Математическая статистика

Практическое задание 4

В данном задании предлагается провести некоторое исследование доверительных интервалов и байесовких методов.

Правила:

- Выполненную работу нужно отправить на почту `probability.diht@yandex.ru`, указав тему письма `"[номер группы] Фамилия Имя - Задание 4"`. Квадратные скобки обязательны. Вместо `Фамилия Имя` нужно подставить свои фамилию и имя.
- Прислать нужно ноутбук и его pdf-версию. Названия файлов должны быть такими: `4.N.ipynb` и `4.N.pdf`, где `N` - ваш номер из таблицы с оценками.
- Никакой код из данного задания при проверке запускаться не будет.
- Некоторые задачи отмечены символом $*$. Эти задачи являются дополнительными. Успешное выполнение большей части таких задач (за все задания) является необходимым условием получения бонусного балла за практическую часть курса.
- Баллы за каждую задачу указаны далее. Если сумма баллов за задание меньше 25% (без учета доп. задач), то все задание оценивается в 0 баллов.

Баллы за задание:

- Задача 1 - 5 баллов
- Задача 2$*$ - 3 балла
- Задача 3$*$ - 3 балла
- Задача 4 - 5 баллов
- Задача 5$*$ - 2 балла
- Задача 6 - 4 балла
- Задача 7 - 1 балл
- Задача 8 - 3 балла
- Задача 9$*$ - 5 баллов
- Задача 10 - 5 баллов
- Задача 11$*$ - 3 балла

In [1]:
```
import numpy as np
import scipy.stats as sps
import matplotlib.pyplot as plt
from math import sqrt, pi
from statsmodels.distributions.empirical_distribution import ECDF
from heapq import heappush, heappop

%matplotlib inline
```

1. Доверительные интервалы

Задача 1. В этой задаче нужно визуализировать доверительные интервалы для выборок из различных распределений. Чтобы не плодить код, напишите следующую функцию. Пример построения есть в ячейке №22 ноутбука python_6.

In [2]:
```
α = 0.95
n = 100


def draw_confidence_interval(left_estimator,  # левая граница интервала
                             right_estimator,  # правая граница интервала
                             true_theta,  # если задана, то рисуется график оценки
                             sample,
                             draw_sample=True,  # рисовать ли точки выборки
                             ylim=(None, None),  # ограничение по оси y
                             title=None):
    xs = np.arange(1, n + 1)
    left = [left_estimator(sample[:i]) for i in xs]
    right = [right_estimator(sample[:i]) for i in xs]

    plt.figure(figsize=(15, 5))
    if draw_sample:
        plt.scatter(xs, sample, alpha=0.5, label='sample')
    plt.fill_between(xs, left, right, alpha=0.15, label='confidence interval')
    plt.plot(xs, [true_theta] * n, color='red', label='true $\\theta$')
    plt.xlim((1, n))
    if ylim != (None, None):
        plt.ylim(ylim)
    plt.title(title)
    plt.grid()
    plt.legend()
    plt.show()
```

Сгенерируйте выборки и постройте графики доверительных интервалов по следующей схеме.

- Выборка из распределения $\mathcal{N}(0, 1)$; точный доверительный интервал минимальной длины в модели $\mathcal{N}(\theta, 1)$; нужно нанести на график точки выборки.
- Выборка из распределения $U[0, 1]$; точный доверительный интервал минимальной длины в модели $U[0, \theta]$ на основе статистики $X_{(n)}$; нужно нанести на график точки выборки.
- Выборка из распределения $\Gamma(3, 2)$; точный асимптотический доверительный интервал в модели $\Gamma(\theta, 2)$; точки выборки наносить на график не нужно.

- Выборка из стандартного распределения Коши; точный асимптотический доверительный интервал в модели распределения Коши со сдвигом; нужно нанести на график точки выборки.
- Выборка из стандартного распределения Коши; точный доверительный интервал минимальной длины в модели $\mathcal{N}(\theta, 1)$; нужно нанести на график точки выборки.

Генерировать выборки размера 100, уровень доверия брать $\alpha = 0.95$. Для вычисления квантилей у каждого распределения из `scipy.stats` есть функция `ppf`.

Сделайте вывод. Насколько часто истинное значение параметра попадает в доверительный интервал? Как длина интервала зависит от размера выборки?

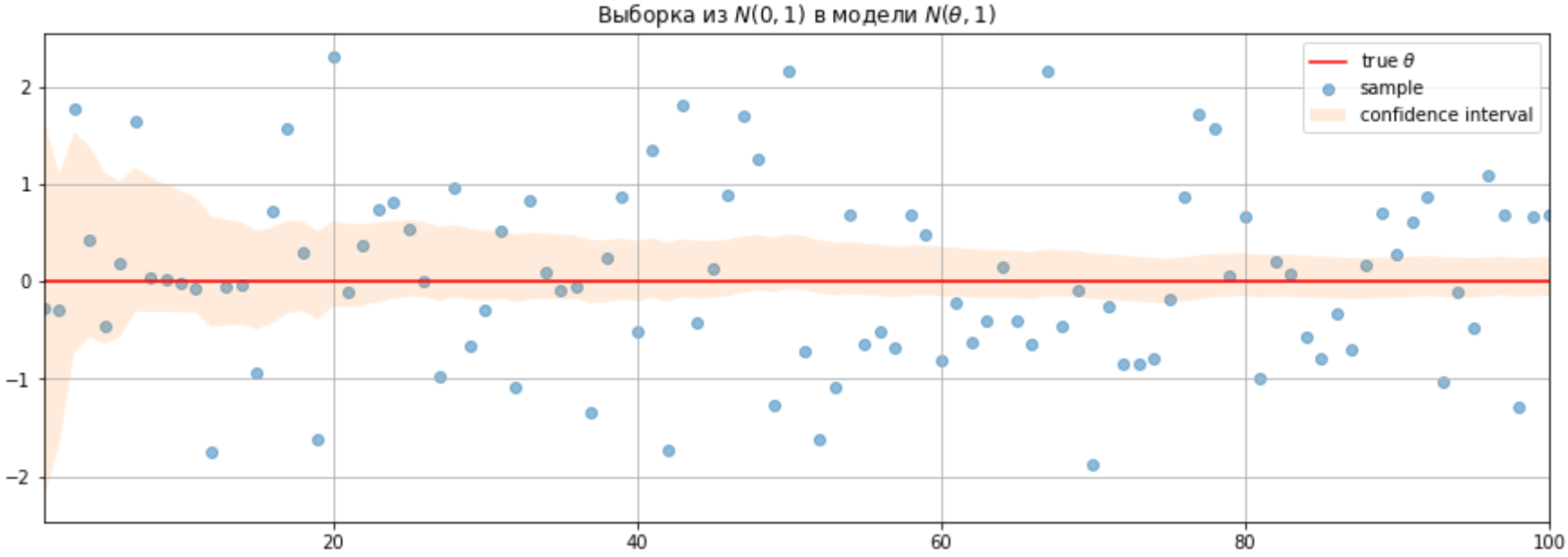
- $X_1 \dots X_n \sim N(\theta, 1)$
  $\sqrt{n}(\overline{X} - \theta) \sim N(0, 1)$
  ДИ: $(\overline{X} + \frac{z_{\frac{1-\alpha}{2}}}{\sqrt{n}}, \overline{X} + \frac{z_{\frac{1+\alpha}{2}}}{\sqrt{n}})$, где z_γ --- квантиль $N(0, 1)$ уровня γ
- $X_1 \dots X_n \sim U[0, \theta]$
  ДИ: $(X_{(n)}, \frac{X_{(n)}}{\sqrt[n]{1-\alpha}})$
- $X_1 \dots X_n \sim \Gamma(\theta, 2)$
  Из ЦПТ: $\sqrt{\frac{n}{\beta}}(\overline{X}\theta - \beta) \xrightarrow{d} N(0, 1)$, где $\beta == 2$
  ДИ: $(\frac{\sqrt{\frac{\beta}{n}} z_{\frac{1-\alpha}{2}} + \beta}{\overline{X}}, \frac{\sqrt{\frac{\beta}{n}} z_{\frac{1+\alpha}{2}} + \beta}{\overline{X}})$
- $X_1 \dots X_n \sim Cauchy(\theta)$ --- распределение Коши со сдвигом
  Медиана распределения $Cauchy(\theta)$ равна θ
  По теореме о выборочной медиане

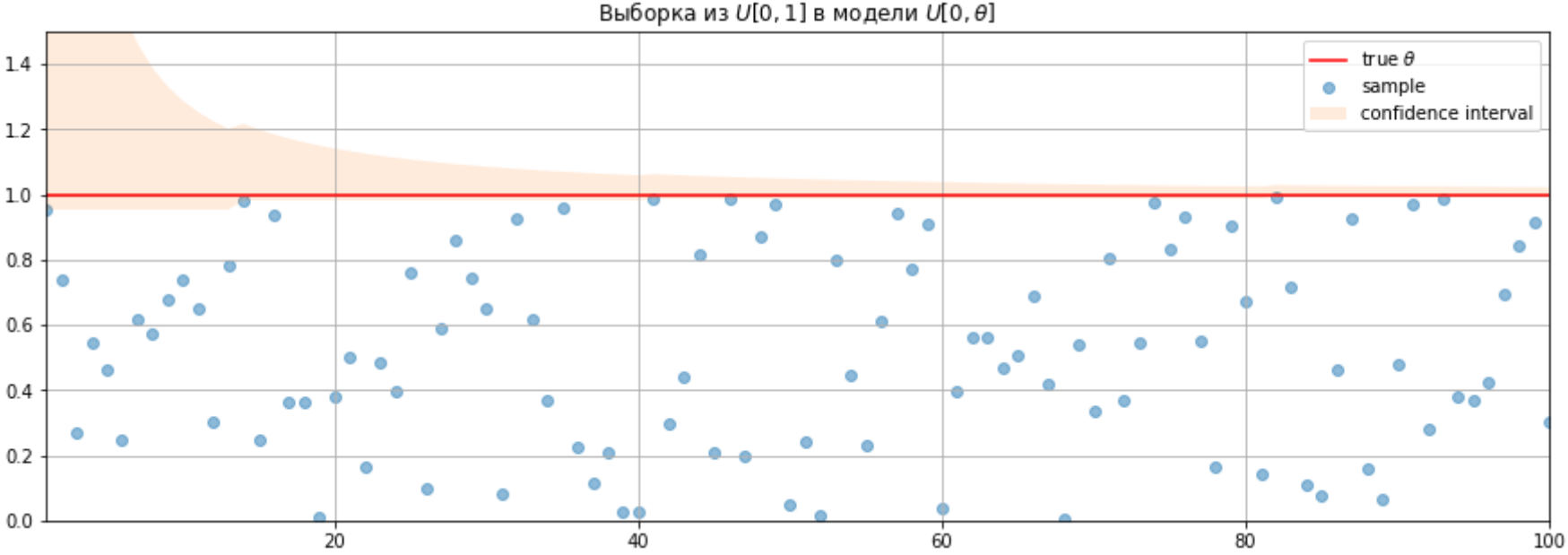
$$\sqrt{n}(\widehat{\mu} - z_{\frac{1}{2}}) \xrightarrow{d} N(0, \frac{\pi^2}{4})$$

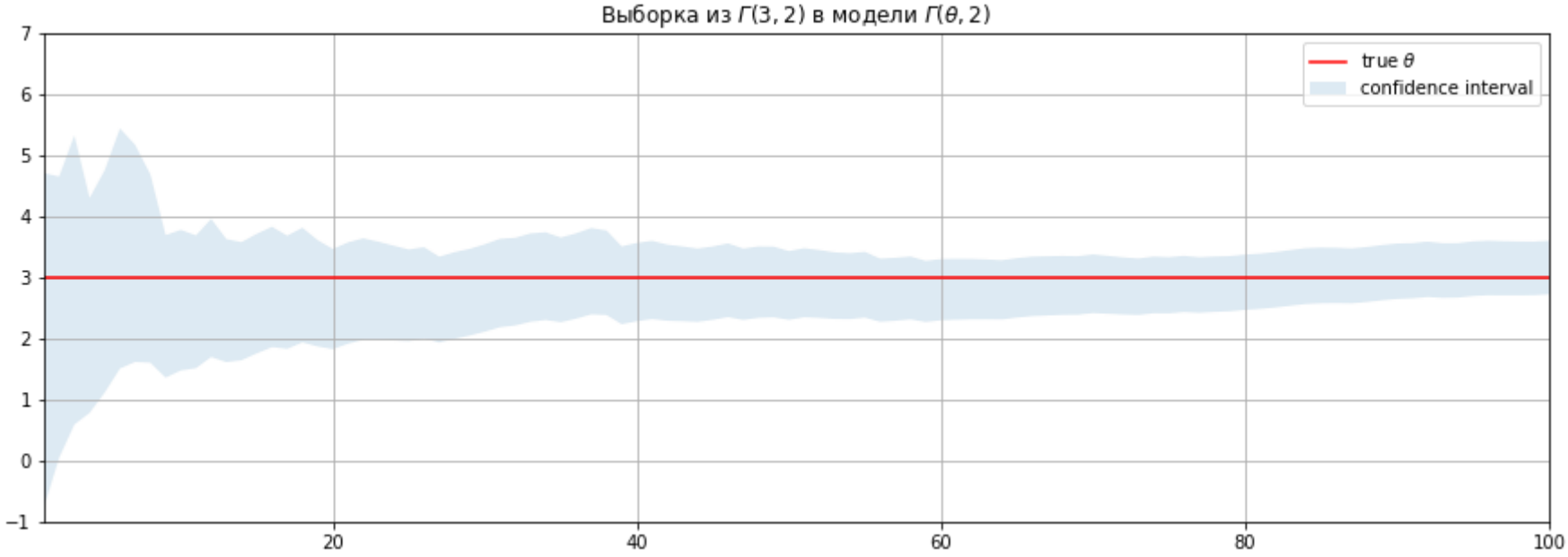
$$\frac{2\sqrt{n}}{\pi}(\widehat{\mu} - \theta) \xrightarrow{d} N(0, 1)$$

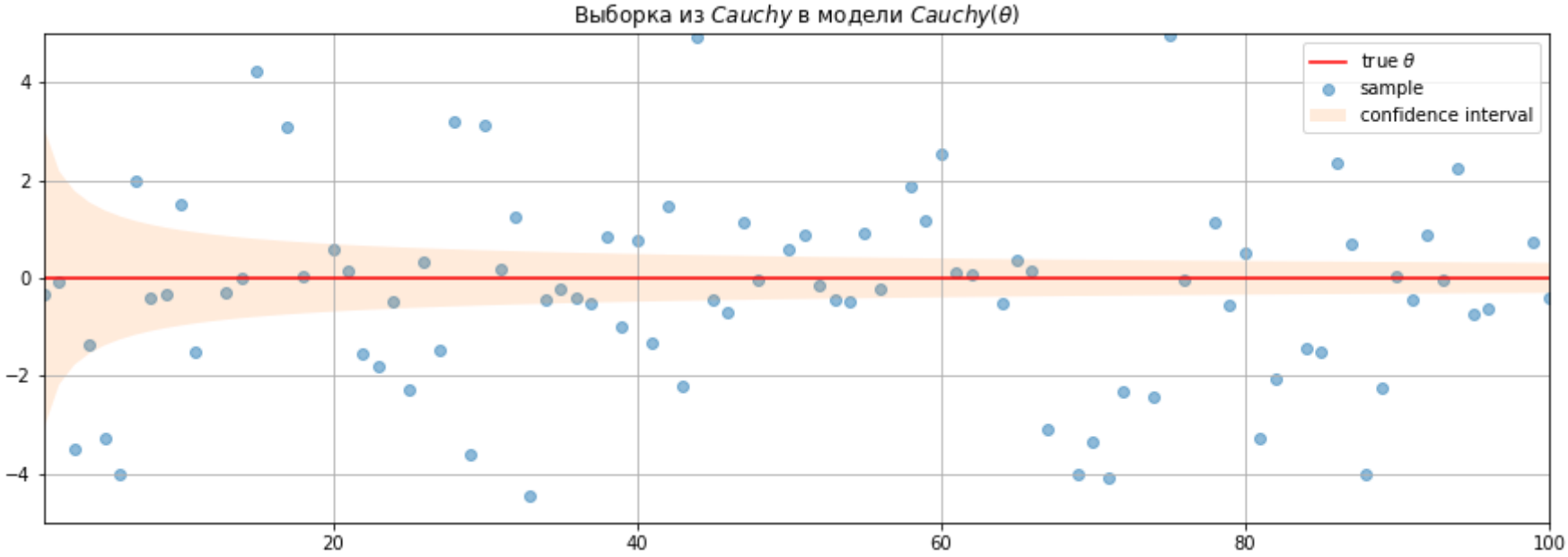
ди: $(\widehat{\mu} + \frac{\pi}{2\sqrt{n}} z_{\frac{1-\alpha}{2}}, \widehat{\mu} + \frac{\pi}{2\sqrt{n}} z_{\frac{1+\alpha}{2}})$

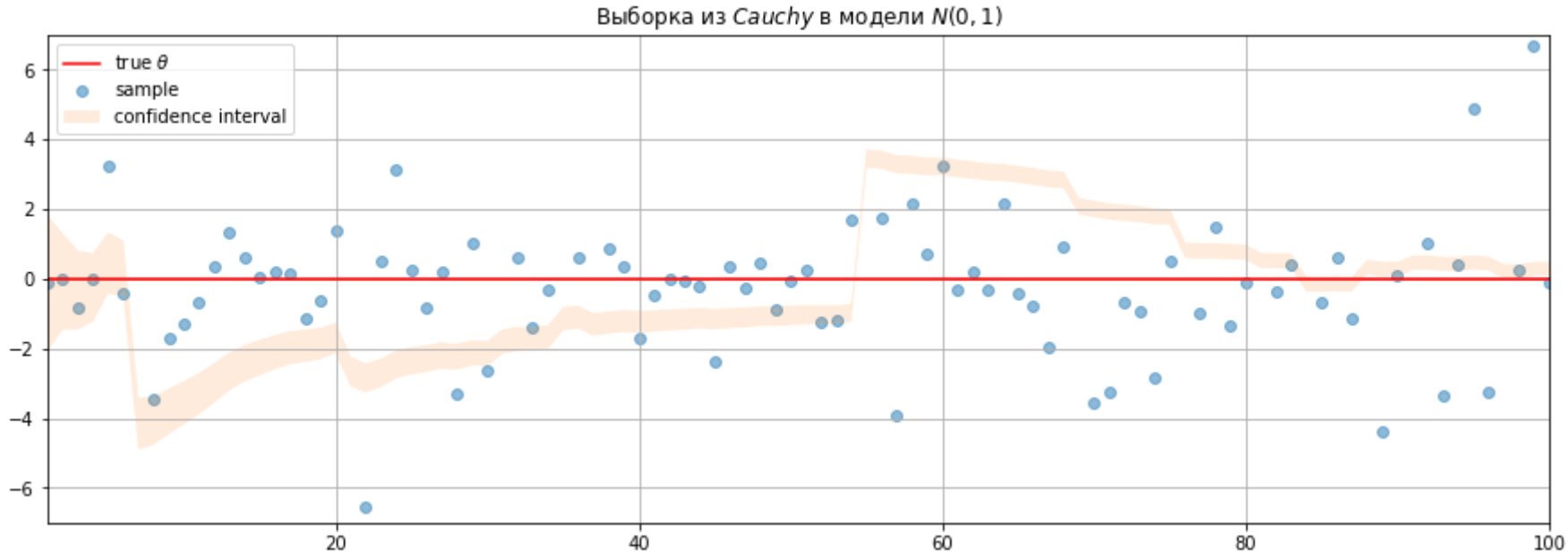
In [3]:
```
z1 = sps.norm.ppf((1 - α) / 2)
z2 = sps.norm.ppf((1 + α) / 2)
draw_confidence_interval(lambda sample: sample.mean() + z1 / sqrt(len(sample)),
                         lambda sample: sample.mean() + z2 / sqrt(len(sample)),
                         true_theta=0,
                         sample=sps.norm.rvs(size=n),
                         title='Выборка из $N(0,1)$ в модели $N(\\theta,1)$')

draw_confidence_interval(lambda sample: sample.max(),
                         lambda sample: sample.max() / (1 - α) ** (1 / len(sample)),
                         true_theta=1,
                         sample=sps.uniform.rvs(size=n),
                         ylim=(0, 1.5),
                         title='Выборка из $U[0,1]$ в модели $U[0,\\theta]$')

draw_confidence_interval(lambda sample: (sqrt(2 / len(sample)) * z1 + 2) / sample.mean(),
                         lambda sample: (sqrt(2 / len(sample)) * z2 + 2) / sample.mean(),
                         true_theta=3,
                         sample=sps.gamma(a=2, scale=1 / 3).rvs(size=n),
                         draw_sample=False,
                         title='Выборка из $Г(3,2)$ в модели $Г(\\theta,2)$',
                         ylim=(-1, 7))

draw_confidence_interval(lambda sample: (pi * z1) / (2 * sqrt(len(sample))),
                         lambda sample: (pi * z2) / (2 * sqrt(len(sample))),
                         true_theta=0,
                         sample=sps.cauchy.rvs(size=n),
                         ylim=(-5, 5),
                         title='Выборка из $Cauchy$ в модели $Cauchy(\\theta)$')

draw_confidence_interval(lambda sample: sample.mean() + z1 / sqrt(len(sample)),
                         lambda sample: sample.mean() + z2 / sqrt(len(sample)),
                         true_theta=0,
                         sample=sps.cauchy.rvs(size=n),
                         ylim=(-7, 7),
                         title='Выборка из $Cauchy$ в модели $N(0,1)$')
```


Выборка из $N(0,1)$ в модели $N(\theta,1)$
true θ
sample
confidence interval
2
1
0
−1
−2
20
40
60
80
100


Выборка из $U[0,1]$ в модели $U[0,\theta]$
true θ
sample
confidence interval
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
20
40
60
80
100


Выборка из $\Gamma(3, 2)$ в модели $\Gamma(\theta, 2)$
true θ
confidence interval
7
6
5
4
3
2
1
0
−1
20
40
60
80
100


Выборка из *Cauchy* в модели *Cauchy*(θ)
true θ
sample
confidence interval
4
2
0
−2
−4
20
40
60
80
100



Вывод: Для обычных доверительных интервалов истинное значение параметра попадает во все интервалы почти каждом запуске. Для асимптотических доверительных интервалов истинное значение параметра попадает во все интервалы, кроме нескольких первых, также в почти каждом запуске. В последнем графике истинное значение параметра почти не попадает в интервалы, так как мы находим доверительные интервалы в модели нормального распределения, а выборка у нас из распределения Коши. Чем больше размер выборки, тем уже инервал.

Задача 2*. Аналогично заданию 1 постройте доверительные интервалы для следующих случаев

- Выборка из распределения $\Gamma(3, 2)$; точный асимптотический доверительный интервал для θ в модели $\Gamma(\theta, \beta)$, причем β неизвестно; точки выборки наносить на график не нужно. Сравните с интервалом для случая известного β.
- Выборка из распределения $\Gamma(3, 2)$; точный асимптотический доверительный интервал для β в модели $\Gamma(\theta, \beta)$, причем θ неизвестно; точки выборки наносить на график не нужно.

Задача 3*. Сгенерируйте выборку размера 200 из распределения $\mathcal{N}((0, 0)^T, ((2, 1)^T, (1, 3)^T))$. Постройте

точную доверительную область для θ в модели $\mathcal{N}(\theta, ((2,1)^T, (1,3)^T))$. Нанесите на график точки выборки.

Задача 4. При использовании асимптотических доверительных интервалов важно понимать, какой размер выборки является достаточным для хорошего приближения. Иначе говоря, пусть $\xi_n \xrightarrow{d} \mathcal{N}(0,1)$. Начиная с какого n распределение статистики ξ_n хорошо приближается нормальным распределением?

Для ответа на этот вопрос проведите следующее исследование. Сгенерируйте $K = 10^5$ выборок $(X_{i,k},\ i \leqslant N)$ размера $N = 300$, где $k \leqslant K$ --- номер выборки. Для каждой подвыборки k-ой выборки $(X_{i,k},\ i \leqslant n)$ посчитайте значение статистики $T_{n,k}$ (определение далее) для всех $n \leqslant N$. Далее для каждого фиксированного n постройте эмпирическую функцию распределения F_n^* по выборке $(T_{n,k}, k \leqslant K)$ и посчитайте точное значение статистики $D_n = \sup\limits_{x \in \mathbb{R}} |F_n^*(x) - F(x)|$, где F --- функция распределения $\mathcal{N}(0,1)$ (см. задачу 4 задания 1). Постройте график зависимости D_n от n.

Рассмотрите следующие случаи

- $X_1, \ldots, X_n \sim \mathcal{N}(0,1)$. Рассмотреть $T = \sqrt{n} \cdot \overline{X}$ и $T = \sqrt{n} \cdot \overline{X} \Big/ \sqrt{S^2}$.
- $X_1, \ldots, X_n \sim Bern(p), p = 0.5$. Рассмотреть $T = \sqrt{n}\frac{\overline{X} - p}{\sqrt{p(1-p)}}$ и $T_n = \sqrt{n}\frac{\overline{X} - p}{\sqrt{S^2}}$.
- $X_1, \ldots, X_n \sim Cauchy$. Рассмотреть $T = \sqrt{n}\frac{\widehat{\mu}}{\pi/2}$.

В первых двух пунктах нужно построить две зависимости на одном графике для сравнения. Масштаб графика должен быть таким, чтобы четко можно было увидеть различие между двумя статистиками. Например, поставьте ограничение сверху по оси y на 0.05. Незабудьте добавить сетку и легенду.

Старайтесь не копировать много кода, пишите вспомогательные функции. Обратите внимание, что оптимальный код для первых двух пунктов выполняется за 30 секунд, для третьего --- за 3 минуты. Неоптимальный код может выполняться более часа.

Сделайте вывод о том, в каком случае распределение статистики быстрее приближается нормальным распределением. Начиная с какого размера выборки можно пользоваться приближением нормальным распределением?

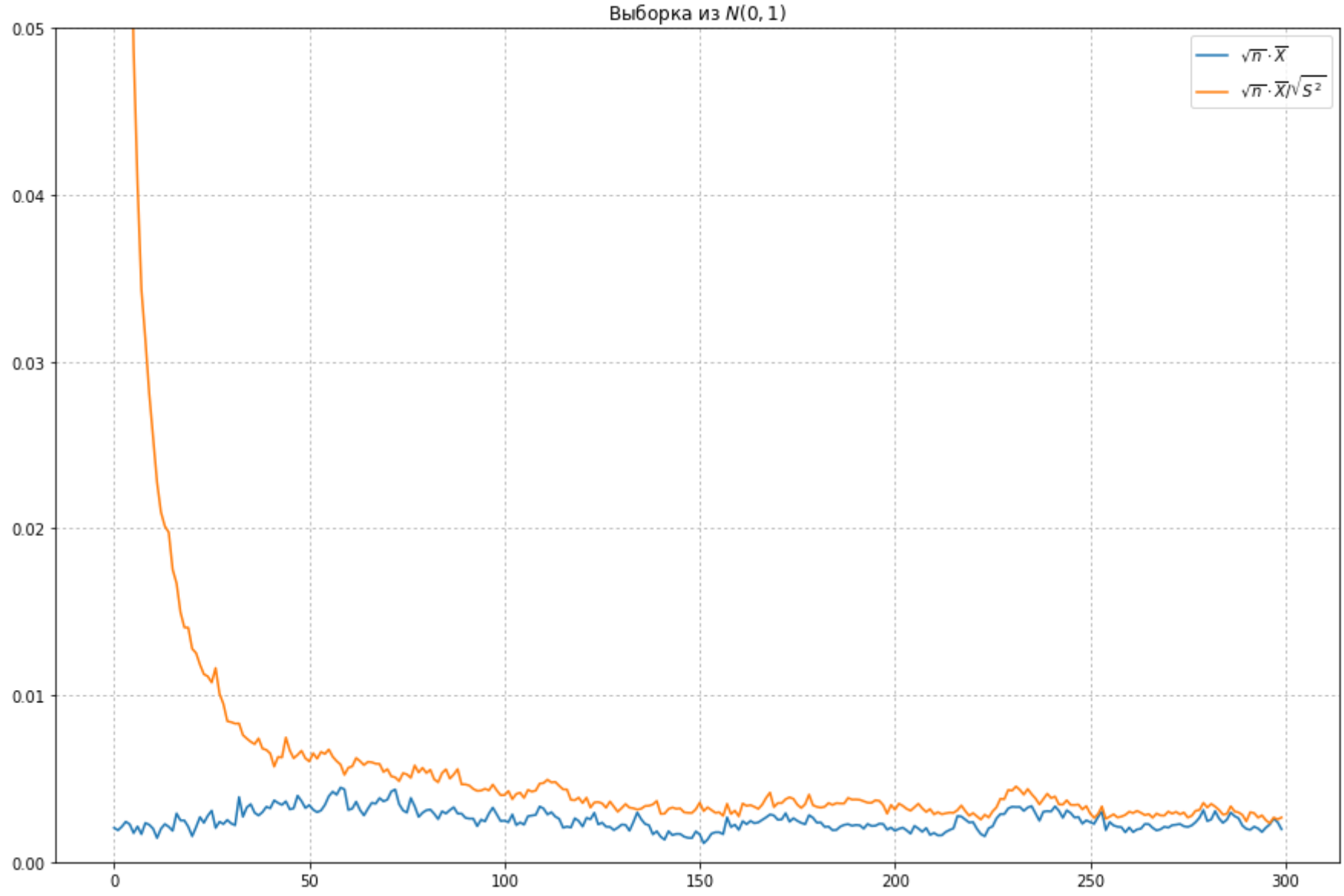
In [4]:
```
K = 10 ** 5
N = 300


def get_supremum(sample):
    """По выборке строит эмпирическую функцию распределения F*, считает точное значение статистики sup|F* - F
    eps = 10 ** -7
    ecdf = ECDF(sample)
    return max([np.abs(ecdf(sample + offset) - sps.norm.cdf(sample)).max() for offset in (-eps, 0)])


def cumavg(X):
    """
    По матрице (x_i_j) строит последовательность средних частичных сумм каждой строки
    result_i_j == X[i, :j+1].mean()
    """
    return X.cumsum(axis=1) / np.arange(1, len(X[0]) + 1)


def cumstd(X):
    """
    По матрице (x_i_j) строит последовательность стандартных отклонений (S^2) каждой строки
    result_i_j == X[i, :j+1].std()
    """
    return np.sqrt(cumavg(X ** 2) - cumavg(X) ** 2)


def cummedian_line(a):
    """
    По строке (a_i) строит последовательность медиан
    result_i == a[:i+1].median()

    http://stackoverflow.com/a/10657732
    O(nlog(n))
    """

    class minheap(list):
        def heappush(self, value):
            heappush(self, value)

        def heappop(self):
```

```
            return heappop(self)

        def heappeek(self):
            return self[0]

    class maxheap(list):
        def heappush(self, value):
            heappush(self, -value)

        def heappop(self):
            return -heappop(self)

        def heappeek(self):
            return -self[0]

    assert len(a) >= 2
    halfmin = maxheap()
    halfmax = minheap()
    halfmin.heappush(min(a[:2]))
    halfmax.heappush(max(a[:2]))
    medians = [a[0], (a[0] + a[1]) / 2]
    a = a[2:]

    for ai in a:
        (halfmin if ai <= halfmax.heappeek() else halfmax).heappush(ai)
        if len(halfmin) - len(halfmax) > 1:
            halfmax.heappush(halfmin.heappop())
        if len(halfmax) - len(halfmin) > 1:
            halfmin.heappush(halfmax.heappop())
        assert abs(len(halfmin) - len(halfmax)) <= 1
        medians.append((halfmin.heappeek() + halfmax.heappeek()) / 2
                       if (len(halfmin) + len(halfmax)) % 2 == 0
                       else (halfmin if len(halfmin) > len(halfmax) else halfmax).heappeek())
    return np.array(medians)


def cummedian_matrix(X):
    """
    По матрице (x_i_j) строит последовательность медиан каждой строки
    result_i_j == X[i, :j+1].median()
    """
    return np.array([cummedian_line(line) for line in X])
```

In [5]:
```
def draw(rvs, t_estimators, title, ylim=None):
    plt.figure(figsize=(15, 10))
    samples = rvs(size=(K, N))
    for t_estimator, t_estimator_label in t_estimators:
        ts = t_estimator(samples)
        supremums = [get_supremum(ts[:, n]) for n in range(N)]
        plt.plot(range(N), supremums, label=t_estimator_label)
    if ylim is not None:
        plt.ylim(ylim)
    plt.title(title)
    plt.grid(ls=':')
    plt.legend()
    plt.show()

def part1():
    t_estimators = [
        (lambda samples: np.sqrt(np.arange(1, N + 1))
         * cumavg(samples), '$\sqrt{n}\cdot\overline{X}$'),
        (lambda samples: np.sqrt(np.arange(1, N + 1))
         * cumavg(samples) / cumstd(samples), '$\sqrt{n}\cdot\overline{X}/\sqrt{S^2}$')
    ]
    draw(sps.norm.rvs, t_estimators, 'Выборка из $N(0, 1)$', (0, 0.05))


def part2():
    t_estimators = [
        (lambda samples: np.sqrt(np.arange(1, N + 1))
         * (cumavg(samples) - 0.5) * sqrt(2), '$\sqrt{n} \\frac{\overline{X} - p}{\sqrt{p(1-p)}}$'),
        (lambda samples: np.sqrt(np.arange(1, N + 1))
         * (cumavg(samples) - 0.5) / cumstd(samples), '$\sqrt{n} \\frac{\overline{X} - p}{\sqrt{S^2}}$')
    ]
    draw(sps.bernoulli(0.5).rvs, t_estimators, 'Выборка из $Bern(0.5)$')


def part3():
    t_estimators = [(lambda samples: np.sqrt(np.arange(1, N + 1))
                     * cummedian_matrix(samples) / (pi / 2), '$\sqrt{n} \\frac{\widehat\mu}{\pi/2}$')]
    draw(sps.cauchy.rvs, t_estimators, 'Выборка из $Cauchy$', (0, 0.05))
```

In [6]:
```
%time part1()
```

```
/usr/lib/python3.6/site-packages/ipykernel_launcher.py:18: RuntimeWarning: divide by zero encountered in tru
e_divide
```

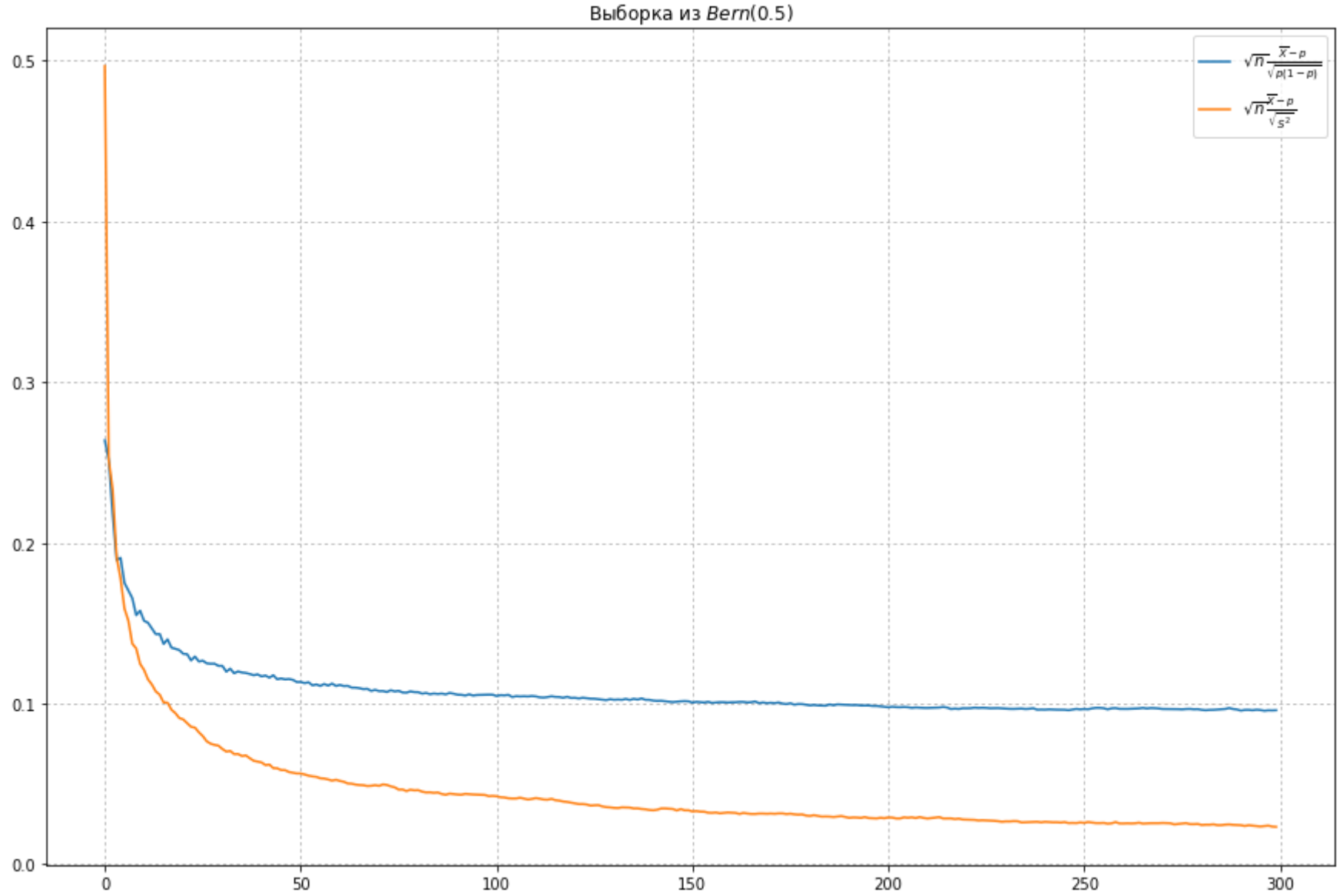



```
CPU times: user 46.3 s, sys: 2.98 s, total: 49.3 s
```

```
Wall time: 1min 51s
```

In [7]:
```
%time part2()
```

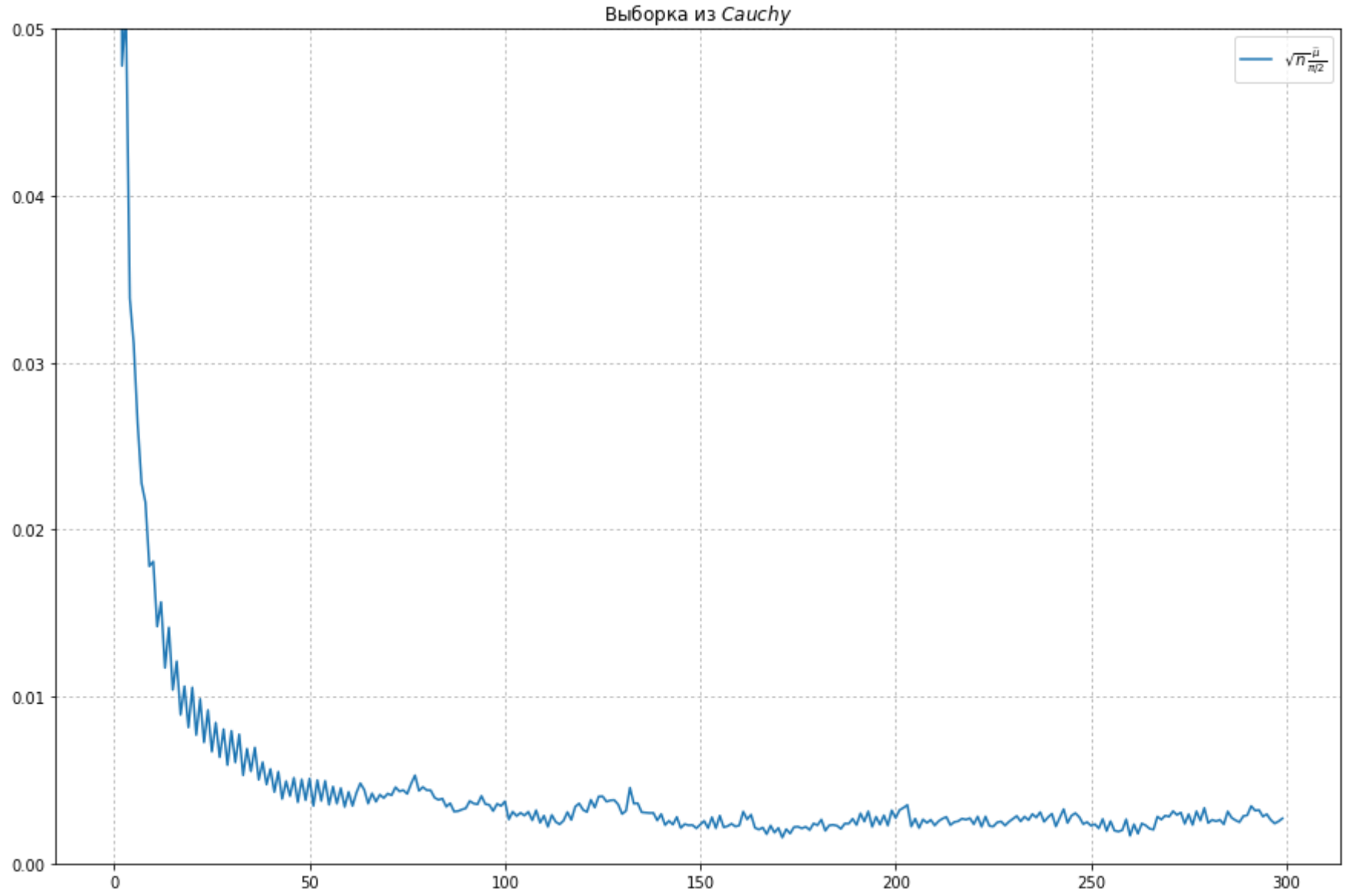
```
/usr/lib/python3.6/site-packages/ipykernel_launcher.py:26: RuntimeWarning: divide by zero encountered in tru
e_divide
```




```
CPU times: user 34.9 s, sys: 3.1 s, total: 38 s
```

```
Wall time: 1min 5s
```

In [8]:
```
%time part3()
```




```
CPU times: user 2min 22s, sys: 1.34 s, total: 2min 24s
Wall time: 2min 23s
```

Вывод

- Для выборки из $N(0, 1)$ лучшее приближение достигается со статистикой $T = \sqrt{n} \cdot \overline{X}$
- Для выборки из $Bern(0.5)$ лучшее приближение достигается со статистикой $T_n = \sqrt{n}\frac{\overline{X} - p}{\sqrt{S^2}}$

  Во всех случаях приближением нормальноым распределением можно пользоваться, начиная с выборки размера 100

Задача 5*. Проведите исследование аналогичное задаче 4 для статистик из задачи 2.

Задача 6. Реализуйте следующую функцию для выборки из нормального распределения

In [9]:
```
def get_supremum(sample, cdf):
    """По выборке строит эмпирическую функцию распределения ecdf, считает точное значение статистики sup|ecdf
    eps = 10 ** -7
    ecdf = ECDF(sample)
    return max([np.abs(ecdf(sample + offset) - cdf(sample)).max() for offset in (-eps, 0)])


def normal_summary(sample, name=None):
    α = 0.95

    n = len(sample)
    mean = sample.mean()
    std = sample.std()
    distribution = sps.norm(loc=mean, scale=std)

    if name is not None:
        print('\tSummary of {}:'.format(name))
    print('size: %d' % n)
    print('sample mean: %.2f' % mean)
    print('sample median: %.2f' % np.median(sample))
    print('sample std: %.2f' % std)  # стандартное отклонение == корень из дисперсии
    print('0.95 confidence interval: (%.2f, %.2f)' %
          (distribution.ppf((1 - α) / 2),
           distribution.ppf((1 + α) / 2)))
    # значение статистики из теоремы Колмогорова-Смирнова,
    # взяв в качестве F функцию распределения нормального
    # распределения с оценёнными выше параметрами
    print('KS-stat: %.3f' % get_supremum(sample, distribution.cdf))
    print()
```

Протестируйте функцию на выборках из нормального распределения и на выборках из других распределений. Какой вывод можно сделать о поведении статистики Колмогорова-Смирнова?

Чем больше размер выборки, тем меньше значение статистики критерия:

In [10]:
```
normal_summary(sps.norm.rvs(size=10), 'N(0, 1)')
normal_summary(sps.norm.rvs(size=100), 'N(0, 1)')
normal_summary(sps.norm.rvs(size=178), 'N(0, 1)')
normal_summary(sps.norm.rvs(size=1000), 'N(0, 1)')
```

```
        Summary of N(0, 1):
size: 10
sample mean: 0.33
sample median: 0.20
sample std: 0.64
0.95 confidence interval: (-0.92, 1.58)
KS-stat: 0.169

        Summary of N(0, 1):
size: 100
sample mean: -0.02
sample median: -0.18
sample std: 1.01
0.95 confidence interval: (-2.00, 1.97)
KS-stat: 0.085

        Summary of N(0, 1):
size: 178
sample mean: 0.04
sample median: 0.06
sample std: 1.02
0.95 confidence interval: (-1.96, 2.03)
KS-stat: 0.044

        Summary of N(0, 1):
size: 1000
sample mean: -0.03
sample median: -0.03
sample std: 0.96
0.95 confidence interval: (-1.91, 1.86)
KS-stat: 0.015
```

Для распределений $N(\theta, 1)$ значение статистики критерия от θ не зависит:

In [11]:
```
normal_summary(sps.norm(loc=0).rvs(size=100), 'N(0, 1)')
normal_summary(sps.norm(loc=1).rvs(size=100), 'N(1, 1)')
normal_summary(sps.norm(loc=10).rvs(size=100), 'N(10, 1)')
normal_summary(sps.norm(loc=100).rvs(size=100), 'N(100, 1)')
normal_summary(sps.norm(loc=-100).rvs(size=100), 'N(-100, 1)')
```

```
        Summary of N(0, 1):
size: 100
sample mean: -0.09
sample median: 0.04
sample std: 0.89
0.95 confidence interval: (-1.84, 1.67)
KS-stat: 0.071

        Summary of N(1, 1):
size: 100
sample mean: 0.90
sample median: 0.90
sample std: 1.06
0.95 confidence interval: (-1.17, 2.97)
KS-stat: 0.065

        Summary of N(10, 1):
size: 100
sample mean: 9.89
sample median: 9.84
sample std: 0.94
0.95 confidence interval: (8.06, 11.72)
KS-stat: 0.048

        Summary of N(100, 1):
size: 100
sample mean: 100.02
sample median: 100.11
sample std: 1.07
0.95 confidence interval: (97.92, 102.13)
KS-stat: 0.062

        Summary of N(-100, 1):
size: 100
sample mean: -100.10
```

```
sample median: -100.19
sample std: 0.97
0.95 confidence interval: (-102.01, -98.20)
KS-stat: 0.054
```

Для распределений $N(0, \sigma^2)$ значение статистики критерия от σ^2 не зависит:

In [12]:
```
normal_summary(sps.norm(scale=1).rvs(size=100), 'N(0, 1)')
normal_summary(sps.norm(scale=2).rvs(size=100), 'N(0, 4)')
normal_summary(sps.norm(scale=10).rvs(size=100), 'N(0, 100)')
normal_summary(sps.norm(scale=100).rvs(size=100), 'N(0, 10000)')
```

```
        Summary of N(0, 1):
size: 100
sample mean: 0.07
sample median: -0.01
sample std: 0.91
0.95 confidence interval: (-1.72, 1.86)
KS-stat: 0.068

        Summary of N(0, 4):
size: 100
sample mean: -0.14
sample median: -0.05
sample std: 2.20
0.95 confidence interval: (-4.44, 4.17)
KS-stat: 0.059

        Summary of N(0, 100):
size: 100
sample mean: 1.39
sample median: 1.34
sample std: 8.69
0.95 confidence interval: (-15.64, 18.41)
KS-stat: 0.059

        Summary of N(0, 10000):
size: 100
sample mean: -5.02
sample median: -1.16
sample std: 114.41
0.95 confidence interval: (-229.26, 219.21)
KS-stat: 0.059
```

Таким образом делаем вывод, что для выборки из нормального распределения значение статистики критерия зависит только от размера выборки.

На других распределениях значение критерия больше, чем значения критерия на выборке такого же размера из нормального распределения:

In [13]:
```
normal_summary(sps.uniform.rvs(size=10), 'U(0, 1)')
normal_summary(sps.expon.rvs(size=10), 'Exp(1)')
normal_summary(sps.uniform.rvs(size=100), 'U(0, 1)')
normal_summary(sps.expon.rvs(size=100), 'Exp(1)')
normal_summary(sps.uniform.rvs(size=1000), 'U(0, 1)')
normal_summary(sps.expon.rvs(size=1000), 'Exp(1)')
```

```
        Summary of U(0, 1):
size: 10
sample mean: 0.45
sample median: 0.40
sample std: 0.30
0.95 confidence interval: (-0.14, 1.03)
KS-stat: 0.177

        Summary of Exp(1):
size: 10
sample mean: 0.95
sample median: 0.44
sample std: 1.08
0.95 confidence interval: (-1.17, 3.07)
KS-stat: 0.249

        Summary of U(0, 1):
size: 100
sample mean: 0.50
sample median: 0.53
sample std: 0.30
0.95 confidence interval: (-0.09, 1.09)
KS-stat: 0.103

        Summary of Exp(1):
size: 100
sample mean: 1.12
sample median: 0.92
sample std: 1.06
0.95 confidence interval: (-0.95, 3.19)
KS-stat: 0.155

        Summary of U(0, 1):
size: 1000
```

```
sample mean: 0.51
sample median: 0.50
sample std: 0.29
0.95 confidence interval: (-0.06, 1.08)
KS-stat: 0.059

        Summary of Exp(1):
size: 1000
sample mean: 0.99
sample median: 0.72
sample std: 0.97
0.95 confidence interval: (-0.91, 2.88)
KS-stat: 0.154
```

Скачайте данные http://archive.ics.uci.edu/ml/datasets/Wine (http://archive.ics.uci.edu/ml/datasets/Wine), файл `wine.data`. Что вы можете сказать про столбцы 1, 4, 8 (нумерация с нуля), соответсвующие 'Alcohol', 'Alcalinity of ash', 'Nonflavanoid phenols'?

In [14]:
```
normal_summary(sps.norm(loc=13, scale=1).rvs(size=178), 'N(13, 1)')
normal_summary(sps.norm(loc=19.5, scale=3.33).rvs(size=178), 'N(19.5, 3.33)')
normal_summary(sps.norm(loc=0.35, scale=0.1).rvs(size=178), 'N(0.35, 0.1)')
```

```
        Summary of N(13, 1):
size: 178
sample mean: 12.99
sample median: 13.01
sample std: 0.99
0.95 confidence interval: (11.06, 14.92)
KS-stat: 0.036

        Summary of N(19.5, 3.33):
size: 178
sample mean: 19.53
sample median: 19.57
sample std: 3.15
0.95 confidence interval: (13.36, 25.71)
KS-stat: 0.035

        Summary of N(0.35, 0.1):
size: 178
sample mean: 0.36
sample median: 0.35
sample std: 0.10
0.95 confidence interval: (0.17, 0.54)
KS-stat: 0.033
```

In [15]:
```
data = np.loadtxt('wine.data', delimiter=',')
normal_summary(data[:, 1], 'alcohol')
normal_summary(data[:, 4], 'alcalinity of ash')
normal_summary(data[:, 8], 'nonflavanoid phenols')
```

```
        Summary of alcohol:
size: 178
sample mean: 13.00
sample median: 13.05
sample std: 0.81
0.95 confidence interval: (11.41, 14.59)
KS-stat: 0.069

        Summary of alcalinity of ash:
size: 178
sample mean: 19.49
sample median: 19.50
sample std: 3.33
0.95 confidence interval: (12.97, 26.02)
KS-stat: 0.063

        Summary of nonflavanoid phenols:
size: 178
sample mean: 0.36
sample median: 0.34
sample std: 0.12
0.95 confidence interval: (0.12, 0.61)
KS-stat: 0.115
```

Вывод: на основании статистики критерия Колмогорова-Смирнова можно сказать, что столбец 8 ('nonflavanoid phenols') не является выборкой из нормального распределения, а вот столбцы 1 и 4 ('alcohol' и 'alcalinity of ash') вполне могут быть выборками из распределений $N(13, \frac{4}{5})$ и $N(19.5, \frac{10}{3})$ соответсвтенно.

2. Байесовские методы

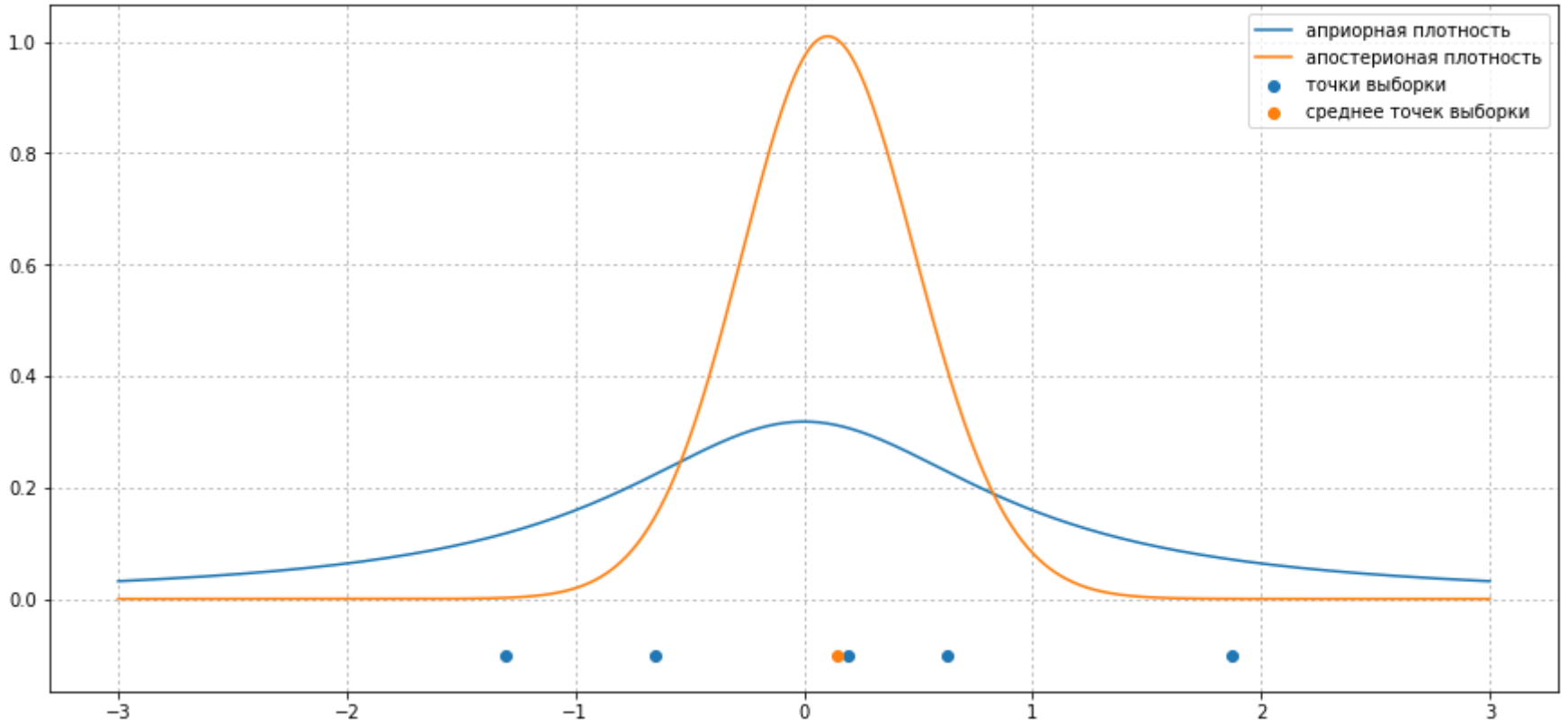
Задача 7. Пусть $X_1, \ldots, X_n \sim \mathcal{N}(\theta, 1)$ и θ имеет априорное распределение Коши. Как было сказано на лекции, аналитически интеграл в знаменателе формулы Байеса посчитать не удастся. Однако, поскольку в данном случае параметр один, можно его посчитать с помощью приближенного интегрирования.

В качестве метода приближенного интегрирования можно использовать следующую модификацию известного метода Монте-Карло. В качестве оценки интеграла $\int\limits_{\mathbb{R}} f(x)p(x)dx$, где $p(x)$ --- некоторая плотность, можно взять величину $\sum\limits_{j=1}^{k} f(Y_i)$, где $Y_1, \ldots, Y_k$ --- сгенерированная выборка из распределения, имеющего плотность $p(x)$.

Сгенерируйте выборку размера 5 из стандартного нормального распределения. Посчитайте для нее C --- знаменатель в формуле Байеса. Какой размер вспомогательной выборки в методе приближенного интегрирования необходим, чтобы с большой точностью посчитать значение C?

Нарисуйте график плотности апостериорного распределения. Посчитайте математическое ожидание по апостериорному распределению.

In [16]:
```
xs = sps.norm.rvs(size=5)
ys = sps.cauchy.rvs(size=1000)
знаменатель = sps.norm(loc=ys[:, np.newaxis]).pdf(xs[np.newaxis, :]).prod(axis=1).mean()

plt.figure(figsize=(15, 7))
grid = np.linspace(-3, 3, 1000)
plt.plot(grid, sps.cauchy.pdf(grid), label='априорная плотность')
plt.plot(grid, sps.cauchy.pdf(grid) * sps.norm(loc=grid[:, np.newaxis]).pdf(xs[np.newaxis, :]).prod(axis=1) /
plt.scatter(xs, [-0.1] * 5, label='точки выборки')
plt.scatter(xs.mean(), -0.1, label='среднее точек выборки')
plt.legend()
plt.grid(ls=':')
plt.show()
```



Мне кажется, что 1000 --- достаточный размер для вспомогательный выборки, в принципе можно было и больше взять.

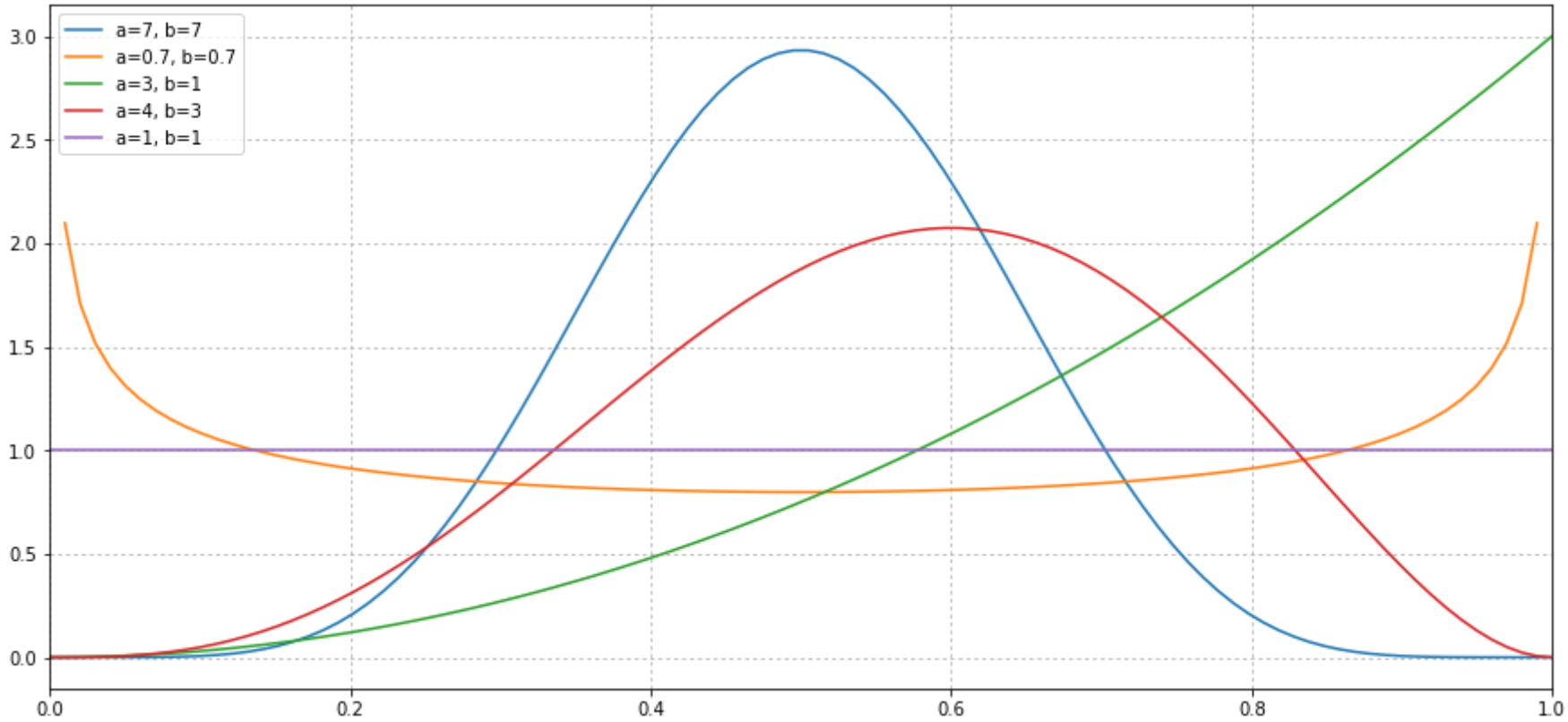
Задача 8. Рассмотрим схему испытаний Бернулли (т.е. броски монет) с вероятностью успеха p.

Постройте несколько графиков сопряженного распределения для разных параметров и охарактеризуйте, как значения его параметров соотносятся с априорными знаниями о монете. Это могут быть, например, знания вида

- монета скорее честная (при таком априорном распределении наиболее вероятны значения p в окрестности 0.5)
- монета скорее нечестная, перевес неизвестен (наименее вероятны значения p в окрестности 0.5)
- монета скорее нечестная, перевес в сторону герба (наиболее вероятны значения p в окрестности 1)
- монета скорее честная, либо с небольшим перекосом вправо (наиболее вероятны значения p в окрестности ~0.6)
- ничего не известно (все значения равновероятны)

Для каждого случая из перечисленных выше постройте график плотности сопряженного распределения (на одной фигуре).

In [17]:
```
plt.figure(figsize=(15, 7))
grid = np.linspace(0, 1, 100)
parameters = [
    [7, 7],
    [0.7, 0.7],
    [3, 1],
    [4, 3],
    [1, 1]
]
for a, b in parameters:
    plt.plot(grid, sps.beta(a=a, b=b).pdf(grid), label='a={}, b={}'.format(a, b))
plt.legend()
plt.grid(ls=':')
plt.xlim((0, 1))
plt.show()
```



- $a = 7, b = 7$: монета скорее честная
- $a = 0.7, b = 0.7$: монета скорее нечестная, перевес неизвестен
- $a = 3, b = 1$: монета скорее нечестная, перевес в сторону герба
- $a = 4, b = 3$: монета скорее честная, либо с небольшим перекосом вправо
- $a = 1, b = 1$: ничего не известно

Ниже приведена реализация некоторых вспомогательных функций.

In [27]:
```
def draw_posteriori(grid, distr_class, post_params, xlim=None):
    ''' Рисует серию графиков апостериорных плотностей.
        grid --- сетка для построения графика
        distr_class --- класс распределений из scipy.stats
        post_params --- параметры апостериорных распределений
                        shape=(размер выборки, кол-во параметров)
    '''

    size = post_params.shape[0] - 1

    plt.figure(figsize=(12, 7))
    for n in range(size+1):
        plt.plot(grid,
                 distr_class(post_params[n]).pdf(grid) if np.isscalar(post_params[n])
                 else distr_class(*post_params[n]).pdf(grid),
                 label='n={}: {}'.format(n, post_params[n]),
                 lw=2.5,
                 color=(1-n/size, n/size, 0))
    plt.grid(ls=':')
    plt.legend()
    plt.xlim(xlim)
    plt.show()


def draw_estimations(ml, distr_class, post_params, confint=True, ylim=None, xlim=None, title=''):
    ''' Рисует графики байесовской оценки (м.о. и дов. инт.) и ОМП.
        ml --- Оценка максимального правдоподобия для 1 <= n <= len(sample)
        distr_class --- класс распределений из scipy.stats
        post_params --- параметры апостериорных распределений
                        shape=(размер выборки+1, кол-во параметров)
    '''

    size = len(ml)
    distrs = []
    for n in range(size + 1):
        distrs.append(distr_class(post_params[n]) if np.isscalar(post_params[n])
                      else distr_class(*post_params[n]))

    plt.figure(figsize=(12, 4))
    plt.plot(np.arange(size + 1), [d.mean() for d in distrs], label='Bayes', lw=1.5)
    plt.fill_between(np.arange(size + 1), [d.ppf(0.975) for d in distrs],
```

```
                    [d.ppf(0.025) for d in distrs], alpha=0.1)
    plt.plot(np.arange(size) + 1, ml, label='ML', lw=1.5)
    plt.grid(ls=':')
    plt.ylim(ylim)
    plt.xlim(xlim)
    plt.legend()
    plt.title(title)
    plt.show()
```

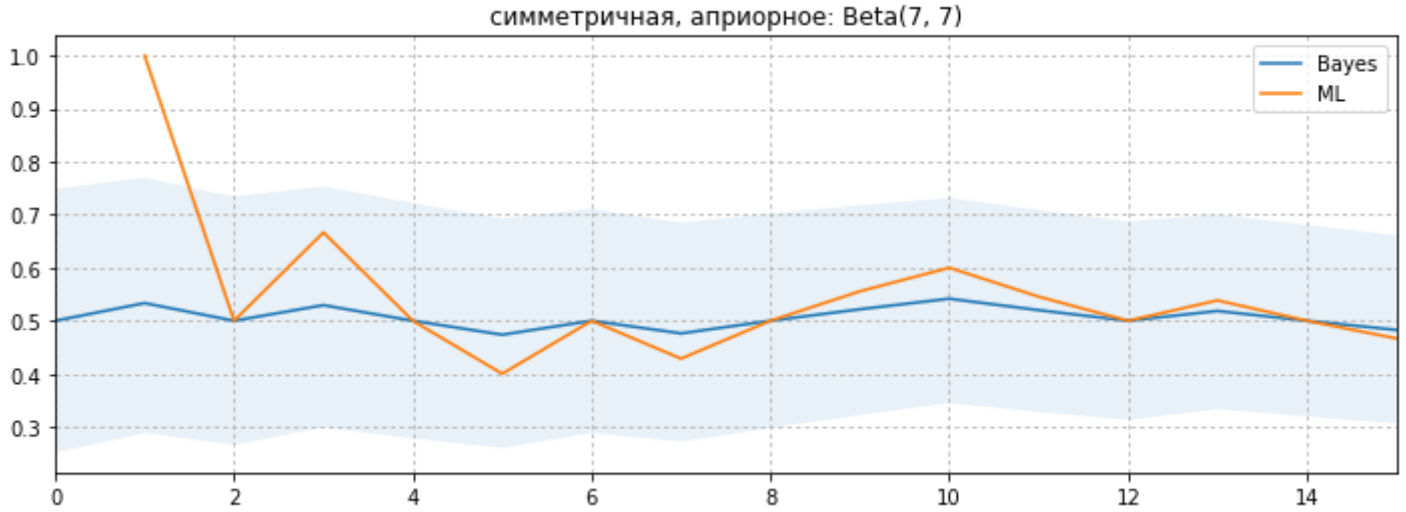
Реализуйте следующую функцию

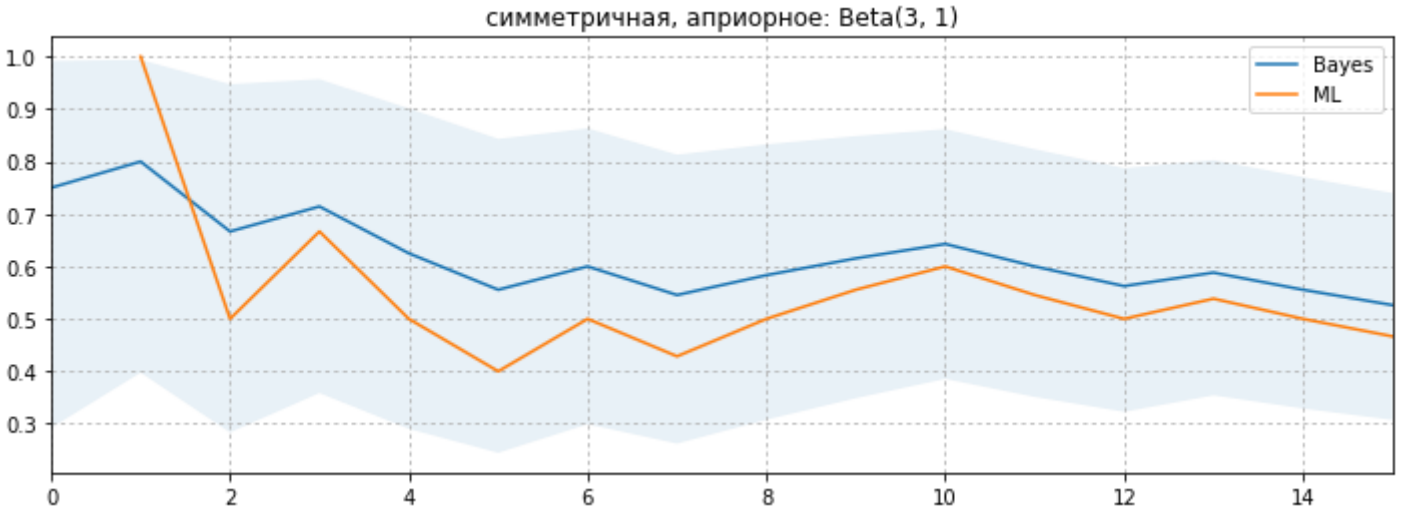
In [19]:
```
def bern_posterior_params(sample, a, b):
    ''' Возвращает параметры апостериорного распределения для всех 0 <= n <= len(sample).
        a, b --- параметры априорного распределения.
    '''
    # Beta(a + \sum X_i, b + n - \sum X_i) --- апостериорное распределение для Bern(p)
    a_aposterior = a + sample.cumsum()
    b_aposterior = b + np.arange(1, len(sample) + 1) - sample.cumsum()
    return np.hstack((a, a_aposterior)), np.hstack((b, b_aposterior))
```

Проведите по 15 бросков симметричной и несимметричной монет (можно сгенерировать) и рассмотрите для каждой из них два случая --- параметры априорного распределения подобраны правильно или неправильно. Постройте графики, воспользовавшись функциями `draw_posteriori` и `draw_estimations`.

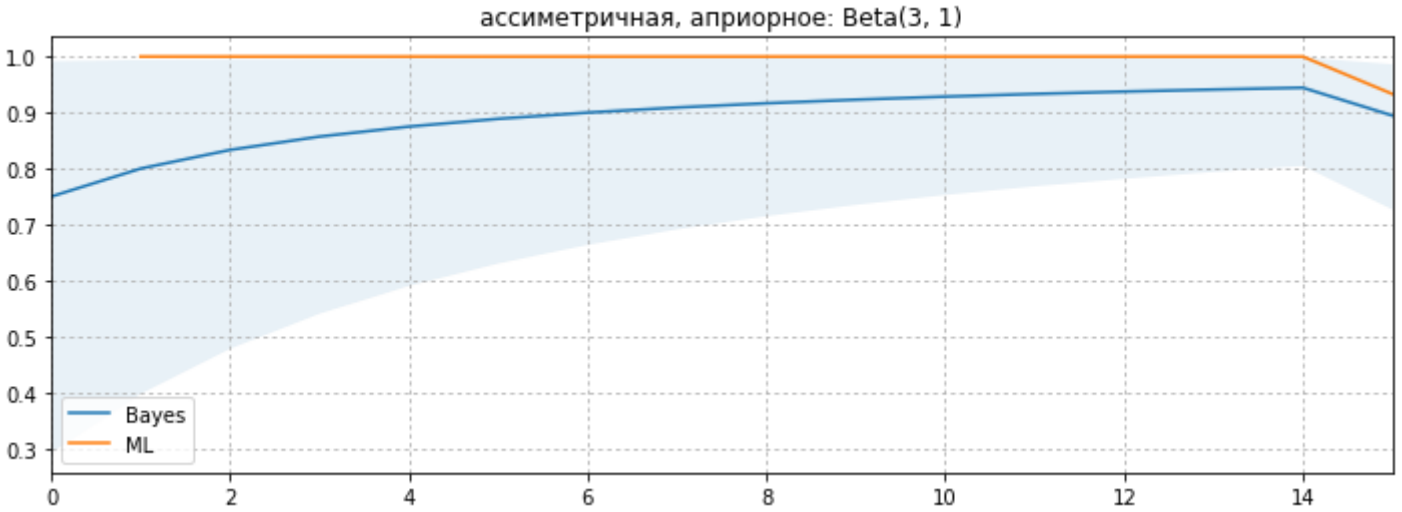
Сделайте вывод. Что можно сказать про зависимость от параметров априорного распределения? Сравните байесовские оценки с оценкой максимального правдоподобия.

In [35]:
```
sample_symmetric = sps.bernoulli(p=0.5).rvs(size=15)
sample_asymmetric = sps.bernoulli(p=0.9).rvs(size=15)
grid = np.linspace(0, 1)
for sample, title in [(sample_symmetric, 'симметричная'), (sample_asymmetric, 'ассиметричная')]:
    for a, b in [(7, 7), (3, 1)]:
        draw_estimations(bern_cumlikelihood(sample),
                         sps.beta,
                         np.column_stack(bern_posterior_params(sample, a, b)),
                         title='{}, априорное: Beta({}, {})'.format(title, a, b), xlim=(0, 15))
```




симметричная, априорное: Beta(3, 1)
Bayes
ML
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
2
4
6
8
10
12
14


ассиметричная, априорное: Beta(7, 7)
Bayes
ML
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
2
4
6
8
10
12
14



Вывод: матожидание апостериорного распределения более точно оценивает параметр, чем ОМП, если правильно подобрано априорное распределение. Если априорное распределение выбрано неправильно, то, конечно, ОМП оказывается лучше.

Задача 9*. Один экзаменатор на экзамене по теории вероятностей при выставлении оценки студенту пользуется следующей схемой. В течении экзамена экзаменатор задает студенту некоторое количество вопросов, получая тем самым выборку $X_1, \ldots, X_n \sim Bern(p)$ --- индикаторы того, что студент на вопрос ответил правильно. При этом сначала он подбирает некоторое априорное распределение на основе его знаний о студенте к моменту начала ответа. После каждого ответа студента экзаменатор вычисляет апостериорное распределение и строит байесовский доверительный интервал для p уровня доверия 0.95. Если после очередного ответа студента доверительный интервал содержит лишь одно число $i/10$, где $i \in \{0, \ldots, 10\}$, то экзаменатор выставляет студенту оценку $i + 1$.

Ответьте на следующие вопросы:

- Квантили какого уровня нужно выбирать экзаменатору при построении доверительного интервала, чтобы задавать студенту как можно меньше вопросов? Какие оценки будет выставлять экзаменатор в таком случае?
- Как зависит оценка студента и среднее количество заданных вопросов у различных студентов (по уровню знаний) при различных априорных представлений экзаметора?
- Нужно ли дружить с таким экзаменатором?

Задача 10. Проведите исследование, аналогичное задаче 8 для выборок из распределений

- $\mathcal{N}(\theta, 1)$
- $Exp(\theta)$

Задача 11*. Проведите исследование, аналогичное задаче 8 для выборки из распределения $\mathcal{N}(\theta_1, \theta_2^{-1})$.