Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №5

по дисциплине

«Вычислительная математика»

Вариант №13

Выполнил:

Студент группы P3213

Султанов А.Р.

Проверила:

Машина Е.А.

г. Санкт-Петербург

2024г.

Цель работы

Решить задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек.

Исходные данные:

$X_1 = 1,168, X_2 = 1,463$

x_i	y_i
1,10	0,2234
1,25	1,2438
1,40	2,2644
1,55	3,2984
1,70	4,3222
1,85	5,3516
2,00	6,3867

Задание

Вычислительная реализация задачи

Таблица конечных разностей

$\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i$

x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$	$\Delta^6 y_i$
1,10	0,2234	1,0204	0,0002	0,0132	-0,0368	0,0762	-0,1313
1,25	1,2438	1,0206	0,0134	-0,0236	0,0394	-0,0551	
1,40	2,2644	1,034	-0,0102	0,0158	-0,0157		
1,55	3,2984	1,0238	0,0056	0,0001			
1,70	4,3222	1,0294	0,0057				
1,85	5,3516	1,0351					
2,00	6,3867						

Вычисление значения функции для аргумента X_1 с использованием интерполяционной формулы Ньютона

$X_1 = 1,168$. Воспользуемся первую интерполяционную формулу, т.к. точка находится слева от середины отрезка.

$$t = \frac{x-x_o}{h} = 0,4533$$

$$N_6 = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \frac{t(t-1)(t-2)}{3!}\Delta^3 y_0 + \frac{t(t-1)(t-2)(t-3)}{4!}\Delta^4 y_0 + \frac{t(t-1)(t-2)(t-3)(t-4)}{5!}\Delta^5 y_0 + \frac{t(t-1)(t-2)(t-3)(t-4)(t-5)}{6!}\Delta^6 y_0 \Rightarrow$$

$$Y_1 = 0,2234 + 0,46250 + 0,0008 + 0,0015 + 0,0022 + 0,0029 = 0,6933$$

Вычисление значения функции для аргумента X_2 с использованием интерполяционной формулы Гаусса

$X_2 = 1,463$, $a = 1,55$ (центр отрезка). Т.к. $X_2 < a$, воспользуемся Второй интерполяционной формулой Гаусса.

$$P_6 = y_0 + t\Delta y_{-1} + \frac{t(t+1)}{2!}\Delta^2 y_{-1} + \frac{t(t+1)(t-1)}{3!}\Delta^3 y_{-2} + \frac{t(t+1)(t-1)(t+2)}{4!}\Delta^4 y_{-2} + \frac{t(t+1)(t-1)(t+2)(t-2)}{5!}\Delta^5 y_{-3} + \frac{t(t+1)(t-1)(t+2)(t-2)(t+3)}{6!}\Delta^6 y_{-3} \Rightarrow$$

$$Y_2 = 3,2984 - 0,5997 + 0,0012 - 0,0015 + 0,0009 - 0,0009 + 0,0006 =$$
$$= 2,699$$

Программная реализация задачи

Листинг программы доступен по ссылке:

https://github.com/MakeCheerfulInstall/Computational-Math-2024/pull/40

Многочлен Лагранжа:

```
def lagrange(dots: Dots, X: float) -> float:
    n = dots.get_n()
    xs = dots.get_xs()
    ys = dots.get_xs()
```

```
    res = 0

    for i in range(n):
        y_i = ys[i]

        q = 1
        for j in range(n):
            if i != j:
                q *= (X - xs[j]) / (xs[i] - xs[j])

        res += y_i * q

    return rounded(res)
```

Многочлен Ньютона с разделенными разностями:

```
def get_factors(dots: Dots):
    n = dots.get_n()

    result = [
        [
            ''
            for _ in range(n)
        ]
        for _ in range(n)
    ]

    xs = dots.get_xs()
    ys = dots.get_ys()

    for i in range(n):
        result[i][0] = ys[i]

    for i in range(1, n):
        for j in range(0, n - i):
            result[j][i] = rounded((result[j + 1][i - 1] - result[j][i - 
1]) / (xs[j + i] - xs[j]))

    return result


def newton_div(dots: Dots, X: float) -> float:
    factors = get_factors(dots)
```

```python
    n = dots.get_n()

    xs = dots.get_xs()
    ys = dots.get_ys()
    result = ys[0]

    for i in range(1, n):
        q = factors[0][i]

        for j in range(0, i):
            q *= (X - xs[j])

        result += q

    return rounded(result)
```

Многочлен Ньютона с конечными разностями:

```python
def _check_equidistancy(dots: Dots) -> bool:
    n = dots.get_n()
    xs = dots.get_xs()

    if len(xs) < 2:
        return True

    h = rounded(xs[1] - xs[0])

    for i in range(1, n):
       if rounded(xs[i] - xs[i - 1]) != h:
           return False

    return True


def newton_equidistant(dots: Dots, X: float) -> float:
    n = dots.get_n()
    xs = dots.get_xs()

    if not _check_equidistancy(dots):
        raise ValueError('Точки не равноудалены')
```

```
h = xs[1] - xs[0]
t = (X - xs[0]) / h

finite_diff = get_finite_diffs(dots)

result = finite_diff[0][0]
factor = 1

for i in range(1, n):
    factor *= (t - (i - 1)) / i

    result += finite_diff[0][i] * factor

return rounded(result)
```