**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования  
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ имени Н.Э.БАУМАНА  
(национальный исследовательский университет)»**

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

**Лабораторная работа № 4**

“Численное интегрирование”

по дисциплине «Численные методы»

Вариант 17

Работу выполнил

студент группы ИУ9-62Б

Сербин Денис

Москва, 2022

# **1. Цель работы**

Целью работы является сравнительный анализ 3х методов численного интегрирования, вычисление интеграла методом Монте-Карло, а так же вычисления интегралов с учетом погрешности.

**2. Постановка задачи**

Дано:

Интеграл где a,b – известные пределы интегрирования, f(x) – известная функция

Тестовый вариант:

Найти:

**3. Индивидуальный вариант**

**4. Теоретические сведения**

**4.1 Метод средних прямоугольников**

Метод заключается в вычислении площади под графиком функции с помощью суммирования площадей конечного числа прямоугольников, где ширина прямоугольника – расстояние между узлами интегрирования, а высота – значение функции в узлах.

, где

**4.2 Метод трапеций**

Метод заключается в замене на каждом отрезке функции на многочлен первой степени.

**4.3 Метод Симпсона**

Метод заключается в приближении функции на отрезке [a, b] интерполяционным многочленом 2 степени

**4.4 Метод Монте-Карло**

Для этого метода нужно взять область интегрирования [a, b], ограничить ее прямоугольником с площадью Sint и набросать в этот прямоугольник случайным образом точек. Затем требуется посчитать количество точек K попавших в область под графиком функции и вычислить интеграл (площадь под кривой S) по формуле:

, при этом чем больше значение N, тем точнее аппроксимация.

**4.5 Вычисление интегралов с учетом погрешности(уточнение Ричардсона)**

, где - вычисленное значение интеграла, k-порядок точности метода, = c, где h – шаг, c – константа.

Для метода средних прямоугольников и трапеций: k = 2

Для метода Симпсона: k = 4

где – уточнение по Ричардсону.

**5. Практическая реализация**

**def** exp\_func(x):  
 **return** math.exp(x)  
  
  
**def** var\_func(x):  
 **return** (math.log(x) \*\* 2) / x  
  
  
**def** rectangle(func, a, b, n):  
 h = (b - a) / n  
 sum = 0  
 **for** i **in** range(0, n):  
 sum += h \* func(a + i \* h + h / 2)  
 **return** sum  
  
  
**def** trapezoid(func, a, b, n):  
 h = (b - a) / n  
 xs = [a + i \* h **for** i **in** range(1, n)]  
 sum = 0  
 **for** i **in** range(0, n - 1):  
 sum += func(xs[i])  
  
 **return** h / 2 \* (func(a) + func(b) + 2 \* sum)  
  
  
**def** simpson(func, a, b, n):  
 h = (b - a) / n  
 xs1 = [a + i \* h - h / 2 **for** i **in** range(1, n + 1)]  
 xs2 = [a + i \* h **for** i **in** range(1, n)]  
 sum = 0  
 **for** i **in** range(0, n):  
 sum += 4 \* func(xs1[i])  
 **if** i < n - 1:  
 sum += 2 \* func(xs2[i])  
 **return** h / 6 \* (func(a) + func(b) + sum)  
  
  
**def** monte\_karlo(func, a, b, n, max, min):  
 k = 0  
 **for** i **in** range(0, n):  
 x = a + random.uniform(0, 1)\*(b-a)  
 y = min + random.uniform(0, 1)\*(max - min)  
 **if** y <= func(x):  
 k += 1  
 integ = (max-min) \*(b-a)\* (k / n)  
 **return** integ  
  
  
**def** monte\_karlo\_solve(func, a, b, integ, epsilon, max, min):  
 k = 2  
 iter = 0  
 mk = monte\_karlo(func, a, b, k, max, min)  
 **while** abs(mk - integ) > epsilon:  
 k \*= 2  
 iter += 1  
 mk = monte\_karlo(func, a, b, k, max, min)  
 print(" Iteration: " + str(iter))  
 print(" Result: " + str(mk))  
 print(" DIFF: " + str(abs(mk - integ)))  
  
  
**def** approx\_richardson(i1, i2, k):  
 **return** (i1 - i2) / (2 \*\* k - 1)  
  
  
**def** calc\_int(a, b, epsilon, method, k, func):  
 print("\n Epsilon: " + str(epsilon))  
 n = 1  
 richardson = float('inf')  
 iter = i\_n = 0  
 **while** abs(richardson) >= epsilon:  
 n \*= 2  
 i\_2 = i\_n  
 i\_n = method(func, a, b, n=n)  
 richardson = approx\_richardson(i\_n, i\_2, k)  
 iter += 1  
 print(" Iteration: " + str(iter))  
 print(" Result: " + str(i\_n))  
 print(" Result with Richardson " + str(i\_n + richardson))  
 print(" DIFF: " + str(abs(richardson)))  
  
  
**if** \_\_name\_\_ == '\_\_main\_\_':  
 epsilon = 0.001  
 print("\nExp:" + str(exp\_func(1) - exp\_func(0)))  
 print("\nTrapezoid:")  
 calc\_int(0, 1, epsilon, trapezoid, 2, exp\_func)  
 print("\nRectangle:")  
 calc\_int(0, 1, epsilon, rectangle, 2, exp\_func)  
 print("\nSimpson:")  
 calc\_int(0, 1, epsilon, simpson, 4, exp\_func)  
 print("\nMonte-Karlo:")  
 monte\_karlo\_solve(exp\_func, 0, 1, exp\_func(1) - exp\_func(0), epsilon, exp\_func(1), 0)  
 l = 1 / exp\_func(1)  
 r = exp\_func(1)  
  
 print("\nTrapezoid:")  
 calc\_int(l, r, epsilon, trapezoid, 2, var\_func)  
 print("\nRectangle:")  
 calc\_int(l, r, epsilon, rectangle, 2, var\_func)  
 print("\nSimpson:")  
 calc\_int(l, r, epsilon, simpson, 4, var\_func)  
 print("\nMonte-Karlo:")  
 monte\_karlo\_solve(var\_func, l, r, 2/3, epsilon, exp\_func(1), 0)

**6. Результаты**

Для тестирования был выбран интеграл: , значение

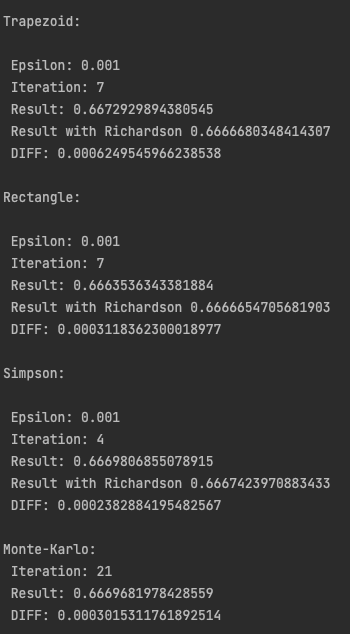


Рис.1 Выходные результаты программы

Таблица 1 Результаты программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод | Количество итераций | Значение без уточнения | Значения с уточнением по Ричардсону | Погрешность |
| Средних треугольников | 7 | 0.6663536343381884 | 0.6666654705681903 | 0.0003118362300018977 |
| Трапеций | 7 | 0.6672929894380545 | 0.6666680348414307 | 0.0006249545966238538 |
| Симпсона | 4 | 0.6669806855078915 | 0.6667423970883433 | 0.0002382884195482567 |
| Монте-Карло | 21 | 0.6669681978428559 | - | 0.0003015311761892514 |

**7. Вывод**

В ходе выполнения лабораторной работы были рассмотрены 4 метода численного интегрирования: метод средних прямоугольников, метод трапеций, метод Симпсона, метод Монте-Карло, и написана реализация на языке программирования Python.

Из всех методов самым точным и самым быстрым оказался метод Симпсона, однако при увеличении количества точек метод Монте-Карло окажется точнее, но заметно медленнее.

При сравнении метода средних прямоугольников и трапеций метод средних прямоугольников оказывается точнее.