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)
«___» _____ 2022 г.

Отчет

по лабораторной работе № 7

"Вычисление интегралов методом Монте-Карло"

по дисциплине
«Вычислительные алгоритмы»

Студенты гр. ПИ-92 Шинтяпин И.И.
Шульпов В.М.

Преподаватель, к.т.н. Проскурин А.В.

Барнаул 2022

Вычисление интегралов методом Монте-Карло

1. Вычислите определенный интеграл функции $f(x)$ на отрезке $[a,b]$ методом Симпсона и Монте-Карло.
2. Отобразите графически точки испытаний методом Монте-Карло.
3. Исследуйте зависимость точности от количества испытаний.
4. Задайте на плоскости произвольную фигуру. Найдите ее центр тяжести методом Монте-Карло, предполагая плотность равномерной.

Алгоритм

$$\int\limits_{x_{i-1}}^{x_i} f(x)\,dx \approx \int\limits_{x_{i-1}}^{x_i} L_{2,i}(x)\,dx = \frac{h}{6}\left(f(x_{i-1}) + 4f(x_{i-1/2}) + f(x_i)\right).$$

Метод Симпсона Формула

Симпсона:

$$\int\limits_a^b f(x)\,dx \approx \sum_{i=1}^{n} \int\limits_{x_{i-1}}^{x_i} f(x)\,dx$$

Метод Монте-Карло

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

N – количество испытаний, u – очередное случайное число принадлежащее отрезку [a,b]Поиск центра тяжести прямоугольного треугольника

Координаты центра тяжести находятся по таким формулам:

$$x_0 = \frac{\iint\limits_D x dxdy}{\iint\limits_D dxdy}, \quad y_0 = \frac{\iint\limits_D y dxdy}{\iint\limits_D dxdy}$$

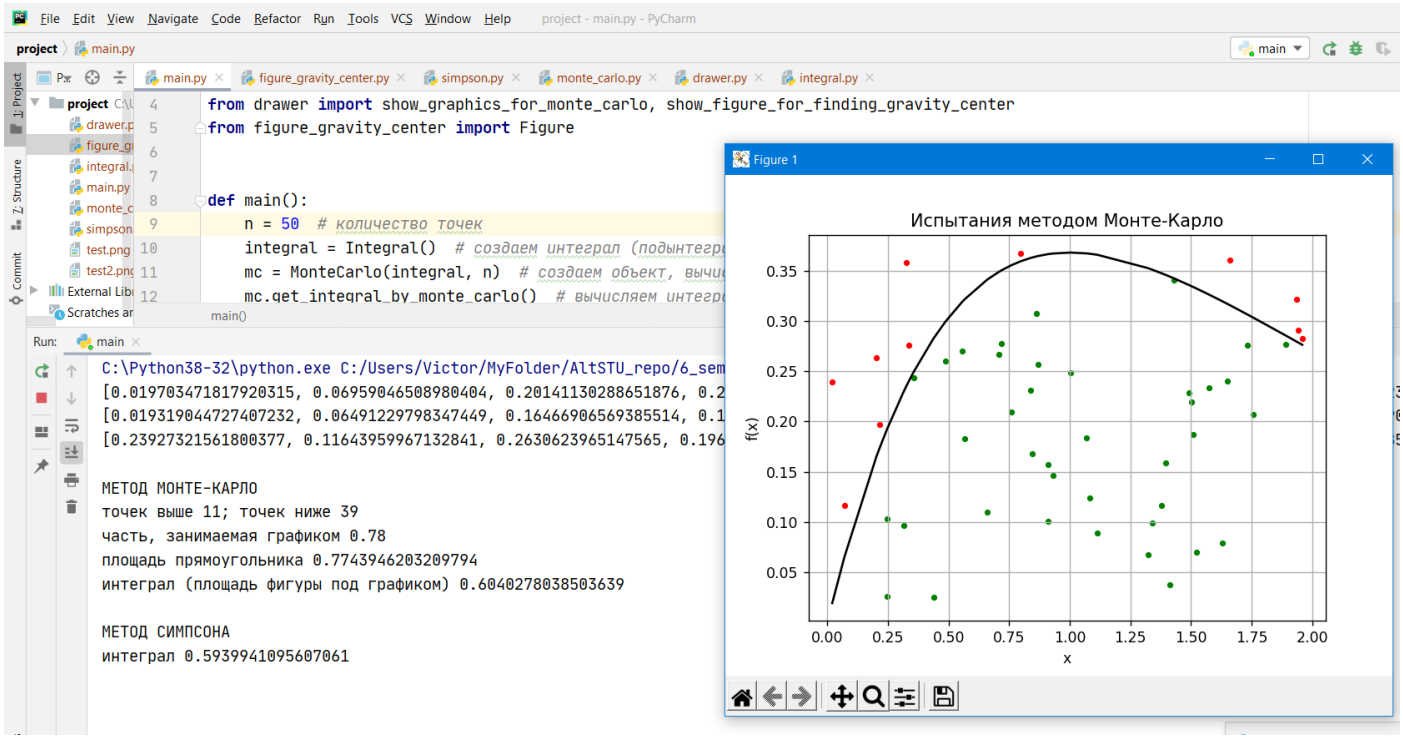
Кратные интегралы находятся по формуле:

$$I \approx J\frac{1}{n}\sum_{i=1}^{n}{}' F(\eta_i).$$

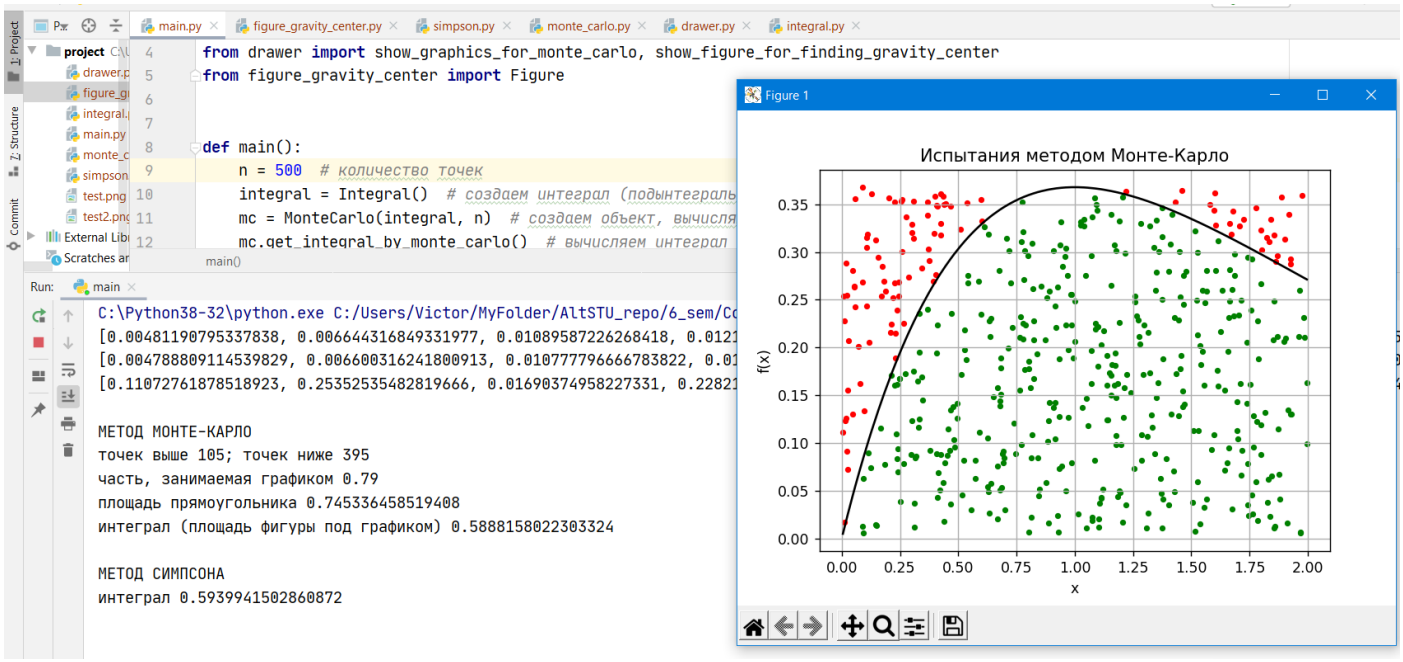
$$\sum_{i=1}^{n}{}' F(\eta_i).$$

n – количество испытаний, ni – случайная точка принадлежащая параллелепипедусодержащему треугольник.
В эту сумму входят только те ni которые принадлежат области фигуры.

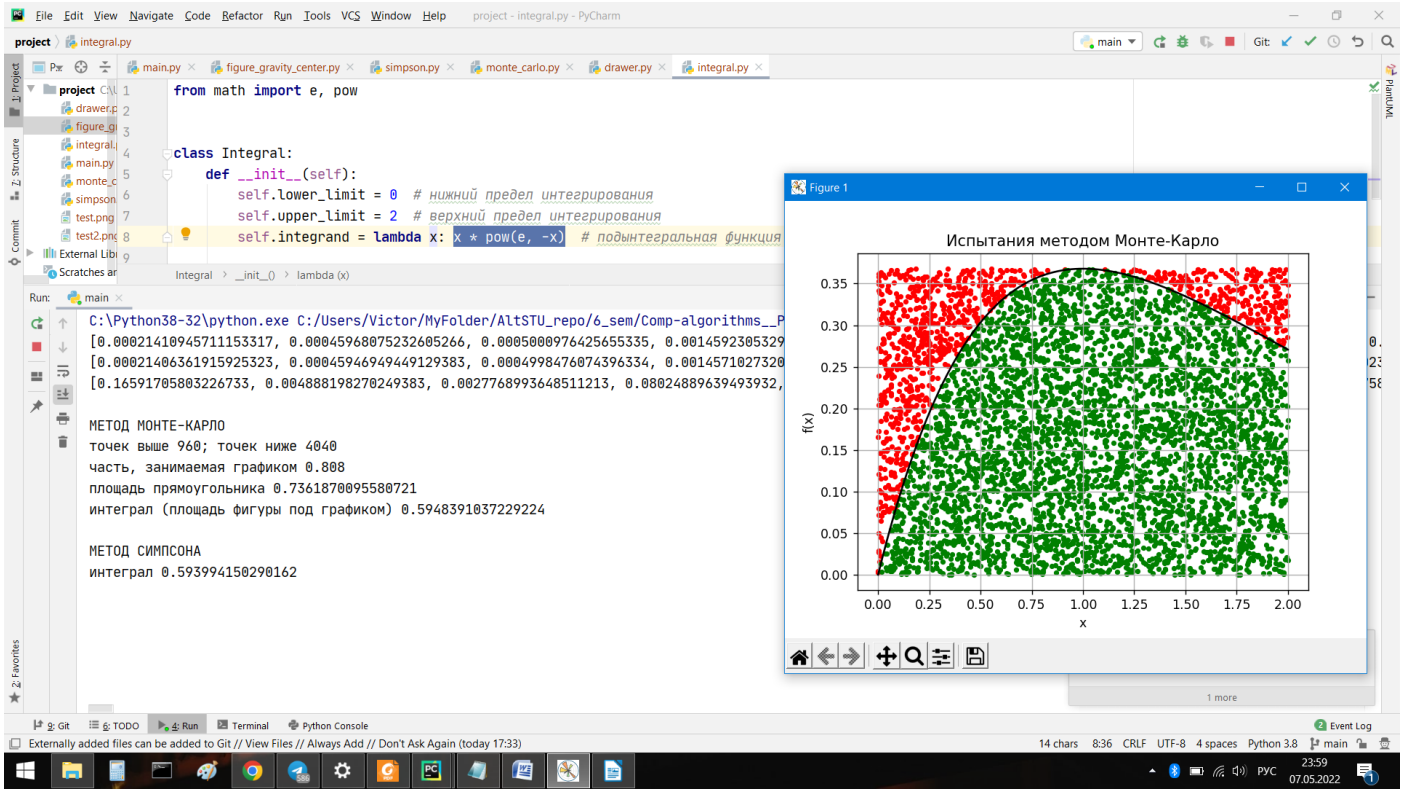
Тесты:
y=x*e^(-x)
интегрирование от 0 до 2
50 точек



500 точек



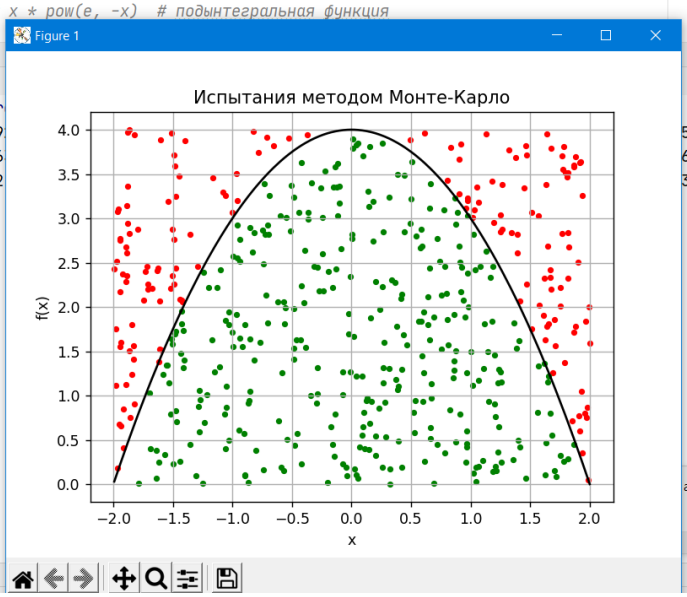
5000 точек


File Edit View Navigate Code Refactor Run Tools VCS Window Help
project - integral.py - PyCharm
main.py
figure_gravity_center.py
simpson.py
monte_carlo.py
drawer.py
integral.py
from math import e, pow
class Integral:
def __init__(self):
self.lower_limit = 0 # нижний предел интегрирования
self.upper_limit = 2 # верхний предел интегрирования
self.integrand = lambda x: x * pow(e, -x) # подынтегральная функция
Figure 1
Испытания методом Монте-Карло
f(x)
x
C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU_repo/6_sem/Comp-algorithms__P
МЕТОД МОНТЕ-КАРЛО
точек выше 960; точек ниже 4040
часть, занимаемая графиком 0.808
площадь прямоугольника 0.7361870095580721
интеграл (площадь фигуры под графиком) 0.5948391037229224
МЕТОД СИМПСОНА
интеграл 0.593994150290162

y=-x*x+4
интегрирование от -2 до 2
500 точек

```
from math import e, pow


class Integral:
    def __init__(self):
        self.lower_limit = -2  # нижний предел интегрирования
        self.upper_limit = 2  # верхний предел интегрирования
        self.integrand = lambda x: -x * x + 4  # x * pow(e, -x)  # подынтегральная функция
```

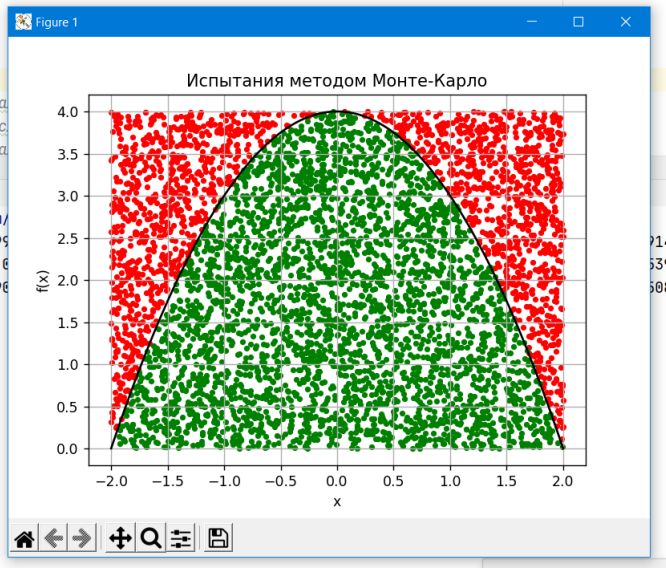
```
C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU_r
[-1.9923247677765725, -1.982800025120413, -1.97757157264999
[0.030642019704026247, 0.06850406038248957, 0.0892106750466
[2.4290231650610035, 1.7506289672685167, 1.1183179420846432

МЕТОД МОНТЕ-КАРЛО
точек выше 164; точек ниже 336
часть, занимаемая графиком 0.672
площадь прямоугольника 16.00455661344683
интеграл (площадь фигуры под графиком) 10.75506204423627

МЕТОД СИМПСОНА
интеграл 10.666666666666664
```



5000 точек

```
from drawer import show_graphics_for_monte_carlo, show_figure_for_finding_gravity_center
from figure_gravity_center import Figure


def main():
    n = 5000  # количество точек
    integral = Integral()  # создаем интеграл (подынтегра
    mc = MonteCarlo(integral, n)  # создаем объект, вычис
    mc.get_integral_by_monte_carlo()  # вычисляем интегра
```

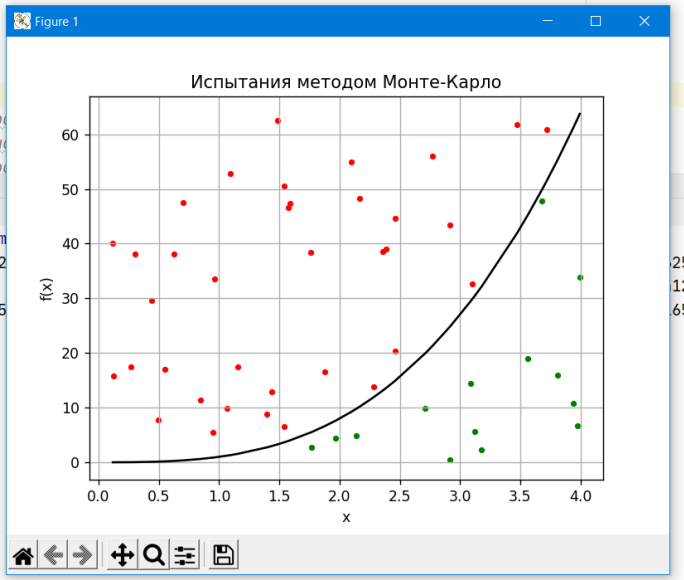
```
C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU_repo/6_sem/
[-1.9996685118510915, -1.9996395385145274, -1.9991550226842079, -1.99
[0.001325842711241254, 0.0014417160094080117, 0.003379195276504543, 0
[2.5726049073249118, 3.986882680387469, 1.489485870323478, 0.3112169

МЕТОД МОНТЕ-КАРЛО
точек выше 1690; точек ниже 3310
часть, занимаемая графиком 0.662
площадь прямоугольника 16.001616621743263
интеграл (площадь фигуры под графиком) 10.59307020359404

МЕТОД СИМПСОНА
интеграл 10.666666666666638
```



Y=x^3
Интегрирование от 0 до 4
50 точек

```
from drawer import show_graphics_for_monte_carlo, show_figure_for_finding_gravity_center
from figure_gravity_center import Figure


def main():
    n = 50  # количество точек
    integral = Integral()  # создаем интеграл (подынтегр
    mc = MonteCarlo(integral, n)  # создаем объект, вычи
    mc.get_integral_by_monte_carlo()  # вычисляем интегр
```

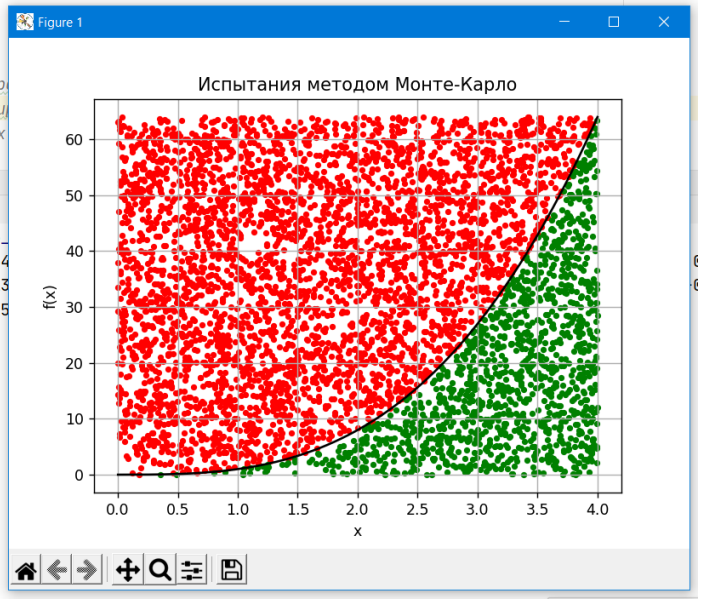
```
C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU_repo/6_sem
[0.11752474688643, 0.1253905236218782, 0.268183572171226, 0.30293562
[0.0016232595759914344, 0.0019714880450960068, 0.01928841376285317,
[40.09265463762774, 15.729033773013105, 17.387612369871345, 38.07775

МЕТОД МОНТЕ-КАРЛО
точек выше 36; точек ниже 14
часть, занимаемая графиком 0.28
площадь прямоугольника 255.01969279207137
интеграл (площадь фигуры под графиком) 71.40551398177999

МЕТОД СИМПСОНА
интеграл 64.0
```



5000 точек

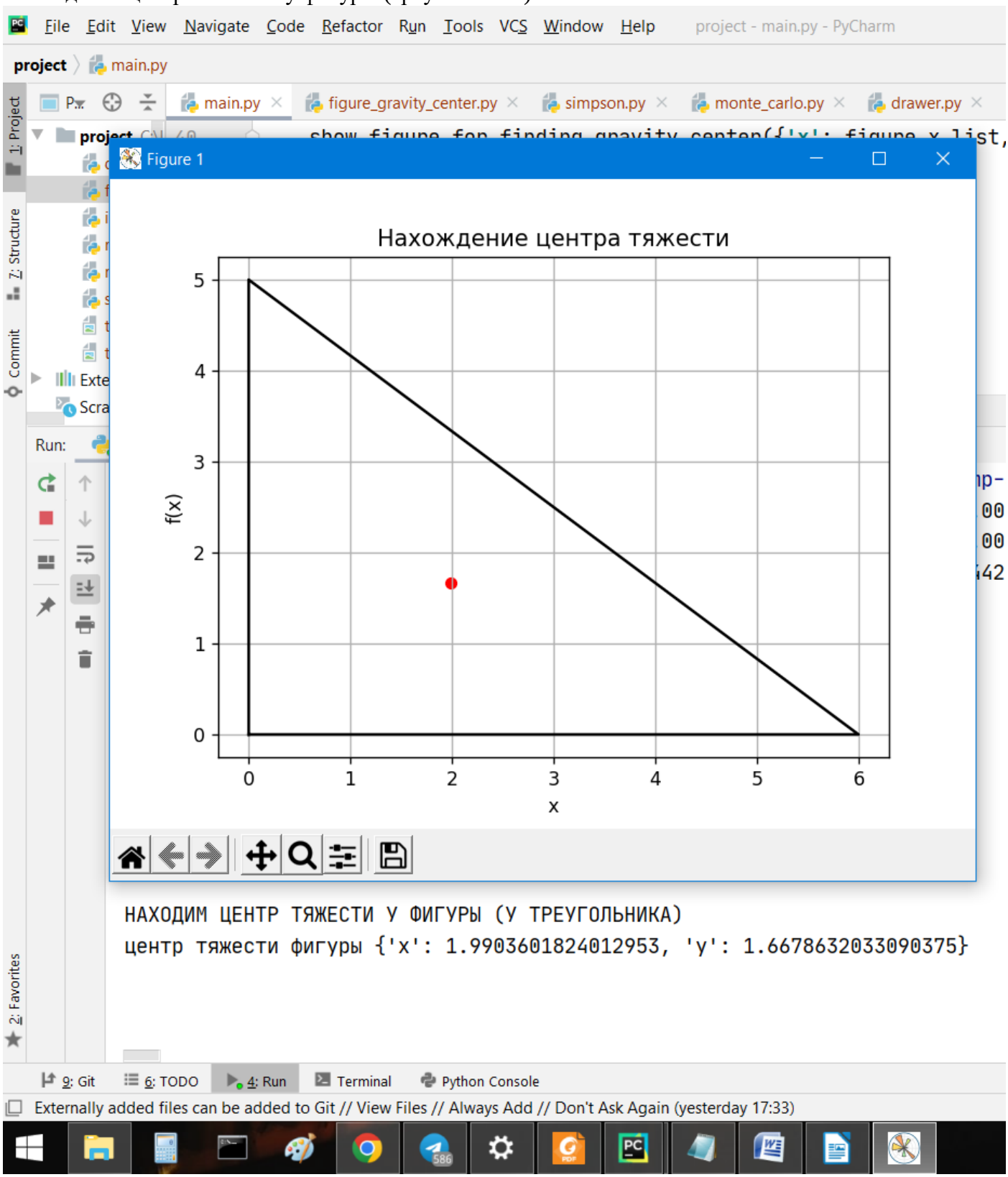
```
from math import e, pow, sin, pi


class Integral:
    def __init__(self):
        self.lower_limit = 0  # нижний предел интегриро
        self.upper_limit = 4  # верхний предел интегри
        self.integrand = lambda x: pow(x, 3)  # -x * x
```

```
C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU_repo/6_
[0.0007596688532891527, 0.0010039665474468329, 0.0015069921923664
[4.384024389636994e-10, 1.0119469052441115e-09, 3.422417648720813
[14.270547872458382, 19.995366693056283, 33.46371312443314, 29.25

МЕТОД МОНТЕ-КАРЛО
точек выше 3759; точек ниже 1241
часть, занимаемая графиком 0.2482
площадь прямоугольника 255.93464977021353
интеграл (площадь фигуры под графиком) 63.522980072966995

МЕТОД СИМПСОНА
интеграл 64.0
```



В методе Монте-Карло при увеличении числа точек, увеличивается точность вычисления интеграла. При маленьком количестве точек, мы можем получить весьма неверный результат, так как точки выбираются случайным образом.

Нахождение центра тяжести у фигуры (треугольника)



Код:

main.py

```
from integral import Integral
from monte_carlo import MonteCarlo
from simpson import Simpson
from drawer import show_graphics_for_monte_carlo, show_figure_for_finding_gravity_center
from figure_gravity_center import Figure


def main():
    n = 50  # количество точек
    integral = Integral()  # создаем интеграл (подынтегральная функция, пределы присваиваются в классе)
    mc = MonteCarlo(integral, n)  # создаем объект, вычисляющий интеграл методом Монте-Карло
    mc.get_integral_by_monte_carlo()  # вычисляем интеграл

    print(mc.x_list)
    print(mc.y_list)
    print(mc.random_y_list)
    print(f'\nМЕТОД МОНТЕ-КАРЛО')
    print(f'точек выше {mc.over_points_num}; точек ниже {mc.under_points_num}')
    print(f'часть, занимаемая графиком {mc.proportion}')
    print(f'площадь прямоугольника {mc.square_area}')
    print(f'интеграл (площадь фигуры под графиком) {mc.integral_value}')

    s = Simpson(integral, n, mc.x_list, mc.y_list)# создаем объект, вычисляющий интеграл методом Симпсона
    s.get_integral_by_simpson()  # вычисляем интеграл

    print(f'\nМЕТОД СИМПСОНА')
    print(f'интеграл {s.integral_value}')

    graphic = {'x': mc.x_list, 'y': mc.y_list}
    under_points = {'x': mc.under_graphic_x_list, 'y': mc.under_graphic_y_list}
    over_points = {'x': mc.over_graphic_x_list, 'y': mc.over_graphic_y_list}
    show_graphics_for_monte_carlo(graphic, under_points, over_points)

    figure = Figure()
    figure.find_gravity_center()

    print(f'\nНАХОДИМ ЦЕНТР ТЯЖЕСТИ У ФИГУРЫ (У ТРЕУГОЛЬНИКА)')
    print(f'центр тяжести фигуры {figure.gravity_center}')

    show_figure_for_finding_gravity_center({'x': figure.x_list, 'y': figure.y_list}, figure.gravity_center)


if __name__ == '__main__':
    main()
```

integral.py

```
from math import e, pow, sin, pi


class Integral:
    def __init__(self):
        self.lower_limit = 0  # нижний предел интегрирования
        self.upper_limit = 4  # верхний предел интегрирования
        self.integrand = lambda x: pow(x, 3)  # -x * x + 4  # x * pow(e, -x)  # подынтегральная функция
```

monte_carlo.py

```python
from random import uniform


class MonteCarlo:
    """ метод Монте-Карло """
    def __init__(self, integral, n):
        self.integral = integral  # пределы интегрирования и подынтегральная функция
        self.n = n  # количество точек
        self.x_list = []  # список x-ов
        self.y_list = []  # список y-ов
        self.min_y = None  # минимальный y из списка y_list
        self.max_y = None  # максимальный y из списка y_list

        self.under_graphic_x_list = []  # координаты x точек находящихся ниже графика
        self.under_graphic_y_list = []  # координаты y точек находящихся ниже графика
        self.over_graphic_x_list = []  # координаты x точек находящихся выше графика
        self.over_graphic_y_list = []  # координаты y точек находящихся выше графика

        self.under_points_num = 0  # кол-во точек, которые ниже графика
        self.over_points_num = 0  # кол-во точек, которые выше графика

        self.square_area = 0  # площадь прямоугольника
        self.proportion = 0  # часть графика, которую он занимает в прямоугольнике
        self.integral_value = 0  # площадь под графиком (значение интеграла)

    def get_integral_by_monte_carlo(self):
        """ вычисление определнного интеграла методом Монте-Карло на отрезке """
        self.__generate_n_x()
        self.__calculate_n_y()
        self.__find_min_max_y()
        self.__generate_n_y()
        self.__get_under_over_points()
        self.__get_square_area()
        self.__get_integral_value()

    def __generate_n_x(self):
        """ генерация координат x для n точек в интервале от lower_limit до upper_limit"""
        self.x_list = [uniform(self.integral.lower_limit, self.integral.upper_limit) for _ in range(0, self.n)]
        self.x_list.sort()

    def __calculate_n_y(self):
        """ подсчет координат y=f(x) для n точек, где f(x) - подынтегральная функция """
        self.y_list = [self.integral.integrand(x) for x in self.x_list]

    def __find_min_max_y(self):
        """ нахождение минимального и максимального y в списке y_list """
        self.min_y = min(self.y_list)
        self.max_y = max(self.y_list)

    def __generate_n_y(self):
        """ генерация координат y для n точек в интервале от min_y до max_y"""
        self.random_y_list = [uniform(self.min_y, self.max_y) for _ in self.x_list]

    def __get_under_over_points(self):
        """ нахождение точек, которые расположены под графиком """
        for rand_y, y, x in zip(self.random_y_list, self.y_list, self.x_list):
            if rand_y < y:
                self.under_graphic_x_list.append(x)
                self.under_graphic_y_list.append(rand_y)
                self.under_points_num += 1
            else:
                self.over_graphic_x_list.append(x)
                self.over_graphic_y_list.append(rand_y)
                self.over_points_num += 1
```

```
    def __get_square_area(self):
        """ подсчет площади прямоугольника, в котором мы генерируем точки """
        x = abs(self.integral.lower_limit) + abs(self.integral.upper_limit)
        y = abs(self.min_y) + (self.max_y)
        self.square_area = x * y

    def __get_integral_value(self):
        """ получение значения интеграла (площади под графиком) """
        self.proportion = self.under_points_num / (self.under_points_num + self.over_points_num)
        self.integral_value = self.square_area * self.proportion
```

simpson.py

```
class Simpson:
    """ метод Симпсона """
    def __init__(self, integral, n, x_list, y_list):
        self.func = integral.integrand  # подынтегральная функция
        self.min_x = integral.lower_limit
        self.max_x = integral.upper_limit  # пределы интегрирования
        self.n = n  # количество точек
        self.x_list = x_list
        self.y_list = y_list

        self.integral_value = 0


    def get_integral_by_simpson(self):
        """ вычисление определнного интеграла методом Симпсона на отрезке """
        h = (self.max_x - self.min_x) / self.n  # шаг
        _sum = self.func(self.min_x) + self.func(self.max_x)
        k = 0
        for i in range(1, self.n):
            k = 2 + 2 * (i % 2)
            _sum += k * self.func(self.min_x + i * h)
        _sum *= h/3
        self.integral_value = _sum
```

figure_gravity_center.py

```
from random import uniform

class Figure:
    """ фигура, у которой находим центр тяжести """
    def __init__(self):
        # треугольник
        self.x_list = [0, 0, 6, 0]
        self.y_list = [0, 5, 0, 0]
        self.max_x = max(self.x_list)
        self.min_x = min(self.x_list)
        self.max_y = max(self.y_list)
        self.min_y = min(self.y_list)
        self.tries_num = 100000  # кол-во испытаний
        self.gravity_center = {'x': 0, 'y': 0}

    def find_gravity_center(self):
        """ нахождение центра тяжести """
        upx = upy = dow = 0  # upx - числитель в формуле поиска x центра тяжести, upy - то же самое, но для y,
        # dow - знаменатель для обеих формул
        for i in range(0, self.tries_num):
            x = uniform(self.min_x, self.max_x)
            y = uniform(self.min_y, self.max_y)
```

```
            # проверка того, входит ли очередная точка в область фигуры
            if self.min_y < y < -x * (5/6) + 5:
                upx += x
                upy += y
                dow += 1
        self.gravity_center['x'] = upx/dow
        self.gravity_center['y'] = upy/dow
```

drawer.py

```
import matplotlib.pyplot as plt


def show_graphics_for_monte_carlo(graphic, under_points, over_points):
    """ графическое отображение испытаний методом Монте-Карло """
    fig, ax = plt.subplots()
    ax.plot(graphic['x'], graphic['y'], color='black')
    plt.scatter(over_points['x'], over_points['y'], color='red', s=10)
    plt.scatter(under_points['x'], under_points['y'], color='green', s=10)

    ax.set(xlabel='x', ylabel='f(x)',
           title='Испытания методом Монте-Карло')
    ax.grid()

    fig.savefig("test.png")
    plt.show()


def show_figure_for_finding_gravity_center(figure, gr_center_point):
    """ графическое отображение фигуры и ее центра тяжести """
    fig, ax = plt.subplots()
    ax.plot(figure['x'], figure['y'], color='black')
    plt.scatter(gr_center_point['x'], gr_center_point['y'], color='red', s=30)

    ax.set(xlabel='x', ylabel='f(x)',
           title='Нахождение центра тяжести')
    ax.grid()

    fig.savefig("test2.png")
    plt.show()
```

Вывод: как видно из тестов, метод Симпсона использовать явно предпочтительней, так как он сходится по времени при увеличении числа процессов так же, как и методы Монте-Карло, при этом точность вычисления остается выше и увеличивается стабильно. Для методов Монте-Карло такого эффекта не наблюдается, поэтому их разумно использовать только для быстрой оценки значения интеграла в особых случаях.