

Математическая статистика

Практическое задание 4

В данном задании предлагается провести некоторое исследование доверительных интервалов и байесовских 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 \backslash^* . 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 \backslash^* - 3 балла
- Задача 3 \backslash^* - 3 балла
- Задача 4 - 5 баллов
- Задача 5 \backslash^* - 2 балла
- Задача 6 - 4 балла
- Задача 7 - 1 балл
- Задача 8 - 3 балла
- Задача 9 \backslash^* - 5 баллов
- Задача 10 - 5 баллов
- Задача 11 \backslash^* - 3 балла

In [128]:

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

```
%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 [101]:

```
sample_size = 100#size of sample
alpha= 0.95

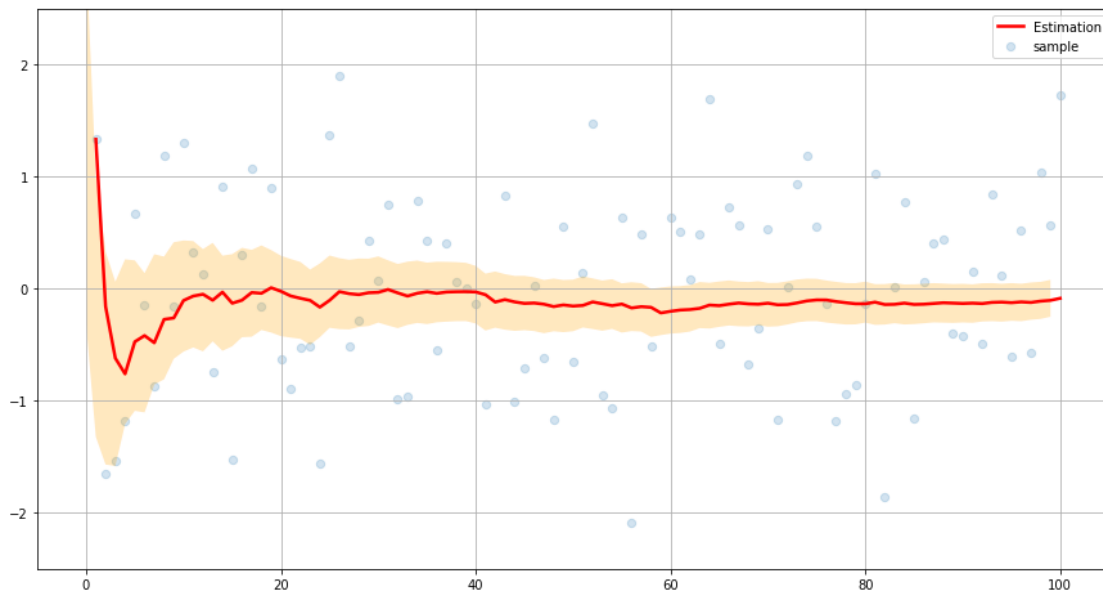
def draw_confidence_interval(left, # левая граница интервала
                             right, # правая граница интервала
                             estimation=None, # если задана, то рисуется графи
к оценки
                             sample=None, # если задано, то рисуются точки выб
орки
                             ylim=(None, None)): # ограничение по оси y
    plt.figure(figsize=(15, 8))
    if (not(estimation is None)):
        plt.plot(np.arange(1, sample_size + 1), estimation, color='red', linewi
dth=2.5,
                 label='Estimation') # Linewidth - толщина линии
    if (not(sample is None)):
        plt.scatter(np.arange(1, sample_size + 1), sample, alpha=0.2, s=40, lab
el='sample')
    plt.fill_between(range(sample_size), left, right, facecolor='orange', alpha
= 0.25)
    plt.legend()
    plt.ylim(ylim)
    plt.grid() # добавляем сетку
    plt.show()
```

$1.\mathcal{N}(0, 1)$

Доверительный интервал для θ это $(\bar{X} - \frac{z_{\frac{1+\alpha}{2}}}{\sqrt{n}}, \bar{X} + \frac{z_{\frac{1+\alpha}{2}}}{\sqrt{n}})$

In [102]:

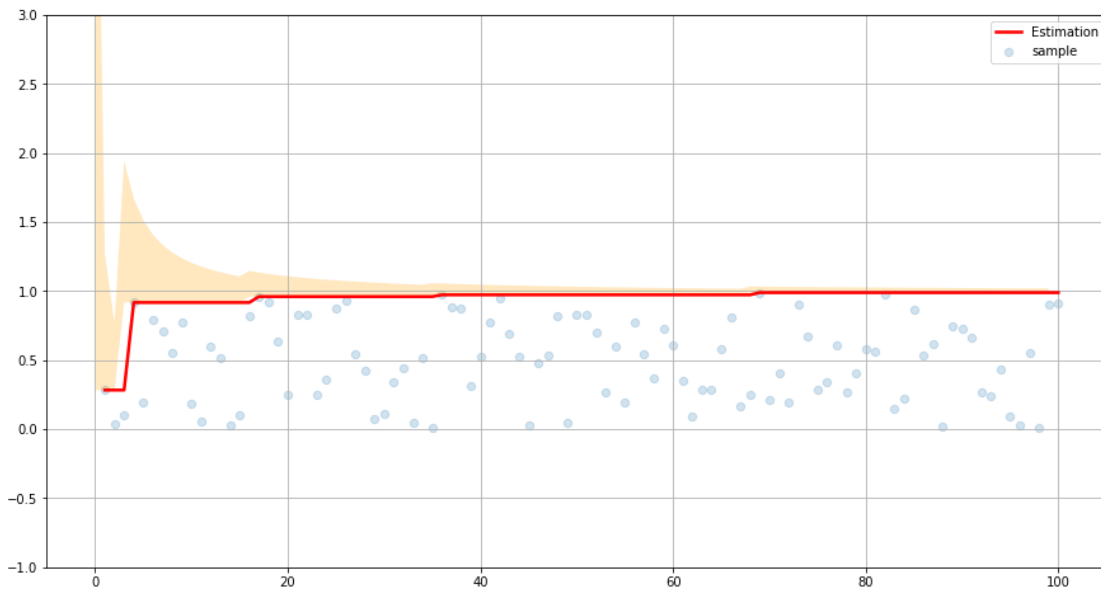
```
sample = sps.norm.rvs(loc=0, scale=1, size=sample_size)
estimation = sample.cumsum() / np.arange(1, sample_size + 1)
quantile = sps.norm.ppf(alpha)/np.sqrt(np.arange(1, sample_size + 1))
draw_confidence_interval(estimation - quantile, estimation + quantile, estimation,
                          sample, (-2.5, 2.5))
```



2 Доверительный интервал уровня α для $U(0, \theta)$ - это $(X_{(n)}, \frac{X_{(n)}}{\sqrt[n]{1-\alpha}})$.

In [10]:

```
sample = sps.uniform.rvs(loc=0, scale=1, size=sample_size)
estimation = np.array([np.max(sample[:i+1]) for i in range(sample_size)])
left = estimation
right = np.array([left[i]*1./((1-alpha)**(1/(i+1))) for i in range(sample_size)])
draw_confidence_interval(left, right, estimation, sample, (-1, 3))
```

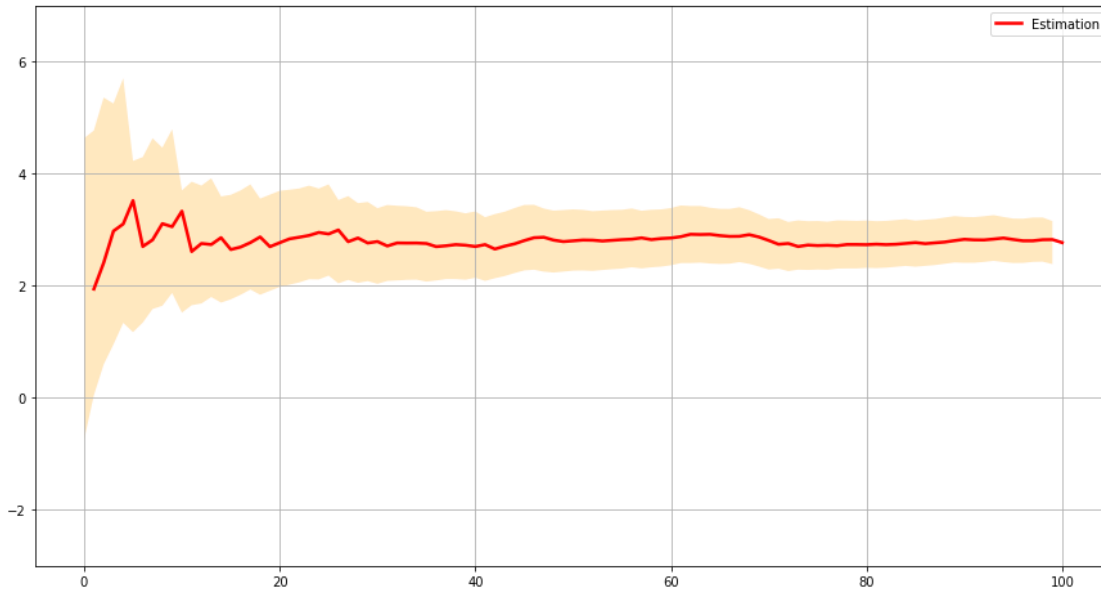


3. Асимптотический доверительный интервал уровня α для $\Gamma(\theta, \lambda)$ - это

$\left(\frac{\lambda + z \frac{1-\alpha}{2} \sqrt{\frac{\lambda}{n}}}{\bar{X}}, \frac{\lambda + z \frac{1+\alpha}{2} \sqrt{\frac{\lambda}{n}}}{\bar{X}} \right)$, где z - квантиль нормального распределения соответствующего уровня.

In [103]:

```
sample = sps.gamma.rvs(size=sample_size, a=2, scale=1/3)
estimation = sample.cumsum() / np.arange(1., sample_size + 1)
left = (2 + sps.norm.ppf(0.5 - alpha / 2) * ((2 / np.arange(1.,
    sample_size + 1.)) ** 0.5)) / estimation # нижняя оценка
right = (2 + sps.norm.ppf(0.5 + alpha / 2) * ((2 / np.arange(1.,
    sample_size + 1.)) ** 0.5)) / estimation # верхняя оценка
draw_confidence_interval(left, right, 2/estimation, None, (-3, 7))
```



4. Асимптотический доверительный интервал уровня α для $Cauchy(\theta, 1)$ -

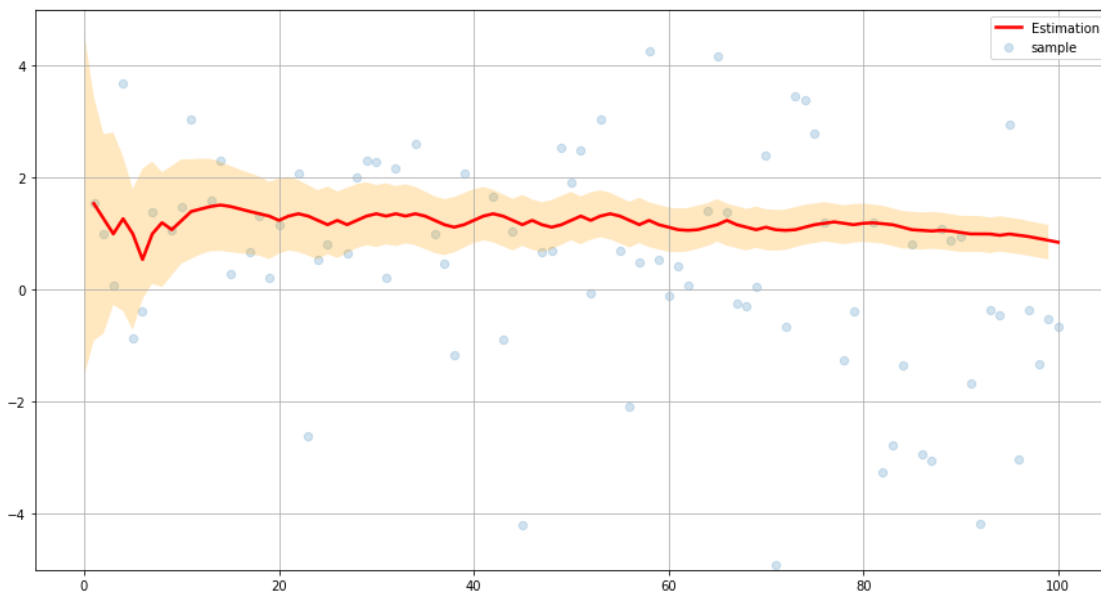
$(\mu - \frac{\pi z \frac{1+\alpha}{2}}{2\sqrt{n}}, \mu - \frac{\pi z \frac{1-\alpha}{2}}{2\sqrt{n}})$, где z - квантиль нормального распределения, μ - медиана

In [39]:

```
sample = sps.cauchy.rvs(size=sample_size, loc=1, scale=1) # генерируем выборка
z = sps.norm.ppf(0.5 + alpha / 2) # считаем квантиль
# Считаем выборочные медианы
med_ar = np.zeros(sample_size)

for n in range(1, sample_size+1):
    med_ar[n-1] = np.median(sample[:n])

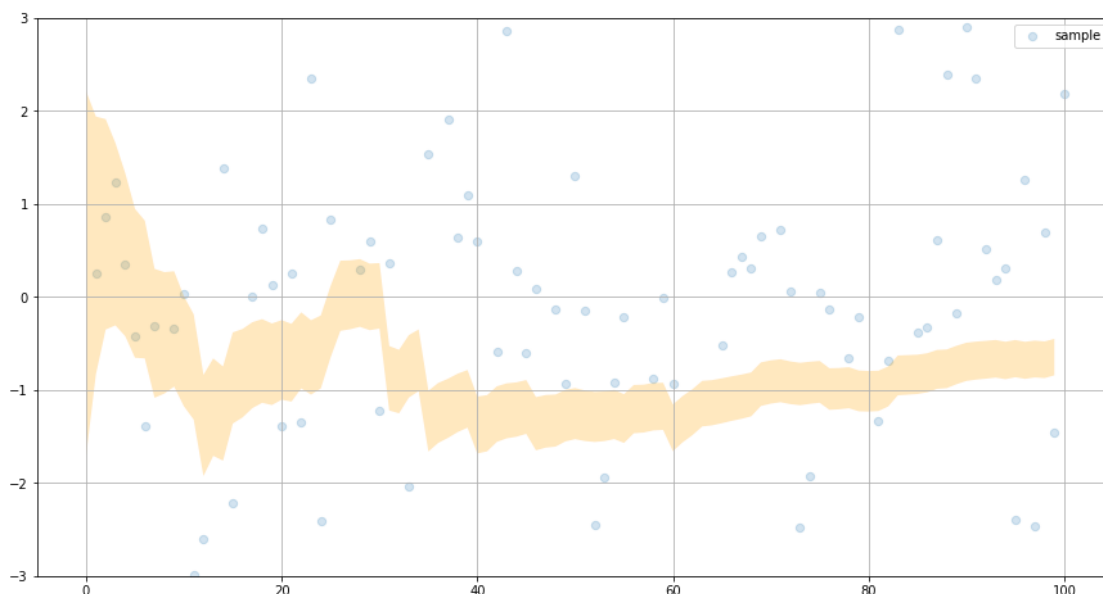
left = med_ar - np.pi * sps.norm.ppf(0.5 + alpha/ 2) * 0.5 * ((1 / np.arange(1.
,
                                sample_size + 1.)) ** 0.5) # нижняя оценка
right = med_ar - np.pi * sps.norm.ppf(0.5 - alpha/ 2) * 0.5 * ((1 / np.arange(
1.,
                                sample_size + 1.)) ** 0.5) # верхняя оценка
estimation = med_ar
draw_confidence_interval(left, right, estimation, sample, (-5, 5))
```



5. Доверительный интервал для θ это $(\bar{X} - \frac{z_{\frac{1+\alpha}{2}}}{\sqrt{n}}, \bar{X} + \frac{z_{\frac{1+\alpha}{2}}}{\sqrt{n}})$

In [58]:

```
sample = sps.cauchy.rvs(size=sample_size)
mean_ar = []
for i in range(1, sample_size+1):
    mean_ar.append(np.mean(sample[:i]))
mean_ar = np.array(mean_ar)
left = mean_ar - sps.norm.ppf((1 + alpha) / 2) / np.sqrt(np.arange(1, sample_size+1))
right = mean_ar + sps.norm.ppf((1 + alpha) / 2) / np.sqrt(np.arange(1, sample_size+1))
draw_confidence_interval(left, right, None, sample, (-3, 3))
```



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?

Вывод: Истинное значение параметра в большом числе случаев попадает в доверительный интервал.

В случае обычных доверительных интервалов так и должно происходить, поскольку с вероятностью не менее 0.95 значение параметра должно попасть в доверительный интервал уровня доверия 0.95.

В случае асимптотических интервалов все тоже хорошо сработало, хотя нашего размера выборки могло быть, в общем случае, не достаточно.

Исключение составило распределение Коши в модели $\mathcal{N}(\theta, 1)$, поскольку мы исходили из неправильных предположений при составлении интервала.

Длина интервала сокращается при увеличении размера выборки.

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

Рассмотрите следующие случаи

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

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. Не забудьте 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 [115]:

```
#воспользуемся теоремой Глибенко-Кантелли
def one_calculate(sample, ecdf, cdf):
    sample_size = len(sample)
    ecdf = np.hstack((0, ecdf(np.sort(sample))))
    cdf = cdf(np.sort(sample))
    return np.max(np.abs(np.hstack((cdf - ecdf[:sample_size],
                                     cdf - ecdf[1:])))))

def all_calculate(T_nk):
    ecdf_array = list(map(lambda sample: ECDF(sample), T_nk.T))
    return np.array(list(map(lambda n: one_calculate(T_nk.T[n], ecdf_array[n],
                                                       sps.norm.cdf), np.arange(0, N))))
```

$X_1, \dots, X_n \sim \mathcal{N}(0, 1)$. Рассмотреть $T = \sqrt{n} \cdot \bar{X}$ и $T = \sqrt{n} \cdot \bar{X} / \sqrt{S^2}$.

In [120]:

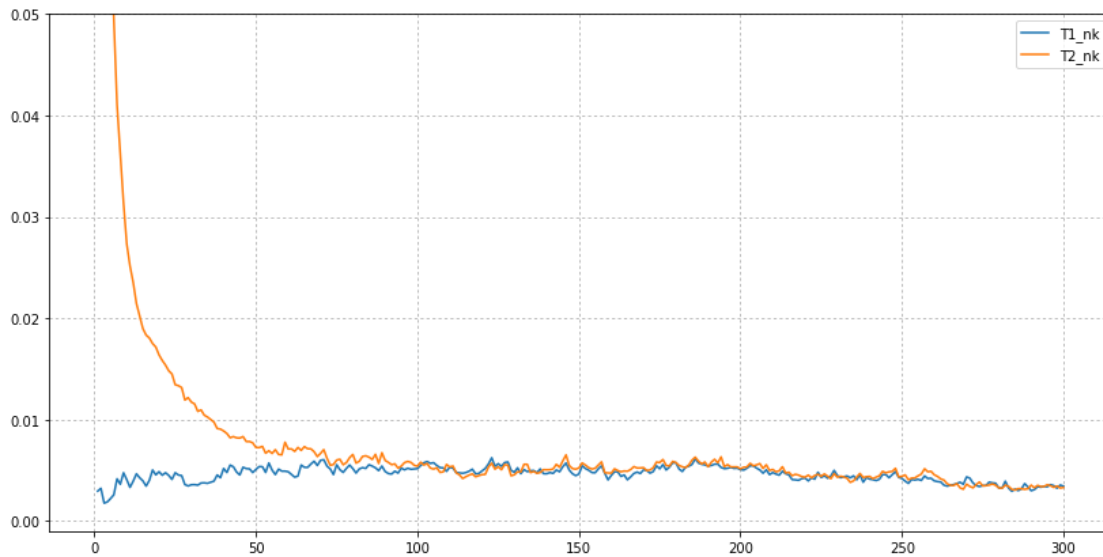
```
N = 300
K = 10**5
samples = sps.norm.rvs(size=(K, N))
T1 = lambda sample: np.cumsum(sample) / np.arange(1,
                                                    N + 1) * np.sqrt(np.arange(1, N + 1))
S_sqr = lambda sample: np.cumsum(sample ** 2) / np.arange(1,
                                                           N + 1) - (np.cumsum(sample) / np.arange(1, N + 1)) ** 2
T2 = lambda sample: T1(sample) / (np.sqrt(S_sqr(sample)))
T1_nk = np.array(list(map(lambda sample: T1(sample), samples)))
T2_nk = np.array(list(map(lambda sample: T2(sample), samples)))
```

In [63]:

```
from statsmodels.distributions.empirical_distribution import ECDF
```

In [117]:

```
%%time
plt.figure(figsize=(14, 7))
D1_n = calc_Glivenko_for_all(T1_nk)
D2_n = calc_Glivenko_for_all(T2_nk)
plt.plot(np.arange(1, N + 1), D1_n, label='T1_nk')
plt.plot(np.arange(1, N + 1), D2_n, label='T2_nk')
plt.ylim(-0.001, 0.05)
plt.legend(loc='best')
plt.grid(ls=':')
plt.show()
```



Wall time: 1min 10s

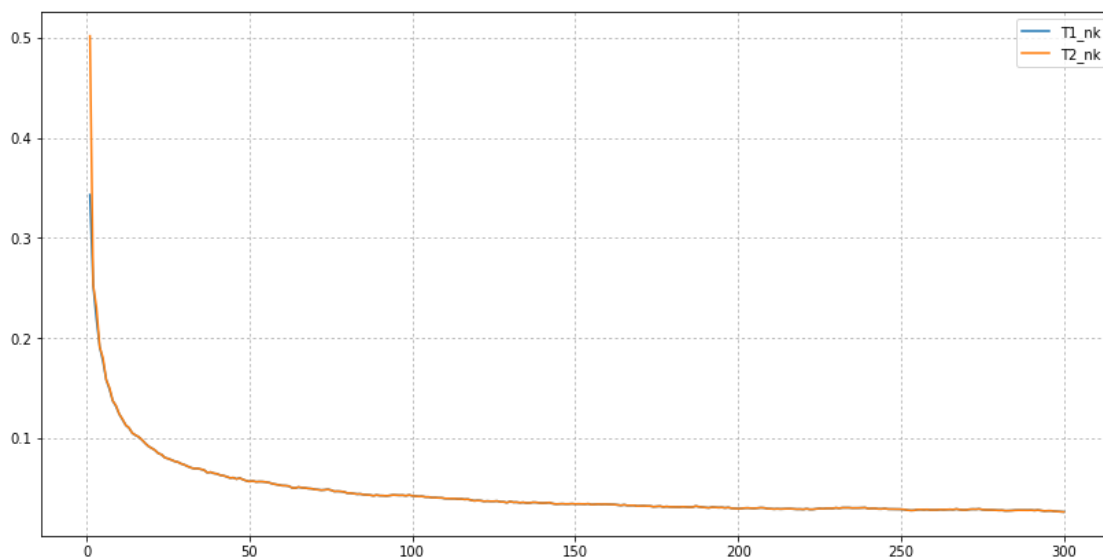
$X_1, \dots, X_n \sim \text{Bern}(p), p = 0.5$. Рассмотреть $T = \sqrt{n} \frac{\bar{X} - p}{\sqrt{p(1-p)}}$ и $T_n = \sqrt{n} \frac{\bar{X} - p}{\sqrt{S^2}}$.

In [121]:

```
p = 0.5
N, K = 300, 10**5
samples = sps.bernoulli(p).rvs(size=(K, N))
arange = np.arange(1, N + 1)
cummean = lambda sample: np.cumsum(sample) / arange
T1 = lambda sample: np.sqrt(arange) * (cummean(sample) - p) / np.sqrt(p * (1 - p))
S_sqr = lambda sample: np.cumsum(sample**2)/arange-(np.cumsum(sample)/arange)**2
T2 = lambda sample: np.sqrt(arange) * (cummean(sample)-p)/(np.sqrt(S_sqr(sample)))
T1_nk = np.array(list(map(lambda sample: T1(sample), samples)))
T2_nk = np.array(list(map(lambda sample: T2(sample), samples)))
```

In [119]:

```
%%time
plt.figure(figsize=(14, 7))
D1_n = calc_Glivenko_for_all(T1_nk)
D2_n = calc_Glivenko_for_all(T2_nk)
plt.plot(np.arange(1, N + 1), D1_n, label='T1_nk', linewidth=1.5)
plt.plot(np.arange(1, N + 1), D2_n, label='T2_nk', linewidth=1.5)
plt.legend(loc='best')
plt.grid(ls=':')
plt.show()
```



Wall time: 48.8 s

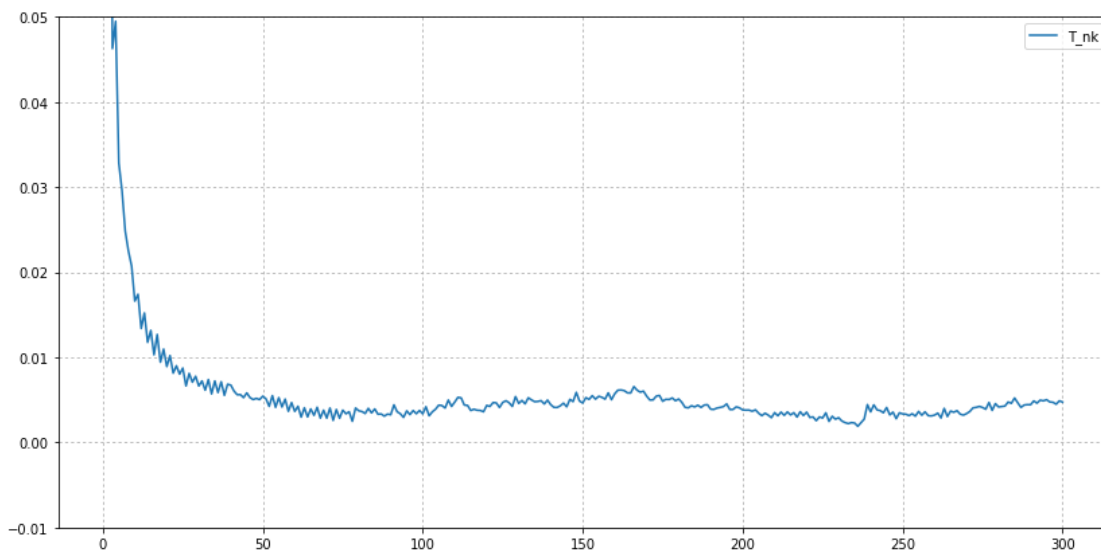
In [68]:

```
%%time
N = 300
K = 10**5
samples = sps.cauchy.rvs(size=(K, N))
T_nk = np.array(list(map(lambda n: np.sqrt(n) * np.median(samples[:, :n],
                                                             axis=1) / (np.pi / 2), range(1, N +
                                                             1)))))
```

Wall time: 4min 17s

In [69]:

```
plt.figure(figsize=(14, 7))
D_n = all_calculate(T_nk.T)
plt.plot(np.arange(1, N + 1), D_n, label='T_nk')
plt.legend()
plt.ylim(-0.01, 0.05)
plt.grid(ls=':')
plt.show()
```



Вывод:

Эмпирическая функция распределения для исследуемых нами статистик довольно хорошо приближается стандартным нормальным распределением, начиная с размера выборки близкому к 50, ибо значение статистики здесь меньше, чем 0.1. Чем больше размер выборки, тем ближе к нулю статистика. Это так же подтверждает теорема Гливенко-Кантелли.

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

Задача 6. Реализуйте следующую функцию для выборки из нормального распределения

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

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), соответствующие 'Alcohol', 'Alcalinity of ash', 'Nonflavanoid phenols'?

Посмотрим подробнее на 'Alcohol', 'Alcalinity of ash', 'Nonflavanoid phenols':

Очень похоже на нормальное распределение Тогда мы можем найти его параметры через нашу реализованную функцию :)

2. Байесовские методы

Задача 7. Пусть $X_1, \dots, X_n \sim \mathcal{N}(\theta, 1)$ 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 $\int_{\mathbb{R}} f(x)p(x)dx$,

где $p(x)$ --- некоторая плотность, можно взять величину $\sum_{j=1}^k f(Y_j)$, где Y_1, \dots, 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.

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

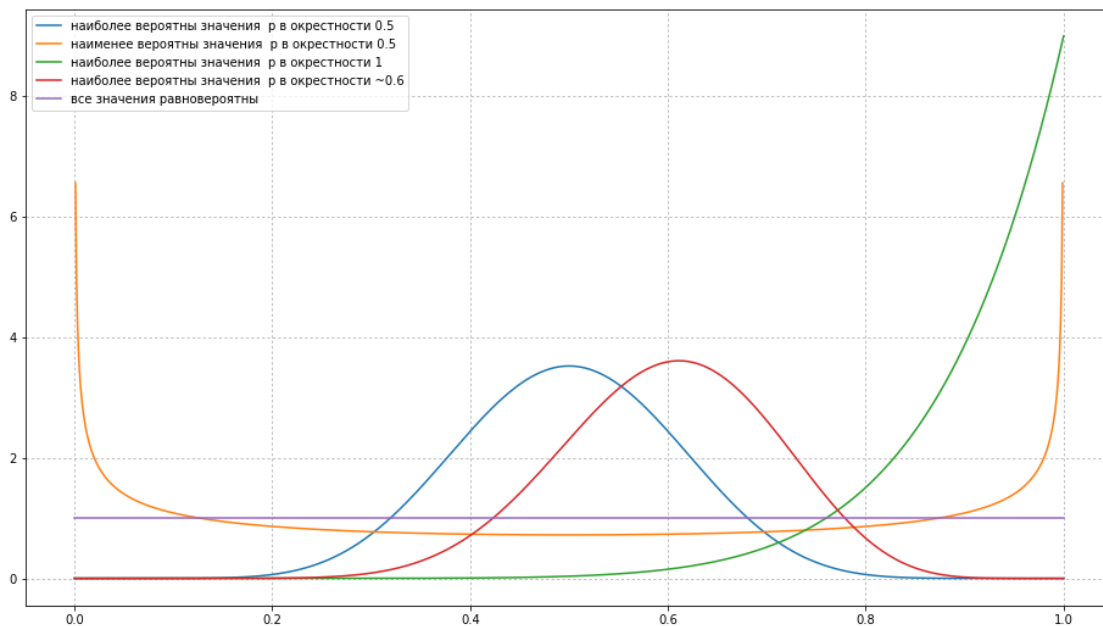
Постройте несколько графиков сопряженного распределения для разных параметров и охарактеризуйте, как значения параметров его соотносятся с априорными знаниями о монете. Это могут быть, например, знания вида

- 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 [123]:

```
plt.figure(figsize=(16, 9))
x = np.linspace(0, 1, 1000)
plt.plot(x, sps.beta(a=10, b=10).pdf(x), label='наиболее вероятны значения p в окрестности 0.5')
plt.plot(x, sps.beta(a=0.6, b=0.6).pdf(x), label='наименее вероятны значения p в окрестности 0.5')
plt.plot(x, sps.beta(a=9, b=1).pdf(x), label='наиболее вероятны значения p в окрестности 1')
plt.plot(x, sps.beta(a=12, b=8).pdf(x), label='наиболее вероятны значения p в окрестности ~0.6')
plt.plot(x, sps.beta(a=1, b=1).pdf(x), label='все значения равновероятны')
plt.grid(ls=':')
plt.legend()
plt.show()
```



Сопряженное для бернуллиевского есть бета распределение. Гиперпараметры бета распределения α и β говорят нам об априорных знаниях о монете. Расписав плотность выборки бернуллиевского распределения: $p^{\sum x_i} (1 - p)^{n - \sum x_i}$ и пристально посмотрев на плотность бета распределения: $\frac{x^{\alpha-1} (1 - x)^{\beta-1}}{B(\alpha, \beta)}$, понятно, что гиперпараметр α отвечает за количество успехов в n испытаниях Бернулли, а гиперпараметр β отвечает за количество неудач в n испытаниях Бернулли, что действительно согласуется с вышеописанными графиками и логикой: матожидание бета распределение, т.е. наша оценка параметра θ есть

$$\frac{\alpha}{\alpha + \beta} = \frac{\sum \text{успех}}{\sum \text{успех} + \sum \text{неудачи}} = \frac{\sum \text{успех}}{\text{общее количество испытаний}}$$

Ниже приведена реализация некоторых вспомогательных функций.

In [87]:

```
def draw_posteriori(grid, distr_class, post_params, xlim=None):
    ''' Рисуем 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):
    ''' Рисуем графики байесовской оценки (м.о. и дов. инт.) и ОМП.
        ml --- Оценка максимального правдоподобия для  $1 \leq n \leq \text{len}(\text{sample})$ 
        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.legend()
    plt.show()
```

Реализуйте следующую функцию

In [88]:

```
def bern_posterior_params(sample, a, b):  
    ''' Возвращает параметры апостериорного распределения для всех  $0 \leq n \leq \text{len}(\text{sample})$ .  
        a, b --- параметры априорного распределения.  
    '''  
    params = np.array(list(map(lambda n: np.array([a + np.sum(sample[:n]),  
                                                b + len(sample[:n]) - np.sum(sample  
[:n]))],  
                               range(len(sample)+1))))  
    return params
```

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`.

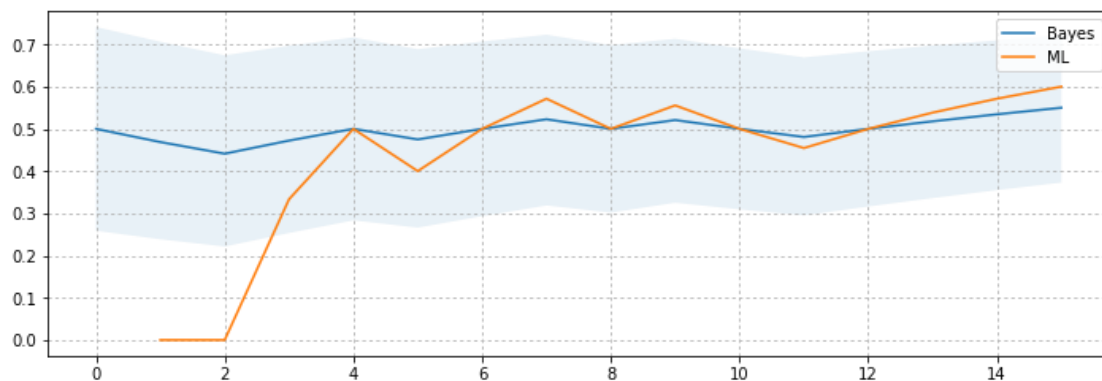
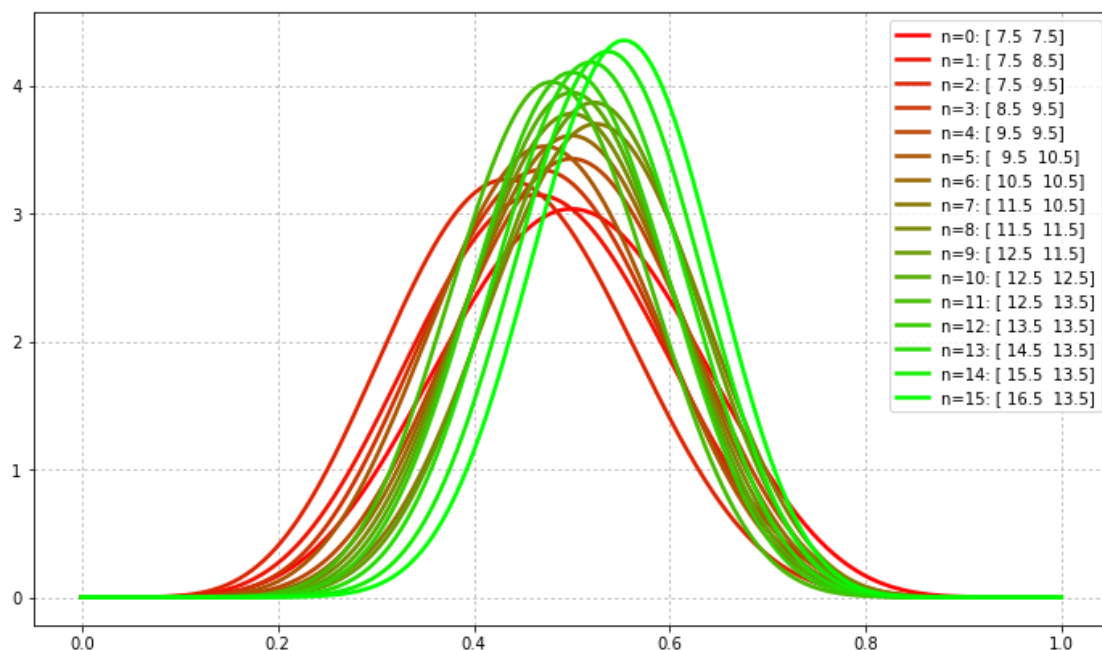
Сделайте вывод. Что можно сказать про зависимость от параметров априорного распределения? Сравните байесовские оценки с оценкой максимального правдоподобия.

1. Симметричная монета

1) Параметры подобраны правильно

In [89]:

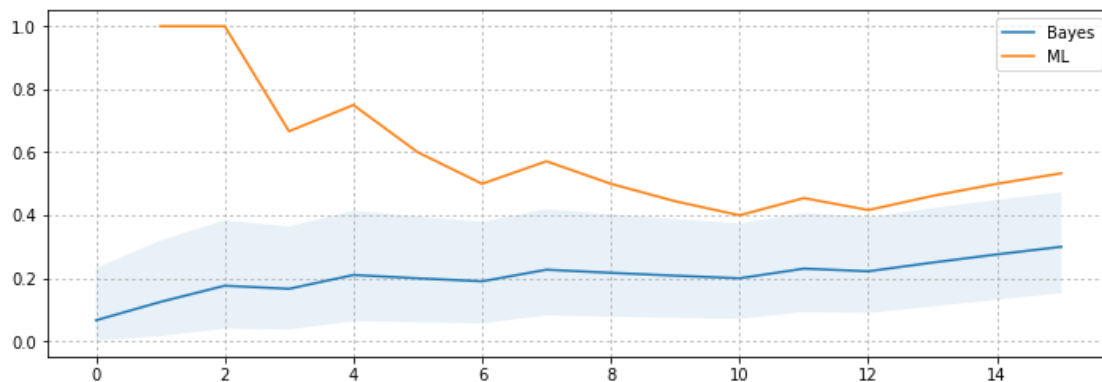
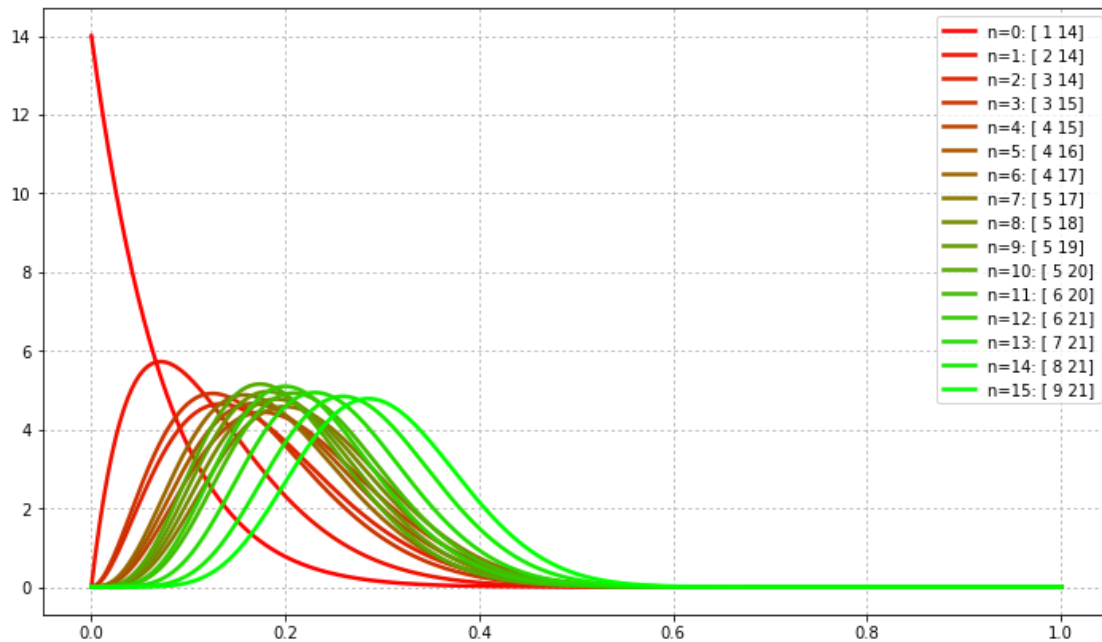
```
sample = sps.bernoulli(0.5).rvs(15)
grid = np.linspace(0, 1, 1000)
draw_posteriori(grid, sps.beta, bern_posterior_params(sample, 7.5, 7.5))
likelihood = np.cumsum(sample) / np.arange(1, len(sample) + 1)
draw_estimations(ml=likelihood, distr_class=sps.beta,
                 post_params=bern_posterior_params(sample, 7.5, 7.5))
```



2) Параметры подобраны неправильно

In [90]:

```
sample = sps.bernoulli(0.5).rvs(15)
grid = np.linspace(0, 1, 1000)
draw_posteriori(grid, sps.beta, bern_posterior_params(sample, 1, 14))
likelihood = np.cumsum(sample) / np.arange(1, len(sample) + 1)
draw_estimations(ml=likelihood, distr_class=sps.beta,
                 post_params=bern_posterior_params(sample, 1, 14))
```

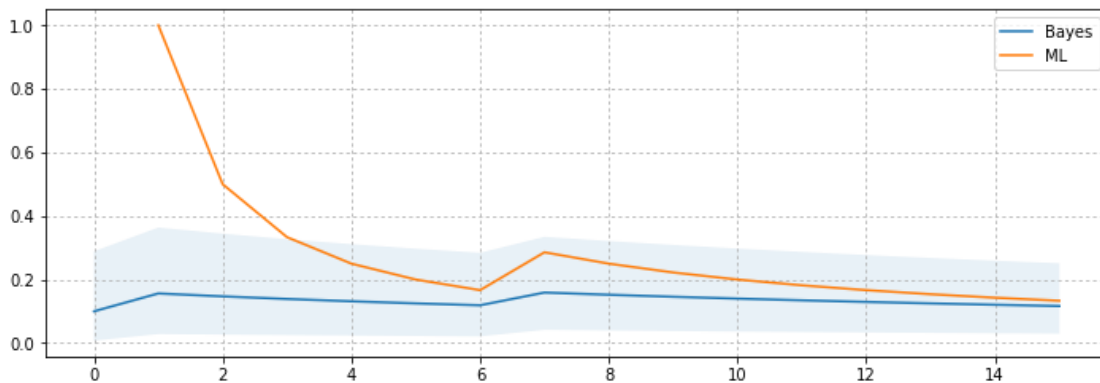
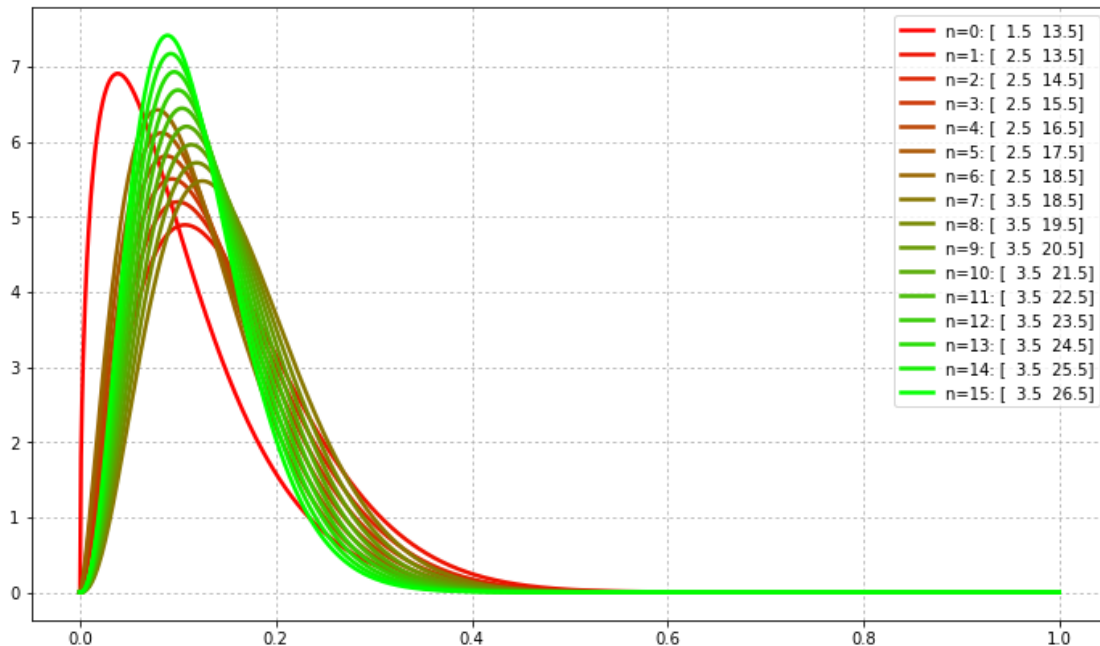


2. Несимметричная монета

1) Параметры подобраны правильно

In [91]:

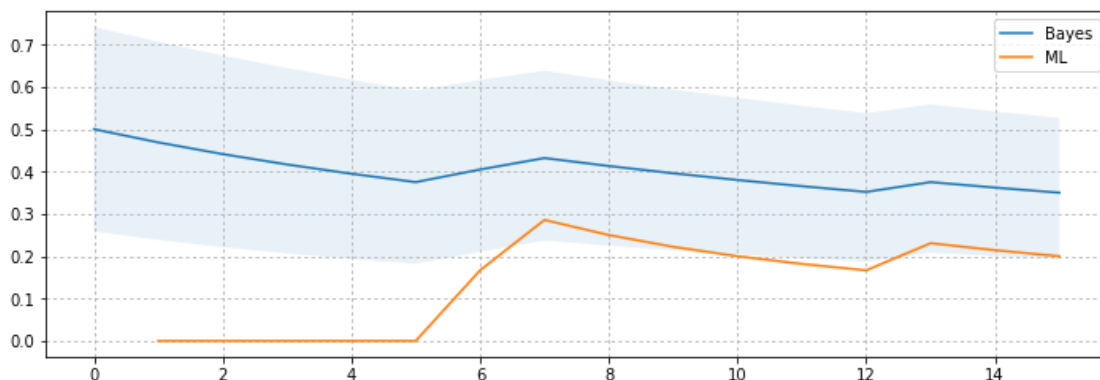
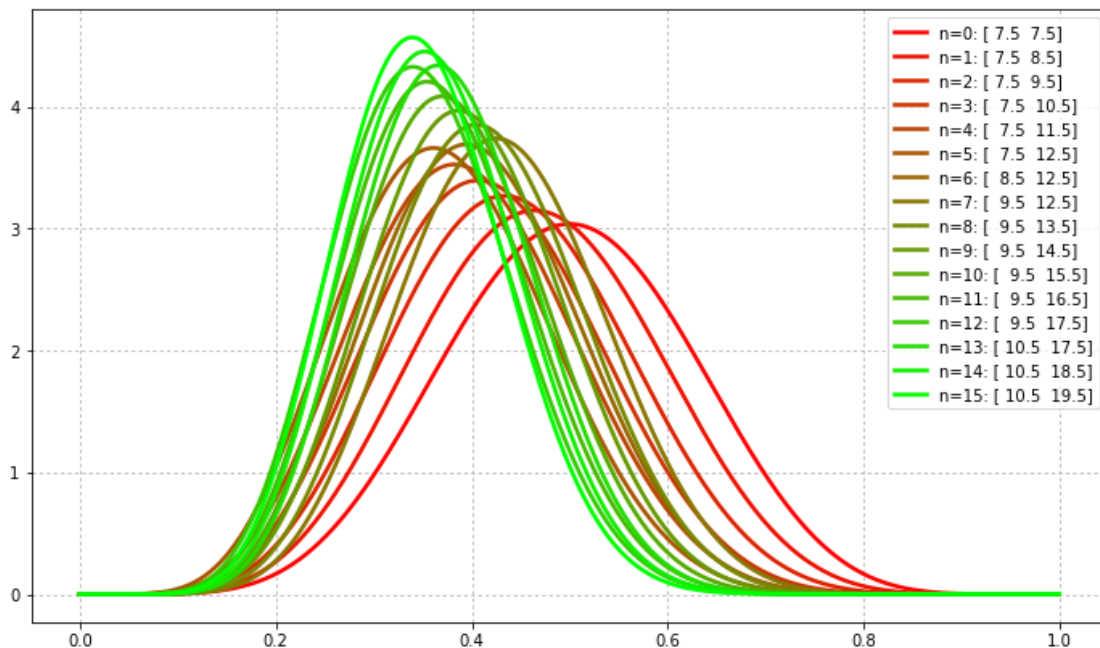
```
sample = sps.bernoulli(0.1).rvs(15)
grid = np.linspace(0, 1, 1000)
draw_posteriori(grid, sps.beta, bern_posterior_params(sample, 0.1 * 15, 0.9 * 15))
likelihood = np.cumsum(sample) / np.arange(1, len(sample) + 1)
draw_estimations(ml=likelihood, distr_class=sps.beta,
                 post_params=bern_posterior_params(sample, 0.1 * 15, 0.9 * 15))
```



2)Параметры подобраны неправильно

In [92]:

```
sample_simmetr = sps.bernoulli(0.1).rvs(15)
grid = np.linspace(0, 1, 1000)
draw_posteriori(grid, sps.beta, bern_posterior_params(sample_simmetr, 7.5, 7.5
))
likelihood = np.cumsum(sample_simmetr) / np.arange(1, len(sample_simmetr) + 1)
draw_estimations(ml=likelihood, distr_class=sps.beta,
                 post_params=bern_posterior_params(sample_simmetr, 7.5, 7.5))
```



Вывод: По графикам видим, что при правильно подобранных параметрах априорного распределения искомая апостериорная плотность изменяется не сильно (не сильно меняются основные оценки, такие как математическое ожидание, медиана, мода). При сравнении байесовской оценки с ОМП, замечаем, что байесовская оценка гораздо лучше ведет себя на выборках малого размера, что дает ей заметное преимущество по сравнению с частотными подходами.

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 $X_1, \dots, X_n \sim \text{Bern}(p)$ --- 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 $i \in \{0, \dots, 10\}$, 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 представлениях экзаменатора?
- Нужно ли дружить с таким экзаменатором?

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

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

1. $\mathcal{N}(\theta, 1)$

Сопряженным распределением для нормального с неизвестным матожиданием и известной дисперсией, равной 1, является нормальное распределение. Апостериорное распределение

параметра θ есть $\mathcal{N}\left(\frac{\mu_0 + \sum X_i}{1 + n\sigma_0^2}, \frac{\sigma_0^2}{1 + n\sigma_0^2}\right)$, где $\mathcal{N}(\mu_0, \sigma_0^2)$ -- априорное распределение параметра θ

1. Параметры подобраны удачно

$$\theta = 0$$

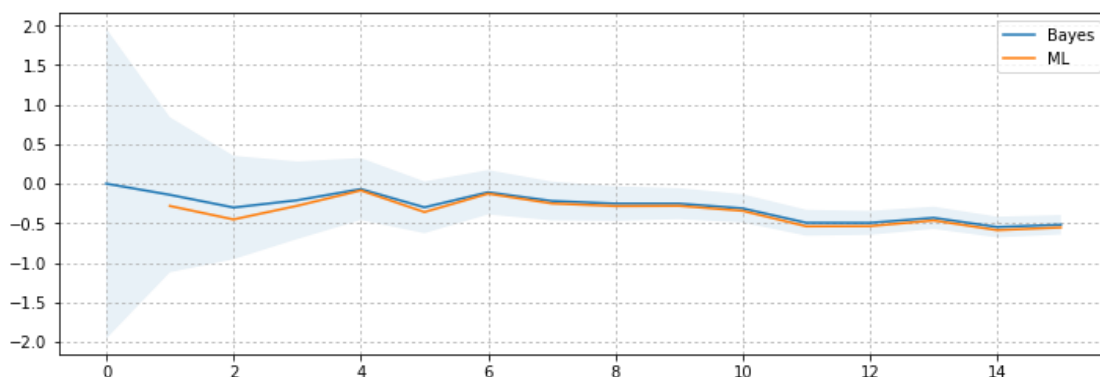
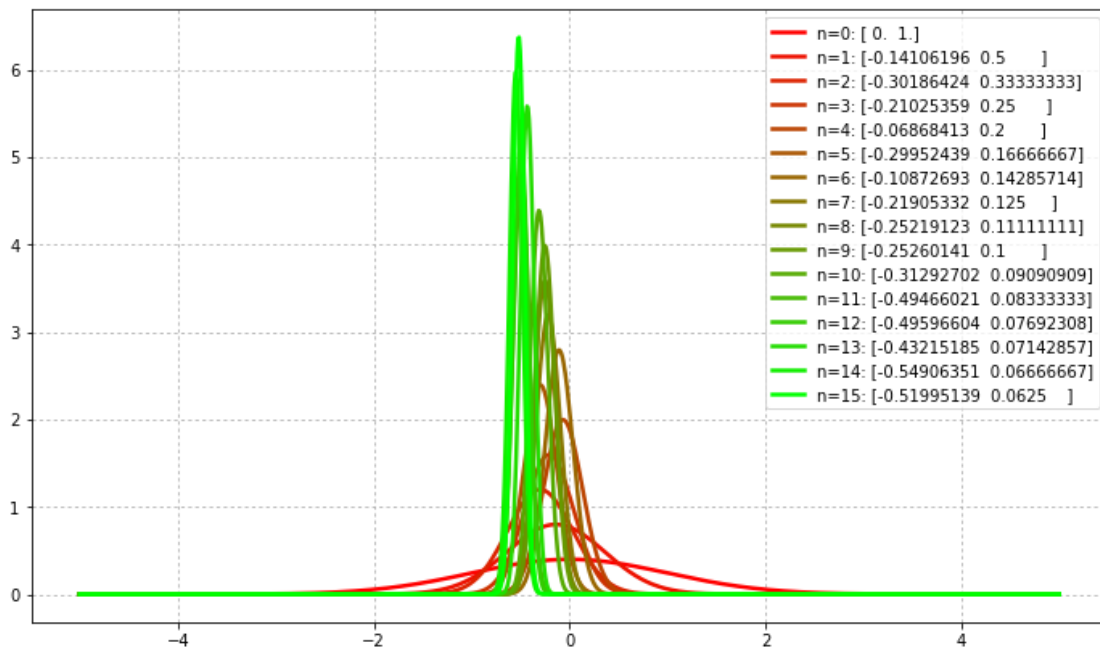
In [93]:

```
def norm_posterior_params(sample, mu_0, sigma_0):
    ''' Возвращает параметры апостериорного распределения для всех  $\theta \leq n \leq$ 
        len(sample).
        a, b --- параметры априорного распределения.
    '''
    params = np.array(list(map(lambda n: np.array([mu_0 + np.sum(sample[:n])
        /
        (1 + n * sigma_0 ** 2),
        sigma_0 ** 2 / (1 + n * sigma_0 ** 2)]),
        range(len(sample)+1))))

    return params
```

In [94]:

```
sample = sps.norm.rvs(size=15)
grid = np.linspace(-5, 5, 1000)
draw_posteriori(grid, sps.norm, norm_posterior_params(sample, 0, 1))
likelihood = np.cumsum(sample) / np.arange(1, len(sample) + 1)
draw_estimations(ml=likelihood, distr_class=sps.norm,
    post_params=norm_posterior_params(sample, 0, 1))
```

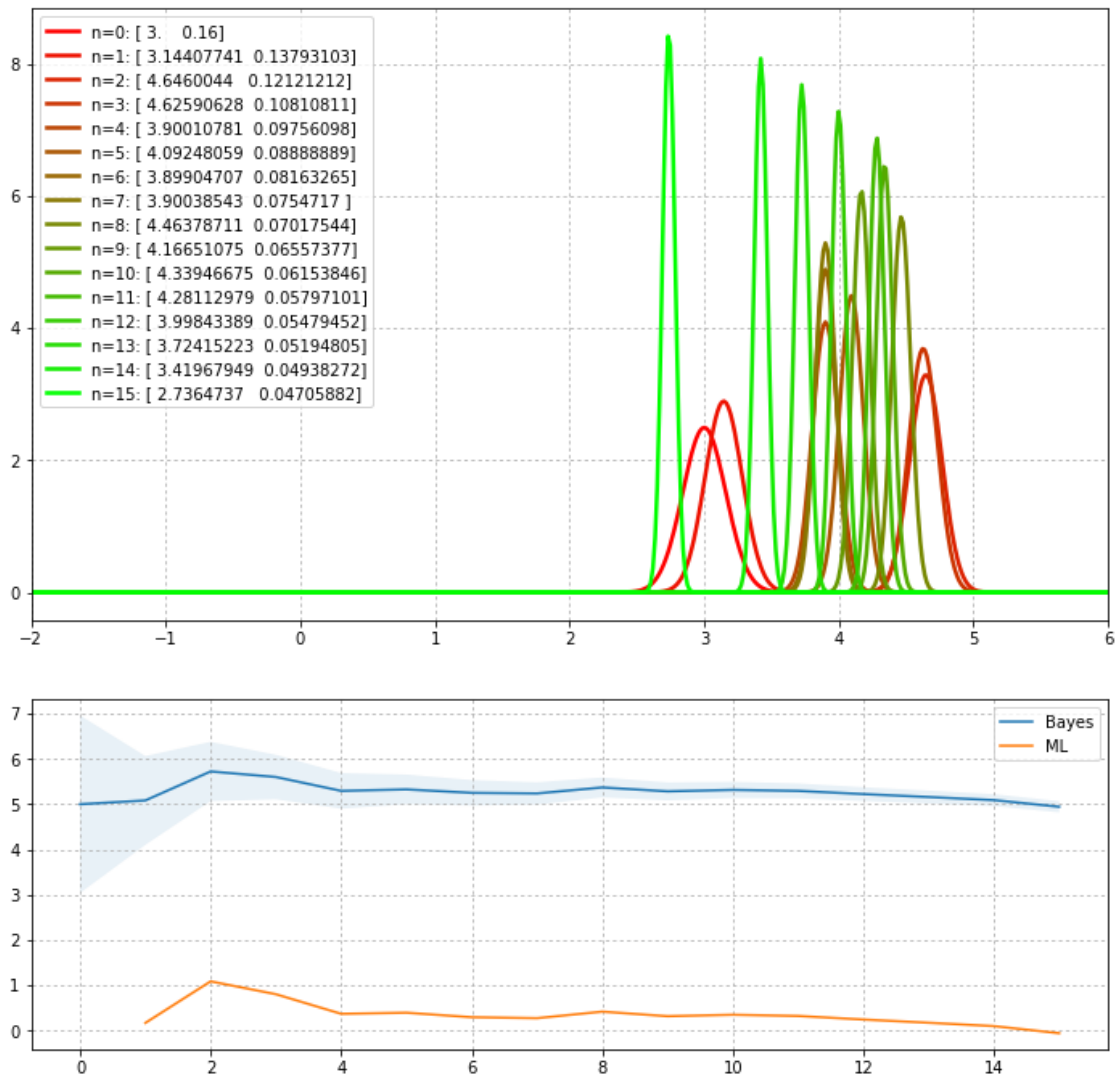


2. Параметры подобраны неудачно

$$\theta = 0$$

In [95]:

```
sample = sps.norm.rvs(size=15)
grid = np.linspace(-8, 8, 1000)
draw_posteriori(grid, sps.norm, norm_posterior_params(sample, 3, 0.4), xlim=(-2, 6))
likelihood = np.cumsum(sample) / np.arange(1, len(sample) + 1)
draw_estimations(ml=likelihood, distr_class=sps.norm,
                 post_params=norm_posterior_params(sample, 5, 1))
```



2. $Exp(\theta)$ Сопряженным распределением для экспоненциального является гамма-распределение. Апостериорное распределение параметра θ есть $\Gamma(\alpha + \sum X_i, \beta + n)$, где $\Gamma(\alpha, \beta)$ -- априорное распределение параметра θ

In [124]:

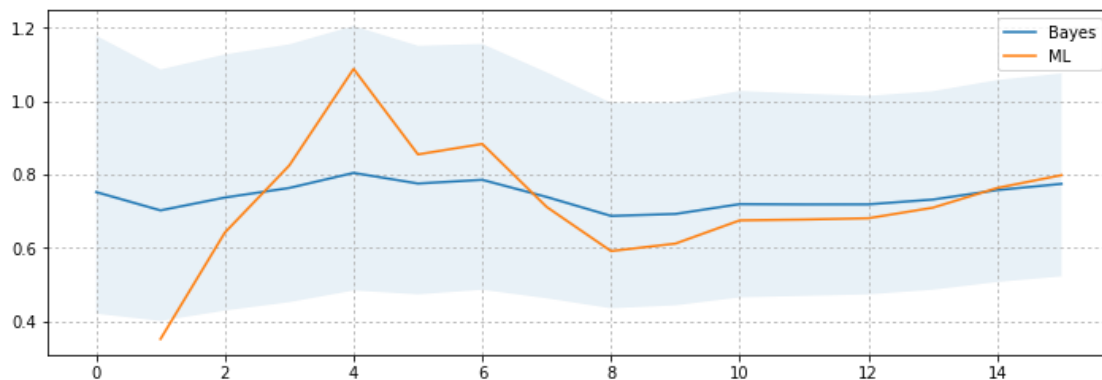
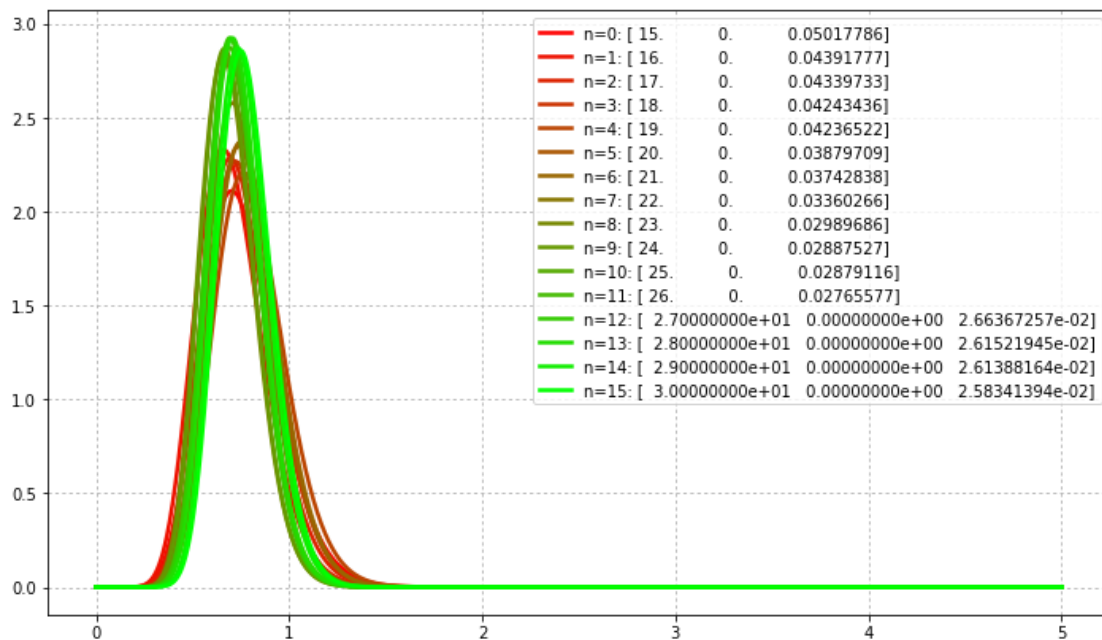
```
def exp_posterior_params(sample, a, b):  
    ''' Возвращает параметры апостериорного распределения для всех  $\theta \leq n \leq$   
     $\text{len}(\text{sample})$ .  
     $a, b$  --- параметры априорного распределения.  
    ...  
    return np.array(list(map(lambda n: np.array([b + n, 0, 1/(a + np.sum(sample[:n]))]),  
                                range(len(sample)+1))))
```

1. Параметры подобраны удачно

$$\theta = 1$$

In [126]:

```
sample = sps.expon.rvs(size=15)
grid = np.linspace(0, 5, 1000)
help_sample = sps.expon.rvs(size=15)
draw_posteriori(grid, sps.gamma, exp_posterior_params(sample, np.sum(help_sample), 15))
likelihood = np.arange(1, len(sample) + 1) / np.cumsum(sample)
draw_estimations(ml=likelihood, distr_class=sps.gamma,
                 post_params=exp_posterior_params(sample, np.sum(help_sample),
3 15))
```

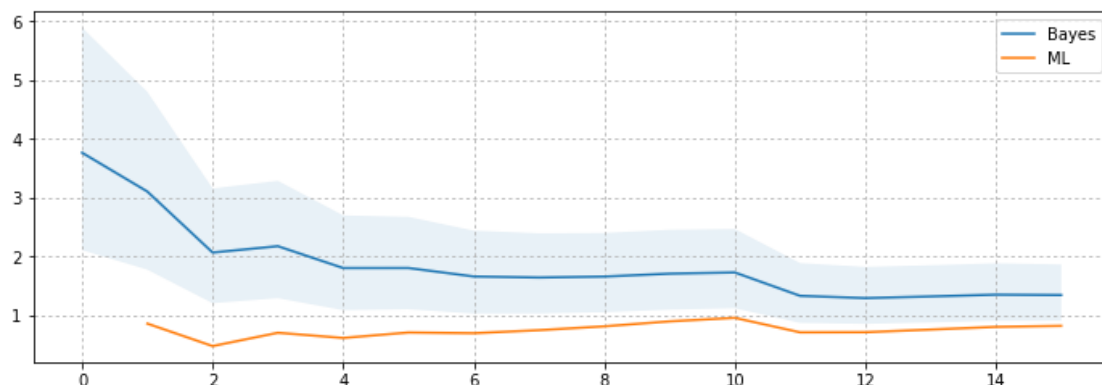
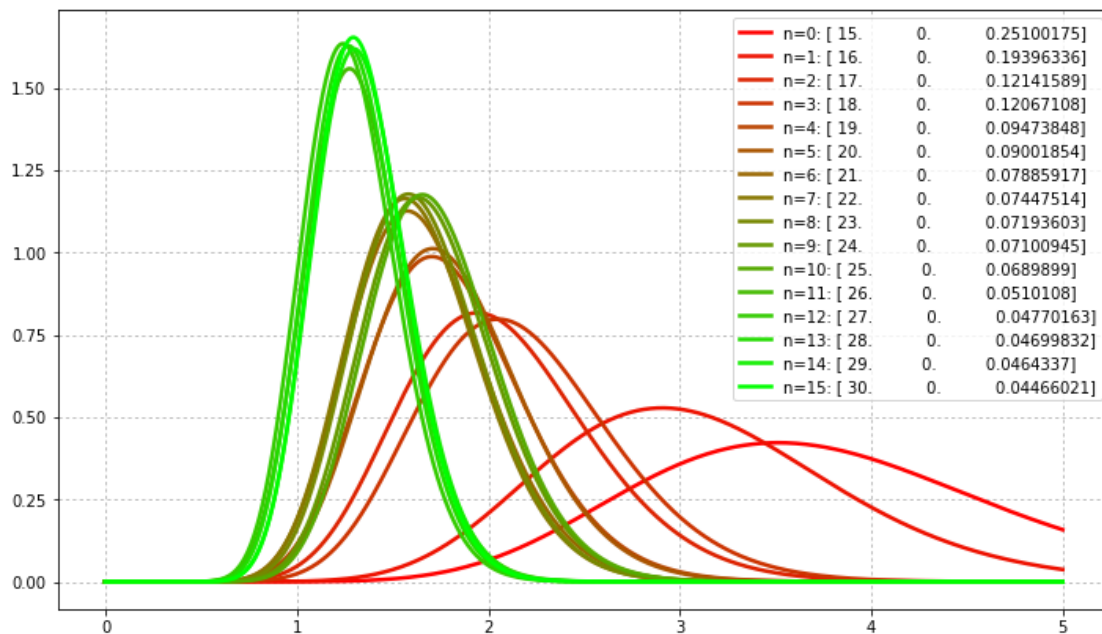


2. Параметры подобраны неудачно

$$\theta = 1$$

In [127]:

```
sample = sps.expon.rvs(size=15)
grid = np.linspace(0, 5, 1000)
help_sample = sps.expon(scale=1/4).rvs(size=15)
draw_posteriori(grid, sps.gamma, exp_posterior_params(sample, np.sum(help_sample), 15))
likelihood = np.arange(1, len(sample) + 1) / np.cumsum(sample)
draw_estimations(ml=likelihood, distr_class=sps.gamma,
                 post_params=exp_posterior_params(sample, np.sum(help_sample), 15))
```



Вывод: Как и в задаче 8, мы еще раз убедились в том, что на мадых выборках байесовский подход проявляет себя намного лучше ОМП. Так же мы научились подбирать гиперпараметры априорного распределения.

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