

estimates
```

Out[7]:

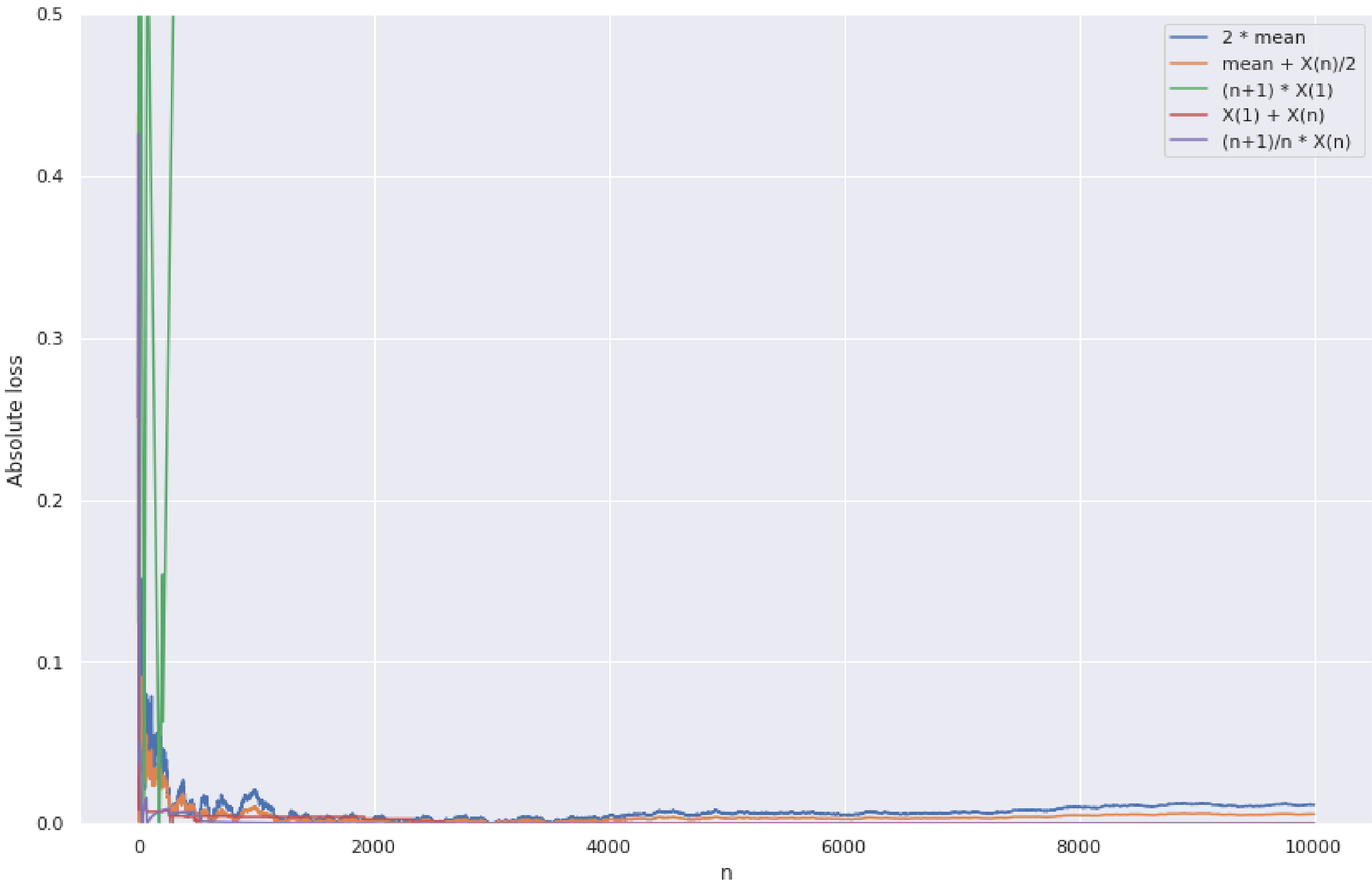
```
{'2 * mean': array([0.74908024, 1.32525443, 1.37149891, ..., 0.98839387, 0.98837452,
        0.98831912]),
 'mean + X(n)/2': array([0.56181018, 1.13798437, 1.16110661, ..., 0.99405577, 0.9940461 ,
        0.99401839]),
 '(n+1) * X(1)': array([0.74908024, 1.12362036, 1.49816048, ..., 0.11633592, 0.11634755,
        0.11635919]),
 'X(1) + X(n)': array([0.74908024, 1.32525443, 1.32525443, ..., 0.99972931, 0.99972931,
        0.99972931]),
 '(n+1)/n * X(n)': array([0.74908024, 1.42607146, 1.26761908, ..., 0.99981767, 0.99981766
,
        0.99981765])}
```

Постройте на одном графике разными цветами для всех оценок функции модуля разности оценки и истинного значения θ в зависимости от n. Если некоторые оценки (при фиксированном значении n) сильно отличаются от истинного значения параметра θ, то исключите их и постройте еще один график со всеми кривыми (для измененного значения θ). Для избавления от больших значений разности в начале ограничьте масштаб графика. Для наглядности точки можно соединить линиями.

Не забудьте подписать оси, а также добавить легенду к графику.

In [8]:

```
# YOUR CODE GOES HERE
PlotAbsLoss(estimates, scale)
```



In [9]:

```
# best estimate with n=N
print(BestEstimate(estimates, scale))
```

```
('(n+1)/n * X(n)', 0.0001823549465408414)
```

Какая оценка получилась лучше (в смысле упомянутого модуля разности при $n = N$)?

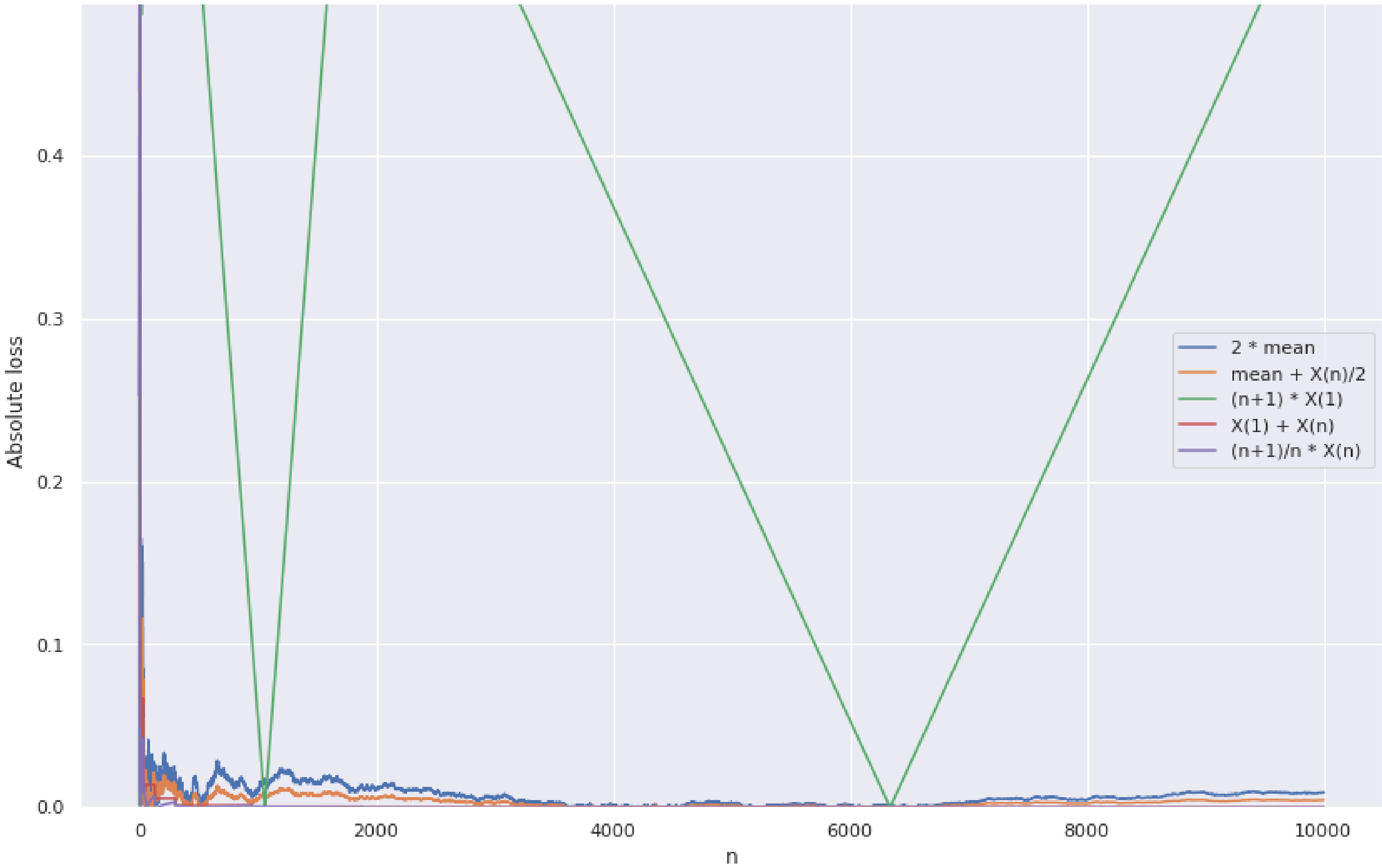
Ответ: $\frac{n+1}{n}X_{(n)}$

Проведите эксперимент для разных значений θ (количество графиков равно количеству значений θ)

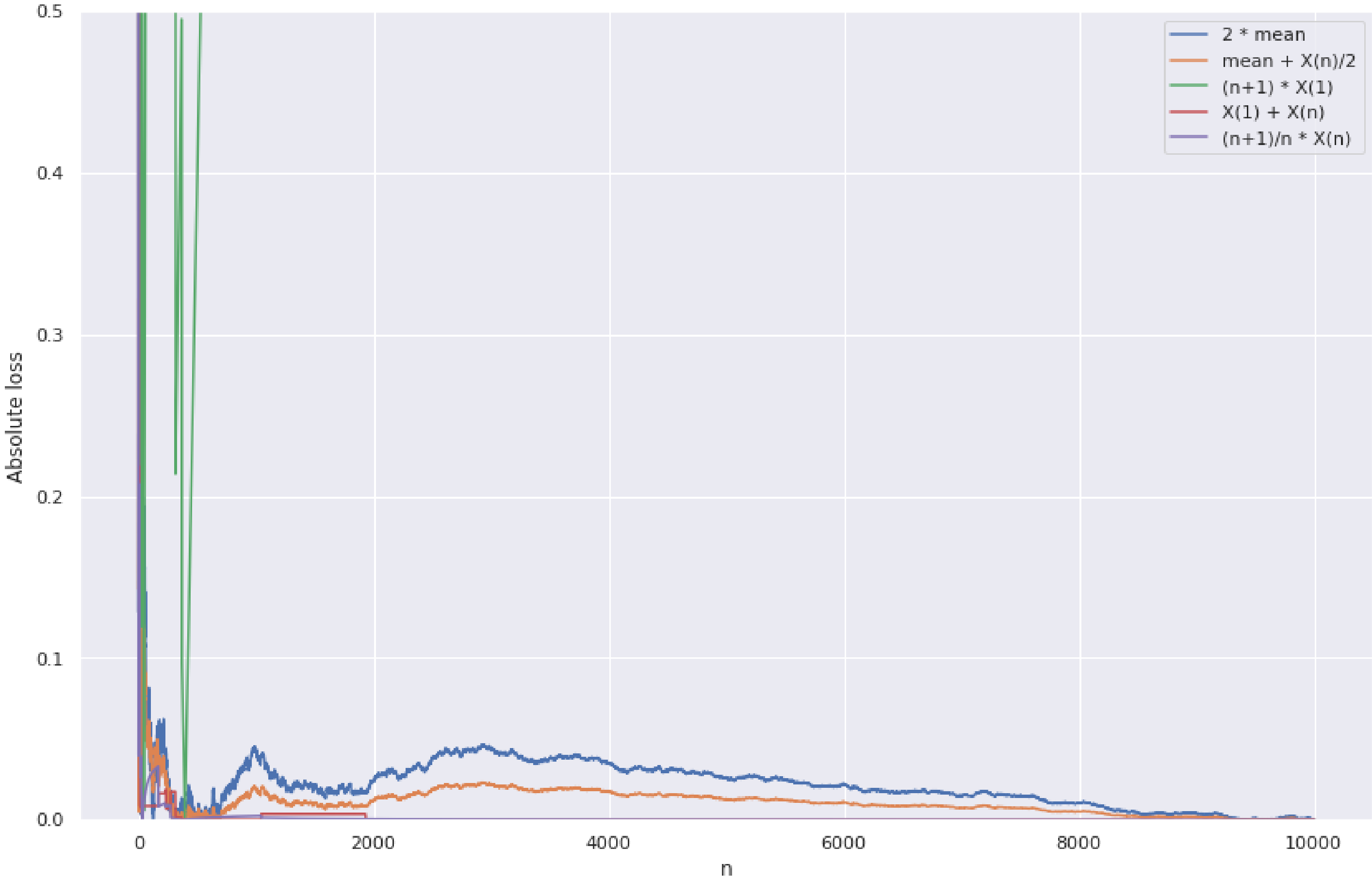
In [10]:

```
thetas = np.append(np.arange(1, 4.5, 0.5), 10, 100)
samples = np.array([sps.uniform.rvs(loc=0, scale=theta, size=N) for theta in thetas])
estimates = np.array([CalcUniformEstimates(sample) for sample in samples])

RunExperiments(estimates, thetas, "uniform dist.")
```

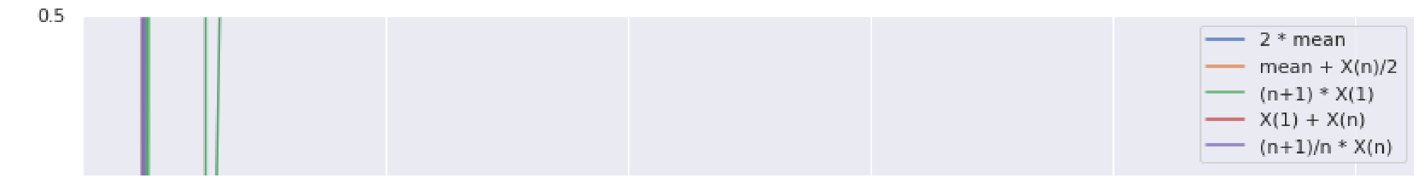
```
uniform dist.  with parameter  1.0
for param = 1.0 best estimate (n+1)/n * X(n) with abs loss 2.4819315859847535e-05
```

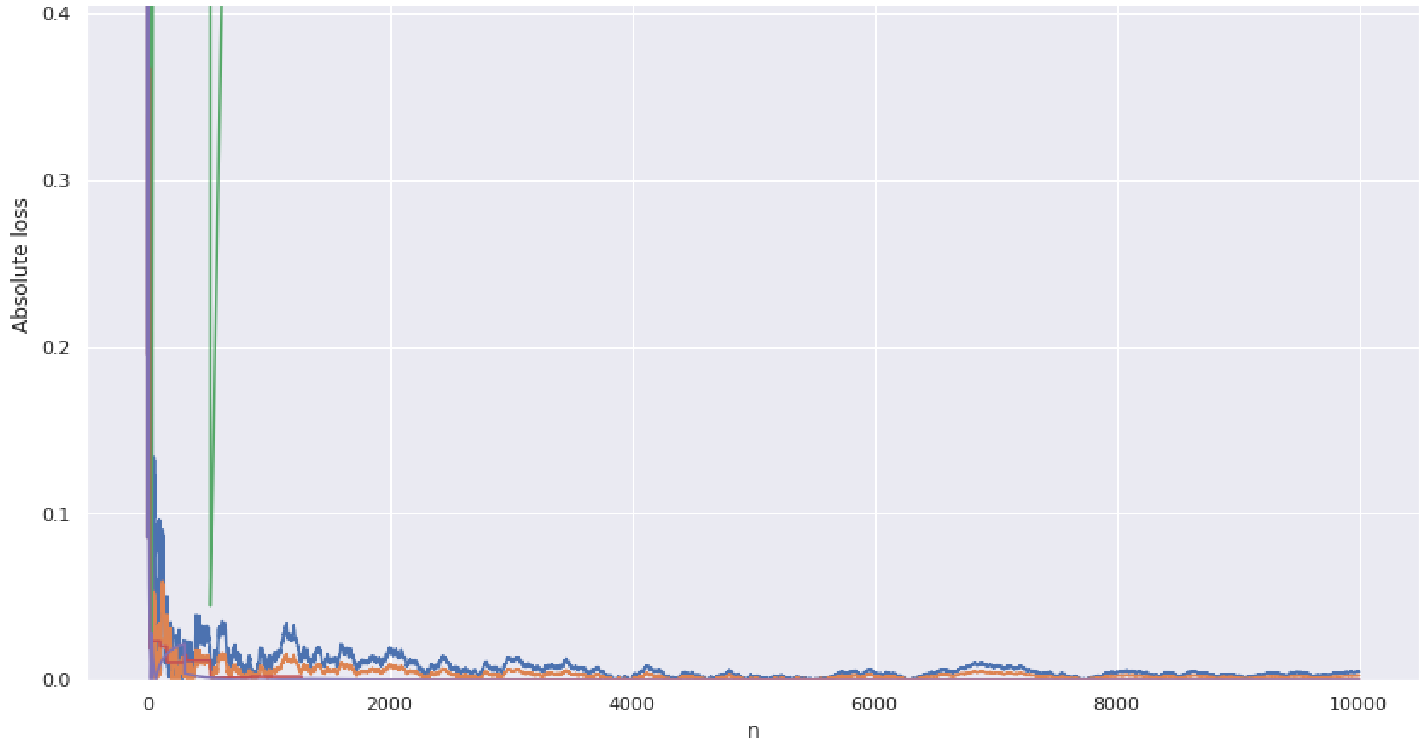



```
uniform dist.  with parameter  1.5
for param = 1.5 best estimate mean + X(n)/2 with abs loss 1.3246253942611474e-06
```

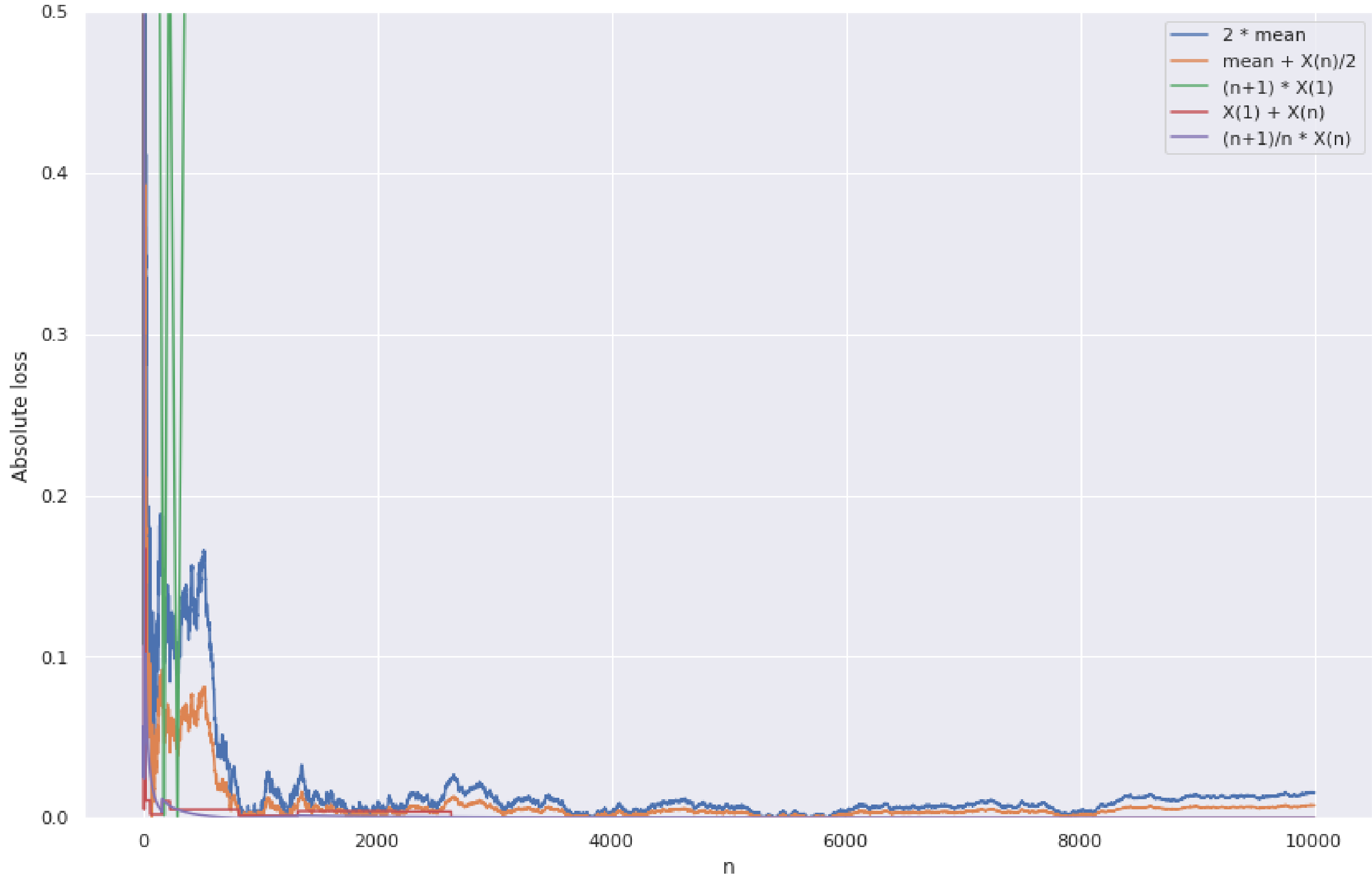



```
uniform dist.  with parameter  2.0
for param = 2.0 best estimate (n+1)/n * X(n) with abs loss 0.00022129850044683153
```

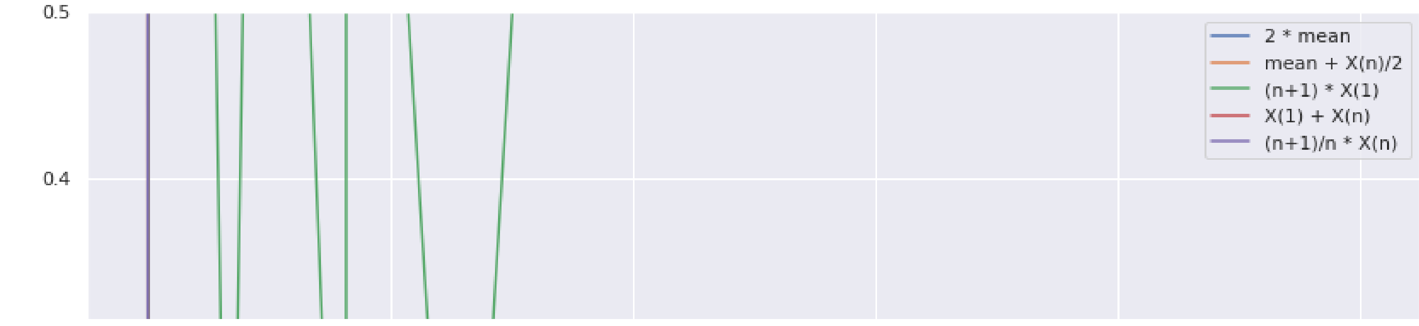


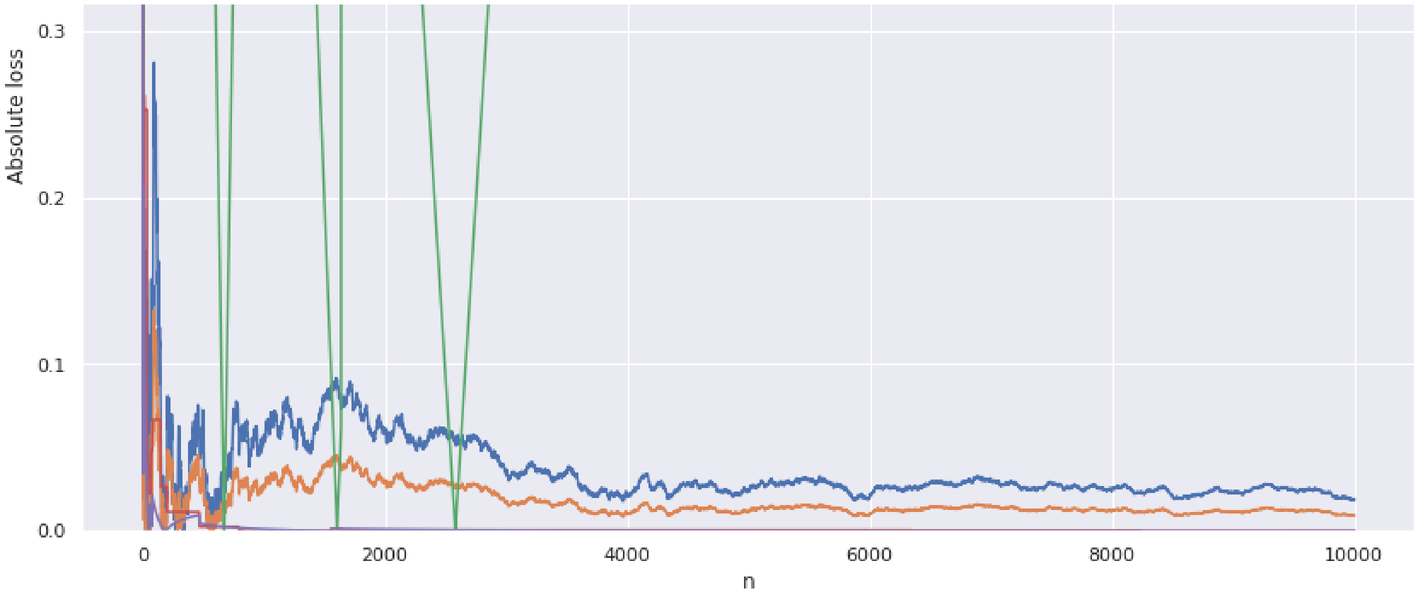



```
uniform dist.  with parameter  2.5
for param = 2.5 best estimate X(1) + X(n) with abs loss 2.7790935207328005e-05
```

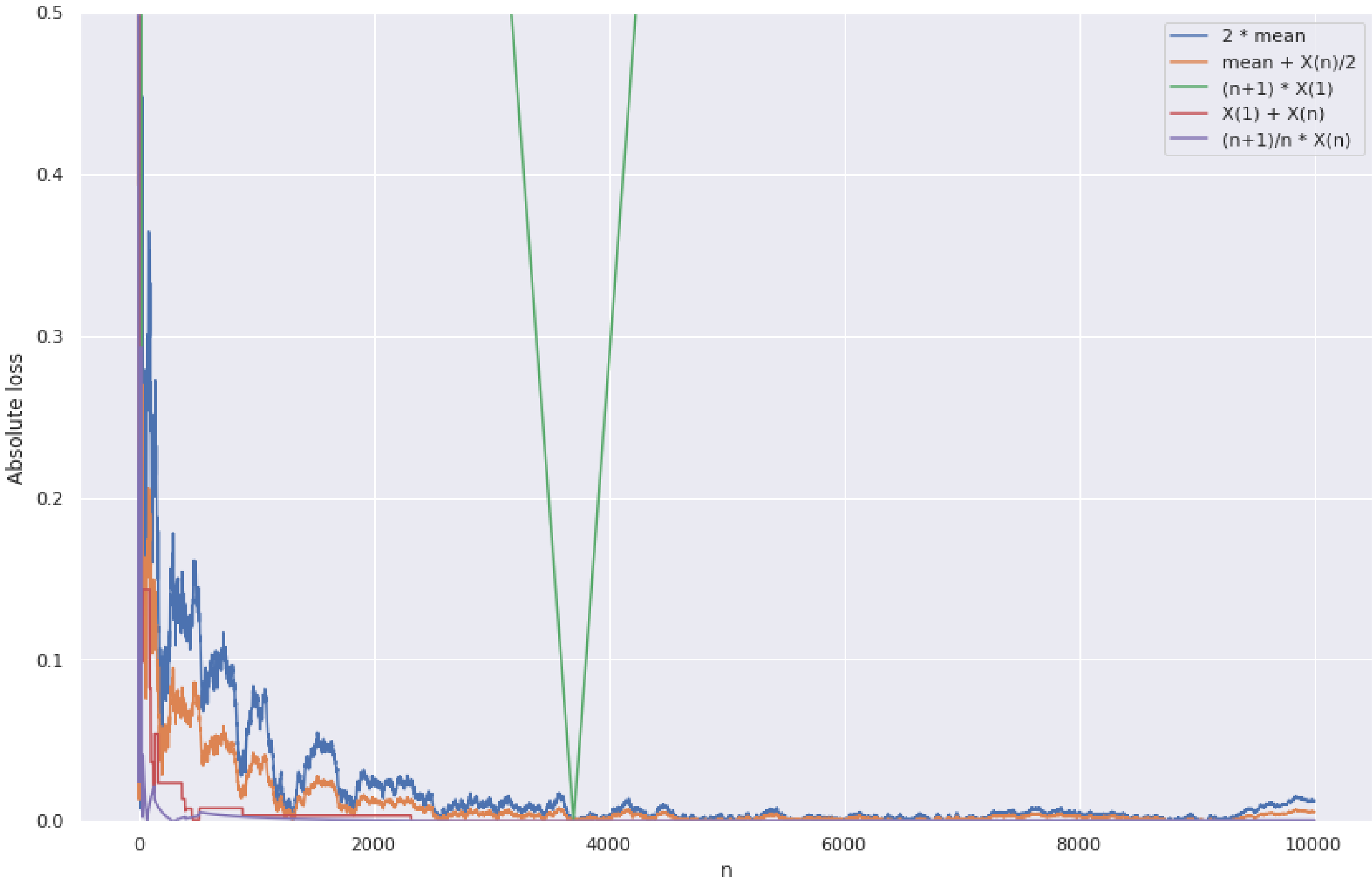



```
uniform dist.  with parameter  3.0
for param = 3.0 best estimate (n+1)/n * X(n) with abs loss 0.00011907004596967141
```

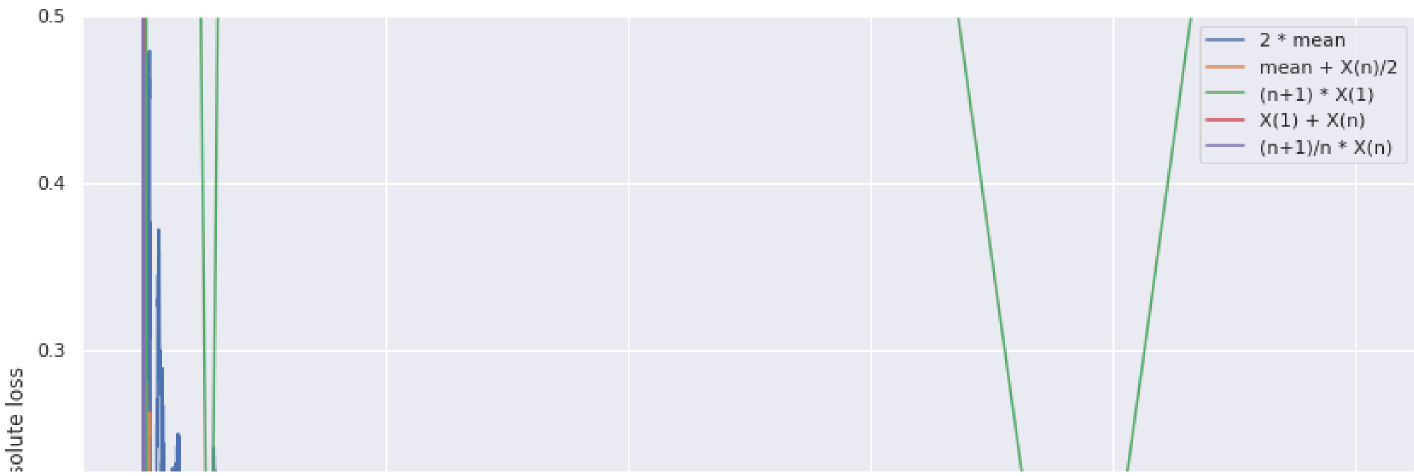


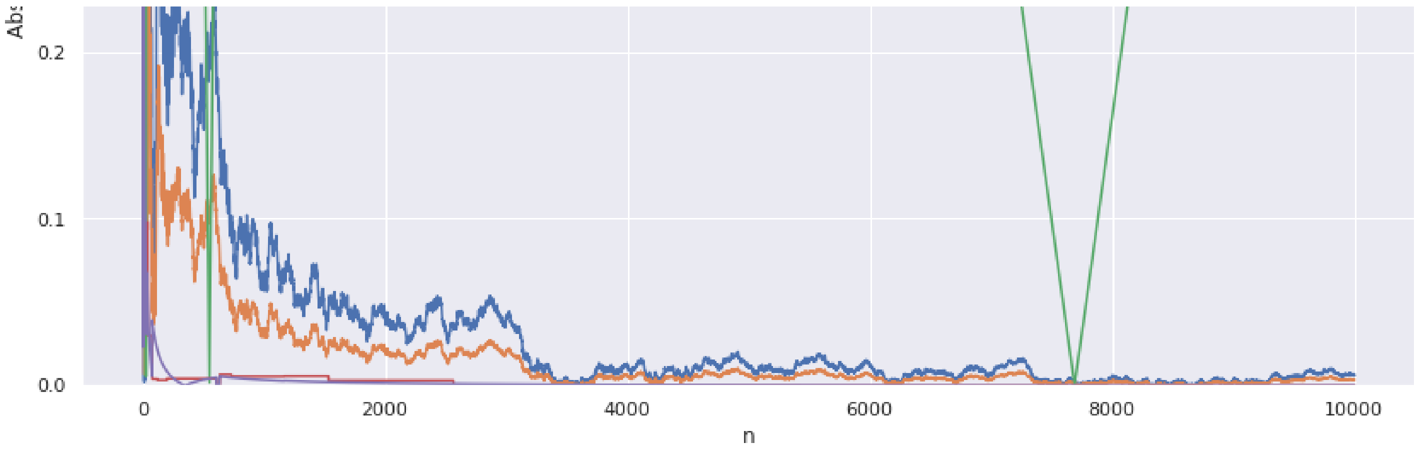



```
uniform dist.  with parameter  3.5
for param = 3.5 best estimate X(1) + X(n) with abs loss 0.00014000090031363044
```




```
uniform dist.  with parameter  4.0
for param = 4.0 best estimate X(1) + X(n) with abs loss 2.0441648359792453e-05
```






```
uniform dist.  with parameter  10.0
for param = 10.0 best estimate (n+1)/n * X(n) with abs loss 0.0009204150719632764
```

Сделайте вывод.

Вывод:

- Из графиков хорошо заметны:
  - Несостоятельность оценки $(n+1)X_{(1)}$ -- график сильно удален от нуля и не стремится к нему
  - Смещенность и состоятельность оценки $\overline{X} + \frac{X_{(n)}}{2}$ (график ошибки выходит на постоянное значение отличное от нуля)
  - Состоятельность и несмещенность $2\overline{X}$, $X_{(1)} + X_{(n)}$ и $\frac{n+1}{n}X_{(n)}$, а также что $\frac{n+1}{n}X_{(n)}$ сходится к параметру θ быстрее всех

Задача 2

Сгенерируйте выборку $X_1, \ldots, X_N$ из экспоненциального распределения с параметром $\theta = 1$ для $N = 10^4$.

In [11]:

```
def CalcExponenEstimates(samples, ks):
    return { "k = {}".format(k) :
             (np.math.factorial(k) / (np.cumsum(samples**k) / np.arange(1, len(samples)+
1)) ) ** (1./k)
             for k in ks}
```

In [12]:

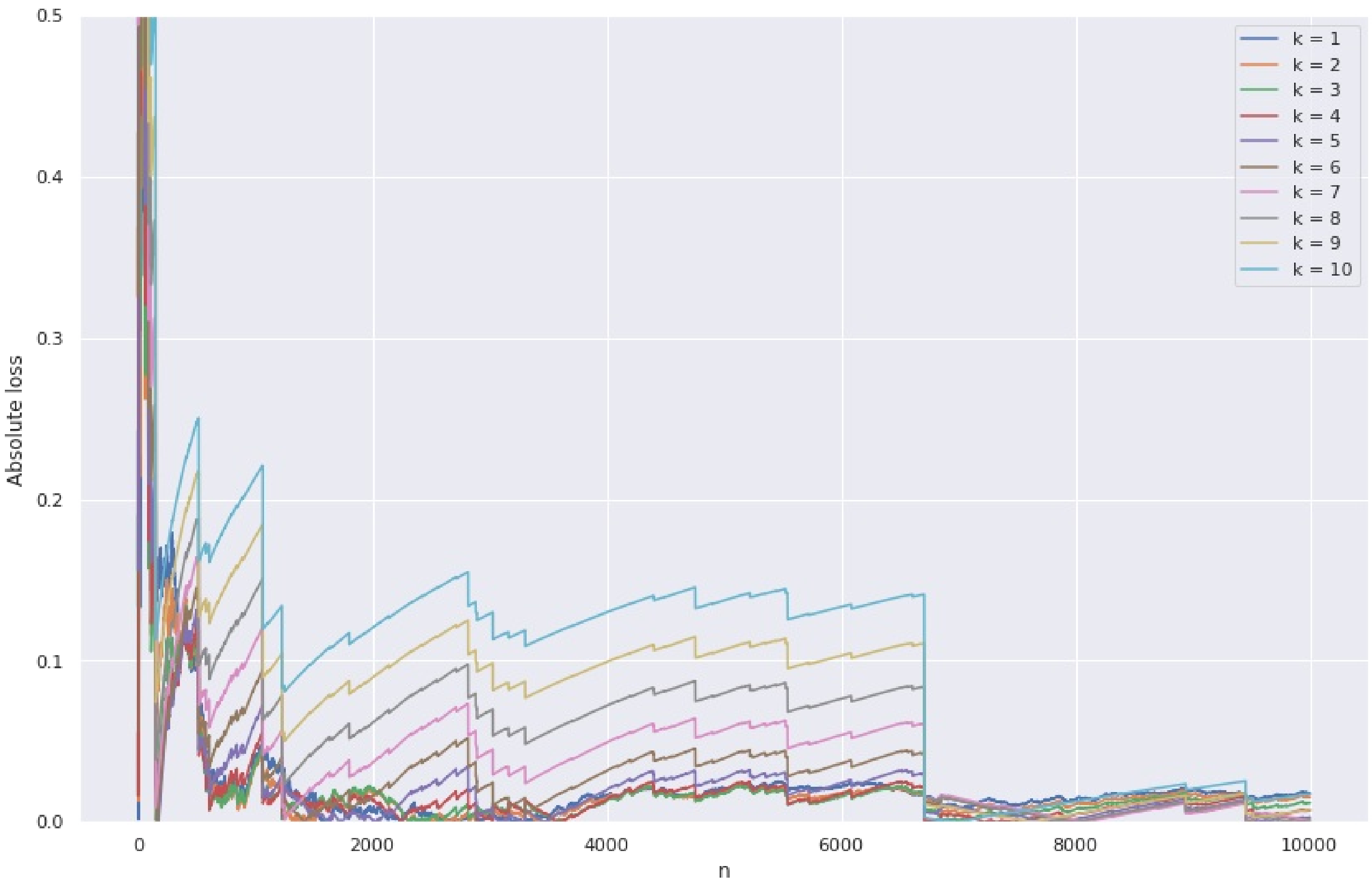
```
# YOUR CODE GOES HERE
N = int(1e4)

ks = np.arange(1, 11)
thetas = np.arange(1, 5.5, 0.5)
samples = np.array([sps.expon.rvs(size=N, scale=1/theta) for theta in thetas])
estimates = np.array([CalcExponenEstimates(sample, ks) for sample in samples])
```

Для всех $n \leqslant N$ посчитайте оценку $(\frac{k!}{\overline{X^k}})^{\frac{1}{k}}$ параметра θ. Проведите исследование, аналогичное предыдущей задаче, и выясните, при каком **k** оценка ведет себя лучше (рассмотрите не менее 10 различных значений k).

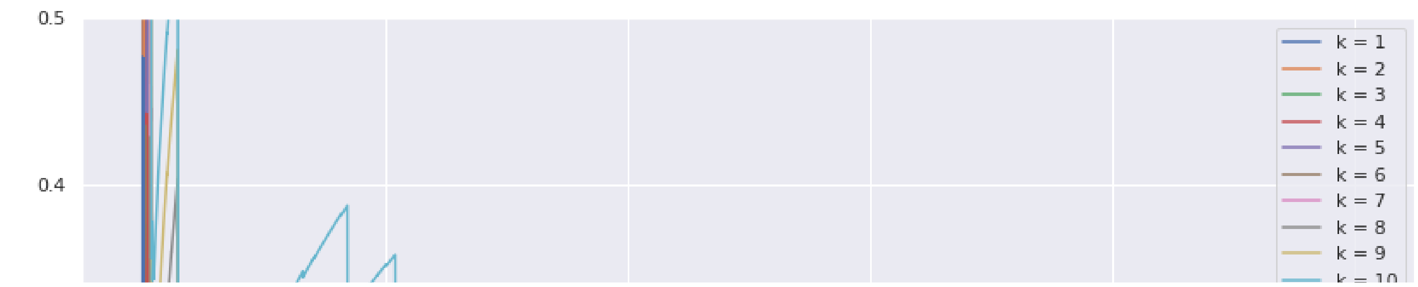
In [13]:

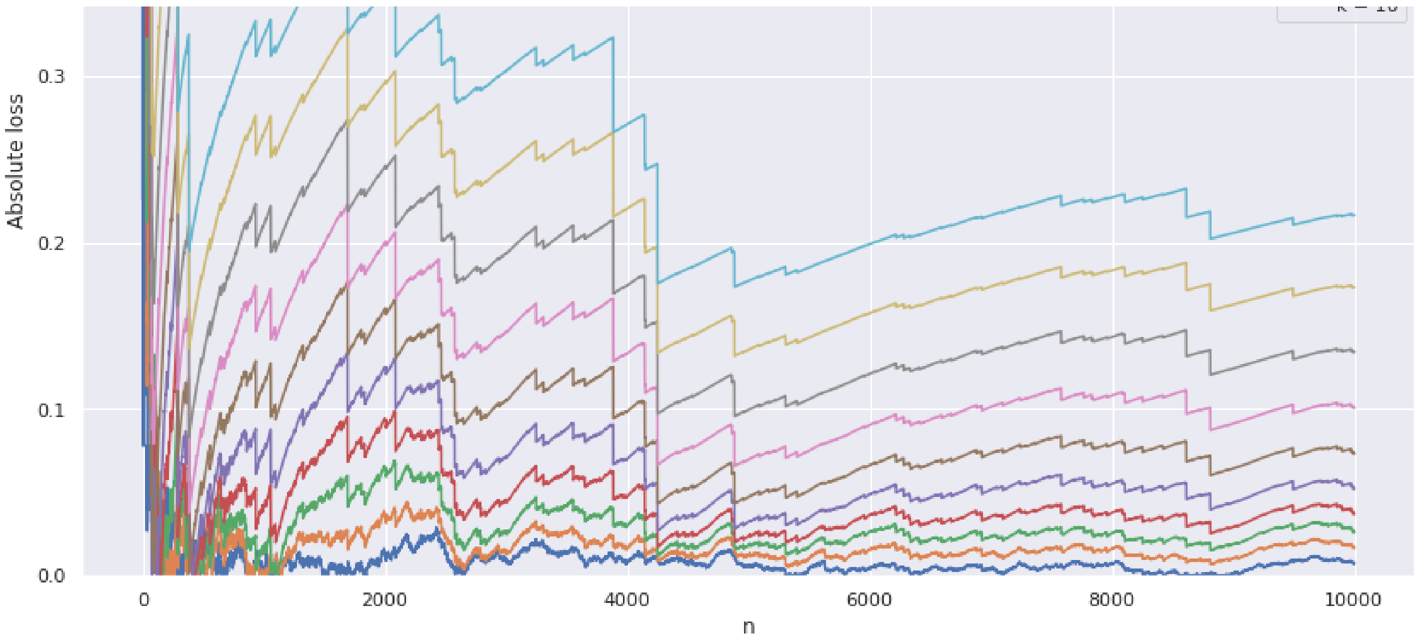
```
# YOUR CODE GOES HERE
RunExperiments(estimates, thetas, "exponential dist.")
```

```
exponential dist.  with parameter  1.0
for param = 1.0 best estimate k = 6 with abs loss 0.00031006935512367 66
```

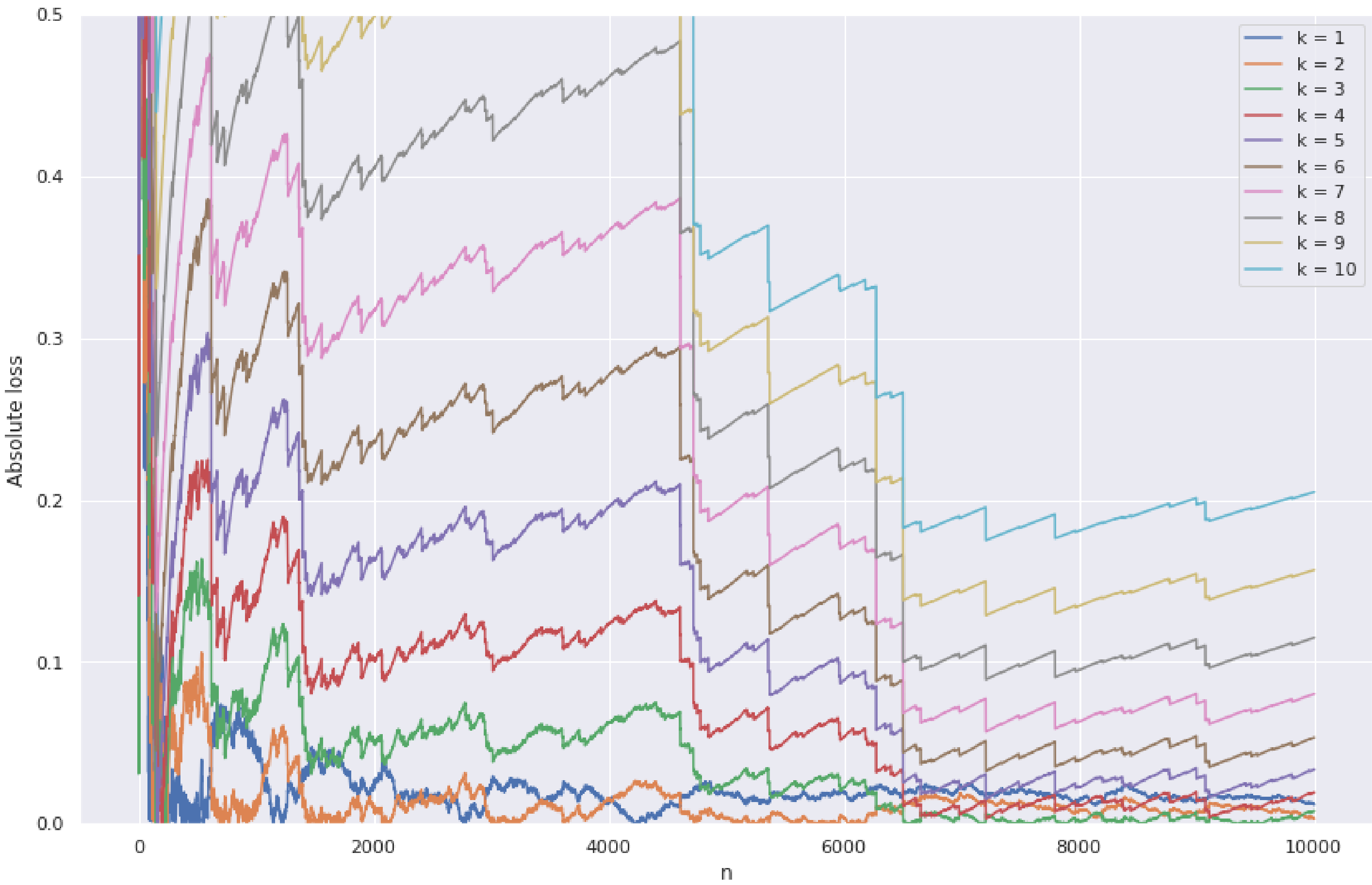



```
exponential dist.  with parameter  1.5
for param = 1.5 best estimate k = 1 with abs loss 0.007746323970645452
```

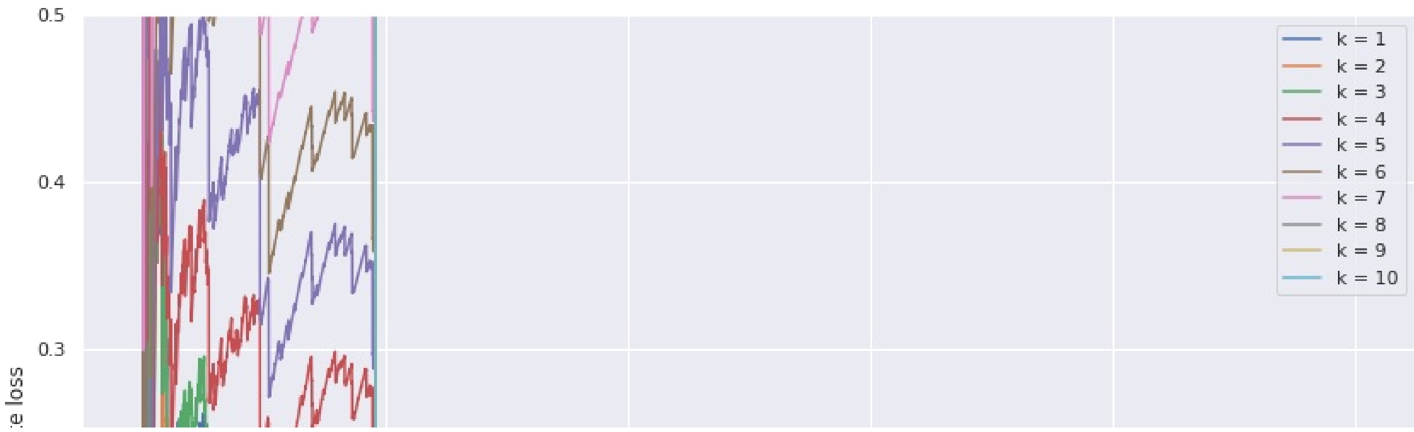


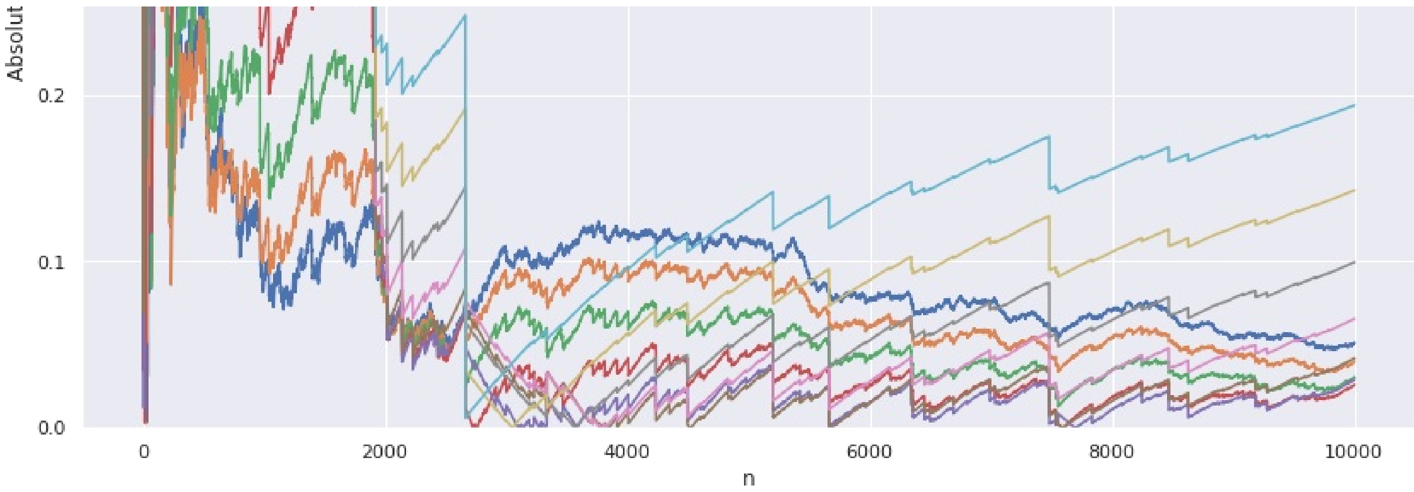



```
exponential dist.  with parameter  2.0
for param = 2.0 best estimate k = 2 with abs loss 0.003084688513363742 5
```

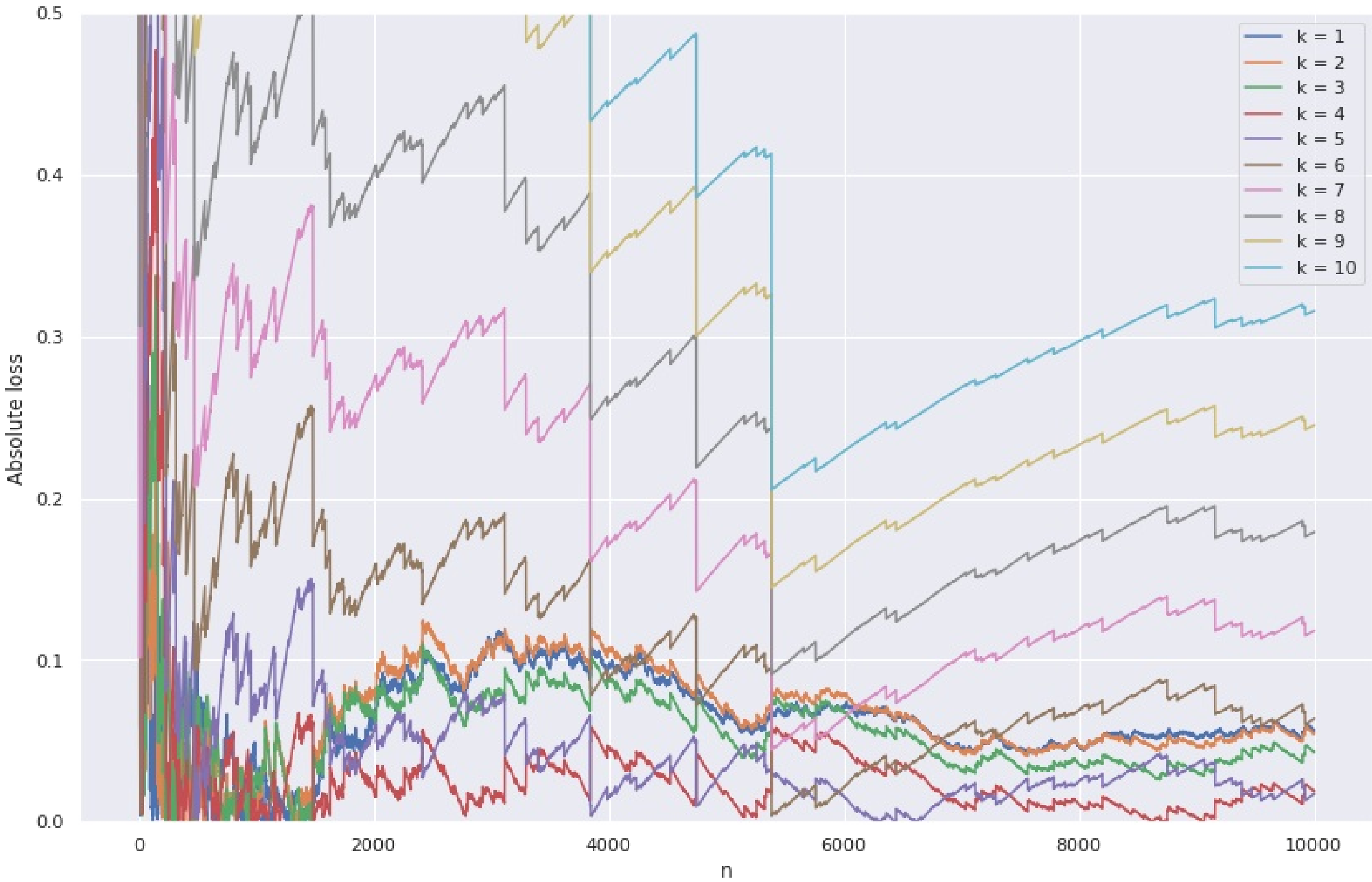



```
exponential dist.  with parameter  2.5
for param = 2.5 best estimate k = 4 with abs loss 0.025971873219029717
```

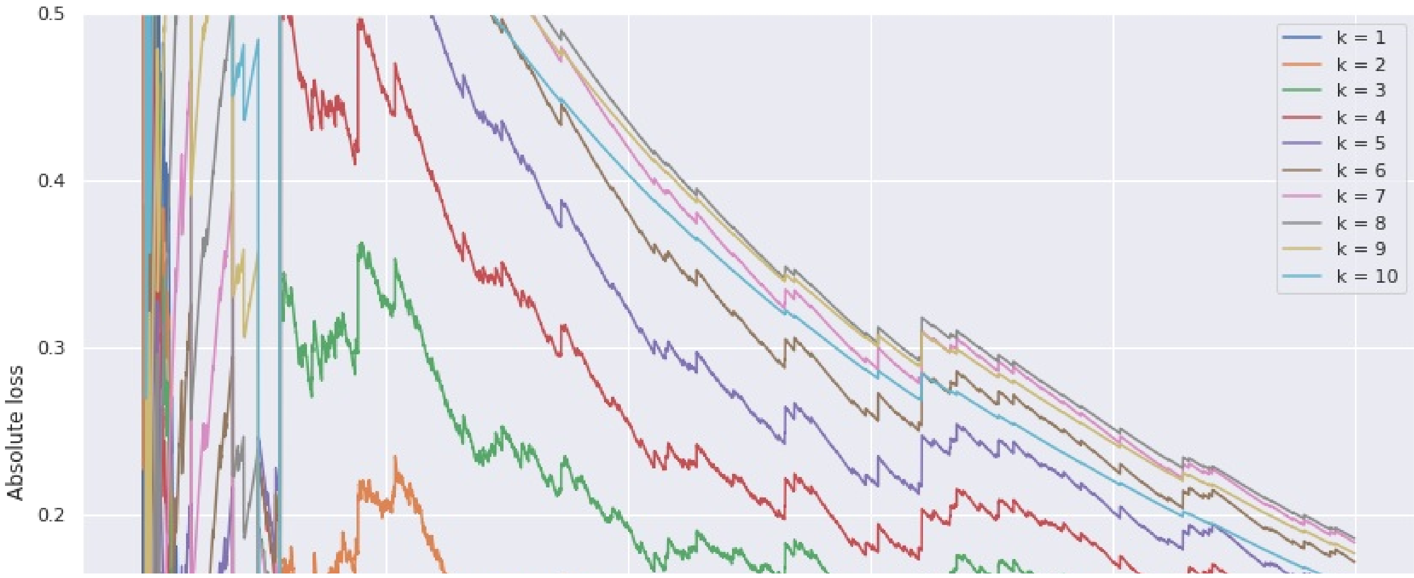


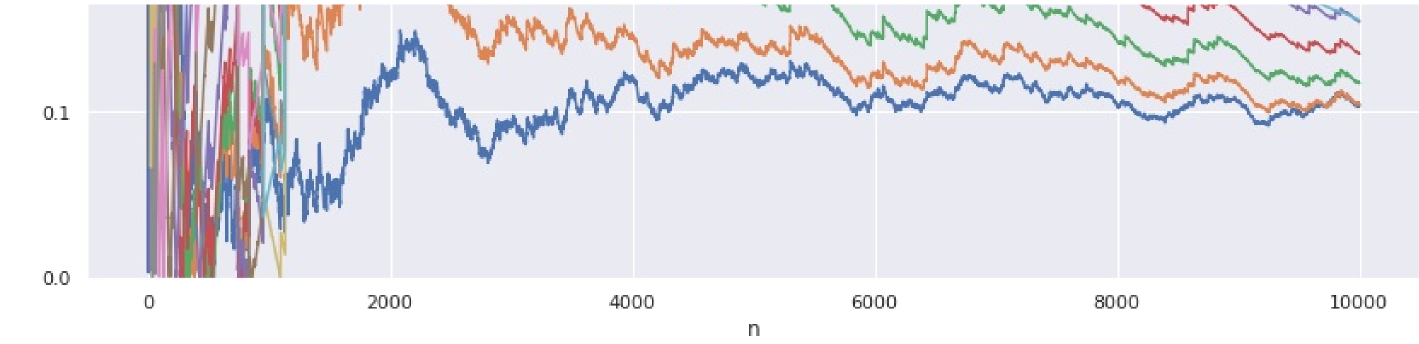



```
exponential dist.  with parameter  3.0
for param = 3.0 best estimate k = 5 with abs loss 0.01735333294563368
```



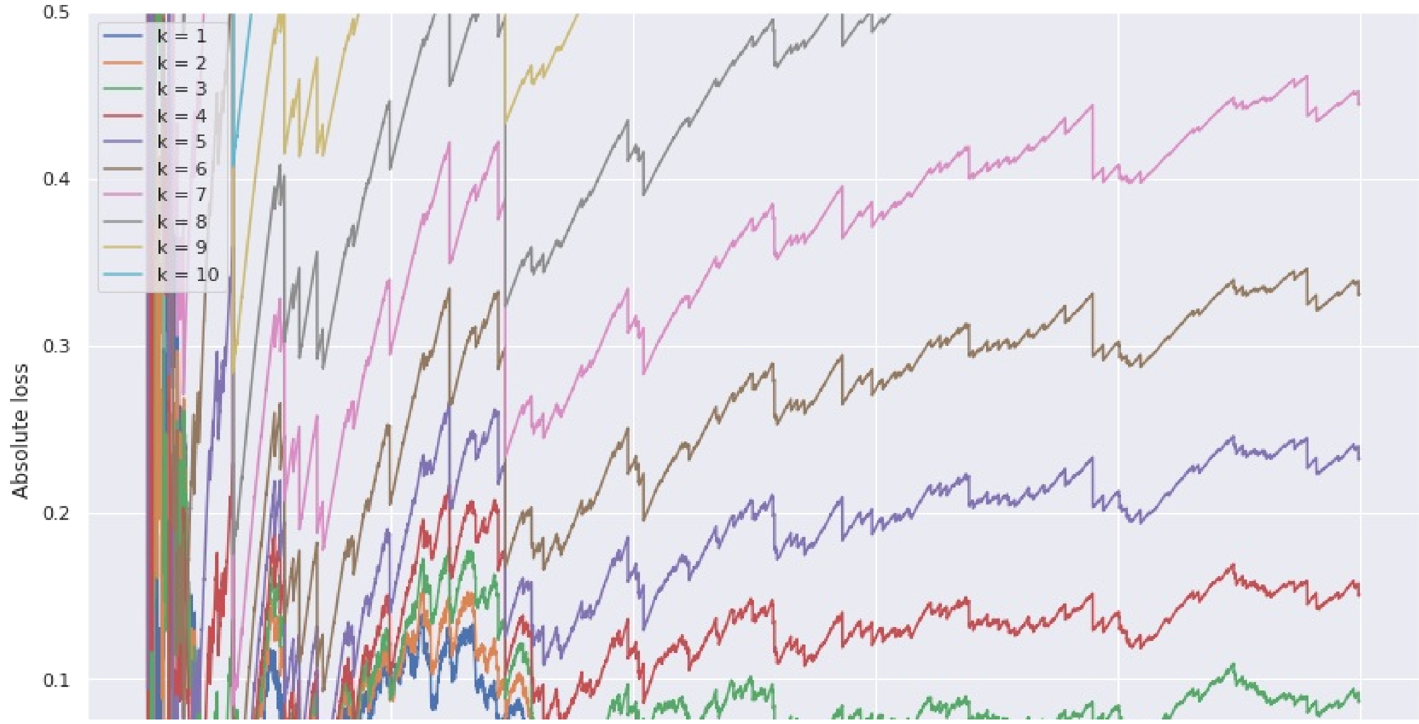

```
exponential dist.  with parameter  3.5
for param = 3.5 best estimate k = 1 with abs loss 0.10361064863536296
```

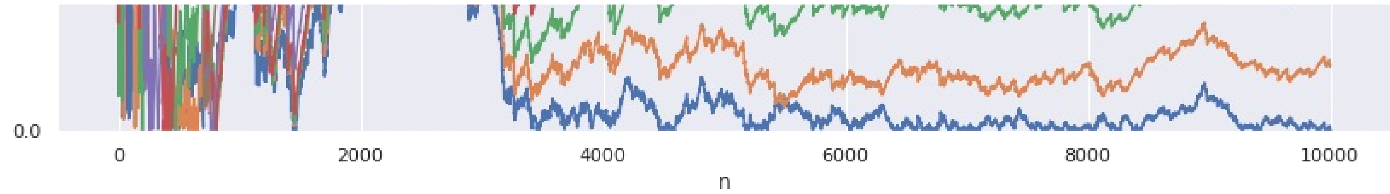





```
exponential dist.  with parameter  4.0
for param = 4.0 best estimate k = 7 with abs loss 0.00818438649539921
```


```
exponential dist.  with parameter  4.5
for param = 4.5 best estimate k = 1 with abs loss 0.0011951078193073172
```






```
exponential dist.  with parameter  5.0
for param = 5.0 best estimate k = 3 with abs loss 0.003609104829286558
```

Сделайте вывод.

Вывод:

- Лучшая оценка соответствует **k=1,** также видно что графики соотв. меньшим **k** лежат ближе к оси абсцисс
- Из теории мы знаем что оценка является ассимптотически нормальной **->** явл-ся состоятельной

Задача 3

Придумайте распределение, у которого конечны первые четыре момента, а пятый - нет. Сгенерируйте выборку $X_1, \ldots, X_N$ из этого распределения для $N = 10^4$.

рассмотрим

$$p(x) = \frac{5}{x^6} \cdot I(x >= 1)$$

действительно: $\int_1^{+\inf} \frac{5}{x^6} dx = 1$ и $E(x^5) > inf$

для генерации выборки воспользуемся методом обратного преобразования для функции распределения $F(x) = 1 - \frac{1}{x^5}$

In [14]:

```
# YOUR CODE GOES HERE
N = int(1e4)
y = sps.uniform.rvs(size=N)
samples = 1 / (1 - y) ** 0.2
```

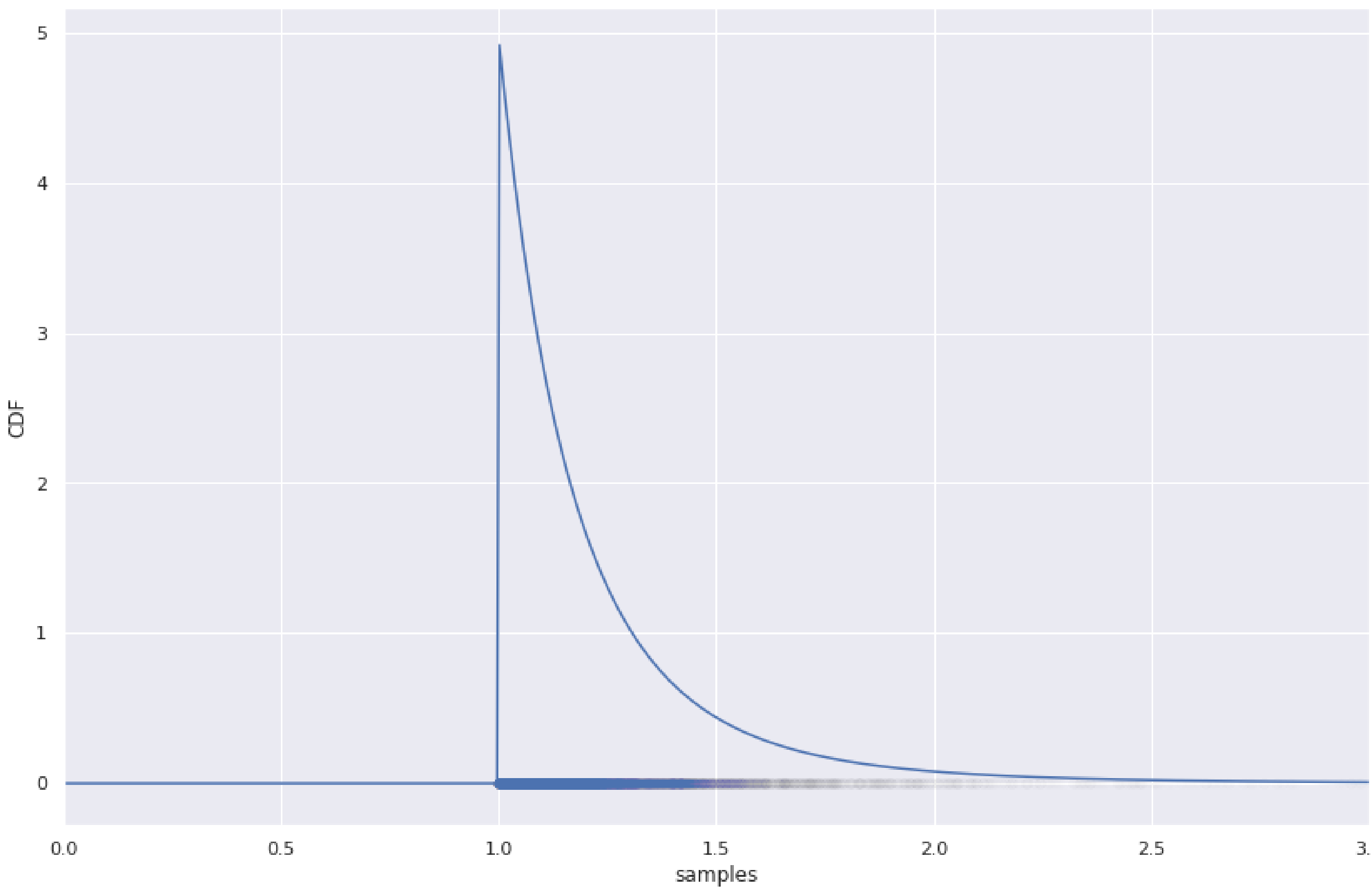
Постройте график плотности, а также нанесите точки выборки на график (с нулевой **y**-координатой)

Подсказка: Может быть полезен параметр `alpha` в функции `plt.plot`

In [15]:

```
# YOUR CODE GOES HERE
linspace = np.linspace(0, samples.max(), 1000)
plt.figure(figsize=(14, 9))
plt.plot(linspace, np.where(linspace >= 1, 5 / (linspace ** 6), 0))
plt.scatter(samples, np.zeros(len(samples)), alpha=0.005, label='Точки выборки')
plt.xlim(0, 3)
plt.xlabel("samples")
plt.ylabel("CDF")

plt.show()
```

```
/home/archie/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4: RuntimeWarnin
g: divide by zero encountered in true_divide
  after removing the cwd from sys.path.
```



Для всех $n \leq N$ посчитайте оценку $s^2 = s^2(X_1, \dots, X_N)$ для дисперсии.

In [16]:

```
# YOUR CODE GOES HERE
ns = np.arange(1, len(samples)+1)
Ssqr_estimate = np.cumsum(samples**2) / ns - (np.cumsum(samples) / ns) ** 2
```

Постройте график зависимости модуля разности оценки дисперсии и ее истинного значения от n.

In [17]:

```
# YOUR CODE GOES HERE
Dx = 5 / 48
PlotAbsLoss({"sample variance loss" : Ssqr_estimate}, Dx, ylim=0.2)
```

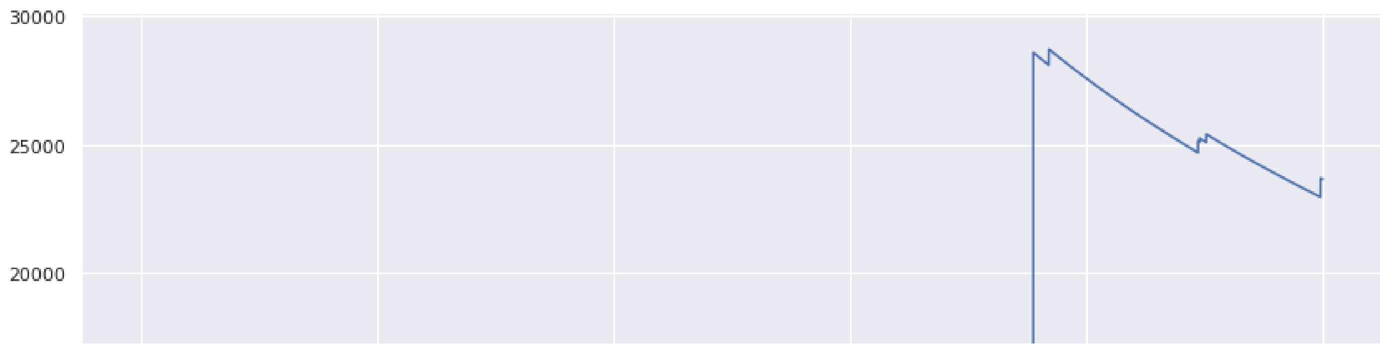
Проведите аналогичное исследование для выборки из распределения Коши, где вместо графика модуля разности оценки дисперсии и ее истинного значения (которого не существует) постройте график оценки дисперсии.

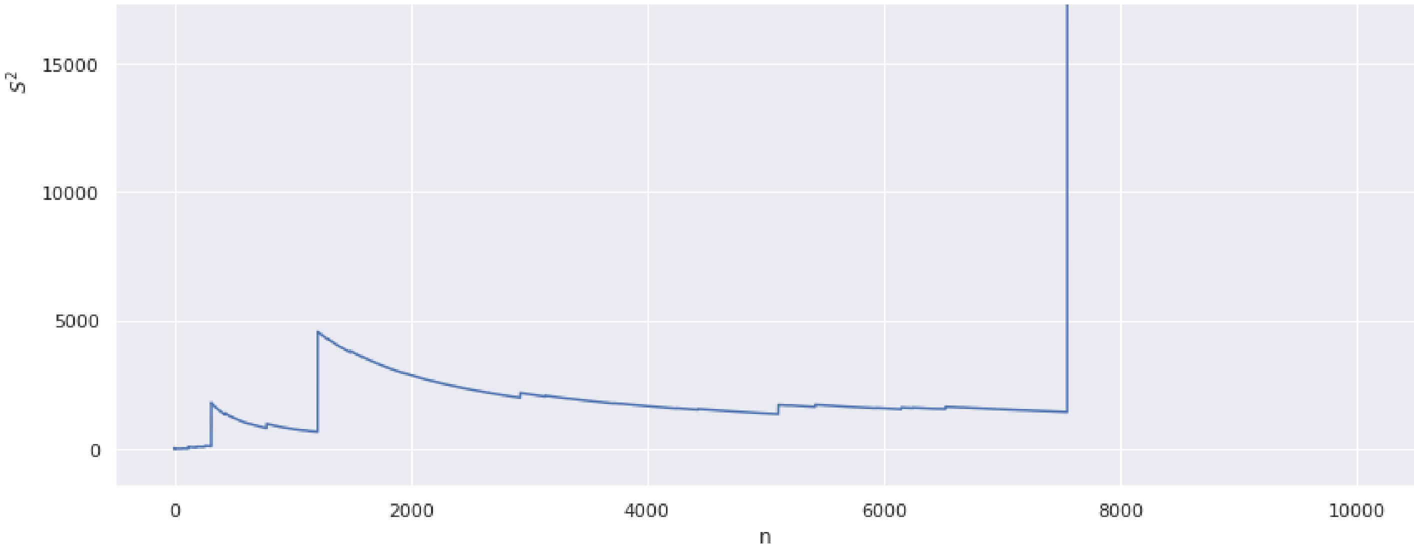
In [18]:

```
# YOUR CODE GOES HERE
N = int(1e4)
samples = sps.cauchy.rvs(size=N)
```

In [19]:

```
ns = np.arange(1, len(samples)+1)
Ssqr_estimate = np.cumsum(samples**2) / ns - (np.cumsum(samples) / ns) ** 2
plt.figure(figsize=(14, 9))
plt.plot(ns, Ssqr_estimate)
plt.ylabel("$S^2$")
plt.xlabel("n")

plt.show()
```





Задача 4

Сгенерируйте выборку $X_1, \ldots, X_N$ из стандартного нормального распределения для $N = 10^4$.

In [20]:

```
# YOUR CODE GOES HERE
N = int(1e4)
samples = sps.norm.rvs(size=N)
```

Для всех $n \leqslant N$ посчитайте по ней эмпирическую функцию распределения.

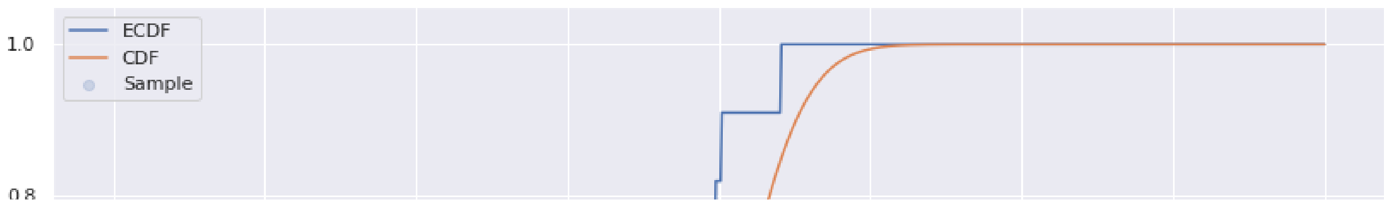
In [21]:

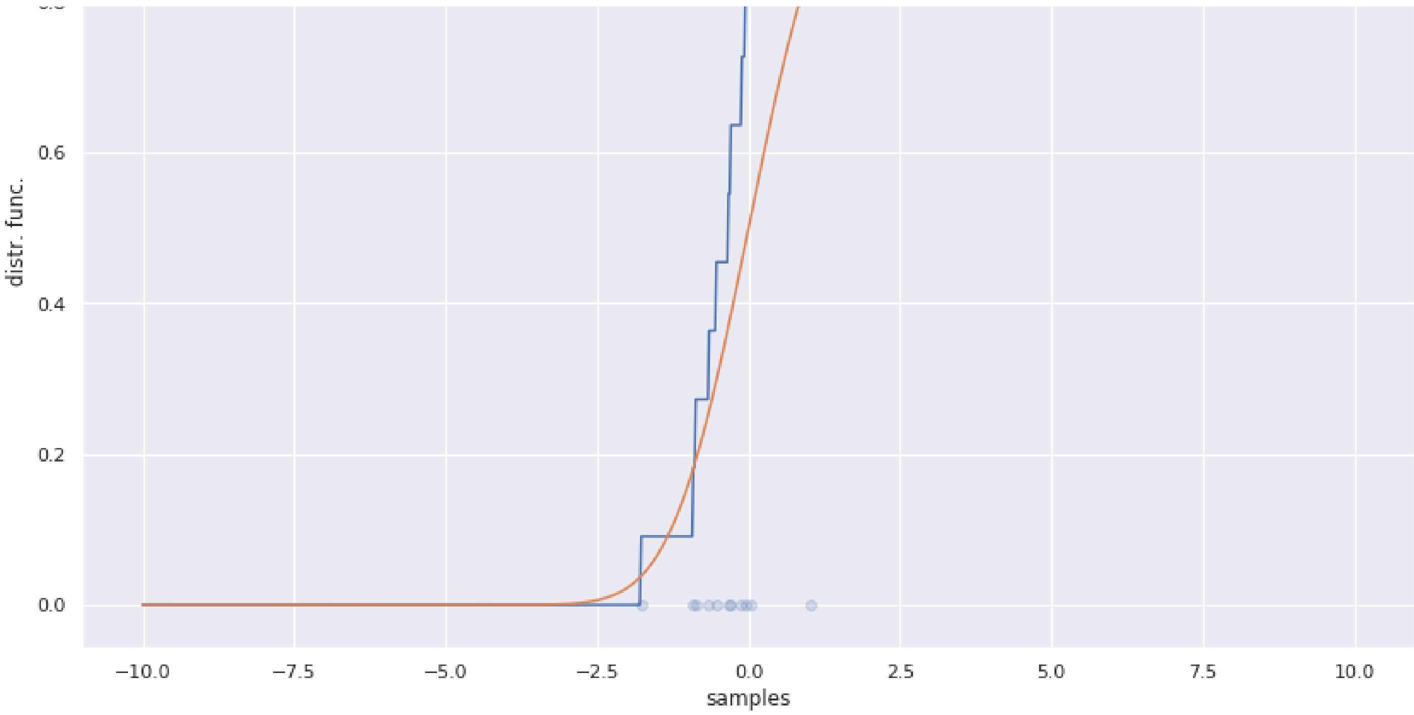
```
from statsmodels.distributions.empirical_distribution import ECDF  # can be useful, but n
ot necessary
ecdf = np.array([ECDF(samples[:i]) for i in range(1, len(samples)+1)])
# YOUR CODE GOES HERE
```

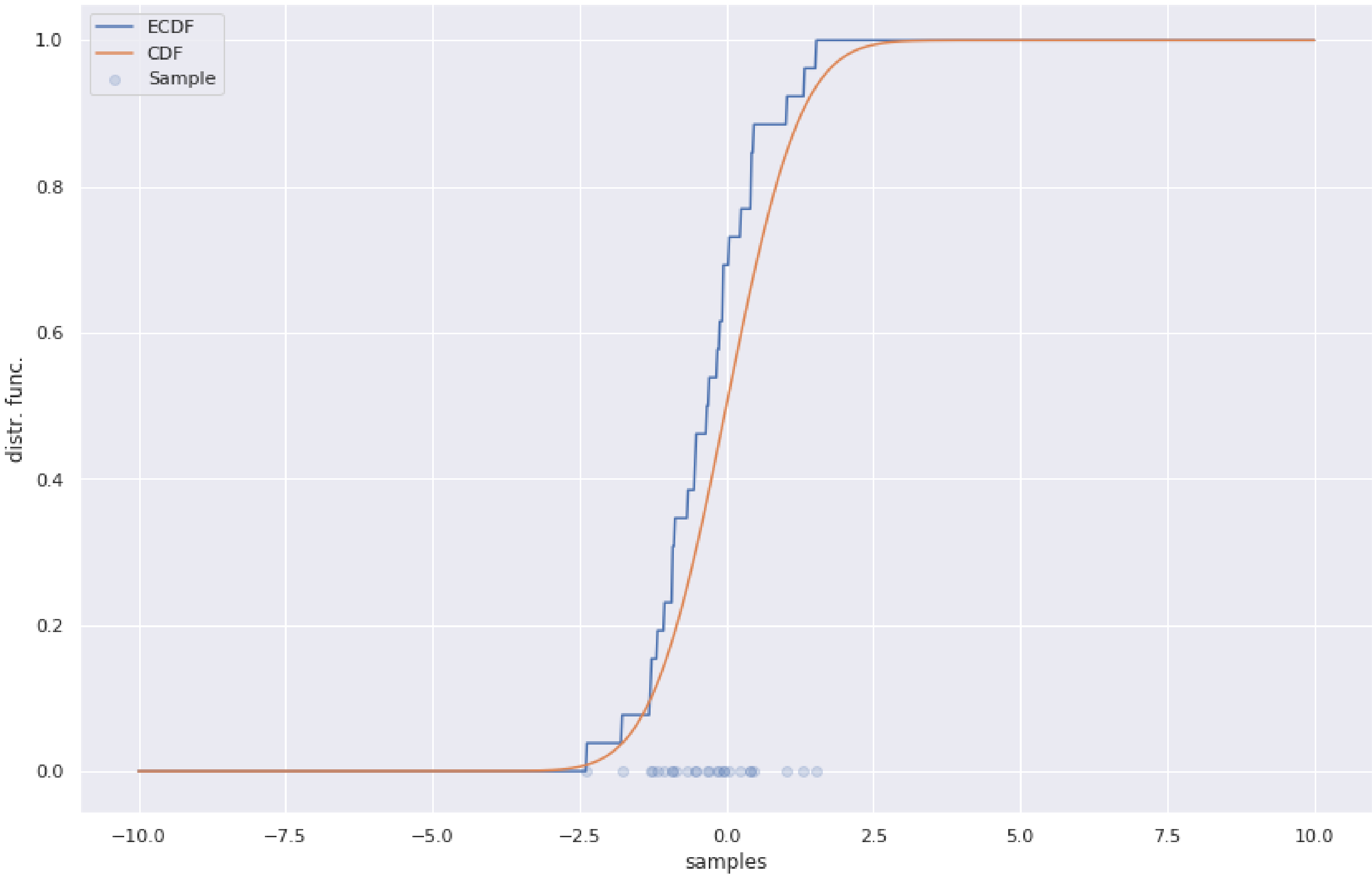
Для некоторых **n** (например, $n \in \{10, 25, 50, 100, 1000, N\}$ постройте графики эмпирической функции распределения (отметьте на оси абсцисс точки "скачков" кривых, нанеся каждую из "подвыборок" на ось абсцисс на каждом соответствующем графике с коэффициентом прозрачности `alpha=0.2`), нанеся на каждый из них истинную функцию распределения (количество графиков равно количеству различныз значений n).

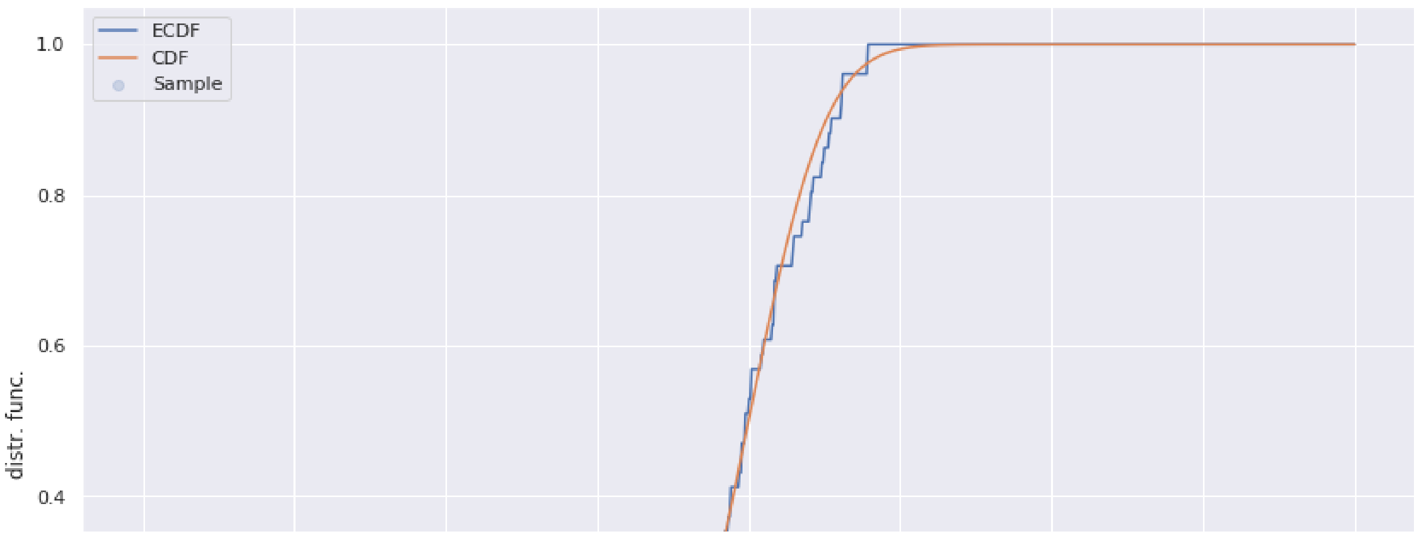
In [22]:

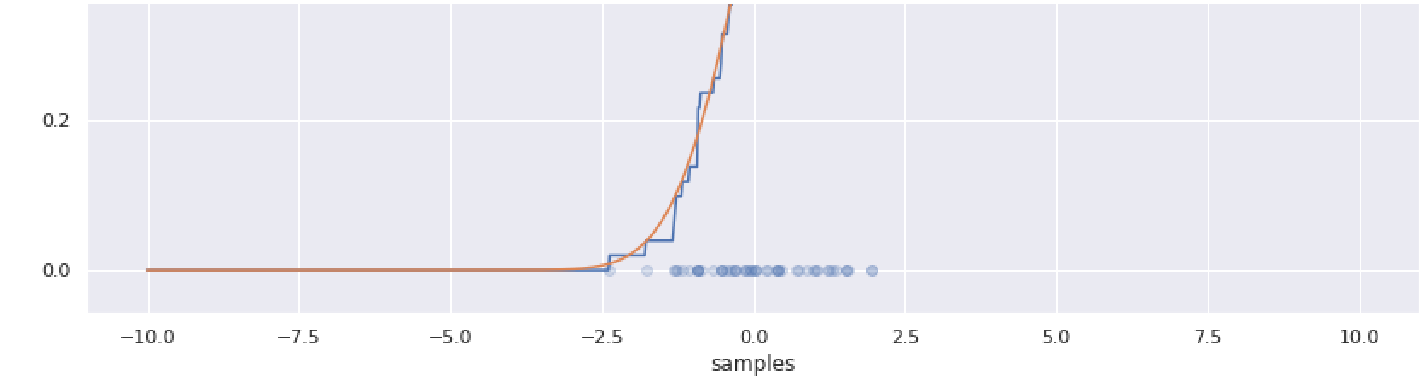
```
# YOUR CODE GOES HERE
ns = np.array([10, 25, 50, 200, 1000])
linspace = np.linspace(-10, 10, 1000)

for n in ns:
    plt.figure(figsize=(14, 9))
    plt.plot(linspace, ecdf[n](linspace))
    plt.plot(linspace, sps.norm.cdf(linspace))
    plt.scatter(samples[:n+1], np.zeros(len(samples[:n+1])), alpha = 0.2)
    plt.legend(["ECDF", "CDF", "Sample"])
    plt.xlabel("samples")
    plt.ylabel("distr. func.")

    plt.show()
```



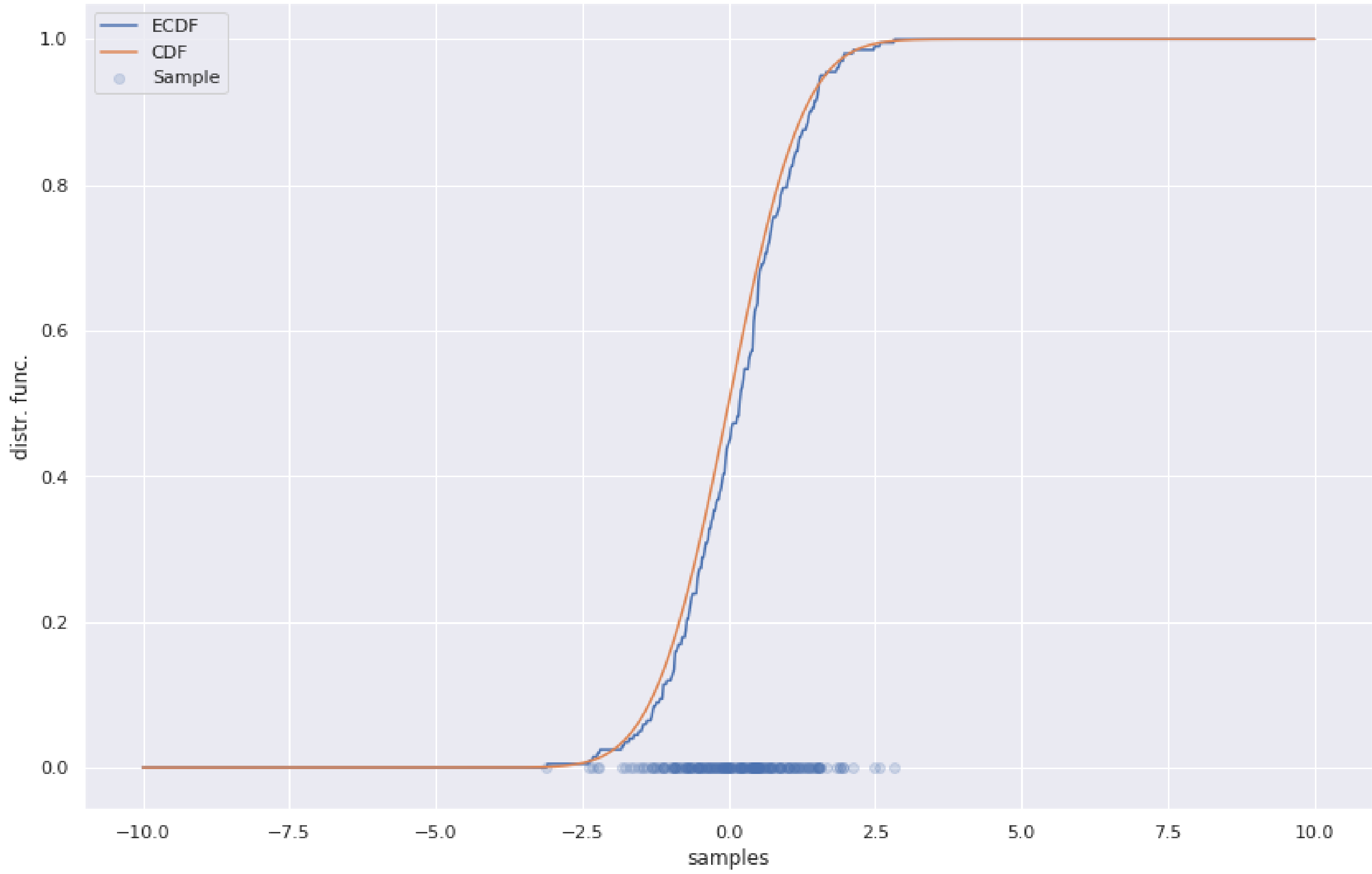

distr. func.
samples
0.0
0.2
0.4
0.6
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


ECDF
CDF
Sample
distr. func.
samples
0.0
0.2
0.4
0.6
0.8
1.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

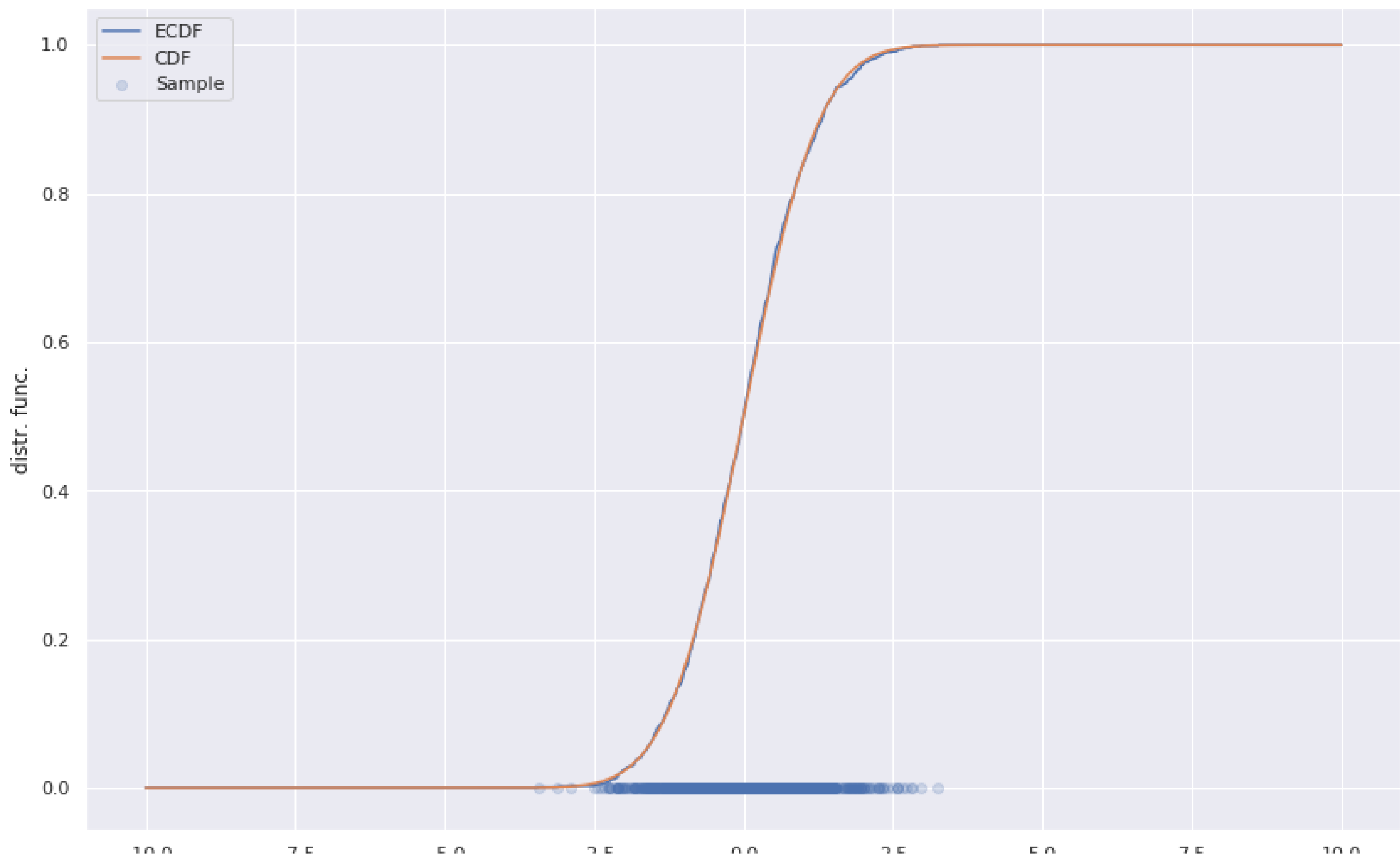

ECDF
CDF
Sample
distr. func.
0.4
0.6
0.8
1.0



0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
samples



ECDF
CDF
Sample
1.0
0.8
0.6
0.4
0.2
0.0
distr. func.
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
samples



ECDF
CDF
Sample
1.0
0.8
0.6
0.4
0.2
0.0
distr. func.

Для всех $n \le N$ посчитайте точное значение $D_n = \sup\limits_{x \in R} |\hat{F}_n(x) - F(x)|$ и постройте график зависимости статистик D_n и $\sqrt{n} D_n$ от n.

In [23]:

```
# YOUR CODE GOES HERE
Dn = np.array([np.max(np.abs(ecdf[n](samples[:n+1]) - sps.norm.cdf(samples[:n+1])))
               for n in range(len(samples))])

plt.figure(figsize=(14, 9))
plt.plot(np.arange(1, len(samples) + 1), Dn)
plt.plot(np.arange(1, len(samples) + 1), np.sqrt(np.arange(1, len(samples) + 1)) * Dn)

plt.legend(["$D_n$", "$\sqrt{n}D_n$"])
plt.xlabel("n")

plt.show()
```

Задача 5

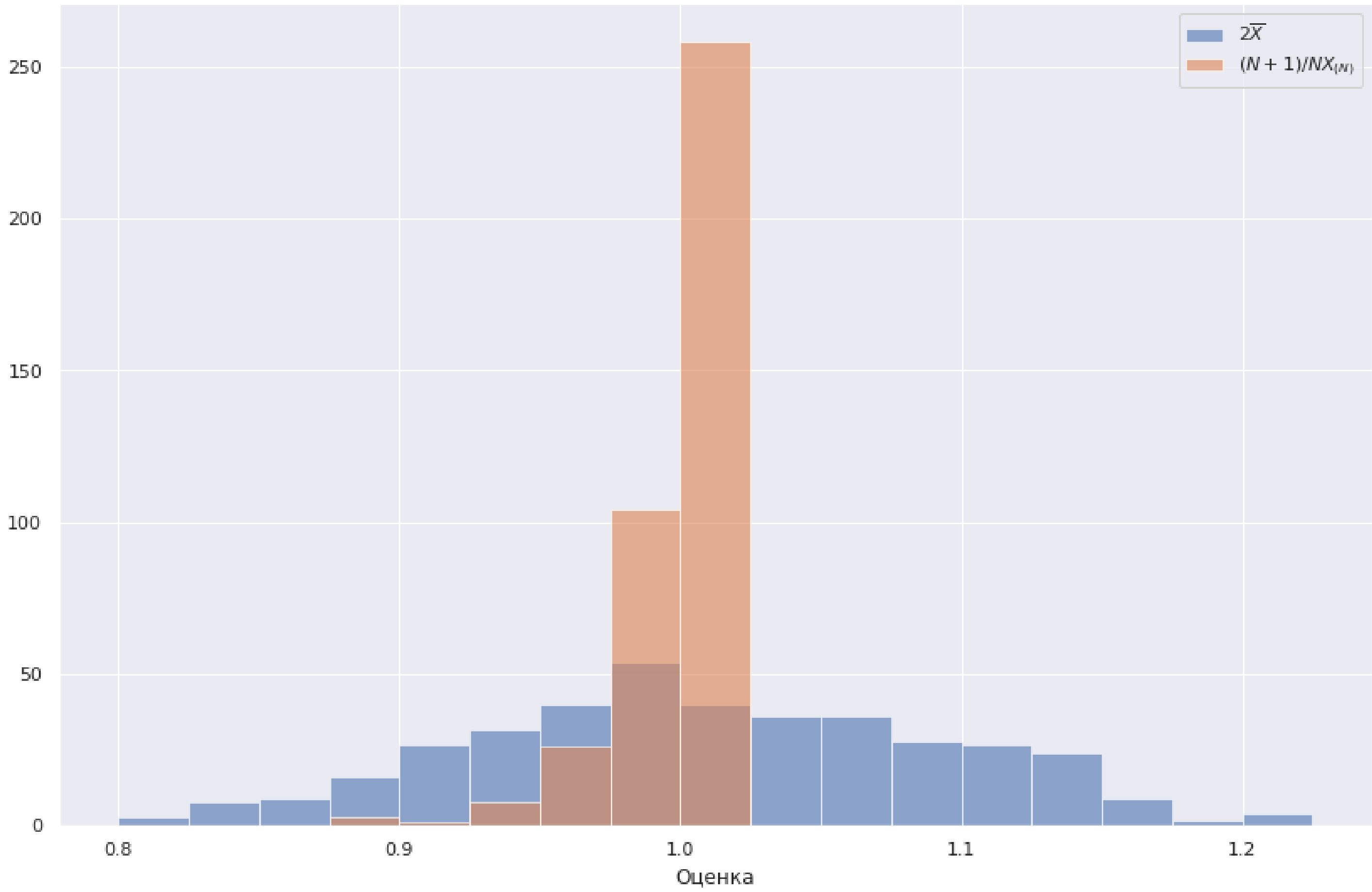
Сгенерируйте $N_{\text{samples}} = 400$ выборок из равномерного распределения $U_{[0,\theta]}$ $\theta = 1$ размера $N = 40$. Для каждой выборки посчитайте статистики $\hat{\theta} = 2\overline{X}$, $\theta^* = \frac{n+1}{n} X_{(n)}$. Постройте гистограмму получившихся значений каждой из статистик на одном графике, в качестве параметра `bins` функции `plt.hist` передайте значение ниже, а также передайте параметр `alpha=0.6`.

In [24]:

```
N_samples = 400
N = 40
theta = 1

samples = np.array([sps.uniform.rvs(loc=0, scale=theta, size=N) for i in range(N_samples
)])
bins = [i / 40 + 0.8 for i in range(18)]

theta_mean = 2 * np.mean(samples, axis=1)
theta_n = (N + 1)/N * np.max(samples, axis=1)
```

In [25]:

```
plt.figure(figsize=(14, 9))
plt.hist(theta_mean, bins=bins, label="$2\overline{X}$", alpha=0.6)
plt.hist(theta_n, bins=bins, label='$(N + 1)/NX_{(N)}$', alpha=0.6
)

plt.xlabel("Оценка")
plt.legend()
plt.show()
```

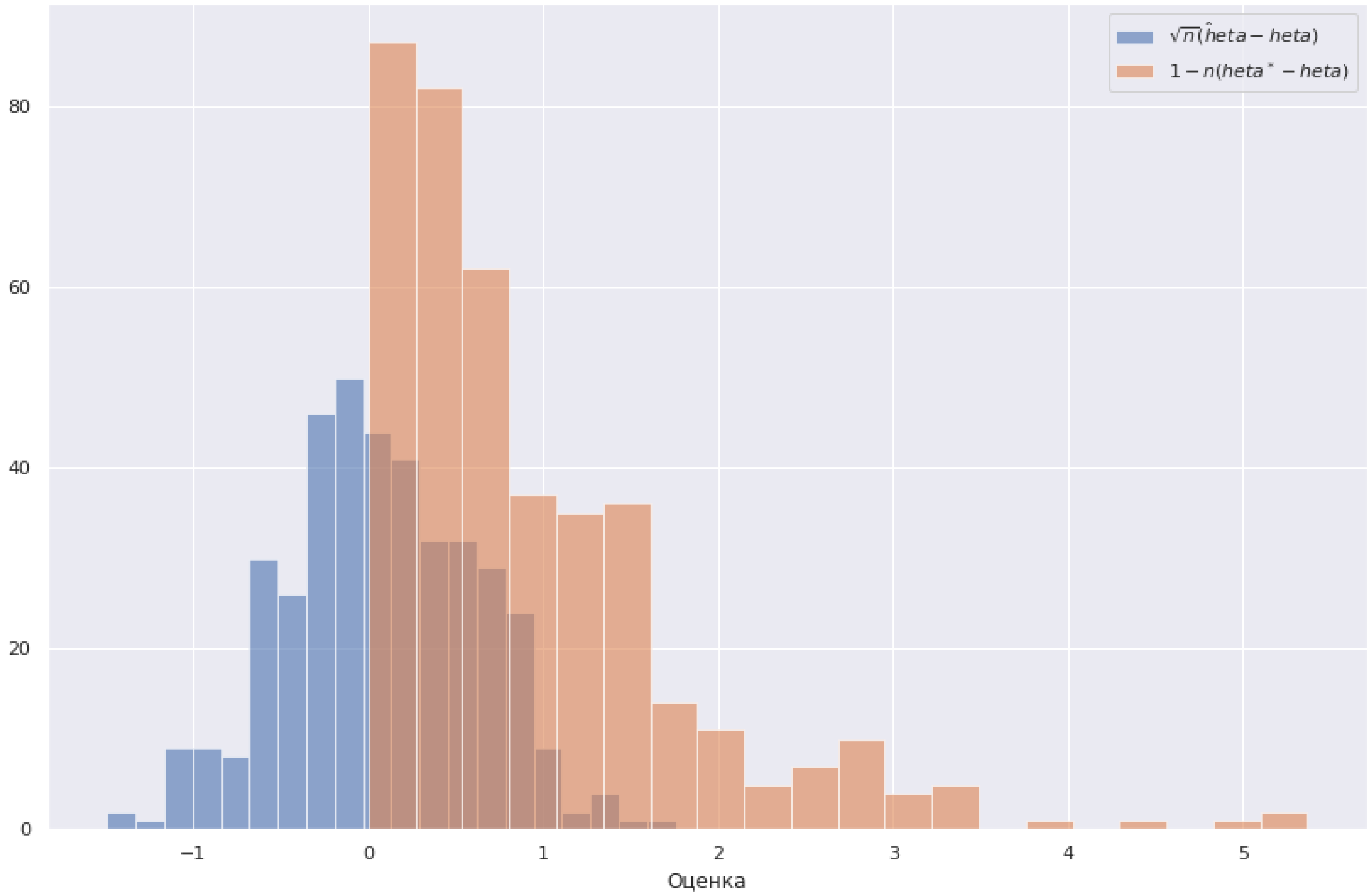


Постройте гистограммы для статистик $\sqrt{n}(\hat\theta - \theta)$ и $1 - n(\theta^* - \theta)$

In [26]:

```
# YOUR CODE GOES HERE
theta=1

plt.figure(figsize=(14, 9))
plt.hist(np.sqrt(N)*(theta_mean - theta), bins=20, label="$\sqrt{n} (\hat\theta - \theta
)$", alpha=0.6)
plt.hist(theta - N*(theta_n - theta), bins=20, label="$1 - n  (\theta^* - \theta)$", alp
ha=0.6)

plt.xlabel("Оценка")
plt.legend()
```

```
plt.show()
```



На какие распределения похожи получившиеся гистограммы?

Ответ:

- $\sqrt{n}(\hat{\theta} - \theta)$ похожа на нормальную (по цпт она будет стремиться к ф-ии нормального распределения)
- экспоненциальное ??

Вспомните чему равен коэффициент $\sigma(\theta)$ для асимптотиически нормальной оценки $\hat{\theta} = 2\overline{X}$ для параметра θ равномерного распределения в формуле

$$\sqrt{n}\frac{\left(\hat{\theta} - \theta\right)}{\sqrt{\sigma(\theta)}}$$

$$\xrightarrow{d} N(0,1)$$

.

Ответ: Из ЦПТ следует, что $\sigma(\theta) = 2\sigma_{U[0,\theta]}$, где $\sigma^2_{U[0,\theta]} = \frac{\theta^2}{12}$

$$\sigma(\theta) = \frac{\theta}{\sqrt{3}}$$

Посчитайте значения статистики

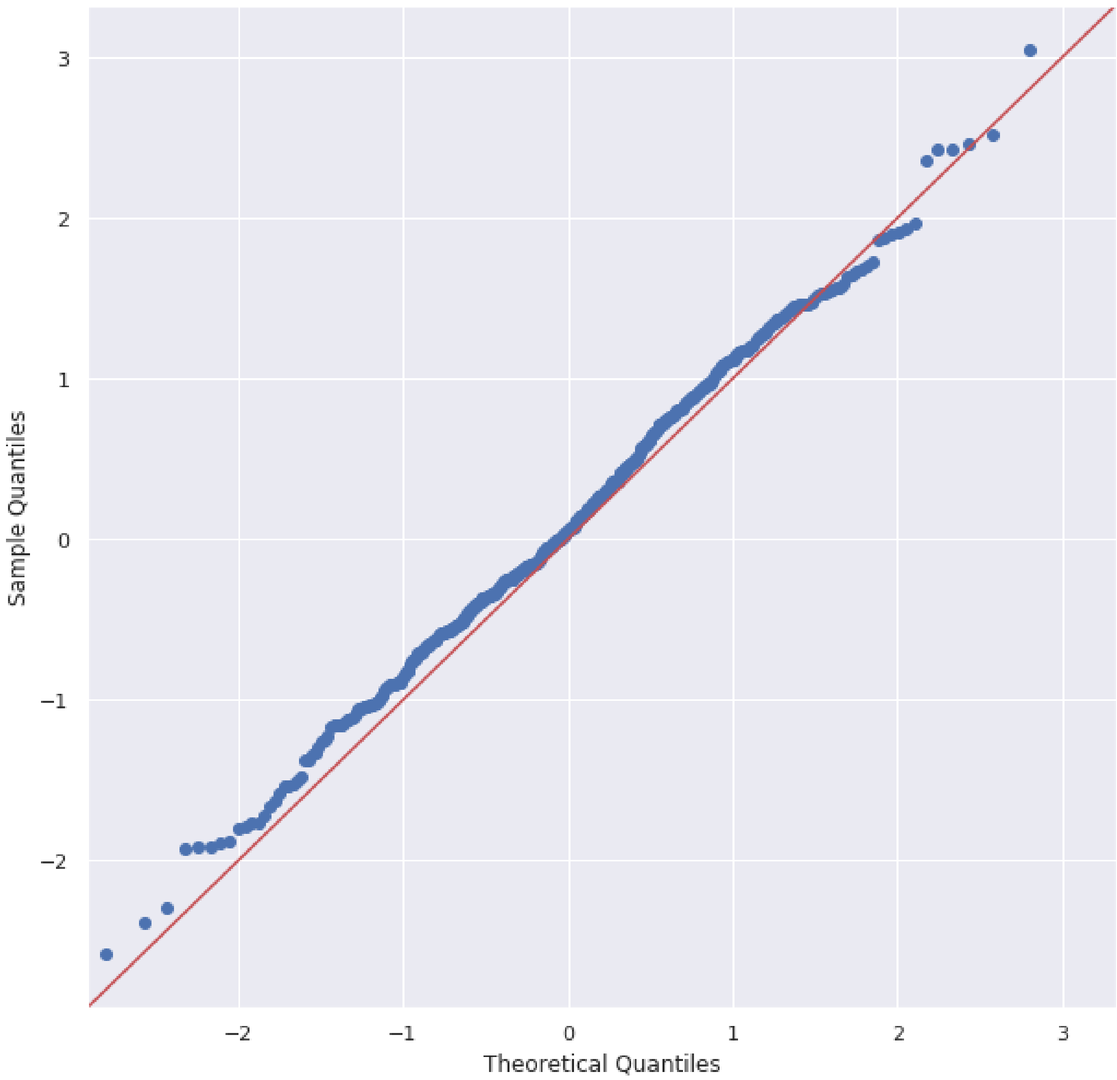
$$\sqrt{n}\frac{\left(\hat{\theta} - \theta\right)}{\sqrt{\sigma(\theta)}}$$

для каждой выборки. Передайте получившиеся значения в переменную `theta_norm`. И запустите ячейку снизу.

In [27]:

```
import statsmodels.api as sm
```

```
theta_norm = np.sqrt(N)*(theta_mean - theta) * np.sqrt(3)  # YOUR CODE GOES HERE
fig, ax = plt.subplots(figsize=(10, 10))
sm.qqplot(theta_norm, line='45', ax=ax)
plt.show()
```



Для быстрой проверки гипотезы о том, что выборка принадлежит какому-либо распределению часто используется инструмент под названием `QQ-plot` (первые буквы означают `Quantile`). На нем по оси x отложены теоретические значения квантиля, а по оси y -- квантили тестируемой выборки. Очевидно, в идеале такие квантили должны совпадать, поэтому на графике можно увидеть красную линию соответствующую графику функции $y = x$.

Сделайте вывод по графику выше. Можно ли утверждать, что выборка взята из нормального распределения?

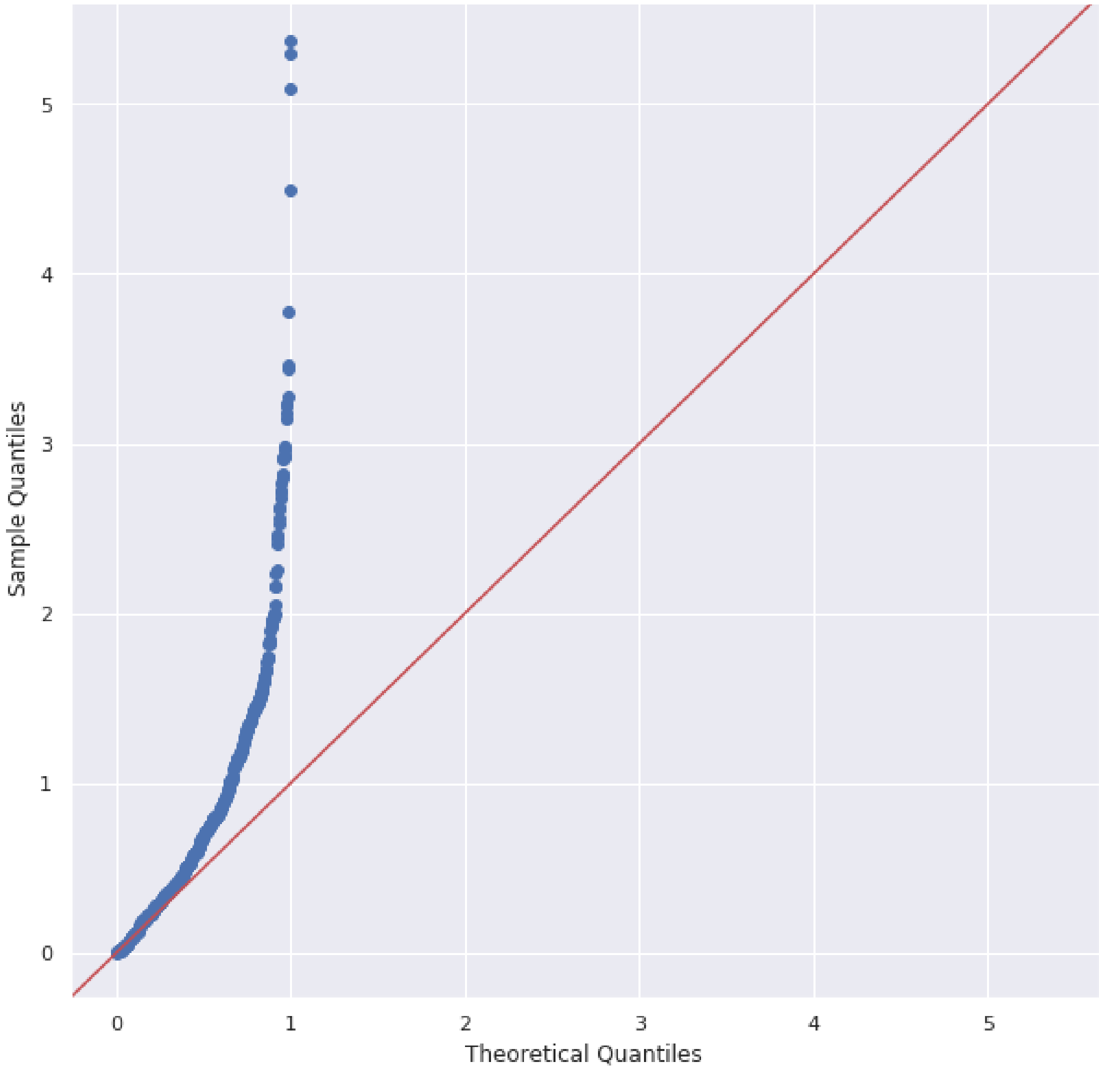
Ответ: Да, тк большая часть лежит на прямой и рядом с ней

Вернемся к статистике $\theta - n(\theta^* - \theta)$. Еще раз взгляните на гистограмму, соотвутствующую этой статистике. Попробуйте построить `QQ-plot` для различных распределений (например можно передать в параметр `dist=sps.uniform` в функцию `sm.qqplot` или любое другое из модуля `scipy.stats`). Какое распределение подходит лучше всего?

Ответ: экспоненциальное (см. ниже)

In [28]:

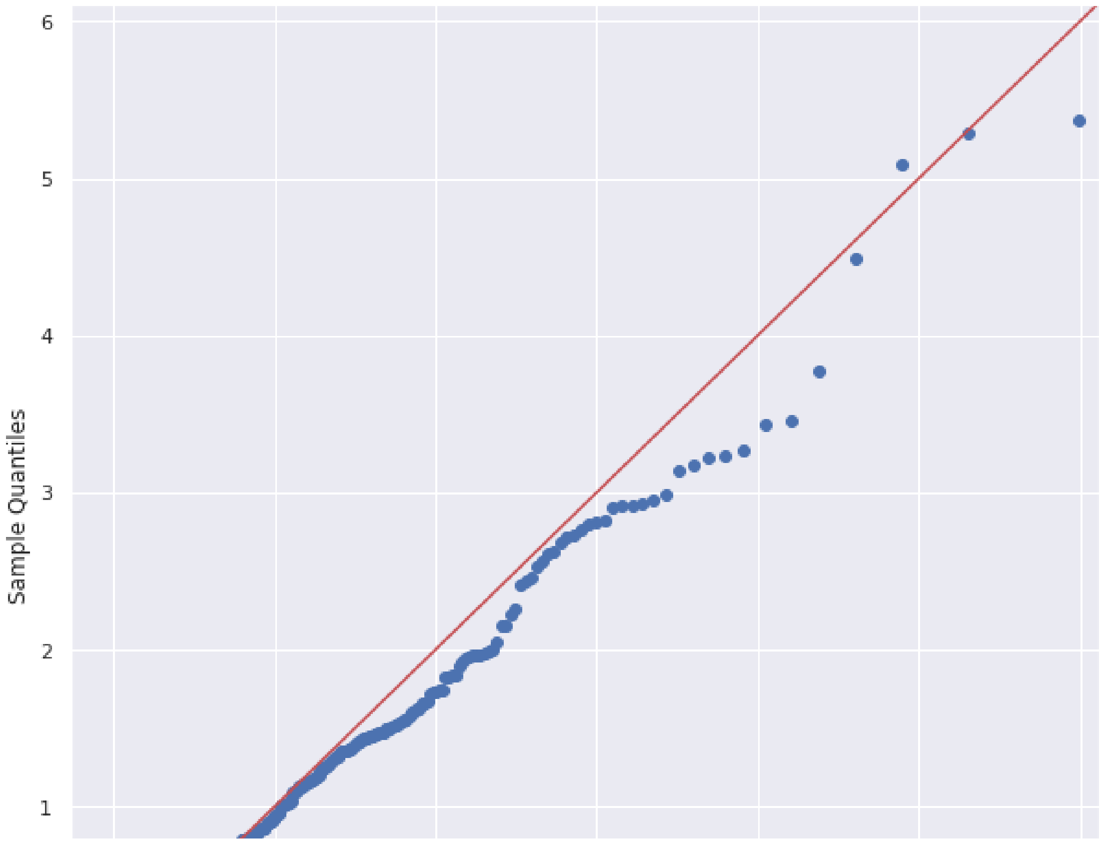
```
# YOUR CODE GOES HERE
stat=theta - N*(theta_n - theta)

fig, ax = plt.subplots(figsize=(10, 10))
sm.qqplot(stat, dist=sps.uniform, line='45', ax=ax)
plt.show()
```

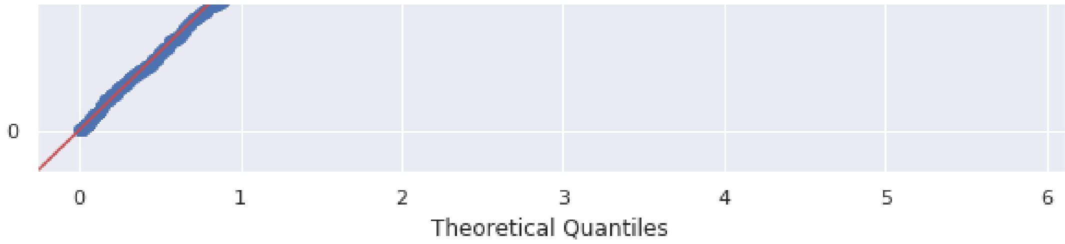


Попробуем экспоненциальное распределение, потому что диаграма похожа на него

In [29]:

```
fig, ax = plt.subplots(figsize=(10, 10))
sm.qqplot(stat, line='45', ax=ax, dist=sps.expon)
plt.show()
```





Плохо ли, что оценка θ^* не асимптотически нормальна? Сделайте вывод о скорости сходимости оценок. Какая из них «выгоднее»?

Ответ: несмотря на то, что θ^* не асимптотически нормальна, $\theta - n(\theta^* - \theta)$ можно хорошо описать экспоненциальным распределением (исходя из **QQ-plot**'а) с конечным мат ожиданием, и тогда скорость сходимости этой оценки будет примерно **n** (из ЗБЧ). Когда у $\sqrt{n}(\hat{\theta} - \theta)$ скорость сходимости примерно $\sqrt{n}$ (из ЦПТ)

т.е. первая выгоднее

Перед отправкой нажмите `Restart and run all`. Проверьте, что все работает без ошибок.