Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

по дисциплине: «Вычислительная математика»

Студент	Платонова Елена Павловна
Группа	РК6-61Б
Тип задания	Лабораторная работа №1 (продвинутая)
Название	«Численное дифференцирование»
Вариант	4

Студент ____________________ **Платонова Е.П.**
подпись, дата *фамилия, и.о.*

Преподаватель ____________________ **Соколов А.П.**
подпись, дата *фамилия, и.о.*

Москва, 2021 г.

Оглавление

Задание на лабораторную работу

Даны функции

$$f(x) = \frac{x^5 + 2x^4 - 3x^3 + 4x^2 - 5}{x + 2}$$

$$g(x) = x^2 \sin(x)$$

и узел $x_0 = 2$.

Требуется (базовая часть):

1. Вывести улучшенную формулу численного дифференцирования для нахождения значения первой производной функции, используя в качестве основной формулы формулу численного дифференцирования 1-го порядка и применив к ней экстраполяцию Ричардсона. Экстраполяция Ричардсона основывается на том, что аппроксимацию значения производной можно записать в следующем виде:
$$M = N_1(h) + K_1 h + K_2 h^2 + K_3 h^3 + \cdots,$$
где M точное значение производной в заданной точке, $N_1(h)$ - формула численного дифференцирования первого порядка и K_1, K_2, K_3 … – константы, появляющиеся в остаточном члене при разложении его в ряд Тейлора. Используя $\frac{h}{2}$ вместо h в выражении выше, выведете новую формулу, которая будет иметь второй порядок точности. Можно ли получить эту формулу другими способами? Если да, то какими?
2. Написать функцию $diff1(x_0, h, f)$, которая возвращает значение первой производной функции на основе центральной формулы численного дифференцирования 1-го порядка в точке для шага дифференцирования h.
3. Написать функцию $diff2(x_0, h, f)$, которая возвращает значение первой производной функции на основе новой формулы

численного дифференцирования 2-го порядка в точке x_0 для шага дифференцирования h.

4. Рассчитать производную $g'(x)$ в точке x_0 для множества значений $h \in [10^{-16}; 1]$ сначала с помощью функции $diff1$, а затем с помощью функции $diff2$. Для обоих случаев постройте $log - log$ графики зависимости абсолютной погрешности численного дифференцирования от шага дифференцирования.

Требуется (продвинутая часть):

1. Для случая функций diff1 и diff2 из базовой части ответить на следующие вопросы:
   - Каким образом на log-log графике можно увидеть порядок формулы дифференцирования? Докажите это формульно и продемонстрируйте на графике по аналогии с лекциями.
   - Каков оптимальный шаг дифференцирования, при котором абсолютная погрешность минимальна? С чем связано существование такого минимума? Обоснуйте свой ответ, ссылаясь на данные log-log графика.
2. Реализовать прямой режим автоматического дифференцирования с использованием дуальных чисел:
   - Написать класс $AutoDiffNumber$, использование которого позволяет построить вычислительный граф.
   - Написать функцию $forward\_autodiff(fun\_args)$, вычисляющую значение производной функции, вычислительный граф которой передается в fun_args.
   - Продемонстрировать корректность автоматического дифференцирования на примере функции $f(x)$ и 100 случайных точек $x \in [-1; 1]$ и сравнить полученные значения производных с аналитически полученными значениями.

- Вычислить значения производных в тех же точках, используя функции $diff1$ и $diff2$, и сравнить полученные значения с аналитическими и полученными с помощью автоматического дифференцирования.

Цель выполнения работы

Цель данной лабораторной работы в первую очередь - знакомство с библиотеками $numpy$ и $matplotlib$. Применение методов численного дифференцирования на практике для анализа погрешностей, возникающих при переходе от дискретной модели к компьютерной.

Убедиться в вычислительной неустойчивости численных методов дифференцирования.

Реализовать прямой режим автоматического дифференцирования и сравнить результаты, полученные с помощью двух подходов вычисления производной (численного и автоматического дифференцирования), проанализировать, какой из них лучше и сделать выводы.

Базовая часть

Вывод улучшенной формулы численного дифференцирования

Экстраполяцию Ричардсона можно рассматривать как общую процедуру повышения точности аппроксимации, когда известна погрешность метода.

Покажем, как преобразовать формулу численного дифференцирования 1-го порядка с остаточным членом первого порядка точности в приближение со вторым порядком точности той же величины. Мы уже знаем, что можем написать приближение $f'(x)$ первого порядка точности с учетом значений функции в $f(x_0)$ и $f(x_0 + h)$:

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h} \tag{1}$$

Чтобы улучшить аппроксимацию $f'(x)$, воспользуемся разложением функции $f(x)$ в ряд Тейлора в точке x_0.

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)(x - x_0)^2}{2!} + \frac{f'''(x_0)(x - x_0)^3}{3!} + \cdots$$
$$+ \frac{f^n(\xi)(x - x_0)^n}{n!},$$

где $\xi \in (x_0; x)$. Тогда значение ряда в точке $x_0 + h$ будет равно:

$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f'''(x_0).$

Сделав некоторые преобразования и перенеся $hf'(x_0)$ влево от знака равенста, получим:

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{h}{2!}f''(x_0) - \frac{h^2}{3!}f'''(x_0) - \frac{h^3}{4!}f^4(x_0) + \cdots \quad (2)$$

Перепишем (2):

$$M = N_1(h) + K_1h + K_2h^2 + K_3h^3 + \cdots \quad (3)$$

получив тем самым формулу Ричардсона.

Здесь мы обозначили величину M, которую мы аппроксимируем, то есть

$M = f'(x_0),$

И $N_1(h)$ – приближенное значение для первой производной, которое в данном случае имеет вид:

$N_1(h) = \frac{f(x_0+h)-f(x_0)}{h}.$

И значение погрешности

$E = K_1h + K_2h^2 + K_3h^3 + \cdots,$

где K_i – коэффициент при h^i в формуле (2). Можно заметить, что эти коэффициенты K_1, K_2, K_3 ... не зависят от h.

Помним, что формула:

$$M \approx N_1(h)$$

это аппроксимация первой производной первого порядка точности в точках $x_0, x_0 + h$. Пусть $K_i \neq 0$. Тогда чтобы улучшить аппроксимацию M

попробуем исключить член $K_1 h$ из ошибки. Для этого запишем (3) в других точках, пусть это будут точки $x_0,\ x_0+\frac{h}{2}$:

$$M = N_1\left(\frac{h}{2}\right) + K_1\frac{h}{2} + K_2\frac{h^2}{4} + K_3\frac{h^3}{8} + \cdots \qquad (4)$$

Объединим (3) с (4) так, чтобы член с h в ошибке исчез. Действительно, вычитая друг из друга следующие уравнения:

$$\begin{cases} M = N_1\left(\frac{h}{2}\right) + K_1\frac{h}{2} + K_2\frac{h^2}{4} + K_3\frac{h^3}{8} + \cdots \\ \frac{M}{2} = \frac{N_1(h)}{2} + \text{К}_1\frac{h}{2} + K_2\frac{h^2}{2} + K_3\frac{h^3}{2} + \cdots \end{cases}$$

Получаем:

$$M = 2N_1\left(\frac{h}{2}\right) - N_1(h) - K_2\frac{h^2}{2} - K_3\frac{3h^3}{4} + \cdots$$

Обозначим $N_2(h) = 2N_1\left(\frac{h}{2}\right) - N_1(h)$.

Получаем $M - N_2(h) = \sum_{i=2}^{n} K_i h^i \left(\frac{1}{2^{i-1}} - 1\right)$.

Следовательно, $N_2(h)$ это уже аппроксимация производной второго порядка точности.

Запишем итоговую формулу аппроксимации:

$$f'(x_0) = 2\left(\frac{f\left(x_0 + \frac{h}{2}\right) - f(x_0)}{\frac{h}{2}}\right) - \frac{f(x_0+h) - f(x_0)}{h} -$$

$$-K_2\frac{h^2}{2} - K_3\frac{3h^3}{4} - K_4\frac{7h^4}{8} - \cdots$$

После преобразований получим:

$$f'(x_0) = \frac{-3f(x_0) + 4f\left(x_0 + \frac{h}{2}\right) - f(x_0+h)}{h} + O(h^2) \qquad (5)$$

Таким образом, мы получили формулу численного дифференцирования второго порядка точности из формулы численного дифференцирования первого порядка, используя экстраполяцию

Ричардсона. Причем, как было отмечено выше, коэффициенты $K_1, K_2, \dots, K_n$ не зависят от h, что дает нам право использовать экстраполяцию Ричардсона, даже не зная точный вид остаточного члена.

Попробуем получить формулу (5) другими способами.

Для этого воспользуемся интерполяционным многочленом Лагранжа, используя значения функции в трех узлах $x_0 + \frac{h}{2}, x_0, x_0 + h$. Данный метод называется методом дифференцирования многочлена Лагранжа.

$$f(x) = L_2(x) + \frac{f'''(\xi)}{3!}\prod_{i=1}^{3}(x - x_i) = f(x_0)\frac{(x-x_0-h)\left(x-x_0-\frac{h}{2}\right)}{(x_0-x_0-h)\left(x_0-x_0-\frac{h}{2}\right)} +$$

$$f\left(x_0 + \frac{h}{2}\right)\frac{(x-x_0)(x-x_0-h)}{\left(x_0+\frac{h}{2}-x_0-h\right)\left(x_0+\frac{h}{2}-x_0\right)} + f(x_0 + h)\frac{(x-x_0)\left(x-x_0-\frac{h}{2}\right)}{(x_0+h-x_0)\left(x_0+h-x_0-\frac{h}{2}\right)} +$$

$$\frac{f'''(\xi)}{3!}(x - x_0)(x - x_0 - h)\left(x - x_0 - \frac{h}{2}\right).$$

Продифференцируем полученную функцию:

$$f'(x) = f(x_0)\frac{\left(2x - 2x_0 - \frac{3h}{2}\right)}{\frac{h^2}{2}} + f(x_0 + h)\frac{\left(2x - 2x_0 - \frac{h}{2}\right)}{\frac{h^2}{2}}$$

$$- f\left(x_0 + \frac{h}{2}\right)\frac{(2x - 2x_0 - h)}{\frac{h^2}{4}}$$

$$+ \frac{f'''(\xi)}{3!}\left(\frac{h^2}{2} - 3hx + 3hx_0 + 3x^2 - 6x_0x + 3x_0^2\right)$$

Найдем эту производную в точке x_0:

$$f'(x_0) = \frac{-3f(x_0) + 4f\left(x_0 + \frac{h}{2}\right) - f(x_0 + h)}{h} + f'''(\xi)\frac{h^2}{12} \qquad (6)$$

Таким образом, мы получили аппроксимирующую формулу для производной, близкую к формуле (5), также со вторым порядком точности, используя метод дифференцирования многочлена Лагранжа.

Вывели формулу для экстраполяции Ричардсона, используя разложение функций в ряд Тейлора и с помощью всех этих методов пришли к формуле численного дифференцирования с более высокой степенью точности.

Реализация функции diff1

Для начала реализуем код для функции g, которая задана нам по условию:

$$g(x) = x^2 \sin(x)$$

Код для нее на Python представлен в листинге 1.

Листинг 1 – функция, возвращающая значение функции $g(x)$.

```
def g(x):
    return (x ** 2) * np.sin(x)
```

Теперь реализуем функцию $diff1(x_0, h, f)$, которая возвращает значение первой производной на основе формулы:

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}$$

Листинг 2 - функция, возвращающая производную на основе формулы численного дифференцирования 1-го порядка.

```
def diff1(x_0, h, f):
    return (f(h + x_0) - f(x_0)) / h
```

На вход функции $diff1$ мы передаем: x_0 – точку, в которой необходимо найти производную, h - шаг численного дифференцирования и функцию f, которую будем дифференцировать.

Реализация функции diff2

Напишем функцию $diff2(x_0, h, f)$, для нахождения производной в точке x_0 с помощью улучшенной формулы (5), которую мы вывели, только для шага h, а не для шага $\frac{h}{2}$, как в (5).

Теперь улучшенная формула дифференцирования 2 порядка точности будет иметь вид:

$$f'(x_0) = \frac{-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)}{2h}$$

Реализация функции $diff2$ на Python будет иметь следующий вид:

Листинг 3 - функция, возвращающая производную на основе формулы численного дифференцирования 2-го порядка.

```
def diff2(x_0, h, f):
    return (4 * f(x_0+h) - 3 * f(x_0) - f(x_0 + 2 * h)) / (h + h)
```

На вход функции $diff2$ мы передаем: x_0 – точку, в которой необходимо найти производную, h - шаг численного дифференцирования и функцию f, которую будем дифференцировать.

Строим log-log графики

При переходе от математической модели к дискретной (по сути, наше выведенное равенство в формуле (5)) из-за дискретизации математического языка возникает погрешность метода, которую мы также представили в формуле (5). Она равная $O(h^2)$. Далее при переходе уже от дискретной к компьютерной модели у нас возникает вычислительная погрешность из-за аппроксимации вещественных чисел.

Таким образом, при округлении $g(x_0)$, $g(x_0+h)$ и $g(x_0+2h)$ возникают вычислительные погрешности, равные $e(x_0)$, $e(x_0+h)$ и $e(x_0+2h)$. Тогда запишем наши значения функций с учетом вычислительной погрешности:

$g(x_0) = \tilde{g}(x_0) + \ e(x_0),$

$g(x_0+h) = \tilde{g}(x_0+h) + \ e(x_0+h),$

$g(x_0+2h) = \tilde{g}(x_0+2h) + \ e(x_0+2h),$

где $\tilde{g}(x_0)$, $\tilde{g}(x_0+h)$, $\tilde{g}(x_0+2h)$ - значения функций, которые реально хранятся в памяти компьютера после округления.

Для начала найдем точное значение производной $g'(x)$:

$$g'(x) = 2x \cdot sin(x) + x^2 \cos(x)$$

Реализуем функцию $derivative\_g$ на Python для нахождения точного значения производной.

Листинг 4 – функция, возвращающая точное значение производной $g'(x)$.

```
def derivative_g(x):
    return 2 * x * np.sin(x) + (x ** 2) * np.cos(x)
```

Значение этой функции сопоставимо с машинным эпсилон.

Теперь мы можем рассчитать абсолютную погрешность, как разность между точным значением производной в точке x_0 и «компьютерным» значением.

Абсолютная погрешность для 1 метода дифференцирования:

$$E_1 = \left| g'(x_o) - \frac{\tilde{g}(x_0+h) - \tilde{g}(x_0)}{h} \right| = \left| \frac{e(x_0+h) - e(x_0)}{h} + O(h) \right| \qquad (7)$$

Абсолютная погрешность для 2 метода дифференцирования:

$$E_2 = \left| g'(x_o) - \frac{-3\tilde{g}(x_0) + 4\tilde{g}(x_0+h) - \tilde{g}(x_0+2h)}{2h} \right| = \left| \frac{-3e(x_0) + 4e(x_0+h) - e(x_0+2h)}{2h} + O(h^2) \right| \qquad (8)$$

Назовем функцию для получения полной погрешности для первого метода $diff1$, как $error\_diff1$, а для второго метода соответственно - $error\_diff2$. И напишем к ним реализацию на Python, исходя из определения абсолютной погрешности (см. листинг 5).

Листинг 5 – Функции, определяющие значения полных погрешностей двух формул дифференцирования.

```
def error_diff1(x, h):
    return np.abs(derivative_g(x) - diff1(x, h, g))


def error_diff2(x, h):
    return np.abs(derivative_g(x) - diff2(x, h, g))
```

Листинг 6 - log-log графики для функций $error\_diff1$ и $error\_diff2$.

```
diff_1_line = axes.loglog(h, error_diff1(x, h), '-', label='diff1')
diff_2_line = axes.loglog(h, error_diff2(x, h), '-', label='diff2')
```

Шаги $h \in [10^{-16}; 1]$ мы задаем на равномерной логарифмической сетке и для массива этих шагов вычисляем полные погрешности численного дифференцирования $errror\_diff1$ и $error\_diff2$.

В результате получим график, представленный на рисунке ниже:

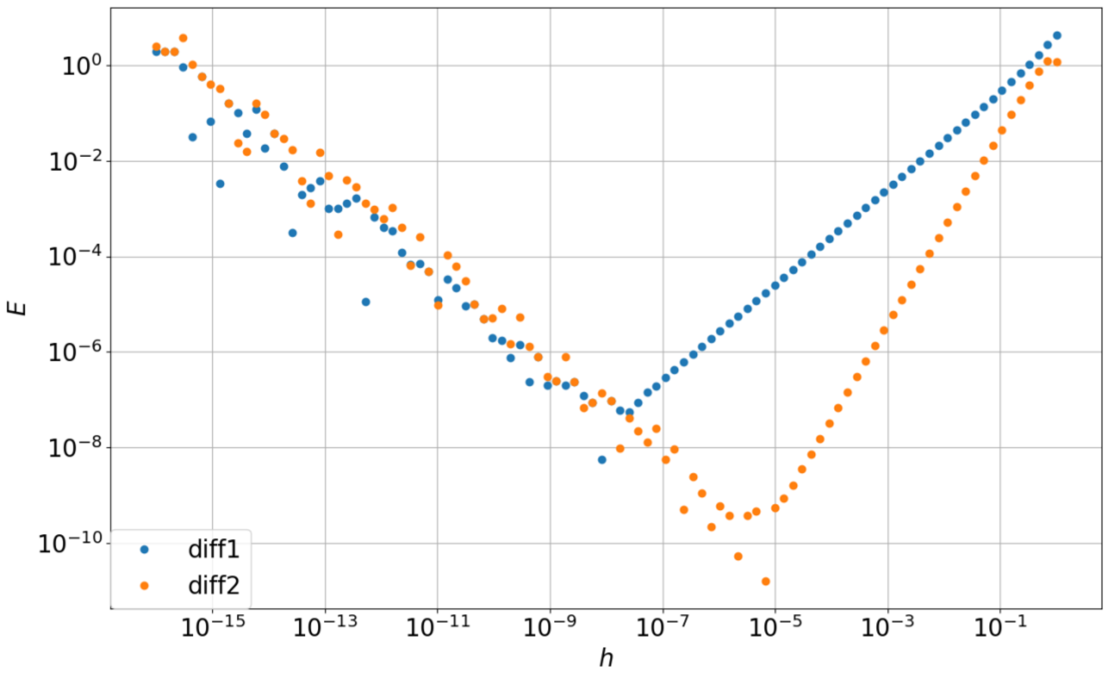

Рисунок 1 – График зависимости абсолютной погрешности для $diff1$ и $diff2$ от шага дифференцирования h.

Анализ полученных с помощью графика данных

Из графика на рисунке 1 можно сделать выводы:

1. При уменьшении шага дифференцирования после какого-то минимального для метода шага h^{opt} график абсолютной погрешности начинает устремляться в бесконечность, то есть возникает вычислительная неустойчивость.
2. На графике видно, что при использовании формулы численного дифференцирования 1-го порядка можно добиться вычислительной погрешности порядка 10^{-8}. При этом минимальный шаг дифференцирования $\approx 10^{-8}$.

3. В свою очередь, используя формулу численного дифференцирования 2-го порядка получаем полную погрешность порядка 10^{-11}. Точность формулы численного дифференцирования 2-го порядка действительно оказалась выше, чем точность формулы численного дифференцирования 1-го порядка. При этом минимальный шаг дифференцирования $\approx 10^{-5}$.

Данные были рассчитаны для двойной точности чисел с плавающей точкой, этот тип берется по умолчанию. ($numpy.float64$)

Продвинутая часть

Определение порядка формулы дифференцирования по log-log графику

Покажем, что график степенной функции на log-log графике представляет из себя прямую, наклон которой и определяет ее степень.

Докажем это, рассмотрев функцию $y = x^n$. Прологарифмируем ее:

$$\log(y) = \log(x^n) = n \cdot \log(x)$$

Оси на log-log графике представляют из себя: ось абсцисс - $\tilde{x} = \log(x)$, а ось ординат - $\tilde{y} = \log(y)$. Тогда с учетом этой замены получим:

$$\tilde{y} = n \cdot \tilde{x}$$

Мы доказали, что степенная функция на log-log графике – наклонная прямая.

Таким образом, для $diff1$ порядок формулы определяется углом наклона функции $\tilde{y} = 1 \cdot \tilde{x}$, а для $diff2$ - $\tilde{y} = 2 \cdot \tilde{x}$.

Порядок формулы дифференцирования для $diff1$ и $diff2$ наглядно продемонстрирован на рисунке 2.

Поиск оптимального шага дифференцирования для diff1

Рассмотрим опять формулу численного дифференцирования первого порядка более детально:

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{h}{2} f''\big(\xi(x)\big) \qquad (9)$$

Поиск оптимального шага для формулы (9) возможен при введении некоторых упрощений.

Пусть ошибка округления или вычислительная ошибка ограничена машинным эпсилон, а производная в остаточном члене ограничена некоторым числом M. То есть:

$$|e(x_0 + h)|, |e(x_0)| \le \varepsilon,$$

$$|f''(\xi(x))| \le M$$

Тогда, используя уже выведенное равенство из формулы (7) и зная, что модуль суммы всегда меньше или равен сумме модулей, можем записать следующее неравенство:

$$E_1 = \left| \frac{e(x_0 + h) - e(x_0)}{h} - \frac{h}{2} f''(\xi(x)) \right|$$

$$\le \frac{|e(x_0 + h)| - |e(x_0)|}{h} + \frac{h}{2} \left| f''\big(\xi(x)\big) \right| \le \frac{2\varepsilon}{h} + \frac{h}{2} M$$

Из этого неравенства очевидно, что если будем уменьшать шаг дифференцирования h и пытаться устремлять полную погрешность E_1 к нулю, то какое-то время ошибка метода $\frac{h}{2} M$ будет стремиться к нулю. Но в какой-то момент времени (зависящий от машинного эпсилон) слагаемое $\frac{2\varepsilon}{h}$ станет доминировать в сумме и начнет уходить в бесконечность, как и полная погрешность. Это приводит к вычислительной неустойчивости методов численного дифференцирования и приводит к наличию оптимального шага. Что мы и видим на рисунке 1: что при устремлении шага дифференцирования h к нулю абсолютная погрешность будет возрастать пропорционально $O(h^{-1})$, а при устремлении h к бесконечности абсолютная погрешность будет возрастать пропорционально $O(h)$.

Продемонстрируем это для наглядности с помощью графика:

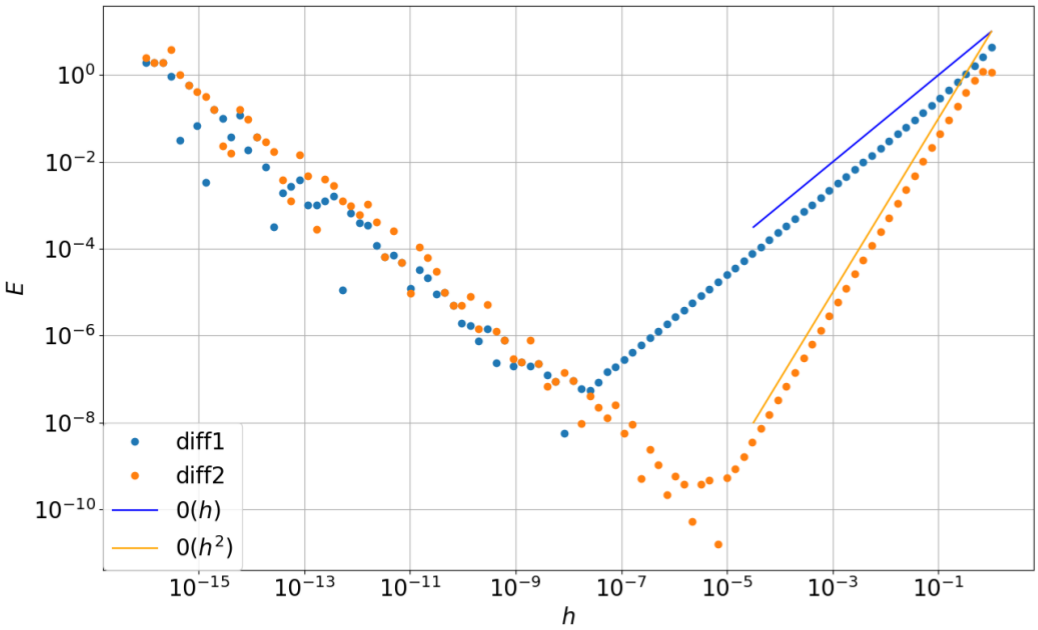

Рисунок 2. График зависимости абсолютной погрешности для $diff1$ и $diff2$ от шага дифференцирования h с изображением на нем порядка роста погрешности при $h > h^{opt}$.

Чтобы найти оптимальное значение шага, для которого полная погрешность будет минимальна, нужно продифференцировать по h значение полной погрешности и найти экстремум:

$$\left(\frac{2\varepsilon}{h} + \frac{h}{2}M\right)'_h = 0$$

$$-\frac{2\varepsilon}{h^2} + \frac{M}{2} = 0 \Rightarrow h^{opt} = \sqrt{\frac{4\varepsilon}{M}}$$

Найдем верхнюю границу для значения M:

$$|f''(\xi(x))| \leq M$$

$\xi(x) \in (x_0, x_0 + h)$, так как $x_0 = 2$, а $h \in [10^{-16}; 1]$ $\Rightarrow h_{\max} = 1$, значит $\xi(x) \in (2, 3)$.

Вычислим максимальное значение $g''(\xi)$, при $\xi(x) \in (2, 3)$:

$$g(\xi) = \sin(\xi)\,\xi^2$$

$$g'(\xi) = \sin(\xi)\cdot\xi\cdot 2 - \cos(\xi)\,\xi^2$$

$$g''(\xi) = \sin(\xi)\cdot 2 + 4\xi\cdot\cos(\xi) - \xi^2\sin(\xi)$$

Для определения максимального значения $g''(\xi)$ продифференцируем его и найдем экстремум:

$$(g''(\xi))'_{\xi} = 6\cos(\xi) - 6\xi\sin(\xi) - \xi^2\cos(\xi) = 0$$

$$(6 - \xi^2)\cos(\xi) - 6\xi\sin(\xi) = 0$$

$$\xi = 2.98147$$

Найдем значение $g''(\xi)$ в найденной точке ξ:

$$g''(2.98147) = -2\sin(2.981472) + 4 \cdot 2.98147\cdot\cos(2) - 2.98147^2\sin(2.98147) = -12.8717$$

Таким образом, максимальное значение $M = |-12.8717| = 12.8712$.

Значение машинного эпсилон зависит от точности $float$, который использовался в программе (он по умолчанию $numpy.float64$). Для $numpy.float64$ машинный эпсилон равен $2.2204\cdot 10^{-16}$.

Тогда с учетом вышесказанного, найдем оптимальный шаг:

$$h^{opt} = \sqrt{\frac{4\cdot 2.2204\cdot 10^{-16}}{12.8712}} = 8.307\cdot 10^{-9}$$

Найдем минимальное значение погрешности E_1 с помощью функции min:

Листинг 7 – Нахождение минимального значения E_1

```
[[id_E1]] = np.argwhere(error_diff1(x, h) == min(error_diff1(x, h)))
h1_opt = h[id_E1]
print(f"h1_opt = {h1_opt})
```

После выполнения листинга 7, получим значение шага:

$$h^{opt} = 8.302176\cdot 10^{-9}$$

Значение близко, к полученному выше оптимальному шагу, вычисленному аналитически.

Поиск оптимального шага дифференцирования для diff2

Аналогичным образом найдем оптимальный шаг для улучшенной формулы дифференцирования:

$$f'(x_0) = \frac{-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)}{2h} + f'''(\xi)\frac{h^2}{3}$$

Заметим, что здесь $\xi \in (x_0, x_0 + 2h)$.

Также выведем значение E_2, воспользовавшись упрощениями:

$$|e(x_0 + 2h)|, |e(x_0 + h)| \le \varepsilon,$$

$$|f'''(\xi(x))| \le M$$

Найдем абсолютная погрешность для 2 метода дифференцирования исходя из формулы (9) и введенных приближений:

$$\begin{aligned} E_2 = &\left|\frac{-3e(x_0) + 4e(x_0 + h) - e(x_0 + 2h)}{2h} + f'''(\xi)\frac{h^2}{3}\right| \\ &\le \frac{-3|e(x_0)| + 4|e(x_0 + h)| - |e(x_0 + 2h)|}{2h} + \frac{h^2}{3}\left|f'''\big(\xi(x)\big)\right| \\ &\le \frac{4\varepsilon}{h} + \frac{h^2}{3}M \end{aligned}$$

При устремлении шага дифференцирования h к нулю абсолютная погрешность будет возрастать пропорционально $O(h^{-1})$, а при устремлении h к бесконечности абсолютная погрешность будет возрастать пропорционально $O(h^2)$. Что мы и видим на рисунке 2.

Найдем значение оптимального шага, найдя экстремум функции $\frac{4\varepsilon}{h} + \frac{h^2}{3}M$:

$$\left(\frac{4\varepsilon}{h} + \frac{h^2}{3}M\right)'_h = 0$$

$$-\frac{4\varepsilon}{h^2} + \frac{2hM}{3} = 0 \Rightarrow h^{opt} = \sqrt[3]{\frac{6\varepsilon}{M}}$$

Аналогично найдем верхнюю границу для этого неравенства $E_2 \leq \frac{4\varepsilon}{h} + \frac{h^2}{3}M$, для этого найдем максимальное значение M:

$$g^{(4)}(\xi) = \xi^2 \sin(\xi) - 12 \cdot \sin(\xi) - 8\xi \cos(\xi) = 0$$

$$\xi = 2.02463$$

Подставим ξ в $g'''(\xi)$ и найдем тем самым максимальное значение M:

$$|g'''(2.02463)| = |6 \cos(2.02463) - 6 \cdot 2.02463 \sin(2.02463) - 2.02463^2 \cos(2.02463)| = 11{,}7515$$

Найдем

$$h_{opt} = \sqrt[3]{\frac{6 \cdot 2.2204 \cdot 10^{-16}}{11{,}7515}} \approx 5 \cdot 10^{-6}$$

Найдем минимальное значение погрешности E_2 с помощью функции min:

Листинг 8 – Нахождение минимального значения E_2

```
[[id_E2]] = np.argwhere(error_diff2(x, h) == min(error_diff2(x, h)))
h2_opt = h[id_E2]
print(f"h2_opt = {h2_opt}")
```

После выполнения листинга 8, получим значение шага:

$$h^{opt} = 6.73415 \cdot 10^{-6}$$

Значение h^{opt} для $diff2$ также близко, к полученному выше оптимальному шагу, вычисленному аналитически.

Прямой режим автоматического дифференцирования

Реализуем класс для дуальных чисел $AutoDiffNumber$, в котором перегрузим все нужные арифметические операции для функции $f(x)$, данной нам по условию. Для каждого перегруженного оператора сделаем проверку типа с помощью функции $isinstance()$, чтобы можно было делать операции как с дуальными числами, так и с константами численного типа. Это условие предусмотрено для перегрузки операторов $\_\_add\_\_$ (сложения),

$\_\_sub\_\_$ (вычитания), $\_\_mul\_\_$ (умножения), $\_\_truediv\_\_$ (деления), так как здесь операндом может являться, как обычное число, так и дуальное (справа). В $\_\_pow\_\_$ справа всегда находится вещественное число, поэтому эту проверку для возведения в степень мы не делаем.

Также перегрузим операторы для сложения, вычитания, умножения и деления для левого операнда соответственно $\_\_radd\_\_$, $\_\_rsub\_\_$, $\_\_rmul\_\_$, $\_\_rtruediv\_\_$.

Листинг 9 – Часть кода из реализации класса $AutoDiffNumber$, для операций с дуальными числами.

```
class AutoDiffNumber:
    def __init__(self, a, b):
        self.a=a
        self.b=b
    def __add__(self, other):
         # (a1 + b1 * e) + (a2 + b2 * e) = (a1 + a2) +  (b1 + b2) * e
        if isinstance(other, AutoDiffNumber):
            return AutoDiffNumber(self.a+other.a,self.b+other.b)
        else:
            return AutoDiffNumber(self.a+other,self.b)

    def __radd__(other, self):
         # (a1 + b1 * e) + (a2 + b2 * e) = (a1 + a2) +  (b1 + b2) * e
        if isinstance(other, AutoDiffNumber):
            return AutoDiffNumber(self.a+other.a,self.b+other.b)
        else:
            return AutoDiffNumber(self.a+other,self.b)
```

Реализация функции forward_autodiff

Главное свойство дуальных чисел:

$$f(a + b\varepsilon) = f(a) + b\varepsilon \cdot f'(a)$$

Чтобы найти производную, будем считать ее в нужной нам точке x_0 и использовать $b = 1$, чтобы сразу находить значение $f'(x_0)$ без деления на число b.

Создадим функцию $forward\_autodiff$, которая будет рассчитывать значение производной. На вход ей мы подадим функцию и точку, в которой хотим найти производную.

Листинг 10 – Реализация функции $forward\_autodiff$.

```
def forward_autodiff(f,x):
    auto_number = AutoDiffNumber(x, 1)
    result = f(auto_number)
    return(result.b)
```

$forward\_autodiff$ принимает на вход вершину вычислительного графа и возвращает дуальную часть полученного результата.

Продемонстрируем вычисление производной с помощью реализованного класса для дуальных чисел с помощью графика на 100 случайных точках $x \in [-1; 1]$ и сравним полученные значения производных с аналитическими значениями.

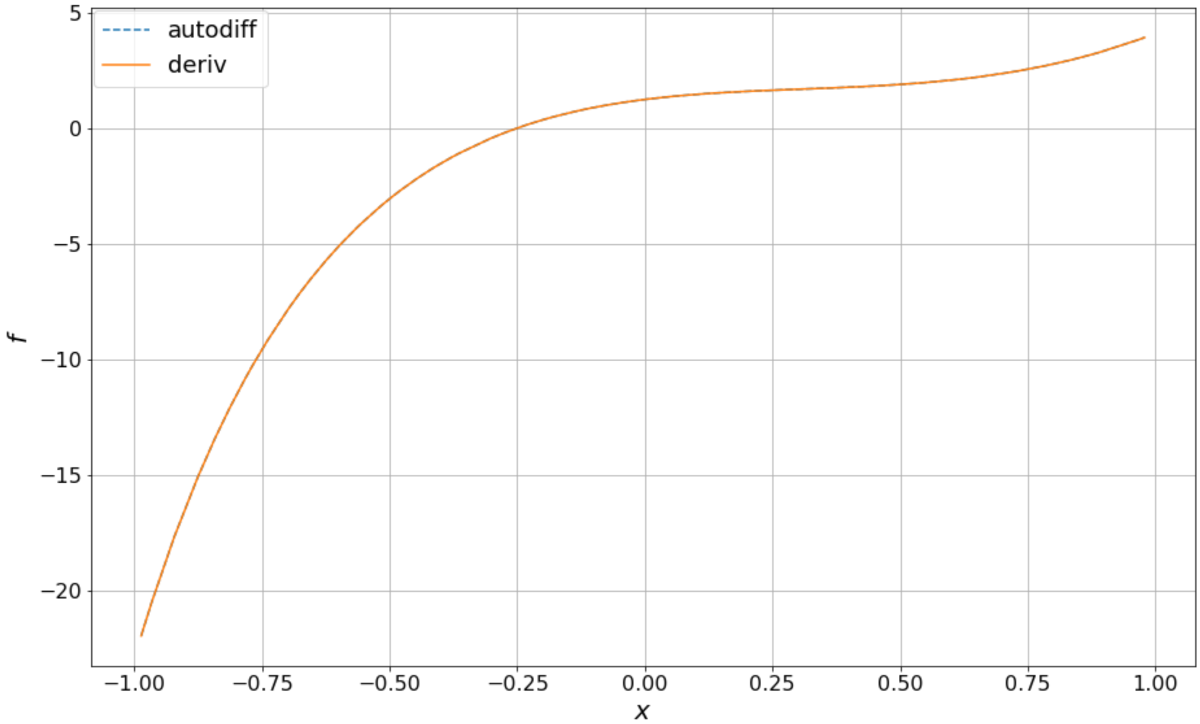

Рисунок 3 – График производной, полученной аналитически, и производной, полученной с помощью автоматического дифференцирования на интервале $[-1; 1]$. ($deriv$ – график производной, полученной аналитически, $autodiff$- график производной, полученной с помощью автоматического дифференцирования).

Видно, что оба графика производных полностью совпали, что говорит о корректности метода автоматического дифференцирования.

Вычислим значения производных в тех же точках, используя функции $diff1$ и $diff2$, и сравним полученные значения с аналитическими значениями и значениями, полученными с помощью автоматического дифференцирования.

Вычислять значения производных используя $diff1$ и $diff2$ будем с использованием оптимальных шагов, для каждого из методов, которые мы вывели выше.

Листинг 11 – Вычисление производной для четырех методов с использованием 100 случайных точек.

```
autodiff = []
deriv = []
diff1_p = []
diff2_p = []

for i in range(0, 100):
    autodiff.append(forward_autodiff(f, points[i]))
    deriv.append(derivative_f(points[i]))
    diff1_p.append(diff1(points[i], h1_opt, f))
    diff2_p.append(diff2(points[i], h2_opt, f))
```

Продемонстрируем результат для всех четырех методов с помощью графика:

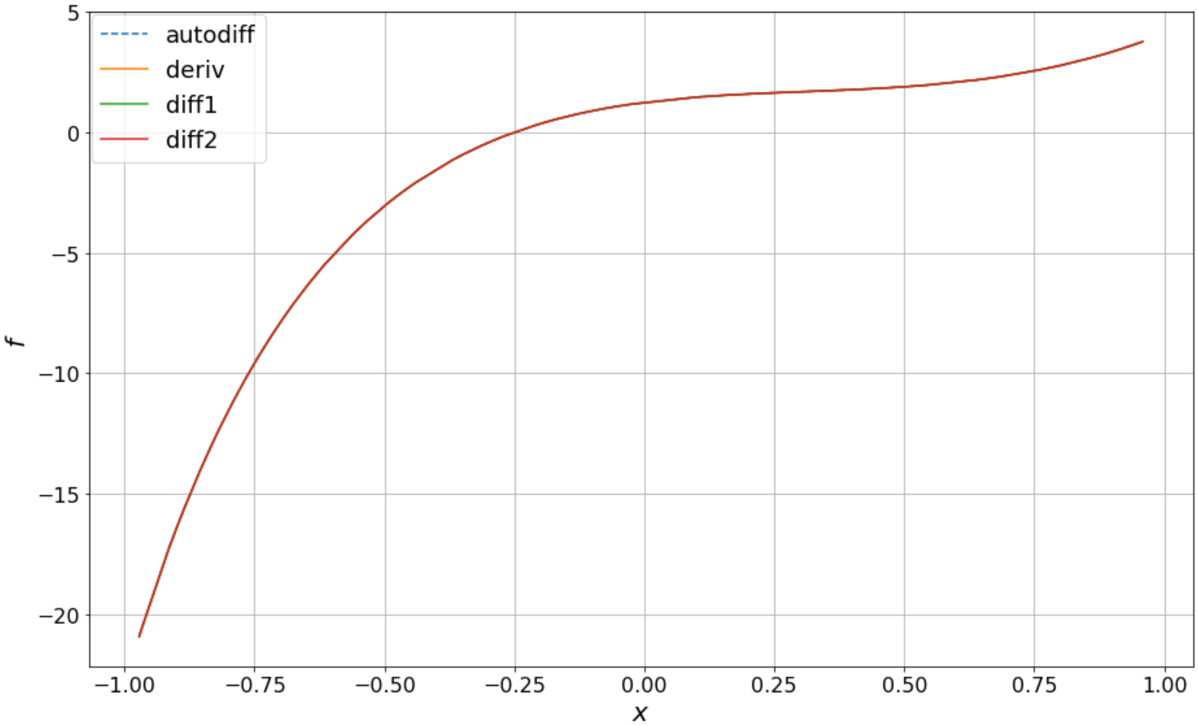

Рисунок 4 – График производных, полученных аналитически, численным методом и методом автоматического дифференцирования на основе 100 точек на интервале $[-1;1]$. ($deriv$ – график производной, полученной аналитически, $autodiff$- график производной, полученной с помощью автоматического дифференцирования, $diff1$ – график производной полученной, с помощью 1-го метода численного дифференцирования, $diff2$ – график производной полученной, с помощью 1-го метода численного дифференцирования).

Значения отличаются очень незначительно:

i	x	autodiff	deriv	diff1_p	diff2_p
0	-0.971478249837009	-20.924296206161006	-20.924296206161006	-20.924296202949790	-20.924296015496072
1	-0.969417583487012	-20.781193386359984	-20.781193386359980	-20.781193383238449	-20.781193301808706
2	-0.967451036053786	-20.645456044876649	-20.645456044876649	-20.645456041767520	-20.645455883426944
3	-0.914402784153407	-17.270063144438929	-17.270063144438932	-17.270063141839582	-17.270062918087490
4	-0.899163577925292	-16.394576880153998	-16.394576880153998	-16.394576877763743	-16.394576684703804
5	-0.885244774506944	-15.628400746563999	-15.628400746563999	-15.628400744111351	-15.628400560773784

Рисунок 5 – Иллюстрация того, как незначительно отличаются значения для каждого метода от точного значения производной (столбец со значениями $deriv$).

Попробуем передавать в функции $diff1$ и $diff2$ не оптимальный шаг, а какой-то $h = 0.3$. И продемонстрируем на графике изменение поведения функций $diff1$ и $diff2$.

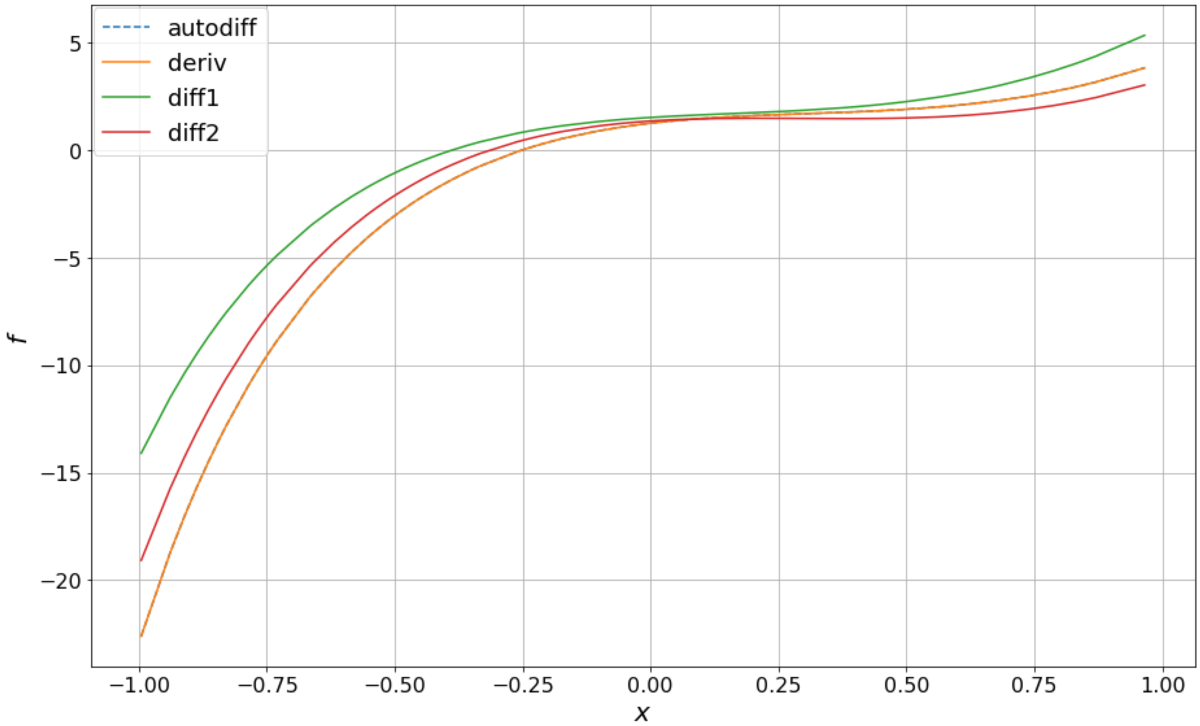

Рисунок 6 – Иллюстрация отклонения графиков для численных методов $diff1$ и $diff2$ от графика, использующего аналитический способ поиска производной, $defiv$ и графика, использующего автоматического $autodiff$.

Исходя из данных, полученных на рисунках 5 и 6, можно сделать вывод, что производные, полученные численными методами, не дают точных результатов, если мы не вычислили для них оптимальный шаг дифференцирования, в отличие от автоматического дифференцирования, где результат точный.

Заключение

Таким образом, в ходе лабораторной работы были реализованы 2 метода численного дифференцирования на языке $Python$ с дальнейшим анализом полученных результатов.

Была проделана работа по уменьшению погрешности численного метода с помощью экстраполяции Ричардсона, которая позволила нам повысить порядок точности до $O(h^2)$. Также были рассмотрены аналогичные способы увеличения порядка точности с помощью разложения функций в ряд Тейлора и интерполяции Лагранжа.

Также в ходе работы был выявлен основной недостаток методов численного дифференцирования – появление вычислительной неустойчивости: точность вычислений растёт при уменьшении шага дифференцирования до определённого момента. После достижения какого-то минимального шага h^{opt} абсолютная вычислительная погрешность резко начинает увеличиваться и уходить в бесконечность.

На языке $Python$ был реализован еще один метод для нахождения производной функции – прямой метод автоматического дифференцирования. Он позволяет рассчитать точное значение производной в точке, используя свойство дуальных чисел. При этом этот метод лишен необходимости использовать значения функции в соседних точках и возникновения вычислительной погрешности, что объясняет его широкое использование по сравнению с численными методами.

Список использованных источников

1. Першин А. Ю. Лекции по вычислительной математике. [Электронный ресурс] // Москва, МГТУ им. Н. Э. Баумана, 2020. - 145с. – Режим доступа:

https://archrk6.bmstu.ru

2. Першин А. Ю. Семинар №2 по курсу «Вычислительная математика». [Электронный ресурс] – Режим доступа: https://archrk6.bmstu.ru

3. Вычислительная математика, лекция № 4 // Видеохостинг "Youtube" URL: https://www.youtube.com/watch?v=d5xHDd_K_jE (дата обращения: 30.03.2021).

4. Вычислительная математика, лекция № 5 // Видеохостинг "Youtube" URL: https://www.youtube.com/watch?v=6TcTplqaMxg (дата обращения: 30.03.2021).