Лабораторная работа №2

Морозов Антон, M32381

Задача 1. Вариант 2 Случайная величина X принимает ровно два значения. Доказать, что она не может быть представлена как сумма двух независимых случайных величин, каждая из которых имеет невырожденное распределение

Решение:

Предположим, что наша случайная величина представима в виде суммы двух невырожденных случайных величин: $X = Y + Z$ тогда каждая из них имеет хотя бы 2 различных значения. Рассмотрим, когда у каждой два значения $\Rightarrow \phi_Y = p_1 e^{x_1 it} + p_2 e^{x_2 it}, \phi_Z = q_1 e^{y_1 it} + q_2 e^{y_2 it} \Rightarrow$

$\phi_X = \phi_y \cdot \phi_Z = p_1 q_1 e^{(x_1+y_1)it} + p_1 q_2 e^{(x_1+y_2)it} + p_2 q_1 e^{(x_2+y_1)it} + p_2 q_2 e^{(x_2+y_2)it}$

Тогда заметим, что для любой возможной комбинации, когда остается только 2 значения, то мы приходим к противоречию. Например если мы хотим объединить первые 3, то получается, что $x_1 + y_1 = x_1 + y_2 = x_2 + y_1 \Rightarrow y_1 = y_2$ – противоречие с не выражденностью. Тогда получается, что если у нас есть сумма двух независимых случайных величин, каждая из которых имеет невырожденное распределение, то она имеет не менее 3х значений. Для 3х существует пример $(x_1 = y_1, x_2 = y_2)$

Задача 2. Вариант 3 Пусть $U, V \sim U[0,1]$ и они независимы. Определим

$$X = \sqrt{-2\ln V}\cos 2\pi U, Y = \sqrt{-2\ln V}\sin 2\pi U$$

Показать, что (X, Y) – стандартный гауссовский вектор.

Решение:

Чтобы показать, что это стандартный гауссовский вектор, то найдем плотность этого совместного распределения. Для этого воспоьзуемся формулой для вычисления плотности преобразования сулчайных величин, для двумерного случая:

$$p_{Y_1,Y_2}(y1, y2) = p_{X_1,X_2}(x_1(y_1,y_2), x_2(y_1,y_2))|\det D(x_1(y_1,y_2), x_2(y_1,y_2))|$$

Для нахождения матрицы якоби выразим U, V через X, Y:

$$x^2 + y^2 = -2\ln v \Rightarrow v = e^{-\frac{x^2+y^2}{2}}$$
$$\frac{y}{x} = tg(2\pi u) \Rightarrow u = \frac{1}{2\pi}\arctan\frac{y}{x}$$

$$\det D = \begin{vmatrix} \mathrm{u}_x^{'} & \mathrm{u}_y^{'} \\ \mathrm{v}_x^{'} & \mathrm{v}_y \end{vmatrix} = \begin{vmatrix} \frac{-y}{2\pi(x^2+y^2)} & \frac{x}{2\pi(x^2+y^2)} \\ -xe^{-\frac{x^2+y^2}{2}} & -ye^{-\frac{x^2+y^2}{2}} \end{vmatrix} = x^2\frac{e^{-\frac{x^2+y^2}{2}}}{2\pi(x^2+y^2)} + y^2\frac{e^{-\frac{x^2+y^2}{2}}}{2\pi(x^2+y^2)} = \frac{1}{2\pi}e^{-\frac{x^2+y^2}{2}}$$

Так как $U, V \sim U[0,1] \Rightarrow p_{U,V}(u,v) = 1$. Тогда

$$p_{X,Y}(x,y) = p_{U,V}(u,v)|detD| = 1\cdot\frac{1}{2\pi}e^{-\frac{x^2+y^2}{2}}$$

А как известно, это плостность стандартного гауссовского вектора.

Задача 3. Вариант 2

Для заданной плотности $p(x) = \frac{5x^4}{\sqrt{2\pi}} e^{-\frac{(5-x^5)^2}{2}}$ реализовать генерацию случайных чисел и сравнить производительность:

1. Для заданной плотности p наследуйтесь от класса rv_continuous и сгенерируйте с помощью реализованного подкласса для вашего распределения n случайных чисел из данного распределения. Посмотрите, как будет работать алгоритм при росте n (сразу большим n не делайте).

2. Реализуйте генерацию для вашего распределения с помощью обратной к функции распределения (при написании функции для F^{-1} вызов специальных функций, например, квантили стандартного нормального закона разрешается). Проведите тот же эксперимент

3. Выберите еще один метод для генерации случайных чисел (можно, например, rejecting sampling, ratio of uniforms или другой метод). Опишите его математическое обоснование. Воспользуйтесь реализацией или сами реализуйте. Проведите тот же эксперимент.

Решение:

1. Пришлось ограничить отрезок, из которого выбирается x, потому что иначе получаются слишком большие значение вида 1e+1000001, из-за чего вычисления становятся невероятно долгими и неправильными из-за переполнения или длинной арифметики, можно посмотреть на график нашего распределения и ограничить отрезком: $[-2, 2]$, в граничных точках плотность соответственно равны $10^{-296}, 10^{-157}$ Реализация:

```python
class MyDistribution(ss.rv_continuous):
    def __init__(self):
        super().__init__(momtype=0, a=-2, b=2)

    def _pdf(self, x, *args):
        return density(x)


def method_1(count_points=100):
    my = MyDistribution()
    rng = np.random.SeedSequence().entropy % (2 ** 32)
    return my.rvs(size=count_points, random_state=rng)
```

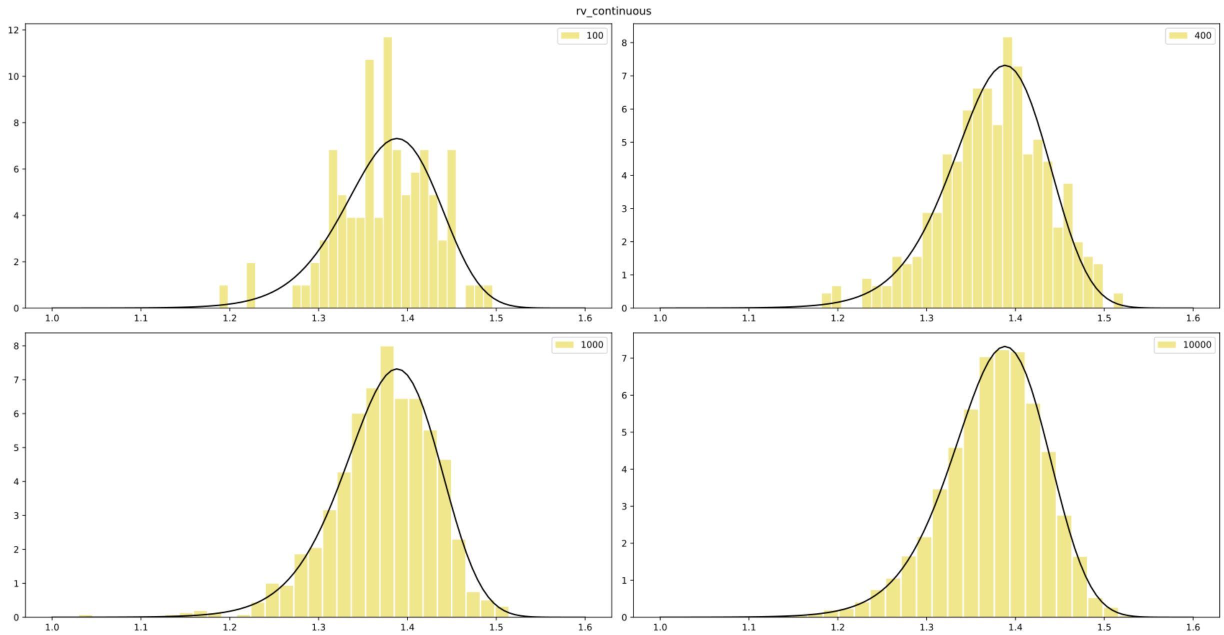

Рис. 1: rv_continuous: выборки 100, 400, 1000 и 10000 значений

Можно сразу из Рис. 1 видеть, что при больших n распределение стремится к заданному, при $n = 10000$, можно видеть очень близкое распределение. Однако данный способ работает очень долго, особенно при больших n, в среднем выходит около 0.06. Однако получается, аккуратный ООП код

2. Метод обратной функции. В данном случае, чтобы сгенерировать точку, равномерно генерируем из промежутка $[0, 1]$, после чего находим квантиль с такой вероятность, здесь нам и нужна обратная фукнция, чтобы быстро решать уравнение. Это и есть генератор случайной точки с данным распределением. Тогда, найдем функцию распределение в моем варианте:

$$F(x) = \frac{1}{2} \operatorname{erf} \frac{x^5 - 5}{\sqrt{2}}$$

$$F^{-1}(p) = (\sqrt{2} \cdot \operatorname{erf}^{-1}(2p) + 5)^{\frac{1}{5}}$$

```
def method_2(count_points=100):
def gen():
    rnd = random.uniform(0, 1)
    return (ssp.erfinv(2 * (rnd - 0.5)) * math.sqrt(2) + 5) ** (1 / 5)

return [gen() for _ in range(count_points)]
```

Аналогично, при больших n стремится к нашей плотности. Однако работает гораздо быстрее, но стоит заметить, что в нашем случае обратная функция считается очень быстро и вообще имеет красивое аналитическое выражение, такое бывает достаточно редко.

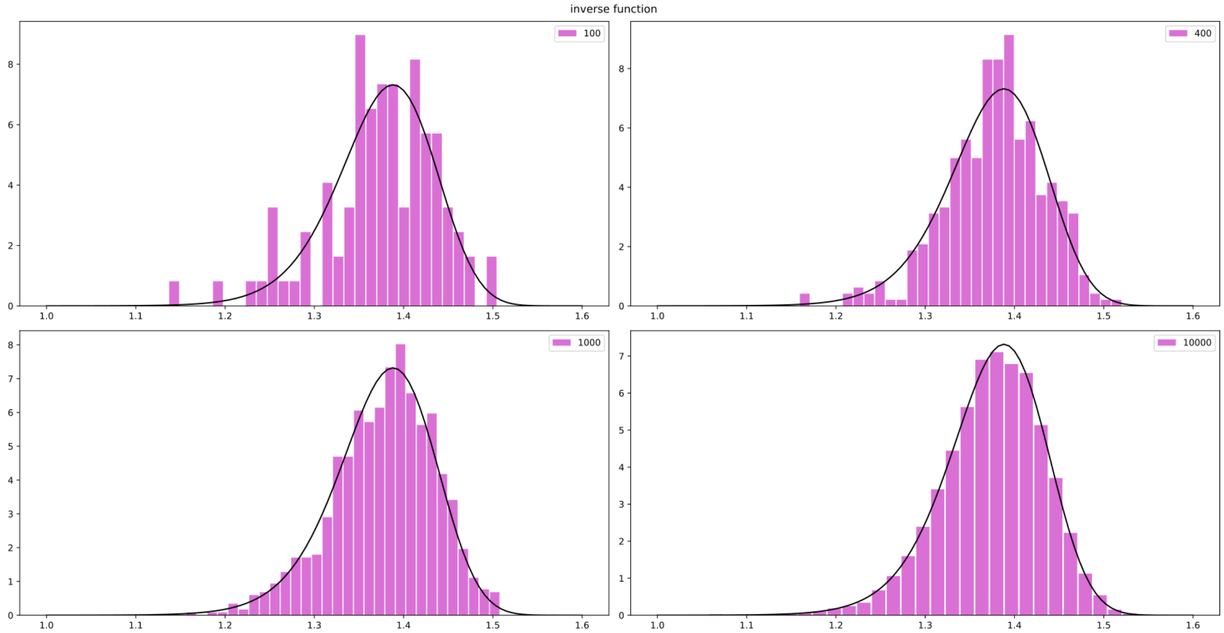

Рис. 2: Метод обратной функции: выборки 100, 400, 1000 и 10000 значений

3. Rejecting sampling. Идея алгоритма, пусть у нас есть данная сложная плотность $f(x)$. Тогда давайте сведем ее к плотности проще $g(X)$, которую мы умеем генерировать. Но это плотность должна покрывать нашу, то есть $c > 1\ \forall\ x \in E : f(x) < cg(x)$. Алгоритм:

   (a) Генерируем значение x по g

   (b) Генерируем равномерно значение t в $[0, cg(x))$

   (c) Если $t <= f(x)$ — есть наше значение, иначе повторяем

   Давайте найдем вероятность того, что значение t будет принято

$$P(t \text{ accepted}) = P(cg(t) \leqslant f(t)) = \int P(cg(t) \leqslant f(t) | t = x) g(x) dx = \int \frac{f(x)}{cg(x)} g(x) dx = \frac{1}{c}$$

Давайте покажем, что условное распределения будет совпадать, с нашим исходным $F(x)$:

$$P(t \leqslant a | t \text{ accepted}) \underset{\text{фор-ла Байеса}}{=} \frac{P(t \leqslant a \vee t \text{ accepted})}{P(t \text{ accepted})} = \frac{\int_{-\infty}^{a} f_{T,ACCEPTED}(t,1)dt}{1/c} =$$

$$= c\int_{-\infty}^{a} f_{ACCEPTED|T}(1,t)g(t)dt = c\int_{-\infty}^{a} \frac{f(t)}{cg(t)}g(t)dt = c\frac{F(a)}{c} = F(a)$$

Таким образом, показали, что корректно, тогда с данной плотность я выберу $g \sim N[1.4, 0.04]$

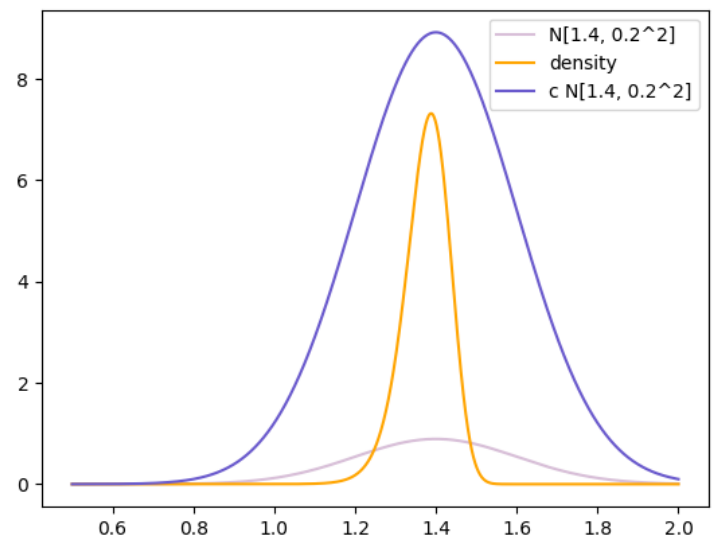

Рис. 3: Графики плотностей $f(x)$, $g(x)$ и $cg(x)$

```
def method_3(count_points=100):
c = 10

def approx(x):
    sigma = 0.2
    return 1 / math.sqrt(sigma * 2 * math.pi) * math.exp(- (x - 1.4) ** 2 / 2 / sigma ** 2)

def gen():
    while True:
        x = random.normalvariate(1.4, 0.2)
        u = random.uniform(0, c * approx(x))
        if u <= density(x):
            return x

return [gen() for _ in range(count_points)]
```

Результаты сошлись к распределению и по времени быстрою (Рис. 4)

По следующей таблице, в которой сравнивается время, можно увидеть, что наследование от rv_continuous работает на несколько порядков дольше, двух остальных, быстрее всех оказался метод обратной функции, так как обратная функция считается очень быстро.

	rv_continuous	Обратная фу-я	rejection sampling
100	8.62628	0.001	0.306
400	31.72289	0.001	0.25213
1000	60.29092	0.003	0.308
10000	572.22059	0.0162	1.349

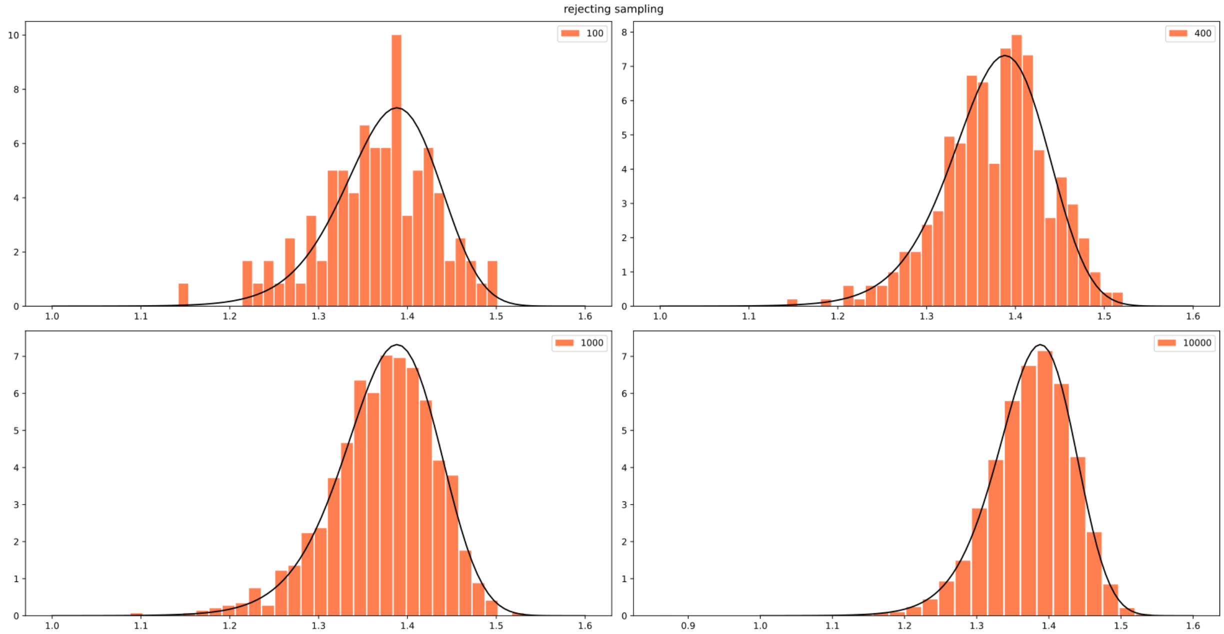

Рис. 4: Rejecting sampling: выборки 100, 400, 1000 и 10000 значений

Задача 4. Вариант 5 Пусть $X_1, \dots X_n$ – i.i.d. величины из некоторого распределения, которое параметризуется параметров $\theta \in E$. Пусть $\overline{X}_n = \frac{1}{n} \cdot (X_1 + \dots + X_n)$, μ_0 – математическое ожидания случайной величины X_0. Найти с помощью неравенства Чебышёва и центральной предельной теоремы номер n, при котором равномерно для $\theta \in E$ и заданных $\delta, \in (0,1)$ выполняется соотношение

$$P(|\overline{X}_n - \mu_n| \leqslant \varepsilon) \geqslant 1 - \delta$$

Для найденного по неравенству Чебышёва n сгенерировать 100 выборок найденного объема n и посчитать количество и долю выборок, для которых $|\overline{X}_n - \mu_n| \leqslant \varepsilon$. Параметры эксперимента: $\varepsilon = 0.01, \delta = 0.05$ и θ, при котором найдена равномерная оценка. То же самое сделать и для n, найденного с помощью ЦПТ.

В вариантах указывается класс распределений, затем множество возможных значений параметра E $\mathrm{Exp}(\lambda)$, $\lambda \in [1,5]$

Решение:

Так как $X_1, \dots X_n$ — i.i.d, такие что $X_1, \dots X_n \sim \mathrm{Exp}(\lambda) \Rightarrow \sum_{i=1}^n X_i \sim \Gamma(n, \frac{1}{\lambda}) \Rightarrow \frac{1}{n} \cdot \sum_{i=1}^n X_i \sim \Gamma(n, \frac{1}{n\lambda})$

Пусть σ^2 - дисперсия $\overline{X}_n$, тогда $\sigma^2 = \frac{n}{n^2\lambda^2} = \frac{1}{n\lambda^2}, \mu_n = \frac{n}{n\lambda} = \frac{1}{\lambda}$

Неравенство Чебышева для $\overline{X}_n$:

$$P(|\overline{X}_n - \mu_n| \leqslant a) \geqslant 1 - \frac{\sigma^2}{a^2}$$

Тогда, чтобы получиться нужное нам неравенство должно выполняться $a = \varepsilon, 1 - \delta = 1 - \frac{\sigma^2}{a^2}$, тогда отсюда получаем выражение для n:

$$1 - \frac{\sigma^2}{\varepsilon^2} = 1 - \frac{1}{n\lambda^2\varepsilon^2} = 1 - \delta \Rightarrow n = \frac{1}{\varepsilon^2\lambda^2\delta}$$

ЦПТ:

$$P(|\frac{1}{B_n}\sum_{i=1}^n (X_k - \mu_0)| \leqslant \varepsilon_0) = \Phi(\varepsilon_0) - \Phi(-\varepsilon_0) = 2\Phi(\varepsilon_0), B_n^2 = \sum_{i=1}^n \sigma_k^2 = \frac{n}{\lambda^2} \Rightarrow B_n = \frac{\sqrt{n}}{\lambda}$$

$$P(|\frac{\lambda}{\sqrt{n}}(n\overline{X}_n - \frac{n}{\lambda})| \leqslant \varepsilon_0) = 2\Phi(\varepsilon_0)$$

$$P(|\overline{X}_n - \frac{1}{\lambda}| \leqslant \frac{\varepsilon_0}{\sqrt{n}\lambda}) = 2\Phi(\varepsilon_0)$$

$$P(|\overline{X}_n - \mu_n| \leqslant \frac{\varepsilon_0}{\sqrt{n}\lambda}) = 2\Phi(\varepsilon_0) = 1 - \delta = 0.95 \Rightarrow \varepsilon_0 = 1.96$$

Тогда $\varepsilon = \frac{\varepsilon_0}{\sqrt{n}\lambda} \Rightarrow n = (\frac{\varepsilon_0}{\varepsilon\lambda})^2$

Тогда подставим для различных целых значение λ

λ	ЦПТ	Не-во Чебышева
1	38416	200000
2	9604	50000
3	4269	22223
4	2401	12500
5	1537	8000

В результате экспериментов получается, что ЦПТ дает лучшую оценку, поэтому значения вероятности находятся в окрестность 0.95, а неравенство Чебышева дает очень грубую оценку и значения вероятности находятся в окрестности 1(преймущественно 1)

```
Chebyshev count: 100, fraction: 1.0
CLT count: 96, fraction: 0.96
```

```python
def chebyshev():
    return 1 / eps ** 2 / delta / theta ** 2


def CLT():
    return (inv_lapalas / theta / eps) ** 2


def experiment(method: typing.Callable):
    cnt = 0
    for _ in range(NUMBER_EXPERIMENTS):
        n = math.ceil(method())
        values = np.random.exponential(scale=1 / theta, size=n)
        if math.fabs(sum(values) / n - 1 / theta) <= eps:
            cnt += 1
    return cnt
```