Жуковский Павел, 2 курс, 13 группа, лаба 4

**Лабораторная №4, Вариант 2**

1. По методу Монте-Карло вычислить приближенное значения интегралов.

2. Сравнить полученное значение либо с точным значением (если его

получится вычислить), либо с приближенным, полученным в каком-либо

математическом пакете (например, в mathematica). Для этого построить

график зависимости точности вычисленного методом Монте-Карло

интеграла от числа итераций n.

**Вариант 2**

**1) I1 =**

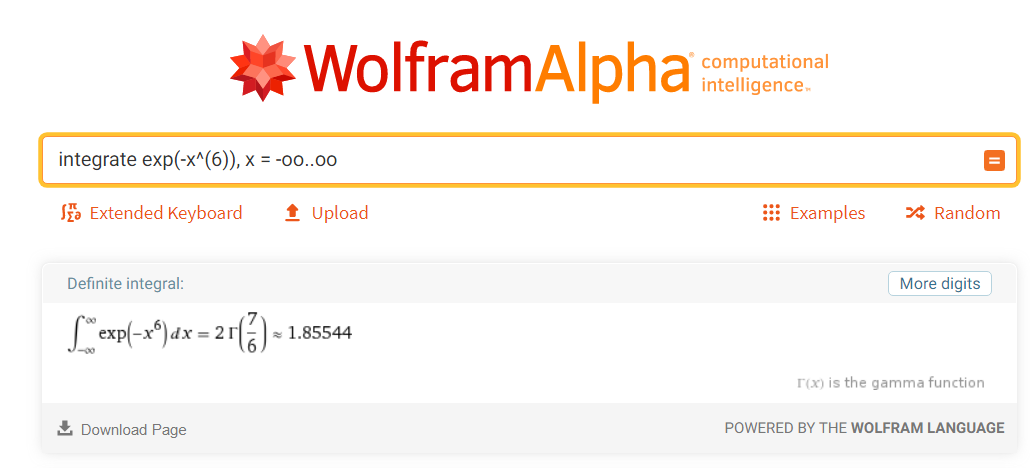
**2) I2 =**

**Небольшое описание программы в виде кода:**

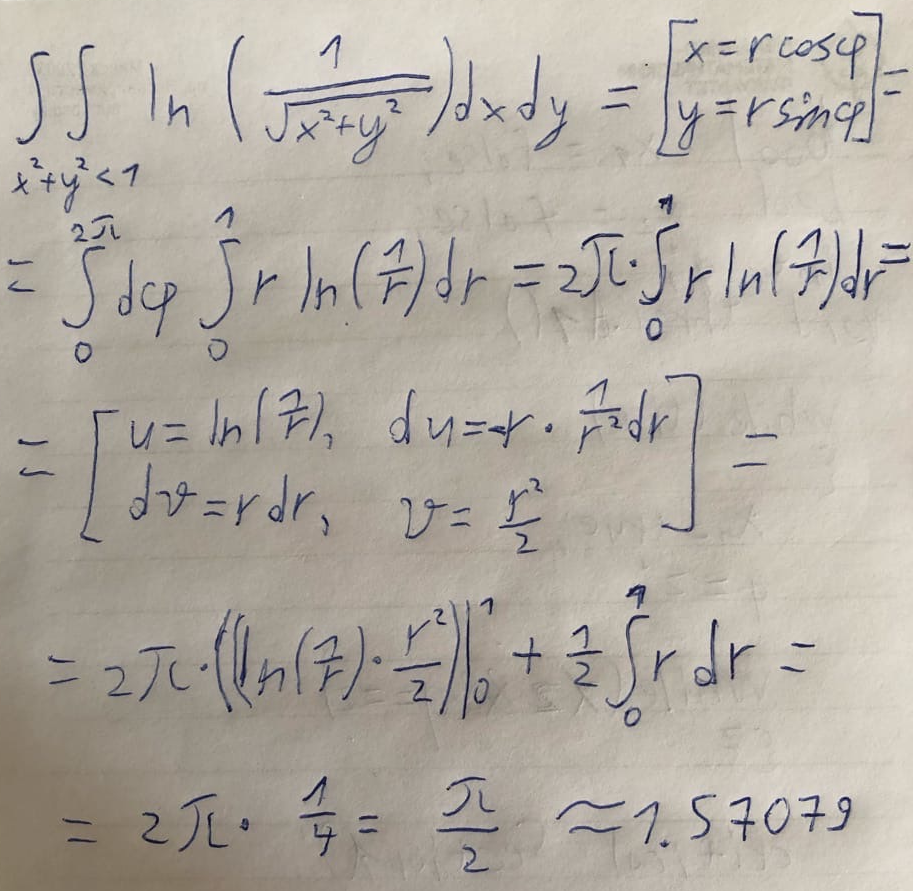
from math import log, sqrt, exp  
from numpy.random import uniform  
import matplotlib.pyplot as plt  
import numpy as np  
  
"""  
Вычислить значение интеграла, используя метод Монте-Карло. Оценить точность.  
1. По методу Монте-Карло вычислить приближенное значения интегралов.  
2. Сравнить полученное значение либо с точным значением (если его  
получится вычислить), либо с приближенным, полученным в каком-либо  
математическом пакете (например, в mathematica). Для этого построить  
график зависимости точности вычисленного методом Монте-Карло  
интеграла от числа итераций n.  
Вариант 2  
I1 = Integral(-oo, oo)\_(e^(-x^6)dx)  
I2 = Integral(x^2 + y^2 < 1)\_((Integral\_(ln(1 / sqrt(x^2 + y^2)))dx)dy)  
"""  
  
size = 6  
  
minus\_infinity = -999  
infinity = 999  
  
ExactFirstResult = 1.85544 # Источник точного ответа: WolframAplha https://www.wolframalpha.com/input/?i=integrate+exp%28-x%5E%286%29%29%2C+x+%3D+-oo..oo  
def CalculateFirstIntegral(n, a, b, sum):  
 for i in range(n):  
 x = a + (b - a) \* uniform()  
 s = exp((-1)\*pow(x, 6))  
 sum += s  
 FirstAnswer = sum \* float(b - a) / float(n)  
 return FirstAnswer  
  
ExactSecondResult = 1.57079 # Источник точного ответа: решение от руки (https://c.radikal.ru/c41/2005/70/f6d4eaecc368.png)  
def CalculateSecondIntegral(n, sum):  
 for i in range(n):  
 x = uniform(-1.0, 1.0)  
 y = uniform(-1.0, 1.0)  
 s = log(1.0 / (sqrt(x\*\*2.0 + y\*\*2.0)))  
 if x\*\*2.0 + y\*\*2.0 < 1.0:  
 sum += s  
 SecondAnswer = sum \* 4.0 / float(n) # Домножаем на 4 по причине двух равномерных распределений с плотностями 1/2 каждая  
 return SecondAnswer  
  
def DrawIntegralsGraphics(n):  
 x = np.arange(100000, n + 1, 100000)  
 FirstPlot = []  
 SecondPlot = []  
 for i in range(100000, n + 1, 100000):  
 FirstPlot.append(abs(CalculateFirstIntegral(i, minus\_infinity, infinity, 0.0) - ExactFirstResult))  
 SecondPlot.append(abs(CalculateSecondIntegral(i, 0.0) - ExactSecondResult))  
 y1 = FirstPlot  
 y2 = SecondPlot  
 plt.plot(x, y1)  
 plt.plot(x, y2)  
 plt.xlabel("Количество итераций (в единицах)")  
 plt.ylabel("Разница в подсчётов")  
 plt.show()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 n = 10\*\*size  
 FirstResult = CalculateFirstIntegral(n, minus\_infinity, infinity, 0.0)  
 print("Расчёт интегралов для количества итераций, равного:", n)  
 print("Расчёт первого интеграла...")  
 print("Приближённое решение первого интеграла программой: I =", FirstResult)  
 print("Точное решение первого интеграла: I =", ExactFirstResult)  
 print("Модуль разницы приближённого и точного решений:", abs(FirstResult - ExactFirstResult))  
 SecondResult = CalculateSecondIntegral(n, 0.0)  
 print("Расчёт второго интеграла...")  
 print("Приближённое решение второго интеграла программой: I =", SecondResult)  
 print("Точное решение второго интеграла: I =", ExactSecondResult)  
 print("Модуль разницы приближённого и точного решений:", abs(SecondResult - ExactSecondResult))  
 print("Расчёт зависимости разницы подсчётов от размерности итераций для графика...")  
 DrawIntegralsGraphics(n)  
 print("График выведен, программа завершила свою работу.")

**Источники точных вычислений**

Точное вычисление для первого интеграла я нашёл с помощью электронного ресурса WolframAlpha (ссылка: <https://www.wolframalpha.com/input/?i=integrate+exp%28-x%5E%286%29%29%2C+x+%3D+-oo..oo>):



Второй интеграл посчитать с помощью WolframAlpha не удалось, поэтому я решил посчитать его в своей тетради методом перехода к полярным координатам:



Таким образом, я получил точные ответы для двух интегралов и сверялся с ними в программе.

**Результат работы программы:**

Время работы программы зависит от выбранного количества итераций, для наглядной и точной демонстрации вычислений я выбрал количество итераций, равное 107, что по времени занимает около часа. Если выбрать количество итераций, скажем, 106 или 105, то программа отработает быстрее, хотя и менеe точно. Здесь я приложу результаты достаточно долгой работы программы на размерности 107:

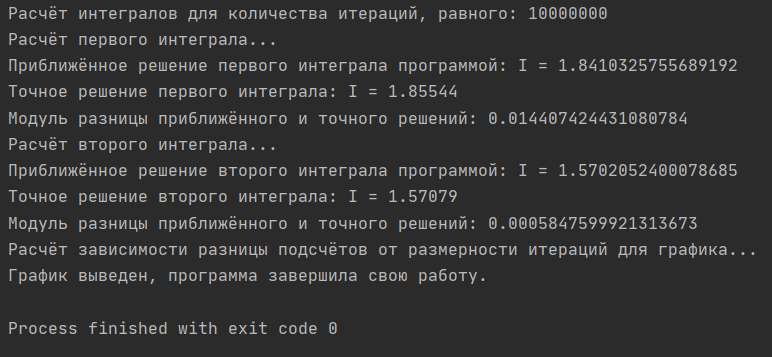


График зависимости разницы точного и приближённого решений от количества итераций для расчёта интегралов (синей линией изображена разница вычислений для первого интеграла, а оранжевой линий – для второго):

