Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №2

по дисциплине «Теория вероятностей»

Выполнил:

Чечеватов Роман

Факультет: ФИТиП

Группа: М32381

Преподаватель: Мурзина А. А.

Санкт-Петербург 2023

ЗАДАЧА 1, ВАРИАНТ 3

Условие: случайная величина X принимает значения $-1, 0, 1$ с вероятностями p_1, p_2, p_3 соответственно. Какие условия нужно наложить на p_1, p_2, p_3, чтобы X была представима в виде суммы двух независимых одинаково распределенных случайных величин?

Примечание: X принимает только описанные значения, и никакие другие. X – дискретная случайная величина.

Решение:

Пусть дискретная С.В. X представима в виде $Y_1 + Y_2$. Если в таком случае Y_i принимает хотя бы три значения – a, b, c, то $Y_1 + Y_2$ принимает в том числе значения $a + a, a + b, a + c, b + c$ – 4 различных значения. Значит, Y_i не может принимать три и более различных значений. Если принимает всего одно, то $Y_1 + Y_2$ тоже принимает одно – что меньше, чем необходимые три. Значит, Y_i принимает два различных значения. Раз множество значений случайной величины Y_i конечно, то сама случайная величина дискретна. Обозначим значения Y_i как a, b. Тогда $\{2 * a, a + b, 2 * b\} = \{-1, 0, 1\}$, откуда $a = \frac{1}{2}, b = -\frac{1}{2}$.

Обозначим $p = P\left(Y = \frac{1}{2}\right), q = P\left(Y = -\frac{1}{2}\right)$, тогда

$$\begin{cases} p_1 = q^2 \\ p_2 = 2 * p * q \\ p_3 = p^2 \end{cases}, \text{то есть } \begin{cases} p_2 = 2\sqrt{p_1 p_3} \\ q = \sqrt{p_1} \\ p = \sqrt{p_3} \end{cases}$$

Ответ:

$$\begin{cases} p_1 + p_2 + p_3 = 1 \\ p_1, p_2, p_3 \geq 0 \\ p_2 = 2\sqrt{p_1 p_3} \end{cases}$$

ЗАДАЧА 2, ВАРИАНТ 3

Условие: 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:

Воспользуемся формулой обращения плотности:

$$p_{X,Y}(U,V) = p_{U,V}\big(U(X,Y), V(X,Y)\big) * \left|det\begin{pmatrix}\nabla U(X,Y)\\ \nabla V(X,Y)\end{pmatrix}\right|$$

$$U(X,Y) = \exp\left(\frac{X^2+Y^2}{-2}\right)$$

$$V(X,Y) = \frac{1}{2\pi} arctg\left(\frac{Y}{X}\right)$$

$p_{U,V}\big(U(X,Y), V(X,Y)\big) = 1$ как плотность равномерых С. В.

$$\begin{pmatrix}\nabla U(X,Y)\\ \nabla V(X,Y)\end{pmatrix} = \begin{pmatrix} -X * \exp\left(\frac{X^2+Y^2}{-2}\right) & -\mathrm{Y} * \exp\left(\frac{X^2+Y^2}{-2}\right) \\ -\frac{1}{2\pi} * \frac{Y}{X^2+Y^2} & \frac{1}{2\pi} * \frac{X}{X^2+Y^2} \end{pmatrix}$$

$$det\begin{pmatrix}\nabla U(X,Y)\\ \nabla V(X,Y)\end{pmatrix} = -\frac{1}{2\pi} * \exp\left(\frac{X^2+Y^2}{-2}\right)$$

$$\left|det\begin{pmatrix}\nabla U(X,Y)\\ \nabla V(X,Y)\end{pmatrix}\right| = \frac{1}{2\pi} * \exp\left(\frac{X^2+Y^2}{-2}\right)$$

$$p_{X,Y}(U,V) = \frac{1}{2\pi} * \exp\left(\frac{X^2+Y^2}{-2}\right)$$

Полученная плотность и есть плотность стандартного гауссовского вектора.

ЗАДАЧА 3, ВАРИАНТ 3

Условие:

- Для заданной плотности $p(x) = \frac{4 \ln x^3}{x} * \mathbb{1}(x \in [1, e])$ 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).
- 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.
- 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. Создаем новый класс, наследуемый от `rv_continuous`, переопределяем плотность распределения (также обозначается pdf). Код представлен на листинге. Функции pdf, pdf_unsafe вынесены отдельно, так как пригодятся при реализации других методов.

```
# Листинг
def pdf_unsafe(x: float) -> float:
    return 4 * math.log(x) ** 3 / x

def pdf(x: float) -> float:
    if x < 1 or x > math.e:
        return 0
    return pdf_unsafe(x)

class Dist(rv_continuous):
    def __init__(self, xtol=1e-14, seed=None):
        super().__init__(0, 1, math.e, xtol=xtol, seed=seed)

    def _pdf(self, x, *args):
        return pdf_unsafe(x)
```

2. Чтобы найти обратную функцию к F, нужно сначала найти F:

$$F(x) = \int_{-\infty}^{x} p(x)\,dx = \int_{1}^{x} \frac{4\ \ln x^3}{x} dx = (\ln x)^4|_1^x = (\ln x)^4$$

$$F^{-1}(x) = \exp\left(\sqrt[4]{x}\right)$$

Теперь, значение искомой случайной величины X можно получить как $F^{-1}(y)$, где y распределен равномерно в диапазоне $[0, 1]$. Код представлен на листинге 2.

```python
# Листинг 2
def cdf_inverse(y: float) -> float:
    return math.exp(y ** .25)

lambda: cdf_inverse(random.uniform(0, 1))  # Получить значение С.В. X
```

3. В качестве третьего подхода к генерации случайной величины я выбрал технику rejection sampling. Идея алгоритма заключается в следующем: выбрать равномерно распределенную точку $K = (x, y) \in [x_{min}, x_{max}] \times [0, y_{max}]$, после этого проверить, правда ли что $y < p(x)$. Если да, то x является искомым значением случайной величины. Иначе, сгенерировать новую точку и повторить. При этом необходимо выполнение следующих условий: $y_{max} \geq \max p(x)$, $\forall x \in \mathbb{R}\backslash[x_{min}, x_{max}]\ p(x) = 0$, $\int_{x_{min}}^{x_{max}} p(x)dx = 1$.

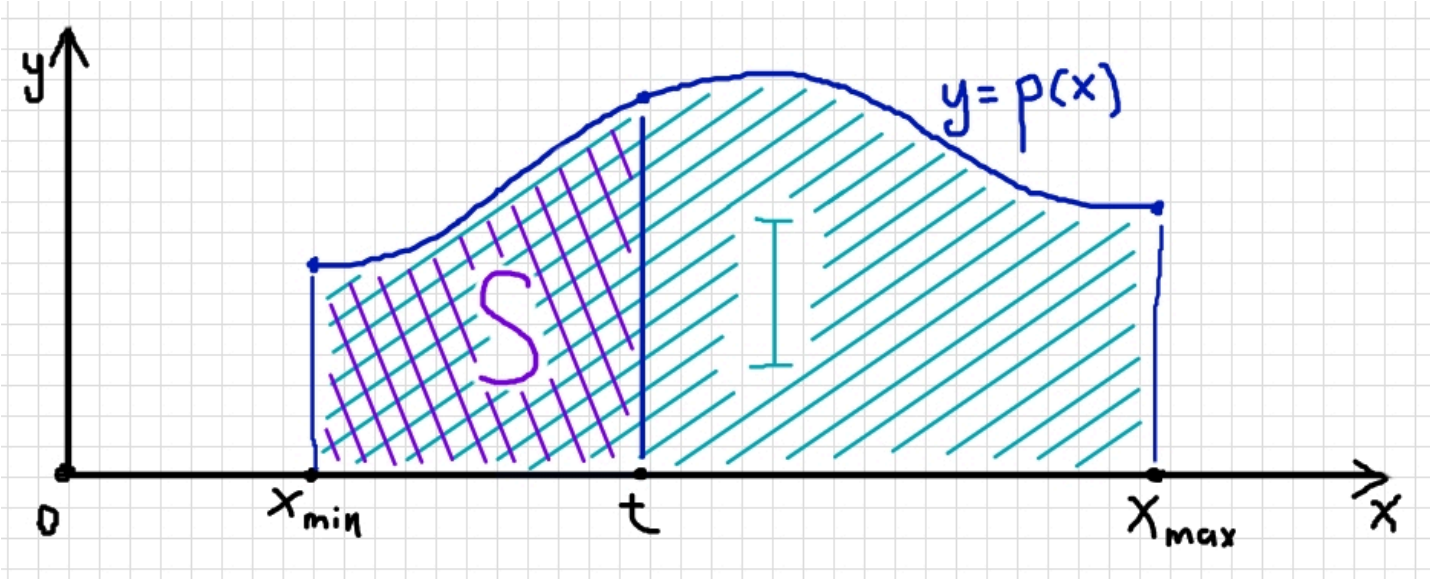

Рисунок 1 – поясняющий график к алгоритму rejection sampling

Почему такой метод работает? Рассмотрим вероятность, с которой сгенерированное значение случайной величины не превосходит заданного значения t. По построению, $K \in I$ (см. рисунок 1), распределена равномерно в I. Сгенерированная случайная величина не превосходит t, если $K \in S$. Тогда искомая вероятность равна

$$F_{X_{gen}}(t) = P\left(X_{gen} \le t\right) = \frac{Area(S)}{Area(I)} = \frac{\int_{x_{min}}^{t} p(x)dx}{1} = F_{X_{real}}(t).$$

Таким образом, функции распределения сгенерированной и настоящей случайной величины совпадают, а значит совпадают и сами величины.

Заметим, что выбор y_{max} не влияет на корректность работы алгоритма, при условии $y_{max} \geq \max p(x)$. Но влияет на время работы – при больших значениях, больше точек будет сгенерировано за пределами I – понадобится больше итераций, чтобы найти подходящую точку.

Реализация алгоритма представлена на листинге 3.

```
# Листинг 3
def rejection_sampling(xmin, xmax, pdf, pmax):
    while True:
        x = random.uniform(xmin, xmax)
        y = random.uniform(0, pmax)
        if y <= pdf(x):
            return x

def getDistByRejection(pmax: float):
    return rejection_sampling(1, math.e, pdf_unsafe, pmax)
```

4. Эксперименты: для начала сгенерируем 100 000 точек, и проверим что они распределены действительно так как нужно. Рассмотрим рисунки 2–4, на них вершины столбцов гистограмм должны идти по графику плотности распределения – обозначен черной линией. Наблюдаемое поведение совпадает с ожидаемым, значит реализации корректны.

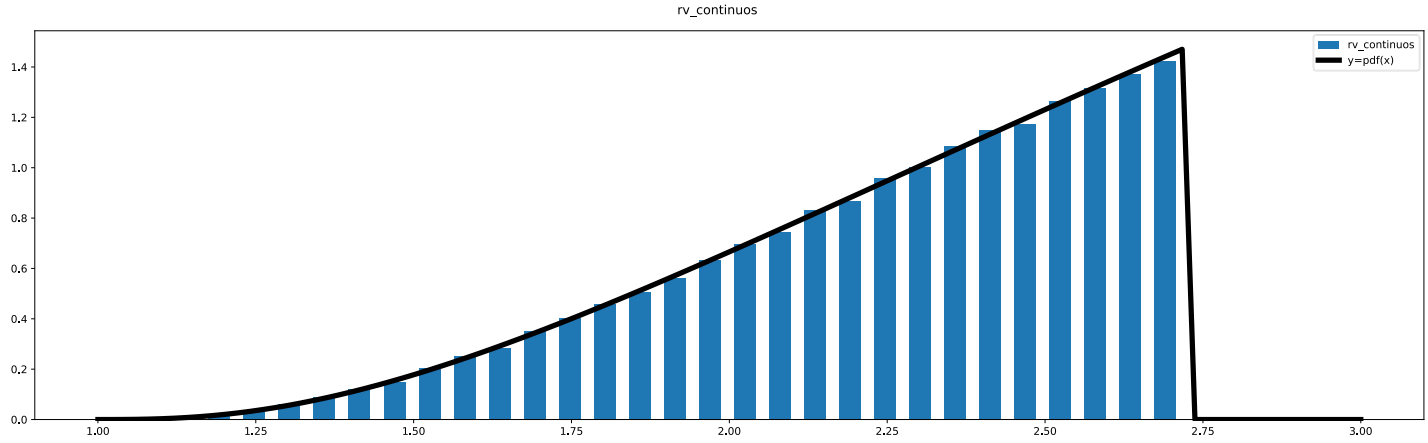

Рисунок 2 – график сгенерированного через rv_continuous распределения

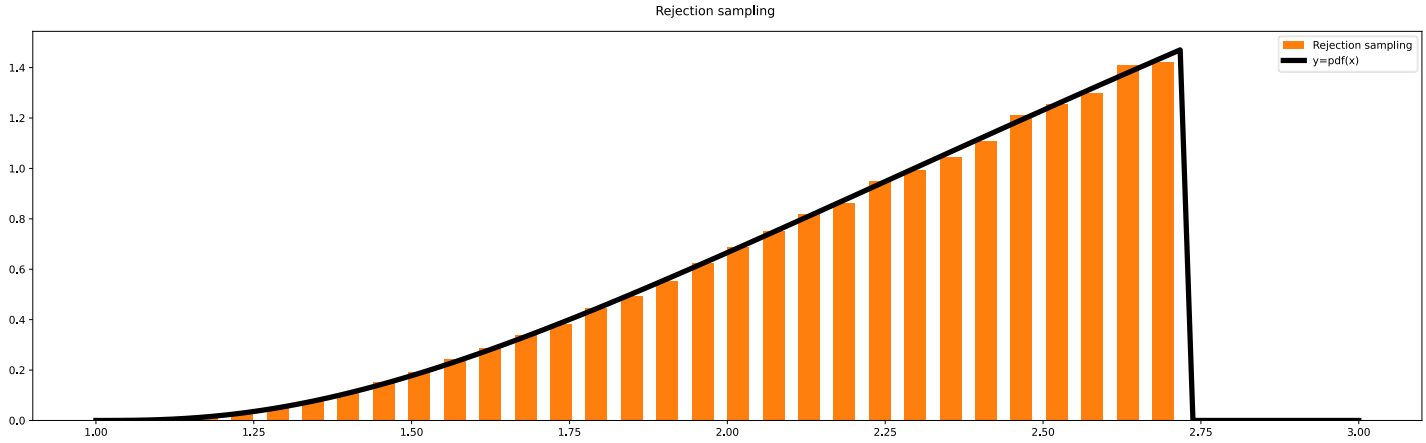

Рисунок 3 – график сгенерированного с помощью rejection sampling распределения

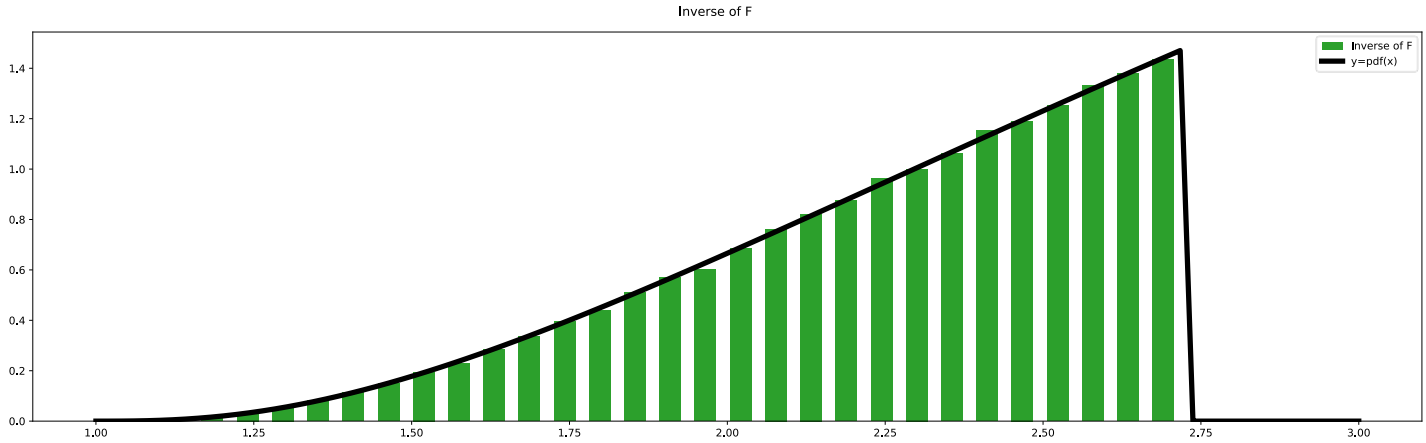

Рисунок 4 – график сгенерированного через F^{-1} распределения

Далее проведем анализ производительности, а именно, времени, затрачиваемого на генерацию одной точки при генерации серии из n точек, для различных значений n. Данные представлены на рисунке 5 (вертикальная ось – логарифмическая).

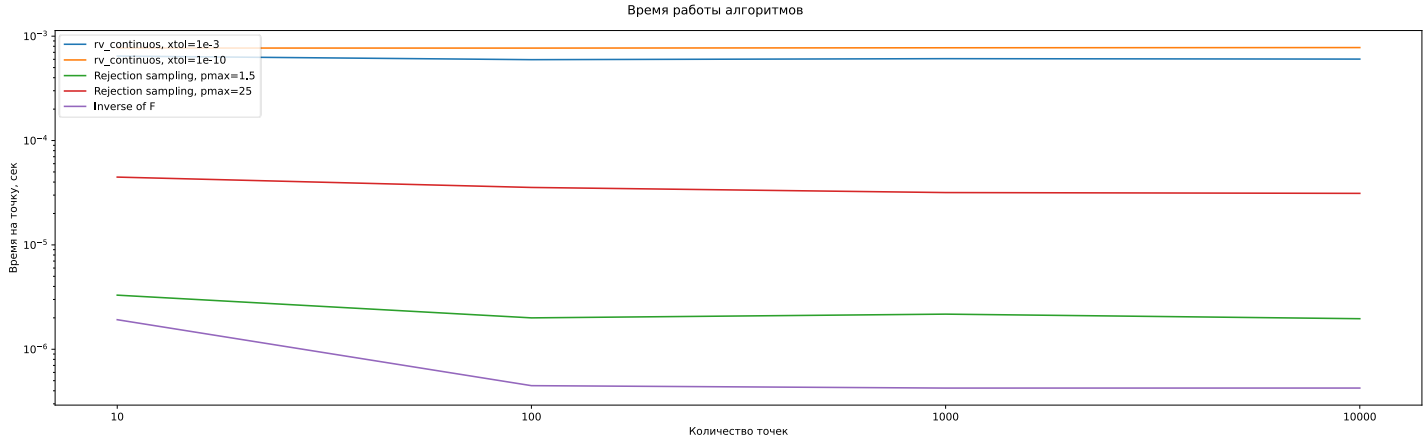

Рисунок 5 – график зависимости времени генерации точки от количество точек

Заметно, что самый простой в реализации способ – наследоваться от rv_continuous – показывает заметно худшие результаты, по сравнению с остальными. Такое поведение

легко объяснить: не зная практически ничего о распределении случайной величины, реализация вынуждена пользоваться низкопроизводительными численными методами. У конструктора класса rv_continuous есть параметр xtol – точность вычислений. Снижение точности увеличивает скорость работы, но незначительно.

Алгоритм, использующий функцию F^{-1} показывает наилучшую производительность среди рассмотренных алгоритмов, что ожидаемо, с учетом того, как на самом деле просто вычислить эту функцию. Однако стоит заметить, что не для каждой функции будет легко или даже возможно посчитать интеграл плотности, выразить его в элементарных функциях, и найти обратную функцию, которая ещё и проста вычислительно.

Производительность rejection sampling, как и было описано ранее, зависит от параметра y_{max}. Однако и полученный мной результат, полагаю, зависит от функции плотности. Если бы плотность распределения была с очень широкими хвостами, и одним узким, но высоким пиком, то прямоугольник $[x_{min}, x_{max}] \times [0, y_{max}]$ оказался бы большим, а площадь подграфика – маленькой, что привело бы к большому числу итераций, необходимому для генерации каждой точки.

Время, затрачиваемое на генерацию одной точки, практически не зависит от количества генерируемых точек, а отклонения от горизонтальной прямой в начале графиков наверняка вызваны работой программы, замеряющей время.

ЗАДАЧА 4, ВАРИАНТ 3

Условие:

Пусть $X_1, \dots, X_n$ – i.i.d. величины из распределения $Geom(p), p \in \left[\frac{1}{4}, \frac{3}{4}\right] = E$. Пусть $\overline{X}_n = {X_1 + \dots + X_n}/{n}$, $\mu_\theta = EX_i$. С помощью неравенства Чебышева и центральной предельной теоремы найти номер $n_{\delta,\varepsilon}$, при котором для любого $\theta \in E$ и заданных δ, ε выполняется соотношение $P\left(\left|\overline{X}_{n_{\delta,\varepsilon}} - \mu_\theta\right| \le \varepsilon\right) \ge 1 - \delta$.

Сгенерировать 100 выборок найденного объема $n_{\delta,\varepsilon}$ и посчитать количество и долю выборок, для которых $\left|\overline{X}_{n_{\delta,\varepsilon}} - \mu_\theta\right| \le \varepsilon$. Параметры: $\varepsilon = 0.01, \delta = 0.05$.

Решение:

Для начала заметим, что $E\overline{X}_n = {EX_1 + \dots + EX_n}/{n} = {EX_1 * n}/{n} = EX_1 = \mu_\theta$. Также обозначим $Y = \overline{X}_{n_{\delta,\varepsilon}}$.

Теперь перейдем к неравенству Чебышева:

$$P(|X - EX| \ge \varepsilon) \le \frac{DX}{\varepsilon^2}$$

$$P(|X - EX| \le \varepsilon) \ge 1 - \frac{DX}{\varepsilon^2}$$

Тогда, ${DY}/{\varepsilon^2} \le \delta$, $DY \le \delta\varepsilon^2$.

Так как случайные величины $X_1, \dots, X_n$ независимы, то $DY = {\sum_{i=1}^{n_{\delta,\varepsilon}} DX_i}/{n_{\delta,\varepsilon}^2} = {DX_1}/{n_{\delta,\varepsilon}}$. $DX_1 = {1-p}/{p^2}$. $DX_1{}'_p = {p-2}/{p^3}$, то есть на отрезке $\left[\frac{1}{4}, \frac{3}{4}\right]$ DX_1 монотонна по p, максимальное значение принимает при $p = \frac{1}{4}$ – это значение 12.

Решаем неравенство $\frac{12}{n_{\delta,\varepsilon}} \le \delta\varepsilon^2$, получаем $n_{\delta,\varepsilon} \ge {12}/{\delta\varepsilon^2}$. Подставляем значения параметров, получаем $n_{\delta,\varepsilon} \ge 2\,400\,000$.

Для решения через центральную предельную теорему сначала выпишем её формулировку, в сокращенном виде: (стремится по распределению)

$$\lim_{n \to +\infty} P\left(\sqrt{n}\frac{\overline{X}_n - \mu_\theta}{\sqrt{DX_1}} \le t\right) = \Phi(t)$$

Иными словами

$$P\left(\left|\overline{X}_n - \mu_\theta\right| \le \varepsilon\right) = P\left(\frac{|S_n - n\mu_\theta|}{\sqrt{n * DX_1}} \le \frac{\varepsilon\sqrt{n}}{\sqrt{DX_1}}\right) \approx \Phi\left(\frac{\varepsilon\sqrt{n}}{\sqrt{DX_1}}\right) - \Phi\left(-\frac{\varepsilon\sqrt{n}}{\sqrt{DX_1}}\right) =$$

$$2 * \Phi\left(\frac{\varepsilon\sqrt{n}}{\sqrt{DX_1}}\right) - 1$$

Таким образом, нужно найти наименьшее n, что $2 * \Phi\left(\frac{\varepsilon\sqrt{n}}{\sqrt{DX_1}}\right) - 1 \ge 1 - \delta$, или

$$2 * \Phi\left(\frac{\varepsilon\sqrt{n}}{\sqrt{12}}\right) \ge 2 - \delta$$

Полученное значение n имеет величину 460976. Результаты эксперимента приведены в таблице 1, сгенерированные значения С.В. в приложении 1.

n	Количество подходящих выборок	Доля подходящих выборок	$1 - \delta$
2400000	100	1.0	0.95
460976	97	0.97	

Таблица 1 – результаты эксперимента

Приложение 1

Сгенерированные в задаче 4 значения

№	n=2400000	n=460976
0	3.003570417	2.997004182
1	2.9997775	3.006876714
2	2.998575	2.995589792
3	2.9982275	2.998448943
4	3.001155833	3.007095814
5	3.005312083	3.001583597
6	2.998291667	3.001993596
7	2.99673875	2.999674603
8	2.9979325	3.000882909
9	2.999652083	3.000902433
10	2.996925417	3.004353806
11	3.001943333	3.002859151
12	3.00223	2.99682413
13	3.001176667	2.995307782
14	3.00026	3.000796137
15	2.995715	3.002397088
16	3.001603333	3.002614019
17	3.000389583	3.001783173
18	3.000610417	3.00022127
19	2.995892083	3.003896081
20	3.000827083	3.00065947
21	2.997047083	2.998099684
22	3.001125	3.00154021
23	3.000175	3.006314862
24	2.99712625	2.996550797
25	2.99938125	3.009401791
26	3.00067875	3.001021745
27	3.00082125	3.00537772
28	3.00258125	2.999201694
29	2.996687083	3.000355767
30	3.001780833	3.001442591
31	2.998910417	3.000624761
32	2.997062917	3.004180261
33	3.00216	3.00676174
34	2.99878625	3.00301968
35	2.999224583	3.002648728
36	3.000231667	3.001442591
37	3.000447917	3.003785447
38	2.999487083	3.008182639
39	3.001495	3.006445021
40	2.998214167	2.998028097
41	3.001888333	2.997310055
42	3.0008475	2.996416299
43	3.000986667	3.005572958
44	2.999141667	3.006753063
45	2.995319583	2.991832547
46	3.00247625	2.998140901
47	3.002726667	2.992930218
48	2.996885	3.006935285
49	2.996405417	3.015558294
50	3.00222125	3.001206137
51	2.996377083	3.012890042
52	2.999612917	2.993821804
53	3.00224625	2.992548419
54	3.00000125	2.992854292
55	3.001327917	3.004783329
56	2.998556667	3.005510048
57	2.999653333	3.014801204
58	2.998985833	2.997244976
59	3.00101375	3.002835289
60	2.999789583	2.994726407
61	2.999949583	3.006798619
62	2.999145417	2.995244872
63	3.0053	3.001086824
64	3.002819583	3.001423068
65	3.001835833	2.999446826
66	3.000965417	2.996262278
67	3.001714167	2.997850213
68	3.000358333	2.995581115
69	3.004301667	3.000065079
70	3.000299167	3.009425653
71	3.000605	2.994455243
72	2.999365417	2.998969578
73	3.001059167	3.004505658
74	3.000144167	2.999767884
75	2.998110833	3.001906824
76	2.999780833	2.994064767
77	2.994915417	2.999822117
78	2.99914375	2.997427198
79	2.998887917	2.999704974
80	2.99804875	2.992886831
81	3.003088333	3.00172894
82	3.001277083	2.996448839
83	2.99781375	3.004692218
84	2.999788333	2.993748048
85	3.001897917	2.997153865
86	2.999335	2.992489848
87	2.999598333	2.993698153
88	3.00201375	3.007069782
89	2.997519167	2.990982177
90	2.999514583	3.005013276
91	2.997515	3.002412273
92	3.00337875	3.004327774
93	3.001399167	3.000381799
94	3.000675	3.00455989
95	3.00409125	3.001475131
96	3.002235417	2.996895717
97	3.00487375	3.002811426
98	3.002035	3.005403752
99	2.99883	2.990901912