МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра вычислительной математики

Жуковский Павел Сергеевич

Отчёт по лабораторной работе №1, вариант 2

(«Методы вычислений»)

Студента 2 курса 13 группы

Преподаватель

Бондарь Иван Васильевич

Минск 2020

Вопрос 2.1. Для чего предназначена функция ‘math.expm1’, входящая в набор стандартных математических функций большинства языков программирования?

Ответ: Функция math.expm1(x) из библиотеки math, действительно входящая в набор математических функций большинства языков программирования (например, в Python эта функция есть точно), нужна для того, чтобы посчитать значение функции вида $y = e^x - 1$.

Можно задаться вопросом: зачем нам нужна эта функция, если есть стандартная функция math.exp(x), которая посчитает нам e^x, после чего мы просто отнимем единицу? Проблема в том, что при вычислении значения функции $y = e^x - 1$ через стандартную math.exp(x) очень высока вероятность того, что мы потеряем точность, особенно если речь идёт об очень маленьких входных значениях x (например: 1^{-17}, 1^{-19}, 1^{-23} и т.д.). Если же мы воспользуемся нашей функцией math.expm1(x), то у нас не будет проблем с точностью, т.к. эта функция способна нам дать точный ответ даже при очень маленьких входных значениях параметра x.

Причины проблемы с точностью, которые преодолевает функция math.expm1(x), будут объяснены в **вопросе 2.3.** ниже.

Вопрос 2.2. Приведите примеры кода (желательно построить графики), подтверждающие необходимость использования этой функции.

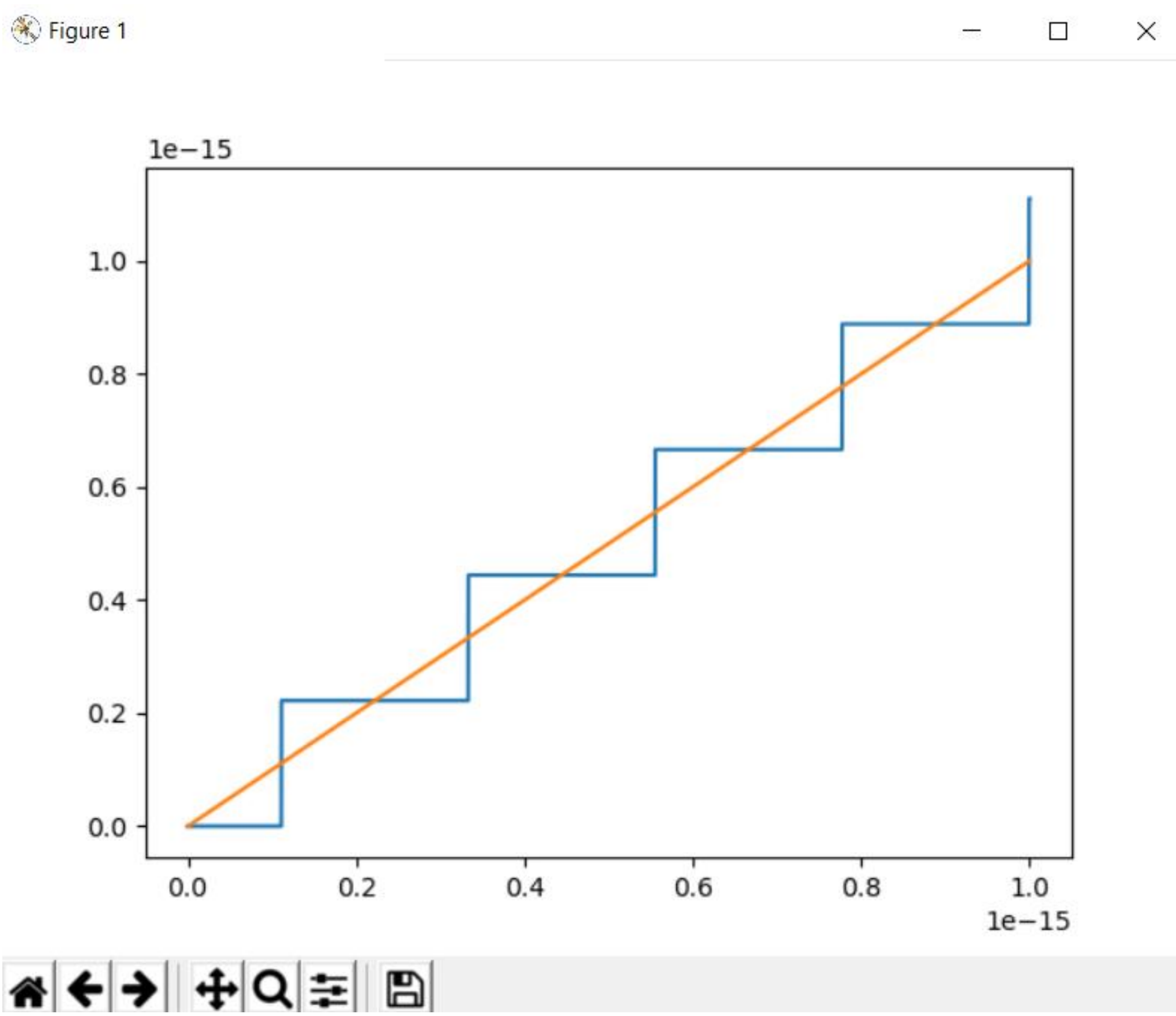
Ответ: Как было сказано выше, функция math.expm1(x) нам необходима для того, чтобы решить проблемы с точностью вычислений при малых значениях параметра x. Для того, чтобы показать, что точность вычислений функций 1) y_1 = math.exp(x) – 1 и 2) y_2 = math.expm1(x) действительно отличается при малых значениях x, я написал небольшую программу на языке Python, которая рисует графики этих функций (т.е. мы можем наблюдать различные значения y при одних и тех же значениях x, что свидетельствует о разной точности вычислений). Для наглядности был выбран диапазон $[0, 10^{-15}]$. Исходный код этой программы будет прикреплён вместе с отчётом под именем main_1.py.

Код программы:

```
# Импорт необходимых модулей для рисования
import math
import numpy as np
import matplotlib.pyplot as plt

# Описание функции y_1 = exp(x) - 1
```

```
y1 = lambda x: np.exp(x) - 1  # Синяя линия

# Описание функции y_2 = expm1(x)
y2 = lambda x: np.expm1(x)  # Оранжевая линия

# Создание рисунка с координатной плоскостью
fig = plt.subplots()

# Создание области, в которой будет отображаться график
# Первые два аргумента - диапозон значений для x, третий аргумент - качество графика
x = np.linspace(0, math.pow(10, -15), 100000)  # Диапозон [0, 10^(-15)]

# Рисование графиков функций
plt.plot(x, y1(x))  # Синяя линия
plt.plot(x, y2(x))  # Оранжевая линия

# Показ графика
plt.show()
```

Получившиеся графики:



На графиках чётко видно, что при разных значениях x (на малых значениях), функции y1 = exp(x) – 1 (синяя линия) и y2 = expm1(x) (оранжевая линия) далеко не всегда вычисляют одно и то же значение. Сам график в среде на Python можно всячески увеличивать/уменьшать, двигать, изменять и т.д. Изображение графика также будет прикреплено рядом с отчётом под именем Figure_1.png.

Из графика можно сделать вывод, что функция math.expm1(x) нам действительно необходима, т.к. на при малых значениях аргумента x стандартная функция math.exp(x) с последующим отниманием единицы работает плохо (точность вычислений страдает). О причинах проблемы будет написано в следующем вопросе.

Вопрос 2.3. Объясните причины проблемы, которую позволяет преодолеть указанная функция.

Ответ: Итак, как упоминалось ранее, функция math.expm1(x) позволяет преодолеть проблему с точностью вычислений. Рассмотрим же причину того, почему стандартная функция math.exp(x) с последующим отниманием единицы не считает также точно.

Для наглядности, рассмотрим конкретный пример. Предположим, нам нужно посчитать значение функции $y = e^x - 1$ для какого-нибудь очень маленького значения x, например, для $x = 10^{-20}$. Если мы будем считать это через обычную экспоненту math.exp(x) с последующим отниманием единицы, то вначале она посчитает нам значение выражения exp(x). По идее, ответом должна быть приблизительно одна целая и где-то 20 нулей после запятой (1,000…(где-то 20 цифр после запятой)). Однако, для хранения чисел с ненулевой частью дробное число типа double может хранить лишь 15 цифр после запятой. Т.е. наша функция math.exp(x) отбросит все цифры дальше 15-ой, таким образом мы получим math.exp(10^{-20}) = 1.0. Соответственно, после отнимания единицы мы получим 0 (это можно увидеть и на графиках из предыдущего вопроса), что конечно же некорректно. На самом деле ответом будет некоторое очень маленькое число, но все же большее, чем ноль (приблизительно 10^{-20}). Функция math.expm1(x) учитывает этот нюанс заранее, и поэтому эта функция считает значение достаточно точно. О том, как реализована функция math.expm1(x) и почему ей удается обойти проблему с точностью, рассказано в следующем вопросе.

Вопрос 2.4. Попробуйте предположить, каким образом реализована функция ‘expml1’. Напишите свою версию этой функции и проверьте полученный результат.

Ответ: Итак, в этом вопросе разберёмся, как реализована функция math.expm1(x). Данная функция, как и функция math.exp(x) реализована с помощью ряда Тейлора, только её ряд начинается не с единицы, а с x. Т.е. это обычная функция, но немного с другим рядом Тейлора, который заранее учитывает ту единицу, которую мы собираемся отнять от экспоненты. Я написал программу на Python, которая рекурсивно считает члены ряда Тейлора для функции math.expm1(x) с некоторой точностью, и сравнил значение с получившимся значением библиотечной функции. Исходный код этой программы будет прикреплён вместе с отчётом под именем main_2.py.

Код программы:

```python
# Подключаем необходимые библиотеки
import math

print("Введите точность вычислений для функции:")
epsilon = float(input())  # Точность вычислений
print("Введите значение аргумента x:")
x = float(input())  # Число x, которые мы будем подавать на вход функции
print("\nВведённая точность вычислений для нашей функции:", epsilon)
print("Введённое значение аргумента x: ", x)
sum = 0  # Накопившаяся сумма ряда Тейлора
a = x  # Дополнительная переменная, которая нам нужна для подсчёта слагаемых ряда Тейлора
i = 1  # Дополнительная переменная, которая нам нужна для подсчёта слагаемых ряда Тейлора
while abs(a) > epsilon:  # Пока не достигнем нужной точности, будем считать новое число ряда (оно все меньше и меньше)
    sum += a  # Добавляем к сумме ряда новое слагаемое
    a *= x / (i + 1)  # Считаем новое слагаемое
    i += 1  # Делаем инкремент переменной i (это нужно для нового факториала)
fault = abs(sum - math.expm1(x))  # Погрешность наших вычислений относительно библиотечной функции
print("Эталонное значение вычислений библиотечной функции math.exmp1(x): ", math.expm1(x))
print("Получившееся значение нашей функции (через ряд Тейлора): ", sum)
print("Погрешность вычислений составляет: ", fault)
```

Пример работы программы:

```
Введите точность вычислений для функции:
1e-100
Введите значение аргумента x:
100

Введённая точность вычислений для нашей функции: 1e-100
Введённое значение аргумента x:  100.0
Эталонное значение вычислений библиотечной функции math.exmp1(x):  2.6881171418161356e+43
Получившееся значение нашей функции (через ряд Тейлора):  2.688117141816137e+43
Погрешность вычислений составляет:  1.4855280471424563e+28
```

Понятное дело, моя функция может допускать некоторые погрешности во время вычисления ряда Тейлора. Я считаю эту погрешность и вывожу её, для наглядности. В программе можно задать любую точность вычислений для epsilon и любое значение числа x. Я привел пример работы программы для точности epsilon = 10^{-100} и x = 100, чтобы было примерно понятно, как должна работать программа. Программа выводит сначала значение, которое посчитала библиотечная функция math.exmp1(x), затем значение, которое собралось в насчитанном программой ряде Тейлора для этой функции, а затем погрешность вычислений (разница между библиотечным вычислением и моим).

Подытожив, можно сказать, что для вычисления очень маленьких значений x лучше всего использовать функцию math.exmp1(x), а не math.exp(x) – 1, так как первая лучше работает в плане точности вычислений, что объясняется особенностями машинной арифметики…