БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №3

Выполнила: Анейчик Ольга
4 курс 2 группа
Преподаватель: Кирлица В.П.

Минск 2018

1. Постановка задачи

Вычислить интеграл, используя метод Монте-Карло:

$$I = \int_0^1 xe^x dx$$

Сравнить полученную оценку с оценкой полученной по методу выделения главной части ($h(x) = x^2$). Сравнить дисперсии этих оценок (с помощью аналитических выкладок).

2. Формулы и краткие пояснения к ним

Метод Монте-Карло

Пусть требуется вычислить определённый интеграл $\int_a^b f(x)dx$.

Рассмотрим случайную величину u, равномерно распределённую на отрезке интегрирования $[a; b]$. Тогда $f(u)$ также будет случайной величиной, причём её математическое ожидание выражается как

$$E\{f(u)\} = \int_a^b f(x)\varphi(x)dx \ = \frac{1}{b-a}\int_a^b f(x)dx$$

Тогда искомый интеграл выражается как

$$\int_a^b f(x)dx = (b-a) * E\{f(u)\}$$

Для определения $E\{f(u)\}$ смоделируем N случайных величин u_i, распределенных равномерно на отрезке $[a; b]$ и для каждой u_i вычислим $f(u_i)$. Затем вычислим выборочное среднее: $\frac{1}{N}\sum_{i=1}^{N} f(u_i)$. В итоге получаем оценку интеграла:

$$\int_a^b f(x)dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(u_i)$$

Среднеквадратическая ошибка интегрирования методом Монте-Карло σ определяется дисперсией значений подынтегральной функции:

$$D(f) = E\{f^2\} - (E\{f\})^2$$

$$\sigma(I) = \sqrt{\frac{D(f)}{N}}$$

Метод главной части

Пусть требуется вычислить интеграл $I = \int_a^b f(x)dx$. Введем в рассмотрение случайную величину X, распределенную равномерно на интервале (a, b). Допустим, что найдена такая функция $h(x)$, которая мало отличается от $f(x)$ и интеграл от которой можно вычислить, не прибегая к методу Монте-Карло. Тогда значение интеграла I равно:

$$I = \ F(X) = (b-a)[f(X) - h(X)] + \int_a^b h(x)dx$$

Таким образом, в качестве оценки математического ожидания $E\{F(X)\}$, а следовательно и интеграла I, можно принять среднее арифметическое N значений функции F(X):

$$\frac{b-a}{N}\sum_{i=1}^{N}[f(x_i) - h(x_i)] + \int_a^b h(x)dx$$

где x_i – возможные значения X.

3. Выходные данные

Найдем точное значение интеграла. Для этого используем метод интегрирования по частям:

$$I = \int_0^1 xe^x dx = [u = x, dv = e^x, du = dx, v = e^x] = (xe^x)|_1 - (xe^x)|_0 - \int_0^1 e^x dx = e - e + 1 = 1$$

$$D(f) = (\int_0^1 (xe^x)^2 dx - \left(\int_0^1 xe^x dx\right)^2)/N = [\text{первый интеграл вычисляется по частям}] = (\frac{e^2}{4} - \frac{1}{4} - 1)/N \approx 0.6/N$$

$$D(f-h) = (\int_0^1 (xe^x - x^2)^2 dx - \frac{\left(\int_0^1 (xe^x - x^2)dx\right)^2}{N} = \frac{\left(\frac{e^2}{4} + 4e - \frac{241}{20}\right) - \frac{2}{3}}{N} \approx 0.23/N$$

Результаты вычислений, N – количество моделируемых СВ для вычисления выборочного среднего:

N	10	50	100	500	1000
Monte Carlo	0.94568	1.00805	0.88137	0.98274	0.96630
Main part	1.19359	0.92689	1.01345	1.00425	1.00334

4. Листинг

```python
# utils.py

import random as r
A = C = 2 ** 15 + 1
M = 2 ** 32 + 1


def linear_congruential_generator(n, a=A, c=C, m=M):
    for i in range(n):
        a = (c * a) % m
        yield a / m


basic_random_values = list(linear_congruential_generator(9999))


def basic_rv():
    return basic_random_values[r.randint(0, 9990)]


def continuous_uniform_rv(a, b):
    return a + (b - a) * basic_rv()



# main.py

from utils import *
from tabulate import tabulate
from math import exp
from numpy import var


a = 0
b = 1
results = []
true_value = 1
```

```python
def f(x):
    return x * exp(x)


def h(x):
    return x ** 2


def integrate_h(t, p):
    return (p ** 3 - t ** 3) / 3


def monte_carlo(n):
    avg = sum([f(continuous_uniform_rv(a, b)) for _ in range(0, n)]) / n
    return (b - a) * avg


def monte_carlo_main_part(n):
    avg = 0
    for _ in range(0, n):
        x_i = continuous_uniform_rv(a, b)
        avg += f(x_i) - h(x_i)
    avg /= n

    return (b - a) * avg + integrate_h(a, b)


N_values = [10, 50, 100, 500, 1000]
with open('output.txt', 'w') as output:

    mcs = [monte_carlo(N) for N in N_values]
    mcs_variance = var(mcs)
    main_parts = [monte_carlo_main_part(N) for N in N_values]
    main_parts_variance = var(main_parts)

    table = tabulate([
        ['N'] + N_values,
        ['Monte Carlo'] + mcs,
        ['Main part'] + main_parts
    ])
    output.write(table + '\n')
    output.write(f'Variance(Monte Carlo): {round(mcs_variance, 5)}\n')
    output.write(f'Variance(Main parts): {round(main_parts_variance, 5)}')
```