СКБ201 Тур Тимофей

Теория вероятности, долгосрочные домашние задания. Вариант 68: дискретное - 3, непрерывное - 5.

3 - Дискретное равномерное 1: $P(x) = \theta^{-1}, x \in \{1, \dots, \theta\}, \theta = 29$

Обозначим дискретное распределение в дальнейшем за ξ

5 - Треугольное: $f(x) = \begin{cases} \frac{2x}{\theta}, & \text{если } x \in [0, \theta] \\ \frac{2(1-x)}{1-\theta}, & \text{если } x \in (\theta, 1] \\ 0, & \text{иначе} \end{cases}, \theta = 0.45$

Обозначим абсолютно непрерывное распределение в дальнейшем за η

Навигация

Домашнее задание 1

Дискретное

Задание 1

Функция распределения

$$F(n) \overset{\text{def}}{=} P(\xi \leq n) = \sum_{k=1}^{n} P(\xi = k) = \sum_{k=1}^{n} \theta^{-1} = \underline{n\theta^{-1}}$$

Математическое ожидание

$$M\xi \overset{\text{def}}{=} \sum_{x=1}^{\theta} xP(\xi = x) = \sum_{x=1}^{\theta} x\theta^{-1} = \theta^{-1}\sum_{x=1}^{\theta} x = \theta^{-1}\frac{1+\theta}{2}\theta = \underline{\frac{\theta+1}{2}}$$

Дисперсия

$$D\xi \overset{\text{def}}{=} M(\xi - M\xi)^2 = M\xi^2 - (M\xi)^2$$

$$M\xi^2 \overset{\text{def}}{=} \sum_{x=1}^{\theta} x^2 P(\xi = x) = \theta^{-1}\sum_{x=1}^{\theta} x^2 = \theta^{-1}\frac{\theta(1-\theta)(1+2\theta)}{6} = \frac{(1+\theta)(1+2\theta)}{6}$$

$$\Rightarrow D\xi = M\xi^2 - (M\xi)^2 = \frac{(1+\theta)(1+2\theta)}{6} - (\frac{\theta+1}{2})^2 = \frac{2(1+3\theta+2\theta^2) - 3(1+2\theta+\theta^2)}{12} = \underline{\frac{\theta^2-1}{12}}$$

Квантиль уровня γ

$$P(\xi \leq x_\gamma) \geq \gamma \text{ и } P(\xi \geq x_\gamma) \geq 1-\gamma$$

$$1.\ P(\xi \leq x_\gamma) \geq \gamma \Rightarrow \sum_{k=1}^{x_\gamma} P(\xi = k) \geq \gamma \Leftrightarrow \sum_{k=1}^{x_\gamma} \theta^{-1} \geq \gamma \Leftrightarrow x_\gamma\theta^{-1} \geq \gamma \Leftrightarrow x_\gamma \geq \gamma\theta$$

$$2.\ P(\xi \geq x_\gamma) \geq 1-\gamma \Rightarrow \sum_{k=x_\gamma}^{\theta} P(\xi = k) \geq 1-\gamma \Leftrightarrow \sum_{k=x_\gamma}^{\theta} \theta^{-1} \geq 1-\gamma \Leftrightarrow (\theta - x_\gamma + 1)\theta^{-1} \geq 1-\gamma \Leftrightarrow$$

$$\Leftrightarrow 1 - x_\gamma\theta^{-1} + \theta^{-1} \geq 1-\gamma \Leftrightarrow x_\gamma \leq \gamma\theta + 1$$

$$\Rightarrow \underline{\gamma\theta \leq x_\gamma \leq \gamma\theta + 1}$$

Задание 2

Примеры

1. Примером события с дискретным равномерным распределением может быть игра "Bingo". Но не вся она, а лишь ее часть. В ней, подобно лото, участникам выдаются цветные листки с числами и маркерами, а ведущий стоит у аппарата, поторый по нажатию кнопки выдает случайный шарик, крутящийся в нем. Шарик имеет цвет и номер, и участники выделяют соответсвующие ячейки на своем листе, пока у них не получатся какая-нибудь соответствующая последовательность. (Лично я увидел эту игру в сериале "Лучше звоните Солу" в первом сезоне). Чтобы эта модель была применима к нашему распределению, игру следует упростить: На листке всего 1 номер и мячики не имеют цвета. Тогда шанс появления какого-то мячика будет равен $\frac{1}{\text{количество мячиков} = \theta} = \theta^{-1}$, и, соответственно шанс выигрыша какого-то игрока тоже равен θ^{-1}.

2. Подбрасывание монетки с числами на ее сторонах "1" и "2". Вероятность выпадения каждой равна $\frac{1}{\text{количество сторон}} = \frac{1}{2}$, что соответствует $\theta = 2$.

3. Любые стандартные игральные кости ($d4$, $d6$, $d8$, $d10$, $d12$, $d20 \ldots$)

Соотношения

1. Из примера 2 следует соотношение $U(\theta = 2) = B(1, \frac{1}{2})$.

2. Если выбрать некоторое $k_i \in \{1, \ldots, \theta\}$ за 1 некоторой функции g, а остальные элементы из $\{1, \ldots, \theta\}$ принять за 0 (g определена на $\{1, \ldots, \theta\}$), то $g(U(\theta)) = B(1, \frac{1}{\theta})$.

3. Тогда, через отслеживание множества функций g, определенных выше, и их значений, то из $G(g_1(U(\theta)), \ldots, g_n(U(\theta)))$ можно построить $B(n, \frac{1}{\theta})$.

4. Множеством таких G по n можно определить геометрическое распределение. (но на самом деле на третьем соотношении можно было закончить, так как оно связывается с остальными через него, а значит уже не напрямую)

Задание 3

Поделим отрезок $[0, 1]$ на сегменты равные θ^{-1}. Их будет в точности θ штук, а выборка определяется вхождением в какой из последовательных отрезков получилось у случайной величины: $\sqsupset u$ - сгенерированная равномерно распределенная величина на отрезке $[0, 1]$, тогда x определяется по формуле $(x-1)\theta^{-1} \le u < x\theta^{-1}$

In [1]:
```python
import numpy as np

def generate_xi(theta=29):
    rng = np.random.default_rng()
    u = rng.uniform()
    for k in range(1, theta + 1):
        if (k - 1) / theta <= u < k / theta:
            return k
```

Абсолютно непрерывное

Задание 1

Функция распределения

$$F(x) \overset{\text{def}}{=} \int_{\mathbb{R}} f(x) = \begin{cases} 0, & \text{если } x < 0 \\ \displaystyle\int_0^x \frac{2t}{\theta} dt, & \text{если } x \in [0, \theta] \\ \displaystyle\int_0^\theta \frac{2t}{\theta} dt + \int_\theta^x \frac{2(1-t)}{1-\theta} dt, & \text{если } x \in (\theta, 1] \\ \displaystyle\int_0^\theta \frac{2t}{\theta} dt + \int_\theta^1 \frac{2(1-t)}{1-\theta} dt, & \text{если } x > 1 \end{cases}$$

$$1) \int_0^x \frac{2t}{\theta} dt = \frac{1}{\theta} \int_0^x 2t dt = \frac{1}{\theta} t^2 \Big|_0^x = \frac{1}{\theta} x^2$$

$$\begin{aligned} 2) \int_0^\theta \frac{2t}{\theta} dt + \int_\theta^x \frac{2(1-t)}{1-\theta} dt &= \theta + \frac{2}{1-\theta} \int_\theta^x (1-t) dt = \theta + \frac{2}{1-\theta}(t - \frac{1}{2}t^2)\Big|_\theta^x = \\ &= \theta + \frac{2}{1-\theta}(x - \frac{1}{2}x^2 - \theta + \frac{1}{2}\theta^2) = \theta + \frac{1}{1-\theta}(2x - x^2 - 2\theta + \theta^2) = \\ &= \frac{1}{1-\theta}(2x - x^2 - \theta) \end{aligned}$$

$$3) \int_0^\theta \frac{2t}{\theta} dt + \int_\theta^1 \frac{2(1-t)}{1-\theta} = \frac{1}{1-\theta}(2 - 1 - \theta) = \frac{1-\theta}{1-\theta} = 1$$

$$\Rightarrow F(x) = \begin{cases} 0, & \text{если } x < 0 \\ \dfrac{1}{\theta} x^2, & \text{если } x \in [0, \theta] \\ \dfrac{1}{1-\theta}(2x - x^2 - \theta), & \text{если } x \in (\theta, 1] \\ 1, & \text{если } x > 1 \end{cases}$$

Математическое ожидание

$$\begin{aligned} M\eta \overset{\text{def}}{=} \int_{\mathbb{R}} f(x) x dx &= \int_0^\theta \frac{2x}{\theta} x dx + \int_\theta^1 \frac{2(1-x)}{1-\theta} x dx = \frac{2}{\theta} \int_0^\theta x^2 dx + \frac{2}{1-\theta} \int_\theta^1 (x - x^2) dx = \\ &= \frac{2}{3\theta} x^3 \Big|_0^\theta + \frac{2}{1-\theta}(\frac{1}{2}x^2 - \frac{1}{3}x^3)\Big|_\theta^1 = \frac{2}{3\theta}\theta^3 + \frac{2}{1-\theta}(\frac{1}{2} - \frac{1}{3} - \frac{1}{2}\theta^2 + \frac{1}{3}\theta^3) = \\ &= \frac{2}{3}\theta^2 + \frac{2}{1-\theta}(\frac{1}{6} + \frac{2\theta^3 - 3\theta^2}{6}) = \frac{2\theta^2}{3} + \frac{2\theta^3 - 3\theta^2 + 1}{3(1-\theta)} = \frac{2\theta^2 - 2\theta^3 + 2\theta^3 - 3\theta^2 + 1}{3(1-\theta)} = \\ &= \frac{1-\theta^2}{3(1-\theta)} = \frac{1+\theta}{3} \end{aligned}$$

Дисперсия

$$D\eta \overset{\text{def}}{=} M(\eta - M\eta)^2 = M\eta^2 - (M\eta)^2$$

$$\begin{aligned} M\eta^2 \overset{\text{def}}{=} \int_{\mathbb{R}} f(x)x^2dx &= \int_0^\theta \frac{2x}{\theta}x^2dx + \int_\theta^1 \frac{2(1-x)}{1-\theta}x^2dx = \frac{2}{\theta}\int_0^\theta x^3dx + \frac{2}{1-\theta}\int_\theta^1 (x^2-x^3)dx = \\ &= \frac{1}{2\theta}x^4\Big|_0^\theta + \frac{2}{1-\theta}(\frac{1}{3}x^3 - \frac{1}{4}x^4)\Big|_\theta^1 = \frac{1}{2\theta}\theta^4 + \frac{2}{1-\theta}(\frac{1}{3} - \frac{1}{4} - \frac{1}{3}\theta^3 + \frac{1}{4}\theta^4) = \\ &= \frac{1}{2}\theta^3 + \frac{2}{1-\theta}(\frac{1}{12} + \frac{3\theta^4 - 4\theta^3}{12}) = \frac{1}{2}\theta^3 + \frac{1}{6(1-\theta)}(3\theta^4 - 4\theta^3 + 1) = \\ &= \frac{1}{6(1-\theta)}(3\theta^4 - 4\theta^3 + 1 + 3\theta^3 - 3\theta^4) = \frac{1}{6(1-\theta)}(1-\theta^3) = \frac{1+\theta+\theta^2}{6} \end{aligned}$$

$$\Rightarrow D\eta = M\eta^2 - (M\eta)^2 = \frac{1+\theta+\theta^2}{6} - (\frac{1+\theta}{3})^2 = \frac{3(1+\theta+\theta^2) - 2(1+2\theta+\theta^2)}{18} = \underline{\frac{1-\theta+\theta^2}{18}}$$

Квантиль уровня γ

$$F(x_\gamma) = \gamma \Rightarrow \begin{cases} x_\gamma = 0, & \text{если } \gamma < 0 \\ \frac{1}{\theta}x_\gamma^2 = \gamma, & \text{если } \gamma \in [0, \theta] \\ \frac{1}{1-\theta}(2x_\gamma - x_\gamma^2 - \theta) = \gamma, & \text{если } \gamma \in (\theta, 1] \\ x_\gamma = 1, & \text{если } \gamma > 1 \end{cases}$$

$$1)\ \frac{1}{\theta}x_\gamma^2 \geq \gamma \Leftrightarrow x_\gamma \geq \sqrt{\theta\gamma} \Rightarrow x_\gamma = \sqrt{\theta\gamma}$$

$$\begin{aligned} 2)\ \frac{1}{1-\theta}(2x_\gamma - x_\gamma^2 - \theta) \geq \gamma \Rightarrow &\ \frac{1}{1-\theta}(2x_\gamma - x_\gamma^2 - \theta) = \gamma \Leftrightarrow -x_\gamma^2 + 2x_\gamma - \theta = (1-\theta)\gamma \Leftrightarrow \\ &\Leftrightarrow -x_\gamma^2 + 2x_\gamma - \theta - \gamma + \theta\gamma = 0 \Rightarrow \\ &\Rightarrow D = 4 - 4(\theta + \gamma - \theta\gamma) = 4(1 - \theta - \gamma + \theta\gamma) \Rightarrow \\ &\Rightarrow x_\gamma = \frac{-2 \pm 2\sqrt{1-\theta-\gamma+\theta\gamma}}{-2} = 1 \pm \sqrt{1-\theta-\gamma+\theta\gamma}. \\ &x_\gamma \in [\theta, 1] \Rightarrow x_\gamma = 1 - \sqrt{1-\theta-\gamma+\theta\gamma} \end{aligned}$$

$$\Rightarrow x_\gamma = \underline{\begin{cases} x_\gamma = 0, & \text{если } \gamma < 0 \\ \sqrt{\theta\gamma}, & \text{если } \gamma \in [0, \theta] \\ 1 - \sqrt{1-\theta-\gamma+\theta\gamma}, & \text{если } \gamma \in (\theta, 1] \\ x_\gamma = 1, & \text{если } \gamma > 1 \end{cases}}$$

Задание 2

Примеры

Треугольное распределение на практике используется часто, потому что оно имеет минимум, максимум и пик, что делает его уже достаточным к реальности распределением, так еще и оно очень простое по своей математике и применению. Конкретно в приведенной формуле распределение ограничено 0 и 1 и имеет пик в θ, а в обычных случаях оно позволяет посчитать предполагаемую прибыль какого-то ресторана, просто делая предположение о минимуме, максимуме и наиболее вероятном значении при помощи анализа полученного распределения (например через математическое ожидание).

Соотношения

Треугольное распределение с параметром $\theta = \frac{1}{2}$ равно $\frac{U_1+U_2}{2}$, где U_1 и U_2 - независимые равномерное случайное распределение от 0 до 1. Пусть $T = \frac{U_1+U_2}{2}$, тогда при $x \in [0, \frac{1}{2}]$ имеем $\{T \leq x\} = \{U_1 + U_2 \leq 2x\} = \{U_1 \leq 2x - U_2\}$,что в точности трегольник с катетами $2x$, а значит его площадь равна $2x^2$. При $x \in [\frac{1}{2}, 1]$ же будем иметь $\{T > x\} = \{U_1 + U_2 > 2x\} = \{U_1 > 2x - U_2\}$, что трегольник с катетами $2 - 2x = 2(1-x)$ и площадью $2(1-x)^2$, а значит область $\{T \leq x\}$ имеет площать $1 - 2(1-x)^2$. При $x < 0$ событие $\{U_1 + U_2 \leq 2x\}$ не выполняется никогда, при $x > 1$ событие $\{U_1 + U_2 \leq 2x\}$ выполняется всегда. Это в конечном итоге дает распределение, которое в точности совпадает с треугольным распределением с параметром $\theta = \frac{1}{2}$:

$$F(x) = \begin{cases} 0, & \text{если } x < 0 \\ 2x^2, & \text{если } x \in [0, \frac{1}{2}] \\ 1 - 2(1-x)^2, & \text{если } x \in (\frac{1}{2}, 1] \\ 1, & \text{если } x > 1 \end{cases}$$

Задание 3

Чтобы построить выборку от равномерного случайного распределения требуется найти $F^{-1}(u)$, что мы фактически искали, вычисляя квантиль уровня γ. Чем я и воспользуюсь, описав код ниже.

In [2]:
```python
import numpy as np
def generate_eta(theta=0.45):
    rng = np.random.default_rng()
    u = rng.uniform()
    if u <= theta: return (theta*u)**0.5
    return 1-(1-theta-u+theta*u)**0.5
```

Домашнее задание 2

Предупреждение: выполнение кода без файлов "sample_xi.pkl" и "sample_eta.pkl" сгенерирует новые выборки и сохранит их прямо у вас. Чтобы такого не было, либо загрузите уже готовые выборки, либо немного модифицируйте код (уберите сериализацию и раскомменьте строку прямо над ней).

Дискретное

Задание 1

In [3]:
```python
# Здесь допустимо использование функций генераторов, указанных ранее
# theta задана в каждой функции генератора параметром по умолчанию
# потому отдельное упоминание не требуется
n = [5, 10, 100, 200, 400, 600, 800, 1000]

#sample_xi = [[np.sort(np.array([generate_xi() for i in range(j)]))
#               for i in range(5)] for j in n]
# BEGIN сериализация
import os, pickle
if 'sample_xi.pkl' in os.listdir():
    with open('sample_xi.pkl', 'rb') as f:
        sample_xi = pickle.load(f)
else:
    sample_xi = [[np.sort(np.array([generate_xi() for i in range(j)]))
                  for i in range(5)] for j in n]
    with open('sample_xi.pkl', 'wb') as f:
        pickle.dump(sample_xi, f)
# END сериализация

# демонстрация первых выборок
for i in range(len(n)):
    print('Пример сгенерированной выборки длины %d:'%n[i], end=' ')
    print(*sample_xi[i][0])
    print('-'*10)
```

```
Пример сгенерированной выборки длины 5: 9 9 10 18 24
----------
Пример сгенерированной выборки длины 10: 7 10 14 18 19 20 20 21 26 28
----------
Пример сгенерированной выборки длины 100: 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4
4 5 5 5 5 5 5 6 6 6 6 6 6 7 7 8 8 8 8 10 10 11 11 11 11 11 12 12 12 12 13 13 13
13 13 14 14 15 15 15 16 16 17 18 18 18 18 19 19 20 20 20 21 21 21 22 22 22 23 23
23 24 24 24 24 24 25 25 26 26 26 26 26 27 27 27 28 28 28 29 29
----------
Пример сгенерированной выборки длины 200: 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 3
3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 8 8
9 9 9 9 9 9 10 10 10 10 10 10 10 11 11 11 11 11 11 11 11 11 11 11 12 12 12 12 12
13 13 13 13 14 14 14 14 14 14 14 14 14 14 14 15 15 16 16 16 16 16 16 16 17 17 17
18 18 18 18 18 18 18 18 19 20 20 20 20 20 20 20 20 20 20 20 21 21 21 21 21 22 22
22 22 23 23 23 23 23 23 23 23 24 24 24 24 24 24 24 25 25 25 25 25 26 26 26 26 26
26 26 26 26 26 26 27 27 27 27 27 27 27 27 27 27 28 28 28 28 28 29 29 29 29 29 29
29 29 29 29
----------
Пример сгенерированной выборки длины 400: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2
2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5
5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 10 10 10 10 10 10 10 10 10 10 10
10 11 11 11 11 11 11 11 11 11 11 11 11 11 11 12 12 12 12 12 12 12 12 12 12 12 12
13 13 13 13 13 13 13 13 13 13 13 13 13 13 14 14 14 14 14 14 14 14 14 14 14 14 14
14 14 14 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 16 16 16 16
16 16 16 16 16 16 16 16 17 17 17 17 17 17 17 17 17 18 18 18 18 18 18 18 18 18 18
18 18 18 18 18 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 21 21 21 21 21 21 21 21 21 21 22 22 22
22 22 22 22 22 22 22 22 22 22 22 22 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 23 23 24 24 24 24 24 24 24 24 24 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 26 26 26 26 26 26 26 26 27 27 27 27 27 27 27 27 27 27 27 27 28 28 28 28
28 28 28 28 28 28 28 28 28 28 28 29 29 29 29 29 29 29 29 29 29 29 29
----------
Пример сгенерированной выборки длины 600: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 11 11 11 11 11 11 11 11 11 11 11 11
11 11 11 11 11 11 11 11 11 11 11 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
12 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 14 14 14 14 14 14 14
14 14 14 14 14 14 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
15 15 15 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 17 17 17
17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 18 18 18 18 18 18 18 18 18 18 18
18 18 18 18 18 18 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 21 21 21 21 21 21 21 21 21
21 21 21 21 21 21 21 21 21 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22
22 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 24 24 24 24 24 24 24
24 24 24 24 24 24 24 24 24 24 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 26 26 26 26 26 26 26 26 26 26 26 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 28 28 28 28 28 28 28 28 28
28 28 28 28 28 28 28 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29
29 29 29 29 29 29
----------
Пример сгенерированной выборки длины 800: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 9 9
```

9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 11 11 11 11 11 11
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 12
12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 13 13 13 13 13
13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 14 14 14
14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
14 14 14 14 14 14 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
15 15 15 15 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
16 16 16 16 16 16 16 16 16 16 16 16 16 17 17 17 17 17 17 17 17 17 17 17 17 17 17
17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 18 18 18 18 18 18 18 18 18 18 18
18 18 18 18 18 18 18 18 18 18 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
19 19 19 19 19 19 19 19 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21
21 21 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22
22 22 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 23 23 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
24 24 24 24 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 26
26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26
26 26 26 26 26 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28
28 28 28 28 28 28 28 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29
29 29 29 29 29 29

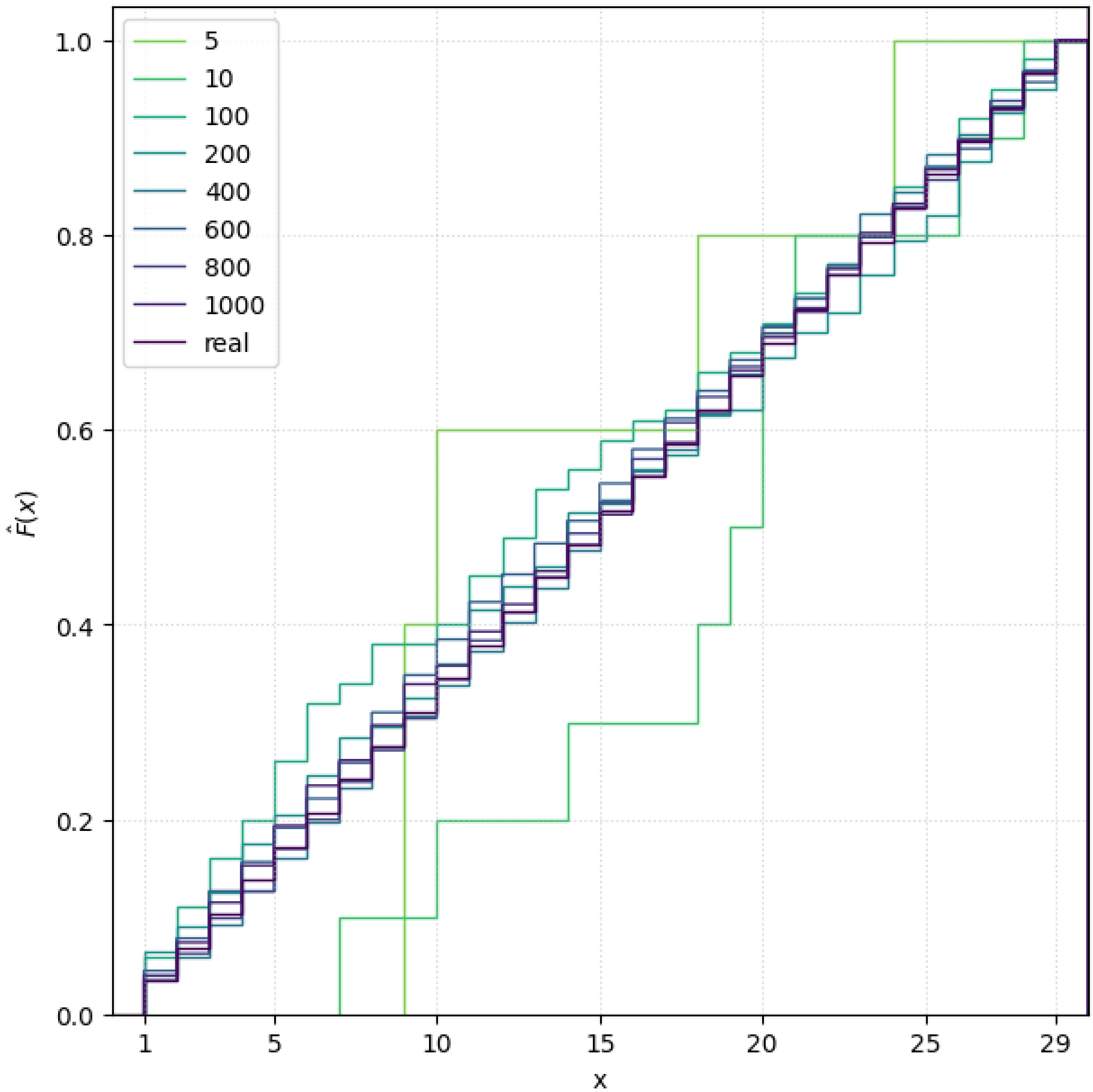
Пример сгенерированной выборки длины 1000: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 9 9
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 11 11 11 11 11 11 11 11
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
12 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
13 13 13 13 13 13 13 13 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
14 14 14 14 14 14 14 14 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
15 15 15 15 15 15 15 15 15 15 15 15 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 17 17 17
17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17
17 17 17 17 17 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18
18 18 18 18 18 18 18 18 18 18 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21
21 21 21 21 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22
22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 23 23 23 23 23 23
23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
24 24 24 24 24 24 24 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 26 26 26 26 26 26 26 26 26 26 26 26
26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 28 28 28 28 28
28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28
28 28 28 28 28 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29
29 29 29 29 29 29 29 29 29 29 29 29 29

Задание 2

In [4]:
```
def xi_distr(sample, x):
    res = 0
    for i in sample:
        if i <= x:
            res += 1
    return res / len(sample)

def xi_distr_real(x: int, theta=29):
    return x / theta

X_realxi = np.arange(1-1, 29+1+1)
Y_realxi = xi_distr_real(X_realxi)

Yxi = [[[np.array(xi_distr(sample_xi[k][j], x)) for x in sample_xi[k][j]]
                 for j in range(5)] for k in range(len(n))]

def xi_Dmn(Yn, Ym, Xn, Xm):
    n = len(Xn)
    m = len(Xm)
    Xn = np.concatenate([[0],Xn,[30]])
    Xm = np.concatenate([[0],Xm,[30]])
    Yn = np.concatenate([[0],Yn,[1]])
    Ym = np.concatenate([[0],Ym,[1]])
    if m>n:
        m, n = n, m
        Xm, Xn = Xn, Xm
        Ym, Yn = Yn, Ym

    res = 0
    for i in range(1, n+1):
        j=0
        while not (Xm[j]<=Xn[i]<Xm[j+1]):
            j+=1
        d = abs(Yn[i]-Ym[j])
        if d>res: res = d
    return (n*m/(n+m))**0.5*res

diffsxi = np.array([[[(xi_Dmn(Yxi[i][k], Yxi[j][k],
                              sample_xi[i][k], sample_xi[j][k]))
                      for i in range(len(n))] for j in range(len(n))]
                    for k in range(5)])
diffsxi_demo = np.array([[[('%.2f' % diffsxi[k][j][i]) for i in range(len(n))]
                           for j in range(len(n))] for k in range(5)])
```

In [5]:
```python
from matplotlib import pyplot as plt
colors = ['#7bd152', '#45be71', '#25a885', '#21908c',
          '#2b798e', '#355f8d', '#414486', '#482574']

# graph[0]
fig, ax = plt.subplots(2,1, figsize=(7,12), height_ratios = [2,1])
for i in range(8):
    ax[0].stairs(Yxi[i][0], np.append(sample_xi[i][0], 30), color = colors[i])
ax[0].stairs(Y_realxi, np.append(X_realxi, 30), color = '#440154')
ax[0].set(xticks = [1]+list(range(5,26,5))+[29], xmargin = 0, ymargin = 0,
          xlabel = 'x', ylabel = r'$\hat{F}(x)$',
          title = 'Дискретное равномерное, выборка 1 \n$\\xi , \\,\\theta$ = 29')
ax[0].legend([*n, 'real'], loc='upper left')
ax[0].grid(True, alpha = 0.5, ls='dotted')

ax[1].table(cellText = diffsxi_demo[0], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');
```

Дискретное равномерное, выборка 1
ξ, θ = 29

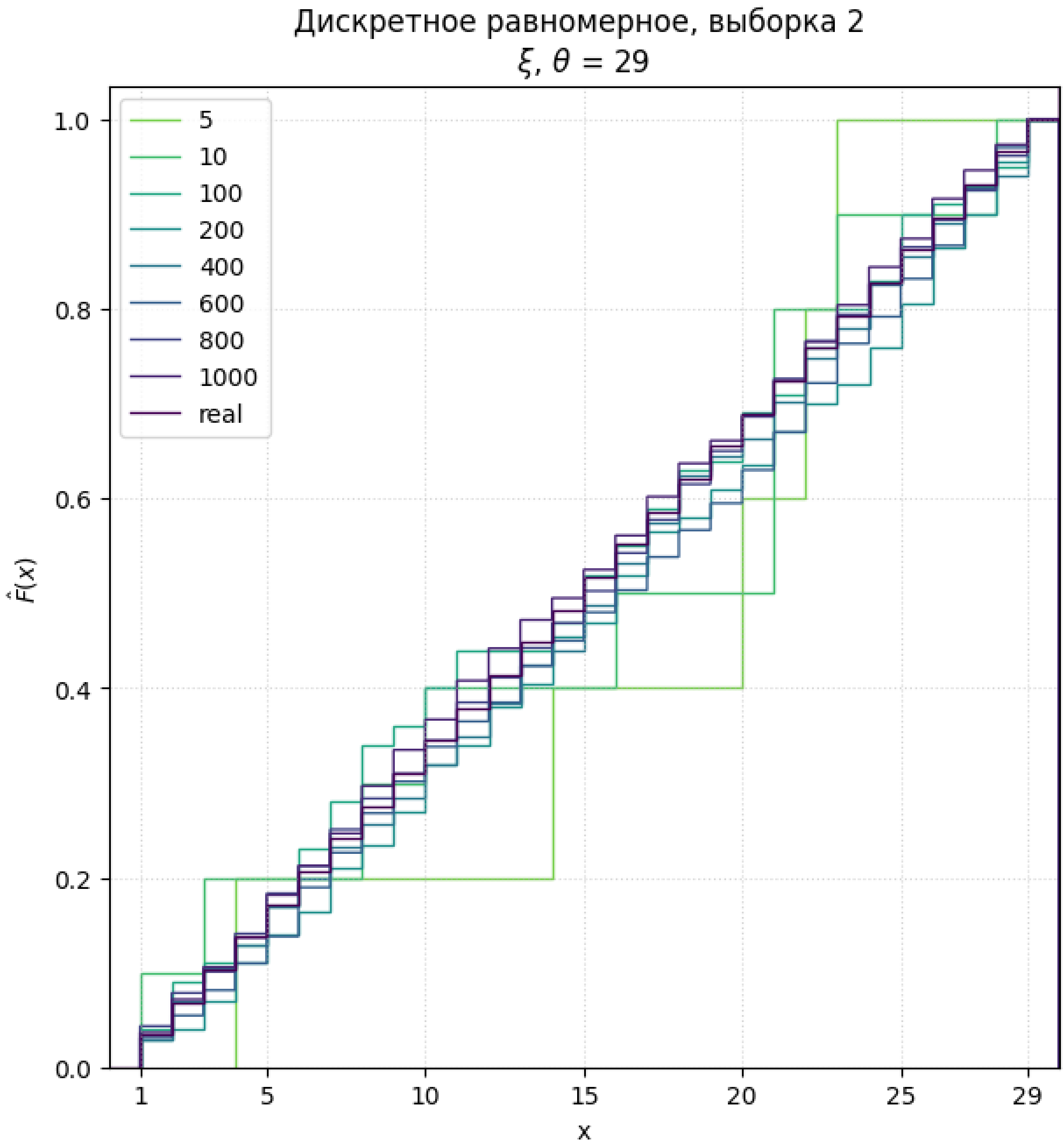


$$D_{m,n} = \sqrt{\frac{nm}{n+m}} \sup_{x \in \mathbb{R}} |F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.73	0.83	0.65	0.61	0.69	0.60	0.66
10	0.73	0.00	1.03	0.85	0.87	0.98	0.97	0.90
100	0.83	1.03	0.00	0.69	1.10	0.91	1.13	0.81
200	0.65	0.85	0.69	0.00	0.72	0.63	0.62	0.61
400	0.61	0.87	1.10	0.72	0.00	0.79	0.45	0.63
600	0.69	0.98	0.91	0.63	0.79	0.00	0.83	0.62
800	0.60	0.97	1.13	0.62	0.45	0.83	0.00	0.74
1000	0.66	0.90	0.81	0.61	0.63	0.62	0.74	0.00

In [6]:

```
# graph[1]
fig, ax = plt.subplots(2,1, figsize=(7,12), height_ratios = [2,1])
for i in range(8):
    ax[0].stairs(Yxi[i][1], np.append(sample_xi[i][1], 30), color = colors[i])
ax[0].stairs(Y_realxi, np.append(X_realxi, 30), color = '#440154')
ax[0].set(xticks = [1]+list(range(5,26,5))+[29], xmargin = 0, ymargin = 0,
          xlabel = 'x', ylabel = r'$\hat{F}(x)$',
          title = 'Дискретное равномерное, выборка 2 \n$\\xi , \\,\\theta$ = 29')
ax[0].legend([*n, 'real'], loc='upper left')
ax[0].grid(True, alpha = 0.5, ls='dotted')

ax[1].table(cellText = diffsxi_demo[1], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');
```

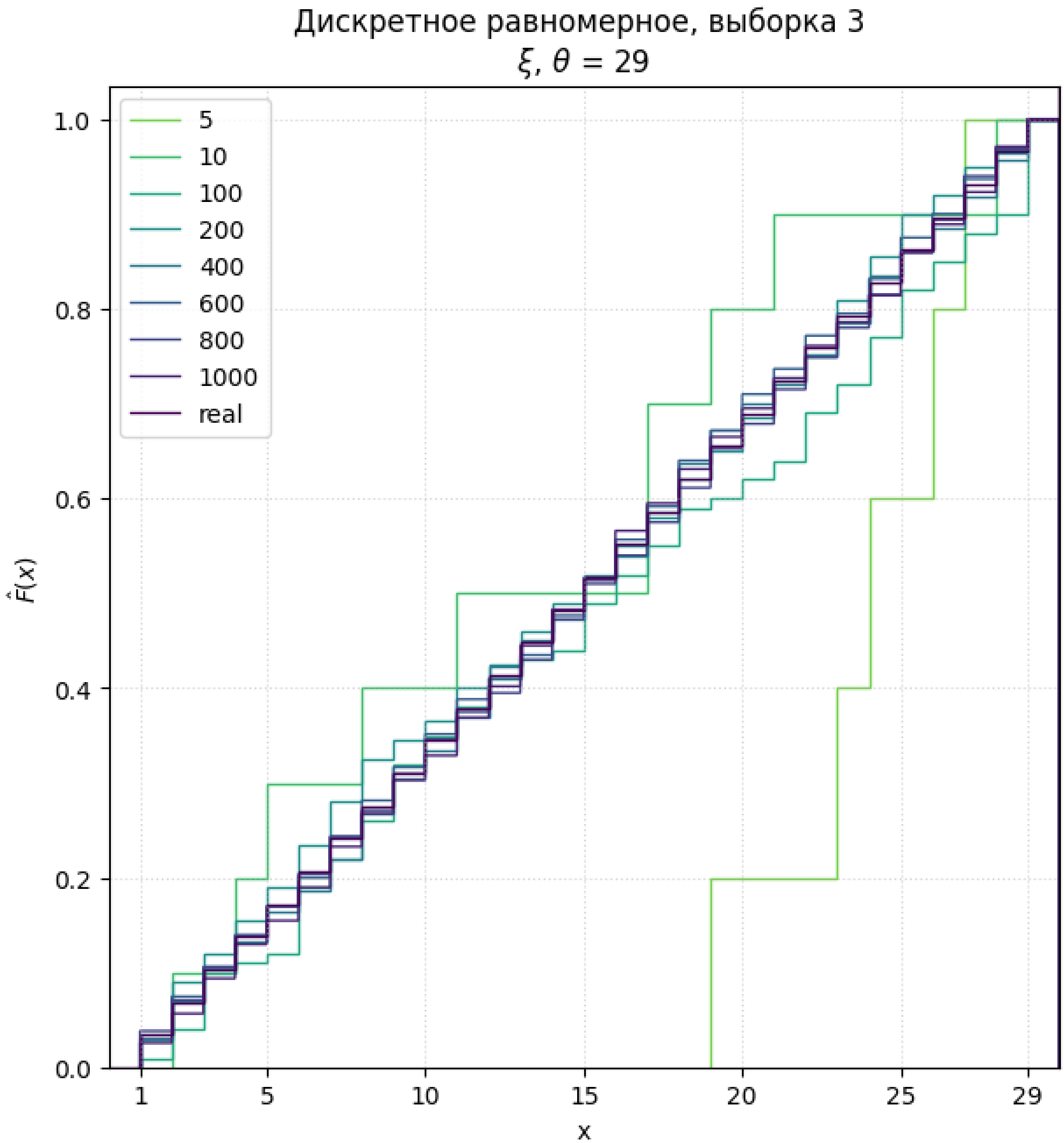


$$D_{m,n} = \sqrt{\frac{nm}{n+m}} \sup_{x \in \mathbb{R}} |F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.37	0.52	0.62	0.54	0.53	0.56	0.61
10	0.37	0.00	0.57	0.56	0.51	0.43	0.59	0.59
100	0.52	0.57	0.00	0.86	0.80	0.69	0.53	0.41
200	0.62	0.56	0.86	0.00	0.75	0.53	0.93	1.08
400	0.54	0.51	0.80	0.75	0.00	0.90	0.57	0.98
600	0.53	0.43	0.69	0.53	0.90	0.00	1.04	1.36
800	0.56	0.59	0.53	0.93	0.57	1.04	0.00	0.65
1000	0.61	0.59	0.41	1.08	0.98	1.36	0.65	0.00

In [7]:

```
# graph[2]
fig, ax = plt.subplots(2,1, figsize=(7,12), height_ratios = [2,1])
for i in range(8):
    ax[0].stairs(Yxi[i][2], np.append(sample_xi[i][2], 30), color = colors[i])
ax[0].stairs(Y_realxi, np.append(X_realxi, 30), color = '#440154')
ax[0].set(xticks = [1]+list(range(5,26,5))+[29], xmargin = 0, ymargin = 0,
          xlabel = 'x', ylabel = r'$\hat{F}(x)$',
          title = 'Дискретное равномерное, выборка 3 \n$\\xi , \\,\\theta$ = 29')
ax[0].legend([*n, 'real'], loc='upper left')
ax[0].grid(True, alpha = 0.5, ls='dotted')

ax[1].table(cellText = diffsxi_demo[2], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');
```

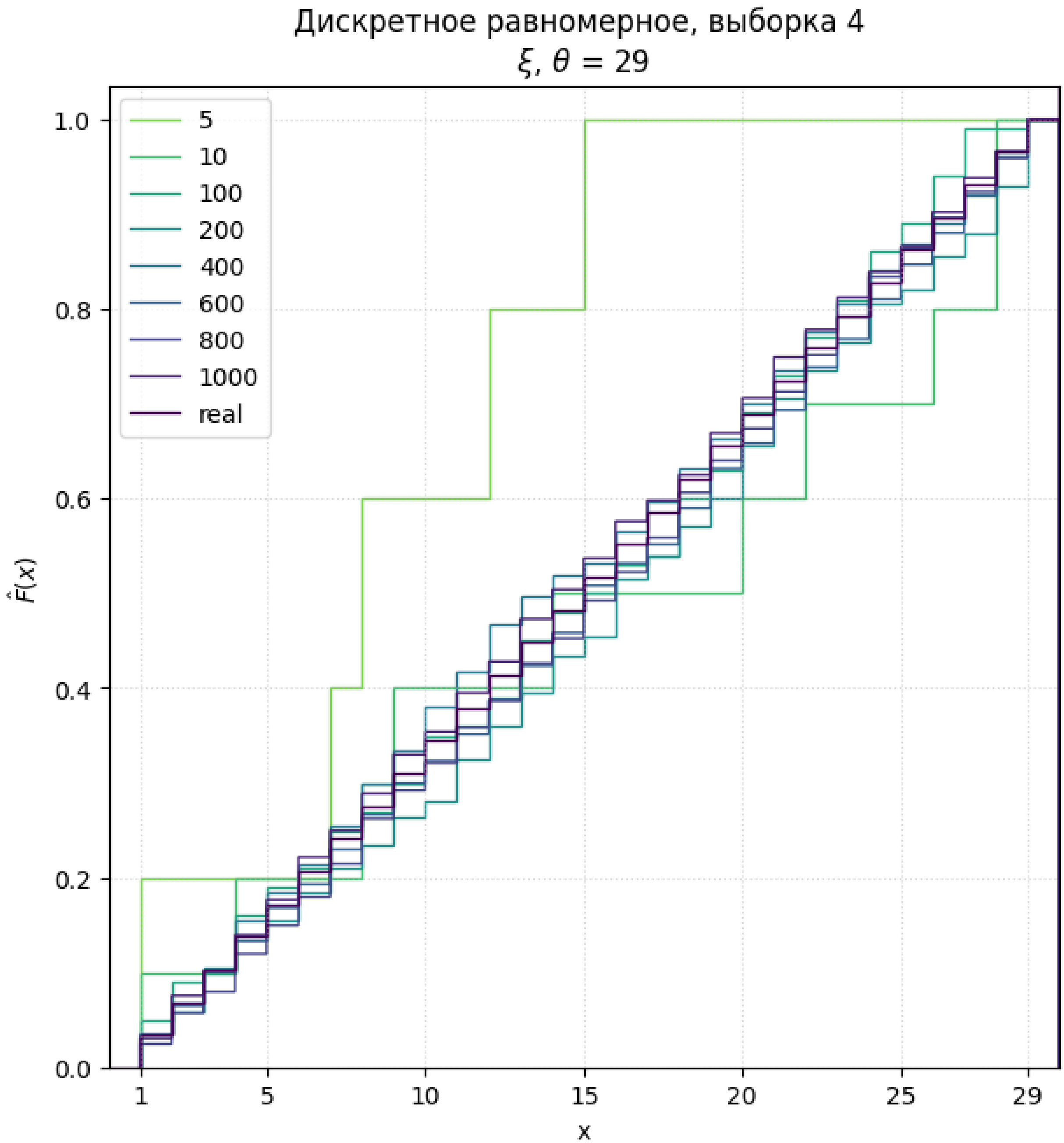


$$D_{m,n} = \sqrt{\frac{nm}{n+m}} \sup_{x \in \mathbb{R}} |F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	1.28	1.29	1.37	1.42	1.43	1.36	1.41
10	1.28	0.00	0.78	0.54	0.56	0.51	0.58	0.54
100	1.29	0.78	0.00	0.73	0.72	0.89	0.71	0.83
200	1.37	0.54	0.73	0.00	0.69	0.53	0.68	0.74
400	1.42	0.56	0.72	0.69	0.00	0.36	0.45	0.35
600	1.43	0.51	0.89	0.53	0.36	0.00	0.58	0.44
800	1.36	0.58	0.71	0.68	0.45	0.58	0.00	0.55
1000	1.41	0.54	0.83	0.74	0.35	0.44	0.55	0.00

In [8]:

```
# graph[3]
fig, ax = plt.subplots(2,1, figsize=(7,12), height_ratios = [2,1])
for i in range(8):
    ax[0].stairs(Yxi[i][3], np.append(sample_xi[i][3], 30), color = colors[i])
ax[0].stairs(Y_realxi, np.append(X_realxi, 30), color = '#440154')
ax[0].set(xticks = [1]+list(range(5,26,5))+[29], xmargin = 0, ymargin = 0,
          xlabel = 'x', ylabel = r'$\hat{F}(x)$',
          title = 'Дискретное равномерное, выборка 4 \n$\\xi , \\,\\theta$ = 29')
ax[0].legend([*n, 'real'], loc='upper left')
ax[0].grid(True, alpha = 0.5, ls='dotted')

ax[1].table(cellText = diffsxi_demo[3], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');
```

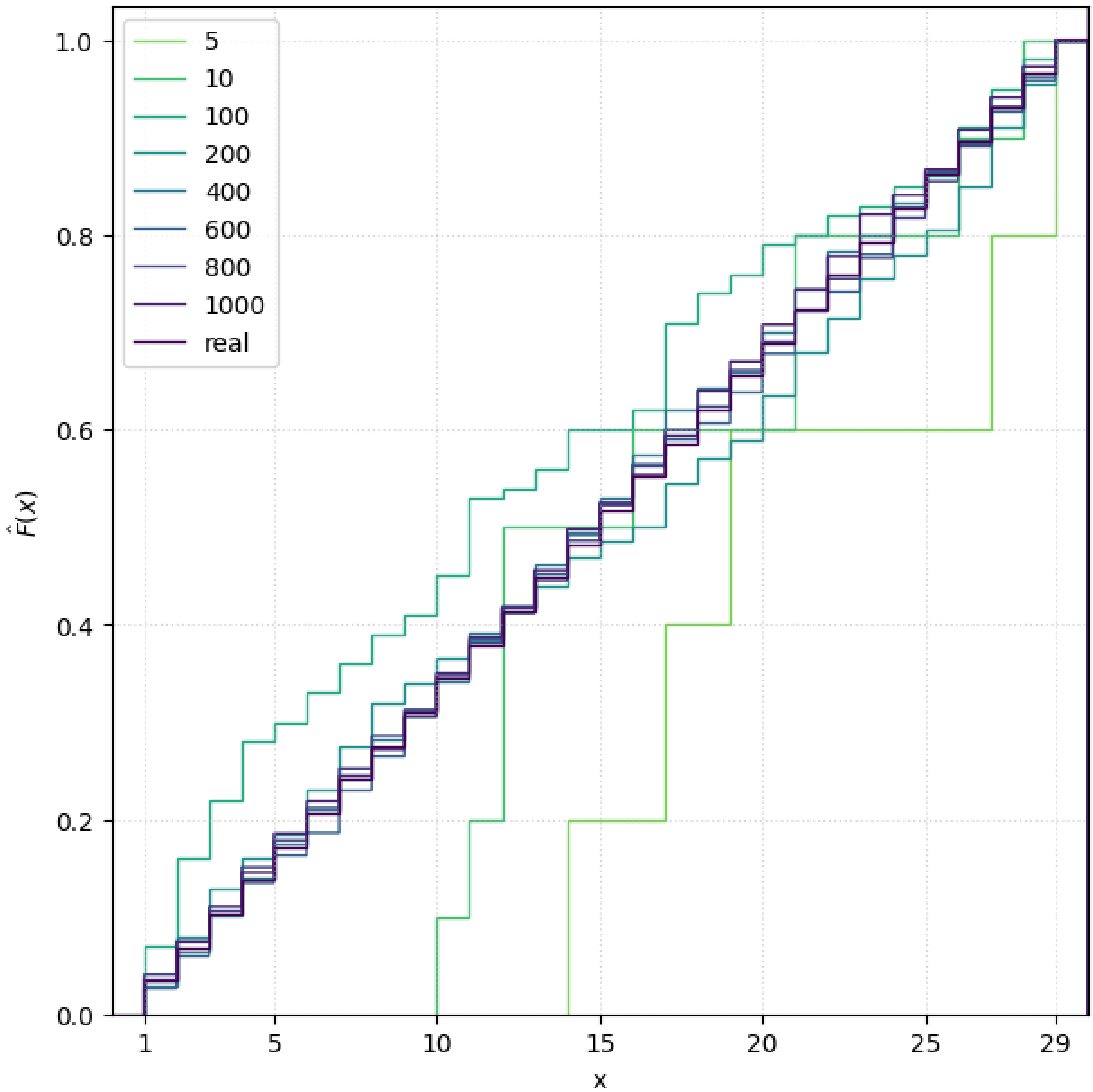


$$D_{m,n} = \sqrt{\frac{nm}{n+m}} \sup_{x \in \mathbb{R}} |F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.73	1.09	1.20	1.04	1.09	1.13	1.03
10	0.73	0.00	0.57	0.42	0.52	0.46	0.51	0.53
100	1.09	0.57	0.00	0.90	0.69	0.65	0.62	0.55
200	1.20	0.42	0.90	0.00	1.24	0.65	0.55	1.06
400	1.04	0.52	0.69	1.24	0.00	1.23	1.33	0.67
600	1.09	0.46	0.65	0.65	1.23	0.00	0.44	1.08
800	1.13	0.51	0.62	0.55	1.33	0.44	0.00	1.13
1000	1.03	0.53	0.55	1.06	0.67	1.08	1.13	0.00

In [9]:
```
# graph[4]
fig, ax = plt.subplots(2,1, figsize=(7,12), height_ratios = [2,1])
for i in range(8):
    ax[0].stairs(Yxi[i][4], np.append(sample_xi[i][4], 30), color = colors[i])
ax[0].stairs(Y_realxi, np.append(X_realxi, 30), color = '#440154')
ax[0].set(xticks = [1]+list(range(5,26,5))+[29], xmargin = 0, ymargin = 0,
          xlabel = 'x', ylabel = r'$\hat{F}(x)$',
          title = 'Дискретное равномерное, выборка 5 \n$\\xi , \\,\\theta$ = 29')
ax[0].legend([*n, 'real'], loc='upper left')
ax[0].grid(True, alpha = 0.5, ls='dotted')

ax[1].table(cellText = diffsxi_demo[4], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');
```

Дискретное равномерное, выборка 5
ξ, θ = 29



$$D_{m,n} = \sqrt{\frac{nm}{n+m}} \sup_{x \in \mathbb{R}} |F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.91	1.22	0.97	1.03	1.01	0.99	1.02
10	0.91	0.00	1.24	1.05	0.98	0.96	0.98	0.96
100	1.22	1.24	0.00	1.39	1.25	1.34	1.40	1.36
200	0.97	1.05	1.39	0.00	0.87	0.80	0.90	1.03
400	1.03	0.98	1.25	0.87	0.00	0.63	0.45	0.44
600	1.01	0.96	1.34	0.80	0.63	0.00	0.48	0.79
800	0.99	0.98	1.40	0.90	0.45	0.48	0.00	0.94
1000	1.02	0.96	1.36	1.03	0.44	0.79	0.94	0.00

Задание 3

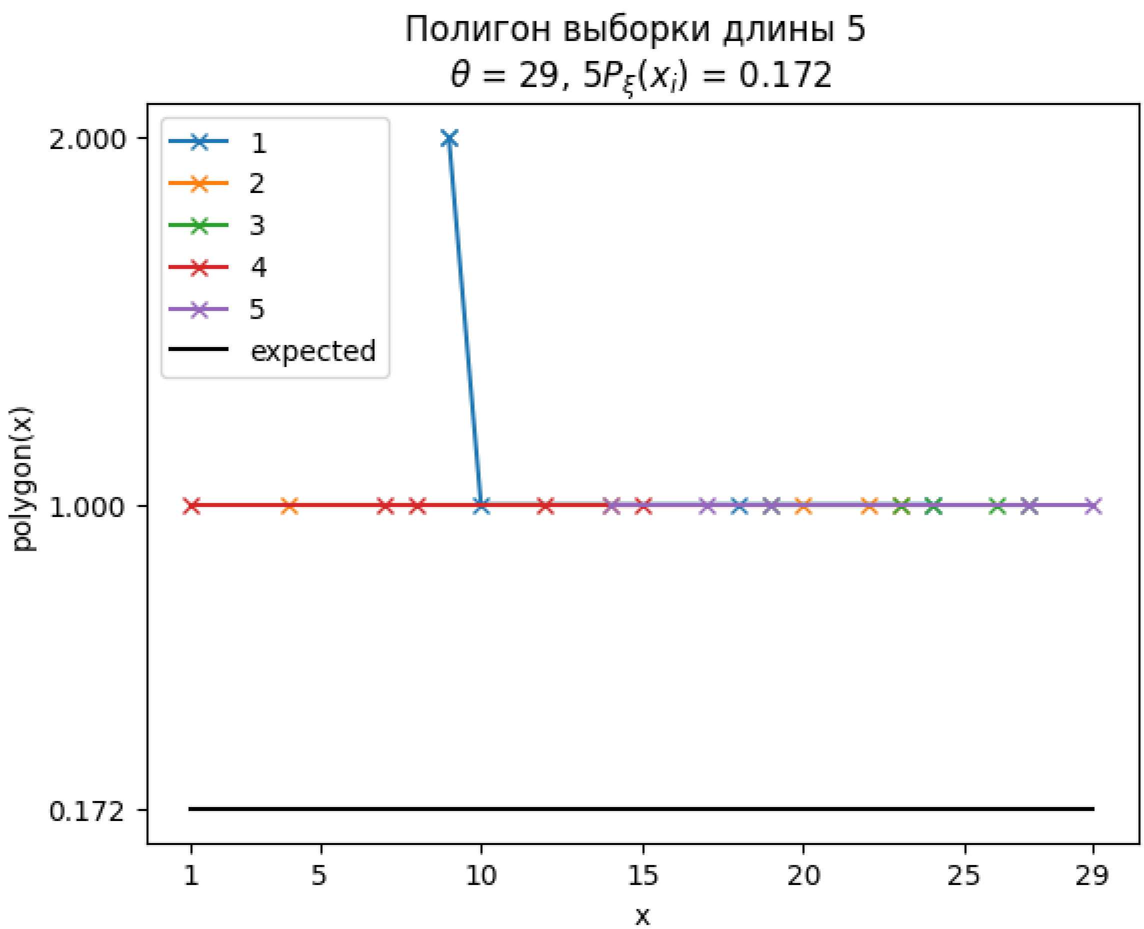
Чтобы сравнить график полигона частот и график функции вероятности нужно в идеале изобразить один на другом, при этом как-то правильно их друг с другом соотнести, чтобы они правильно наложились друг на друга. Здесь я попробую подвести график вероятности к графику полигона частот (то есть второй останется неизменным, а первый будет домножен). Чтобы вычислить коэффициент домножения рассмотрим $\hat{F}(x)$:

$\hat{F}(x) = \frac{1}{n}\sum_{i=1}^{n} I(x_i \leq x) \Rightarrow \hat{P}(x_i) = \hat{F}(x_i) - \hat{F}(x_{i-1}) = \frac{1}{n} k\, I(x_i = x_i) = \frac{k}{n}$, где k - частота встречаемости элемента в выборке.

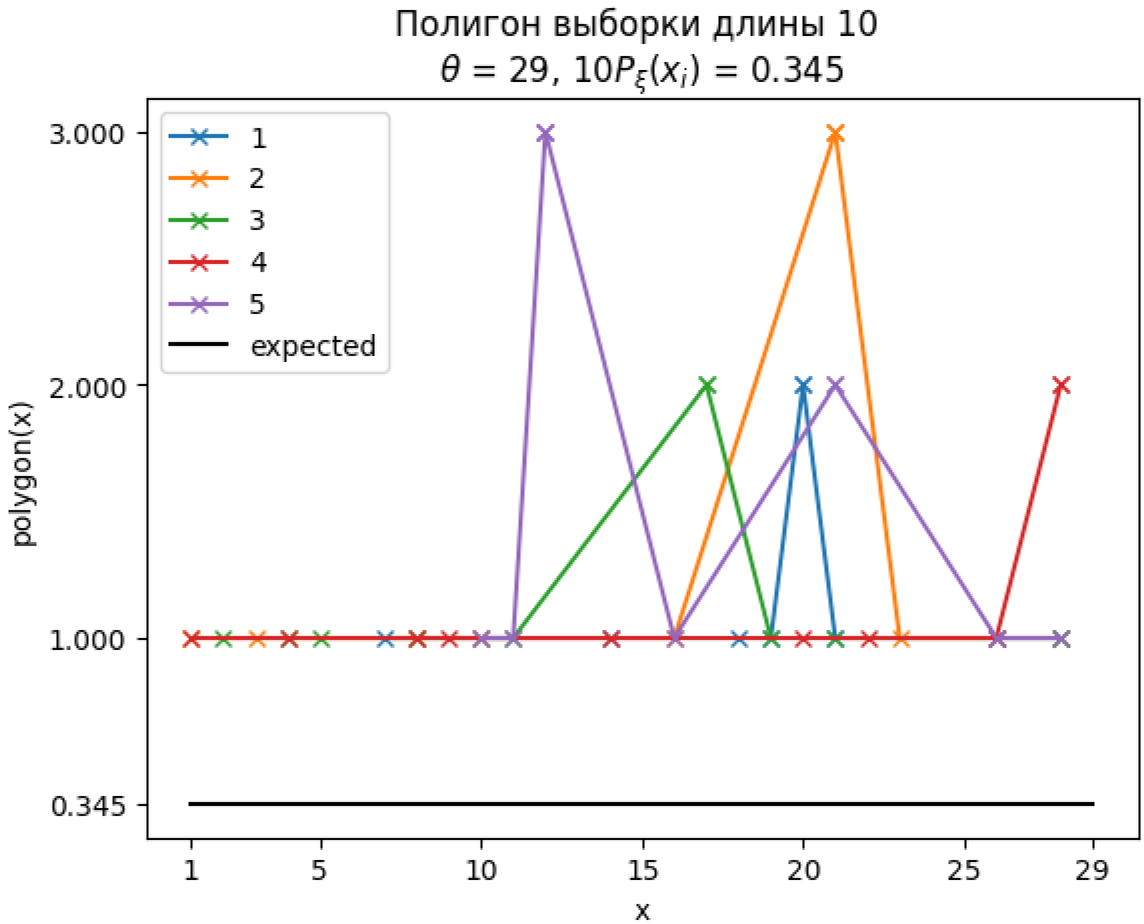
$\Rightarrow$ при $n \to \infty$ $\hat{P}(x_i) \to P(x_i) \Rightarrow P(x_i) = \frac{k}{n} \Rightarrow k = nP(x_i)$ при $n \to \infty$, что значит, что график вероятности подводим к полигону частот через домножение на количество элементов в выборке.

In [10]:
```
def xi_pilygon(sample, x):
    return np.count_nonzero(sample==x)

def min_t(samples):
    res = samples[0][0]
    for i in samples:
        t = min(i)
        if t < res: res = t
    return res

def max_t(samples):
    res = samples[0][0]
    for i in samples:
        t = max(i)
        if t > res: res = t
    return res

Y_polxi = [[np.array([xi_pilygon(sample_xi[k][j], sample_xi[k][j][i])
                      for i in range(n[k])])
            for j in range(5)] for k in range(len(n))]

y_limsxi = np.array([[min_t(j), max_t(j)] for j in Y_polxi])

posibilityxi = np.full((1000), 1/29)
```

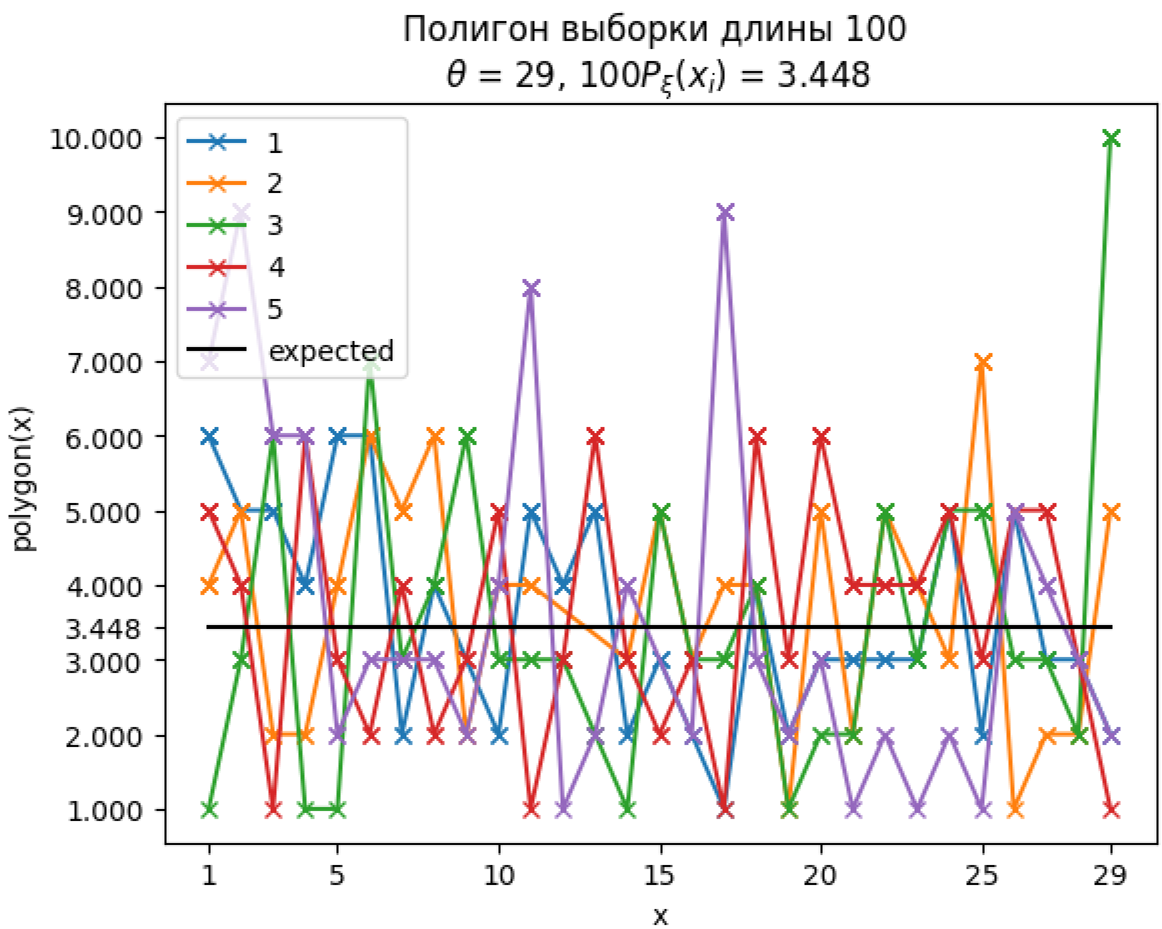
In [11]:
```python
# n=5
for i in range(5):
    plt.plot(sample_xi[0][i], Y_polxi[0][i], '-x')
plt.plot(np.linspace(1, 29, 1000), posibilityxi*5, 'k')
plt.xticks([1]+list(range(5,26,5))+[29])
plt.yticks(np.append(np.arange(y_limsxi[0,0], y_limsxi[0,1]+1), 5/29))
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 5 \n' +
          '$\\theta$ = 29, $5P_{\\xi}(x_i)$ = %.3f'% (5/29))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



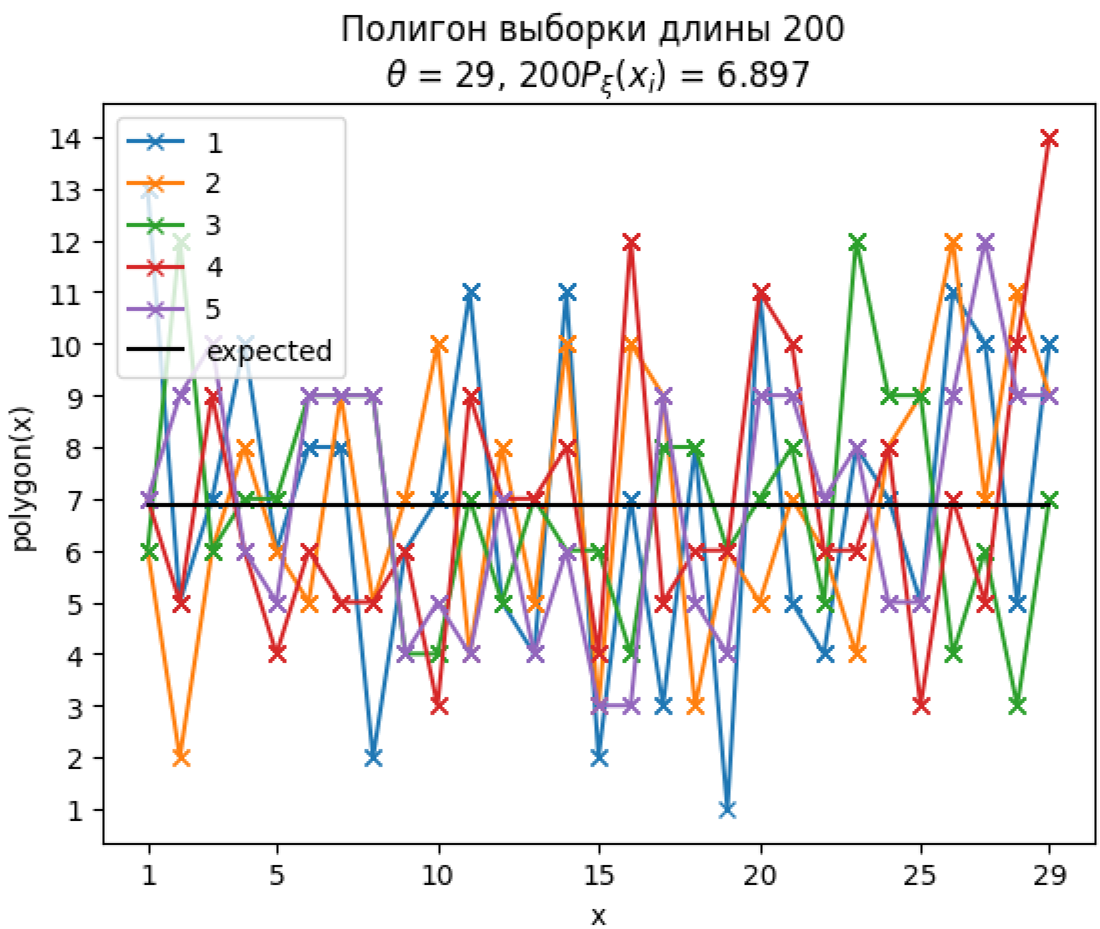
In [12]:
```
# n=10
for i in range(5):
    plt.plot(sample_xi[1][i], Y_polxi[1][i], '-x')
plt.plot(np.linspace(1, 29, 1000), posibilityxi*10, 'k')
plt.xticks([1]+list(range(5,26,5))+[29])
plt.yticks(np.append(np.arange(y_limsxi[1,0], y_limsxi[1,1]+1), 10/29))
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 10 \n' +
          '$\\theta$ = 29, $10P_{\\xi}(x_i)$ = %.3f'% (10/29))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



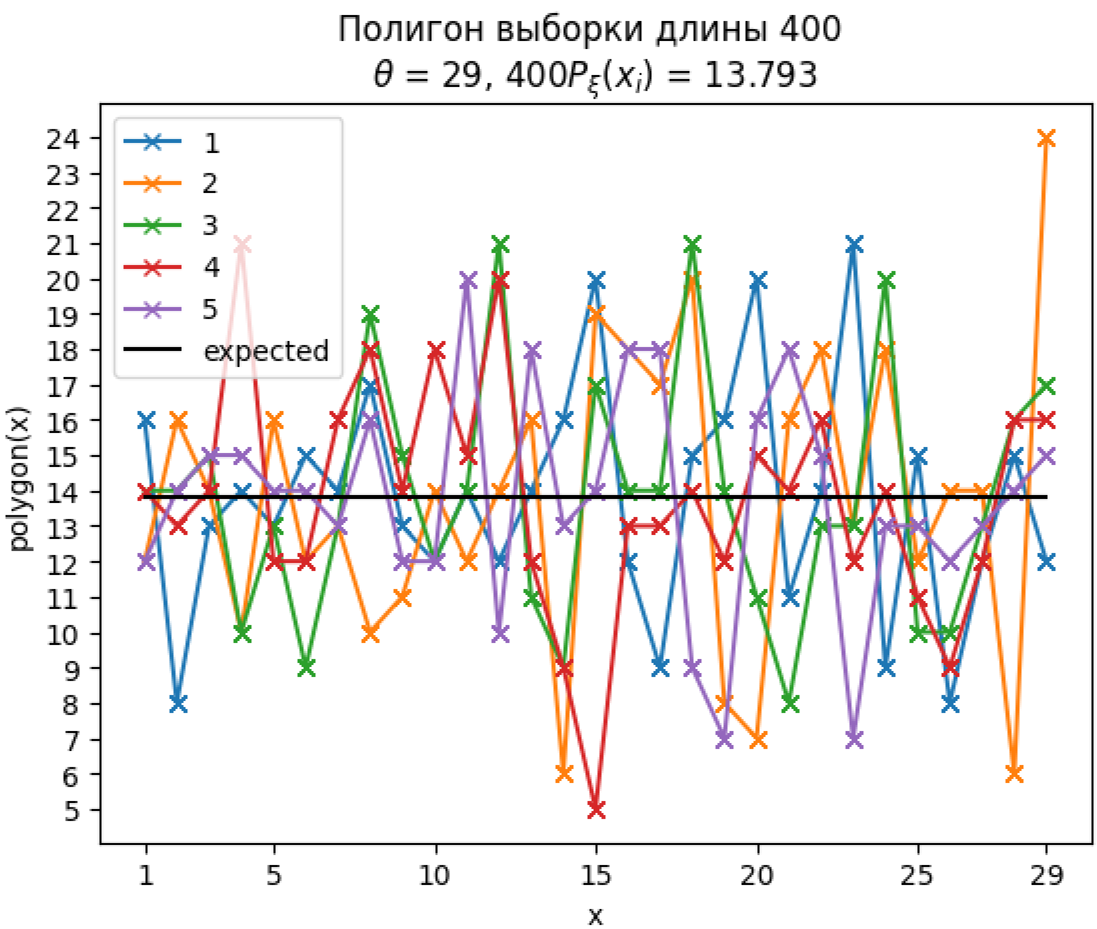
In [13]:
```python
# n=100
for i in range(5):
    plt.plot(sample_xi[2][i], Y_polxi[2][i], '-x')
plt.plot(np.linspace(1, 29, 1000), posibilityxi*100, 'k')
plt.xticks([1]+list(range(5,26,5))+[29])
plt.yticks(np.append(np.arange(y_limsxi[2,0], y_limsxi[2,1]+1), 100/29))
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 100 \n' +
          '$\\theta$ = 29, $100P_{\\xi}(x_i)$ = %.3f'% (100/29))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



In [14]:
```
# n=200
for i in range(5):
    plt.plot(sample_xi[3][i], Y_polxi[3][i], '-x')
plt.plot(np.linspace(1, 29, 1000), posibilityxi*200, 'k')
plt.xticks([1]+list(range(5,26,5))+[29])
plt.yticks(np.arange(y_limsxi[3,0], y_limsxi[3,1]+1))
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 200 \n' +
          '$\\theta$ = 29, $200P_{\\xi}(x_i)$ = %.3f'% (200/29))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```

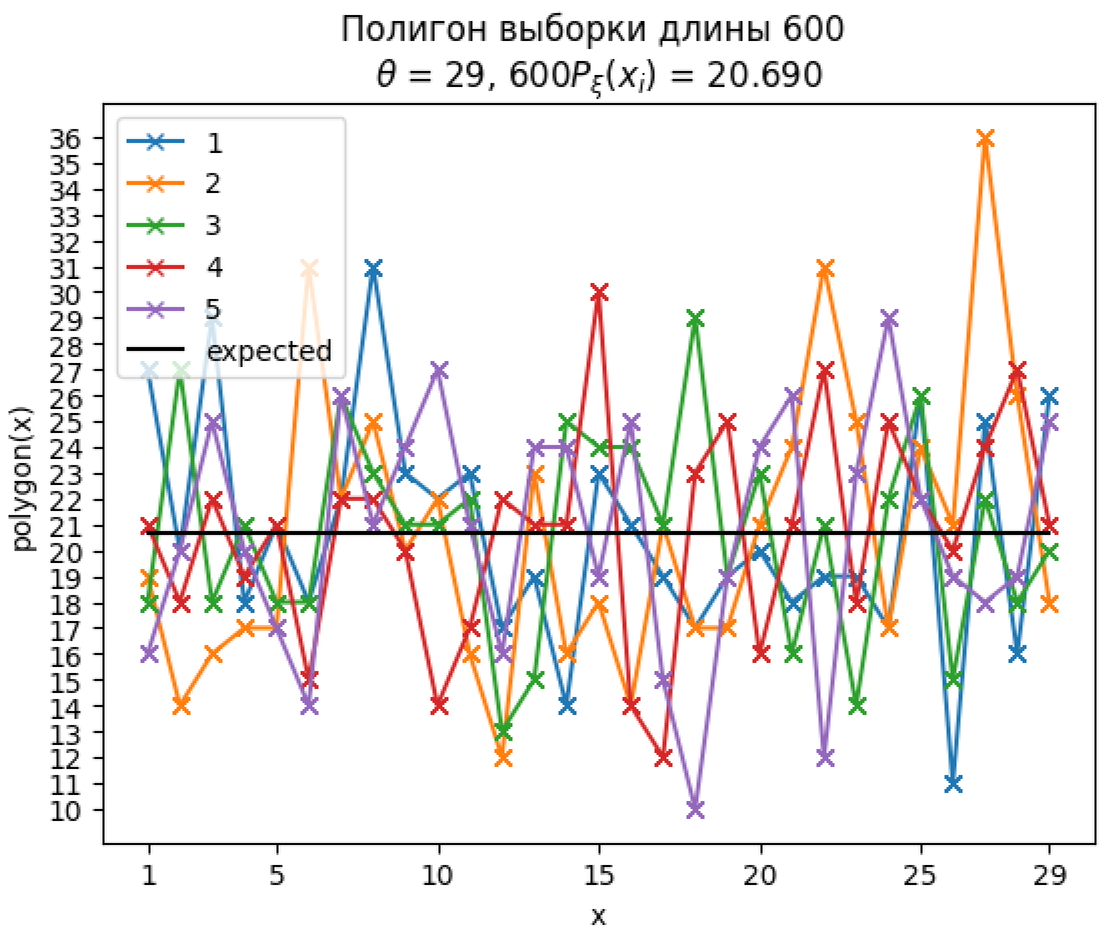


In [15]:
```
# n=400
for i in range(5):
    plt.plot(sample_xi[4][i], Y_polxi[4][i], '-x')
plt.plot(np.linspace(1, 29, 1000), posibilityxi*400, 'k')
plt.xticks([1]+list(range(5,26,5))+[29])
plt.yticks(np.arange(y_limsxi[4,0], y_limsxi[4,1]+1))
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 400 \n' +
          '$\\theta$ = 29, $400P_{\\xi}(x_i)$ = %.3f'% (400/29))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```

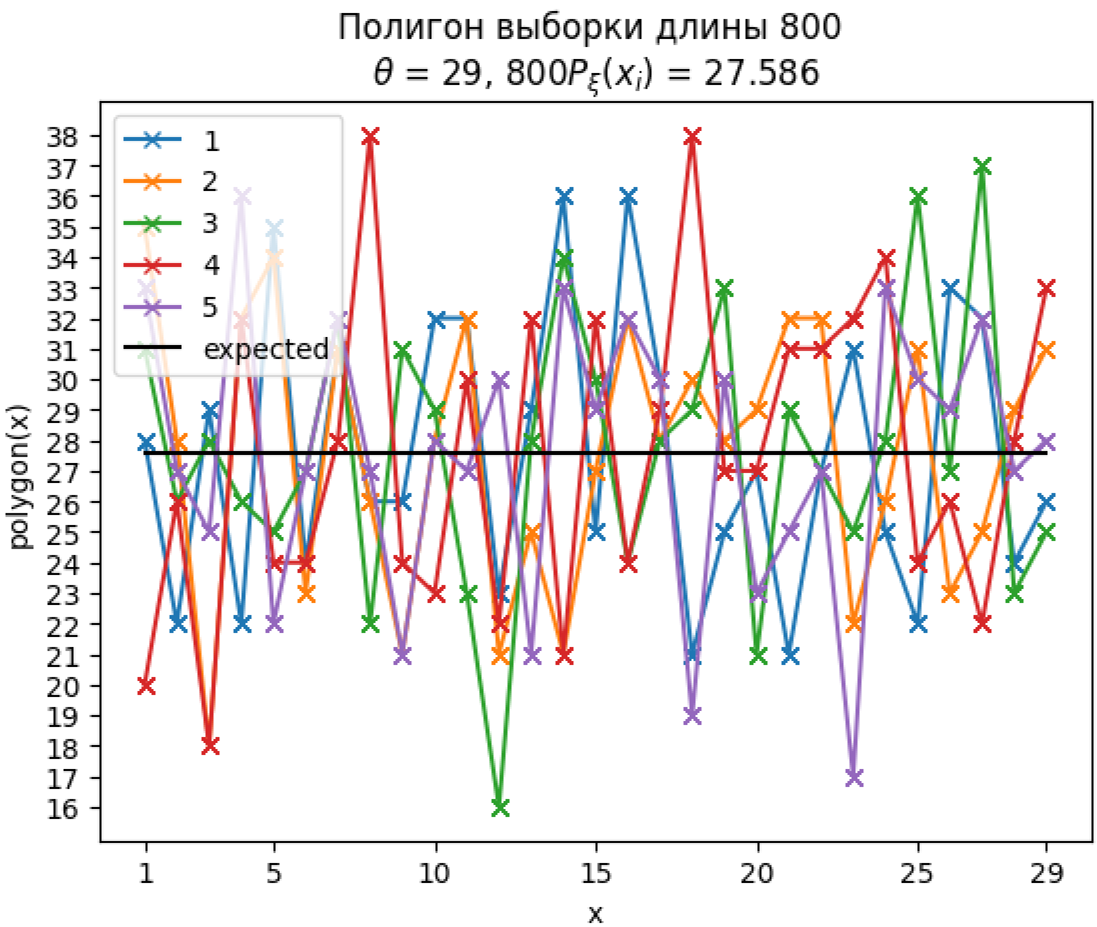


In [16]:

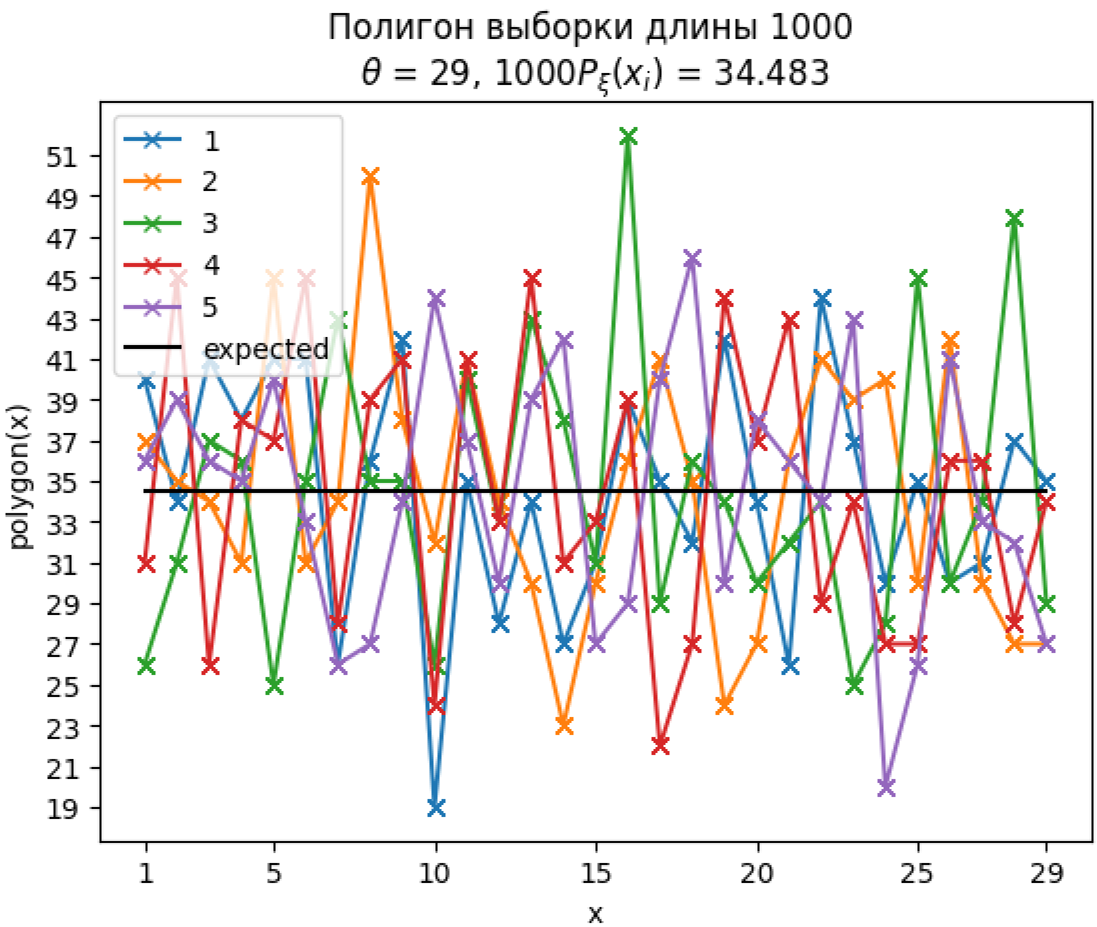
```
# n=600
for i in range(5):
    plt.plot(sample_xi[5][i], Y_polxi[5][i], '-x')
plt.plot(np.linspace(1, 29, 1000), posibilityxi*600, 'k')
plt.xticks([1]+list(range(5,26,5))+[29])
plt.yticks(np.arange(y_limsxi[5,0], y_limsxi[5,1]+1))
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 600 \n' +
          '$\\theta$ = 29, $600P_{\\xi}(x_i)$ = %.3f'% (600/29))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



In [17]:
```
# n=800
for i in range(5):
    plt.plot(sample_xi[6][i], Y_polxi[6][i], '-x')
plt.plot(np.linspace(1, 29, 1000), posibilityxi*800, 'k')
plt.xticks([1]+list(range(5,26,5))+[29])
plt.yticks(np.arange(y_limsxi[6,0], y_limsxi[6,1]+1))
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 800 \n' +
          '$\\theta$ = 29, $800P_{\\xi}(x_i)$ = %.3f'% (800/29))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



In [18]:
```python
# n=1000
for i in range(5):
    plt.plot(sample_xi[7][i], Y_polxi[7][i], '-x')
plt.plot(np.linspace(1, 29, 1000), posibilityxi*1000, 'k')
plt.xticks([1]+list(range(5,26,5))+[29])
plt.yticks(np.arange(y_limsxi[7,0], y_limsxi[7,1]+1, 2))
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 1000 \n' +
          '$\\theta$ = 29, $1000P_{\\xi}(x_i)$ = %.3f'% (1000/29))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



По полигону частот и графику домноженной вероятности можно заметить, что встречаемость возможных значений распределения держится вокруг вероятности каждого из них, и с увеличением числа элементов выборки эту закономерность наблюдать становится легче. Это наблюдение соответствует теореме о схождении эмперической вероятности к математической (в курсе лекций это теорема о функциях распределения, но вероятность из этого следует). Конечно, у нас мог произойти случай, когда все элементы выборки попали в 1, но шанс этого в силу построения равен 29^{-1000}, так что дополнительный разброс в виде 5 выборок на каждое указанное количество элементов создает общую картину, которая со стремлением n к бесконечности, устремит полигон частот к домноженному графику вероятности. Также следует заметить, что на графиках в 5 и 10 элементов выборки домноженная вероятность даже не близка к частотам. Это, во-первых, еще раз подтверждает теорему, а во-вторых, домноженная вероятность сильно меньше единицы, что еще больше мешает приближению.

Задание 4

```
In [19]: def xi_sample_mean(sample):
             return sum(sample)/len(sample)

         def xi_sample_variance(sample):
             return sum((sample-xi_sample_mean(sample))**2)/len(sample)

         meansxi = np.array([[xi_sample_mean(sample_xi[k][j])for j in range(5)]
                             for k in range(len(n))])
         variancesxi = np.array([[xi_sample_variance(sample_xi[k][j])for j in range(5)]
                                 for k in range(len(n))])
         means_pxi = np.array([['%.2f' % i for i in j] for j in meansxi])
         variances_pxi = np.array([['%.2f' % i for i in j] for j in variancesxi])
         expectationxi = (29+1)/2
         variancexi = (29*29-1)/12
         means_difxi = np.array([['%.2f' % i for i in j] for j in (meansxi-expectationxi)])
         variances_difxi = np.array([['%.2f' % i for i in j]
                                     for j in (variancesxi-variancexi)])
```

Для начала следует продемонстрировать выборочные средние и выборочные дисперсии, чтобы дать большее придставление о числах, с которыми идет работа. Благо их не так много.

In [20]:
```
fig, ax = plt.subplots(2,1, figsize=(7,6.5))
ax[0].table(cellText = means_pxi, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('Выборочные средние');
ax[1].table(cellText = variances_pxi, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Выборочные дисперсии');
```

Выборочные средние

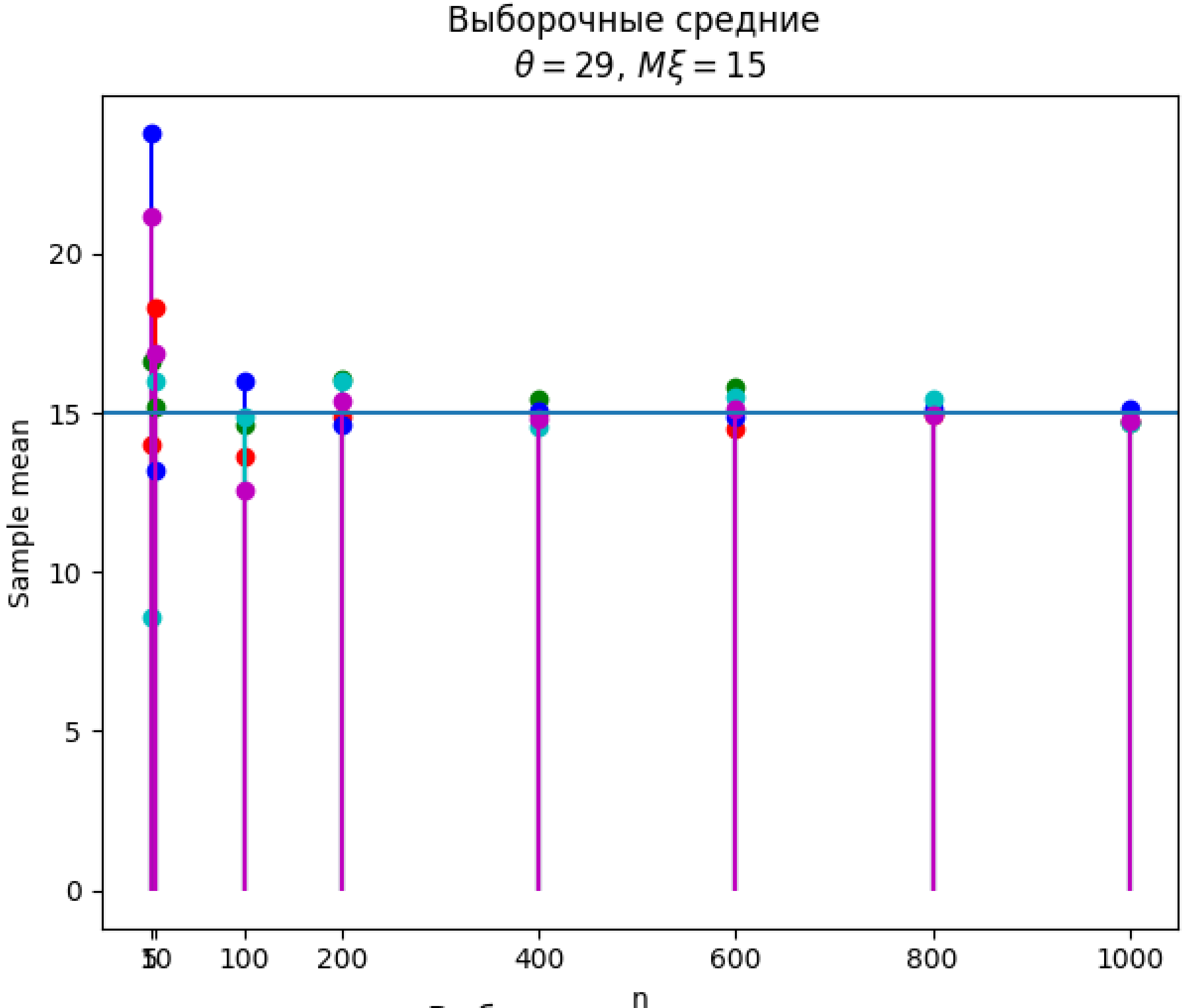
	1	2	3	4	5
5	14.00	16.60	23.80	8.60	21.20
10	18.30	15.20	13.20	16.00	16.90
100	13.60	14.65	15.98	14.88	12.58
200	14.88	16.07	14.62	15.99	15.39
400	14.97	15.41	15.09	14.59	14.79
600	14.49	15.78	14.88	15.47	15.12
800	14.94	15.01	15.14	15.45	14.93
1000	14.78	14.67	15.12	14.67	14.73

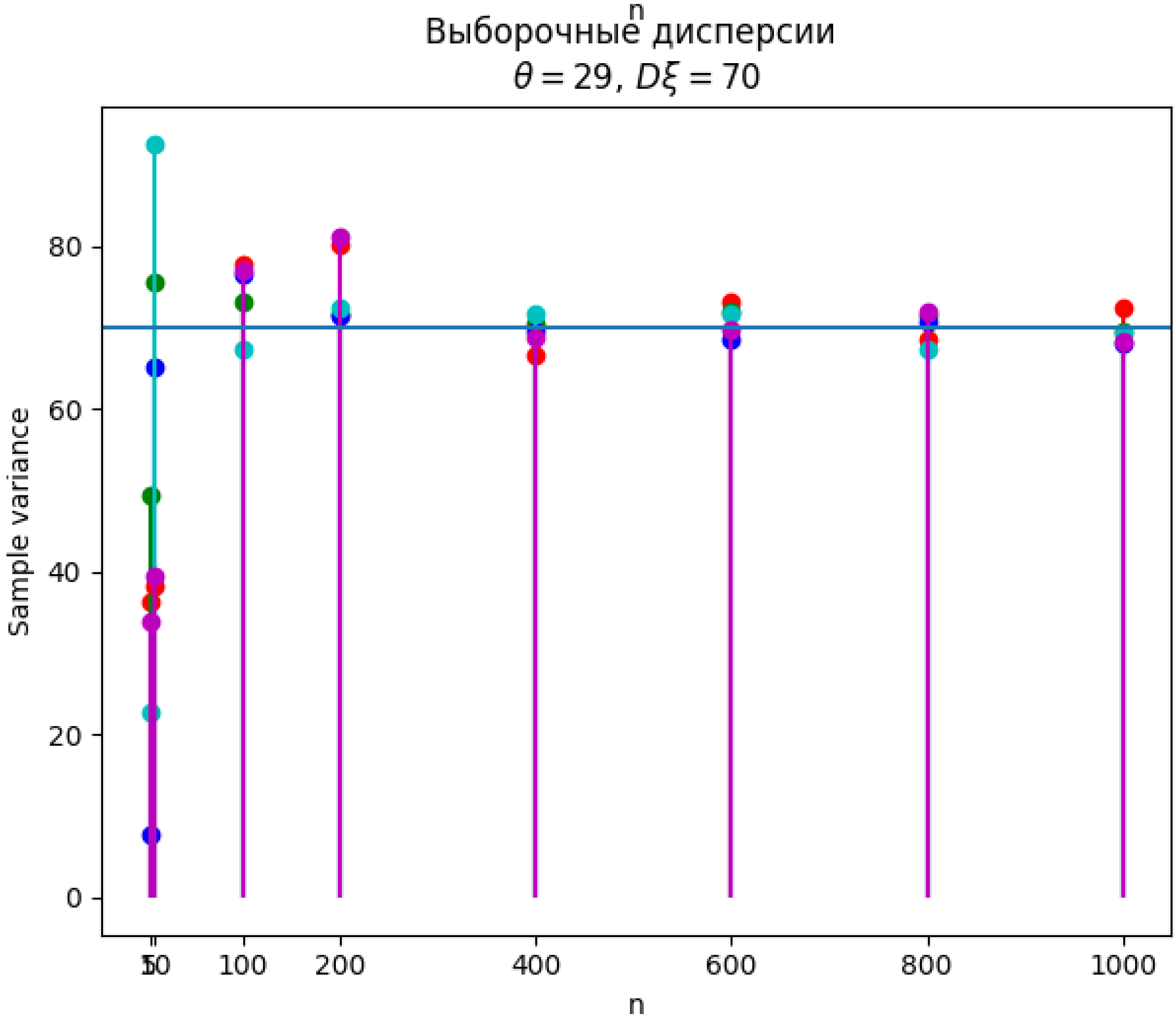
Выборочные дисперсии

	1	2	3	4	5
5	36.40	49.44	7.76	22.64	33.76
10	38.21	75.56	65.16	92.60	39.49
100	77.64	73.25	76.46	67.25	77.12
200	80.09	71.68	71.52	72.36	81.26
400	66.62	70.47	69.57	71.61	68.87
600	73.17	71.88	68.62	71.70	69.70
800	68.51	71.42	70.76	67.28	71.93
1000	72.44	69.49	68.16	69.17	68.42

In [21]:

```
fig, ax = plt.subplots(2,1, figsize=(7,12))
for k in range(len(n)):
    for j in range(5):
        ax[0].stem(n[k], meansxi[k][j], 'rgbcm'[j])
ax[0].axhline(expectationxi)
ax[0].set(xticks = n, xlabel = 'n', ylabel = 'Sample mean',
          title = 'Выборочные средние \n' +
                  '$\\theta = 29, \\,M\\xi = %d$'%expectationxi)

for k in range(len(n)):
    for j in range(5):
        ax[1].stem(n[k], variancesxi[k][j], 'rgbcm'[j])
ax[1].axhline(y = variancexi)
ax[1].set(xticks = n, xlabel = 'n', ylabel = 'Sample variance',
          title = 'Выборочные дисперсии \n' +
                  '$\\theta = 29, \\,D\\xi = %d$'%variancexi);
```


Выборочные средние
$\theta = 29$, $M\xi = 15$
Sample mean
n


Выборочные дисперсии
$\theta = 29$, $D\xi = 70$
Sample variance
n

По графикам можно заметить, что с увеличением количества элементов выборки и математическое ожидание, и дисперсия сходятся к предполагаемому значению. Что еще раз подтверждает теорему, упомянутую в предыдущей задаче.

In [22]:
```
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = means_difxi, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('Смещение оценки: $b(\\theta)=M_{\\theta}T(x)-\\tau(\\theta)$' +
                '\n $M\\xi = $'+str(expectationxi));

ax[1].table(cellText = variances_difxi, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Разница выборочной дисперсии и дисперсии' +
                '\n $D\\xi = $'+str(variancexi));
```

Смещение оценки: $b(\theta) = M_\theta T(x) - \tau(\theta)$
$M\xi = 15.0$

	1	2	3	4	5
5	-1.00	1.60	8.80	-6.40	6.20
10	3.30	0.20	-1.80	1.00	1.90
100	-1.40	-0.35	0.98	-0.12	-2.42
200	-0.12	1.07	-0.38	0.99	0.39
400	-0.03	0.41	0.09	-0.41	-0.21
600	-0.51	0.78	-0.12	0.47	0.12
800	-0.06	0.01	0.14	0.45	-0.07
1000	-0.22	-0.33	0.12	-0.33	-0.27

Разница выборочной дисперсии и дисперсии
$D\xi = 70.0$

	1	2	3	4	5
5	-33.60	-20.56	-62.24	-47.36	-36.24
10	-31.79	5.56	-4.84	22.60	-30.51
100	7.64	3.25	6.46	-2.75	7.12
200	10.09	1.68	1.52	2.36	11.26
400	-3.38	0.47	-0.43	1.61	-1.13
600	3.17	1.88	-1.38	1.70	-0.30
800	-1.49	1.42	0.76	-2.72	1.93
1000	2.44	-0.51	-1.84	-0.83	-1.58

По таблице смещения оценок можно заметить, что с увеличением числа элементов выборки, модуль смещения оценки уменьшается.

Также эти таблицы (как и предыдущее много чего) еще раз демонстрируют справедливость теоремы о схождении функций распределения.

$\sqsupset X_1, \ldots, X_n \sim \xi$

Свойства оценки $\overline{X} = \frac{1}{n}\sum_{i=1}^n X_i$ **:**

$\tau(\theta) = M\xi, T(x) = \overline{X} = \frac{1}{n}\sum_{i=1}^n X_i.$

$M_\theta T(x) = M(\frac{1}{n}\sum_{i=1}^n X_i) = \frac{1}{n}\sum_{i=1}^n MX_i = \frac{1}{n}\cdot n \cdot M\xi = M\xi = \tau(\theta) \Rightarrow$ оценка $\overline{X}$ является **несмещенной**

$DT(x) = D(\frac{1}{n}\sum_{i=1}^n X_i) = \frac{1}{n^2}\sum_{i=1}^n DX_i = \frac{n}{n^2}DX_i = \frac{D\xi}{n}\xrightarrow{n\to\infty} 0 \Rightarrow$ оценка $\overline{X}$ является **состоятельной**

Свойства оценки $\overline{S}^2 = \frac{1}{n}\sum_{i=1}^n (X_i - \overline{X})^2$ **:**

$\tau(\theta) = D\xi, T(x) = \overline{S}^2 = \frac{1}{n}\sum_{i=1}^n (X_i - \overline{X})^2.$

$$
\begin{aligned}
M_\theta T(x) &= M(\frac{1}{n}\sum_{i=1}^n (X_i - \overline{X})^2) = \left|MX_i = M\overline{X}\right| = M(\frac{1}{n}\sum_{i=1}^n (X_i - MX_i + M\overline{X} - \overline{X})^2) = \\
&= \left|Y_i = X_i - MX_i,\ \overline{Y} = \overline{X} - M\overline{X}\right| = M(\frac{1}{n}\sum_{i=1}^n (Y_i - \overline{Y})^2) = M(\frac{1}{n}\sum_{i=1}^n (Y_i^2 + \overline{Y}^2 - 2Y_i\overline{Y})) = \\
&= \left|\overline{Y} = \frac{1}{n}\sum_{i=1}^n Y_i\right| = \frac{1}{n}M(\sum_{i=1}^n Y_i^2 + \sum_{i=1}^n \frac{1}{n^2}\sum_{j,k=1}^n Y_jY_k - \frac{2}{n}\sum_{i=1}^n\sum_{j=1}^n Y_iY_j) = \\
&= \frac{1}{n}M(\sum_{i=1}^n Y_i^2 + \frac{n}{n^2}\sum_{i,j=1}^n Y_iY_j - \frac{2}{n}\sum_{i,j=1}^n Y_iY_j) = \frac{1}{n}M(\sum_{i=1}^n Y_i^2 - \frac{1}{n}\sum_{i,j=1}^n Y_iY_j) = \\
&= \frac{1}{n}(\sum_{i=1}^n MY_i^2 - \frac{1}{n}\sum_{i,j=1}^n M(Y_iY_j)) = \frac{1}{n}(\sum_{i=1}^n MY_i^2 - \frac{1}{n}\sum_{i\neq j}^n MY_iMY_j - \frac{1}{n}\sum_{i=1}^n MY_i^2) = \\
&= \frac{1}{n}(nMY_i^2 - \frac{1}{n}\sum_{i\neq j}^n MY_iMY_j - MY_i^2) = \left|MY_i = MX_i - M(MX_i) = MX_i - MX_i = 0\right| = \\
&= \frac{n-1}{n}MY_i^2 = \left|DY_i = MY_i^2 - (MY_i)^2 = MY_i^2\right| = \frac{n-1}{n}DY_i \neq DY_i
\end{aligned}
$$

$\Rightarrow$ оценка $\overline{S}^2$ **не** является **несмещенной**

Следует заметить, что в описанной математике выше само дискретное равномерное распределение упоминается не более чем символом ξ, что означает что простая замена ξ на η сделает ровно ту же теорию для абсолютно непрерывного треугольного распределения. По этой причине в номере 4 для непрерывного распределения эти выкладки произведены не будут.

Абсолютно непрерывное

Большая часть материала уже разобрана выше на примере дискретного случая. Поэтому здесь различных описаний будет меньше, так как они будут почти 1 в 1 повторяться. Это самое почти, при необходимости, и будет описываться.

Задание 1

In [23]:
```
#sample_eta = [[np.sort(np.array([generate_eta() for i in range(j)]))
#               for i in range(5)] for j in n]
# BEGIN сериализация
if 'sample_eta.pkl' in os.listdir():
    with open('sample_eta.pkl', 'rb') as f:
        sample_eta = pickle.load(f)
else:
    sample_eta = [[np.sort(np.array([generate_eta() for i in range(j)]))
                   for i in range(5)] for j in n]
    with open('sample_eta.pkl', 'wb') as f:
        pickle.dump(sample_eta, f)
# END сериализация

#demo_peta = np.array(['%.3f'%i for i in sample_eta[7][0]]).reshape((-1, 20))
for i in range(len(n)):
    demo_peta = ['%.3f'%j for j in sample_eta[i][0]]
    print('Пример сгенерированной выборки длины %d:'%n[i], end=' ')
    print(*demo_peta)
    print('-'*10)
```

Пример сгенерированной выборки длины 5: 0.511 0.514 0.526 0.528 0.544

Пример сгенерированной выборки длины 10: 0.298 0.352 0.456 0.457 0.463 0.561 0.565 0.663 0.684 0.780

Пример сгенерированной выборки длины 100: 0.104 0.111 0.127 0.135 0.138 0.155 0.174 0.178 0.186 0.203 0.222 0.230 0.237 0.253 0.254 0.261 0.261 0.275 0.277 0.279 0.284 0.286 0.291 0.297 0.327 0.328 0.334 0.334 0.335 0.345 0.352 0.352 0.362 0.367 0.376 0.379 0.387 0.409 0.420 0.423 0.424 0.430 0.433 0.434 0.434 0.437 0.440 0.443 0.443 0.462 0.471 0.485 0.493 0.496 0.502 0.504 0.506 0.512 0.512 0.524 0.529 0.545 0.547 0.548 0.566 0.570 0.575 0.584 0.591 0.596 0.607 0.613 0.616 0.637 0.648 0.651 0.658 0.666 0.668 0.670 0.684 0.690 0.694 0.694 0.695 0.699 0.706 0.725 0.730 0.731 0.736 0.741 0.782 0.786 0.816 0.854 0.857 0.878 0.957 0.961

Пример сгенерированной выборки длины 200: 0.012 0.049 0.071 0.106 0.131 0.162 0.168 0.181 0.190 0.200 0.203 0.207 0.208 0.209 0.216 0.217 0.219 0.225 0.231 0.232 0.239 0.240 0.245 0.246 0.250 0.253 0.256 0.265 0.267 0.270 0.281 0.282 0.286 0.290 0.290 0.291 0.296 0.296 0.300 0.306 0.307 0.308 0.315 0.326 0.327 0.329 0.331 0.336 0.340 0.342 0.346 0.356 0.359 0.363 0.366 0.373 0.382 0.382 0.383 0.383 0.384 0.387 0.388 0.388 0.389 0.391 0.399 0.399 0.400 0.401 0.412 0.417 0.419 0.419 0.420 0.422 0.422 0.424 0.424 0.426 0.427 0.427 0.439 0.440 0.444 0.444 0.445 0.449 0.452 0.461 0.462 0.462 0.462 0.465 0.467 0.469 0.470 0.470 0.474 0.477 0.481 0.483 0.486 0.489 0.491 0.491 0.492 0.492 0.497 0.500 0.505 0.508 0.510 0.518 0.519 0.521 0.522 0.526 0.526 0.528 0.531 0.537 0.540 0.541 0.542 0.552 0.552 0.563 0.564 0.565 0.567 0.571 0.571 0.576 0.580 0.582 0.585 0.591 0.591 0.593 0.595 0.599 0.610 0.614 0.621 0.623 0.623 0.630 0.632 0.640 0.642 0.644 0.652 0.657 0.661 0.661 0.663 0.667 0.678 0.679 0.682 0.687 0.688 0.691 0.695 0.697 0.698 0.706 0.716 0.716 0.721 0.734 0.741 0.745 0.752 0.756 0.760 0.764 0.768 0.774 0.777 0.779 0.789 0.798 0.809 0.815 0.815 0.824 0.829 0.832 0.848 0.850 0.854 0.858 0.860 0.900 0.908 0.926 0.932 0.973

Пример сгенерированной выборки длины 400: 0.024 0.041 0.079 0.104 0.115 0.120 0.122 0.122 0.131 0.131 0.136 0.146 0.151 0.153 0.154 0.158 0.159 0.162 0.165 0.165 0.169 0.170 0.171 0.175 0.184 0.185 0.189 0.191 0.192 0.193 0.195 0.196 0.198 0.209 0.213 0.220 0.220 0.222 0.224 0.227 0.233 0.234 0.241 0.241 0.245 0.248 0.249 0.251 0.253 0.257 0.257 0.258 0.265 0.275 0.275 0.278 0.283 0.285 0.286 0.287 0.288 0.288 0.288 0.290 0.291 0.292 0.292 0.292 0.293 0.296 0.301 0.303 0.304 0.305 0.308 0.309 0.310 0.311 0.312 0.313 0.313 0.313 0.314 0.314 0.316 0.316 0.317 0.317 0.321 0.324 0.329 0.331 0.331 0.333 0.336 0.338 0.338 0.339 0.343 0.344 0.346 0.347 0.349 0.349 0.351 0.354 0.355 0.355 0.355 0.357 0.358 0.362 0.364 0.365 0.365 0.366 0.367 0.372 0.373 0.375 0.380 0.381 0.382 0.384 0.385 0.385 0.387 0.388 0.389 0.390 0.391 0.394 0.396 0.398 0.398 0.399 0.401 0.402 0.403 0.405 0.406 0.407 0.408 0.416 0.417 0.419 0.421 0.421 0.421 0.422 0.423 0.425 0.428 0.430 0.430 0.432 0.435 0.435 0.441 0.441 0.442 0.443 0.445 0.445 0.447 0.448 0.449 0.453 0.454 0.456 0.456 0.458 0.459 0.459 0.460 0.462 0.463 0.467 0.469 0.470 0.471 0.472 0.473 0.475 0.477 0.478 0.479 0.480 0.482 0.484 0.486 0.486 0.486 0.486 0.489 0.489 0.490 0.491 0.492 0.492 0.493 0.498 0.499 0.502 0.503 0.503 0.504 0.504 0.505 0.505 0.506 0.506 0.508 0.509 0.510 0.511 0.513 0.513 0.514 0.515 0.515 0.516 0.516 0.518 0.523 0.523 0.523 0.524 0.525 0.525 0.525 0.526 0.526 0.529 0.529 0.530 0.530 0.531 0.534 0.535 0.535 0.538 0.540 0.542 0.544 0.544 0.545 0.547 0.547 0.548 0.553 0.559 0.560 0.561 0.562 0.564 0.565 0.566 0.568 0.569 0.569 0.569 0.570 0.570 0.571 0.574 0.574 0.578 0.581 0.581 0.582 0.584 0.586 0.588 0.590 0.590 0.592 0.593 0.594 0.597 0.598 0.599 0.600 0.600 0.606 0.609 0.609 0.613 0.613 0.613 0.615 0.615 0.619 0.619 0.620 0.620 0.621 0.630 0.637 0.638 0.640 0.642 0.643 0.644 0.645 0.645 0.646 0.650 0.651 0.651 0.653 0.655 0.655 0.656 0.658 0.659 0.659 0.660 0.661 0.662 0.662 0.665 0.667 0.670 0.671 0.675 0.676 0.677 0.681 0.682 0.682 0.684 0.685 0.686 0.688 0.695 0.696 0.700 0.702 0.704 0.707 0.711 0.712 0.714 0.717 0.717 0.717 0.719 0.721 0.722 0.731 0.732 0.732 0.733 0.737 0.737 0.737 0.738 0.739 0.739 0.741 0.742 0.745 0.749 0.749 0.750 0.755 0.755 0.757 0.761 0.772 0.777 0.786 0.787 0.791 0.794 0.795 0.803 0.806 0.80

8 0.809 0.809 0.815 0.826 0.836 0.837 0.840 0.841 0.851 0.854 0.854 0.860 0.866 0.884 0.897 0.902 0.909 0.912 0.915 0.951

Пример сгенерированной выборки длины 600: 0.029 0.041 0.044 0.055 0.059 0.060 0.064 0.066 0.069 0.072 0.075 0.077 0.085 0.095 0.096 0.099 0.104 0.105 0.117 0.117 0.124 0.125 0.127 0.130 0.130 0.131 0.135 0.138 0.139 0.142 0.143 0.143 0.144 0.145 0.148 0.149 0.150 0.150 0.152 0.156 0.157 0.157 0.166 0.167 0.168 0.177 0.183 0.187 0.187 0.188 0.190 0.194 0.194 0.195 0.195 0.200 0.202 0.202 0.207 0.207 0.208 0.210 0.211 0.212 0.212 0.212 0.214 0.216 0.217 0.217 0.221 0.227 0.229 0.231 0.232 0.232 0.233 0.234 0.239 0.240 0.240 0.243 0.244 0.246 0.246 0.248 0.250 0.252 0.253 0.253 0.256 0.256 0.258 0.259 0.259 0.260 0.261 0.262 0.263 0.264 0.264 0.267 0.268 0.269 0.270 0.271 0.271 0.273 0.274 0.274 0.276 0.277 0.278 0.279 0.280 0.287 0.287 0.289 0.291 0.293 0.294 0.294 0.295 0.296 0.298 0.298 0.300 0.300 0.301 0.303 0.304 0.306 0.308 0.308 0.310 0.311 0.313 0.314 0.314 0.315 0.316 0.317 0.318 0.318 0.319 0.320 0.321 0.325 0.329 0.329 0.333 0.333 0.334 0.336 0.339 0.339 0.340 0.342 0.342 0.342 0.343 0.343 0.343 0.345 0.346 0.348 0.348 0.348 0.351 0.353 0.355 0.355 0.355 0.357 0.358 0.361 0.362 0.362 0.365 0.366 0.366 0.366 0.367 0.367 0.368 0.368 0.369 0.369 0.370 0.373 0.376 0.377 0.378 0.379 0.380 0.381 0.385 0.386 0.386 0.387 0.388 0.388 0.389 0.390 0.391 0.393 0.393 0.395 0.396 0.396 0.397 0.398 0.398 0.398 0.399 0.400 0.402 0.402 0.403 0.404 0.404 0.405 0.406 0.406 0.406 0.406 0.408 0.409 0.410 0.410 0.410 0.411 0.413 0.413 0.414 0.414 0.416 0.417 0.418 0.419 0.422 0.423 0.424 0.424 0.425 0.428 0.428 0.429 0.430 0.430 0.432 0.434 0.435 0.436 0.436 0.437 0.437 0.439 0.439 0.441 0.443 0.444 0.444 0.445 0.445 0.446 0.448 0.448 0.448 0.449 0.449 0.450 0.453 0.455 0.455 0.455 0.455 0.456 0.458 0.458 0.459 0.460 0.461 0.462 0.462 0.462 0.464 0.465 0.465 0.468 0.468 0.468 0.469 0.469 0.470 0.471 0.472 0.473 0.473 0.473 0.474 0.474 0.474 0.475 0.475 0.475 0.480 0.482 0.482 0.482 0.482 0.482 0.482 0.483 0.484 0.485 0.486 0.486 0.487 0.487 0.488 0.488 0.490 0.490 0.491 0.491 0.492 0.492 0.492 0.496 0.496 0.498 0.499 0.500 0.500 0.500 0.501 0.501 0.504 0.505 0.505 0.507 0.507 0.509 0.509 0.511 0.513 0.513 0.514 0.516 0.518 0.519 0.519 0.520 0.522 0.522 0.524 0.524 0.524 0.525 0.525 0.527 0.528 0.531 0.532 0.534 0.536 0.537 0.537 0.538 0.539 0.539 0.539 0.540 0.544 0.544 0.545 0.545 0.545 0.546 0.549 0.550 0.550 0.552 0.553 0.556 0.557 0.557 0.558 0.558 0.558 0.560 0.564 0.565 0.567 0.567 0.568 0.568 0.569 0.570 0.571 0.571 0.574 0.574 0.575 0.576 0.576 0.577 0.577 0.578 0.579 0.579 0.580 0.581 0.582 0.584 0.586 0.586 0.590 0.591 0.591 0.591 0.591 0.591 0.593 0.593 0.594 0.594 0.595 0.595 0.596 0.598 0.599 0.600 0.601 0.602 0.603 0.604 0.604 0.604 0.604 0.605 0.606 0.607 0.607 0.608 0.611 0.613 0.614 0.615 0.616 0.616 0.617 0.620 0.620 0.620 0.621 0.623 0.626 0.628 0.630 0.630 0.633 0.634 0.636 0.638 0.639 0.640 0.641 0.642 0.642 0.647 0.647 0.649 0.649 0.651 0.651 0.652 0.654 0.656 0.660 0.660 0.661 0.663 0.663 0.664 0.665 0.666 0.666 0.674 0.675 0.677 0.677 0.678 0.678 0.678 0.682 0.685 0.686 0.686 0.687 0.688 0.689 0.692 0.692 0.695 0.696 0.697 0.697 0.699 0.703 0.711 0.713 0.721 0.721 0.722 0.725 0.728 0.728 0.729 0.730 0.736 0.738 0.741 0.745 0.745 0.745 0.746 0.746 0.746 0.747 0.748 0.751 0.753 0.754 0.755 0.756 0.764 0.764 0.764 0.768 0.770 0.770 0.771 0.771 0.777 0.777 0.783 0.785 0.791 0.793 0.793 0.795 0.795 0.796 0.797 0.800 0.801 0.803 0.805 0.806 0.811 0.812 0.815 0.817 0.817 0.820 0.822 0.825 0.826 0.832 0.832 0.840 0.842 0.843 0.844 0.848 0.848 0.858 0.862 0.866 0.869 0.874 0.878 0.882 0.886 0.891 0.894 0.896 0.902 0.903 0.906 0.929 0.939 0.940 0.942 0.944 0.945 0.962 0.984

Пример сгенерированной выборки длины 800: 0.004 0.009 0.024 0.025 0.038 0.045 0.051 0.062 0.067 0.079 0.086 0.087 0.088 0.088 0.089 0.094 0.100 0.101 0.102 0.106 0.107 0.111 0.112 0.116 0.119 0.121 0.121 0.127 0.130 0.132 0.135 0.136 0.139 0.141 0.141 0.142 0.143 0.143 0.143 0.145 0.150 0.150 0.150 0.152 0.153 0.156 0.157 0.158 0.158 0.161 0.162 0.164 0.165 0.165 0.167 0.168 0.170 0.171 0.174 0.175 0.176 0.177 0.178 0.179 0.180 0.181 0.183 0.188 0.191 0.192 0.195 0.199 0.200 0.200 0.202 0.204 0.204 0.204 0.206 0.209 0.209 0.212 0.215 0.217 0.217 0.217 0.219 0.219 0.220 0.221 0.222 0.222 0.223 0.223 0.224 0.224 0.225 0.226 0.227 0.227 0.230 0.231 0.231 0.232 0.232 0.233 0.239 0.240 0.242 0.243 0.244 0.244 0.245 0.245 0.248 0.248 0.249 0.250 0.250 0.252 0.252 0.252 0.252 0.252 0.253 0.254 0.255 0.255 0.255 0.259 0.260 0.261 0.262 0.263 0.267 0.267 0.267 0.269 0.269 0.269 0.269 0.270 0.271 0.271 0.272 0.272 0.279 0.281 0.281 0.281 0.283 0.286 0.287

0.287 0.289 0.289 0.289 0.290 0.291 0.291 0.296 0.297 0.298 0.299 0.300 0.301 0.301 0.301 0.303 0.303 0.304 0.305 0.309 0.309 0.310 0.311 0.311 0.312 0.312 0.313 0.314 0.314 0.315 0.315 0.316 0.317 0.318 0.320 0.321 0.321 0.324 0.325 0.325 0.326 0.326 0.327 0.327 0.328 0.328 0.331 0.333 0.333 0.336 0.336 0.337 0.337 0.340 0.340 0.341 0.341 0.341 0.342 0.342 0.345 0.345 0.346 0.348 0.348 0.348 0.349 0.349 0.350 0.351 0.352 0.352 0.353 0.353 0.354 0.355 0.356 0.356 0.357 0.357 0.358 0.360 0.360 0.361 0.362 0.363 0.363 0.363 0.364 0.365 0.365 0.366 0.369 0.369 0.369 0.370 0.370 0.370 0.371 0.371 0.374 0.374 0.375 0.375 0.375 0.375 0.376 0.376 0.376 0.376 0.377 0.380 0.380 0.381 0.382 0.382 0.383 0.386 0.386 0.387 0.388 0.389 0.390 0.390 0.391 0.392 0.392 0.393 0.394 0.394 0.395 0.396 0.397 0.397 0.398 0.399 0.399 0.399 0.399 0.399 0.400 0.402 0.403 0.404 0.404 0.405 0.406 0.406 0.407 0.408 0.409 0.409 0.409 0.409 0.411 0.412 0.412 0.412 0.413 0.414 0.415 0.417 0.418 0.421 0.422 0.423 0.423 0.423 0.424 0.426 0.427 0.427 0.428 0.429 0.429 0.430 0.431 0.431 0.432 0.432 0.432 0.432 0.434 0.435 0.435 0.435 0.438 0.438 0.438 0.439 0.440 0.442 0.442 0.442 0.442 0.443 0.444 0.445 0.445 0.448 0.448 0.449 0.449 0.450 0.451 0.451 0.452 0.453 0.453 0.455 0.455 0.457 0.457 0.457 0.457 0.457 0.459 0.459 0.459 0.459 0.460 0.461 0.462 0.462 0.465 0.466 0.466 0.467 0.467 0.467 0.469 0.469 0.470 0.471 0.471 0.471 0.472 0.472 0.472 0.474 0.475 0.476 0.476 0.479 0.479 0.480 0.480 0.481 0.481 0.482 0.482 0.482 0.482 0.483 0.484 0.485 0.485 0.486 0.487 0.488 0.488 0.489 0.489 0.490 0.490 0.491 0.491 0.491 0.491 0.492 0.492 0.493 0.493 0.494 0.494 0.495 0.497 0.501 0.501 0.502 0.502 0.503 0.503 0.504 0.504 0.505 0.506 0.508 0.508 0.510 0.510 0.510 0.511 0.512 0.512 0.513 0.513 0.514 0.514 0.514 0.514 0.515 0.516 0.516 0.518 0.519 0.520 0.520 0.521 0.524 0.524 0.525 0.525 0.525 0.525 0.526 0.526 0.526 0.527 0.527 0.528 0.528 0.528 0.530 0.531 0.531 0.532 0.532 0.533 0.533 0.533 0.533 0.534 0.534 0.534 0.535 0.535 0.537 0.537 0.538 0.539 0.539 0.540 0.540 0.541 0.541 0.541 0.543 0.544 0.544 0.546 0.547 0.547 0.548 0.551 0.552 0.552 0.552 0.552 0.553 0.555 0.555 0.555 0.556 0.556 0.556 0.557 0.558 0.558 0.559 0.559 0.559 0.561 0.561 0.563 0.563 0.564 0.565 0.565 0.566 0.566 0.566 0.566 0.568 0.568 0.568 0.569 0.569 0.570 0.570 0.570 0.570 0.571 0.571 0.573 0.574 0.577 0.577 0.578 0.579 0.580 0.580 0.583 0.584 0.584 0.585 0.585 0.585 0.585 0.586 0.586 0.586 0.588 0.589 0.589 0.590 0.591 0.593 0.594 0.594 0.595 0.595 0.596 0.596 0.598 0.599 0.600 0.600 0.601 0.602 0.602 0.603 0.605 0.605 0.609 0.610 0.610 0.610 0.616 0.616 0.617 0.618 0.620 0.621 0.622 0.623 0.624 0.625 0.629 0.630 0.631 0.632 0.632 0.633 0.634 0.635 0.637 0.637 0.638 0.641 0.642 0.642 0.643 0.643 0.646 0.646 0.646 0.646 0.647 0.648 0.649 0.650 0.650 0.650 0.651 0.651 0.651 0.652 0.653 0.653 0.653 0.654 0.655 0.656 0.656 0.656 0.657 0.657 0.657 0.659 0.660 0.663 0.665 0.667 0.667 0.668 0.668 0.669 0.669 0.669 0.670 0.671 0.671 0.671 0.673 0.674 0.677 0.678 0.684 0.685 0.686 0.688 0.689 0.691 0.694 0.695 0.696 0.697 0.697 0.698 0.700 0.703 0.703 0.704 0.704 0.704 0.704 0.706 0.707 0.708 0.709 0.710 0.710 0.711 0.712 0.715 0.716 0.717 0.718 0.718 0.719 0.721 0.723 0.723 0.724 0.725 0.725 0.726 0.726 0.727 0.729 0.731 0.731 0.734 0.740 0.740 0.742 0.743 0.743 0.744 0.745 0.751 0.751 0.754 0.754 0.757 0.758 0.762 0.763 0.764 0.765 0.765 0.766 0.768 0.769 0.770 0.771 0.771 0.772 0.773 0.776 0.778 0.778 0.779 0.781 0.784 0.789 0.789 0.791 0.796 0.797 0.798 0.798 0.798 0.799 0.801 0.802 0.802 0.805 0.805 0.806 0.807 0.808 0.817 0.819 0.819 0.823 0.824 0.825 0.828 0.829 0.829 0.831 0.837 0.839 0.839 0.840 0.841 0.841 0.843 0.845 0.846 0.851 0.853 0.854 0.859 0.860 0.861 0.862 0.862 0.864 0.875 0.880 0.884 0.887 0.896 0.897 0.906 0.914 0.916 0.920 0.920 0.951 0.952 0.955 0.963 0.966

Пример сгенерированной выборки длины 1000: 0.024 0.044 0.045 0.057 0.058 0.058 0.062 0.063 0.064 0.066 0.067 0.069 0.071 0.071 0.072 0.074 0.075 0.079 0.079 0.080 0.083 0.094 0.095 0.095 0.095 0.104 0.106 0.108 0.110 0.110 0.110 0.113 0.113 0.116 0.117 0.122 0.122 0.123 0.126 0.127 0.129 0.131 0.132 0.132 0.134 0.134 0.135 0.141 0.141 0.141 0.143 0.144 0.150 0.150 0.150 0.150 0.151 0.152 0.153 0.158 0.158 0.158 0.158 0.164 0.164 0.165 0.165 0.165 0.167 0.170 0.171 0.171 0.175 0.186 0.187 0.188 0.188 0.192 0.192 0.193 0.193 0.193 0.195 0.195 0.198 0.199 0.200 0.201 0.201 0.202 0.203 0.204 0.205 0.207 0.208 0.210 0.212 0.213 0.214 0.214 0.214 0.215 0.217 0.217 0.217 0.218 0.219 0.219 0.220 0.221 0.223 0.224 0.225 0.226 0.226 0.229 0.230 0.230 0.232 0.232 0.233 0.233 0.234 0.235 0.236 0.238 0.239 0.241 0.241 0.242 0.242 0.242 0.243 0.243 0.243 0.245 0.245 0.247 0.247 0.248 0.248 0.252 0.252 0.253 0.254 0.255 0.259 0.259 0.259 0.260 0.261 0.261 0.26

1 0.264 0.264 0.264 0.265 0.265 0.265 0.266 0.266 0.268 0.269 0.270 0.272 0.272
0.273 0.275 0.276 0.276 0.279 0.279 0.280 0.281 0.282 0.282 0.283 0.284 0.284 0.
285 0.285 0.285 0.285 0.286 0.288 0.288 0.288 0.289 0.289 0.292 0.292 0.295 0.29
6 0.296 0.296 0.296 0.296 0.297 0.297 0.298 0.298 0.299 0.300 0.302 0.303 0.303
0.304 0.305 0.305 0.307 0.307 0.307 0.308 0.308 0.308 0.308 0.309 0.311 0.311 0.
313 0.313 0.315 0.315 0.315 0.315 0.316 0.316 0.316 0.316 0.317 0.319 0.320 0.32
2 0.322 0.322 0.323 0.323 0.323 0.324 0.325 0.325 0.325 0.326 0.326 0.327 0.328
0.329 0.330 0.330 0.331 0.331 0.332 0.333 0.333 0.334 0.334 0.334 0.334 0.335 0.
335 0.336 0.336 0.338 0.339 0.339 0.339 0.340 0.341 0.341 0.342 0.342 0.342 0.34
4 0.344 0.344 0.344 0.344 0.344 0.345 0.345 0.345 0.346 0.346 0.347 0.349 0.349
0.350 0.350 0.351 0.351 0.351 0.352 0.352 0.352 0.352 0.353 0.354 0.354 0.354 0.
355 0.355 0.356 0.357 0.358 0.359 0.361 0.361 0.362 0.363 0.363 0.364 0.364 0.36
6 0.366 0.366 0.367 0.368 0.369 0.370 0.371 0.371 0.372 0.372 0.373 0.373 0.373
0.374 0.374 0.375 0.375 0.375 0.375 0.376 0.377 0.377 0.378 0.378 0.379 0.380 0.
381 0.381 0.382 0.382 0.384 0.384 0.384 0.385 0.385 0.385 0.386 0.386 0.387 0.38
7 0.389 0.389 0.389 0.389 0.390 0.391 0.392 0.393 0.394 0.395 0.397 0.397 0.397
0.399 0.399 0.400 0.400 0.400 0.400 0.401 0.401 0.401 0.401 0.402 0.403 0.404 0.
405 0.406 0.406 0.407 0.408 0.408 0.408 0.408 0.409 0.409 0.410 0.411 0.411 0.41
2 0.414 0.414 0.415 0.416 0.417 0.417 0.417 0.417 0.418 0.418 0.418 0.418 0.418
0.418 0.419 0.419 0.420 0.421 0.421 0.421 0.422 0.422 0.422 0.422 0.422 0.422 0.
422 0.423 0.423 0.423 0.423 0.424 0.425 0.426 0.427 0.427 0.428 0.428 0.429 0.42
9 0.431 0.431 0.431 0.431 0.432 0.432 0.432 0.434 0.435 0.435 0.436 0.437 0.437
0.437 0.438 0.439 0.439 0.440 0.441 0.442 0.442 0.442 0.443 0.443 0.445 0.445 0.
446 0.446 0.447 0.447 0.447 0.448 0.449 0.449 0.451 0.451 0.451 0.451 0.452 0.45
2 0.453 0.454 0.454 0.454 0.454 0.455 0.456 0.456 0.456 0.457 0.457 0.457 0.458
0.458 0.459 0.459 0.460 0.460 0.460 0.462 0.462 0.462 0.463 0.464 0.465 0.465 0.
466 0.466 0.466 0.467 0.467 0.468 0.468 0.468 0.469 0.469 0.470 0.470 0.471 0.47
1 0.471 0.471 0.472 0.473 0.473 0.473 0.473 0.473 0.474 0.475 0.476 0.477 0.477
0.477 0.477 0.478 0.478 0.479 0.479 0.480 0.481 0.481 0.481 0.482 0.482 0.482 0.
482 0.482 0.483 0.483 0.483 0.483 0.483 0.484 0.485 0.486 0.486 0.487 0.487 0.48
7 0.487 0.488 0.488 0.488 0.489 0.490 0.490 0.492 0.492 0.493 0.493 0.493 0.493
0.494 0.494 0.494 0.495 0.496 0.496 0.496 0.498 0.498 0.498 0.499 0.499 0.500 0.
501 0.501 0.502 0.503 0.503 0.503 0.503 0.504 0.505 0.505 0.505 0.506 0.506 0.50
6 0.507 0.507 0.507 0.507 0.507 0.508 0.508 0.508 0.508 0.508 0.509 0.509 0.509
0.510 0.512 0.512 0.512 0.512 0.513 0.513 0.513 0.513 0.513 0.514 0.514 0.515 0.
515 0.516 0.516 0.517 0.517 0.517 0.517 0.517 0.518 0.519 0.519 0.519 0.520 0.52
0 0.521 0.521 0.521 0.522 0.522 0.522 0.522 0.523 0.523 0.524 0.525 0.526 0.526
0.526 0.527 0.528 0.529 0.529 0.530 0.530 0.531 0.532 0.532 0.532 0.535 0.536 0.
536 0.536 0.538 0.540 0.540 0.541 0.542 0.543 0.544 0.545 0.545 0.545 0.546 0.54
7 0.547 0.548 0.548 0.549 0.549 0.551 0.552 0.552 0.553 0.553 0.553 0.554 0.556
0.557 0.558 0.561 0.562 0.562 0.562 0.563 0.563 0.563 0.563 0.564 0.564 0.566 0.
568 0.569 0.569 0.569 0.570 0.571 0.571 0.572 0.572 0.572 0.574 0.574 0.575 0.57
6 0.577 0.577 0.579 0.580 0.581 0.582 0.582 0.583 0.584 0.584 0.584 0.585 0.586
0.586 0.588 0.588 0.588 0.589 0.591 0.592 0.592 0.593 0.593 0.593 0.594 0.594 0.
594 0.596 0.596 0.597 0.597 0.598 0.599 0.599 0.600 0.601 0.602 0.603 0.604 0.60
5 0.606 0.607 0.611 0.612 0.613 0.613 0.613 0.613 0.614 0.615 0.615 0.616 0.616
0.616 0.618 0.618 0.619 0.620 0.620 0.620 0.621 0.621 0.621 0.621 0.621 0.624 0.
624 0.625 0.626 0.628 0.628 0.630 0.631 0.633 0.633 0.633 0.633 0.634 0.634 0.63
6 0.636 0.639 0.639 0.640 0.640 0.641 0.641 0.641 0.644 0.647 0.647 0.647 0.649
0.652 0.652 0.654 0.654 0.654 0.655 0.656 0.659 0.660 0.661 0.663 0.663 0.663 0.
664 0.664 0.665 0.666 0.666 0.667 0.667 0.668 0.668 0.668 0.669 0.670 0.671 0.67
1 0.674 0.675 0.675 0.675 0.676 0.676 0.676 0.677 0.679 0.679 0.680 0.682 0.683
0.685 0.687 0.687 0.690 0.690 0.691 0.692 0.693 0.695 0.696 0.696 0.696 0.696 0.
698 0.699 0.699 0.699 0.701 0.705 0.705 0.706 0.706 0.707 0.707 0.708 0.709 0.70
9 0.710 0.710 0.713 0.717 0.719 0.719 0.722 0.722 0.724 0.724 0.725 0.728 0.728
0.729 0.730 0.730 0.731 0.732 0.733 0.733 0.734 0.734 0.735 0.736 0.742 0.743 0.
746 0.746 0.748 0.749 0.750 0.753 0.754 0.756 0.757 0.757 0.759 0.760 0.760 0.76
0 0.761 0.761 0.762 0.763 0.763 0.766 0.767 0.768 0.774 0.776 0.777 0.777 0.778
0.778 0.781 0.784 0.787 0.787 0.787 0.789 0.789 0.791 0.794 0.797 0.799 0.801 0.
801 0.801 0.803 0.808 0.809 0.810 0.811 0.816 0.818 0.818 0.820 0.822 0.822 0.82
5 0.829 0.830 0.830 0.830 0.831 0.831 0.831 0.832 0.833 0.833 0.837 0.839 0.840
0.845 0.846 0.846 0.847 0.853 0.854 0.856 0.858 0.862 0.867 0.867 0.870 0.870 0.

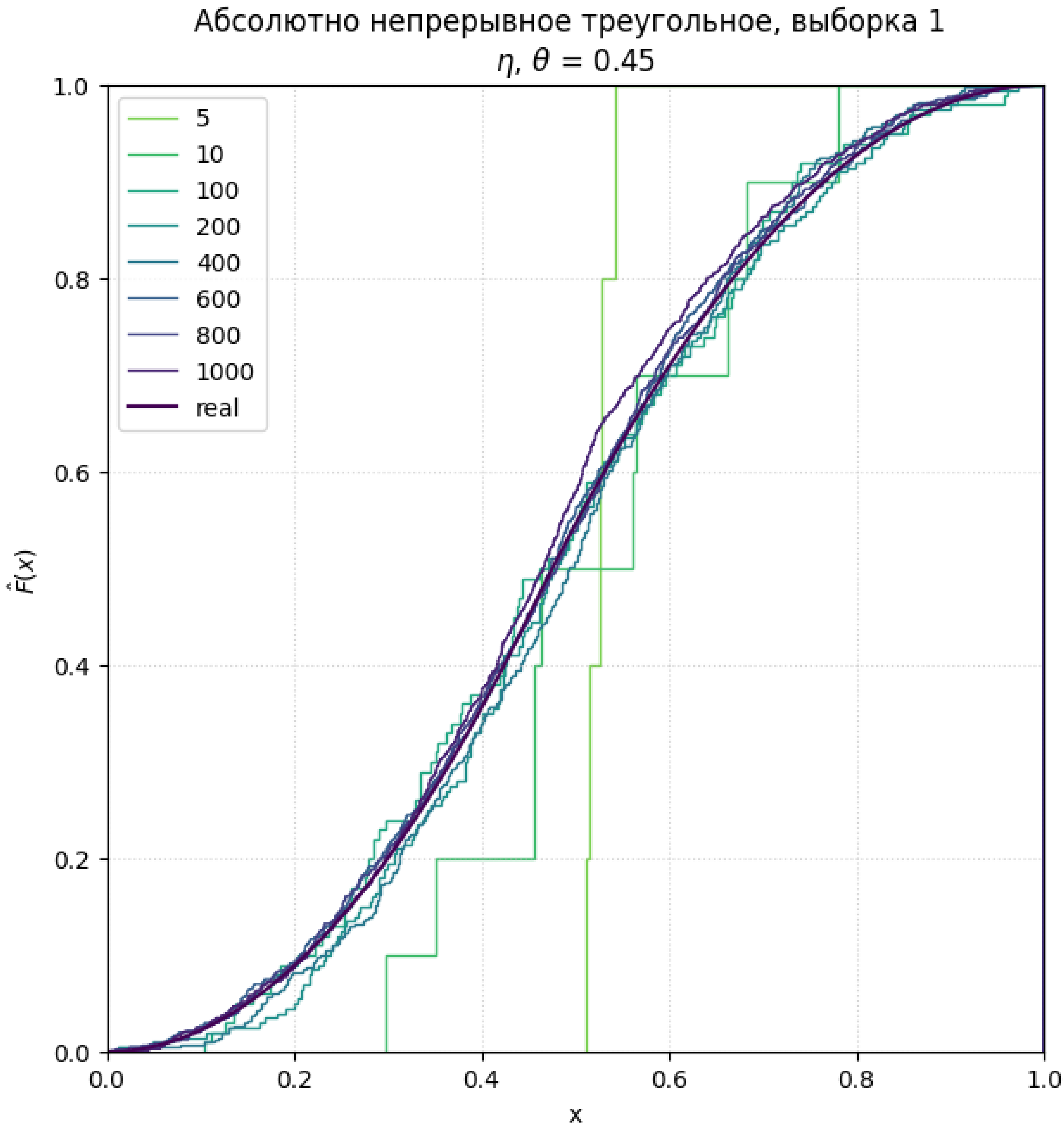
```
877 0.888 0.890 0.892 0.894 0.899 0.904 0.905 0.907 0.913 0.917 0.924 0.924 0.92
6 0.927 0.927 0.946 0.957 0.960 0.967 0.989
----------
```

Задание 2

In [24]:

```python
def eta_distr(sample, x):
    res = 0
    for i in sample:
        if i <= x:
            res += 1
    return res / len(sample)

def eta_distr_real(x, theta=0.45):
    if x<0: return 0
    if x<theta: return x*x/theta
    if x<1: return (2*x-x*x-theta)/(1-theta)
    return 1

X_realeta = np.linspace(0,1,1000)
Y_realeta = np.array([eta_distr_real(x) for x in X_realeta])

Yeta = [[[np.array(eta_distr(sample_eta[k][j], x)) for x in sample_eta[k][j]]
                     for j in range(5)] for k in range(len(n))]

def eta_Dmn(Yn, Ym, Xn, Xm):
    n = len(Xn)
    m = len(Xm)
    Xn = np.concatenate([[0],Xn,[1]])
    Xm = np.concatenate([[0],Xm,[1]])
    Yn = np.concatenate([[0],Yn,[1]])
    Ym = np.concatenate([[0],Ym,[1]])
    if m>n:
        m, n = n, m
        Xm, Xn = Xn, Xm
        Ym, Yn = Yn, Ym

    res = 0
    for i in range(1, n+1):
        j=0
        while not (Xm[j]<=Xn[i]<Xm[j+1]):
            j+=1
        d = abs(Yn[i]-Ym[j])
        if d>res: res = d
    return (n*m/(n+m))**0.5*res

diffseta = np.array([[[(eta_Dmn(Yeta[i][k], Yeta[j][k],
                                sample_eta[i][k], sample_eta[j][k]))
                      for i in range(len(n))] for j in range(len(n))]
                     for k in range(5)])
diffseta_demo = np.array([[[('%.2f' % diffseta[k][j][i]) for i in range(len(n))]
                           for j in range(len(n))] for k in range(5)])
```

In [25]:
```
colors = ['#7bd152', '#45be71', '#25a885', '#21908c',
          '#2b798e', '#355f8d', '#414486', '#482574']

# graph[0]
fig, ax = plt.subplots(2,1, figsize=(7,12), height_ratios = [2,1])
for i in range(8):
    ax[0].stairs(Yeta[i][0], np.append(sample_eta[i][0], 1), color = colors[i])
ax[0].plot(X_realeta, Y_realeta, color = '#440154')
ax[0].set(xmargin = 0, ymargin = 0, xlabel = 'x', ylabel = r'$\hat{F}(x)$',
          title = 'Абсолютно непрерывное треугольное, выборка 1 \n' +
                  '$\\eta , \\,\\theta$ = 0.45')
ax[0].legend([*n, 'real'], loc='upper left')
ax[0].grid(True, alpha = 0.5, ls='dotted')

ax[1].table(cellText = diffseta_demo[0], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$' +
                r'$|F_n(x)-F_m(x)|$');
```

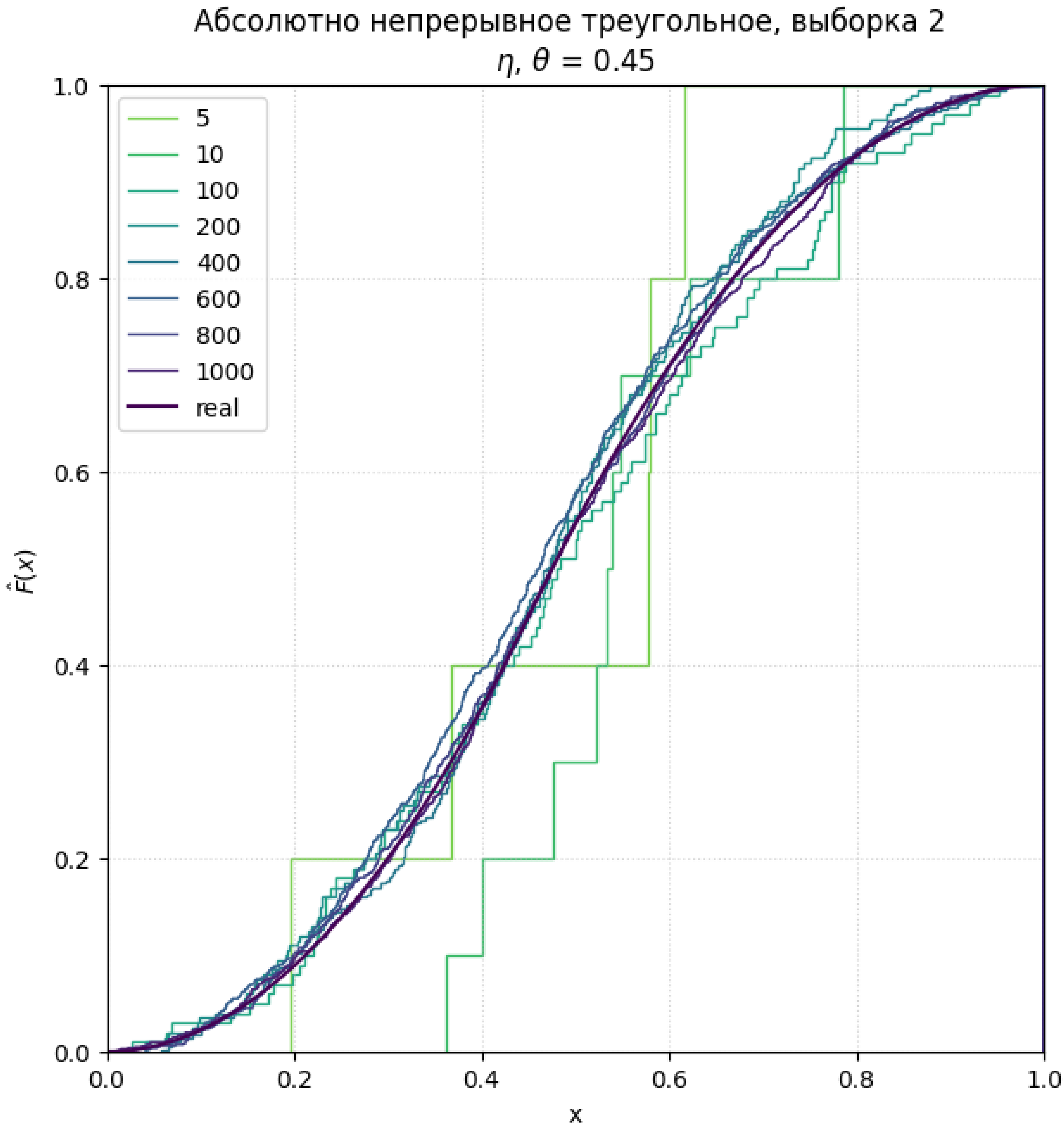


$$D_{m,n} = \sqrt{\frac{nm}{n+m}} \sup_{x \in \mathbb{R}} |F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.91	1.24	1.25	1.19	1.28	1.24	1.35
10	0.91	0.00	0.87	0.76	0.70	0.82	0.80	0.88
100	1.24	0.87	0.00	0.61	0.74	0.49	0.51	0.62
200	1.25	0.76	0.61	0.00	0.52	0.57	0.68	0.80
400	1.19	0.70	0.74	0.52	0.00	0.86	0.82	1.35
600	1.28	0.82	0.49	0.57	0.86	0.00	0.43	0.97
800	1.24	0.80	0.51	0.68	0.82	0.43	0.00	1.36
1000	1.35	0.88	0.62	0.80	1.35	0.97	1.36	0.00

In [26]:

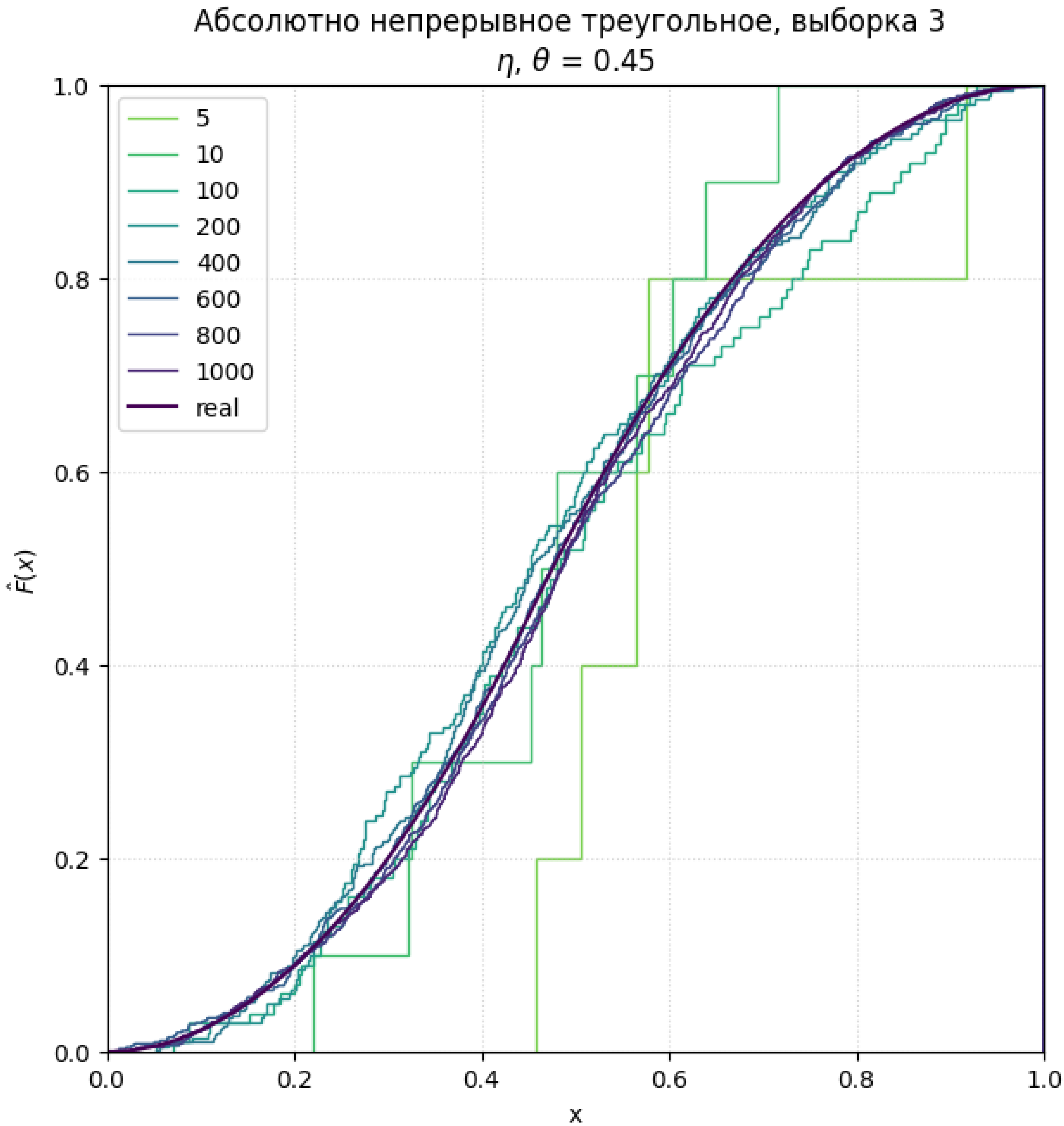
```
# graph[1]
fig, ax = plt.subplots(2,1, figsize=(7,12), height_ratios = [2,1])
for i in range(8):
    ax[0].stairs(Yeta[i][1], np.append(sample_eta[i][1], 1), color = colors[i])
ax[0].plot(X_realeta, Y_realeta, color = '#440154')
ax[0].set(xmargin = 0, ymargin = 0, xlabel = 'x', ylabel = r'$\hat{F}(x)$',
          title = 'Абсолютно непрерывное треугольное, выборка 2 \n' +
                  '$\\eta , \\,\\theta$ = 0.45')
ax[0].legend([*n, 'real'], loc='upper left')
ax[0].grid(True, alpha = 0.5, ls='dotted')

ax[1].table(cellText = diffseta_demo[1], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$' +
                r'$|F_n(x)-F_m(x)|$');
```



$$D_{m,n} = \sqrt{\frac{nm}{n+m}} \sup_{x \in \mathbb{R}} |F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.55	0.63	0.65	0.67	0.68	0.60	0.62
10	0.55	0.00	0.87	0.99	0.98	1.05	0.95	0.95
100	0.63	0.87	0.00	0.90	0.76	0.79	0.68	0.51
200	0.65	0.99	0.90	0.00	0.64	0.69	0.52	0.75
400	0.67	0.98	0.76	0.64	0.00	0.99	0.86	1.08
600	0.68	1.05	0.79	0.69	0.99	0.00	0.86	1.17
800	0.60	0.95	0.68	0.52	0.86	0.86	0.00	0.62
1000	0.62	0.95	0.51	0.75	1.08	1.17	0.62	0.00

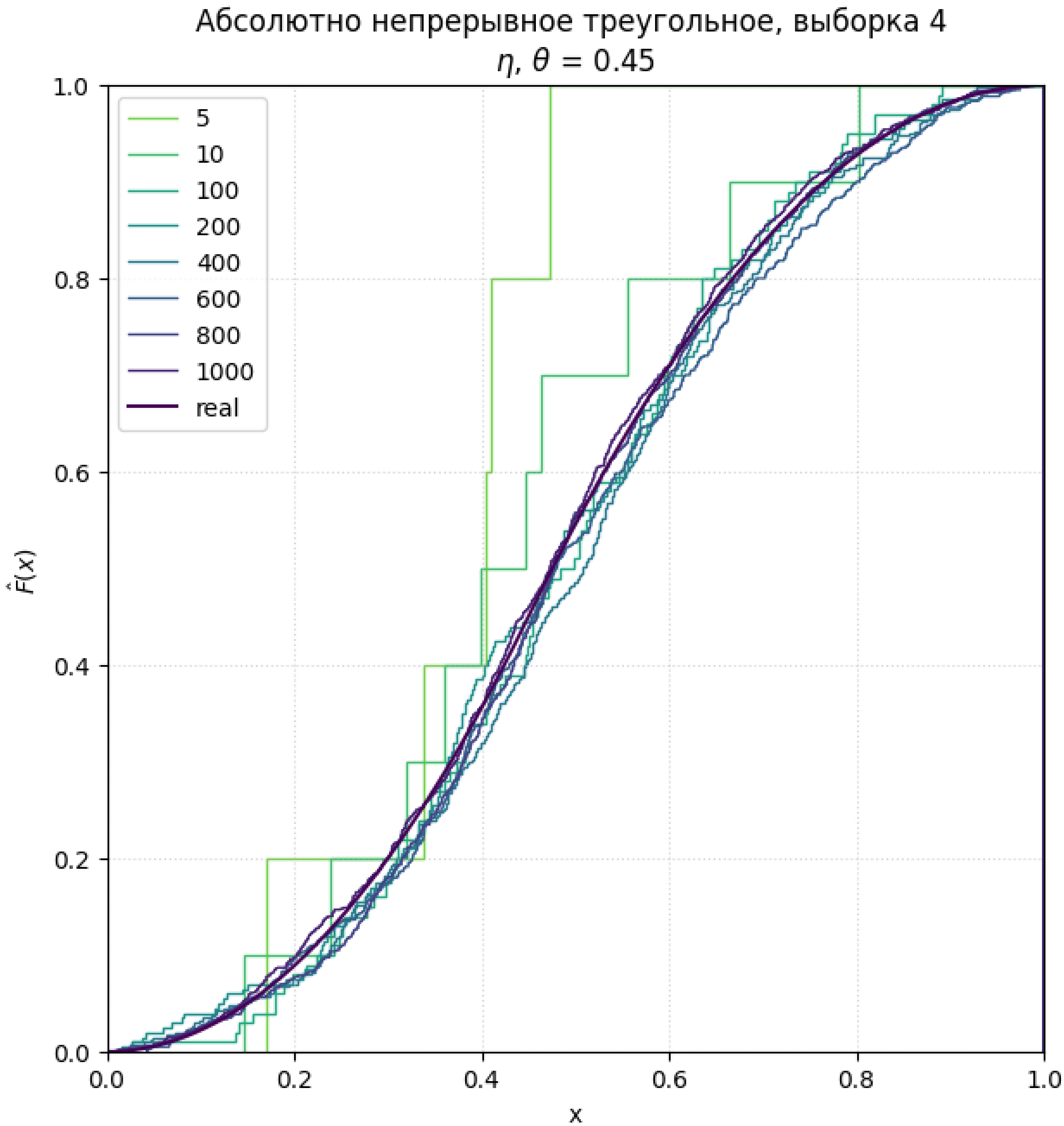
In [27]:
```
# graph[2]
fig, ax = plt.subplots(2,1, figsize=(7,12), height_ratios = [2,1])
for i in range(8):
    ax[0].stairs(Yeta[i][2], np.append(sample_eta[i][2], 1), color = colors[i])
ax[0].plot(X_realeta, Y_realeta, color = '#440154')
ax[0].set(xmargin = 0, ymargin = 0, xlabel = 'x', ylabel = r'$\hat{F}(x)$',
          title = 'Абсолютно непрерывное треугольное, выборка 3 \n' +
                  '$\\eta , \\,\\theta$ = 0.45')
ax[0].legend([*n, 'real'], loc='upper left')
ax[0].grid(True, alpha = 0.5, ls='dotted')

ax[1].table(cellText = diffseta_demo[2], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$' +
                r'$|F_n(x)-F_m(x)|$');
```



$$D_{m,n} = \sqrt{\frac{nm}{n+m}} \sup_{x \in \mathbb{R}} |F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.73	0.98	1.15	1.13	1.05	1.00	0.98
10	0.73	0.00	0.66	0.65	0.62	0.51	0.53	0.49
100	0.98	0.66	0.00	0.73	0.72	0.69	0.81	0.80
200	1.15	0.65	0.73	0.00	0.66	0.92	1.01	1.23
400	1.13	0.62	0.72	0.66	0.00	0.77	1.06	1.27
600	1.05	0.51	0.69	0.92	0.77	0.00	0.66	0.79
800	1.00	0.53	0.81	1.01	1.06	0.66	0.00	0.52
1000	0.98	0.49	0.80	1.23	1.27	0.79	0.52	0.00

In [28]:
```
# graph[3]
fig, ax = plt.subplots(2,1, figsize=(7,12), height_ratios = [2,1])
for i in range(8):
    ax[0].stairs(Yeta[i][0], np.append(sample_eta[i][3], 1), color = colors[i])
ax[0].plot(X_realeta, Y_realeta, color = '#440154')
ax[0].set(xmargin = 0, ymargin = 0, xlabel = 'x', ylabel = r'$\hat{F}(x)$',
          title = 'Абсолютно непрерывное треугольное, выборка 4 \n' +
                  '$\\eta , \\,\\theta$ = 0.45')
ax[0].legend([*n, 'real'], loc='upper left')
ax[0].grid(True, alpha = 0.5, ls='dotted')

ax[1].table(cellText = diffseta_demo[3], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$' +
                r'$|F_n(x)-F_m(x)|$');
```

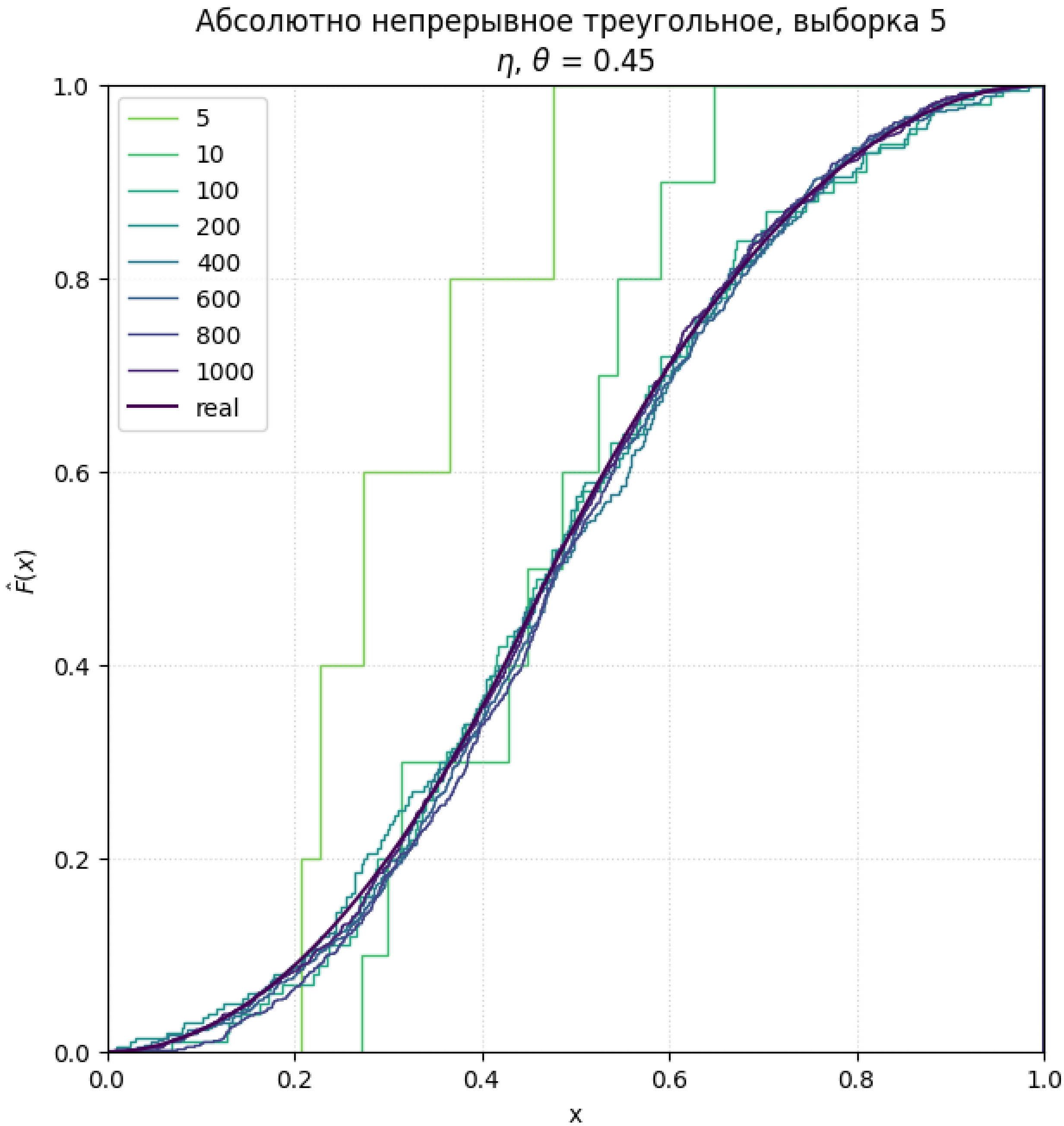


$$D_{m,n} = \sqrt{\frac{nm}{n+m}} \sup_{x \in \mathbb{R}} |F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.37	1.11	1.13	1.21	1.14	1.15	1.12
10	0.37	0.00	0.69	0.69	0.80	0.74	0.74	0.69
100	1.11	0.69	0.00	0.49	0.54	0.74	0.48	0.59
200	1.13	0.69	0.49	0.00	0.89	0.63	0.77	0.74
400	1.21	0.80	0.54	0.89	0.00	0.72	1.18	1.28
600	1.14	0.74	0.74	0.63	0.72	0.00	0.86	1.14
800	1.15	0.74	0.48	0.77	1.18	0.86	0.00	0.91
1000	1.12	0.69	0.59	0.74	1.28	1.14	0.91	0.00

In [29]:

```
# graph[4]
fig, ax = plt.subplots(2,1, figsize=(7,12), height_ratios = [2,1])
for i in range(8):
    ax[0].stairs(Yeta[i][4], np.append(sample_eta[i][4], 1), color = colors[i])
ax[0].plot(X_realeta, Y_realeta, color = '#440154')
ax[0].set(xmargin = 0, ymargin = 0, xlabel = 'x', ylabel = r'$\hat{F}(x)$',
          title = 'Абсолютно непрерывное треугольное, выборка 5 \n' +
                  '$\\eta , \\,\\theta$ = 0.45')
ax[0].legend([*n, 'real'], loc='upper left')
ax[0].grid(True, alpha = 0.5, ls='dotted')

ax[1].table(cellText = diffseta_demo[4], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$' +
                r'$|F_n(x)-F_m(x)|$');
```



$$D_{m,n} = \sqrt{\frac{nm}{n+m}} \sup_{x \in \mathbb{R}} |F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.73	1.05	1.08	1.13	1.15	1.20	1.12
10	0.73	0.00	0.63	0.71	0.72	0.76	0.71	0.69
100	1.05	0.63	0.00	0.53	0.49	0.45	0.57	0.36
200	1.08	0.71	0.53	0.00	0.58	0.69	0.77	0.57
400	1.13	0.72	0.49	0.58	0.00	0.63	0.78	0.78
600	1.15	0.76	0.45	0.69	0.63	0.00	0.49	0.65
800	1.20	0.71	0.57	0.77	0.78	0.49	0.00	0.75
1000	1.12	0.69	0.36	0.57	0.78	0.65	0.75	0.00

Задание 3

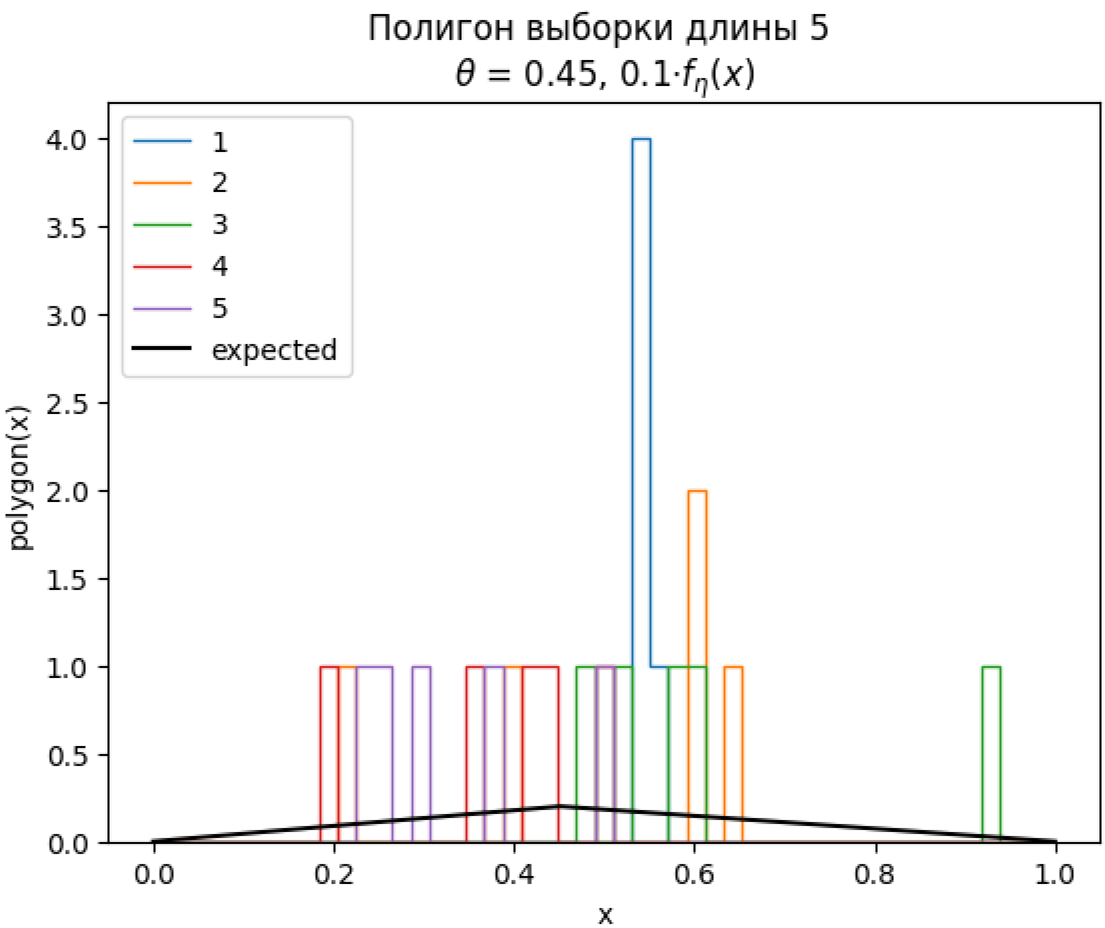
Логика с домножением вероятности здесь также применима, однако она должна быть немного модифицирована: полигон частот как таковой почти не будет иметь смысла, так как вероятность попадания в каждое значение стремится к нулю, что значит что если просто расставить на отрезке от 0 до 1 все значения выборки, то очень наврядли полученный график поднимется выше единицы. Получить хоть какой-то смысл из этого графика будет затруднительно, потому нужно из него будет сложно получить что-то больше чем прямую, не то что сравнить с потенциально сложной функцией вероятности.

Для решения этой проблемы имеет смысл разбить данный отрезок на сколько-то частей. Да, это примерно сведет случай к дискретному, однако это даст весьма уверенную возможность проанализировать график полигона частот относительно графика вероятности. И вот где логика этого номера с дискретной вероятностью ломается: нам не известна функиц я вероятности, нам дана плотность распределения. Но и эта проблема теперь решается достаточно просто: если рассматривать не саму плотность вероятности, а плотность вероятности на каком-то небольшом участке, то тогда плотность вероятности домноженная на длину рассматриваемого участка уже должна соотноситься с формой эмперической вероятности, определенной ранее как $\hat{P}(x)$:

При $n \to \infty$ $\Delta x f(x) = \frac{k}{n} \Rightarrow k = n\Delta x f(x)$. В этом задании я разделил отрезок от 0 до 1 на 50 равных между собой отрезков. Тогда полигон частот должен соотноситься с плотностью через коэффициент домножения плотности $\frac{n = \text{длина выборки}}{50}$

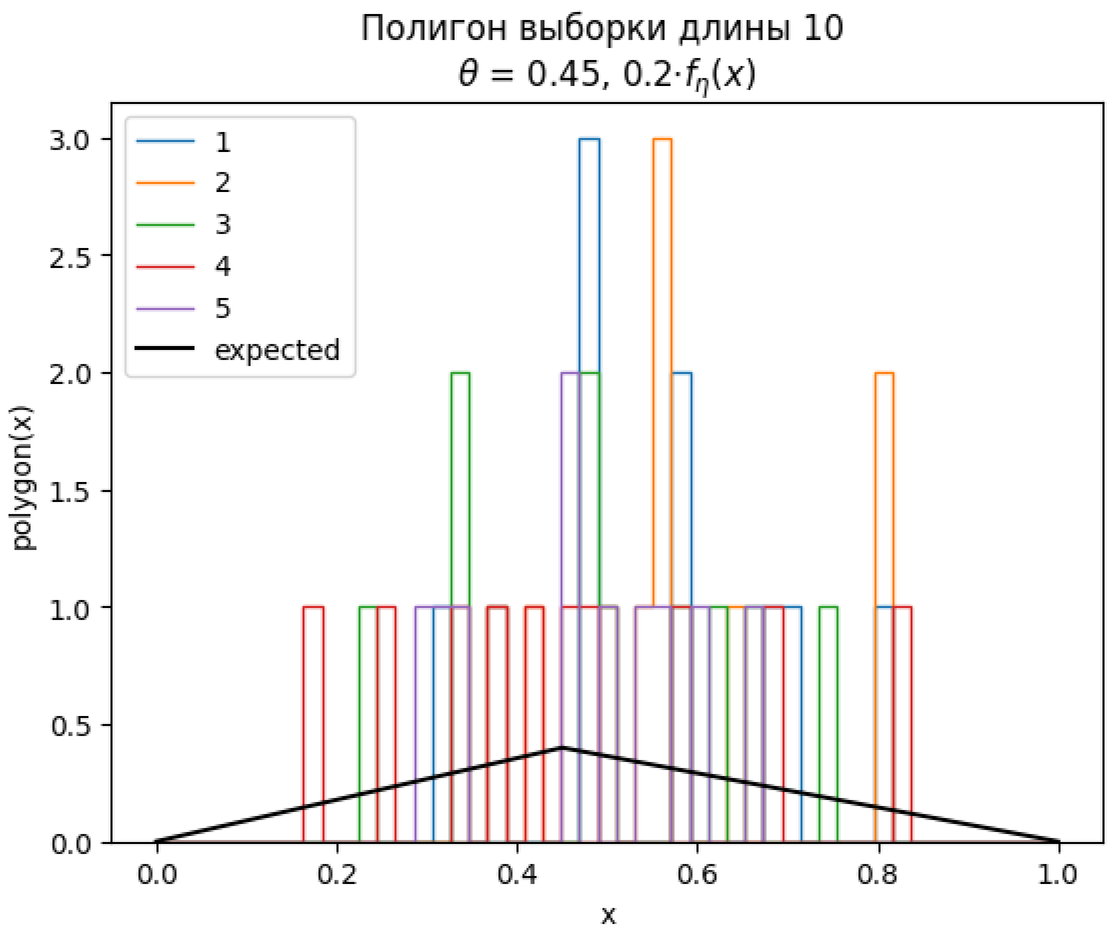
In [30]:
```
def eta_pilygon(sample, X):
    Y = np.zeros(X.shape)
    tick = 1
    for i in sample:
        while i > X[tick]: tick+=1
        Y[tick]+=1
    return Y

def eta_posib(x, theta = 0.45):
    if x<0: return 0
    if x<theta: return 2*x/theta
    if x<1: return 2*(1-x)/(1-theta)
    return 0

X_poleta = np.linspace(0,1,50)
Y_poleta = [[eta_pilygon(sample_eta[k][j], X_poleta) for j in range(5)]
            for k in range(len(n))]

posibilityeta = np.array([eta_posib(x) for x in X_realeta])
```

In [31]:
```python
# n=5
for i in range(5):
    plt.stairs(Y_poleta[0][i], np.append(X_poleta,1))
plt.plot(X_realeta, posibilityeta*5/50, 'k')
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 5 \n$\\theta$ = 0.45, ' +
          '%.1f$\\cdot f_{\\eta}(x)$'%(5/50))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```

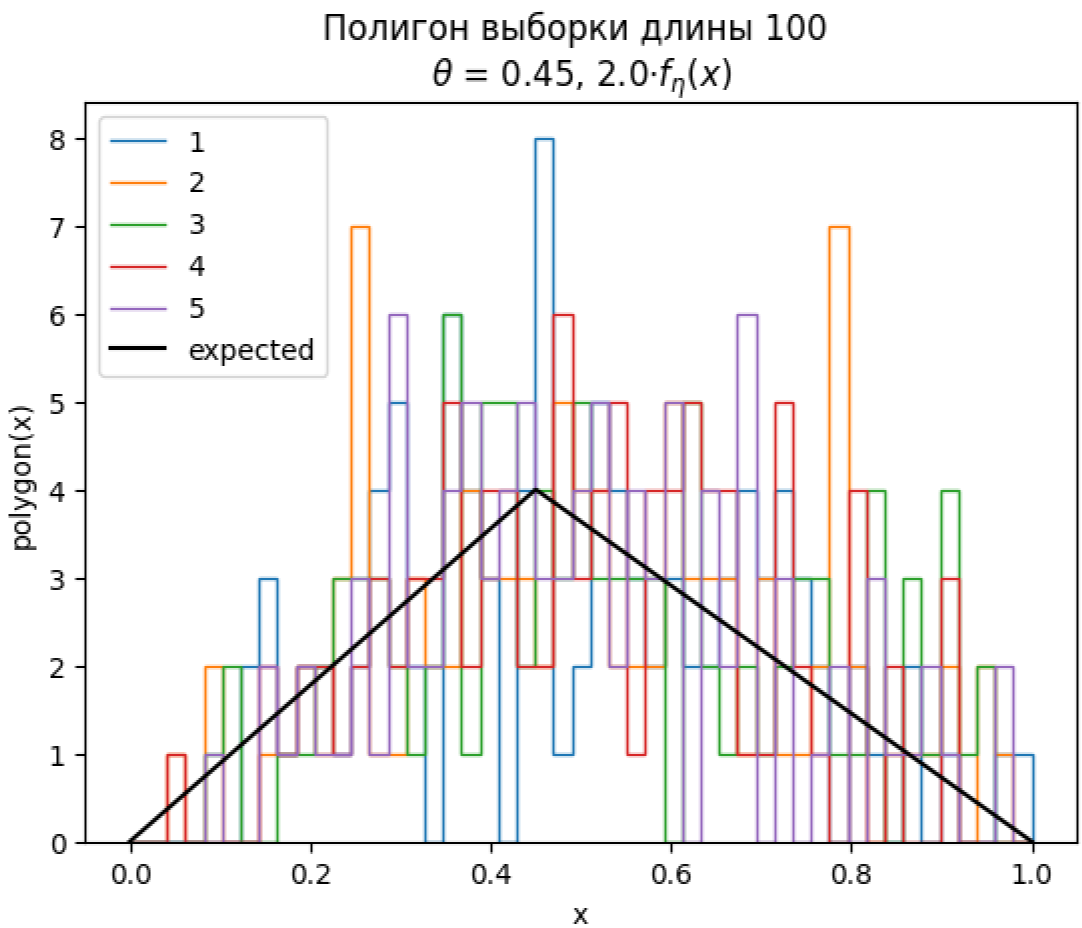


In [32]:

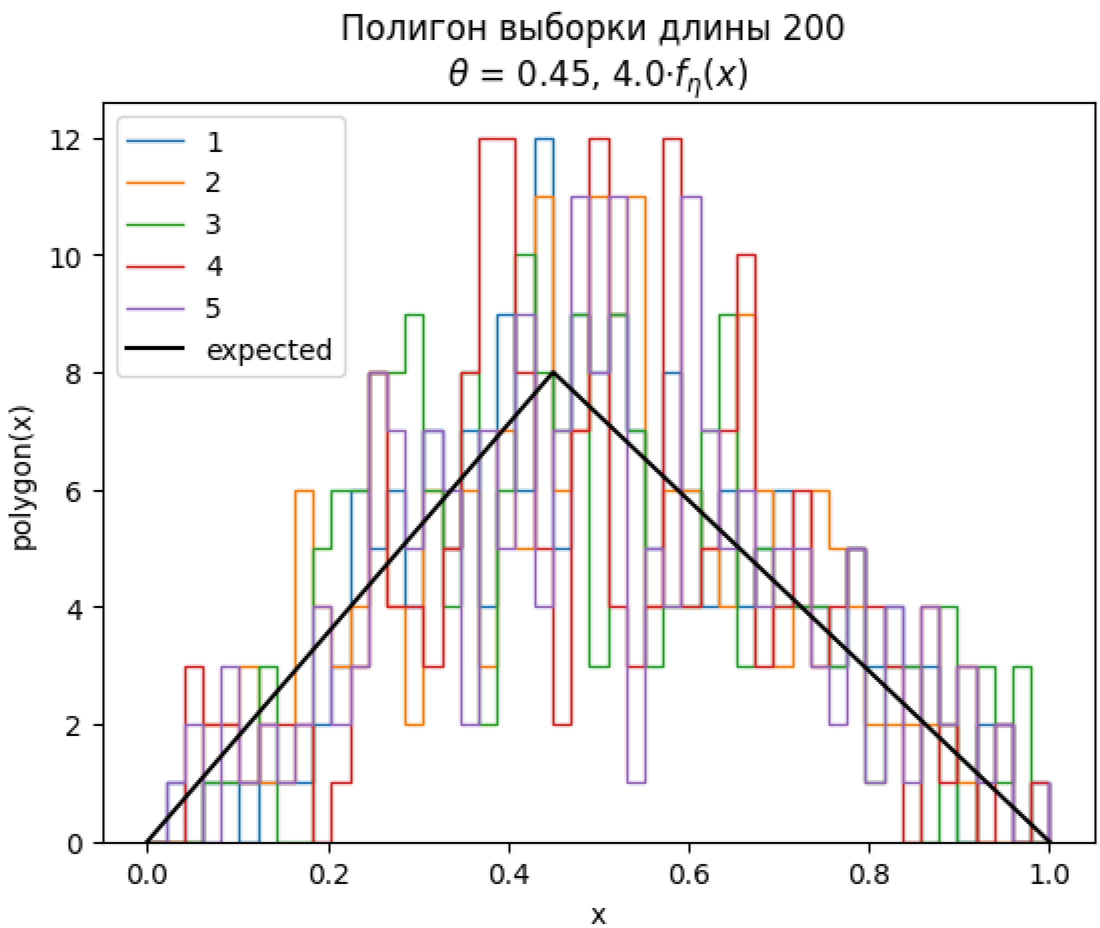
```
# n=10
for i in range(5):
    plt.stairs(Y_poleta[1][i], np.append(X_poleta, 1))
plt.plot(X_realeta, posibilityeta*10/50, 'k')
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 10 \n$\\theta$ = 0.45, ' +
          '%.1f$\\cdot f_{\\eta}(x)$'%(10/50))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



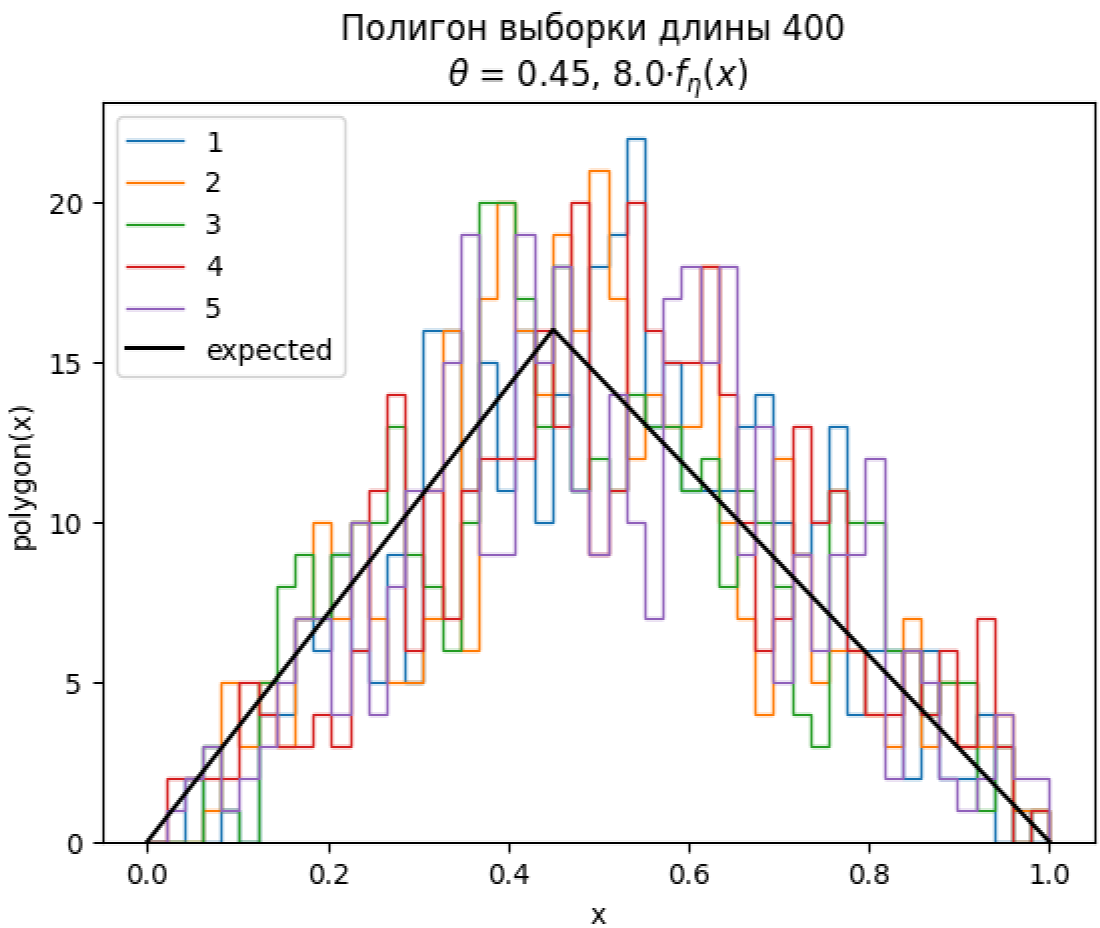
In [33]:
```
# n=100
for i in range(5):
    plt.stairs(Y_poleta[2][i], np.append(X_poleta, 1))
plt.plot(X_realeta, posibilityeta*100/50, 'k')
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 100 \n$\\theta$ = 0.45, ' +
          '%.1f$\\cdot f_{\\eta}(x)$'%(100/50))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



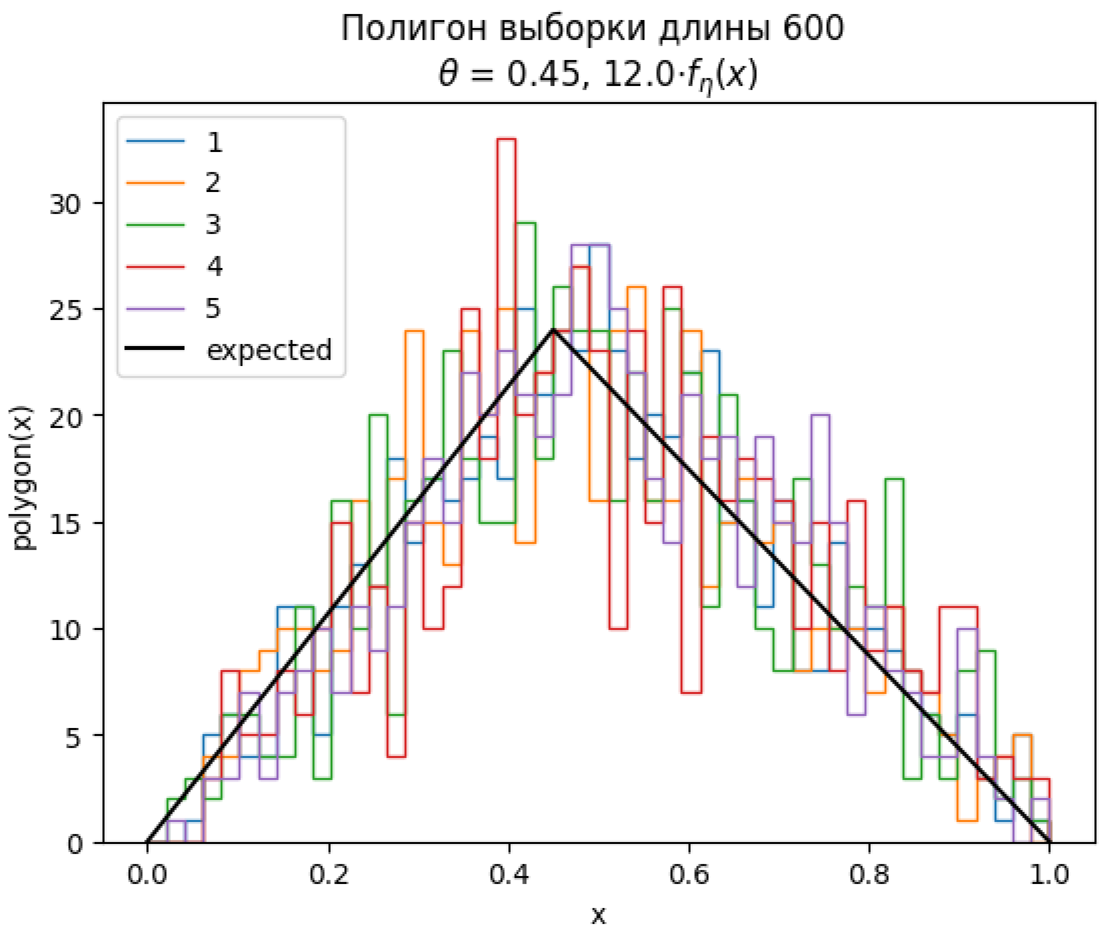
In [34]:
```
# n=200
for i in range(5):
    plt.stairs(Y_poleta[3][i], np.append(X_poleta, 1))
plt.plot(X_realeta, posibilityeta*200/50, 'k')
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 200 \n$\\theta$ = 0.45, ' +
          '%.1f$\\cdot f_{\\eta}(x)$'%(200/50))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



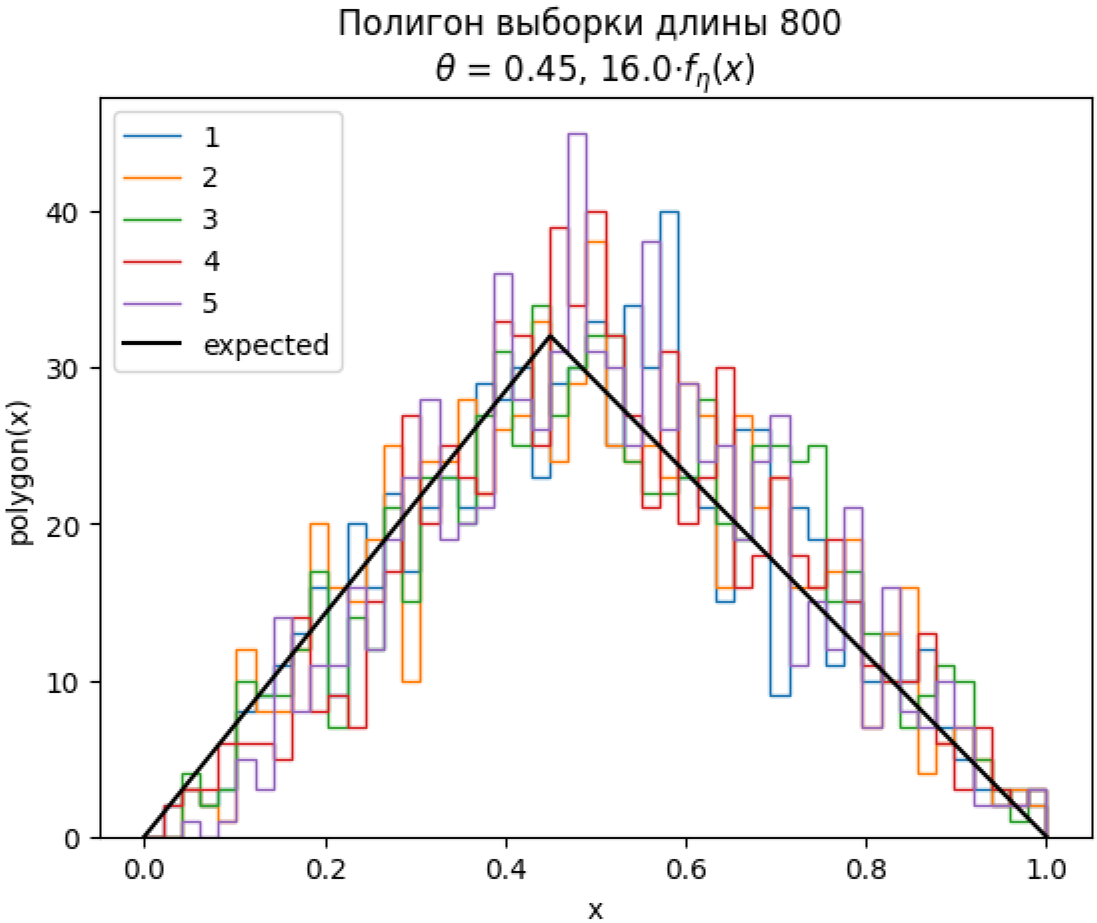
In [35]:
```
# n=400
for i in range(5):
    plt.stairs(Y_poleta[4][i], np.append(X_poleta, 1))
plt.plot(X_realeta, posibilityeta*400/50, 'k')
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 400 \n$\\theta$ = 0.45, ' +
          '%.1f$\\cdot f_{\\eta}(x)$'%(400/50))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



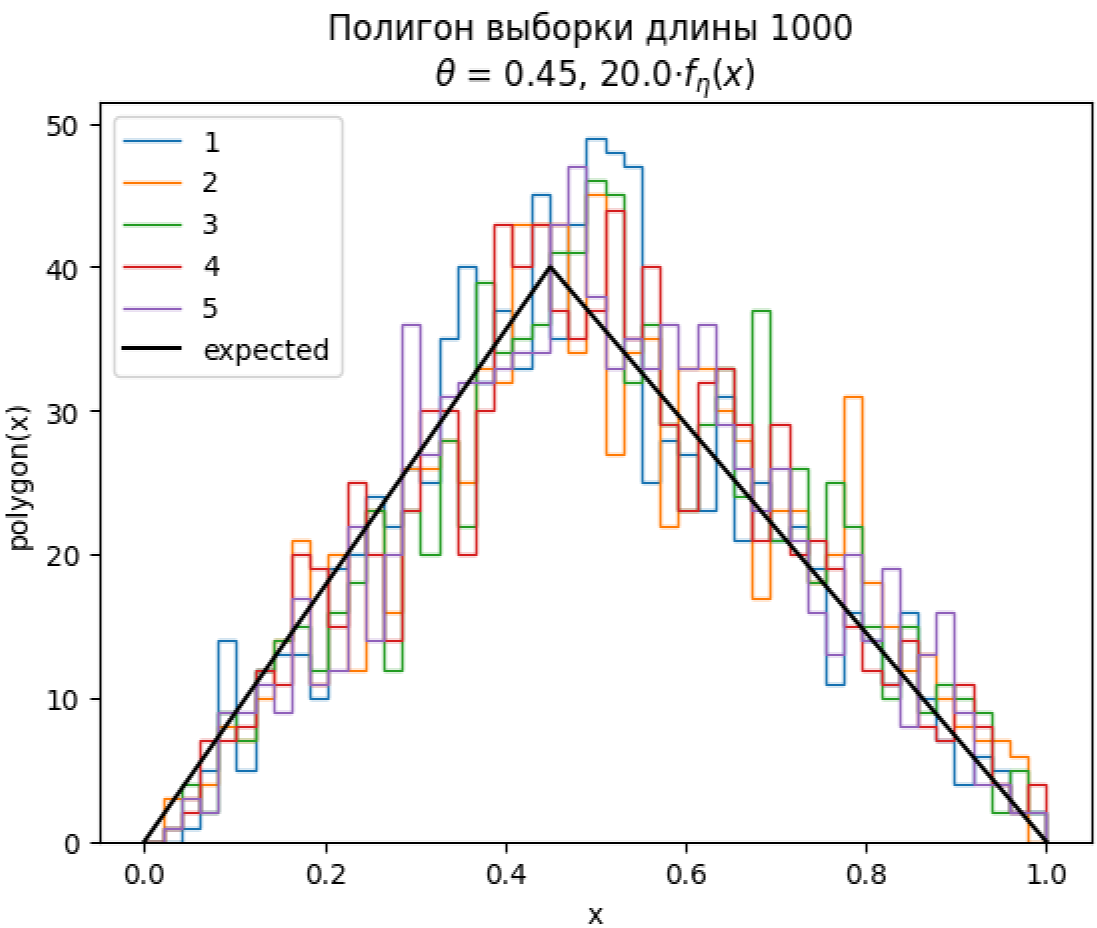
In [36]:
```
# n=600
for i in range(5):
    plt.stairs(Y_poleta[5][i], np.append(X_poleta, 1))
plt.plot(X_realeta, posibilityeta*600/50, 'k')
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 600 \n$\\theta$ = 0.45, ' +
          '%.1f$\\cdot f_{\\eta}(x)$'%(600/50))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



In [37]:
```
# n=800
for i in range(5):
    plt.stairs(Y_poleta[6][i], np.append(X_poleta, 1))
plt.plot(X_realeta, posibilityeta*800/50, 'k')
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 800 \n$\\theta$ = 0.45, ' +
          '%.1f$\\cdot f_{\\eta}(x)$'%(800/50))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



In [38]:
```
# n=1000
for i in range(5):
    plt.stairs(Y_poleta[7][i], np.append(X_poleta, 1))
plt.plot(X_realeta, posibilityeta*1000/50, 'k')
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Полигон выборки длины 1000 \n$\\theta$ = 0.45, ' +
          '%.1f$\\cdot f_{\\eta}(x)$'%(1000/50))
plt.legend([1, 2, 3, 4, 5, 'expected'], loc='upper left');
```



Чтобы отметить, что я все еще анализирую графики, а не просто переписываю код под непрерывный случай: графики и результаты двух предыдущих заданий опять демонстрируют справедливость теоремы.

Задание 4

In [39]:
```
def eta_sample_mean(sample):
    return sum(sample)/len(sample)

def eta_sample_variance(sample):
    return sum((sample-xi_sample_mean(sample))**2)/len(sample)

meanseta = np.array([[eta_sample_mean(sample_eta[k][j])for j in range(5)]
                     for k in range(len(n))])
varianceseta = np.array([[eta_sample_variance(sample_eta[k][j])for j in range(5)]
                         for k in range(len(n))])
means_peta = np.array([['%.4f' % i for i in j] for j in meanseta])
variances_peta = np.array([['%.4f' % i for i in j] for j in varianceseta])
expectationeta = (1+0.45)/3
varianceeta = (1-0.45+0.45**2)/18
means_difeta = np.array([['%.4f' % i for i in j]
                         for j in (meanseta-expectationeta)])
variances_difeta = np.array([['%.4f' % i for i in j]
                             for j in (varianceseta-varianceeta)])
```

In [40]:
```
fig, ax = plt.subplots(2,1, figsize=(7,6.5))
ax[0].table(cellText = means_peta, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('Выборочные средние');
ax[1].table(cellText = variances_peta, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Выборочные дисперсии');
```

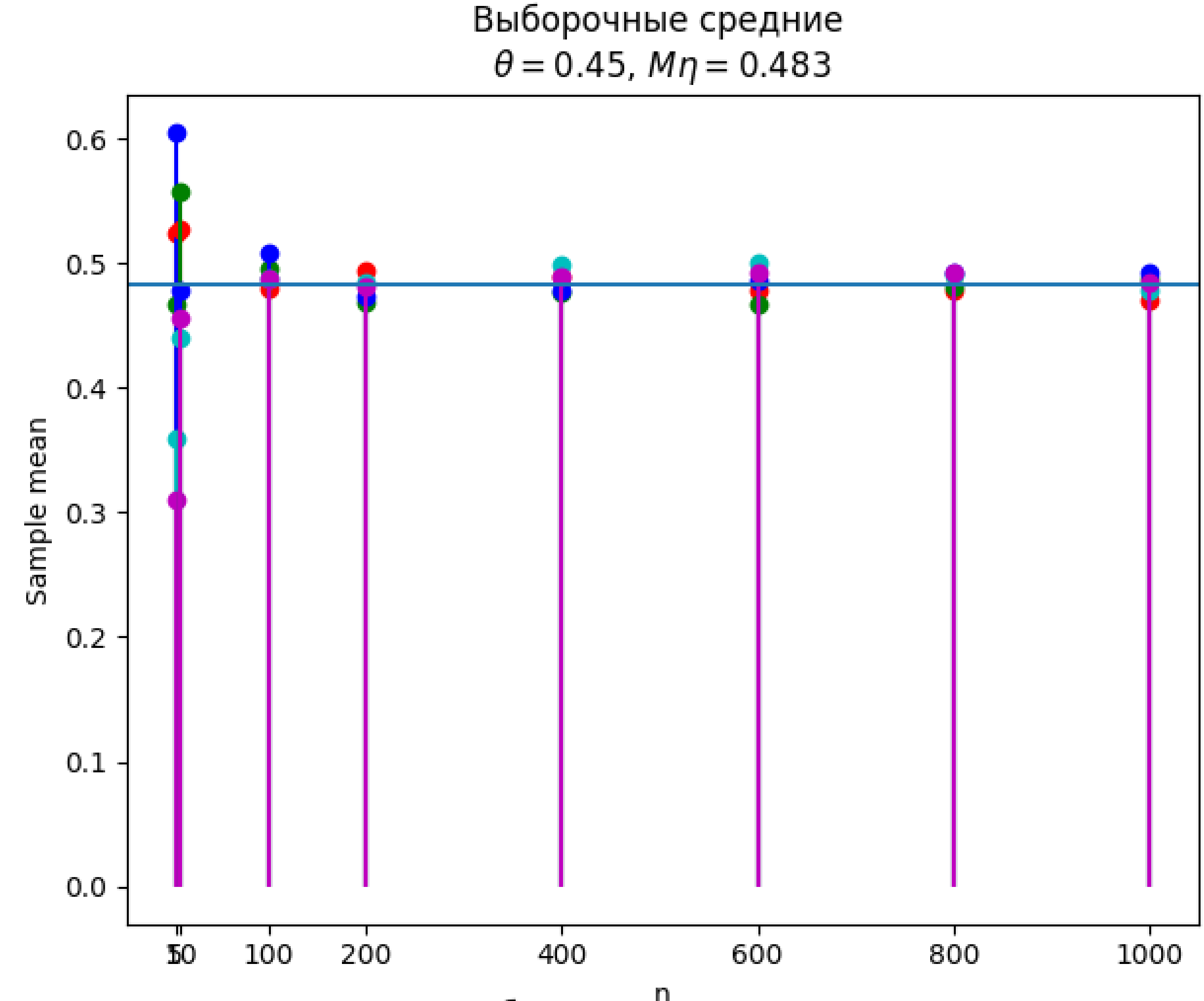
Выборочные средние

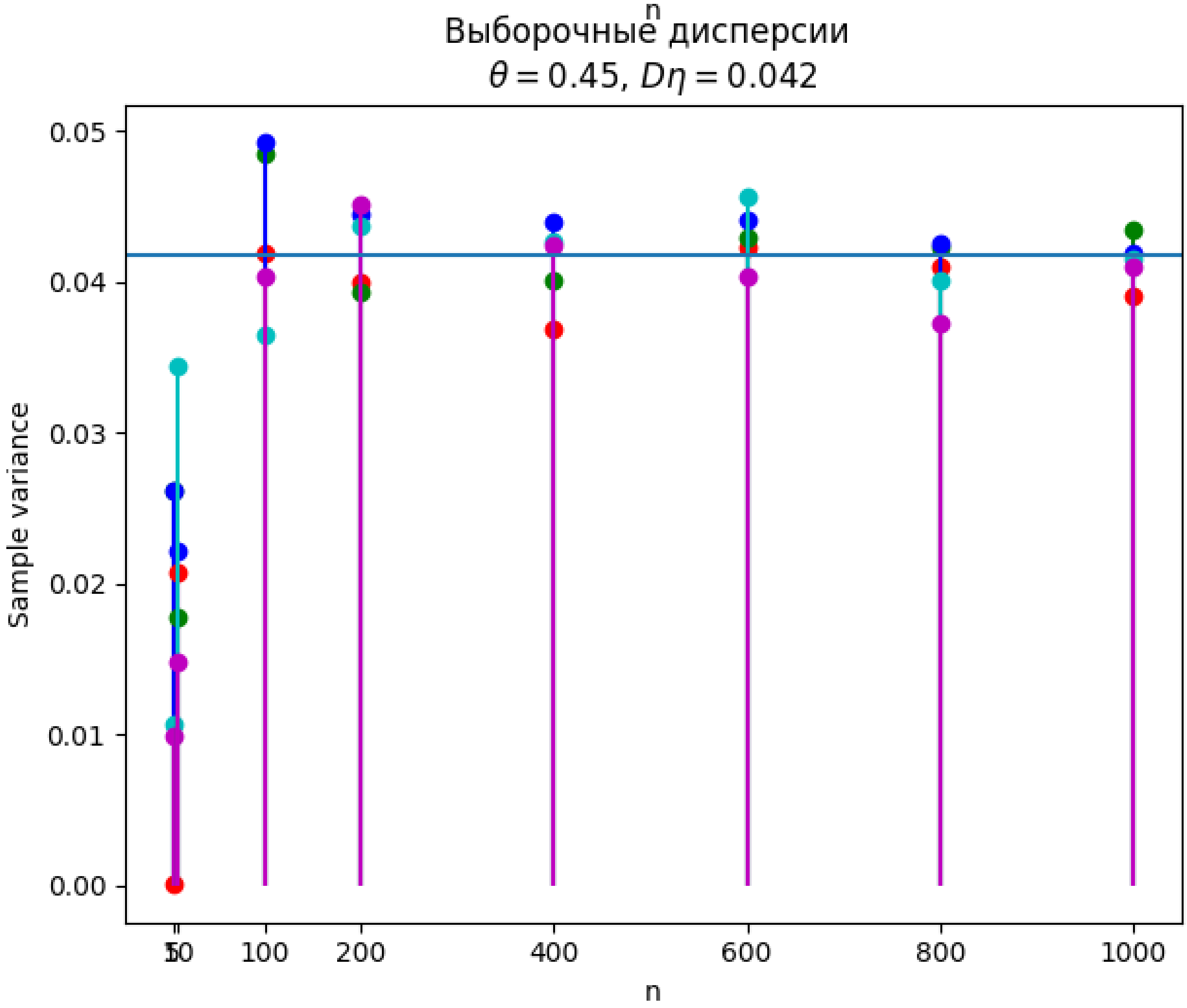
	1	2	3	4	5
5	0.5245	0.4672	0.6051	0.3590	0.3098
10	0.5278	0.5574	0.4788	0.4397	0.4555
100	0.4793	0.4948	0.5075	0.4874	0.4868
200	0.4936	0.4677	0.4728	0.4849	0.4807
400	0.4899	0.4771	0.4776	0.4989	0.4885
600	0.4785	0.4675	0.4853	0.4996	0.4918
800	0.4785	0.4806	0.4930	0.4901	0.4929
1000	0.4695	0.4872	0.4922	0.4776	0.4848

Выборочные дисперсии

	1	2	3	4	5
5	0.0001	0.0261	0.0261	0.0107	0.0099
10	0.0207	0.0178	0.0221	0.0344	0.0148
100	0.0419	0.0485	0.0493	0.0365	0.0404
200	0.0399	0.0393	0.0445	0.0437	0.0451
400	0.0369	0.0400	0.0439	0.0426	0.0424
600	0.0422	0.0429	0.0442	0.0457	0.0403
800	0.0410	0.0422	0.0425	0.0401	0.0373
1000	0.0390	0.0434	0.0419	0.0415	0.0410

In [41]:
```
fig, ax = plt.subplots(2,1, figsize=(7,12))
for k in range(len(n)):
    for j in range(5):
        ax[0].stem(n[k], meanseta[k][j], 'rgbcm'[j])
ax[0].axhline(expectationeta)
ax[0].set(xticks = n, xlabel = 'n', ylabel = 'Sample mean',
          title = 'Выборочные средние \n$\\theta = 0.45, ' +
                  '\\,M\\eta = %.3f$'%expectationeta)

for k in range(len(n)):
    for j in range(5):
        ax[1].stem(n[k], varianceseta[k][j], 'rgbcm'[j])
ax[1].axhline(y = varianceeta)
ax[1].set(xticks = n, xlabel = 'n', ylabel = 'Sample variance',
          title = 'Выборочные дисперсии \n$\\theta = 0.45, ' +
                  '\\,D\\eta = %.3f$'%varianceeta);
```


Выборочные средние
$\theta = 0.45$, $M\eta = 0.483$
Sample mean
0.0
0.1
0.2
0.3
0.4
0.5
0.6
n
5
10
100
200
400
600
800
1000


Выборочные дисперсии
$\theta = 0.45$, $D\eta = 0.042$
Sample variance
0.00
0.01
0.02
0.03
0.04
0.05
n
5
10
100
200
400
600
800
1000

In [42]:

```
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = means_difeta, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('Смещение оценки: $b(\\theta)=M_{\\theta}T(x)-\\tau(\\theta)$' +
                '\n $M\\eta = %.4f$'%expectationeta);

ax[1].table(cellText = variances_difeta, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Разница выборочной дисперсии и дисперсии' +
                '\n $D\\eta = %.4f$'%varianceeta);
```

Смещение оценки: $b(\theta) = M_\theta T(x) - \tau(\theta)$

$M\eta = 0.4833$

	1	2	3	4	5
5	0.0411	-0.0161	0.1217	-0.1243	-0.1736
10	0.0445	0.0741	-0.0046	-0.0436	-0.0278
100	-0.0040	0.0115	0.0242	0.0041	0.0034
200	0.0102	-0.0156	-0.0105	0.0016	-0.0026
400	0.0065	-0.0063	-0.0058	0.0156	0.0051
600	-0.0049	-0.0158	0.0020	0.0163	0.0084
800	-0.0049	-0.0027	0.0097	0.0068	0.0096
1000	-0.0138	0.0038	0.0088	-0.0058	0.0015

Разница выборочной дисперсии и дисперсии

$D\eta = 0.0418$

	1	2	3	4	5
5	-0.0417	-0.0157	-0.0157	-0.0311	-0.0319
10	-0.0211	-0.0241	-0.0197	-0.0074	-0.0270
100	0.0001	0.0067	0.0075	-0.0053	-0.0014
200	-0.0019	-0.0025	0.0027	0.0019	0.0033
400	-0.0049	-0.0018	0.0021	0.0008	0.0006
600	0.0004	0.0011	0.0023	0.0039	-0.0015
800	-0.0008	0.0004	0.0007	-0.0017	-0.0045
1000	-0.0028	0.0016	0.0001	-0.0003	-0.0008

Текст здесь 1 в 1 повторит текст выше.

... - и что еще раз на практике подтверждает теорему - ...

Домашнее задание 3

Дискретное

Задание 1

Метод моментов

Параметр у нас только 1, потому для оценки θ методом моментов будет достаточно одного лишь $M\xi = \dfrac{\theta + 1}{2}$.

$\hat{\alpha}_1 = \dfrac{1}{n}\sum\limits_{i=1}^{n} x_i$ вычислены выше, как выборочные средние. Ими и воспользуемся.

$$M\xi = \hat{\alpha}_1 \Leftrightarrow \frac{\hat{\theta} + 1}{2} = \hat{\alpha}_1 \Leftrightarrow \hat{\theta} = 2\hat{\alpha}_1 - 1$$

Примечание: так как $\hat{\alpha}_1$, скорее всего, не целое число, а θ тут обязательно целое, то результат оценки округляется.

In [43]:
```
estimthetaxi_mm = np.around(2*meansxi-1)
estimthetaxi_mm_demo = np.array([['%.3f' % i for i in j]
                                for j in estimthetaxi_mm])

plt.table(cellText = estimthetaxi_mm_demo, rowLabels=n, colLabels=[1,2,3,4,5],
          loc='center').scale(1, 1.5)
plt.gca().set_axis_off()
plt.title('Оценка $\\theta$ методом моментов для $\\xi$');
```

Оценка θ методом моментов для ξ

	1	2	3	4	5
5	27.000	32.000	47.000	16.000	41.000
10	36.000	29.000	25.000	31.000	33.000
100	26.000	28.000	31.000	29.000	24.000
200	29.000	31.000	28.000	31.000	30.000
400	29.000	30.000	29.000	28.000	29.000
600	28.000	31.000	29.000	30.000	29.000
800	29.000	29.000	29.000	30.000	29.000
1000	29.000	28.000	29.000	28.000	28.000

Метод максимального правдоподобия

Функция правдоподобия $$L(\bar{x},\theta) = \prod_{i=1}^{n} P(x_i) = \prod_{i=1}^{n} \frac{1}{\theta}\mathtt{Ind}(1 \le x_i \le \theta) = \frac{1}{\theta^n}\prod_{i=1}^{n}\mathtt{Ind}(1 \le x_i)\cdot\mathtt{Ind}(x_i \le \theta) = $$
$$= \frac{1}{\theta^n}\mathtt{Ind}(1 \le X_{(1)})\cdot\mathtt{Ind}(X_{(n)} \le \theta)$$.

Максимальное значение этого выражения будет достигаться при минимально допустимом θ, что есть $X_{(n)}$

In [44]:
```
estimthetaxi_mmp = np.array([[i[-1] for i in j] for j in sample_xi])
# для простоты анализа sample уже отсортированы, так что последний - максимальный
estimthetaxi_mmp_demo = np.array([['%.4f' % i for i in j]
                                  for j in estimthetaxi_mmp])

plt.table(cellText = estimthetaxi_mmp_demo, rowLabels=n, colLabels=[1,2,3,4,5],
          loc='center').scale(1, 1.5)
plt.gca().set_axis_off()
plt.title('Оценка $\\theta$ методом максимального правдоподобия для $\\xi$');
```

Оценка θ методом максимального правдоподобия для ξ

	1	2	3	4	5
5	24.0000	23.0000	27.0000	15.0000	29.0000
10	28.0000	28.0000	28.0000	28.0000	28.0000
100	29.0000	29.0000	29.0000	29.0000	29.0000
200	29.0000	29.0000	29.0000	29.0000	29.0000
400	29.0000	29.0000	29.0000	29.0000	29.0000
600	29.0000	29.0000	29.0000	29.0000	29.0000
800	29.0000	29.0000	29.0000	29.0000	29.0000
1000	29.0000	29.0000	29.0000	29.0000	29.0000

Задание 2

$L(\bar{x}, \theta) = \dfrac{1}{\theta^n}\texttt{Ind}(1 \le X_{(1)}) \cdot \texttt{Ind}(X_{(n)} \le \theta) = g(T(x), \theta) \cdot h(\bar{x})$, где
$g(T(x), \theta) = \dfrac{1}{\theta^n}\texttt{Ind}(1 \le X_{(1)}) \cdot \texttt{Ind}(T(x) \le \theta)$, $h(\bar{x}) = 1$ и $T(x) = X_{(n)}$. Тогда по теореме Неймана-Фишера $T(x) = X_{(n)}$ - достаточная статистика.

Покажем ее полноту, но сперва определим функцию вероятности $T(x)$: $F_{X_{(n)}}(t) = P(X_{(n)} \le t) = \Big|$ Если максимальный элемент выборки меньше какого-то числа, то каждый элемент выборки меньше этого числа $\Big| = \prod\limits_{i=1}^{n} P(X_i \le t) = \Big|$ Каждый из них распределен одинаково $\Big| = P(X_1 \le t)^n = (F_\xi(t))^n = (\dfrac{t}{\theta})^n = \dfrac{t^n}{\theta^n}$. Тогда $P_{X_{(n)}}(t) = F_{X_{(n)}}(t) - F_{X_{(n)}}(t-1) = \dfrac{t^n}{\theta^n} - \dfrac{(t-1)^n}{\theta^n} = \dfrac{t^n - (t-1)^n}{\theta^n}$.

Теперь полнота: пусть $M\phi(T(x)) = 0$. Тогда

$$M\phi(T(x)) = \sum_{x=1}^{\theta} \phi(x) P_{X_{(n)}}(x)\, x = \sum_{x=1}^{\theta} \phi(x) \frac{x^n - (x-1)^n}{\theta^n}\, x = \frac{1}{\theta^n} \sum_{x=1}^{\theta} \phi(x)(x^n - (x-1)^n)\, x = 0 \Rightarrow$$
$$\sum_{x=1}^{\theta} \phi(x)(x^n - (x-1)^n)\, x = 0$$

. Последовательность чисел $(x^n - (x-1)^n)\, x$ от 0 до θ возрастает и каждый ее член больше 0, тогда $\phi(x)$ должна либо изначально равняться нулю, либо правильно чередовать домноженные элементы последовательности. Во втором случае построение $\phi(x)$ должно учитывать верхнюю границу саммировния, потому что, например, если при четном θ может быть и можно подобрать такую функцию, то она уже не будет работать при нечетном θ. В то же время θ не уточняется, что значит $\phi(x)$ зависит от θ, но $\phi(x)$ не зависит от θ по определению. Пришли к противоречию, значит такого варианта быть не может, а значит $\phi(x) = 0$. Полнота $T(x)$ доказана.

Таким образом, $T(x) = X_{(n)}$ - полная достаточная статистика, а значит, что произвольная функция от этой оценки $H(T(x))$ будет оптимальной оценкой своего математического ожидания. Таким образом можно построить оценку для произвольной $\tau(\theta)$

Найдем оценку $\tau(\theta) = \theta$:

$$M\,H(T(x)) = \sum_{x=1}^{\theta} H(x) P_{X_{(n)}}(x)\, x = \frac{1}{\theta^n} \sum_{x=1}^{\theta} H(x)(x^n - (x-1)^n)\, x = \theta \Leftrightarrow \sum_{x=1}^{\theta} H(x)(x^n - (x-1)^n)\, x = \theta^{n+1}.$$

Посчитать это в лоб довольно трудно, однако можно воспользоваться теоремой Lehmann–Scheffé ([ссылка (https://en.wikipedia.org/wiki/Lehmann%E2%80%93Scheff%C3%A9_theorem)](https://en.wikipedia.org/wiki/Lehmann%E2%80%93Scheff%C3%A9_theorem)). Она отверждает, что любая несмещенная оценка при условии полной достаточной статистики дает оптимальную оценку.

За несмещенную оценку возьмем такую оценку: $MX_1 = M\xi = \frac{\theta+1}{2} \Rightarrow \hat{(}\theta) = 2X_1 - 1$.
$M\hat{\theta} = M(2X_1 - 1) = 2MX_1 - 1 = 2M\xi - 1 = 2\frac{\theta+1}{2} - 1 = \theta$ - действительно несмещенная оценка. Вычислим $P(X_1 = j \big| X_{(n)} = y)$:

$j = \overline{1, y-1}$:
$P(X_1 = j, X_{(n)} = y) = P(X_1 = j, \max\limits_{2 \le i \le n} X_i = y) = P(X_1 = j) \cdot P(\max\limits_{2 \le i \le n} X_i = y) = \frac{1}{y} \cdot \frac{y^{n-1} - (y-1)^{n-1}}{y^{n-1}}$
$= \frac{y^{n-1} - (y-1)^{n-1}}{y^n}$
, и соответственно $P(X_1 = j \big| X_{(n)} = y) = \frac{P(X_1 = j, X_{(n)} = y)}{P(X_{(n)} = y} = \frac{y^{n-1} - (y-1)^{n-1}}{y^n} \cdot \frac{y^n}{y^n - (y-1)^n} = \frac{y^{n-1} - (y-1)^{n-1}}{y^n - (y-1)^n}$.

$j = y$:

$P(X_1 = j, X_{(n)} = y) = P(X_1 = j, X_{(n)} = y) = P(X_1 = y, X_2 \le y, \ldots, X_n \le y) = P(X_1 = y)$
$\cdot P(X_2 \le y, \ldots, X_n \le y) = \frac{1}{y} \cdot \frac{y^{n-1}}{y^{n-1}} = \frac{1}{y}$
, и соответственно $P(X_1 = j | X_{(n)} = y) = \frac{P(X_1=j, X_{(n)}=y)}{P(X_{(n)}=y} = \frac{1}{y} \cdot \frac{y^n}{y^n-(y-1)^n} = \frac{y^{n-1}}{y^n-(y-1)^n}$.

И, наконец, матожидание:
$M(\hat{\theta}|X_{(n)} = y) = 2M(X_1|X_{(n)} = y) - 1 = 2\left[\frac{y^{n-1}-(y-1)^{n-1}}{y^n-(y-1)^n}\sum_{j=1}^{y-1} j + \frac{y^{n-1}}{y^n-(y-1)^n} y\right] - 1$
$= \frac{\left(y^{n-1}-(y-1)^{n-1}\right) \cdot y(y-1)+2y^n}{y^n-(y-1)^n} - 1 = \frac{y^n(y-1)-y(y-1)^n+2y^n-y^n+(y-1)^n}{y^n-(y-1)^n} = \frac{y^n(y-1+1)+(y-1)^n(1-y)}{y^n-(y-1)^n} = \frac{y^{n+1}-(y-1)^{n+1}}{y^n-(y-1)^n}$
.

Таким образом, искомая оптимальная оценка θ равна $T_1(X) = \dfrac{X_{(n)}^{n+1} - (X_{(n)} - 1)^{n+1}}{X_{(n)}^n - (X_{(n)} - 1)^n}$

Задание 3

В пример равномерного распределения выше я упомянул игру "BINGO", только сильно упрощенную: вместо карточки с 25 (местами 24) числами, на карточке, указанной мной, было всего одно число.

Предлагаю еще раз сменить эту модель, основываясь просто на аппарате выбрасывания шариков. На сайте [stoloto.ru (https://www.stoloto.ru/check-ticket/bingo75/archive)](https://www.stoloto.ru/check-ticket/bingo75/archive) на момент 13 декабря в архиве доступны тиражи серий до 1012. Чтобы "поправить" случайность, ввиду того что выпавшие однажды шарики не выпадут никогда, я выберу первые 5 шариков с 20 записей и составлю из них выборку. Так как игра называется "Бинго-75", предположу что θ должно быть около 75.

Так как оптимальную оценку я не нашел, я предоставлю оценки методом моментов и методом максимального правдоподобия.

In [45]:
```
bingo75_1012 = np.array(list(map(int,'61, 40, 42, 06, 21'.split(', '))))
bingo75_1011 = np.array(list(map(int,'15, 45, 27, 32, 60'.split(', '))))
bingo75_1010 = np.array(list(map(int,'75, 36, 31, 49, 47'.split(', '))))
bingo75_1009 = np.array(list(map(int,'04, 30, 70, 59, 31'.split(', '))))
bingo75_1008 = np.array(list(map(int,'55, 27, 03, 14, 36'.split(', '))))
bingo75_1007 = np.array(list(map(int,'48, 61, 36, 15, 64'.split(', '))))
bingo75_1006 = np.array(list(map(int,'73, 30, 15, 05, 06'.split(', '))))
bingo75_1005 = np.array(list(map(int,'15, 38, 39, 31, 54'.split(', '))))
bingo75_1004 = np.array(list(map(int,'39, 02, 68, 24, 19'.split(', '))))
bingo75_1003 = np.array(list(map(int,'32, 17, 73, 02, 35'.split(', '))))
bingo75_1002 = np.array(list(map(int,'27, 64, 61, 06, 50'.split(', '))))
bingo75_1001 = np.array(list(map(int,'42, 46, 63, 73, 65'.split(', '))))
bingo75_1000 = np.array(list(map(int,'01, 09, 19, 48, 15'.split(', '))))
bingo75_0999 = np.array(list(map(int,'09, 02, 56, 08, 34'.split(', '))))
bingo75_0998 = np.array(list(map(int,'36, 10, 55, 13, 59'.split(', '))))
bingo75_0997 = np.array(list(map(int,'46, 06, 05, 63, 08'.split(', '))))
bingo75_0996 = np.array(list(map(int,'33, 64, 24, 03, 10'.split(', '))))
bingo75_0995 = np.array(list(map(int,'21, 33, 02, 07, 06'.split(', '))))
bingo75_0994 = np.array(list(map(int,'74, 50, 42, 30, 17'.split(', '))))
bingo75_0993 = np.array(list(map(int,'70, 57, 65, 74, 68'.split(', '))))
bingo75_0993to1012 = np.sort(np.concatenate(
     (bingo75_1012, bingo75_1011, bingo75_1010, bingo75_1009,
      bingo75_1008, bingo75_1007, bingo75_1006, bingo75_1005,
      bingo75_1004, bingo75_1003, bingo75_1002, bingo75_1001,
      bingo75_1000, bingo75_0999, bingo75_0998, bingo75_0997,
      bingo75_0996, bingo75_0995, bingo75_0994, bingo75_0993)))

bingo75_0993to1012_mean =  xi_sample_mean(bingo75_0993to1012)
bingo75_0993to1012_variance =  xi_sample_variance(bingo75_0993to1012)

estimbingo75_mm = np.around(2*bingo75_0993to1012_mean-1)

estimbingo75_mmp = bingo75_0993to1012[-1]

X_n = int(estimbingo75_mmp)
estimbingo75_opt = (X_n**101-(X_n-1)**101)/(X_n**100-(X_n-1)**100)

print('"Бинго-75"\n'+
      'Выборочное среднее: %.3f, \n'%bingo75_0993to1012_mean+\
      'Выборочная дисперсия: %.3f, \n'%bingo75_0993to1012_variance+\
      'Оценка методом моментов: %d, \n'%estimbingo75_mm+\
      'Оценка методом максимального правдоподобия: %d \n'%estimbingo75_mmp+\
      'Оптимальная оценка: %d'%estimbingo75_opt)
```

```
"Бинго-75"
Выборочное среднее: 35.060,
Выборочная дисперсия: 515.016,
Оценка методом моментов: 69,
Оценка методом максимального правдоподобия: 75
Оптимальная оценка: 75
```

Абсолютно непрерывное

Задание 1

Метод моментов

$M\xi = \dfrac{1+\theta}{3}$. $\hat{\alpha}_1 = \dfrac{1}{n}\sum_{i=1}^{n} x_i$ вычислены выше, как выборочные средние.

$$M\xi = \frac{1+\theta}{3} = \hat{\alpha}_1 \Leftrightarrow \frac{1+\hat{\theta}}{3} = \hat{\alpha}_1 \Leftrightarrow \hat{\theta} = 3\hat{\alpha}_1 - 1$$

In [46]:
```
estimthetaeta_mm = 3*meanseta-1
estimthetaeta_mm_demo = np.array([['%.4f' % i for i in j]
                                  for j in estimthetaeta_mm])

plt.table(cellText = estimthetaeta_mm_demo, rowLabels=n, colLabels=[1,2,3,4,5],
          loc='center').scale(1, 1.5)
plt.gca().set_axis_off()
plt.title('Оценка $\\theta$ методом моментов для $\\eta$');
```

Оценка θ методом моментов для η

	1	2	3	4	5
5	0.5734	0.4017	0.8152	0.0770	-0.0707
10	0.5834	0.6722	0.4363	0.3192	0.3666
100	0.4380	0.4845	0.5225	0.4623	0.4603
200	0.4807	0.4032	0.4184	0.4547	0.4421
400	0.4696	0.4312	0.4327	0.4968	0.4654
600	0.4354	0.4025	0.4559	0.4988	0.4753
800	0.4354	0.4419	0.4791	0.4703	0.4788
1000	0.4086	0.4615	0.4765	0.4327	0.4545

Метод максимального правдоподобия

Функция правдоподобия $L(\bar{x}, \theta) = \prod_{i=1}^{n} f_\theta(x_i) = \prod_{i=1}^{n} \left(\dfrac{2x_i}{\theta} \cdot \mathtt{Ind}(0 \le x_i \le \theta) + \dfrac{2(1-x_i)}{1-\theta} \cdot \mathtt{Ind}(\theta < x_i \le 1)\right)$.

Рассмотреть ни производную, ни производную логарифма такой функции правдоподобия не получится (или как минимум очень сложно и ничего толкового не даст), потому попробую прибегнуть к логике.

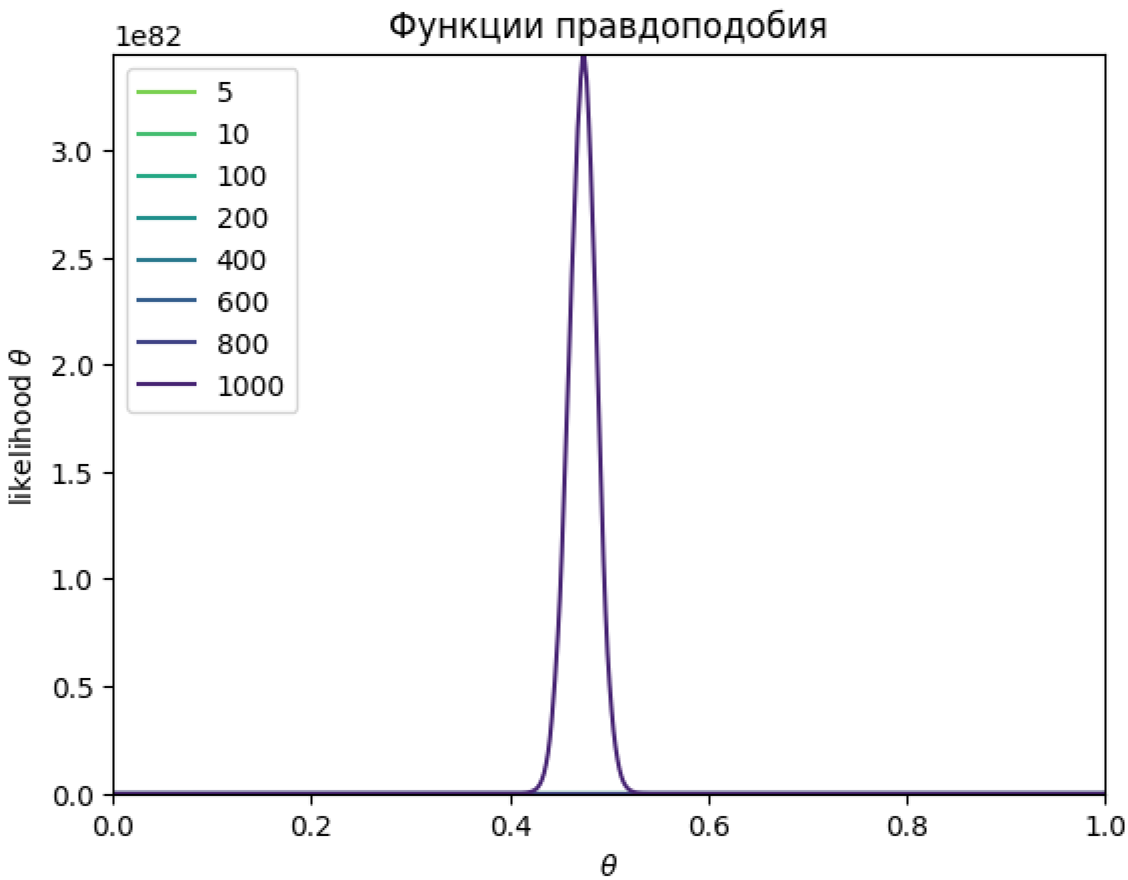
Попробуем задать эту функцию, посмотреть ее график, и найти искомое θ программными методами

In [47]:

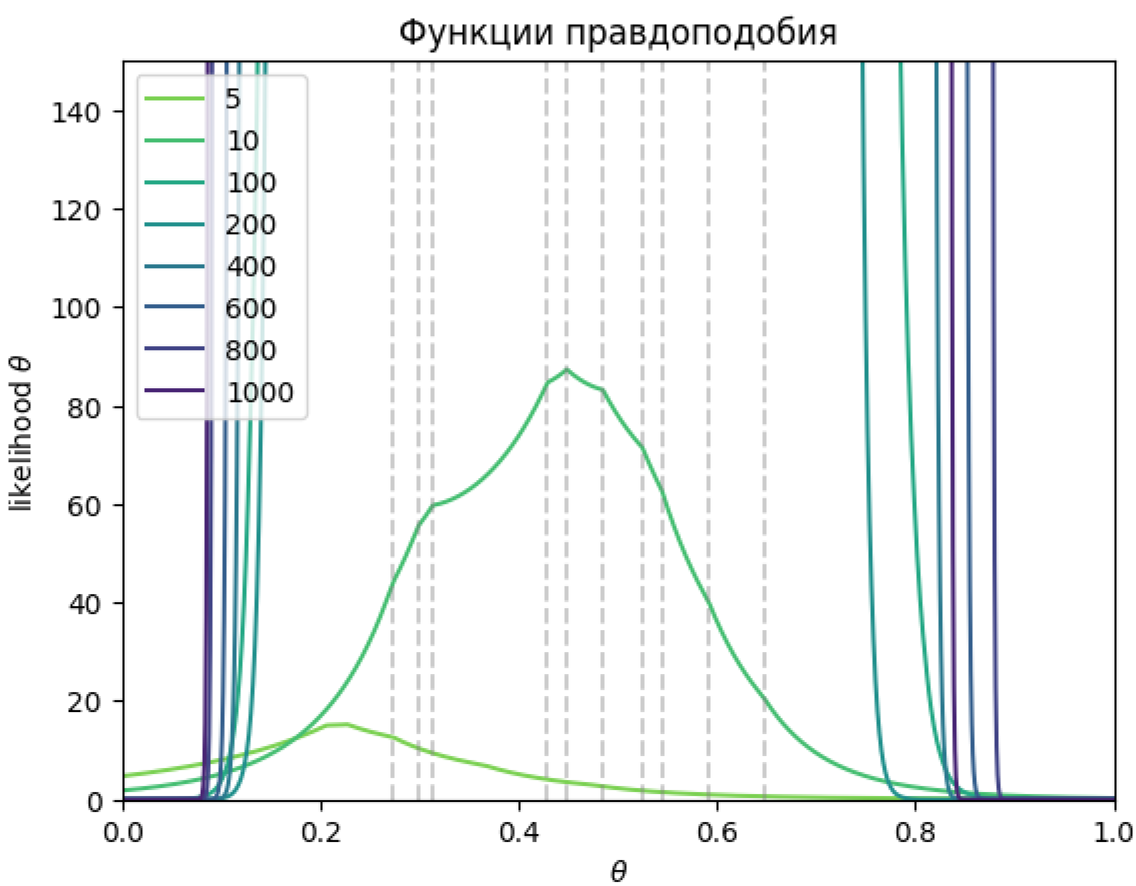
```
likelihood_func_eta = lambda sample, theta: \
    np.prod([2*x/theta if x<theta else 2*(1-x)/(1-theta)
             for x in sample])
likelihood_eta = np.array([[[likelihood_func_eta(sample_eta[j][i], theta)
                             for theta in np.linspace(0,1,1000)]
                            for i in range(5)] for j in range(len(n))])
```

In [48]:

```
for i in range(8):
    plt.plot(np.linspace(0,1,1000), likelihood_eta[i][2], color = colors[i])
plt.gca().set(xmargin = 0, ymargin = 0, xlabel = '$\\theta$',
              ylabel = 'likelihood $\\theta$', title = 'Функции правдоподобия')
plt.legend(n, loc='upper left');
```



In [49]:
```
for i in range(8):
    plt.plot(np.linspace(0,1,1000), likelihood_eta[i][4], color = colors[i])
for xc in sample_eta[1][4]:
    plt.axvline(x=xc, ls='--', c='#808080', alpha=0.4)
plt.gca().set(xmargin = 0, ymargin = 0, xlabel = '$\\theta$',
              ylabel = 'likelihood $\\theta$', title = 'Функции правдоподобия')
plt.legend(n, loc='upper left')
plt.ylim([0, 150]);
```



Код выше реализует демонстрацию функции правдоподобия для θ от 0 до 1 с шагом в 0.001. Как, на самом деле, стоило ожидать, значение функции сильно зависит от количества элементов выборки (а именно максимум ограничен 2^n), поэтому на первом графике выборок длины меньше 1000 даже не видно, они ушли в 0 ввиду масштабирования. Для этого я построил второй график, на котором уже ограничил значения y от нуля до 150, чтобы сделать выводы на простой выборке. В этот отрезок оси y хорошо попадает выборка длины 10.

График имеет 10 пиков, которые совпадают со значениями выборки (в силу изначальной неровности $f_\eta(x)$ по θ). Программными методами найти такой максимум труда не составит, однако сформулировать оценку математическими методами будет проблематично. Код приведен ниже, как и таблица для выборок.

Однако можно описать алгоритм:

1. Вычислить все пары $\big(x_i, L(\bar{x}, \theta)\big)$ для всех $\theta \in \{x_1, x_2, \ldots, x_n\}$;

2. Выбрать из полученных значений пару с максимальным вторым параметром;

3. Вернуть первый параметр выбранной пары.

In [50]:
```
def argmax_mmp_eta(likelihood_arr):
    x, mx = likelihood_arr[0]
    for i in likelihood_arr:
        if i[1]>mx: x, mx = i
    return x
estimthetaeta_mmp = np.array([[argmax_mmp_eta(
    [(x, likelihood_func_eta(sample_eta[j][i], x)) for x in sample_eta[j][i]])
                               for i in range(5)] for j in range(len(n))])
estimthetaeta_mmp_demo = np.array([['%.4f' % i for i in j]
                                   for j in estimthetaeta_mmp])

plt.table(cellText = estimthetaeta_mmp_demo, rowLabels=n, colLabels=[1,2,3,4,5],
          loc='center').scale(1, 1.5)
plt.gca().set_axis_off()
plt.title('Оценка $\\theta$ методом максимального правдоподобия для $\\eta$');
```

Оценка θ методом максимального правдоподобия для η

	1	2	3	4	5
5	0.5261	0.5771	0.5657	0.3375	0.2276
10	0.5610	0.5395	0.4634	0.3611	0.4487
100	0.4340	0.4670	0.4693	0.4634	0.4449
200	0.4624	0.4508	0.3977	0.4080	0.4524
400	0.4893	0.4420	0.4081	0.5183	0.4436
600	0.4488	0.4177	0.4476	0.4612	0.4692
800	0.4567	0.4527	0.4744	0.4655	0.4674
1000	0.4234	0.4486	0.4745	0.4349	0.4552

Задание 2

☻

Задание 3

В примерах выше я указал, что треугольное распределение используется часто, но только потому что оно простое по формулировке, а значит для ее построение требуется мало информации о реальном событии. Так, предлагаю проанализировать методом моментов и методом максимального правдоподобия время моих матчей в игре "Dota 2". Так как не будет учитываться пик персонажей и версия игры, то инфомации достаточно мало. Но ради некоторого ограничения выберу данные игр только в режиме "All Pick" рейтинговом и обычном. Игры в других режимах учитывать не буду, это уже относительно другая игра.

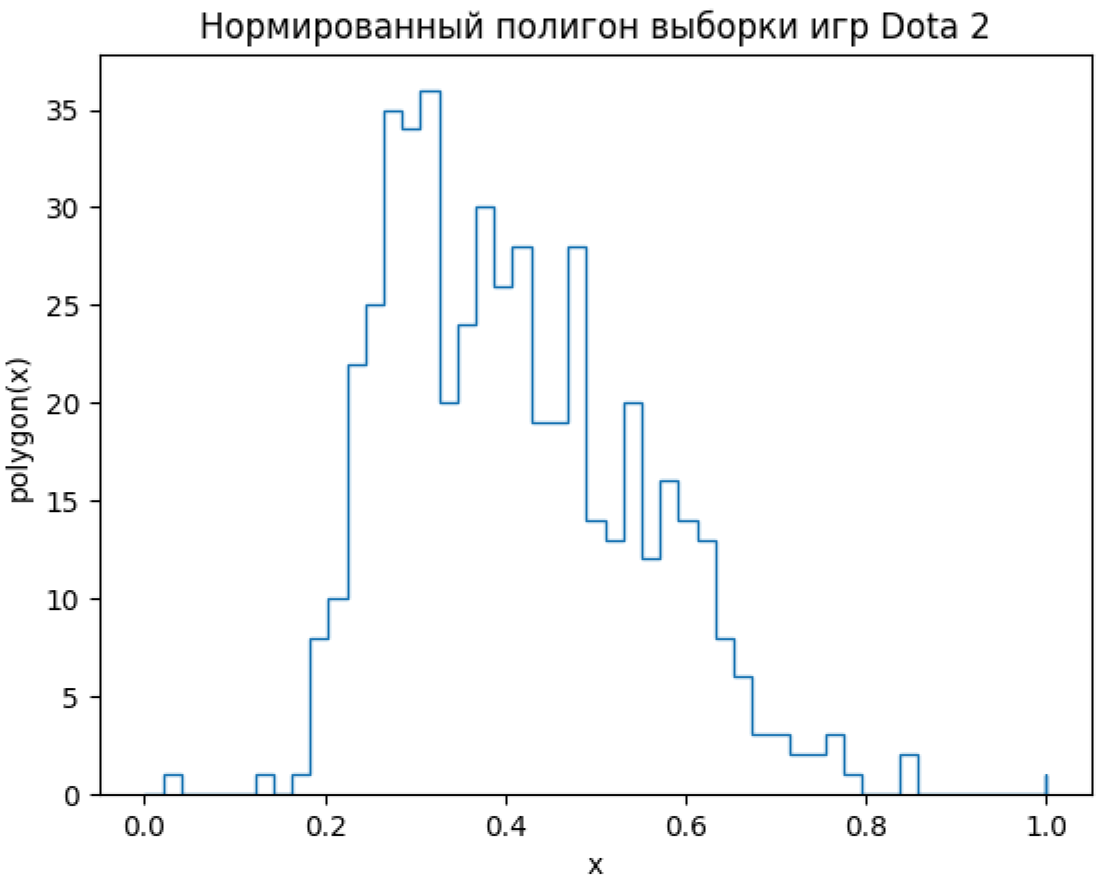
Так как треугольное распределение определено при значениях от 0 до 1, а время игры может варьироваться от 9 минут, до двух часов, время игры я переведу в секунды, вычислю минимальное и максимальное значение и для каждого члена выборки вычту минимальное и поделю результат на разницу максимального и минимального. Тогда получу значения в отрезке от 0 до 1, эту выборки и буду оценивать.

Скрипт, который достает данные моих игр, нормирует их и сохраняет в файл, будет приложен к репозиторию github этого домашнего задания. Также, в силу того что для работы со Steam Web API требуется личный идентификатор, из соображений безопасности, этот идентификатор я уберу из кода (то есть для проверки работоспособности скрипта потребуется ввести его самому).

В силу ограничений самого Steam Web API, выборка будет состоять из 500 последних матчей.

In [51]:
```
dota2_sample = np.sort(np.load('dota2_sample.npz')['dota2_sample'])

dota2_mean =  eta_sample_mean(dota2_sample)
dota2_variance =  eta_sample_variance(dota2_sample)

estimdota2_mm = 3*dota2_mean-1

# там есть 0 и 1
estimdota2_mmp = argmax_mmp_eta(
    [(x, likelihood_func_eta(dota2_sample[1:-1], x)) for x in dota2_sample[1:-1]])

print('"Dota 2"\n'+\
      'Выборочное среднее: %.3f, \n'%dota2_mean+\
      'Выборочная дисперсия: %.3f, \n'%dota2_variance+\
      'Оценка методом моментов: %.3f, \n'%estimdota2_mm+\
      'Оценка методом максимального правдоподобия: %.3f'%estimdota2_mmp)
print('\nОценки, возвращенные в секунды (x*4764+50)\n'+\
      'Выборочное среднее: %d, \n'%(dota2_mean*4764+50)+\
      'С выборочной дисперсией простое линейное преобразование не сработает\n'+\
      'Оценка методом моментов: %d, \n'%(estimdota2_mm*4764+50)+\
      'Оценка методом максимального правдоподобия: %d'%(estimdota2_mmp*4764+50))
# и полигон, потому что мне интересно
X_poldota2 = np.linspace(0,1,50)
Y_poldota2 = eta_pilygon(dota2_sample, X_poleta)
plt.stairs(Y_poldota2, np.append(X_poldota2,1))
plt.xlabel('x')
plt.ylabel('polygon(x)')
plt.title('Нормированный полигон выборки игр Dota 2');
```

```
"Dota 2"
Выборочное среднее: 0.390,
Выборочная дисперсия: 0.019,
Оценка методом моментов: 0.170,
Оценка методом максимального правдоподобия: 0.291

Оценки, возвращенные в секунды (x*4764+50)
Выборочное среднее: 1907,
С выборочной дисперсией простое линейное преобразование не сработает
Оценка методом моментов: 858,
Оценка методом максимального правдоподобия: 1433
```



Домашнее задание 4

Дискретное

Задание 1

Критерий согласия Колмагорова (Смирнова)

В случае дискретного распределения потребуется сперва дополнить распределение так, чтобы получилось непрерывное. Для этого вместо $F(x)$ будем рассматривать $U(x) = F(x-0) + \gamma\big(F(x) - F(x-0)\big)$, где $F(x-0) = \lim_{x \to x-0} F(x)$ и $\gamma \sim R[0,1]$. В неразрывных точках $F(x)$ обращает $F(x) - F(x-0)$ в ноль, тем самым $U(x) = F(x-0)$. В точках разрыва же $F(x) - F(x-0) \neq 0$, а значит случайная величина γ выберет случайное число в вероятностном отрезке $F(x) - F(x-0)$, после чего к нему будет прибавлено само F(x). В силу случайности и непрерывности γ мы получим непрерывное случайное распределение, которое уже можно использовать для критерия согласия Колмагорова.

В критерии рассматривается величина $D_n = \sup\limits_{x\in\mathbb{R}} \left|\hat{F}_n(x) - F(x)\right|$, что в силу способа задания $\hat{F}_n(x)$ все равно что $D_n = max\{D_n^-, D_n^+\}$, где $D_n^- = \max\limits_{1\le k\le n} \left|F(x_{(k)}) - \frac{k-1}{n}\right|$ и $D_n^+ = \max\limits_{1\le k\le n} \left|\frac{k}{n} - F(x_{(k)})\right|$. Далее D_n домножается на $\sqrt{n}$ и сравнивается с квантилем t_α распределения Колмагорова. Если $\sqrt{n}D_n \ge t_\alpha$ то гипотеза о верности предположения о распределении выборки отвергается. Иначе - считаем, что данные не противоречат гипотезе.

В силу асимптотичности критерия, для выборки длины меньшей 20 предлагается вычислять не $\sqrt{n}D_n$, а $S_n = \frac{6nD_n+1}{6\sqrt{n}}$, которая также сходится к распределению Колмагорова, но быстрее.

Выберем ошибку первого рода α за 0.05. Для такого α квантиль $t_\alpha = 1.36$ по табличным данным.

In [52]:
```
# модифицированное распределение
def xi_distr_real_continuous(x, theta=29):
    if np.issubdtype(x, np.integer):#если мы попали в целое
        left=xi_distr_real(x-1, theta)
        right=xi_distr_real(x, theta)
        return left+np.random.uniform()*(right-left)
    return xi_distr_real(int(x), theta)

def xi_Dn(sample, theta=29):
    n=len(sample)
    res=-1
    for k in range(n):
        t=np.abs(k/n-xi_distr_real_continuous(sample[k], theta))#plus
        if t>res: res=t
        t=np.abs(xi_distr_real_continuous(sample[k], theta)-(k-1)/n)#minus
        if t>res: res=t
    return res
# модификация для малых выборок
def xi_Sn(sample, theta=29):
    return (6*len(sample)*xi_Dn(sample, theta)+1)/(6*np.sqrt(len(sample)))

kolmag_xi = np.array([[xi_Dn(sample_xi[k][j])*np.sqrt(n[k]) if n[k]>20 else xi_Sn(
sample_xi[k][j])
                       for j in range(5)] for k in range(len(n))])
kolmag_xi_demo = np.array([['%.4f' % i for i in j] for j in kolmag_xi])

kolmag_xi_good = np.array([[i<1.36 for i in j] for j in kolmag_xi])
kolmag_xi_good_demo = np.array([['Подходит' if i else 'Не подходит' for i in j]
                                for j in kolmag_xi_good])
```

In [53]:

```
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = kolmag_xi_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('$\\sqrt{n}D_n$ критерия согласия Колмагорова')

ax[1].table(cellText = kolmag_xi_good_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Подходит ли выборка по критерию');
```

$\sqrt{n}D_n$ критерия согласия Колмагорова

	1	2	3	4	5
5	1.1607	1.1436	1.9196	0.7735	1.5846
10	1.3454	0.9893	0.5989	0.9291	1.3597
100	1.1933	0.7082	1.1862	0.8606	1.7495
200	1.0120	1.4085	1.0014	1.4517	1.3466
400	1.1440	1.3931	1.0404	1.7086	1.0776
600	1.8384	2.2524	1.1786	1.6269	1.2634
800	1.3059	1.3798	1.4203	1.7527	1.2914
1000	1.8439	1.7598	1.6213	1.7591	1.8357

Подходит ли выборка по критерию

	1	2	3	4	5
5	Подходит	Подходит	Не подходит	Подходит	Не подходит
10	Подходит	Подходит	Подходит	Подходит	Подходит
100	Подходит	Подходит	Подходит	Подходит	Не подходит
200	Подходит	Не подходит	Подходит	Не подходит	Подходит
400	Подходит	Не подходит	Подходит	Не подходит	Подходит
600	Не подходит	Не подходит	Подходит	Не подходит	Подходит
800	Подходит	Не подходит	Не подходит	Не подходит	Подходит
1000	Не подходит	Не подходит	Не подходит	Не подходит	Не подходит

Критерий хи-квадрат

Выберем разбиение на $N=5$ участков: от 1 до 6, от 6 до 12, от 12 до 18, от 18 до 24 и от 24 до 30. Обозначим минимальную границу отрезка за a_i, максимальную за b_i, $i=\overline{1,5}$. Тогда $p_i=F(b_i)-F(a_i)=\frac{b_i}{\theta}-\frac{a_i}{\theta}=\frac{b_i-a_i}{\theta}=\frac{6}{\theta}=\frac{6}{29}$ для любых $i=\overline{2,5}$, для $i=1$ $p_i=\frac{5}{29}$. Статистика $X_N^2=\sum_{i=1}^N=\frac{\left(\nu_i^{(n)}-np_i\right)^2}{np_i}$, где $\nu_i^{(n)}=\sum_{k=1}^n Ind(X_k=i)$ - число элементов выборки, попавших в выделенный отрезок, сравнивается с квантилем распределения χ^2_{N-1}.

Так как критерий хи-квадрат является асимптотическим, для выборок длины меньше 20, критерий использоваться не будет.

Выберем ошибку первого рода α за 0.05. Для такого α при $N-1=4$ степеней свободы квантиль $t_\alpha=9.5$ по табличным данным.

```
In [54]: def xi_nu1_vec(sample):
             res=np.zeros(5)
             for x in sample:
                 for i in range(5):
                     if x<6*(i+1):
                         res[i]+=1
                         break
             return res
         def xi_hi1_sqr(sample, theta=29):
             n=len(sample)
             nu=xi_nu1_vec(sample)
             res=0
             for i in range(5):
                 res += (nu[i]-n/theta*(5 if i==0 else 6))**2/(n/theta*(5 if i==0 else 6))
             return res

         hi1sqr_xi = np.array([[xi_hi1_sqr(sample_xi[k][j]) if n[k]>20 else None
                                for j in range(5)] for k in range(len(n))])
         hi1sqr_xi_demo = np.array([['%.4f'%i if i!=None else '-' for i in j] for j in hi1s
         qr_xi])

         hi1sqr_xi_good = np.array([[i<9.5 if i!=None else None for i in j] for j in hi1sqr
         _xi])
         hi1sqr_xi_good_demo = np.array([[('Подходит' if i else 'Не подходит') if i!=None e
         lse '-' for i in j]
                                          for j in hi1sqr_xi_good])
```

```python
In [55]: fig, ax = plt.subplots(2,1, figsize=(7,7))
         ax[0].table(cellText = hi1sqr_xi_demo, rowLabels=n, colLabels=[1,2,3,4,5],
                     loc='center').scale(1, 1.5)
         ax[0].set_axis_off()
         ax[0].set_title('$X_N^2$ критерия согласия хи-квадрат при первом разбиении')

         ax[1].table(cellText = hi1sqr_xi_good_demo, rowLabels=n, colLabels=[1,2,3,4,5],
                     loc='center').scale(1, 1.5)
         ax[1].set_axis_off()
         ax[1].set_title('Подходит ли выборка по критерию');
```

X_N^2 критерия согласия хи-квадрат при первом разбиении

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	5.6180	3.5203	6.8553	3.2497	14.3567
200	4.8898	9.3510	1.8593	2.8115	3.7685
400	4.5403	2.3847	0.4803	3.3078	2.5442
600	5.3347	11.9628	0.4837	3.8622	2.2559
800	2.4856	1.8609	0.9027	5.0762	5.5342
1000	4.5078	3.7905	5.2318	2.7610	6.9535

Подходит ли выборка по критерию

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	Подходит	Подходит	Подходит	Подходит	Не подходит
200	Подходит	Подходит	Подходит	Подходит	Подходит
400	Подходит	Подходит	Подходит	Подходит	Подходит
600	Подходит	Не подходит	Подходит	Подходит	Подходит
800	Подходит	Подходит	Подходит	Подходит	Подходит
1000	Подходит	Подходит	Подходит	Подходит	Подходит

Возьмем другое разбиаение: пусть на этот раз N=10, и отрезки все также равны по длине, кроме первого. Тогда $p_i = \frac{3}{29}$ для $i = \overline{2, 10}$, для $i = 1$ $p_i = \frac{2}{29}$. В этом случае степеней свободы будет $N - 1 = 9$ и квантиль $t_\alpha = 16.9$ по табличным данным.

In [56]:
```
def xi_nu2_vec(sample):
    res=np.zeros(10)
    for x in sample:
        for i in range(10):
            if x<3*(i+1):
                res[i]+=1
                break
    return res
def xi_hi2_sqr(sample, theta=29):
    n=len(sample)
    nu=xi_nu2_vec(sample)
    res=0
    for i in range(10):
        res += (nu[i]-n/theta*(2 if i==0 else 3))**2/(n/theta*(2 if i==0 else 3))
    return res

hi2sqr_xi = np.array([[xi_hi2_sqr(sample_xi[k][j]) if n[k]>20 else None
                       for j in range(5)] for k in range(len(n))])
hi2sqr_xi_demo = np.array([['%.4f'%i if i!=None else '-' for i in j] for j in hi2s
qr_xi])

hi2sqr_xi_good = np.array([[i<16.9 if i!=None else None for i in j] for j in hi2sq
r_xi])
hi2sqr_xi_good_demo = np.array([[('Подходит' if i else 'Не подходит') if i!=None e
lse '-' for i in j]
                                for j in hi2sqr_xi_good])
```

In [57]:
```
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = hi2sqr_xi_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('$X_N^2$ критерия согласия хи-квадрат при втором разбиении')

ax[1].table(cellText = hi2sqr_xi_good_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Подходит ли выборка по критерию');
```

X_N^2 критерия согласия хи-квадрат при втором разбиении

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	8.8950	11.2150	9.0400	8.3150	21.0267
200	7.9300	11.0233	7.5433	5.9000	12.5700
400	6.1933	8.1267	3.0033	6.1571	4.7433
600	8.1703	22.8636	6.6236	5.2058	4.8756
800	3.8558	4.9735	1.4694	6.8525	6.8767
1000	6.6093	13.9560	7.2522	5.0047	12.2402

Подходит ли выборка по критерию

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	Подходит	Подходит	Подходит	Подходит	Не подходит
200	Подходит	Подходит	Подходит	Подходит	Подходит
400	Подходит	Подходит	Подходит	Подходит	Подходит
600	Подходит	Не подходит	Подходит	Подходит	Подходит
800	Подходит	Подходит	Подходит	Подходит	Подходит
1000	Подходит	Подходит	Подходит	Подходит	Подходит

Критерий согласия Колмагорова (Смирнова) для сложной гипотезы

Для этого критерия сперва находится оценка методом максимального правдоподобия, затем статистика вычисляется, и к ней критерий применяется уже как к простой гипотезе. Метод максимального правдоподобия уже реализован в домашней работе 3, так что числа будут взяты оттуда. α и t_α оставим прежними.

In [58]:
```
# я модифицировал функции выше, чтобы просто дописывать туда theta
# и уже с данной theta будут идти рассчеты
# при неуказанной theta она по умолчанию равна 29

kolmag_xi_tough = np.array([[xi_Dn(sample_xi[k][j],estimthetaxi_mmp[k][j])*np.sqrt
(n[k])
                             if n[k]>20 else xi_Sn(sample_xi[k][j])
                             for j in range(5)] for k in range(len(n))])
kolmag_xi_tough_demo = np.array([['%.4f' % i for i in j] for j in kolmag_xi_tough
])

kolmag_xi_tough_good = np.array([[i<1.36 for i in j] for j in kolmag_xi_tough])
kolmag_xi_tough_good_demo = np.array([['Подходит' if i else 'Не подходит' for i in
 j]
                                 for j in kolmag_xi_tough_good])
```

In [59]:
```
fig, ax = plt.subplots(3,1, figsize=(7,10.5))
ax[0].table(cellText = estimthetaxi_mmp_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('Оценка $\\theta$ методом максимального правдоподобия')

ax[1].table(cellText = kolmag_xi_tough_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('$\\sqrt{n}D_n$ критерия согласия Колмагорова')

ax[2].table(cellText = kolmag_xi_tough_good_demo, rowLabels=n, colLabels=[1,2,3,4,
5],
            loc='center').scale(1, 1.5)
ax[2].set_axis_off()
ax[2].set_title('Подходит ли выборка по критерию');
```

Оценка θ методом максимального правдоподобия

	1	2	3	4	5
5	24.0000	23.0000	27.0000	15.0000	29.0000
10	28.0000	28.0000	28.0000	28.0000	28.0000
100	29.0000	29.0000	29.0000	29.0000	29.0000
200	29.0000	29.0000	29.0000	29.0000	29.0000
400	29.0000	29.0000	29.0000	29.0000	29.0000
600	29.0000	29.0000	29.0000	29.0000	29.0000
800	29.0000	29.0000	29.0000	29.0000	29.0000
1000	29.0000	29.0000	29.0000	29.0000	29.0000

$\sqrt{n}D_n$ критерия согласия Колмагорова

	1	2	3	4	5
5	1.1406	1.1515	1.9248	0.7232	1.5792
10	1.3428	1.0311	0.5612	0.9259	1.3542
100	1.2384	0.7498	1.1583	0.7692	1.6511
200	1.0438	1.4099	1.1103	1.3593	1.4199
400	1.1629	1.5385	1.0299	1.5183	1.2547
600	1.6858	2.2900	1.2321	1.5040	1.2326
800	1.3337	1.2430	1.4553	1.7226	1.3008
1000	1.8937	1.8917	1.6144	1.7238	1.7427

Подходит ли выборка по критерию

	1	2	3	4	5
5	Подходит	Подходит	Не подходит	Подходит	Не подходит
10	Подходит	Подходит	Подходит	Подходит	Подходит
100	Подходит	Подходит	Подходит	Подходит	Не подходит
200	Подходит	Не подходит	Подходит	Подходит	Не подходит
400	Подходит	Не подходит	Подходит	Не подходит	Подходит
600	Не подходит	Не подходит	Подходит	Не подходит	Подходит
800	Подходит	Подходит	Не подходит	Не подходит	Подходит
1000	Не подходит	Не подходит	Не подходит	Не подходит	Не подходит

Критерий согласия хи-квадрат для сложной гипотезы

Сперва берется оценка методом максимльного правдоподобия, но не для распределения предполагаемого, а для полиномиального распределения, построенном на разбиении этого самого предполагаемого распределения. α и оставим прежней, t_α же изменится, так как изменится количество степеней свободны. Так как будет взята оценка максимального правдоподобия, получим N-2=3 здесь, и $t_\alpha = 7.8$

Найдем оценку $\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} \left(p_i(\theta)\right)^{\nu_i}$:

$\prod_{i=1}^{N} \left(p_i(\theta)\right)^{\nu_i} = \prod_{i=1}^{N} \left(\frac{b_i - a_i}{\theta}\right)^{\nu_i} = \theta^{-\sum_{i=1}^{N} \nu_i} \prod_{i=1}^{N} \left(b_i - a_i\right)^{\nu_i} = \theta^{-n} \prod_{i=1}^{N} \left(b_i - a_i\right)^{\nu_i}$, где a_i и b_i соответственно минимум и максимум рассматриваемого отрезка. Тогда $\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} \left(p_i(\theta)\right)^{\nu_i} = \underset{\theta}{\operatorname{argmax}} \left[\theta^{-n} \prod_{i=1}^{N} (b_i - a_i)^{\nu_i}\right]$ будет оценкой максимального правдоподобия, при минимально возможном θ. Что значит $\theta = X_{(n)}$, что есть оценка максимального правдоподобия, предложенная в третьем домашнем задании.

In [60]:
```
# в написанном выше я снова пришел к омп третьего дз
# что значит я опять могу просто модифицировать функции
# как и в предыдущем задании

hi1sqr_xi_tough = np.array([[xi_hi1_sqr(sample_xi[k][j],estimthetaxi_mmp[k][j])
                             if n[k]>20 else None for j in range(5)]
                            for k in range(len(n))])
hi1sqr_xi_tough_demo = np.array([['%.4f'%i if i!=None else '-' for i in j] for j i
n hi1sqr_xi_tough])

hi1sqr_xi_tough_good = np.array([[i<7.8 if i!=None else None for i in j] for j in
hi1sqr_xi_tough])
hi1sqr_xi_tough_good_demo = np.array([[('Подходит' if i else 'Не подходит') if i!=
None else '-' for i in j]
                                  for j in hi1sqr_xi_tough_good])
```

In [61]:
```
fig, ax = plt.subplots(3,1, figsize=(7,10.5))
ax[0].table(cellText = estimthetaxi_mmp_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('Оценка $\\theta$ методом максимального правдоподобия')

ax[1].table(cellText = hi1sqr_xi_tough_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('$X_N^2$ критерия согласия хи-квадрат при первом разбиении')

ax[2].table(cellText = hi1sqr_xi_tough_good_demo, rowLabels=n, colLabels=[1,2,3,4,
5],
            loc='center').scale(1, 1.5)
ax[2].set_axis_off()
ax[2].set_title('Подходит ли выборка по критерию');
```

Оценка θ методом максимального правдоподобия

	1	2	3	4	5
5	24.0000	23.0000	27.0000	15.0000	29.0000
10	28.0000	28.0000	28.0000	28.0000	28.0000
100	29.0000	29.0000	29.0000	29.0000	29.0000
200	29.0000	29.0000	29.0000	29.0000	29.0000
400	29.0000	29.0000	29.0000	29.0000	29.0000
600	29.0000	29.0000	29.0000	29.0000	29.0000
800	29.0000	29.0000	29.0000	29.0000	29.0000
1000	29.0000	29.0000	29.0000	29.0000	29.0000

X_N^2 критерия согласия хи-квадрат при первом разбиении

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	5.6180	3.5203	6.8553	3.2497	14.3567
200	4.8898	9.3510	1.8593	2.8115	3.7685
400	4.5403	2.3847	0.4803	3.3078	2.5442
600	5.3347	11.9628	0.4837	3.8622	2.2559
800	2.4856	1.8609	0.9027	5.0762	5.5342
1000	4.5078	3.7905	5.2318	2.7610	6.9535

Подходит ли выборка по критерию

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	Подходит	Подходит	Подходит	Подходит	Не подходит
200	Подходит	Не подходит	Подходит	Подходит	Подходит
400	Подходит	Подходит	Подходит	Подходит	Подходит
600	Подходит	Не подходит	Подходит	Подходит	Подходит
800	Подходит	Подходит	Подходит	Подходит	Подходит
1000	Подходит	Подходит	Подходит	Подходит	Подходит

И для второго разбиения. N-2=8, $t_\alpha = 15.5$

In [62]:
```
hi2sqr_xi_tough = np.array([[xi_hi2_sqr(sample_xi[k][j],estimthetaxi_mmp[k][j])
                             if n[k]>20 else None for j in range(5)]
                            for k in range(len(n))])
hi2sqr_xi_tough_demo = np.array([['%.4f'%i if i!=None else '-' for i in j] for j i
n hi2sqr_xi_tough])

hi2sqr_xi_tough_good = np.array([[i<15.5 if i!=None else None for i in j] for j in
 hi2sqr_xi_tough])
hi2sqr_xi_tough_good_demo = np.array([[('Подходит' if i else 'Не подходит') if i!=
None else '-' for i in j]
                                  for j in hi2sqr_xi_tough_good])
```

In [63]:
```
fig, ax = plt.subplots(3,1, figsize=(7,10.5))
ax[0].table(cellText = estimthetaxi_mmp_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('Оценка $\\theta$ методом максимального правдоподобия')

ax[1].table(cellText = hi2sqr_xi_tough_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('$X_N^2$ критерия согласия хи-квадрат при втором разбиении')

ax[2].table(cellText = hi2sqr_xi_tough_good_demo, rowLabels=n, colLabels=[1,2,3,4,
5],
            loc='center').scale(1, 1.5)
ax[2].set_axis_off()
ax[2].set_title('Подходит ли выборка по критерию');
```

Оценка θ методом максимального правдоподобия

	1	2	3	4	5
5	24.0000	23.0000	27.0000	15.0000	29.0000
10	28.0000	28.0000	28.0000	28.0000	28.0000
100	29.0000	29.0000	29.0000	29.0000	29.0000
200	29.0000	29.0000	29.0000	29.0000	29.0000
400	29.0000	29.0000	29.0000	29.0000	29.0000
600	29.0000	29.0000	29.0000	29.0000	29.0000
800	29.0000	29.0000	29.0000	29.0000	29.0000
1000	29.0000	29.0000	29.0000	29.0000	29.0000

X_N^2 критерия согласия хи-квадрат при втором разбиении

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	8.8950	11.2150	9.0400	8.3150	21.0267
200	7.9300	11.0233	7.5433	5.9000	12.5700
400	6.1933	8.1267	3.0033	6.1571	4.7433
600	8.1703	22.8636	6.6236	5.2058	4.8756
800	3.8558	4.9735	1.4694	6.8525	6.8767
1000	6.6093	13.9560	7.2522	5.0047	12.2402

Подходит ли выборка по критерию

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	Подходит	Подходит	Подходит	Подходит	Не подходит
200	Подходит	Подходит	Подходит	Подходит	Подходит
400	Подходит	Подходит	Подходит	Подходит	Подходит
600	Подходит	Не подходит	Подходит	Подходит	Подходит
800	Подходит	Подходит	Подходит	Подходит	Подходит
1000	Подходит	Подходит	Подходит	Подходит	Подходит

Задание 2

Критерий однородности Смирнова

Критерий определяется через $D_{n,m} \leq t_\alpha(n,m)$, где $D_{n,m} = \sup\limits_{x\in\mathbb{R}} \left|\hat{F_n}(x) - \hat{F_m}(x)\right|$ и $t_\alpha(n,m) = \sqrt{\frac{1}{n}+\frac{1}{m}}t_\alpha$.

Значения $D_{n,m}$ мы посчитали во втором домашнем задании, их использовать и будем. α оставим такойже, 0.05, тогда $t_\alpha = 1.36$

In [64]:
```
t_alpha_diff_xi = np.array([[np.sqrt(1/n[i]+1/n[j])*1.36 for i in range(len(n))]
                            for j in range(len(n))])
t_alpha_diff_xi_demo =  np.array([['%.4f'%i for i in j] for j in t_alpha_diff_xi])
diffsxi_good = np.array([[[diffsxi[k][j][i]<t_alpha_diff_xi[i,j] for i in range(le
n(n))]
                          for j in range(len(n))] for k in range(5)])
diffsxi_good_demo = np.array([[[('Подходят' if i else 'Не подходят') if i!=None el
se '-' for i in j]
                               for j in k] for k in diffsxi_good])
```

In [65]:
```
# t_alpha

ax = plt.gca()
plt.gcf().set_size_inches(7, 3.5, forward=True)
ax.table(cellText = t_alpha_diff_xi_demo, rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax.set_axis_off()
ax.set_title(r'$t_{\alpha}(n,m)$');
```

$t_\alpha(n,m)$

	5	10	100	200	400	600	800	1000
5	0.8601	0.7449	0.6232	0.6158	0.6120	0.6107	0.6101	0.6097
10	0.7449	0.6082	0.4511	0.4407	0.4354	0.4336	0.4327	0.4322
100	0.6232	0.4511	0.1923	0.1666	0.1521	0.1469	0.1442	0.1426
200	0.6158	0.4407	0.1666	0.1360	0.1178	0.1110	0.1075	0.1053
400	0.6120	0.4354	0.1521	0.1178	0.0962	0.0878	0.0833	0.0805
600	0.6107	0.4336	0.1469	0.1110	0.0878	0.0785	0.0734	0.0702
800	0.6101	0.4327	0.1442	0.1075	0.0833	0.0734	0.0680	0.0645
1000	0.6097	0.4322	0.1426	0.1053	0.0805	0.0702	0.0645	0.0608

In [66]:
```python
# 0
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = diffsxi_demo[0], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');

ax[1].table(cellText = diffsxi_good_demo[0], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Критерий однородности при $t_{\\alpha}=1.36$');
```

$$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}|F_n(x)-F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.73	0.83	0.65	0.61	0.69	0.60	0.66
10	0.73	0.00	1.03	0.85	0.87	0.98	0.97	0.90
100	0.83	1.03	0.00	0.69	1.10	0.91	1.13	0.81
200	0.65	0.85	0.69	0.00	0.72	0.63	0.62	0.61
400	0.61	0.87	1.10	0.72	0.00	0.79	0.45	0.63
600	0.69	0.98	0.91	0.63	0.79	0.00	0.83	0.62
800	0.60	0.97	1.13	0.62	0.45	0.83	0.00	0.74
1000	0.66	0.90	0.81	0.61	0.63	0.62	0.74	0.00

Критерий однородности при $t_{\alpha}=1.36$

	5	10	100	200	400	600	800	1000
5	Подходят	Подходят	Не подходят	Не подходят	Подходят	Не подходят	Подходят	Не подходят
10	Подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
100	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
200	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят
400	Подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят
600	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят
800	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят
1000	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят

In [67]:
```
# 1
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = diffsxi_demo[1], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');

ax[1].table(cellText = diffsxi_good_demo[1], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Критерий однородности при $t_{\\alpha}=1.36$');
```

$$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}|F_n(x)-F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.37	0.52	0.62	0.54	0.53	0.56	0.61
10	0.37	0.00	0.57	0.56	0.51	0.43	0.59	0.59
100	0.52	0.57	0.00	0.86	0.80	0.69	0.53	0.41
200	0.62	0.56	0.86	0.00	0.75	0.53	0.93	1.08
400	0.54	0.51	0.80	0.75	0.00	0.90	0.57	0.98
600	0.53	0.43	0.69	0.53	0.90	0.00	1.04	1.36
800	0.56	0.59	0.53	0.93	0.57	1.04	0.00	0.65
1000	0.61	0.59	0.41	1.08	0.98	1.36	0.65	0.00

Критерий однородности при $t_{\alpha}=1.36$

	5	10	100	200	400	600	800	1000
5	Подходят	Подходят	Подходят	Не подходят	Подходят	Подходят	Подходят	Подходят
10	Подходят	Подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят
100	Подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
200	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят
400	Подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят
600	Подходят	Подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят
800	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят
1000	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят

In [68]:

```python
# 2
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = diffsxi_demo[2], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');

ax[1].table(cellText = diffsxi_good_demo[2], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Критерий однородности при $t_{\\alpha}=1.36$');
```

$$D_{m,n} = \sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}|F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	1.28	1.29	1.37	1.42	1.43	1.36	1.41
10	1.28	0.00	0.78	0.54	0.56	0.51	0.58	0.54
100	1.29	0.78	0.00	0.73	0.72	0.89	0.71	0.83
200	1.37	0.54	0.73	0.00	0.69	0.53	0.68	0.74
400	1.42	0.56	0.72	0.69	0.00	0.36	0.45	0.35
600	1.43	0.51	0.89	0.53	0.36	0.00	0.58	0.44
800	1.36	0.58	0.71	0.68	0.45	0.58	0.00	0.55
1000	1.41	0.54	0.83	0.74	0.35	0.44	0.55	0.00

Критерий однородности при $t_\alpha = 1.36$

	5	10	100	200	400	600	800	1000
5	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
10	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
100	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
200	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят
400	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят
600	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят
800	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят
1000	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят

In [69]:

```
# 3
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = diffsxi_demo[3], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');

ax[1].table(cellText = diffsxi_good_demo[3], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Критерий однородности при $t_{\\alpha}=1.36$');
```

$$D_{m,n} = \sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}|F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.73	1.09	1.20	1.04	1.09	1.13	1.03
10	0.73	0.00	0.57	0.42	0.52	0.46	0.51	0.53
100	1.09	0.57	0.00	0.90	0.69	0.65	0.62	0.55
200	1.20	0.42	0.90	0.00	1.24	0.65	0.55	1.06
400	1.04	0.52	0.69	1.24	0.00	1.23	1.33	0.67
600	1.09	0.46	0.65	0.65	1.23	0.00	0.44	1.08
800	1.13	0.51	0.62	0.55	1.33	0.44	0.00	1.13
1000	1.03	0.53	0.55	1.06	0.67	1.08	1.13	0.00

Критерий однородности при $t_{\alpha} = 1.36$

	5	10	100	200	400	600	800	1000
5	Подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
10	Подходят	Подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят
100	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
200	Не подходят	Подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят
400	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят
600	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят
800	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят
1000	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят

In [70]:
```python
# 4
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = diffsxi_demo[4], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');

ax[1].table(cellText = diffsxi_good_demo[4], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Критерий однородности при $t_{\\alpha}=1.36$');
```

$$D_{m,n} = \sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}|F_n(x) - F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.91	1.22	0.97	1.03	1.01	0.99	1.02
10	0.91	0.00	1.24	1.05	0.98	0.96	0.98	0.96
100	1.22	1.24	0.00	1.39	1.25	1.34	1.40	1.36
200	0.97	1.05	1.39	0.00	0.87	0.80	0.90	1.03
400	1.03	0.98	1.25	0.87	0.00	0.63	0.45	0.44
600	1.01	0.96	1.34	0.80	0.63	0.00	0.48	0.79
800	0.99	0.98	1.40	0.90	0.45	0.48	0.00	0.94
1000	1.02	0.96	1.36	1.03	0.44	0.79	0.94	0.00

Критерий однородности при $t_{\alpha} = 1.36$

	5	10	100	200	400	600	800	1000
5	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
10	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
100	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
200	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят
400	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят
600	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят
800	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят
1000	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят

Задание 3

In [71]:
```python
#asd
```

Абсолютно непрерывное

Задание 1

Критерий согласия Колмагорова (Смирнова)

Здесь распределение уже непрерывное, так что никаких $U(x) = F(x-0) + \gamma\big(F(x) - F(x-0)\big)$ строить и рассматривать не надо.

$D_n = max\{D_n^-, D_n^+\}$, где $D_n^- = \max\limits_{1\le k\le n}\left|F(x_{(k)}) - \frac{k-1}{n}\right|$ и $D_n^+ = \max\limits_{1\le k\le n}\left|\frac{k}{n} - F(x_{(k)})\right|$ в коде определяется также как и в дискретном случае.

α оставим такой же, всё те же 0.05, соответственно $t_\alpha = 1.36$.

In [72]:
```python
def eta_Dn(sample, theta=0.45):
    n=len(sample)
    res=-1
    for k in range(n):
        t=np.abs(k/n-eta_distr_real(sample[k], theta))#plus
        if t>res: res=t
        t=np.abs(eta_distr_real(sample[k], theta)-(k-1)/n)#minus
        if t>res: res=t
    return res

def eta_Sn(sample, theta=0.45):
    return (6*len(sample)*eta_Dn(sample, theta)+1)/(6*np.sqrt(len(sample)))

kolmag_eta = np.array([[eta_Dn(sample_eta[k][j])*np.sqrt(n[k]) if n[k]>20 else eta_Sn(sample_eta[k][j])
                       for j in range(5)] for k in range(len(n))])
kolmag_eta_demo = np.array([['%.4f' % i for i in j] for j in kolmag_eta])

kolmag_eta_good = np.array([[i<1.36 for i in j] for j in kolmag_eta])
kolmag_eta_good_demo = np.array([['Подходит' if i else 'Не подходит' for i in j]
                                 for j in kolmag_eta_good])
```

In [73]:
```
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = kolmag_eta_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('$\\sqrt{n}D_n$ критерия согласия Колмагорова')

ax[1].table(cellText = kolmag_eta_good_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Подходит ли выборка по критерию');
```

$\sqrt{n}D_n$ критерия согласия Колмагорова

	1	2	3	4	5
5	1.7844	1.1362	1.5659	0.7592	0.7516
10	1.1948	1.3203	0.8599	0.5186	0.8913
100	0.4434	0.8335	0.9687	0.5884	0.4861
200	0.6892	0.5729	0.9602	0.6906	0.4740
400	0.9441	0.9640	0.9322	1.3634	1.1062
600	0.5297	1.1492	0.6181	1.1916	0.7430
800	0.4847	0.5567	1.1671	0.9698	1.1525
1000	1.7160	0.7291	1.0632	0.6162	0.5809

Подходит ли выборка по критерию

	1	2	3	4	5
5	Не подходит	Подходит	Не подходит	Подходит	Подходит
10	Подходит	Подходит	Подходит	Подходит	Подходит
100	Подходит	Подходит	Подходит	Подходит	Подходит
200	Подходит	Подходит	Подходит	Подходит	Подходит
400	Подходит	Подходит	Подходит	Не подходит	Подходит
600	Подходит	Подходит	Подходит	Подходит	Подходит
800	Подходит	Подходит	Подходит	Подходит	Подходит
1000	Не подходит	Подходит	Подходит	Подходит	Подходит

Критерий хи-квадрат

Выберем N=5 и поделим отрезок $[0, 1]$ на соответственно 5 равных отрезков. Их вероятности буду вычислять на месте, что упростит реализацию при переходе к сложной гипотезе.

α сохраним за 0.05, N-1=4, и $t_\alpha = 9.5$.

In [74]:
```
def eta_nu1_vec(sample):
    res=np.zeros(5)
    for x in sample:
        for i in range(5):
            if x<0.2*(i+1):
                res[i]+=1
                break
    return res

eta_hi_p_i = lambda a, b, theta=0.45: eta_distr_real(b, theta)-eta_distr_real(a, t
heta)
def eta_hi1_sqr(sample, theta=0.45):
    n=len(sample)
    nu=eta_nu1_vec(sample)
    res=0
    p_i=0
    for i in range(5):
        p_i = eta_hi_p_i(0.2*i, 0.2*(i+1), theta)
        res += (nu[i]-n*p_i)**2/(n*p_i)
    return res


hi1sqr_eta = np.array([[eta_hi1_sqr(sample_eta[k][j]) if n[k]>20 else None
                        for j in range(5)] for k in range(len(n))])
hi1sqr_eta_demo = np.array([['%.4f'%i if i!=None else '-' for i in j] for j in hi1
sqr_eta])

hi1sqr_eta_good = np.array([[i<9.5 if i!=None else None for i in j] for j in hi1sq
r_eta])
hi1sqr_eta_good_demo = np.array([[('Подходит' if i else 'Не подходит') if i!=None
else '-' for i in j]
                                  for j in hi1sqr_eta_good])
```

In [75]:
```
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = hi1sqr_eta_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('$X_N^2$ критерия согласия хи-квадрат при первом разбиении')

ax[1].table(cellText = hi1sqr_eta_good_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Подходит ли выборка по критерию');
```

X_N^2 критерия согласия хи-квадрат при первом разбиении

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	0.6656	0.5406	7.3784	1.1106	1.3916
200	4.1713	4.4718	5.9309	3.5151	0.6727
400	2.3242	2.4730	3.5913	3.0242	1.4152
600	0.7084	4.6453	0.4911	8.3945	2.5042
800	1.3361	1.5784	4.4692	2.2923	6.7460
1000	7.9504	1.9829	4.6312	1.7488	1.1653

Подходит ли выборка по критерию

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	Подходит	Подходит	Подходит	Подходит	Подходит
200	Подходит	Подходит	Подходит	Подходит	Подходит
400	Подходит	Подходит	Подходит	Подходит	Подходит
600	Подходит	Подходит	Подходит	Подходит	Подходит
800	Подходит	Подходит	Подходит	Подходит	Подходит
1000	Подходит	Подходит	Подходит	Подходит	Подходит

Теперь разбиение в N=10 равных между собой по длине отрезков. $t_\alpha = 16.9$.

In [76]:
```
def eta_nu2_vec(sample):
    res=np.zeros(10)
    for x in sample:
        for i in range(10):
            if x<0.1*(i+1):
                res[i]+=1
                break
    return res

eta_hi_p_i = lambda a, b, theta=0.45: eta_distr_real(b, theta)-eta_distr_real(a, t
heta)
def eta_hi2_sqr(sample, theta=0.45):
    n=len(sample)
    nu=eta_nu2_vec(sample)
    res=0
    p_i=0
    for i in range(10):
        p_i = eta_hi_p_i(0.1*i, 0.1*(i+1), theta)
        res += (nu[i]-n*p_i)**2/(n*p_i)
    return res


hi2sqr_eta = np.array([[eta_hi2_sqr(sample_eta[k][j]) if n[k]>20 else None
                       for j in range(5)] for k in range(len(n))])
hi2sqr_eta_demo = np.array([['%.4f'%i if i!=None else '-' for i in j] for j in hi2
sqr_eta])

hi2sqr_eta_good = np.array([[i<16.9 if i!=None else None for i in j] for j in hi2s
qr_eta])
hi2sqr_eta_good_demo = np.array([[('Подходит' if i else 'Не подходит') if i!=None
else '-' for i in j]
                                  for j in hi2sqr_eta_good])
```

In [77]:
```
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = hi2sqr_eta_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('$X_N^2$ критерия согласия хи-квадрат при втором разбиении')

ax[1].table(cellText = hi2sqr_eta_good_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Подходит ли выборка по критерию');
```

X_N^2 критерия согласия хи-квадрат при втором разбиении

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	6.4150	5.1367	9.1054	3.9035	2.6473
200	6.0584	8.5370	22.6417	12.4356	5.0815
400	12.0132	11.9709	10.2875	14.5082	4.1063
600	2.4270	6.6955	6.2871	14.5735	3.7045
800	7.1942	2.5776	6.6175	11.0898	13.5342
1000	9.3320	7.0186	6.2470	3.8856	5.4462

Подходит ли выборка по критерию

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	Подходит	Подходит	Подходит	Подходит	Подходит
200	Подходит	Подходит	Не подходит	Подходит	Подходит
400	Подходит	Подходит	Подходит	Подходит	Подходит
600	Подходит	Подходит	Подходит	Подходит	Подходит
800	Подходит	Подходит	Подходит	Подходит	Подходит
1000	Подходит	Подходит	Подходит	Подходит	Подходит

Критерий согласия Колмагорова (Смирнова) для сложной гипотезы

$t_\alpha = 1.36$

In [78]:
```
kolmag_eta_tough = np.array([[eta_Dn(sample_eta[k][j],estimthetaeta_mmp[k][j])*np.
sqrt(n[k])
                              if n[k]>20 else eta_Sn(sample_eta[k][j])
                              for j in range(5)] for k in range(len(n))])
kolmag_eta_tough_demo = np.array([['%.4f' % i for i in j] for j in kolmag_eta_toug
h])

kolmag_eta_tough_good = np.array([[i<1.36 for i in j] for j in kolmag_eta_tough])
kolmag_eta_tough_good_demo = np.array([['Подходит' if i else 'Не подходит' for i i
n j]
                                  for j in kolmag_eta_tough_good])
```

In [79]:
```
fig, ax = plt.subplots(3,1, figsize=(7,10.5))
ax[0].table(cellText = estimthetaeta_mmp_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('Оценка $\\theta$ методом максимального правдоподобия')

ax[1].table(cellText = kolmag_eta_tough_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('$\\sqrt{n}D_n$ критерия согласия Колмагорова')

ax[2].table(cellText = kolmag_eta_tough_good_demo, rowLabels=n, colLabels=[1,2,3,4
,5],
            loc='center').scale(1, 1.5)
ax[2].set_axis_off()
ax[2].set_title('Подходит ли выборка по критерию');
```

Оценка θ методом максимального правдоподобия

	1	2	3	4	5
5	0.5261	0.5771	0.5657	0.3375	0.2276
10	0.5610	0.5395	0.4634	0.3611	0.4487
100	0.4340	0.4670	0.4693	0.4634	0.4449
200	0.4624	0.4508	0.3977	0.4080	0.4524
400	0.4893	0.4420	0.4081	0.5183	0.4436
600	0.4488	0.4177	0.4476	0.4612	0.4692
800	0.4567	0.4527	0.4744	0.4655	0.4674
1000	0.4234	0.4486	0.4745	0.4349	0.4552

$\sqrt{n}D_n$ критерия согласия Колмагорова

	1	2	3	4	5
5	1.7844	1.1362	1.5659	0.7592	0.7516
10	1.1948	1.3203	0.8599	0.5186	0.8913
100	0.5071	0.7963	0.9166	0.4614	0.5033
200	0.6554	0.5747	0.6301	1.0528	0.4563
400	0.7098	0.8889	0.9059	0.5849	1.1891
600	0.5154	0.5426	0.6359	1.0754	0.5773
800	0.6008	0.6000	0.8899	0.8299	0.7800
1000	1.1195	0.7421	0.6163	0.6780	0.5225

Подходит ли выборка по критерию

	1	2	3	4	5
5	Не подходит	Подходит	Не подходит	Подходит	Подходит
10	Подходит	Подходит	Подходит	Подходит	Подходит
100	Подходит	Подходит	Подходит	Подходит	Подходит
200	Подходит	Подходит	Подходит	Подходит	Подходит
400	Подходит	Подходит	Подходит	Подходит	Подходит
600	Подходит	Подходит	Подходит	Подходит	Подходит
800	Подходит	Подходит	Подходит	Подходит	Подходит
1000	Подходит	Подходит	Подходит	Подходит	Подходит

Критерий согласия хи-квадрат для сложной гипотезы

$\hat{\theta}=\underset{\theta}{\operatorname{argmax}}\prod_{i=1}^{N}\left(p_i(\theta)\right)^{\nu_i}$: $\prod_{i=1}^{N}\left(p_i(\theta)\right)^{\nu_i}=\prod_{i=1}^{N}\left(F(b_i)-F(a_i)\right)^{\nu_i}$, где a_i и b_i соответственно минимум и максимум рассматриваемого отрезка. Тогда, среди этих пар a_i, b_i есть такой a_k, b_k, $k=\overline{1,N}$ что $\theta\in[a_k,b_k]$. Тогда $\prod_{i=1}^{N}\left(F(b_i)-F(a_i)\right)^{\nu_i}=\prod_{i=1}^{k-1}\left(F(b_i)-F(a_i)\right)^{\nu_i}\cdot\left(F(b_k)-F(a_k)\right)^{\nu_i}\cdot\prod_{i=k+1}^{N}\left(F(b_i)-F(a_i)\right)^{\nu_i}=$ $\prod_{i=1}^{k-1}\left(\frac{b_i^2-a_i^2}{\theta}\right)^{\nu_i}\cdot\left(F(b_k)-F(\theta-0)+F(\theta+0)-F(a_k)\right)^{\nu_i}\cdot\prod_{i=k+1}^{N}\left(\frac{2b_i-b_i^2-\theta-2a_i+a_i^2+\theta}{1-\theta}\right)^{\nu_i}=\theta^{-\sum_{i=0}^{k-1}\nu_i}$ $\cdot(1-\theta)^{-\sum_{i=k+1}^{N}\nu_i}\cdot\left(\frac{2b_k-b_k^2-\theta-2\theta+\theta^2+\theta}{1-\theta}+\frac{\theta^2-a_k^2}{\theta}\right)^{\nu_k}\cdot\prod_{i=1}^{k-1}\left(b_i^2-a_i^2\right)^{\nu_i}\cdot\prod_{i=k+1}^{N}\left(2b_i-b_i^2-2a_i+a_i^2\right)^{\nu_i}$

.

$\hat{\theta}=\underset{\theta}{\operatorname{argmax}}\prod_{i=1}^{N}\left(p_i(\theta)\right)^{\nu_i}$ не зависит от правых двух произведений (потому что это константы), потому далее будем рассматривать $\theta^{-\sum_{i=0}^{k-1}\nu_i}\cdot(1-\theta)^{-\sum_{i=k+1}^{N}\nu_i}\cdot\left(\frac{2b_k-b_k^2-2\theta+\theta^2}{1-\theta}+\frac{\theta^2-a_k^2}{\theta}\right)^{\nu_k}$:

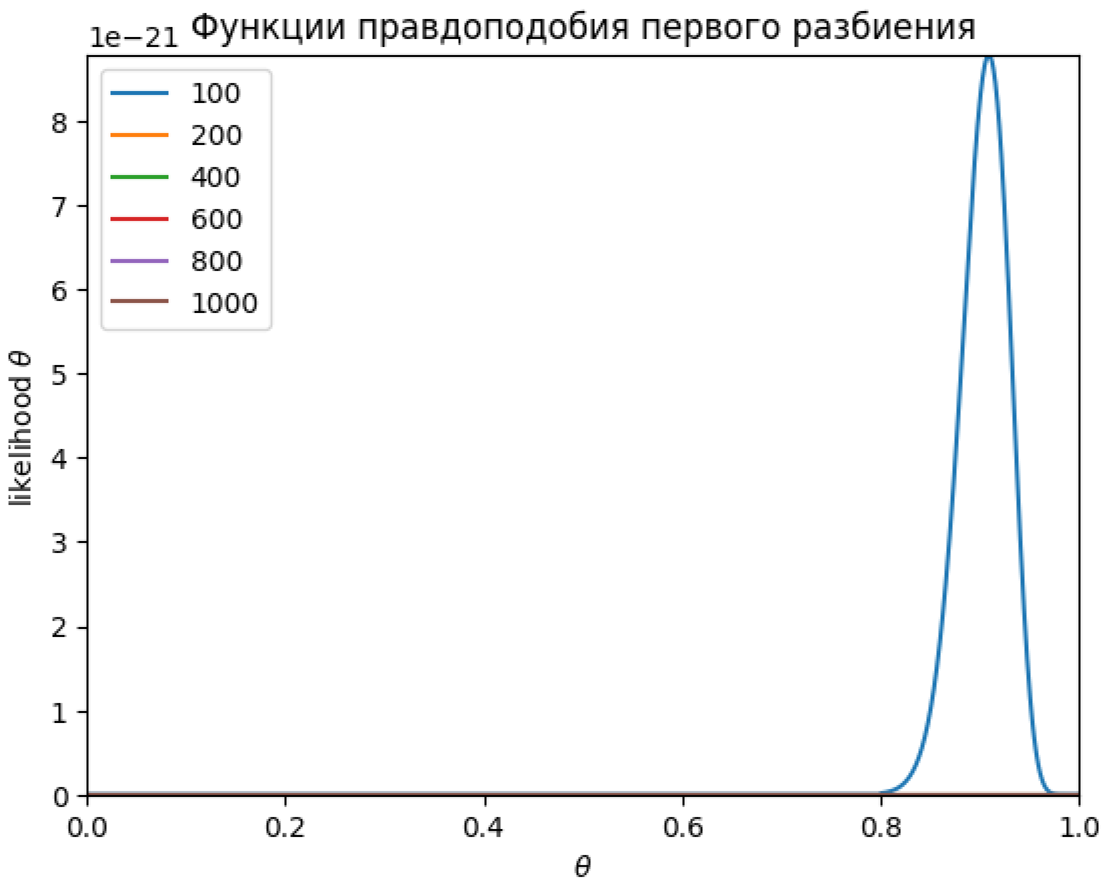
$\theta^{\sum_{i=0}^{k-1}\nu_i}\cdot(1-\theta)^{\sum_{i=k+1}^{N}\nu_i}\cdot\left(\frac{2b_k-b_k^2-2\theta+\theta^2}{1-\theta}+\frac{\theta^2-a_k^2}{\theta}\right)^{\nu_k}=\theta^{-\sum_{i=0}^{k-1}\nu_i}\cdot(1-\theta)^{-\sum_{i=k+1}^{N}\nu_i}\cdot\left(\frac{(2b_k-b_k^2-2\theta+\theta^2)\theta+(\theta^2-a_k^2)(1-\theta)}{(1-\theta)\theta}\right)^{\nu_k}$

$=\theta^{-\sum_{i=0}^{k}\nu_i}\cdot(1-\theta)^{-\sum_{i=k}^{N}\nu_i}\cdot\left(2b_k\theta-b_k^2\theta-a_k^2+a_k^2\theta-\theta^2\right)^{\nu_k}.$

На вид полученная функция ничего толкового не дает по поиску argmax математическими методами (что оказалось правдой на черновике), потому посмотрю на графики, как я это делал в третьем домашнем задании.

In [80]:
```
def eta_hi1_likelihood_func_range(sample, N=1000):
    res=np.ones(N, dtype=np.float64)
    thetas=np.linspace(0,1,N)
    nu=np.concatenate([eta_nu1_vec(sample),[0]])# чтобы не было ошибки индексов
    k=0 #в каком отрезке мы находимся
    for i in range(N):
        while not(0.2*k<=thetas[i]<0.2*(k+1)): k+=1
        res[i] *= thetas[i]**(np.sum(nu[:k+1]))
        res[i] *= (1-thetas[i])**(np.sum(nu[k:]))
        res[i] *= (0.32*k*thetas[i]+0.36*thetas[i]-0.04*k*k-thetas[i]*thetas[i])**
nu[k]
    return res

likelihood_eta_hi1_range=np.array([[eta_hi1_likelihood_func_range(sample_eta[j][i
])
                                    for i in range(5)] for j in range(len(n))])
likelihood_eta_hi1_range[:,:,-1]=0
```

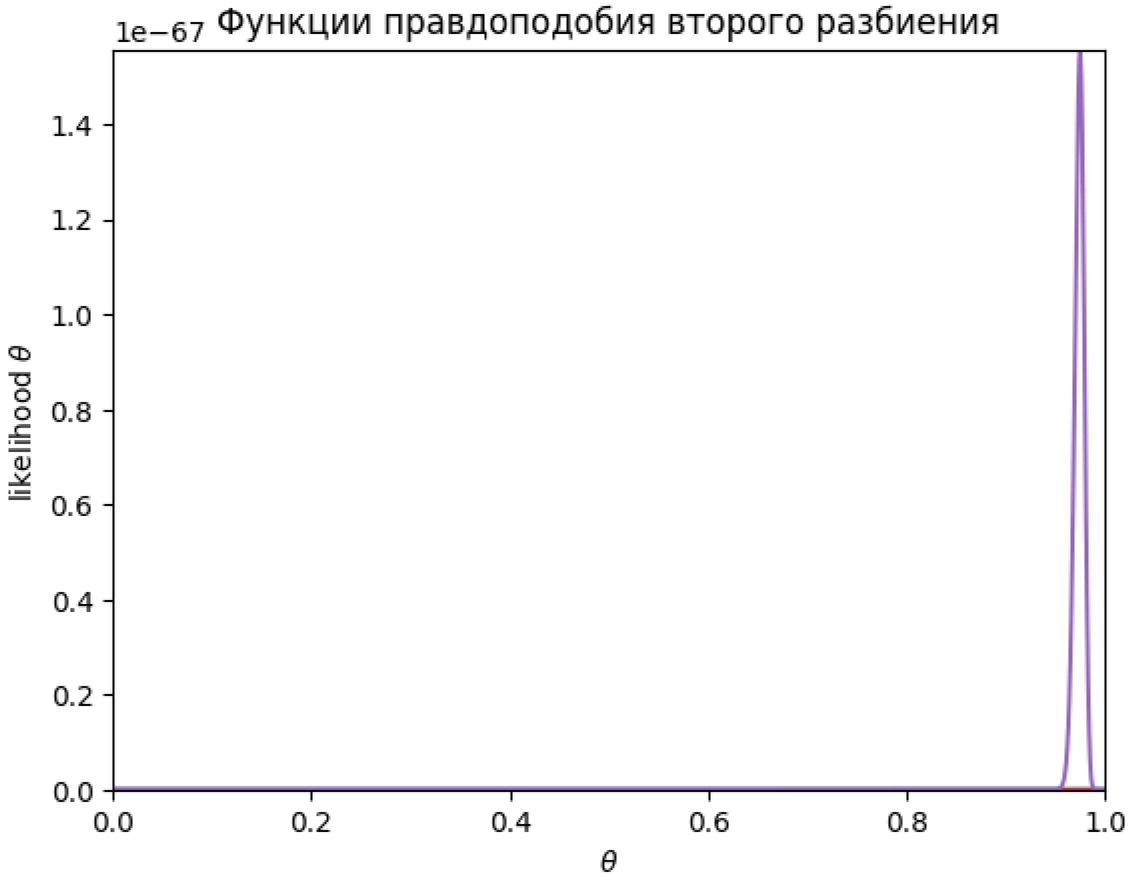
In [81]:
```
for i in range(2,8):
    plt.plot(np.linspace(0,1,1000), likelihood_eta_hi1_range[i][0])
#plt.plot(np.linspace(0,1,1000), likelihood_eta_hi1_range[4][0])
plt.gca().set(xmargin = 0, ymargin = 0, xlabel = '$\\theta$',
              ylabel = 'likelihood $\\theta$', title = 'Функции правдоподобия перв
ого разбиения');
plt.legend(n[2:], loc='upper left');
```



Оно выглядит ну очень странно, учитывая что в коде ошибок вроде как нет (на момент написания они были, я их поправил, но стало еще страннее). Но может быть это связано с разбиением, потому я посмотрю как будет выглядеть тоже самое при разбиении на 10 отрезков

In [82]:
```
def eta_hi2_likelihood_func_range(sample, N=1000):
    res=np.ones(N, dtype=np.float64)
    thetas=np.linspace(0,1,N)
    nu=np.concatenate([eta_nu2_vec(sample),[0]])# чтобы не было ошибки индексов
    k=0 #в каком отрезке мы находимся
    for i in range(N):
        while not(0.1*k<=thetas[i]<0.1*(k+1)): k+=1
        res[i] *= thetas[i]**(np.sum(nu[:k+1]))
        res[i] *= (1-thetas[i])**(np.sum(nu[k:]))
        res[i] *= (0.18*k*thetas[i]+0.19*thetas[i]-0.01*k*k-thetas[i]*thetas[i])**
nu[k]
    return res

likelihood_eta_hi2_range=np.array([[eta_hi2_likelihood_func_range(sample_eta[j][i
])
                                    for i in range(5)] for j in range(len(n))])
likelihood_eta_hi2_range[:,:,-1]=0
```

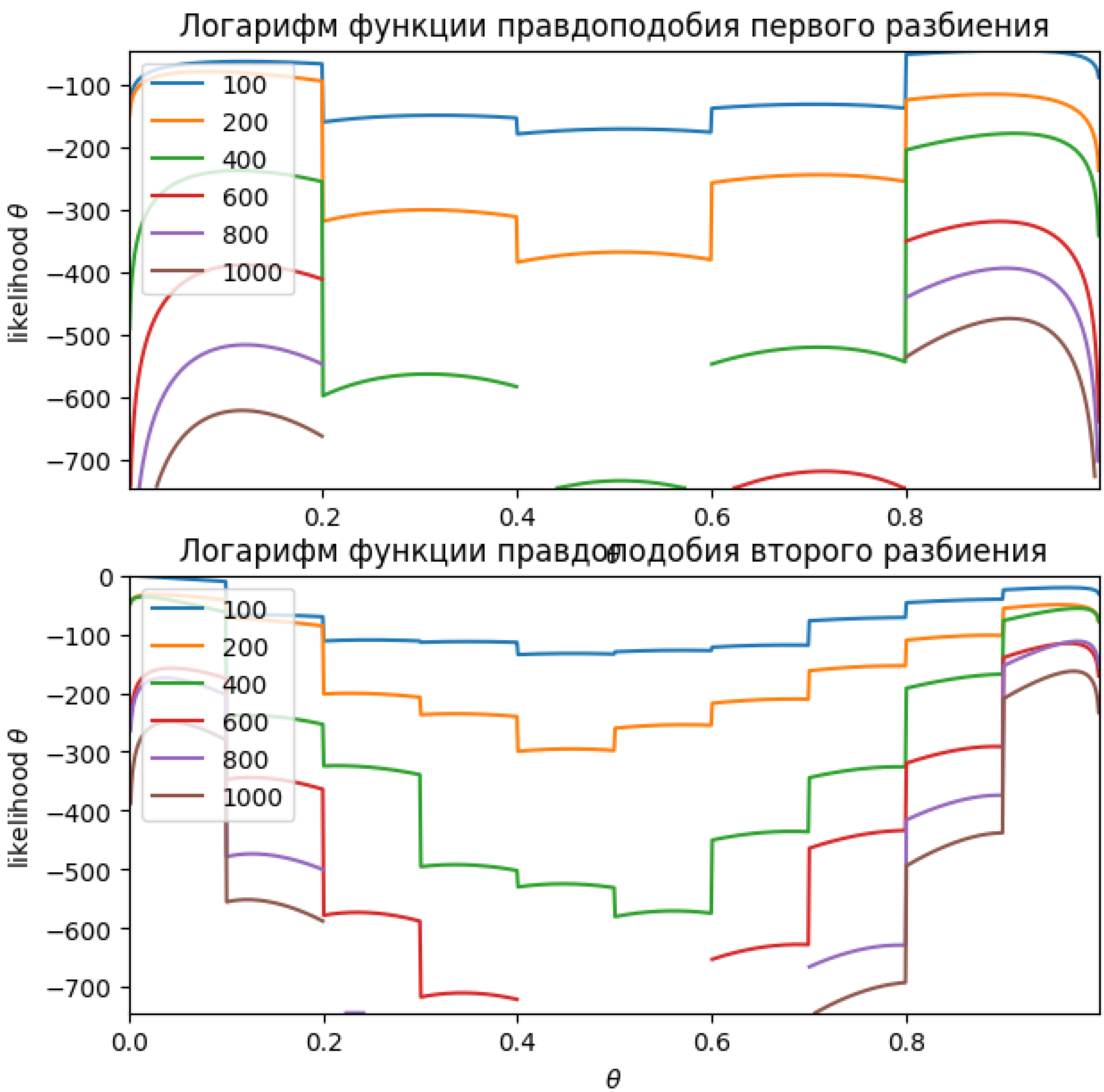
In [83]:
```
for i in range(5):
    plt.plot(np.linspace(0,1,1000), likelihood_eta_hi2_range[7][i])
#plt.plot(np.linspace(0,1,1000), likelihood_eta_hi2_range[3][2])
plt.gca().set(xmargin = 0, ymargin = 0, xlabel = '$\\theta$',
              ylabel = 'likelihood $\\theta$', title = 'Функции правдоподобия втор
ого разбиения');
```



Можно я просто возьму оценку ОМП из дз3, ладно? Для иллюстративности что происходит, вот пара графиков логарифмов правдоподобия.

In [84]:
```
fig, ax = plt.subplots(2,1, figsize=(7,7))
for i in range(2,8):
    ax[0].plot(np.linspace(0,1,1000), np.log(likelihood_eta_hi1_range[i][0]))
    ax[1].plot(np.linspace(0,1,1000), np.log(likelihood_eta_hi2_range[i][0]))
ax[0].set(xmargin = 0, ymargin = 0, xlabel = '$\\theta$',
              ylabel = 'likelihood $\\theta$', title = 'Логарифм функции правдопод
обия первого разбиения');
ax[0].legend(n[2:], loc='upper left');
ax[1].set(xmargin = 0, ymargin = 0, xlabel = '$\\theta$',
              ylabel = 'likelihood $\\theta$', title = 'Логарифм функции правдопод
обия второго разбиения');
ax[1].legend(n[2:], loc='upper left');
```

```
C:\Users\64p4h\AppData\Local\Temp\ipykernel_2388\4063359791.py:3: RuntimeWarnin
g: divide by zero encountered in log
  ax[0].plot(np.linspace(0,1,1000), np.log(likelihood_eta_hi1_range[i][0]))
C:\Users\64p4h\AppData\Local\Temp\ipykernel_2388\4063359791.py:4: RuntimeWarnin
g: divide by zero encountered in log
  ax[1].plot(np.linspace(0,1,1000), np.log(likelihood_eta_hi2_range[i][0]))
```



Берем квантиль $\alpha = 0.05$ при степенях свободы N-2=3, $t_\alpha = 7.8$

In [85]:
```
hi1sqr_eta_tough = np.array([[eta_hi1_sqr(sample_eta[k][j],estimthetaeta_mmp[k][j
])
                              if n[k]>20 else None for j in range(5)]
                             for k in range(len(n))])
hi1sqr_eta_tough_demo = np.array([['%.4f'%i if i!=None else '-' for i in j] for j
 in hi1sqr_eta_tough])

hi1sqr_eta_tough_good = np.array([[i<7.8 if i!=None else None for i in j] for j in
 hi1sqr_eta_tough])
hi1sqr_eta_tough_good_demo = np.array([[('Подходит' if i else 'Не подходит') if i!
=None else '-' for i in j]
                                   for j in hi1sqr_eta_tough_good])
```

In [86]:

```
fig, ax = plt.subplots(3,1, figsize=(7,10.5))
ax[0].table(cellText = estimthetaeta_mmp_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('Оценка $\\theta$ методом максимального правдоподобия')

ax[1].table(cellText = hi1sqr_eta_tough_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('$X_N^2$ критерия согласия хи-квадрат при первом разбиении')

ax[2].table(cellText = hi1sqr_eta_tough_good_demo, rowLabels=n, colLabels=[1,2,3,4
,5],
            loc='center').scale(1, 1.5)
ax[2].set_axis_off()
ax[2].set_title('Подходит ли выборка по критерию');
```

Оценка θ методом максимального правдоподобия

	1	2	3	4	5
5	0.5261	0.5771	0.5657	0.3375	0.2276
10	0.5610	0.5395	0.4634	0.3611	0.4487
100	0.4340	0.4670	0.4693	0.4634	0.4449
200	0.4624	0.4508	0.3977	0.4080	0.4524
400	0.4893	0.4420	0.4081	0.5183	0.4436
600	0.4488	0.4177	0.4476	0.4612	0.4692
800	0.4567	0.4527	0.4744	0.4655	0.4674
1000	0.4234	0.4486	0.4745	0.4349	0.4552

X_N^2 критерия согласия хи-квадрат при первом разбиении

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	0.6398	0.4414	6.9796	1.0253	1.3744
200	4.1424	4.4691	4.0163	3.0966	0.6914
400	2.5298	2.2597	1.2017	1.5205	1.7479
600	0.6702	0.7838	0.4721	7.2850	1.7225
800	1.9685	1.7345	2.6322	1.3741	6.2539
1000	5.8191	2.0458	1.2448	2.2978	1.2797

Подходит ли выборка по критерию

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	Подходит	Подходит	Подходит	Подходит	Подходит
200	Подходит	Подходит	Подходит	Подходит	Подходит
400	Подходит	Подходит	Подходит	Подходит	Подходит
600	Подходит	Подходит	Подходит	Подходит	Подходит
800	Подходит	Подходит	Подходит	Подходит	Подходит
1000	Подходит	Подходит	Подходит	Подходит	Подходит

Разбиение N=10. $t_\alpha = 15.5$

In [87]:

```
hi2sqr_eta_tough = np.array([[eta_hi2_sqr(sample_eta[k][j],estimthetaeta_mmp[k][j
])
                              if n[k]>20 else None for j in range(5)]
                             for k in range(len(n))])
hi2sqr_eta_tough_demo = np.array([['%.4f'%i if i!=None else '-' for i in j] for j
 in hi2sqr_eta_tough])

hi2sqr_eta_tough_good = np.array([[i<15.5 if i!=None else None for i in j] for j i
n hi2sqr_eta_tough])
hi2sqr_eta_tough_good_demo = np.array([[('Подходит' if i else 'Не подходит') if i!
=None else '-' for i in j]
                                   for j in hi2sqr_eta_tough_good])
```

In [88]:
```
fig, ax = plt.subplots(3,1, figsize=(7,10.5))
ax[0].table(cellText = estimthetaeta_mmp_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title('Оценка $\\theta$ методом максимального правдоподобия')

ax[1].table(cellText = hi2sqr_eta_tough_demo, rowLabels=n, colLabels=[1,2,3,4,5],
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('$X_N^2$ критерия согласия хи-квадрат при первом разбиении')

ax[2].table(cellText = hi2sqr_eta_tough_good_demo, rowLabels=n, colLabels=[1,2,3,4
,5],
            loc='center').scale(1, 1.5)
ax[2].set_axis_off()
ax[2].set_title('Подходит ли выборка по критерию');
```

Оценка θ методом максимального правдоподобия

	1	2	3	4	5
5	0.5261	0.5771	0.5657	0.3375	0.2276
10	0.5610	0.5395	0.4634	0.3611	0.4487
100	0.4340	0.4670	0.4693	0.4634	0.4449
200	0.4624	0.4508	0.3977	0.4080	0.4524
400	0.4893	0.4420	0.4081	0.5183	0.4436
600	0.4488	0.4177	0.4476	0.4612	0.4692
800	0.4567	0.4527	0.4744	0.4655	0.4674
1000	0.4234	0.4486	0.4745	0.4349	0.4552

X_N^2 критерия согласия хи-квадрат при первом разбиении

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	6.2919	4.9776	8.7664	3.6294	2.6195
200	6.1965	8.5378	19.4431	11.3975	5.1470
400	10.3911	11.5197	7.8508	9.8030	4.4684
600	2.3829	2.8544	6.2515	13.8449	3.3304
800	7.2924	2.7067	5.6396	11.7014	12.7212
1000	6.9829	7.0347	4.3104	4.1642	5.4001

Подходит ли выборка по критерию

	1	2	3	4	5
5	-	-	-	-	-
10	-	-	-	-	-
100	Подходит	Подходит	Подходит	Подходит	Подходит
200	Подходит	Подходит	Не подходит	Подходит	Подходит
400	Подходит	Подходит	Подходит	Подходит	Подходит
600	Подходит	Подходит	Подходит	Подходит	Подходит
800	Подходит	Подходит	Подходит	Подходит	Подходит
1000	Подходит	Подходит	Подходит	Подходит	Подходит

Задание 2

In [89]:
```
t_alpha_diff_eta = np.array([[np.sqrt(1/n[i]+1/n[j])*1.36 for i in range(len(n))]
                                for j in range(len(n))])
t_alpha_diff_eta_demo =  np.array([['%.4f'%i for i in j] for j in t_alpha_diff_et
a])
diffseta_good = np.array([[[diffseta[k][j][i]<t_alpha_diff_eta[i,j] for i in range
(len(n))]
                              for j in range(len(n))] for k in range(5)])
diffseta_good_demo = np.array([[[[('Подходят' if i else 'Не подходят') if i!=None e
lse '-' for i in j]
                                     for j in k] for k in diffseta_good])
```

In [90]:
```
# t_alpha
ax = plt.gca()
plt.gcf().set_size_inches(7, 3.5, forward=True)
ax.table(cellText = t_alpha_diff_eta_demo, rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax.set_axis_off()
ax.set_title(r'$t_{\alpha}(n,m)$');
```

$t_{\alpha}(n, m)$

	5	10	100	200	400	600	800	1000
5	0.8601	0.7449	0.6232	0.6158	0.6120	0.6107	0.6101	0.6097
10	0.7449	0.6082	0.4511	0.4407	0.4354	0.4336	0.4327	0.4322
100	0.6232	0.4511	0.1923	0.1666	0.1521	0.1469	0.1442	0.1426
200	0.6158	0.4407	0.1666	0.1360	0.1178	0.1110	0.1075	0.1053
400	0.6120	0.4354	0.1521	0.1178	0.0962	0.0878	0.0833	0.0805
600	0.6107	0.4336	0.1469	0.1110	0.0878	0.0785	0.0734	0.0702
800	0.6101	0.4327	0.1442	0.1075	0.0833	0.0734	0.0680	0.0645
1000	0.6097	0.4322	0.1426	0.1053	0.0805	0.0702	0.0645	0.0608

In [91]:

```python
# 0
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = diffseta_demo[0], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');

ax[1].table(cellText = diffseta_good_demo[0], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Критерий однородности при $t_{\\alpha}=1.36$');
```

$$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}|F_n(x)-F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.91	1.24	1.25	1.19	1.28	1.24	1.35
10	0.91	0.00	0.87	0.76	0.70	0.82	0.80	0.88
100	1.24	0.87	0.00	0.61	0.74	0.49	0.51	0.62
200	1.25	0.76	0.61	0.00	0.52	0.57	0.68	0.80
400	1.19	0.70	0.74	0.52	0.00	0.86	0.82	1.35
600	1.28	0.82	0.49	0.57	0.86	0.00	0.43	0.97
800	1.24	0.80	0.51	0.68	0.82	0.43	0.00	1.36
1000	1.35	0.88	0.62	0.80	1.35	0.97	1.36	0.00

Критерий однородности при $t_{\alpha}=1.36$

	5	10	100	200	400	600	800	1000
5	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
10	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
100	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
200	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят
400	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят
600	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят
800	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят
1000	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят

In [92]:

```python
# 1
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = diffseta_demo[1], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');

ax[1].table(cellText = diffseta_good_demo[1], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Критерий однородности при $t_{\\alpha}=1.36$');
```

$$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}|F_n(x)-F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.55	0.63	0.65	0.67	0.68	0.60	0.62
10	0.55	0.00	0.87	0.99	0.98	1.05	0.95	0.95
100	0.63	0.87	0.00	0.90	0.76	0.79	0.68	0.51
200	0.65	0.99	0.90	0.00	0.64	0.69	0.52	0.75
400	0.67	0.98	0.76	0.64	0.00	0.99	0.86	1.08
600	0.68	1.05	0.79	0.69	0.99	0.00	0.86	1.17
800	0.60	0.95	0.68	0.52	0.86	0.86	0.00	0.62
1000	0.62	0.95	0.51	0.75	1.08	1.17	0.62	0.00

Критерий однородности при $t_\alpha = 1.36$

	5	10	100	200	400	600	800	1000
5	Подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят
10	Подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
100	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
200	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят
400	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят
600	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят
800	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят
1000	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят

In [93]:

```
# 2
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = diffseta_demo[2], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');

ax[1].table(cellText = diffseta_good_demo[2], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Критерий однородности при $t_{\\alpha}=1.36$');
```

$$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}|F_n(x)-F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.73	0.98	1.15	1.13	1.05	1.00	0.98
10	0.73	0.00	0.66	0.65	0.62	0.51	0.53	0.49
100	0.98	0.66	0.00	0.73	0.72	0.69	0.81	0.80
200	1.15	0.65	0.73	0.00	0.66	0.92	1.01	1.23
400	1.13	0.62	0.72	0.66	0.00	0.77	1.06	1.27
600	1.05	0.51	0.69	0.92	0.77	0.00	0.66	0.79
800	1.00	0.53	0.81	1.01	1.06	0.66	0.00	0.52
1000	0.98	0.49	0.80	1.23	1.27	0.79	0.52	0.00

Критерий однородности при $t_{\alpha} = 1.36$

	5	10	100	200	400	600	800	1000
5	Подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
10	Подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
100	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
200	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят
400	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят
600	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят
800	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят
1000	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят

In [94]:

```python
# 3
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = diffseta_demo[3], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');

ax[1].table(cellText = diffseta_good_demo[3], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Критерий однородности при $t_{\\alpha}=1.36$');
```

$$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}|F_n(x)-F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.37	1.11	1.13	1.21	1.14	1.15	1.12
10	0.37	0.00	0.69	0.69	0.80	0.74	0.74	0.69
100	1.11	0.69	0.00	0.49	0.54	0.74	0.48	0.59
200	1.13	0.69	0.49	0.00	0.89	0.63	0.77	0.74
400	1.21	0.80	0.54	0.89	0.00	0.72	1.18	1.28
600	1.14	0.74	0.74	0.63	0.72	0.00	0.86	1.14
800	1.15	0.74	0.48	0.77	1.18	0.86	0.00	0.91
1000	1.12	0.69	0.59	0.74	1.28	1.14	0.91	0.00

Критерий однородности при $t_{\alpha}=1.36$

	5	10	100	200	400	600	800	1000
5	Подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
10	Подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
100	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
200	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят
400	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят
600	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят
800	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят
1000	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят

In [95]:
```python
# 4
fig, ax = plt.subplots(2,1, figsize=(7,7))
ax[0].table(cellText = diffseta_demo[4], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[0].set_axis_off()
ax[0].set_title(r'$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}$'+
                r'$|F_n(x)-F_m(x)|$');

ax[1].table(cellText = diffseta_good_demo[4], rowLabels=n, colLabels=n,
            loc='center').scale(1, 1.5)
ax[1].set_axis_off()
ax[1].set_title('Критерий однородности при $t_{\\alpha}=1.36$');
```

$$D_{m,n}=\sqrt{\frac{nm}{n+m}}\sup_{x\in\mathbb{R}}|F_n(x)-F_m(x)|$$

	5	10	100	200	400	600	800	1000
5	0.00	0.73	1.05	1.08	1.13	1.15	1.20	1.12
10	0.73	0.00	0.63	0.71	0.72	0.76	0.71	0.69
100	1.05	0.63	0.00	0.53	0.49	0.45	0.57	0.36
200	1.08	0.71	0.53	0.00	0.58	0.69	0.77	0.57
400	1.13	0.72	0.49	0.58	0.00	0.63	0.78	0.78
600	1.15	0.76	0.45	0.69	0.63	0.00	0.49	0.65
800	1.20	0.71	0.57	0.77	0.78	0.49	0.00	0.75
1000	1.12	0.69	0.36	0.57	0.78	0.65	0.75	0.00

Критерий однородности при $t_{\alpha}=1.36$

	5	10	100	200	400	600	800	1000
5	Подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
10	Подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
100	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят
200	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят	Не подходят
400	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят	Не подходят
600	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят	Не подходят
800	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят	Не подходят
1000	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Не подходят	Подходят

вывод: ☻

Задание 3

In [96]:
```python
#asd
```

Домашнее задание 5

Дискретное

Задание 1

Функция отношения правдоподобия

Функция отношения правдоподобия $l(\bar{X}) = \frac{L(\bar{x},\theta_1)}{L(\bar{x},\theta_0)}$. Предполагается, что берется гипотеза H_0, что некоторое распределение параметризированно с помощью θ_0, и гипотеза H_1, что оно же параметризированно с помощью θ_1. Берется некоторая константа c_α, которая как-то отражает ошибку первого рода α, и если $l(\bar{X}) > c_\alpha$, то принимается гипотеза H_1, иначе - H_0. $l(\bar{X}) > c_\alpha \Leftrightarrow \frac{L(\bar{x},\theta_1)}{L(\bar{x},\theta_0)} > c_\alpha \Leftrightarrow L(\bar{x},\theta_1) > c_\alpha \cdot L(\bar{x},\theta_0)$ и соответственно θ_1 более правдоподобна, чем θ_0.

В случае дискретного равномерного распределения имеем $L(\bar{x},\theta) = \frac{1}{\theta^n}\mathtt{Ind}(1 \le X_{(1)}) \cdot \mathtt{Ind}(X_{(n)} \le \theta)$, что значит $l(\bar{X}) = \frac{L(\bar{x},\theta_1)}{L(\bar{x},\theta_0)} = \frac{\theta_1^{-n}\cdot\mathtt{Ind}(1\le X_{(1)})\cdot\mathtt{Ind}(X_{(n)}\le\theta_1)}{\theta_0^{-n}\cdot\mathtt{Ind}(1\le X_{(1)})\cdot\mathtt{Ind}(X_{(n)}\le\theta_0)} = \frac{\theta_0^n\cdot\mathtt{Ind}(X_{(n)}\le\theta_1)}{\theta_1^n\cdot\mathtt{Ind}(X_{(n)}\le\theta_0)} = \left(\frac{\theta_0}{\theta_1}\right)^n \cdot \frac{\mathtt{Ind}(X_{(n)}\le\theta_1)}{\mathtt{Ind}(X_{(n)}\le\theta_0)}$.

Задание 2

Заметим, что если $X_{(n)} > \theta_1$, то $l(\bar{X})=0\theta_0, тоl(\bar{X})=+\inf>c_{\alpha} \forall\alpha ипринимаетсягипотезаH_1. Еслижеи\theta_0, и\theta_1меньшеX_{(n)}, тоимеемнеопределенность\left[\frac{0}{0}\right]$ и отвергаются обе гипотезы.

В дальнейшем будем считать $\theta_1 > \theta_0$.

Рассмотрим отношение правдоподобия для одного измерения: $l(X) = \frac{\theta_0}{\theta_1}\cdot\frac{\mathtt{Ind}(X\le\theta_1)}{\mathtt{Ind}(X\le\theta_0)}

Ошибка первого рода $\alpha =

```
In [97]: #ЗЫ. Я просто не успеваю это сделать. Оставлю как черновичок.
```

Абсолютно непрерывное

Задание 1

$$l(\bar{X}) = \frac{L(\bar{x}, \theta_1)}{L(\bar{x}, \theta_0)} = \prod_{i=1}^{n} \frac{\frac{2x_i}{\theta_1} \cdot \mathtt{Ind}(0 \le x_i \le \theta_1) + \frac{2(1-x_i)}{1-\theta_1} \cdot \mathtt{Ind}(\theta_1 < x_i \le 1)}{\frac{2x_i}{\theta_0} \cdot \mathtt{Ind}(0 \le x_i \le \theta_0) + \frac{2(1-x_i)}{1-\theta_0} \cdot \mathtt{Ind}(\theta_0 < x_i \le 1)}.$$

Предположим, что $\theta_1 > \theta_0$. В силу непрерывности разбиения, найдутся такие k_1 и k_0, что $x_{k_1}<\theta_1\theta_0
, тоk_1\ge k_0$. Тогда

$$l(\bar{X}) = \prod_{i=1}^{k_0} \frac{2x_i \cdot \theta_0}{2x_i \cdot \theta_1} \cdot \prod_{i=k_0+1}^{k_1} \frac{2x_i \cdot (1-\theta_0)}{2(1-x_i) \cdot \theta_1} \cdot \prod_{i=k_1+1}^{n} \frac{2(1-x_i) \cdot (1-\theta_0)}{2(1-x_i) \cdot (1-\theta_1)} = \left(\frac{\theta_0}{\theta_1}\right)^{k_0} \cdot \left(\frac{1-\theta_0}{1-\theta_1}\right)^{n-k_1}$$
$$\cdot \left(\frac{1-\theta_0}{\theta_1}\right)^{k_1-k_0} \cdot \prod_{i=k_0+1}^{k_1} \frac{x_i}{1-x_i} =$$

$.

Задание 2

In [98]:
```
#ы
```