Интерполирование функции одной переменной с помощью интерполяционной формулы Лагранжа

Описание метода.

Даны n точек и n значений функции в этих точках. Требуется произвести интерполирование этой функции. Оно будет производиться посредством экспоненциального интерполирования. Многочлены этого интерполяционного полинома будут иметь вид:

$$\Phi_k(x) = \prod_{j \neq k} \frac{\mathrm{sh}\left((x - x_j)/2\right)}{\mathrm{sh}\left((x_k - x_j)/2\right)}$$

Или в более экономном варианте:

$$\Phi_k(x) = \frac{F(x)}{2\mathrm{sh}\left((x_k - x_j)/2\right)F'(x_k)}$$

Где *F(x) и F'(x$_k$)*:

$$F(x) = \prod_{j=0}^{n} \mathrm{sh}\left(\frac{x - x_j}{2}\right),\ F'(x_k) = \frac{1}{2}\prod_{j \neq k}^{n} \mathrm{sh}\left(\frac{x_k - x_j}{2}\right)$$

Сам интерполфционный полином будет таким:

$$g_n(x) = \sum_{k=0}^{n} f_k \Phi_k(x),\ f_i = f(x_i)$$

Тогда по этим формулам можно посчитать все промежуточные значения функции по имеющемуся дискретному набору.

Программная реализация.

```python
import math
import numpy as np
import matplotlib.pyplot as plt


def get_points(x0: float, x1: float, n: int):
    x = np.linspace(x0, x1, n)
    y = np.random.uniform(x0, x1, n)
    return x, y


def get_points_from_func(x0: float, x1: float, n: int, f):
    x = np.linspace(x0, x1, n)
    y = np.zeros(n)
    for i in range(n):
        y[i] = f(x[i])
    return x, y


def fi_k(x: np.array, x0: float, k: int, n: int):
    res = 1
    for j in range(k):
        res *= np.sinh((x0 - x[j]) / 2) / np.sinh((x[k] - x[j]) / 2)
    for j in range(k + 1, n):
        res *= np.sinh((x0 - x[j]) / 2) / np.sinh((x[k] - x[j]) / 2)
    return res


def interpolation(points: tuple, x0: float):
    x, y = points
    res = 0
    n = x.shape[0]
    for k in range(n):
        res += fi_k(x, x0, k, n) * y[k]
    return res
```

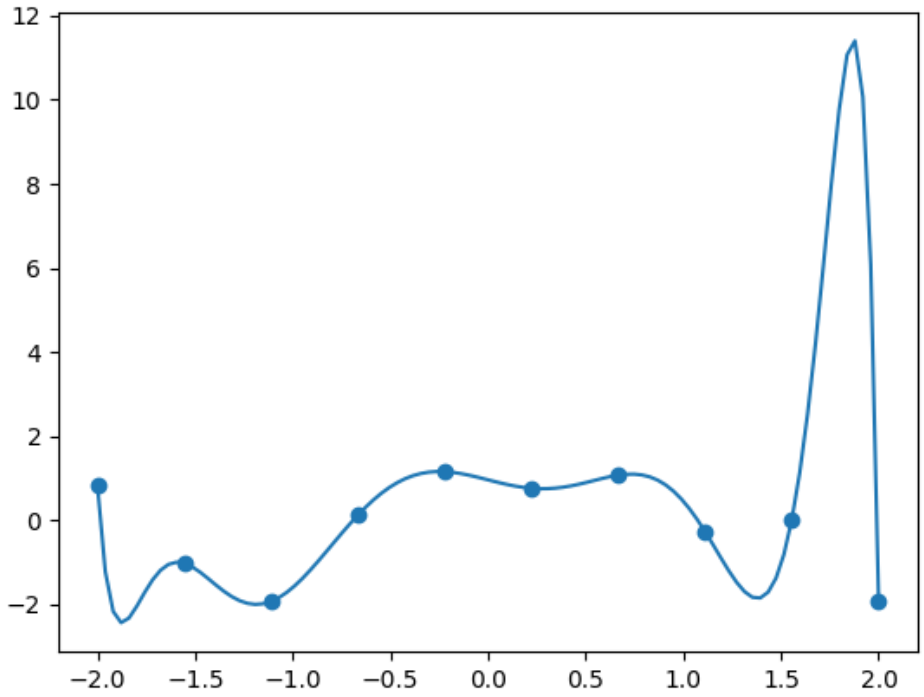
Проверка правильности решения

Сначала проверю работу на случайных точках на плоскости.

```
def test1(n):
    p = get_points(-2, 2, n, )
    plt.scatter(*p)
    x = np.linspace(-2, 2, 10 * n)
    y = np.zeros(10 * n)
    for i in range(10 * n):
        y[i] = interpolation(p, x[i])
    plt.plot(x, y)
    plt.show()


test1(10)
```

Результат:



Как видно, построенное приближение проходит через все заданные точки.

Теперь возьму функцию

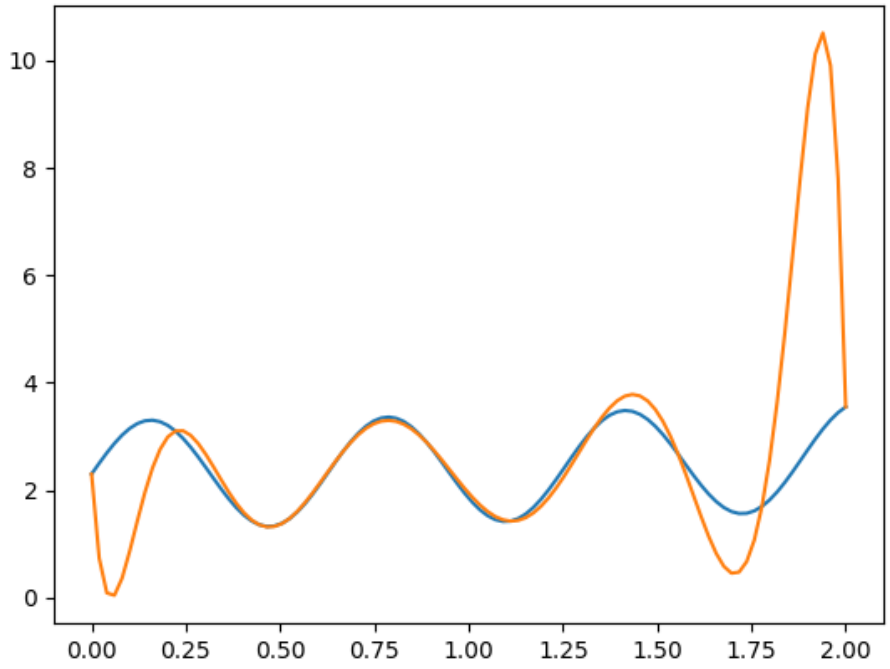
$$f(x) = sin(10x) + ln(x^2 + 10)$$

на промежутке от 0 до 2

```
def test2(n):
    x0, x1 = 0, 2
    f = lambda x: math.sin(10*x)+math.log(x*x+10)
    p = get_points_from_func(x0, x1, n, f)
    plt.plot(*get_points_from_func(x0, x1, 10*n, f))
    x = np.linspace(x0, x1, 10 * n)
    y = np.zeros(10 * n)
    for i in range(10 * n):
        y[i] = interpolation(p, x[i])
    plt.plot(x, y)
    plt.show()


test2(10)
```

Результат:



Как видно, программа работает правильно, интерполяция строится.