Лабораторная работа № 1. Численное дифференцирование и интегрирование.

Выполнили студенты группы М32051 Быстров Денис, Хащук Денис, Максимович Кирилл.

1. **Реализация методов нахождения производной при фиксированном значении шага**

   Разностные производные — это численный метод для нахождения производных функций. Они основаны на приближении производной функции в точке с помощью ее значения в этой точке и значениях функции в соседних точках.

   Правая разностная производная использует значения функции в точке и соседней точке справа от нее для приближенного вычисления производной в этой точке.

   Левая разностная производная использует значения функции в точке и соседней точке слева от нее для приближенного вычисления производной в этой точке.

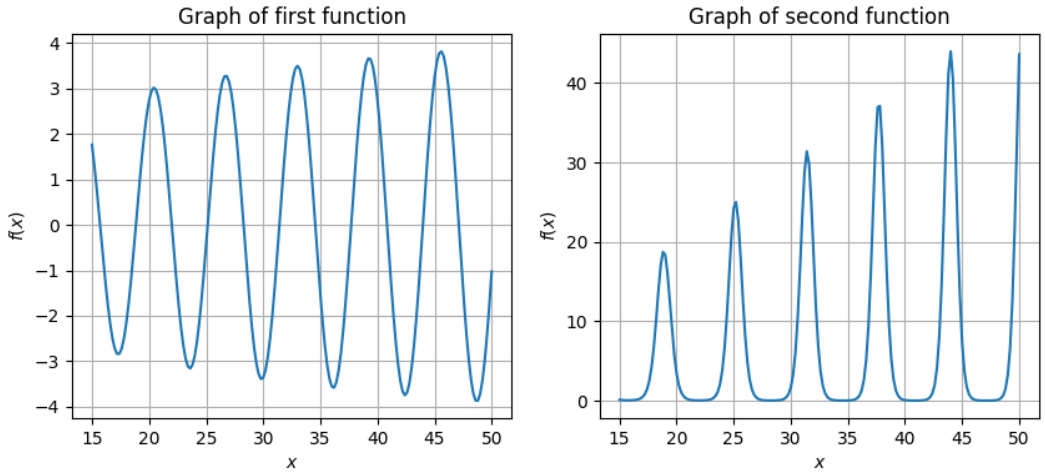
   Центральная разностная производная использует значения функции в точке и ее соседних точках справа и слева от нее для приближенного вычисления производной в этой точке.

   Реализации этих трех методов на Python могут быть найдены на github в репозитории этой лабораторной в файле differentiation.py.

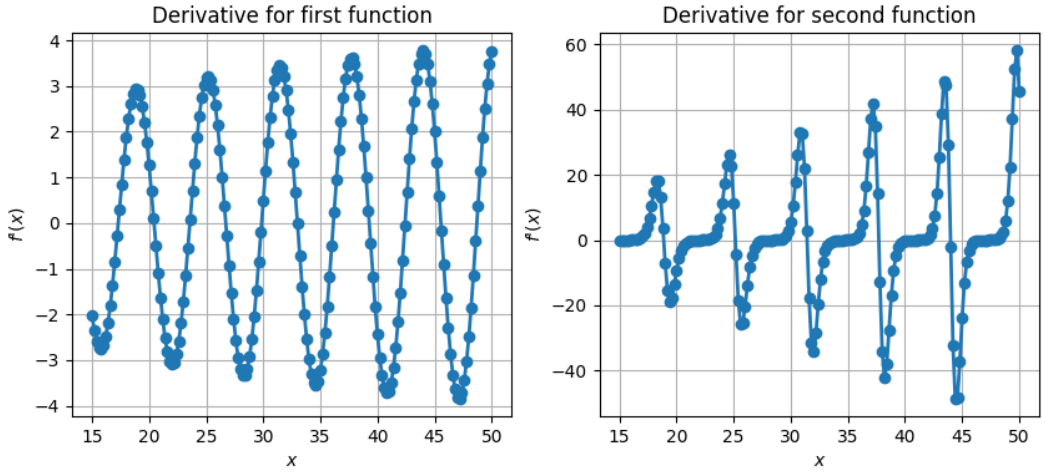
2. **Вычисление производных**

   Теперь возьмем две произвольные функции: $ln(x)sin(x)$ и $exp(cos(x))ln(x)$.

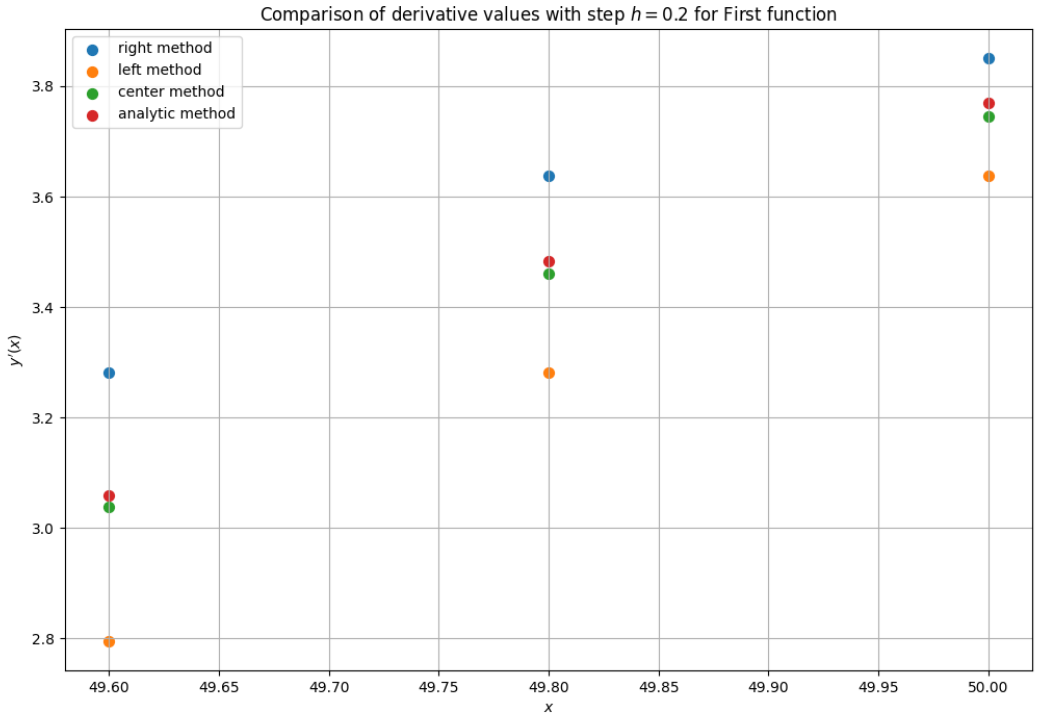
   В коде (файл task2.py в том же репозитории) пропишем методы для вычисления этих функций и их производных, с помощью библиотеки matplotlib построим графики функций.

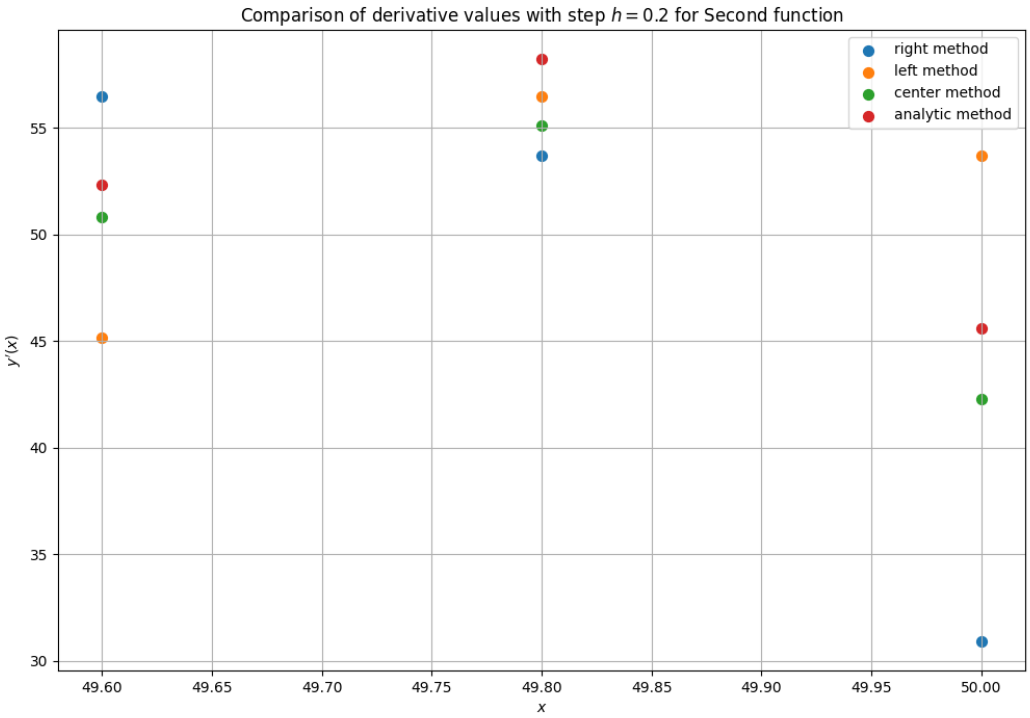


   После этого аналитически вычислим значения производных в узлах сетки и также нанесем их на графики.



Затем найдем значения производных, используя численные методы, описанные выше. Сравнивая последние 3 значения производных в узлах сетки, получим следующие графики:





3. **Поиск среднеквадратичного отклонения полученных результатов**

Сравним численно найденные значения производных с истинными. Для этого найдем среднеквадратичное отклонение, реализацию на Python можно найти в том же репозитории в файле task3.py. Полученные результаты для наших функций:

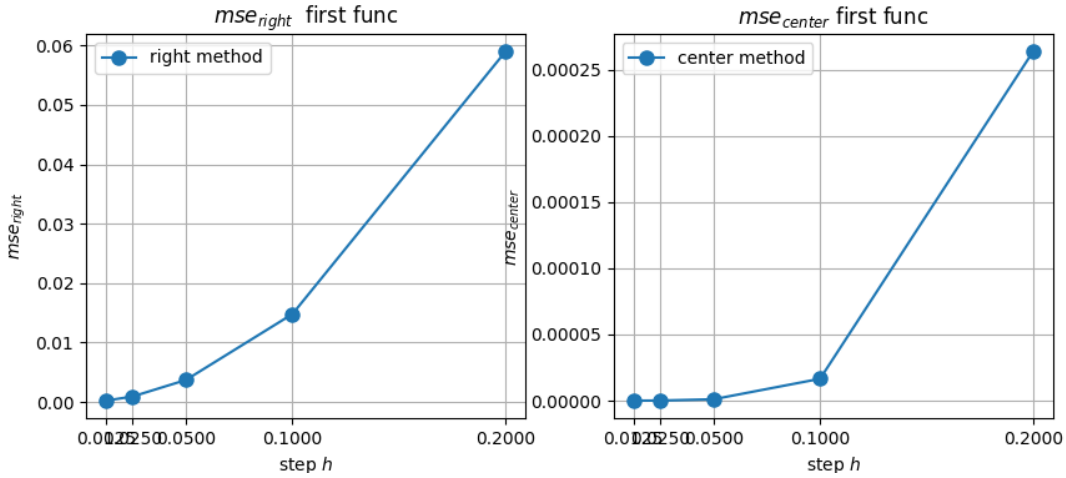
for the first function
left derivatives mse are: 0.0591659829649412
right derivatives mse are: 0.059023920138696354
center derivatives mse are: 0.00026395420589440455

for the second function
left derivatives mse are: 13.638122254859367
right derivatives mse are: 14.615281727954418
center derivatives mse are: 0.631386782051856

4. **Уменьшим шаг в 2, 4, 8 и 16 раз**

В файле task4.py репозитория этой лабораторной реализуем метод, позволяющий получать СКО функций, представленные в форме массивов. Затем вычислим СКО для значений шага, уменьшенных соответственно в 2, 4, 8 и 16 раз.

Для метода правой и центральной разностной производной для первой функции получим:



5. Методы численного интегрирования

Численные методы интегрирования, такие как формула прямоугольников, формула трапеций и формула Симпсона, позволяют приближенно вычислить значение определенного интеграла от функции на заданном интервале.

Формула прямоугольников использует прямоугольники, построенные на заданных узлах интегрирования, для приближенного вычисления интеграла. При этом значение функции на каждом прямоугольнике приближенно заменяется на значение функции в левой или правой границе прямоугольника.

Формула трапеций использует трапеции, построенные на заданных узлах интегрирования, для приближенного вычисления интеграла. При этом значение функции на каждой трапеции приближенно заменяется на среднее арифметическое значения функции в левой и правой границах трапеции.

Формула Симпсона использует параболы, построенные на заданных узлах интегрирования, для приближенного вычисления интеграла. При этом значение функции на каждой параболе приближенно заменяется на квадратурную формулу Симпсона, которая является интерполяционной формулой на трех точках.

Все эти методы имеют различную точность, зависящую от шага интегрирования и свойств функции. Обычно, формула Симпсона имеет наибольшую точность, но также требует больше вычислительных ресурсов. Формулы прямоугольников и трапеций обычно используются в качестве базовых методов, а формула Симпсона - для повышения точности.

Реализации этих трех методов на Python могут быть найдены на github в репозитории этой лабораторной в файле task5.py.

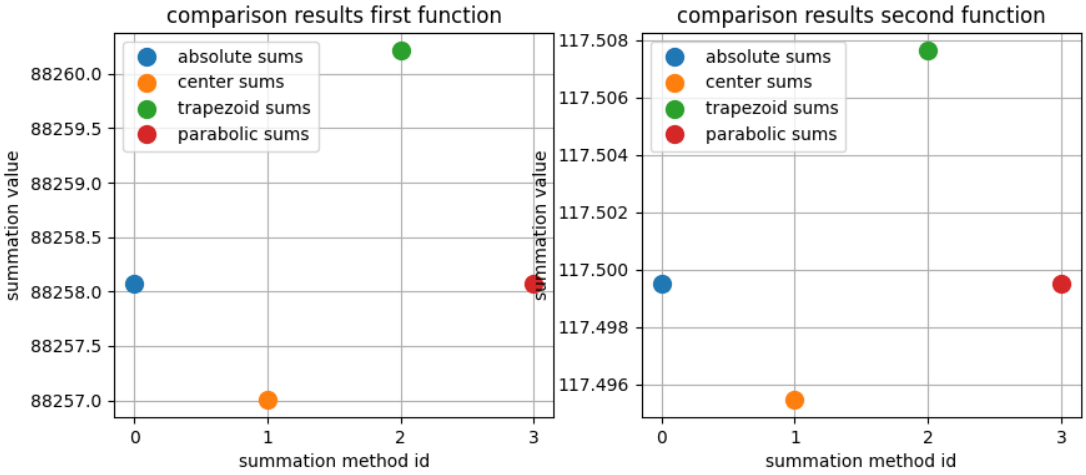
6. Вычисление интегралов

Теперь выберем две произвольных функции: $exp(\sqrt{x})\sqrt{x}$ и $ln(x) + ecos(x)$. В файле task6.py в репозитории этой лабораторной зададим отрезок [15, 50] и аналитически вычислим на нем определенный интеграл для наших функций. Затем вычислим интегралы выше описанными методами и получим следующие результаты:

first function:
absolute sums: 88258.07605804339
center sums: 88257.0065515732
trapezoid sums: 88260.21507506825
parabolic sums: 88258.07605940489

second function:
absolute sums: 117.49952283863921
center sums: 117.49546099558795
trapezoid sums: 117.50764238406241
parabolic sums: 117.49952145841273

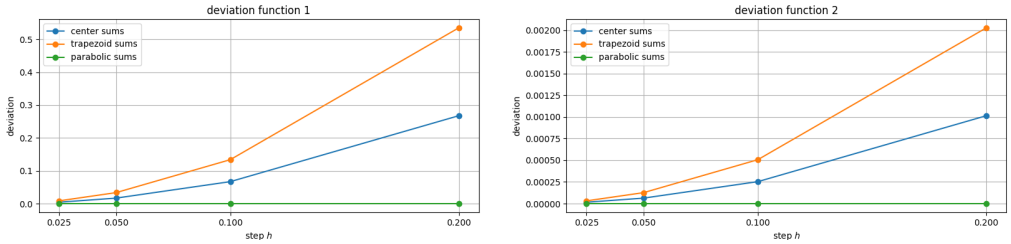
Близкие значения нанесем на графики:



7. Зависимость отклонения численного ответа от аналитического в зависимости от шага при уменьшении его в 2, 4, 8 и 16 раз

Составим массив всех отклонений для каждого метода и для каждого значения шага для обеих функций, после чего построим зависимости отклонения

(посчитанных численными методами значений интегральных сумм от аналитически выведенного значения) от шага разбиения сетки для каждого метода подсчета (реализация [в том же репозитории]() в файле task7.py).



Вывод: В результате проведенной лабораторной работы были реализованы методы численного дифференцирования и интегрирования, а также исследована зависимость отклонения от шага разбиения сетки. Построенный график явно демонстрирует квадратичную зависимость отклонения от шага, что говорит о значительном влиянии выбора метода на точность решения. Было выяснено, что эффективность применения тех или иных методов сильно зависит от характера исследуемой функции. По результатам сравнительного анализа можно сделать вывод, что метод парабол может быть более эффективным при поиске интегральных сумм для некоторых функций, поскольку его ошибка растет с меньшей скоростью, чем ошибка при использовании метода трапеций и метода центральных сумм.