Отчёт по лабораторной работе №1.

Чилеше Абрахам. Б9122-02.03.01 СЦТ (1)

Вариант-17

Цель работы:

1. Построить интерполяционный многочлен Лагранжа для заданной функции.
2. Построить таблицу абсолютной и относительной погрешностей и остаточного члена для каждой n.
3. Построить график зависимостей $\Delta fn(n)$, $\boldsymbol{rn(n)}$
4. делать вывод.

Условия:

- функция: $f(x) = 0.5x^2 + \cos(2x)$
- Интервал: [0.6, 1.1]
- n ∈ [3, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

Ход работы:

я проводил лабораторную работу, используя язык программирования python

библиотеки

я использовал следующие библиотеки python, чтобы помочь мне с этой лабораторной работой

```python
import math
import matplotlib.pyplot as plt
import numpy as np
from tabulate import tabulate
from tkinter import Tk, Text, END

```

- **math:** Стандартная библиотека Python для математических функций и констант. Используется специально для вычисления факториалов с помощью math.factorial().

- **matplotlib.pyplot:** Библиотека для создания различных типов графиков и визуализаций. В этом коде он используется для построения графиков для визуализации результатов интерполяции и анализа ошибок.

- **numpy** - фундаментальная библиотека для научных вычислений, предлагающая поддержку массивов, матриц и математических функций. Используется для численных вычислений, таких как операции с массивами, вычисления абсолютных значений и нахождение максимальных значений.

- **tabulate:** Используется для создания отформатированных таблиц из структур данных Python. В этом коде используется для создания отформатированной таблицы, отображающей результаты интерполяции и анализ ошибок.

- **tkinter:** стандартный набор инструментов Python с графическим интерфейсом, используемый для создания графических пользовательских интерфейсов. Здесь он используется для создания простого графического окна для отображения таблицы, созданной библиотекой tabulate.

функциональный график

используя библиотеку matplotlib.pyplot, я построил график $f(x) = 0.5x^2 + \cos(2x)$ в интервале [0.6, 1.1]

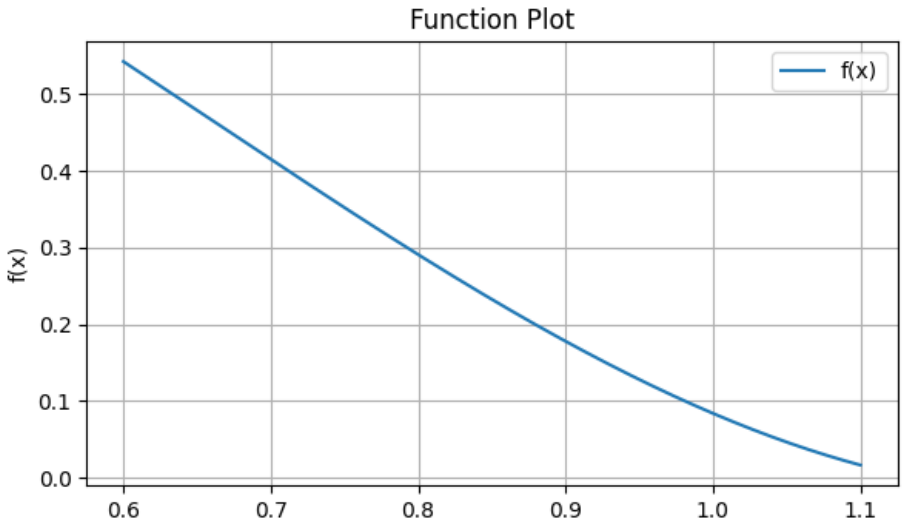


Рис. 1. График исходной функции *f*

```
def f(x):
    return 0.5*x**2 + np.cos(2*x)
```

полный код для построения графика можно найти в прикрепленном файле на python

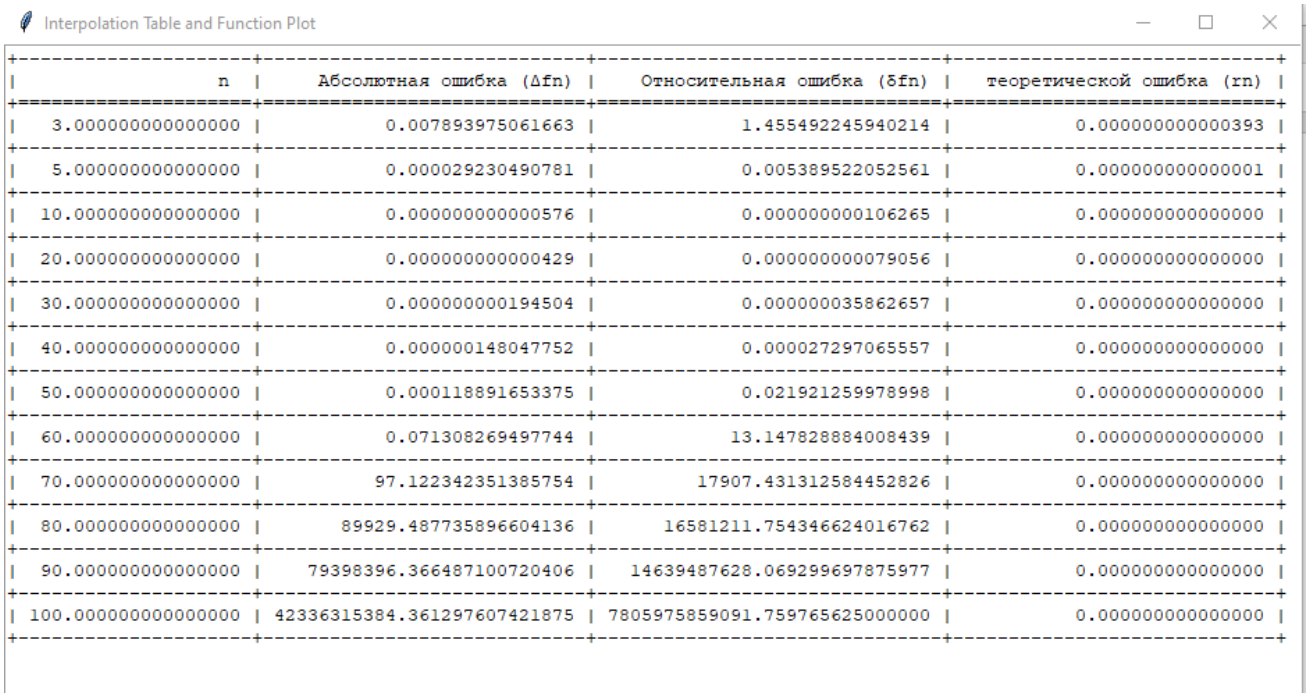
функции полинома Лагранжа

```python
def lagrange_interpolation(xi, fi, x):
    n = len(xi)
    result = 0
    for i in range(n):
        term = fi[i]
        for j in range(n):
            if j != i:
                term *= (x - xi[j]) / (xi[i] - xi[j])
        result += term
    return result
```

Таблица с данными

Для создания таблицы я использовал библиотеки tabulate и tkinter, которые помогли мне создать окно, а также правильно упорядочить значения в таблице.

Значения для абсолютного и относительного столбцов кажутся правильными, но из-за проблемы с моей теоретической функцией я получал странные значения для столбца теоретической ошибки



n	Абсолютная ошибка (Δfn)	Относительная ошибка (δfn)	теоретической ошибка (rn)
3.000000000000000	0.007893975061663	1.455492245940214	0.000000000000393
5.000000000000000	0.000029230490781	0.005389522052561	0.000000000000001
10.000000000000000	0.000000000000576	0.000000000106265	0.000000000000000
20.000000000000000	0.000000000000429	0.000000000079056	0.000000000000000
30.000000000000000	0.000000000194504	0.000000035862657	0.000000000000000
40.000000000000000	0.000000148047752	0.000027297065557	0.000000000000000
50.000000000000000	0.000118891653375	0.021921259978998	0.000000000000000
60.000000000000000	0.071308269497744	13.147828884008439	0.000000000000000
70.000000000000000	97.122342351385754	17907.431312584452826	0.000000000000000
80.000000000000000	89929.487735896604136	16581211.754346624016762	0.000000000000000
90.000000000000000	79398396.366487100720406	14639487628.069299697875977	0.000000000000000
100.000000000000000	42336315384.361297607421875	7805975859091.759765625000000	0.000000000000000

Ошибки

абсолютная ошибка

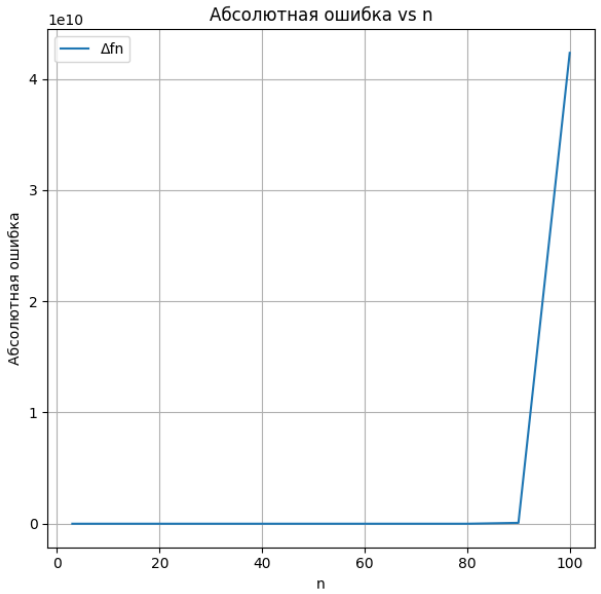


Рис. 2. Графк зависимости абсолютной ошибки от количества разбиений n

Относительная ошибка

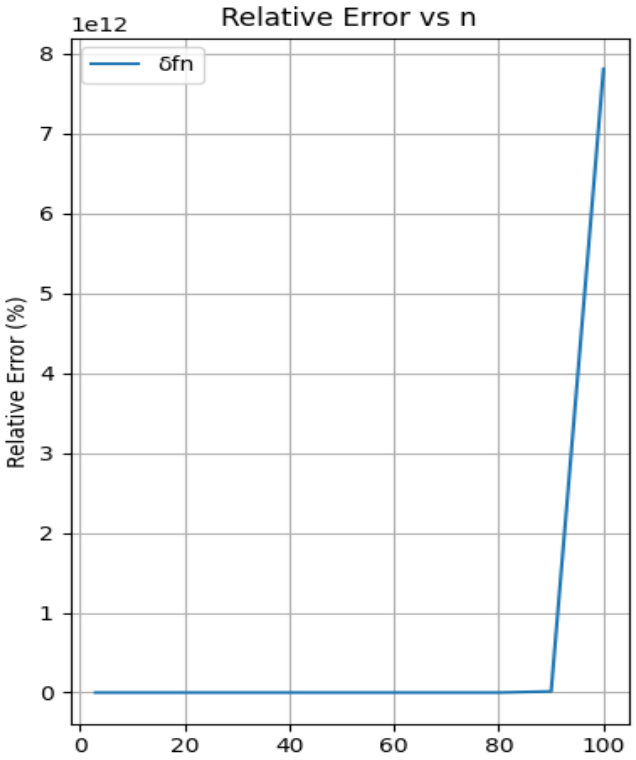


Рис. 2. График зависимости Относительная ошибка от количества разбиений n

Теоретическая ошибка

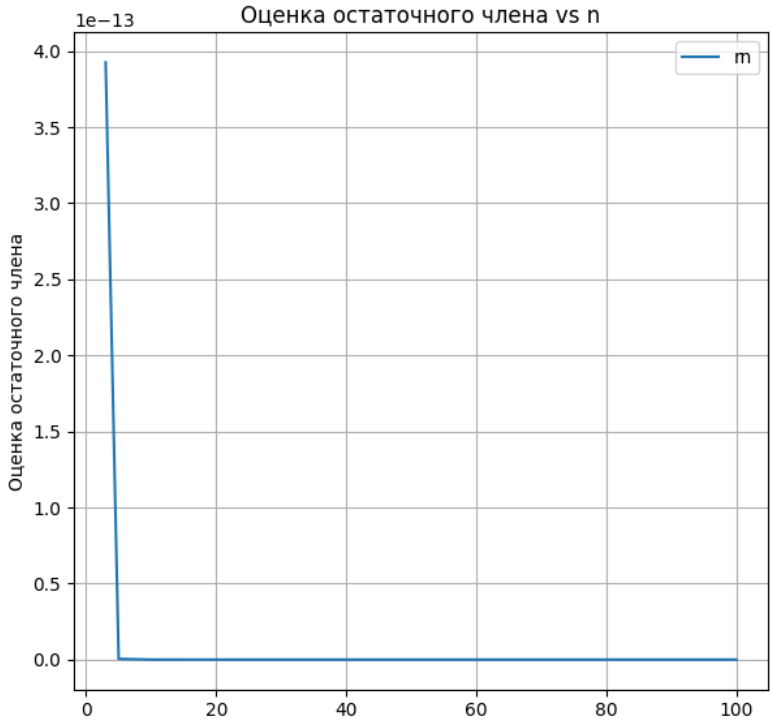
я столкнулся с проблемой, пытаясь построить график функции для теоретической ошибки. моя функция не давала мне правильную точку

```
1 usage
def residual_estimate(f, n, a, b):
    # Step 1: Compute the maximum absolute value of the (n+1)-th derivative of f(x)
    x_vals = np.linspace(a, b, num: 1000)
    f_n_plus_1 = np.max(np.abs(np.diff(f(x_vals), n+1)))  # Maximum absolute difference of

    # Step 2: Compute the factorial of (n+1)
    factorial_n_plus_1 = math.factorial(n+1)

    # Step 3: Compute (b - a) raised to the power of (n+1)
    b_minus_a_power_n_plus_1 = (b - a)**(n+1)

    # Step 4: Compute the final estimate
    r_n = f_n_plus_1 / (factorial_n_plus_1 * b_minus_a_power_n_plus_1)
    return r_n
```



Вывод

Практическая абсолютная ошибка (Δfn) уменьшается с увеличением количества точек интерполяции, что указывает на улучшение точности интерполяции. С другой стороны, теоретическая остаточная ошибка (rn) также уменьшается в начале, но может стабилизироваться или проявить колебания за определенную

точку, что дает представление о ожидаемом поведении ошибки на основе свойств интерполируемой функции.

я прикрепил файл на python