Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования   
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Отчет

По лабораторной работе №7

**«Вычисление интегралов методом Монте-Карло»**

по дисциплине «Вычислительные алгоритмы»

Студент группы ПИ-02

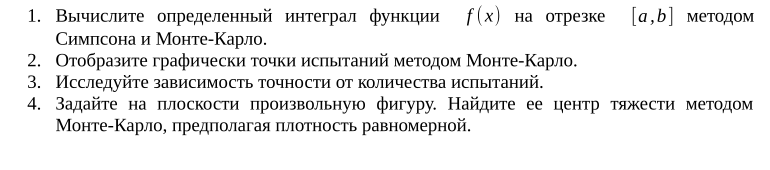
Чередов Р. А.

Преподаватель, к.ф-м.н., доцент,

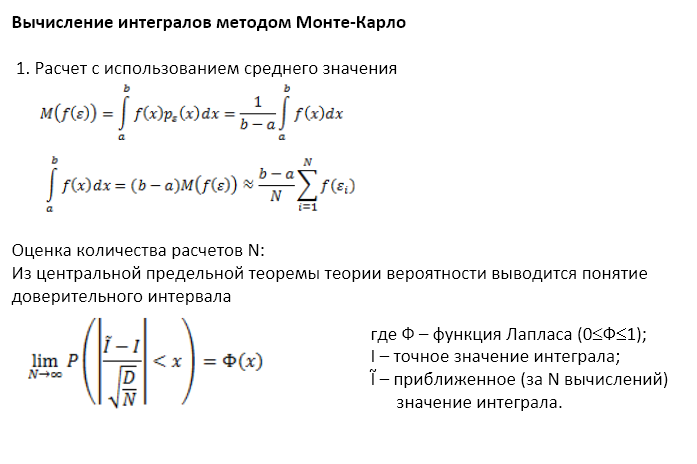
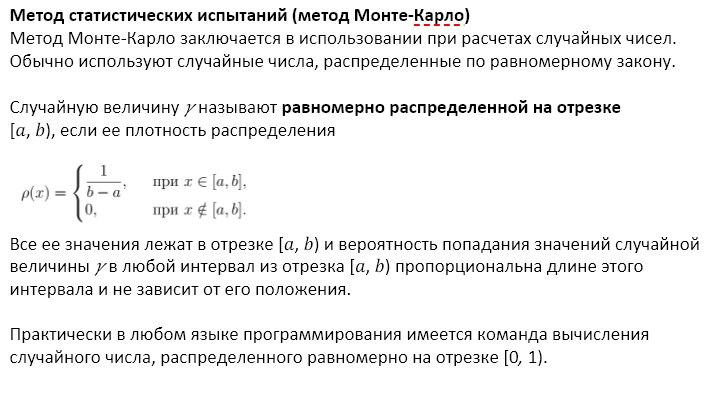
Проскурин А. В.

Барнаул 2023

**Задание к лабораторной работе:**



**Описание метода:**

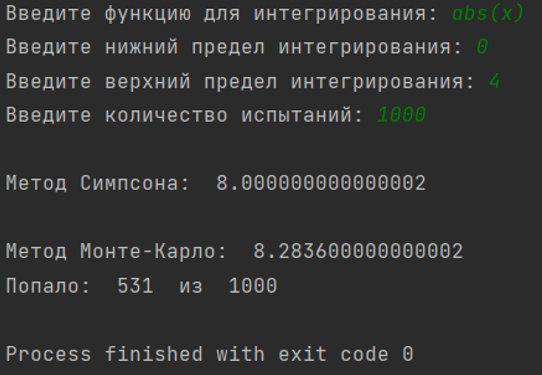
****

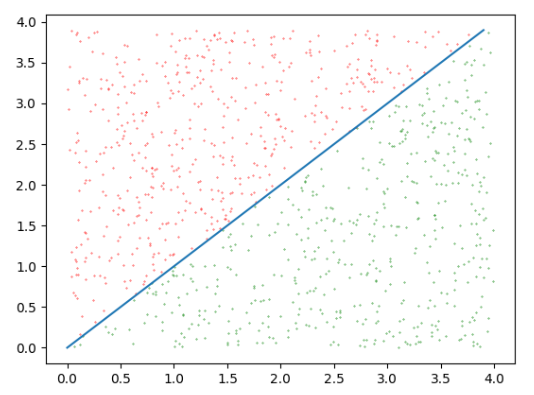
**Программа:**

import random  
import matplotlib.pyplot as plt  
import numpy as np  
from math import \*  
  
def inpXY (a, b, step, f):  
 # Массив значений х на промежутке [a, b + step) с шагом step  
 X = np.arange(a, b, step)  
 Y = []  
 # Массив значений функции f в точках X[]  
 for x in X:  
 Y.append(eval(f))  
 return Y  
  
# Метод Симпсона  
def simpson (a, b, n, f):  
 step = (b - a) / n  
 # Массив значений функции f в точках X[]  
 Y = inpXY(a, b + step, step, f)  
 simp = 0  
 for i in range(0, n):  
 if i == 0 | i == n:  
 simp += Y[i]  
 elif i % 2 == 0:  
 simp += 2 \* Y[i]  
 else:  
 simp += 4 \* Y[i]  
 simp \*= step / 3  
 print("\nМетод Симпсона: ", str(simp))  
  
#Метод Монте-Карло  
def monteCarlo(a, b, n, f, maxY):  
 count = 0  
 for i in range(n):  
 x = random.uniform(a, b)  
 y = random.uniform(0, maxY)  
# Подсчет попавших и отрисовка всех точек  
 if y <= eval(f, {"x": x}):  
 count += 1  
 plt.scatter(x, y, color='green', s=0.1)  
 else:  
 plt.scatter(x, y, color='red', s=0.1)  
 integral = (b - a) \* maxY \* (count / n)  
 print("\nМетод Монте-Карло: ", str(integral),  
 "\nПопало: ", count, " из ", n)  
  
f = input("Введите функцию для интегрирования: ")  
a = float(input("Введите нижний предел интегрирования: "))  
b = float(input("Введите верхний предел интегрирования: "))  
n = int(input("Введите количество испытаний: "))  
simpson(a, b, n, f)  
#Отрисовка введенной функции  
x0 = np.arange(a, b, 0.1)  
y0 = []  
for x in x0:  
 y0.append(eval(f))  
plt.plot(x0, y0)  
# Нахождение максимального у  
maxY = eval(f, {"x": x0[0]})  
for i in range(len(y0)):  
 if maxY < y0[i]:  
 maxY = y0[i]  
monteCarlo(a, b, n, f, maxY)  
plt.show()

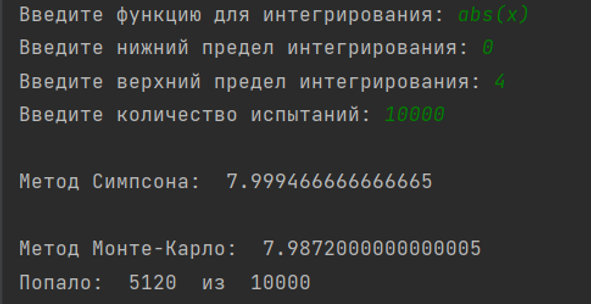
**Тесты:**

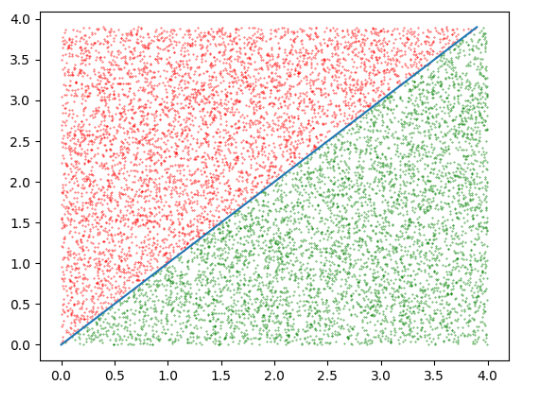
1. Вычисляем интеграл от функции y = |x| при нижнем пределе интегрирования 0 и верхнем 4. Количество испытаний: 1000. Ответ должен быть приближен к 8.



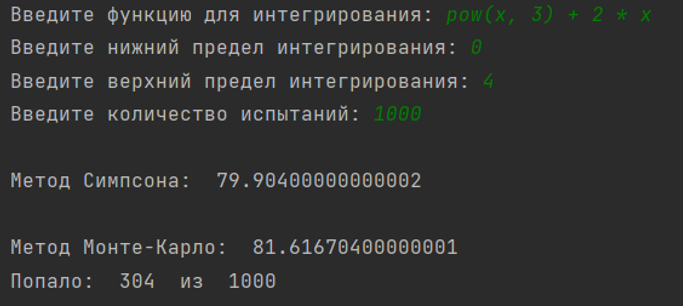
****

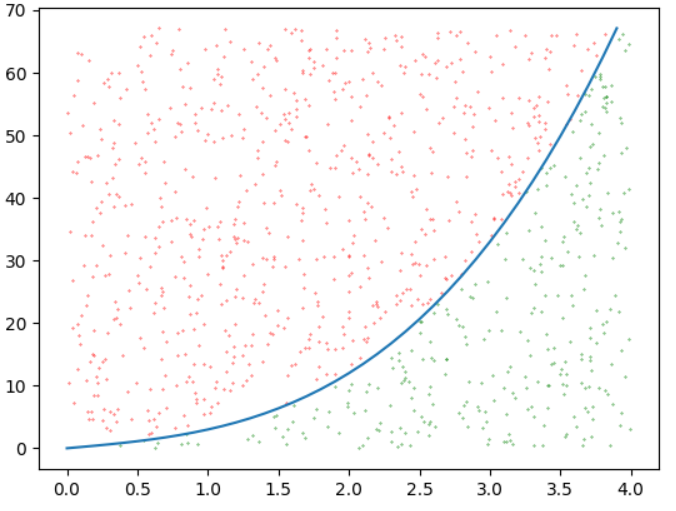
1. Вычисляем интеграл от функции y = |x| при нижнем пределе интегрирования 0 и верхнем 4. Количество испытаний: 10000. Ответ должен быть приближен к 8.



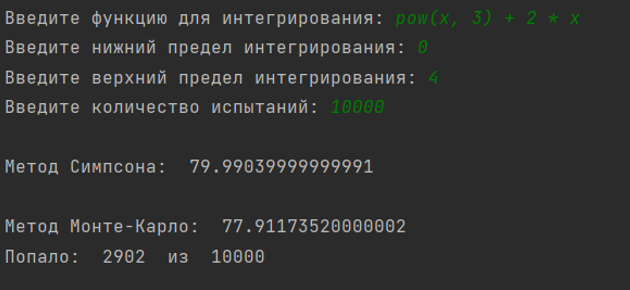
****

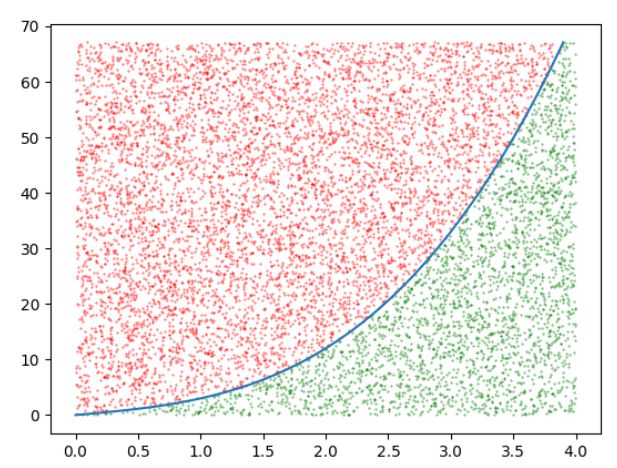
1. Вычисляем интеграл от функции y = x3 + 2x при нижнем пределе интегрирования 0 и верхнем 4. Количество испытаний: 1000. Ответ должен быть приближен к 80.



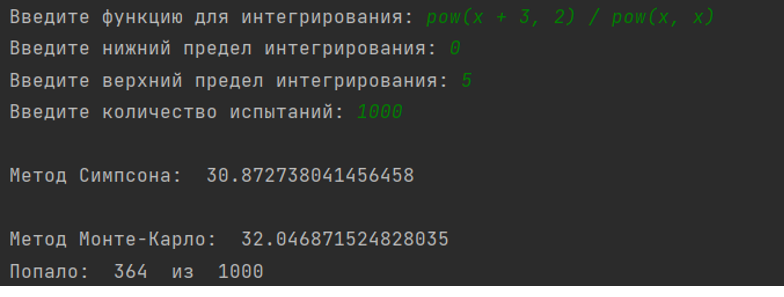
****

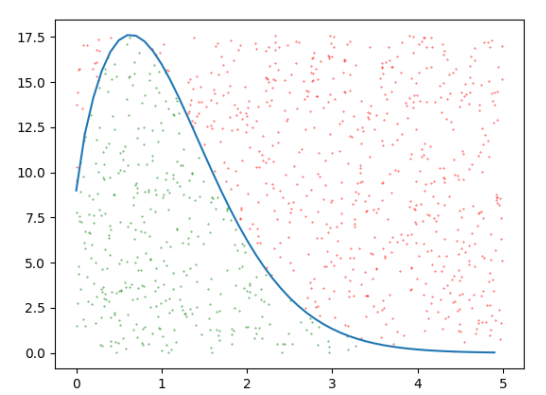
1. Вычисляем интеграл от функции y = x3 + 2x при нижнем пределе интегрирования 0 и верхнем 4. Количество испытаний: 10000. Ответ должен быть приближен к 80.



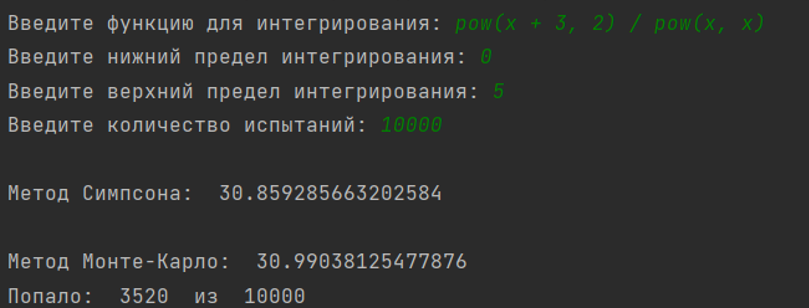
****

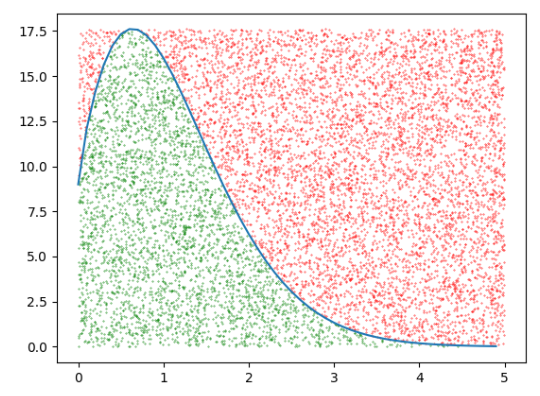
1. Вычисляем интеграл от функции y = (x + 3)2 / xx при нижнем пределе интегрирования 0 и верхнем 5. Количество испытаний: 1000. Ответ должен быть приближен к 30,86.



****

1. Вычисляем интеграл от функции y = (x + 3)2 / xx при нижнем пределе интегрирования 0 и верхнем 5. Количество испытаний: 10000. Ответ должен быть приближен к 30,86.



****

**Вывод:** Исходя из полученных результатов можно сделать вывод, что при увеличении количества испытаний увеличивается точность вычислений. Однако в методе Монте-Карло большую роль играет случай, так как при одном и том же наборе данных получается разный результат, ближе или дальше от истины, что связано с выбором случайных точек при исследовании.