Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана» (национальный исследовательский университет) (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ФУНДАМЕНТАЛЬНЫЕ НАУКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И МАТЕМАТИЧЕСКАЯ ФИЗИКА (ФН11)

НАПРАВЛЕНИЕ ПОДГОТОВКИ МАТЕМАТИКА И КОМПЬЮТЕРНЫЕ НАУКИ (02.03.01)

О т ч е т

по лабораторной работе № 3

Название лабораторной работы:
Моделирование выборки из абсолютно непрерывного закона распределения методом обратных функций.

Вариант № 9

Дисциплина:

Теория вероятности и математическая статистика

Студент группы ФН11-52Б ____________ **Очкин Н.В.**
(Подпись, дата) (И.О. Фамилия)

Преподаватель ____________ **Облакова Т.В.**
(Подпись, дата) (И.О. Фамилия)

Москва, 2024

Содержание

1 Задание

1. Для данного n методом обратных функций смоделируйте выборку из закона распределения с заданной плотностью $p(x)$.

2. Для полученной выборки найдите гистограмму относительных частот. Постройте на одном рисунке графики теоретической плотности $p(x)$ и гистограмму относительных частот.

3. Вычислите выборочное среднее и выборочную дисперсию и сравните с истинными значениями этих характеристик.

4. Используя неравенство Dvoretzky-Kiefer-Wolfowitz, постройте 90% доверительный интервал для функции распределения $F(x)$.

Приведите графическую иллюстрацию

2 Исходные данные

Вариант: 9 $\quad n : 120$

$$p(x) = \frac{1}{\sqrt{0.4\pi x}} e^{-(\ln x - 2)^2/0.4}, \quad x > 0 \tag{1}$$

3 Решение

3.1 Часть 1

Для данного n методом обратных функций смоделируйте выборку из закона распределения с заданной плотностью $p(x)$.

3.1.1 Функция распределения

Найдем функцию распределения

$$F_X(x) = \int_{-\infty}^{x} f_X(t)dt, \quad \text{где} \tag{2}$$

f_X - плотность распределения.

3.1.1.1 Метод интегрирования Монте-Карло

Для вычисления интеграла воспользуемся численным методом интегрирования Монте-Карло

$$\int_a^b f(x)dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(u_i), \quad \text{где} \tag{3}$$

u - равномерно распредленная на отрезке интегрирования $[a, b]$ случайная величина.

Геометрическая интерпретация данного метода похожа на известный детерминистический метод, с той разницей, что вместо равномерного разделения области интегрирования на маленькие интервалы и суммирования площадей получившихся «столбиков» мы забрасываем область интегрирования случайными точками, на каждой из которых строим такой же «столбик», определяя его ширину как $\frac{b-a}{N}$, и суммируем их площади.

Точность оценки данного метода зависит только от количества точек N.

3.1.1.2 Линейный конгруэнтный метод

Для генерации случайных величин воспользуемся одним из методов генерации псевдослучайных чисел - **Линейным конгруэнтным методом**.
Суть метода заключается в вычислении последовательности случайных чисел X_n, полагая

$$X_{n+1} = (aX_n + c) \bmod m, \quad \text{где} \tag{4}$$

m - модуль ($m \geq 2$);
a - множитель ($0 \leq a < m$);
c - приращение ($0 \leq c < m$);
X_0 - начальное значение ($0 \leq X_0 < m$).

За значениями параметров обратимся к [1].

$$m = 2^{(60)} - 93 \qquad a = 561860773102413563 \qquad c = 0. \tag{5}$$

В случае когда $c = 0$, метод называют **мультипликативным конгруэнтным методом**.

3.1.2 Реализация численного нахождения функции распределения

3.1.2.1 Реализация ЛКМ

Реализуем на языке программирования PYTHON линейный конгруэнтный метод (4), используя параметры (5):

Листинг 1: Реализация ЛКМ

```
class LCG:
  def __init__(self,
               seed, a=561860773102413563, c=0, m=2**60-93):
      self.seed = seed
      self.a = a
      self.c = c
      self.m = m
      self.state = seed

  def next(self):
      self.state = (self.a * self.state + self.c) % self.m
      return self.state / self.m # Normalize to [0, 1)

  def next_in_range(self, a, b):
      return a + (b - a) * self.next()
```

3.1.2.2 Реализация метода Монте-Карло

Теперь реализуем интегрирование методом Монте-Карло, используя ранее описанный ЛКМ (листинг 1):

Листинг 2: Реализация метода Монте-Карло

```
class MonteCarlo:
  def __init__(self, N, PRNG_object):
      self.N = N
      self.PRNG = PRNG_object

  def integrate(self, f, a, b):
      mult = (b - a) / self.N

      generatedValues = []
      for _ in range(self.N):
          randomArg = self.PRNG.next_in_range(a, b)
          randomFuncVal = f(randomArg)

          generatedValues.append(randomFuncVal)
```

```
        return mult * sum(generatedValues)
```

3.1.2.3 Реализация функции распределения

Объединим теперь (2) и (3) и получим:

$$F_X(x) = \int_{-\infty}^{x} f_X(t)dt \approx \frac{x+\infty}{N} \sum_{i=1}^{N} f(u_i), \quad \text{где} \tag{6}$$

u_i ищем в соответствии с (4).

В общем случае пришлось бы производить замену, чтобы свести бесконечные пределы в конечные, однако в нашем случае это не требуется, тк (1) определена при $x > 0$. Подставляя (1) в (6) и (5) в (4):

$$F_X(x) = \int_{0}^{t} \frac{1}{\sqrt{0.4\pi}t} e^{-(\ln t - 2)^2/0.4} dx \approx \frac{x}{N} \sum_{i=1}^{N} \frac{1}{\sqrt{0.4\pi}u_i} e^{-(\ln u_i - 2)^2/0.4}, \quad \text{где} \tag{7}$$

$u_i = (561860773102413563 \cdot u_{i-1}) \bmod 2^{60} - 93$

При программной реализации нас не сильно интересуют конкретные начальное значение в ЛКМ и значение N в методе Монте-Карло.
Первое выбирается так, чтобы $x_0 \neq 0$. Это необходимо для того, чтобы последовательность была полной длины, т.е. имела максимальную периодичность при генерации чисел. Обычно используют случайное или произвольно выбранное значение из множества $\{1, ..., m-1\}$ [1].
Второе, как уже было сказано ранее, отвечает за точность полученной оценки метода, так что чем оно больше, тем лучше.

Листинг 3: Реализация функции распределения

```
if __name__ == '__main__':
    lcg        = LCG(seed=42)
    monteCarlo = MonteCarlo(100000000, lcg)

    def pdf(x): # f_X
        return 1 / (math.sqrt(0.4 * math.pi) * x) \
            * math.exp(-(math.log(x) - 2)**2 / 0.4)

    def cdf(x): # F_X
        return monteCarlo.integrate(pdf, 0, x)
```

где классы **LCG** и **MonteCarlo** представлены в листингах 1 и 2 соответственно.

3.1.3 Обратная функция

3.1.3.1 Метод Ньютона

Для нахождения обратной функции воспользуемся методом касательных (Ньютона).
Рабочая формула

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Вообще говоря, метод используется для нахождения корня заданной функции. Так что для нахождения обратной функции $y = f(x)$, т.е. $x = f^{-1}(y)$ будем искать решение уравнения: $f(x) - y = 0$

$$x_{n+1} = x_n - \frac{f(x_n) - y}{(f(x_n) - y)'_x} = x_n - \frac{f(x_n) - y}{f'(x_n)} \tag{8}$$

Погрешность ε возьмем равной 0.0001.

3.1.3.2 Метод центральных разностей

Производные будем искать методом центральных разностей.
Рабочая формула

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{9}$$

Погрешность определяется как $O(h)$, h примем равной 1e-6.

Подставив (9) в (8), получим:

$$x_{n+1} = x_n - \frac{(f(x_n) - y) \cdot 2h}{f(x_n + h) - f(x_n - h)} \tag{10}$$

4 Список использованных источников

1. L'Ecuyer, Pierre (January 1999). "Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure C. 256