Кафедра дискретной математики МФТИ

Курс математической статистики

Игашов Илья, 593 группа

Задание №4

№1. (К теоретическрй задаче 1)

Сгенерируйте $M = 100$ выборок $X_1,\dots, X_{1000}$ из равномерного распределения на отрезке $[0, \theta]$ (возьмите три произвольных положительных значения θ). Для каждой выборки $X_1,\dots, X_n$ для всех $n\leq 1000$ посчитайте оценки параметра θ из теоретической задачи: $2\bar{X},(n + 1)X_{(1)}, X_{(1)} + X_{(n)}, \frac{n+1}{n}X_{(n)}$. Посчитайте для всех полученых оценок $\hat\theta$ квадратичную функцию потерь $(\hat\theta - \theta)^2$ и для каждого фиксированного n усредните по выборкам. Для каждого из трех значений θ постройте графики усредненных функций потерь в зависимости от n.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import uniform

%matplotlib inline
```

In [2]:
```
M = 100
N = 1000
```

$\theta = 1$

In [3]:
```
# Сгенерируем выборки для theta = 1.
theta = 1
samples = np.array([uniform.rvs(size=N, scale=theta) for m in range(M)])
```

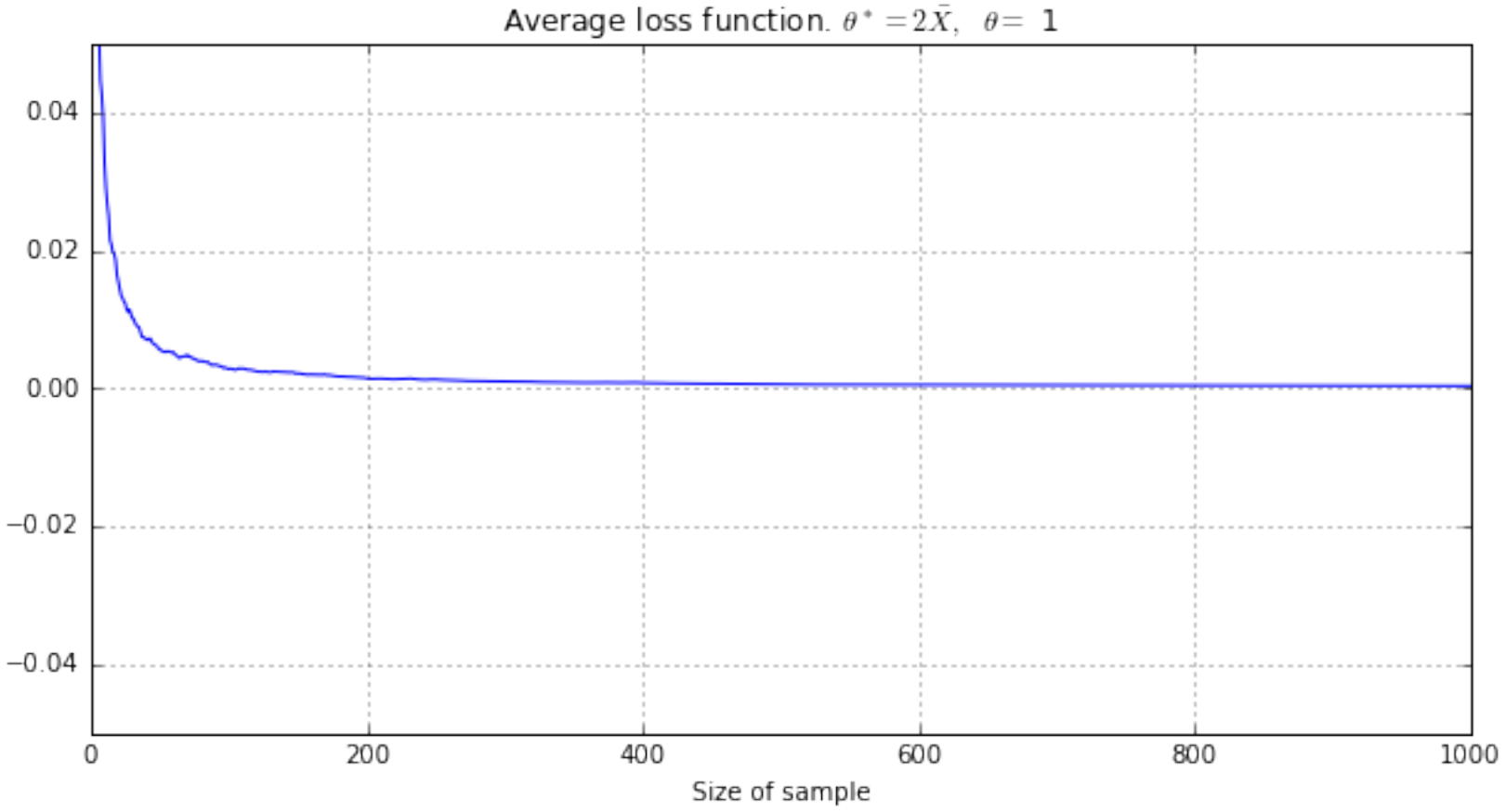
In [4]:
```
# Посчитаем оценки для каждой выборки для каждого n = 1,...,N.
sample_mean_1 = np.array([samples[m].cumsum() * [2 / n
                                                 for n in range(1, N + 1, 1)]
for m in range(M)])
koeff_min_1 = np.array([[(n + 1) * np.min(samples[m][ : n])
                         for n in range(1, N + 1, 1)] for m in range(M)])
min_max_1 = np.array([[np.min(samples[m][ : n + 1]) + np.max(samples[m][ : n
+ 1])
                       for n in range(N)] for m in range(M)])
koeff_max_1 = np.array([[(n + 1) / n * np.max(samples[m][ : n])
                         for n in range(1, N + 1, 1)] for m in range(M)])
```

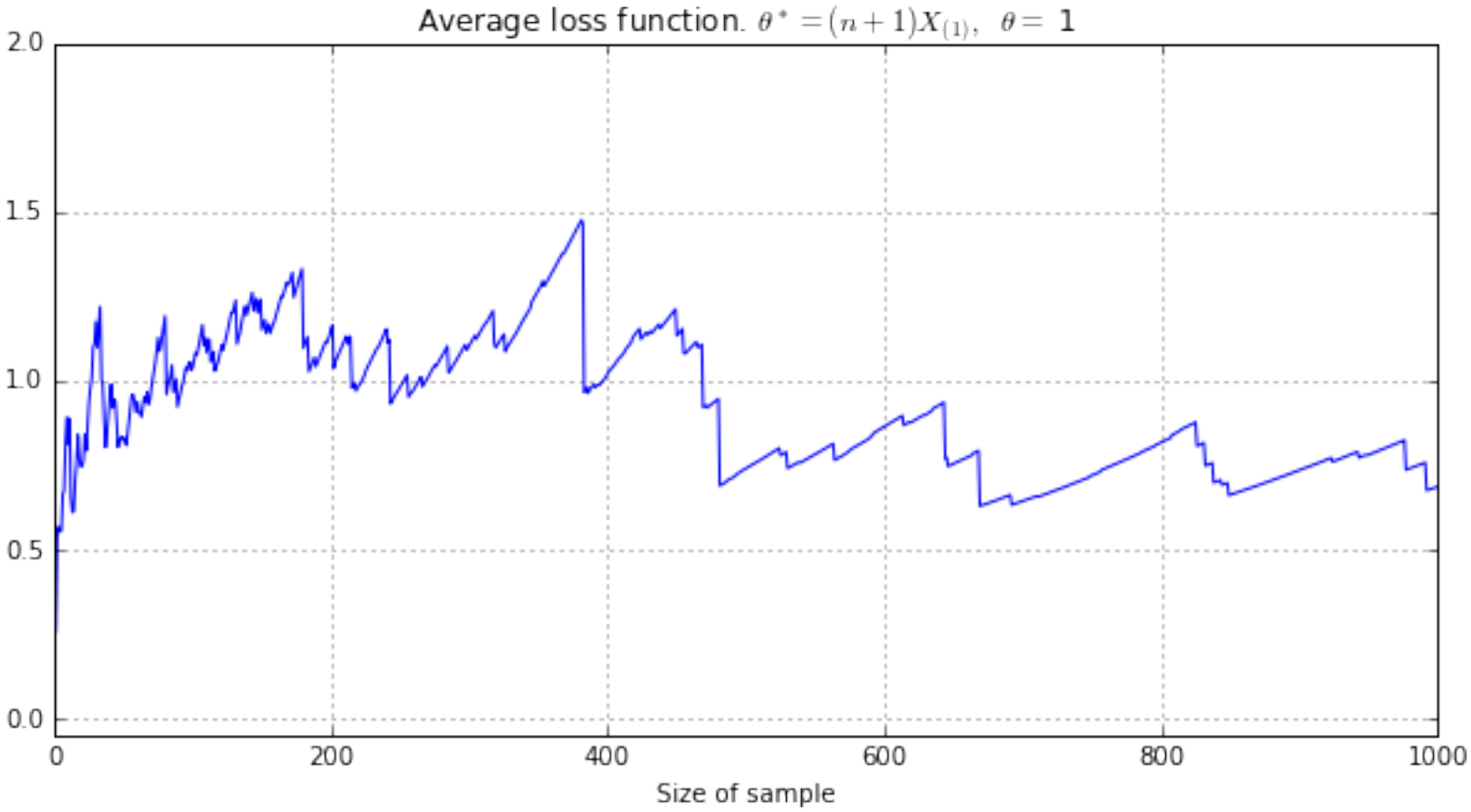
In [5]:
```
# Посчитаем квадратичную функцию потерь для каждого значения оценок.
lf_sample_mean_1 = (sample_mean_1 - theta) ** 2
lf_koeff_min_1 = (koeff_min_1 - theta) ** 2
lf_min_max_1 = (min_max_1 - theta) ** 2
lf_koeff_max_1 = (koeff_max_1 - theta) ** 2
```

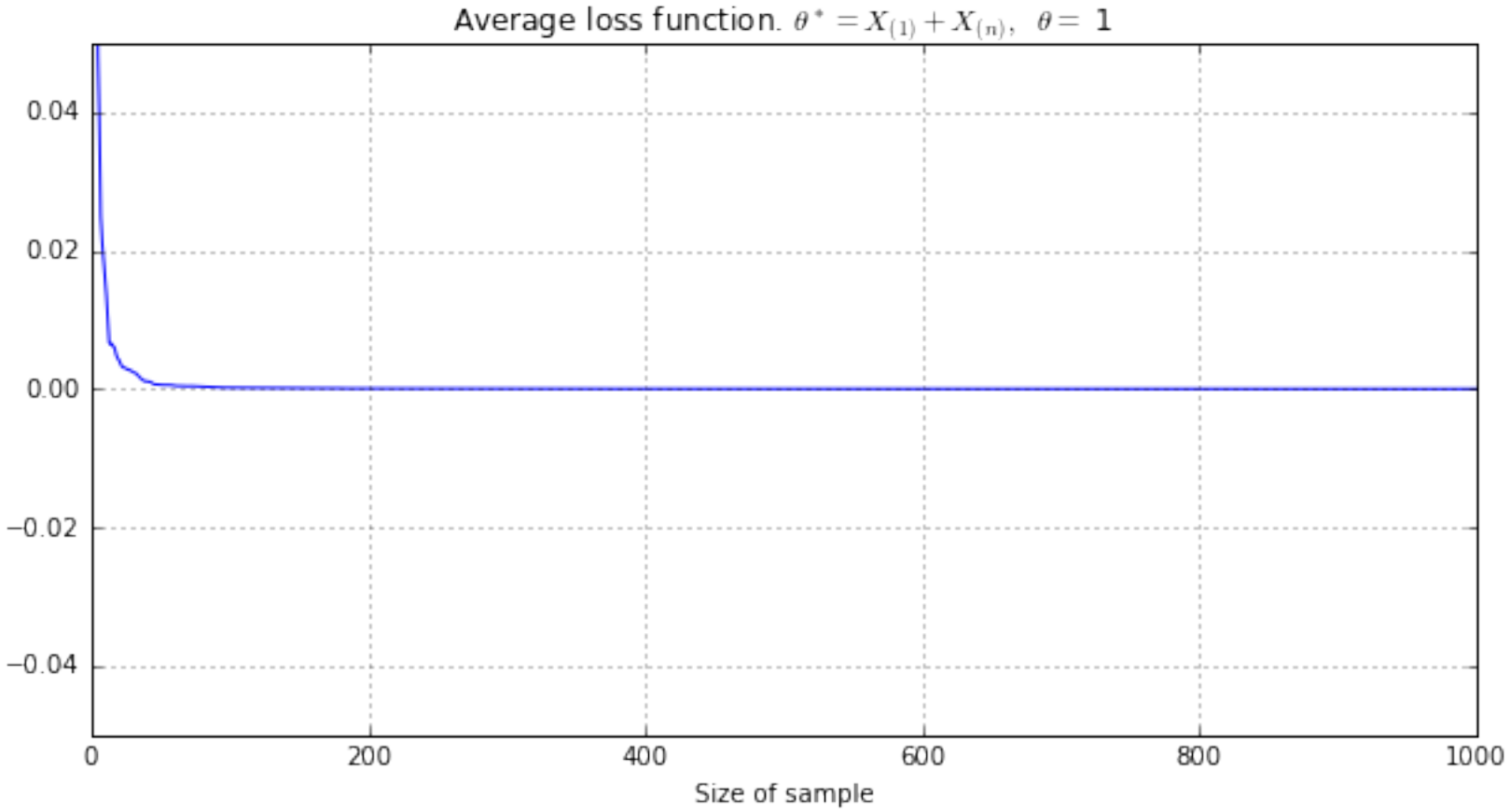
In [6]:
```
# Усредним функцию потерь по выборкам для каждого фиксированного n.
avg_lf_sample_mean_1 = np.array([np.average(lf_sample_mean_1[:, n]) for n in
range(N)])
avg_lf_koeff_min_1 = np.array([np.average(lf_koeff_min_1[:, n]) for n in rang
e(N)])
avg_lf_min_max_1 = np.array([np.average(lf_min_max_1[:, n]) for n in range(N)
])
avg_lf_koeff_max_1 = np.array([np.average(lf_koeff_max_1[:, n]) for n in rang
e(N)])
```

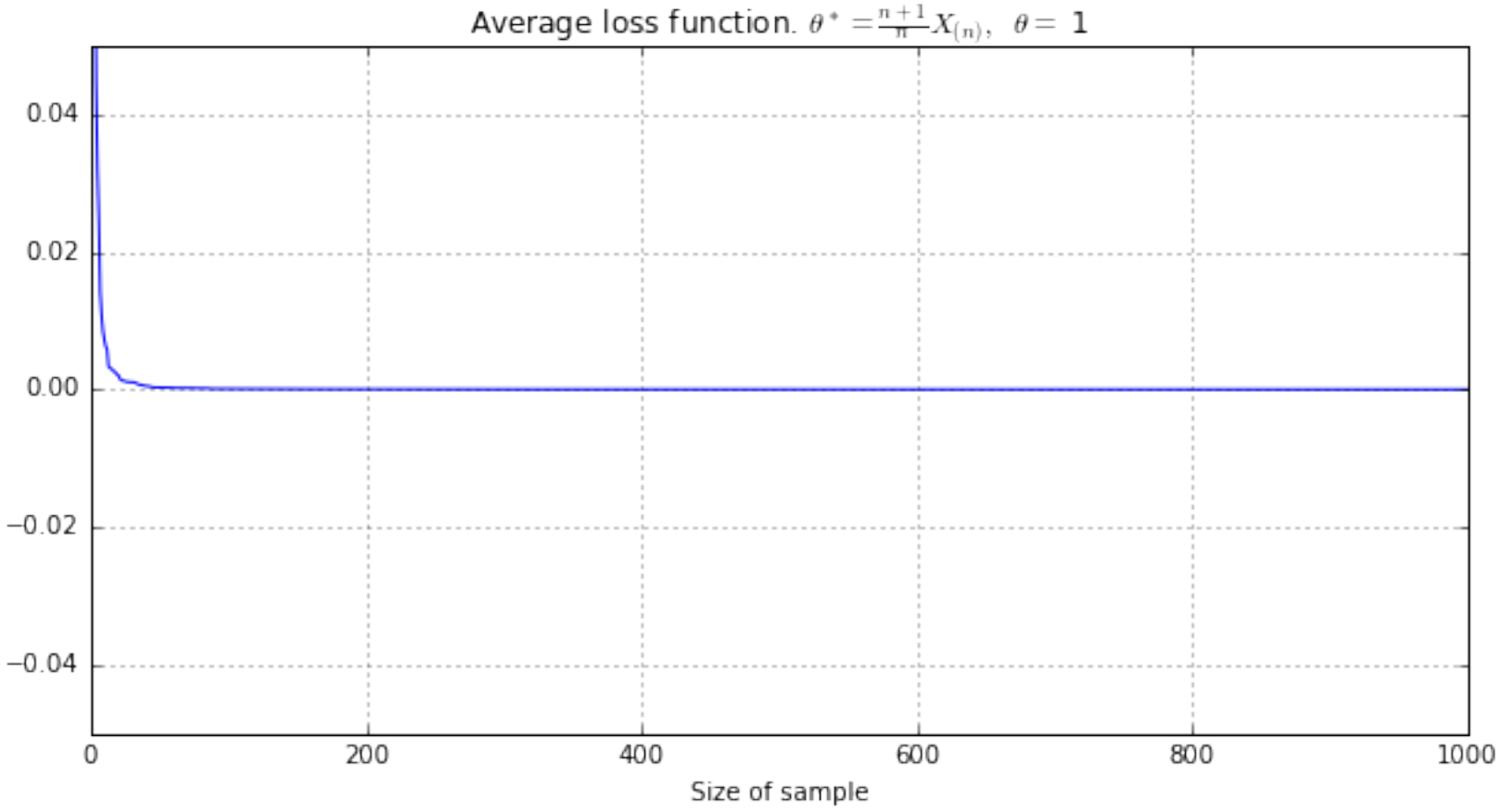
In [7]:
```
# Построим графики усредненных функций потерь.
plt.figure(figsize=(10, 5))
plt.plot(np.linspace(1., N, N), avg_lf_sample_mean_1, label=r'$2\bar{X}$')
plt.ylim(-0.05, 0.05)

plt.title(r'Average loss function. $\theta^*=2\bar{X}, \ \ \theta=$ ' + str(t
heta))
plt.xlabel(r'Size of sample', fontsize='10')
plt.grid()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(np.linspace(1., N, N), avg_lf_koeff_min_1, label=r'$(n + 1)X_{(1)}$'
)
plt.ylim(-0.05, 2)

plt.title(r'Average loss function. $\theta^*=(n + 1)X_{(1)}, \ \ \theta=$ ' +
str(theta))
plt.xlabel(r'Size of sample', fontsize='10')
plt.grid()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(np.linspace(1., N, N), avg_lf_min_max_1, label=r'$X_{(1)} + X_{(n)}$
')
plt.ylim(-0.05, 0.05)

plt.title(r'Average loss function. $\theta^*=X_{(1)} + X_{(n)}, \ \ \theta=$
' + str(theta))
plt.xlabel(r'Size of sample', fontsize='10')
plt.grid()
plt.show()
```

```
plt.figure(figsize=(10, 5))
plt.plot(np.linspace(1., N, N), avg_lf_koeff_max_1, label=r'$\frac{n+1}{n}X_{
(n)}$')
plt.ylim(-0.05, 0.05)

plt.title(r'Average loss function. $\theta^*=\frac{n+1}{n}X_{(n)}, \ \ \theta
=$ ' + str(theta))
plt.xlabel(r'Size of sample', fontsize='10')
plt.grid()
plt.show()
```









$\theta = 30$

In [8]:
```
# Сгенерируем выборки для theta = 30.
theta = 30
samples = np.array([uniform.rvs(size=N, scale=theta) for m in range(M)])
```

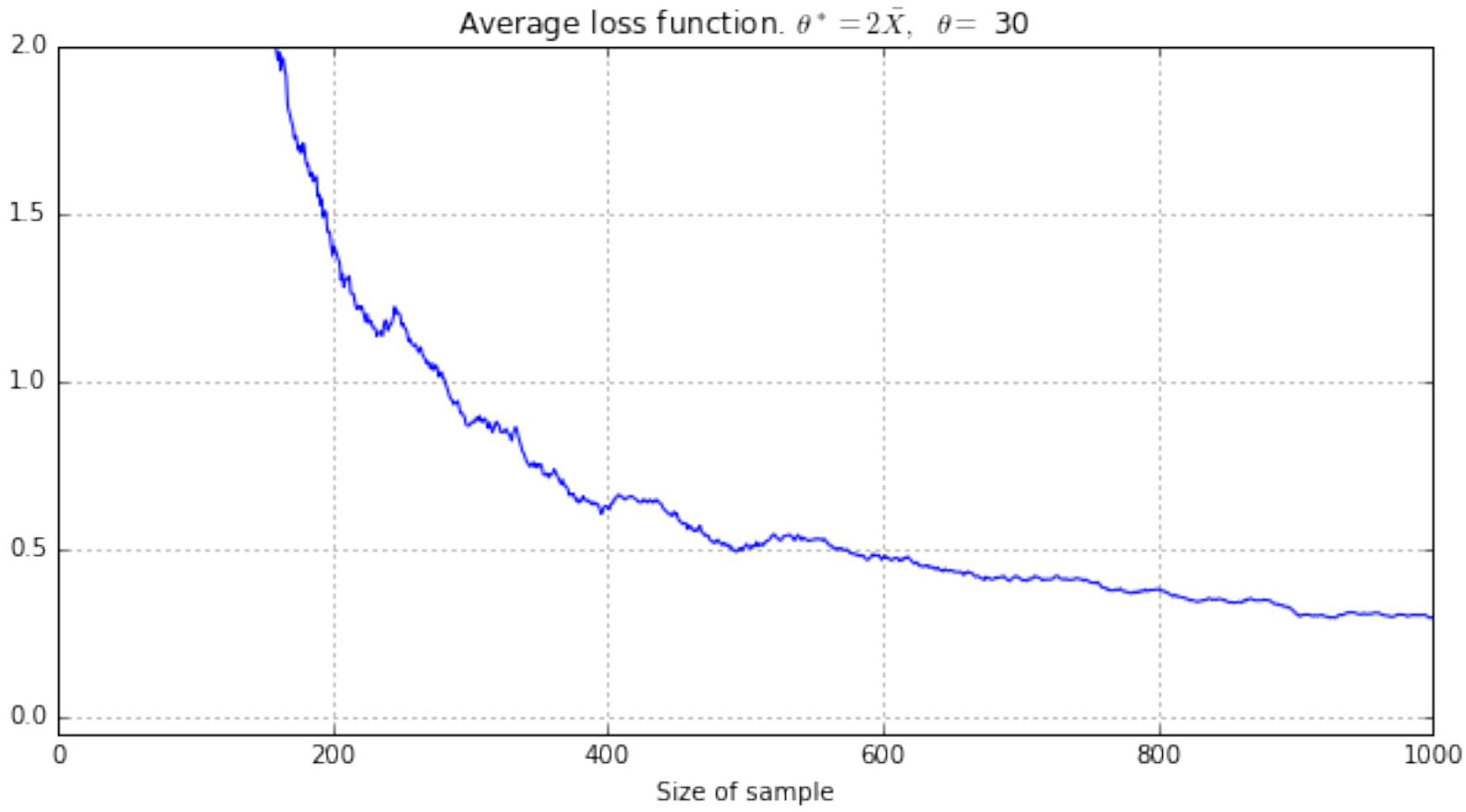
In [9]:
```
# Посчитаем оценки для каждой выборки для каждого n = 1,...,N.
sample_mean_2 = np.array([samples[m].cumsum() * [2 / n
                                                 for n in range(1, N + 1, 1)]
for m in range(M)])
koeff_min_2 = np.array([[(n + 1) * np.min(samples[m][ : n])
                         for n in range(1, N + 1, 1)] for m in range(M)])
min_max_2 = np.array([[np.min(samples[m][ : n + 1]) + np.max(samples[m][ : n
+ 1])
                       for n in range(N)] for m in range(M)])
koeff_max_2 = np.array([[(n + 1) / n * np.max(samples[m][ : n])
                         for n in range(1, N + 1, 1)] for m in range(M)])
```

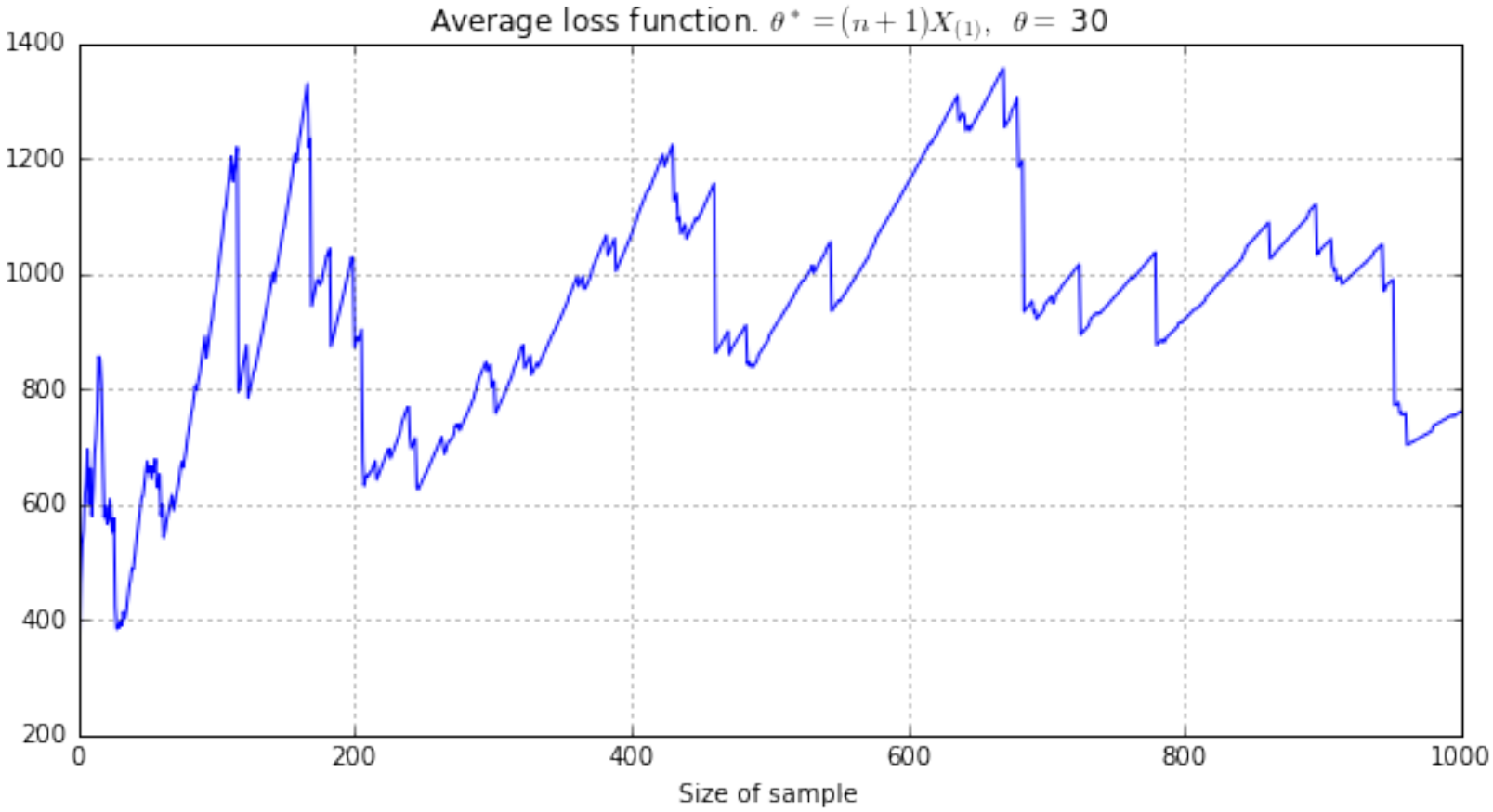
In [10]:
```
# Посчитаем квадратичную функцию потерь для каждого значения оценок.
lf_sample_mean_2 = (sample_mean_2 - theta) ** 2
lf_koeff_min_2 = (koeff_min_2 - theta) ** 2
lf_min_max_2 = (min_max_2 - theta) ** 2
lf_koeff_max_2 = (koeff_max_2 - theta) ** 2
```

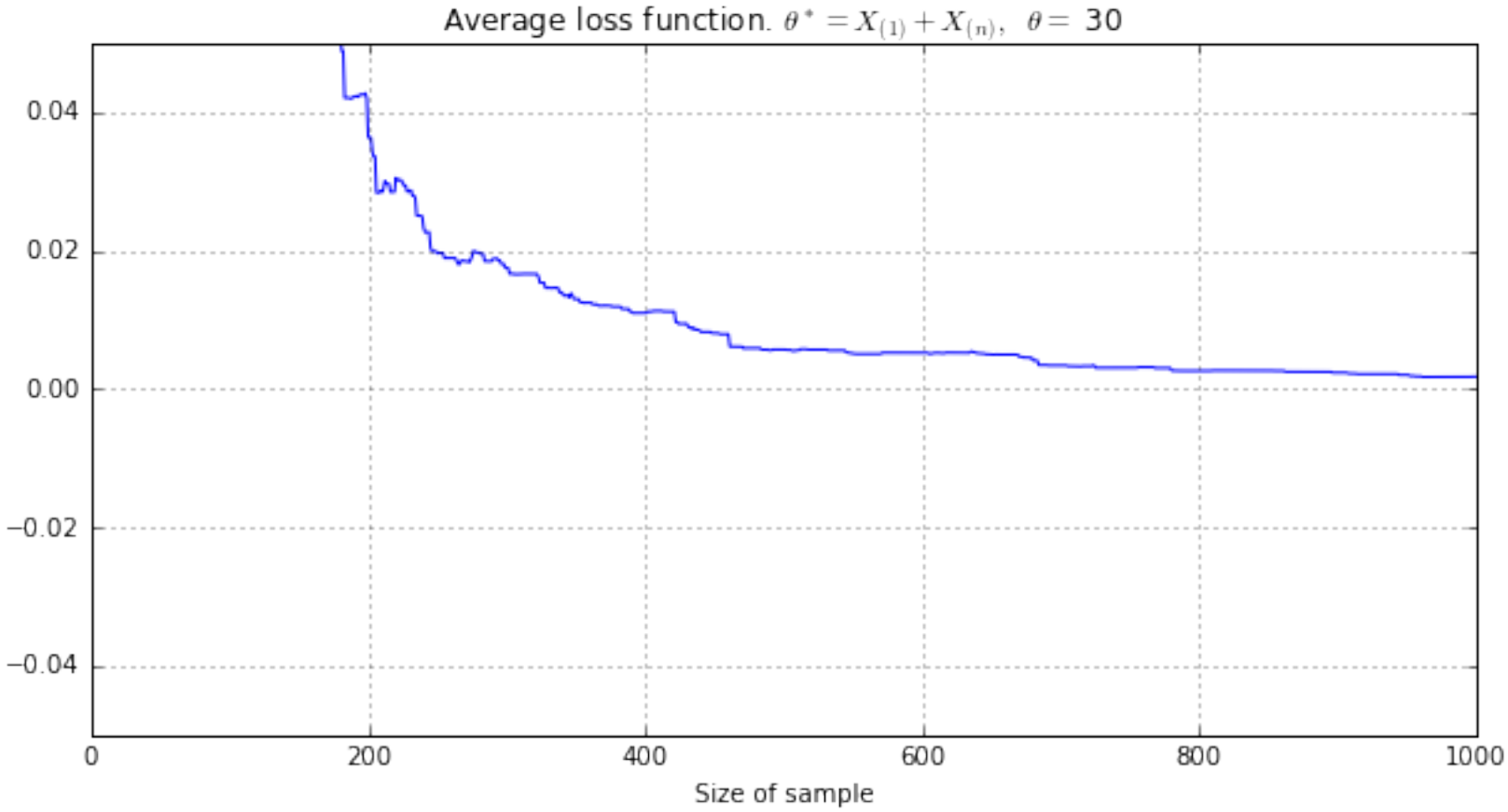
In [11]:
```
# Усредним функцию потерь по выборкам для каждого фиксированного n.
avg_lf_sample_mean_2 = np.array([np.average(lf_sample_mean_2[:, n]) for n in
range(N)])
avg_lf_koeff_min_2 = np.array([np.average(lf_koeff_min_2[:, n]) for n in rang
e(N)])
avg_lf_min_max_2 = np.array([np.average(lf_min_max_2[:, n]) for n in range(N)
])
avg_lf_koeff_max_2 = np.array([np.average(lf_koeff_max_2[:, n]) for n in rang
e(N)])
```

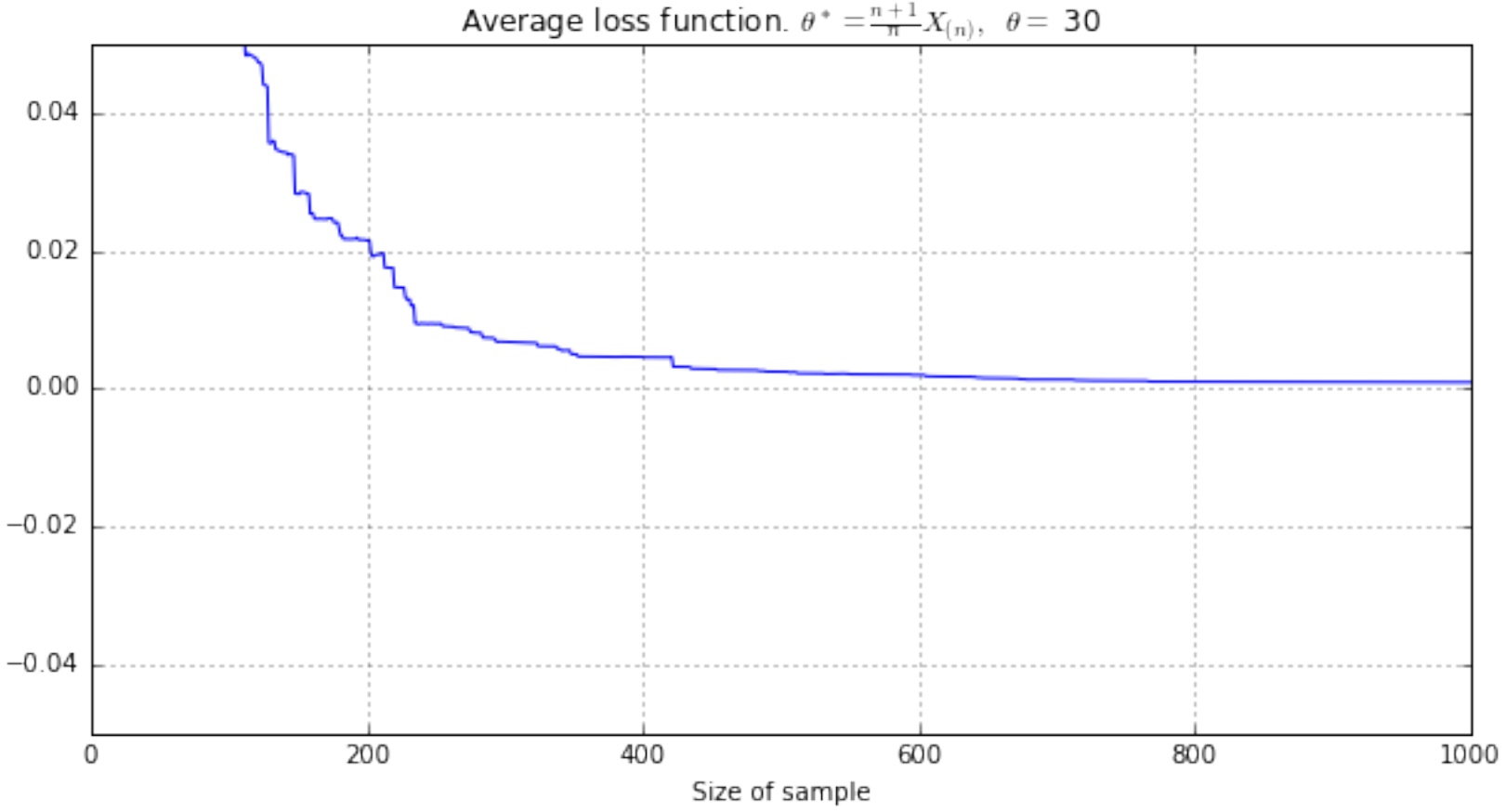
In [12]:
```
# Построим графики усредненных функций потерь.
plt.figure(figsize=(10, 5))
plt.plot(np.linspace(1., N, N), avg_lf_sample_mean_2, label=r'$2\bar{X}$')
plt.ylim(-0.05, 2)

plt.title(r'Average loss function. $\theta^*=2\bar{X}, \ \ \theta=$ ' + str(t
heta))
plt.xlabel(r'Size of sample', fontsize='10')
plt.grid()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(np.linspace(1., N, N), avg_lf_koeff_min_2, label=r'$(n + 1)X_{(1)}$'
)

plt.title(r'Average loss function. $\theta^*=(n + 1)X_{(1)}, \ \ \theta=$ ' +
str(theta))
plt.xlabel(r'Size of sample', fontsize='10')
plt.grid()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(np.linspace(1., N, N), avg_lf_min_max_2, label=r'$X_{(1)} + X_{(n)}$
')
plt.ylim(-0.05, 0.05)

plt.title(r'Average loss function. $\theta^*=X_{(1)} + X_{(n)}, \ \ \theta=$
' + str(theta))
plt.xlabel(r'Size of sample', fontsize='10')
plt.grid()
plt.show()

plt.figure(figsize=(10, 5))
```

```
plt.plot(np.linspace(1., N, N), avg_lf_koeff_max_2, label=r'$\frac{n+1}{n}X_{
(n)}$')
plt.ylim(-0.05, 0.05)

plt.title(r'Average loss function. $\theta^*=\frac{n+1}{n}X_{(n)}, \ \ \theta
=$ ' + str(theta))
plt.xlabel(r'Size of sample', fontsize='10')
plt.grid()
plt.show()
```









$\theta = 100$

In [13]:
```
# Сгенерируем выборки для theta = 100.
theta = 100
samples = np.array([uniform.rvs(size=N, scale=theta) for m in range(M)])
```

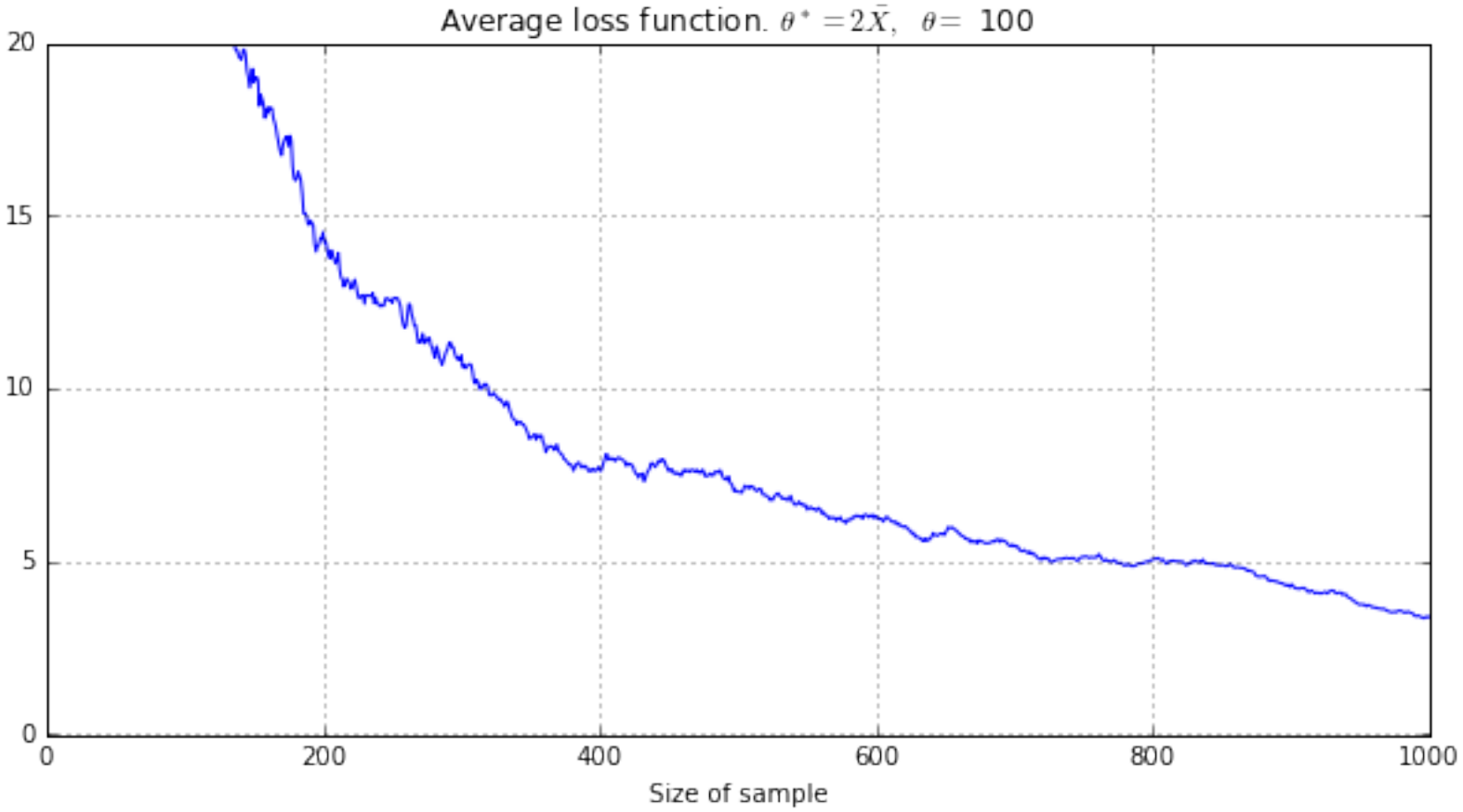
In [14]:
```
# Посчитаем оценки для каждой выборки для каждого n = 1,...,N.
sample_mean_3 = np.array([samples[m].cumsum() * [2 / n
                                                 for n in range(1, N + 1, 1)]
for m in range(M)])
koeff_min_3 = np.array([[(n + 1) * np.min(samples[m][ : n])
                         for n in range(1, N + 1, 1)] for m in range(M)])
min_max_3 = np.array([[np.min(samples[m][ : n + 1]) + np.max(samples[m][ : n
+ 1])
                       for n in range(N)] for m in range(M)])
koeff_max_3 = np.array([[(n + 1) / n * np.max(samples[m][ : n])
                         for n in range(1, N + 1, 1)] for m in range(M)])
```

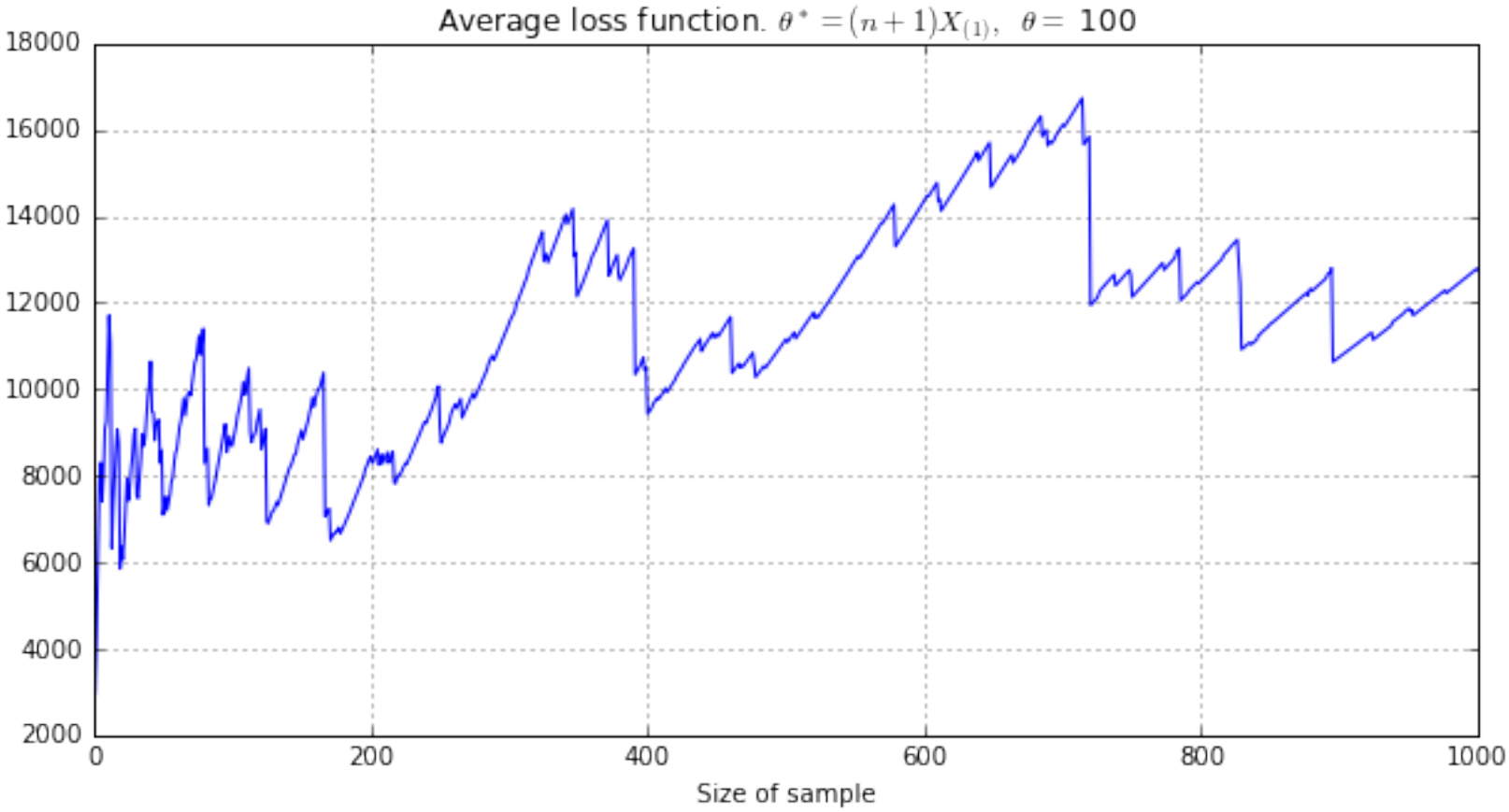
In [15]:
```
# Посчитаем квадратичную функцию потерь для каждого значения оценок.
lf_sample_mean_3 = (sample_mean_3 - theta) ** 2
lf_koeff_min_3 = (koeff_min_3 - theta) ** 2
lf_min_max_3 = (min_max_3 - theta) ** 2
lf_koeff_max_3 = (koeff_max_3 - theta) ** 2
```

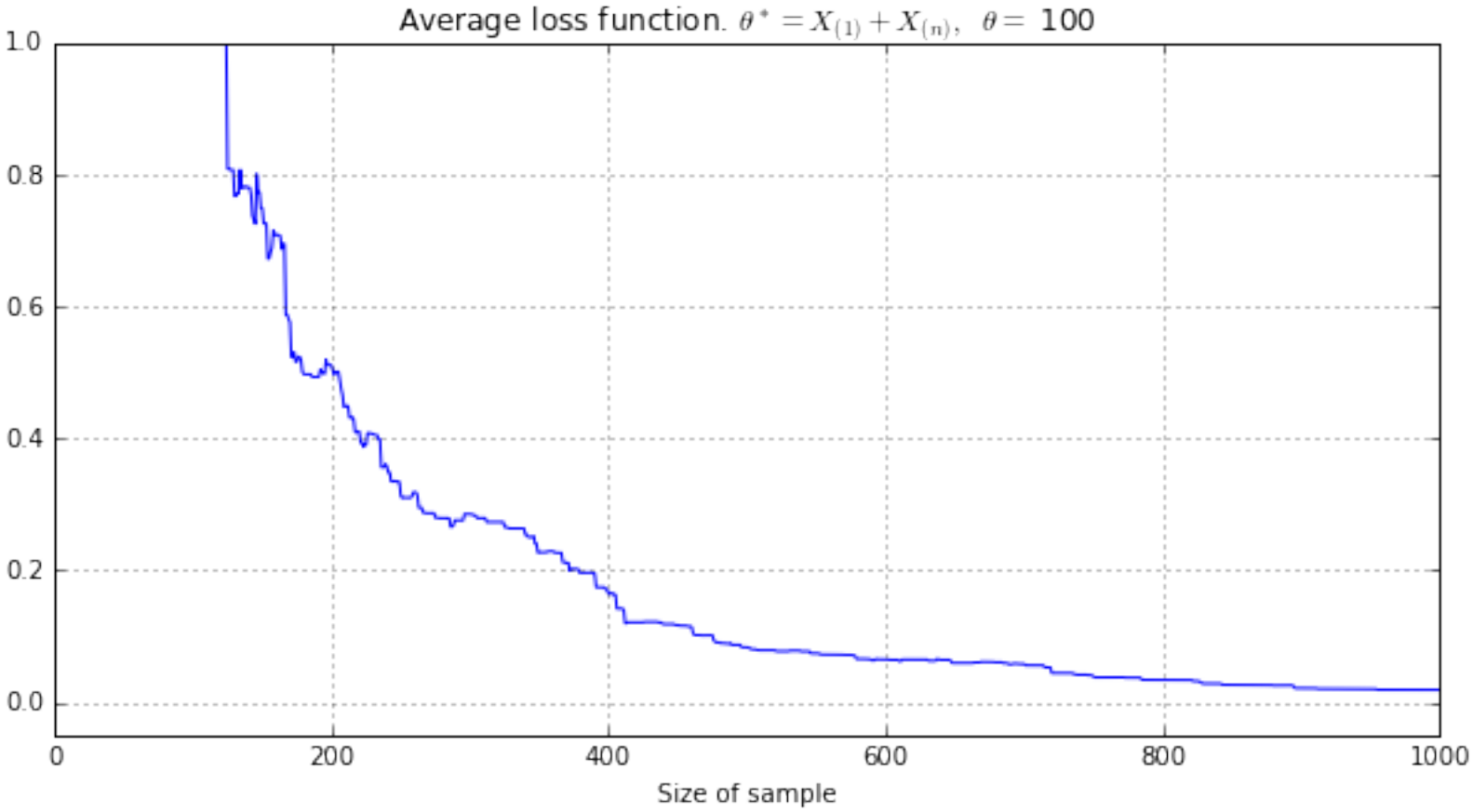
In [16]:
```
# Усредним функцию потерь по выборкам для каждого фиксированного n.
avg_lf_sample_mean_3 = np.array([np.average(lf_sample_mean_3[:, n]) for n in
range(N)])
avg_lf_koeff_min_3 = np.array([np.average(lf_koeff_min_3[:, n]) for n in rang
e(N)])
avg_lf_min_max_3 = np.array([np.average(lf_min_max_3[:, n]) for n in range(N)
])
avg_lf_koeff_max_3 = np.array([np.average(lf_koeff_max_3[:, n]) for n in rang
e(N)])
```

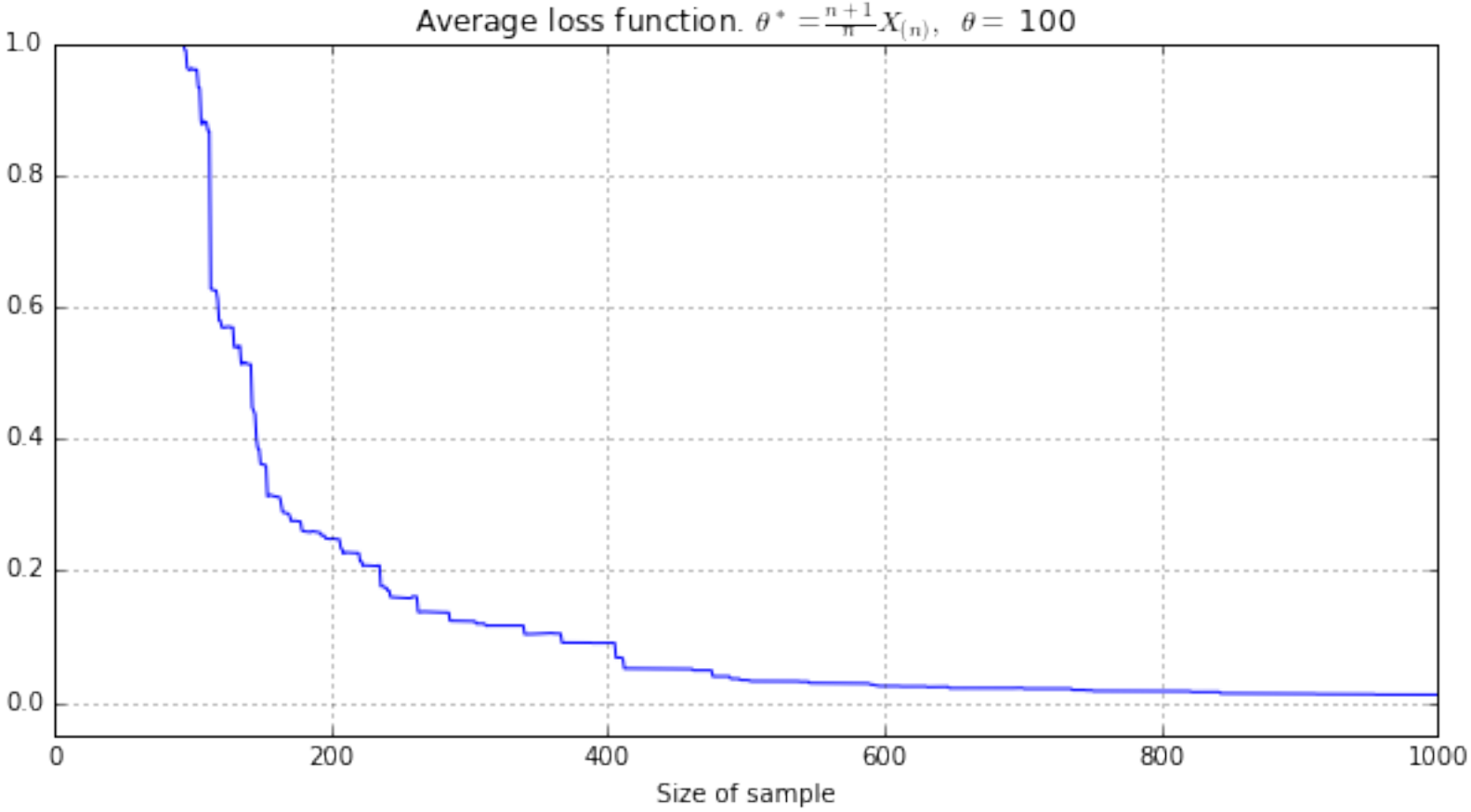
In [17]:
```
# Построим графики усредненных функций потерь.
plt.figure(figsize=(10, 5))
plt.plot(np.linspace(1., N, N), avg_lf_sample_mean_3, label=r'$2\bar{X}$')
plt.ylim(-0.05, 20)

plt.title(r'Average loss function. $\theta^*=2\bar{X}, \ \ \theta=$ ' + str(t
heta))
plt.xlabel(r'Size of sample', fontsize='10')
plt.grid()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(np.linspace(1., N, N), avg_lf_koeff_min_3, label=r'$(n + 1)X_{(1)}$'
)

plt.title(r'Average loss function. $\theta^*=(n + 1)X_{(1)}, \ \ \theta=$ ' +
str(theta))
plt.xlabel(r'Size of sample', fontsize='10')
plt.grid()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(np.linspace(1., N, N), avg_lf_min_max_3, label=r'$X_{(1)} + X_{(n)}$
')
plt.ylim(-0.05, 1)

plt.title(r'Average loss function. $\theta^*=X_{(1)} + X_{(n)}, \ \ \theta=$
' + str(theta))
plt.xlabel(r'Size of sample', fontsize='10')
plt.grid()
plt.show()

plt.figure(figsize=(10, 5))
```

```
plt.plot(np.linspace(1., N, N), avg_lf_koeff_max_3, label=r'$\frac{n+1}{n}X_{
(n)}$')
plt.ylim(-0.05, 1)

plt.title(r'Average loss function. $\theta^*=\frac{n+1}{n}X_{(n)}, \ \ \theta
=$ ' + str(theta))
plt.xlabel(r'Size of sample', fontsize='10')
plt.grid()
plt.show()
```









Вывод

Из графиков видно, что функции потерь последних двух оценок ближе к 0, а функция потерь второй оценки больше всех. Это означает, что оценки $X_{(1)} + X_{(n)}, \frac{n+1}{n}X_{(n)}$ лучше остальных, а оценка $(n + 1)X_{(1)}$ хуже всех.