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

Факультет программной инженерии и компьютерной техники

Вычислительная математика
Лабораторная работа № 5
Вариант: 3

Выполнил:
Вильданов Ильнур Наилевич
Группа №P3212

Проверила:
Машина Екатерина Алексеевна

Санкт-Петербург
2025 г.

ЦЕЛЬ РАБОТЫ 3

ЗАДАНИЕ 3

ЛИСТИНГ ПРОГРАММЫ 4

ПРИМЕРЫ И РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ 7

ВЫВОД 8

Цель работы

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

Задание

Вычислительная реализация задачи:

x	y	X_1	X_2
1,10	0,2234	1,121	1,482
1,25	1,2438		
1,40	2,2644		
1,55	3,2984		
1,70	4,3222		
1,85	5,3516		
2,00	6,3867		

Построим таблицу конечных разностей:

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						

Вычислим значение функции для X_1.
Для этого воспользуемся первой интерполяционной формулой Ньютона, так как $X_1 = 1{,}121$ лежит в левой половине интервала.

$$t = \frac{(x - x_n)}{h} = \frac{1{,}121 - 1{,}10}{0{,}15} = 0{,}14$$

Интерполяционная формула:

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

$$y(1{,}121) = 0{,}2234 + 0{,}14 \cdot 1{,}0204 + \frac{0{,}14(0{,}14-1)}{2} \cdot 0{,}0002 + \frac{0{,}14(0{,}14-1)(0{,}14-2)}{6} \cdot 0{,}0132 + \frac{0{,}14(0{,}14-1)(0{,}14-2)(0{,}14-3)}{24} \cdot (-0{,}0368) + \frac{0{,}14(0{,}14-1)(0{,}14-2)(0{,}14-3)(0{,}14-4)}{120} \cdot 0{,}0762 + \frac{0{,}14(0{,}14-1)(0{,}14-2)(0{,}14-3)(0{,}14-4)(0{,}14-5)}{720} \cdot (-0{,}1313)$$

$$y(1{,}121) \approx 0{,}37148$$

Вычислим значение функции для X_2.
Для этого воспользуемся второй интерполяционной формулой Гаусса, так как $X_2 = 1{,}482$ лежит в левой половине интервала (1,482 < 1,550).

$$t = \frac{(x - x_0)}{h} = \frac{1{,}482 - 1{,}550}{0{,}15} = -0{,}453$$

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

$$y(1{,}482) = 3{,}2984 + (-0{,}453) \cdot 1{,}034 - \frac{0{,}453\big((-0{,}453)+1\big)}{2} \cdot (-0{,}0102) - \frac{0{,}453\big((-0{,}453)+1\big)\big((-0{,}453)-1\big)}{6} \cdot (-0{,}0236) - \frac{0{,}453\big((-0{,}453)+1\big)\big((-0{,}453)-1\big)\big((-0{,}453)+2\big)}{24} \cdot 0{,}0394 - \frac{0{,}453\big((-0{,}453)+1\big)\big((-0{,}453)-1\big)\big((-0{,}453)+2\big)\big((-0{,}453)-2\big)}{120} \cdot 0{,}0762 - \frac{0{,}453\big((-0{,}453)+1\big)\big((-0{,}453)-1\big)\big((-0{,}453)+2\big)\big((-0{,}453)-2\big)\big((-0{,}453)+3\big)}{720} \cdot (-0{,}1313)$$

$$y(1{,}482) \approx 2{,}83018$$

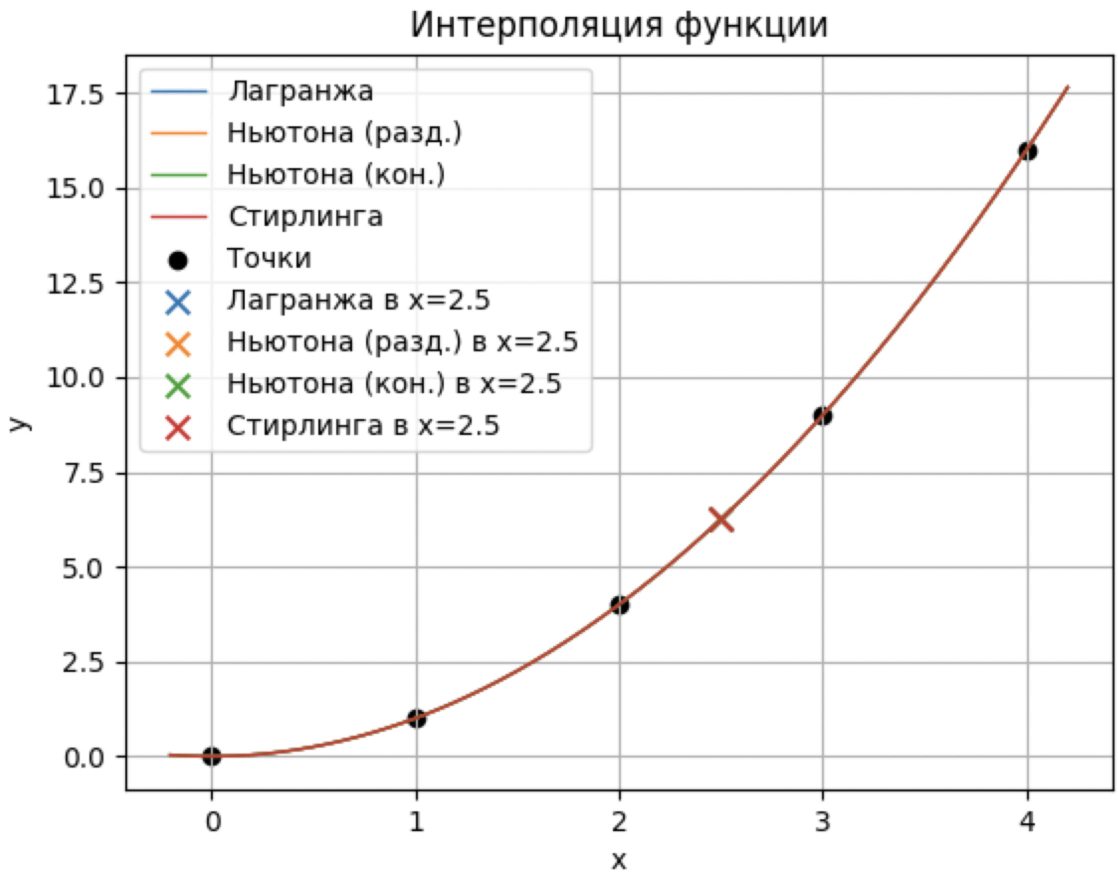
Листинг программы

```
def lagrange(xs, ys):
    n = len(xs)
    def P(x):
        total = 0
        for i in range(n):
            term = ys[i]
            for j in range(n):
                if i != j:
                    term *= (x - xs[j])/(xs[i] - xs[j])
            total += term
        return total
```

```python
    return P

def newton_divided(xs, ys):
    n = len(xs)
    coeffs = [ys[0]] * n
    def compute(a, b):
        if a == b:
            return ys[a]
        num = compute(a+1, b) - compute(a, b-1)
        den = xs[b] - xs[a]
        res = num/den
        if a == 0:
            coeffs[b] = res
        return res
    compute(0, n-1)
    def P(x):
        res = coeffs[0]
        prod = 1
        for k in range(1, n):
            prod *= (x - xs[k-1])
            res += coeffs[k] * prod
        return res
    return P

def is_equidistant(xs, tol=1e-8):
    h = xs[1] - xs[0]
    return all(abs((xs[i] - xs[i-1]) - h) < tol for i in range(2, len(xs)))

def newton_finite(xs, ys, deltas):
    def P(x):
        h = xs[1] - xs[0]
        mid = (xs[0] + xs[-1]) / 2
        if x <= mid:
            t = (x - xs[0]) / h
            res = deltas[0][0]
            for i in range(1, len(deltas)):
                term = deltas[i][0]
                for j in range(i):
                    term *= (t - j)
                term /= math.factorial(i)
                res += term
            return res
        else:
            t = (x - xs[-1]) / h
            res = deltas[0][-1]
            for i in range(1, len(deltas)):
                term = deltas[i][-1]
                for j in range(i):
                    term *= (t + j)
                term /= math.factorial(i)
                res += term
            return res
    return P

def stirling(xs, ys, deltas):
    n = len(xs)
    if not is_equidistant(xs) or n % 2 == 0:
        raise ValueError("Для многочлена Стирлинга нужны нечётные равномерные 
узлы")
    def P(x):
        zero = n // 2
        h = xs[1] - xs[0]
        t = (x - xs[zero]) / h
        res = ys[zero]
```

```python
        for i in range(1, zero+1):
            d1 = deltas[2*i-1][zero - i]
            d2 = deltas[2*i-1][zero - i + 1]
            num = d1 + d2
            term = t
            for j in range(1, i):
                term *= (t**2 - j**2)
            term *= num/(2 * math.factorial(2*i-1))
            res += term
            d  = deltas[2*i][zero - i]
            term2 = 1
            for j in range(i):
                term2 *= (t**2 - j**2)
            term2 *= d/math.factorial(2*i)
            res += term2
        return res
    return P

def bessel(xs, ys, deltas):
    n = len(xs)
    if not is_equidistant(xs) or n % 2 != 0:
        raise ValueError("Число узлов должно быть чётным и равномерным")
    def P(x):
        zero = n//2 - 1
        h = xs[1] - xs[0]
        t = (x - xs[zero]) / h
        res = 0
        for i in range(zero+1):
            d_even = deltas[2*i][zero - i]
            term_e = 1
            for j in range(-i, i):
                term_e *= (t + j)
            term_e *= (d_even)/(math.factorial(2*i)*2)
            res += term_e

            if 2*i+1 < len(deltas):
                d_odd = deltas[2*i+1][zero - i]
                term_o = (t - 0.5)
                for j in range(-i, i):
                    term_o *= (t + j)
                term_o *= d_odd/math.factorial(2*i+1)
                res += term_o
        return res
    return P
```

Примеры и результаты работы программы




```
Выберите способ задания функции: консоль — 1, файл — 2, функция
— 3
Ваш выбор: 1
Введите точки в формате `x y`. По окончании ввода введите `q`.
0   0
1   1
2   4
3   9
4  16
q
Введите точку интерполяции: 2.5

Таблица конечных разностей:
    0.0000     1.0000     2.0000     0.0000     0.0000
    1.0000     3.0000     2.0000     0.0000
    4.0000     5.0000     2.0000
    9.0000     7.0000
   16.0000

Интерполирующая функция Бесселя → ОШИБКА: Число узлов должно
быть чётным и равномерным

Интерполирующая функция: Интерполяционный многочлен Лагранжа
P(2.5)  = 6.25
==================================================
```

```
Интерполирующая функция: Интерполяционный многочлен Ньютона
(разд.)
P(2.5) = 6.25
==================================================
Интерполирующая функция: Интерполяционный многочлен Ньютона
(кон.)
P(2.5) = 6.25
==================================================
Интерполирующая функция: Интерполяционный многочлен Стирлинга
P(2.5) = 6.25
==================================================
```

Вывод

В ходе лабораторной работы я познакомился с методами интерполяции, вручную провел интерполяцию методами Ньютона (с конечными разностями) и Гаусса (модификацией метода Ньютона). Написал программу на языке Python, которая считает приближенные значения.